ปีการศึกษา 2530

ระบบวิทยุติดตามตัว

โดย

นายณรงค์พร เหล่าศรีสิน

นายธัชรวี หาริกุล

นายนิพนธ์ สุกุสวรรณวิทย์

อาจารย์ที่ปรึกษา

รศ.ดร. สิทธิชัย โภไคยอุดม

ปริญญานิพนธ์ปีการศึกษา 2530

ภาควิชา อิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบวิทยุติดตามตัว

ผู้จัดทำ

1. นายณรงค์พร เหล่าศรีสิน รหัสประจำตัว 271059
2. นายชัชรวี หาริกุล รหัสประจำตัว 271077
3. นายนิพนธ์ สุกสวรรณวิทย์ รหัสประจำตัว 271305

.....อาจารย์ที่ปรึกษา

(รศ.ดร. สิทธิชัย โภไคยอุดม)

ระบบวิทยุติดตามตัว

ณรงค์พร เหล่าศรีสิน
ชัชรวี หาริกุล
นิพนธ์ สุกสวรรณวิทย์

รศ.ดร.สิทธิชัย โภไคยอุดม
อาจารย์ที่ปรึกษา
ปีการศึกษา 2530

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เรียบเรียงขึ้นจากผลงานที่ได้สร้างขึ้นเป็นระบบวิทยุติดตามตัว (Nationwide Paging System) ซึ่งทำการส่งสัญญาณโดยการส่งคลื่นพาหะรอง (Subcarrier) ไปกับการกระจายเสียงของสถานีวิทยุที่ส่ง ในระบบมอดูเลทแบบขนาด (Amplitude Modulate) เนื่องจากสถานีวิทยุกระจายเสียงระบบนี้สามารถส่งสัญญาณครอบคลุมพื้นที่ได้กว้างทำให้สามารถใช้งานวิทยุติดตามตัวได้ในพื้นที่กว้างด้วย โดยที่ระบบนี้ประกอบด้วย

1. เครื่องรับข้อมูลจากคู่สายโทรศัพท์ ทำหน้าที่รับและแปลงสัญญาณ ดี.ที.เอ็ม.เอฟ. (Dual Tone Multi Frequency) เป็นสัญญาณดิจิตอลแบบขนาน สามารถรับข้อมูลได้จาก8คู่สายพร้อมกัน

2. ไมโครคอมพิวเตอร์ ทำหน้าที่บันทึกและจัดการข้อมูลและเวลาที่ส่ง

3. วงจรสร้างคลื่นพาหะรองและมอดูเลทข้อมูลกับคลื่นพาหะรองด้วยการมอดูเลทแบบความถี่ (Frequency Modulate)

4. วงจรรับสัญญาณและแสดงผล ทำหน้าที่รับสัญญาณ,ดีมอดูเลท(Demodulate), แปลงข้อมูลแบบอนุกรมเป็นแบบขนานและแสดงผล

NATIONWIDE PAGING SYSTEM

Narongporn Laosrisin
Tatrawee Harikul
Nipon Sugsavanvit

Assosiate Professor Sitthichai Poohaiyaudom Advisor

1987

Abstract

This thesis is about Nationwide Paging System that transmitt signal by mixing subcarrier with audio signal of Broadcast Amplitude Modulate radio station. So Amplitude Modulate radio station can broadcast over long distant that we can use this Paging system in wide area. This system consist of:

1. Data Reciever Circuit using for reciever D.T.M.F. (Dual Tone Multi Frequency) signal from telephone line and convert it to parallel data. It can recieve data from 8 telephone lines simultaneously.

2. Personal Computer using for record and rearrange data and time of tranamission.

3. Subcarrier Modulator Circuit using for convert parallel data from personal computer toserial data mad modulate with subcarrier by using frequency modulation.

4. Reciever and display circuit using for recieve signal , demodulate, convert data from serial to parallel and display.

สารบัญ หน้า

บทที่ 1 บทนำ

วิทยุติดตามตัว หรือ เพจเจอร์ (PAGER) จัดเป็นเครื่องมือสื่อสารชนิดเคลื่อนที่แบบหนึ่งที่เป็นที่นิยมในหมู่นักธุรกิจ ,นายแพทย์, เซลล์แมน, วิศวกร หรือช่างที่ให้บริการนอกสถานที่ และผู้ที่มีภาระหน้าที่ไม่ค่อยประจำที่ แต่จำเป็นต้องมีการติดต่อได้ตลอดเวลา เช่นเดียวกับพวกวิทยุโทรศัพท์, วิทยุโทรศัพท์แบบรวงผึ้ง แต่มีลักษระการสื่อสารในทิศทางเดียวจากผู้ส่งไปยังผู้รับ ข้อเด่นของเพจเจอร์คือ ขนาดเล็กกระทัดรัด สามารถพกติดตัวได้ตลอดเวลา

การติดต่อระหว่างผู้ส่งกับผู้รับอาจเป็นการแจ้งให้ผู้รับหาทางติดต่อกลับไปยังผู้ส่ง หรืออาจเป็นการส่งข่าวสารที่ต้องการไปยังผู้รับโดยตรงก็ได้ เพจเจอร์ที่มีใช้แบ่งได้เป็น 5 แบบ คือ

1.VOICE PAGER ส่งข่าวสารเป็นเสียงพูดไปยังผู้รับ

2.DIGITAL DISPLAY PAGER ส่งข่าวสารเป็นตัวเลขซึ่งตัวเลขนี้จะแทนข่าวสารที่ต้องการ อาจเป็นเบอร์โทรศัพท์ หรือรหัสที่มีความหมายเป็นที่เข้าใจกันระหว่างผู้ส่งกับผู้รับ

3.ALPHA NUMERLIC PAGER แบบนี้สามารถส่งเป็นตัวอักษรได้ด้วย ช่วยส่งข่าวสารได้ระเอียดขึ้น แต่จำเป็นต้องใช้ระบบที่พิเศษออกไป จึงยังไม่ค่อยแพร่หลาย

4.TONE-ALERT PAGER เป็นการส่งสัญญาณเสียงเพื่อเป็นการบอกให้ผู้รับติดต่อไปยังศูนย์ เพื่อรับข่าวสารอีกทีหนึ่ง

5.DUAL ADDRESS PAGER เหมือน TONE-ALERT PAGER เพียงแต่ให้สัญญาณเสียงได้สองลักษณะเพื่อให้ผู้รับทราบว่า จะติดต่อไปยังที่ใด

สำหรับในโครงงานนี้เป็นเพจเจอร์แบบ DIGITAL DISPLAY PAGER แต่ใช้การส่งสัญญาณด้วยคลื่นวิทยุ AM เพื่อให้ได้ขอบเขตการใช้งานที่กว้างไกลกว่าระบบทั่วๆไป

บทที่ 2 หลักการทำงานของวงจร

2.1 หลักการของวิทยุติดตามตัว

ระบบวิทยุติดตามตัวเป็นระบบการสื่อสารที่ใช้กันแพร่หลายในปัจจุบัน ระบบนี้เป็นการสื่อสารทางเดียวคือสามารถส่งสัญญาณจากสถานีส่งไปยังผู้ใช้เพียงทางเดียว ข้อมูลที่ส่งนี้จะเป็นข้อมูลสั้นๆ เช่นหมายเลขโทรศัพท์เพื่อให้ผู้ใช้ติดต่อกลับมาได้ อุปกรณ์ที่จำเป็นสำหรับระบบนี้ได้แก่

1. อุปกรณ์รับข้อมูลจากผู้ต้องการติดต่อ เช่น เครื่องรับข้อมูลทางโทรศัพท์ที่ทำงานโดยอัตโนมัติ หรือแป้นรับข้อมูลจากโอเปอเรเตอร์

2. อุปกรณ์ส่งข้อมูล เช่น วงจรเข้ารหัส (Encoding Circuit), มอดูเลเตอร์ (Modulater) และเครื่องส่ง (Transmitter)

3. อุปกรณ์บันทึกการใช้งาน ใช้บันทึกข้อมูลที่ส่งและเวลาขณะทำการส่งเพื่อให้สามารถตรวจสอบได้ภายหลัง ส่วนนี้มักจะเป็นเครื่องคอมพิวเตอร์

4. อุปกรณ์รับสัญญาณและแสดงข้อมูล ทำหน้าที่รับสัญญาณ, ดีมอดูเลท (Demodulate) และแสดงข้อมูลให้ผู้ใช้เห็น

สำหรับระบบวิทยุติดตามตัวที่สร้างขึ้นนี้ทำการส่งสัญญาณโดยการส่งคลื่นพาหะรอง (Subcarrier) ไปพร้อมกับสัญญาณกระจายเสียงของสถานีวิทยุในระบบ เอ.เอ็ม. ทั้งนี้ก็เนื่องจากสถานีวิทยุในระบบนี้กำลังส่งสูงสามารถส่งสัญญาณได้ในระยะไกลทำให้ขอบเขตการใช้งานของระบบวิทยุติดตามตัวกว้างตามไปด้วย

2.2 หลักการวงจรรับข้อมูลทางโทรศัพท์

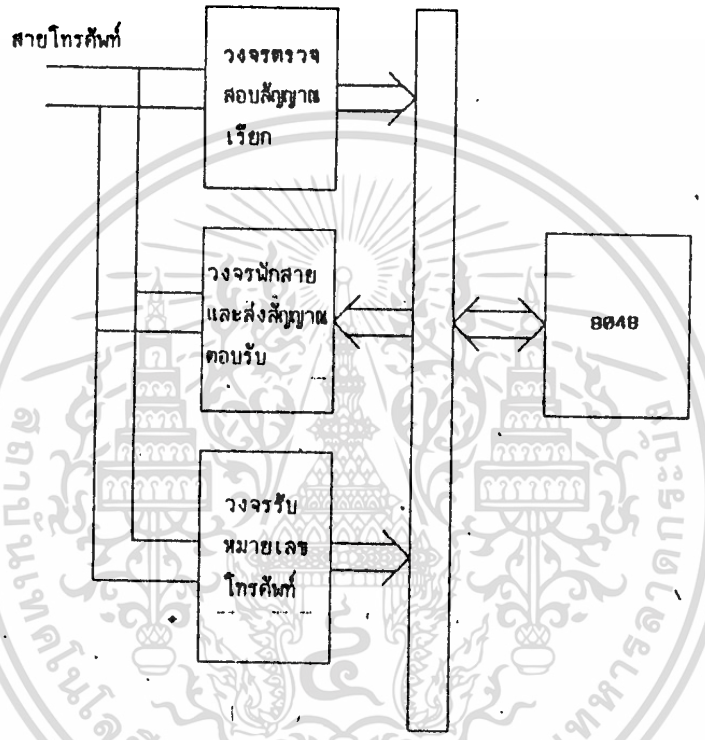
วงจรรับข้อมูลทางโทรศัพท์ จะประกอบด้วยส่วนต่างๆดังรูปที่ 2.1

2.2.1 วงจรตรวจสอบสัญญาณเรียก ซึ่งจะเป็นวงจรที่ตรวจรับว่ามีสัญญาณโทรศัพท์เรียกเข้ามาหรือไม่ ถ้ามีแล้วจะส่งสัญญาณทางดิจิตอลไปบอกไมโครโปรเซสเซอร์ (Micro Processcer) ให้ 8048 รู้ว่ามีการติดต่อเข้ามา

2.2.2 วงจรพักสายและส่งสัญญาณตอบรับ จะเป็นวงจรที่ใช้สำหรับพักสายซึ่งก็คือการรับโทรศัพท์นั่นเอง แล้วก็จะส่งสัญญาณตอบรับผ่านสายโทรศัพท์ไปยังผู้ติดต่อเพื่อเป็นการบอกให้ส่งข้อมูลหมายเลขวิทยุและหมายเลขโทรศัพท์เข้ามาได้ ซึ่งการพักสายและส่งสัญญาณตอบรับนี้จะถูกควบคุมโดย 8048

2.2.3 วงจรรับข้อมูลทางโทรศัพท์ วงจรนี้จะเป็นวงจรที่รับสัญญาณดีทีเอ็ม

เอฟ (Dual Tone Multi Frequency) ซึ่งส่งมาจากการกดคีย์ที่แป้นโทรศัพท์ของผู้ติดต่อ ซึ่งเป็นสัญญาณ 2 ความถี่ ตามหมายเลขที่กดแล้ววงจรนี้จะแปลงให้เป็นสัญญาณดิจิตอล 4 บิท ตามหมายเลขที่ส่งมา เพื่อส่งให้ 8048 รับข้อมูลหมายเลขวิทยุและหมายเลขโทรศัพท์ที่ส่งเข้ามา



รูปที่ 2.1 แสดงส่วนประกอบของวงจรรับข้อมูลทางโทรศัพท์

2.3 หลักการของวงจรควบคุมการรับข้อมูลจากโทรศัพท์

ในการออกแบบศูนย์รับโทรศัพท์ของระบบวิทยุติดตามตัวนี้ ต้องการแบ่งเบาภาระหน้าที่ให้กับโอเปอร์เรเตอร์ โดยเราใช้วงจรอิเลคทรอนิคส์มาทำหน้าที่รับโทรศัพท์และรับข้อมูลต่างๆ แทนโอเปอร์เรเตอร์ การเพิ่มขีดความสามารถ ให้สามารถทำงานกับโทรศัพท์ ได้หลายๆคู่สาย จึงเป็นเรื่องที่น่าสนใจเรื่องหนึ่ง

เนื่องจากงานควบคุมการทำงานของวงจรส่วนหน้าหลายๆชุด เป็นงานที่ยุ่งยากซับซ้อน จึงเลือกใช้ 8048 มาทำหน้าที่ควบคุมนี้ เนื่องจากสาเหตุที่ว่า 8048 มีส่วนประกอบที่จำเป็นหลายๆอย่าง รวมอยู่ภายในตัวมันเองแล้ว ไม่ว่าจะเป็น พอร์ทขนาด 8 บิทจำนวน 3 พอร์ท , ขาอินพุททดสอบ 3 ขา , ไทม์เมอร์/เคาน์เตอร์ขนาด 8 บิท , วง

จรออสซิเลเตอร์ ,ระบบอินเตอร์รัพท์ เป็นต้น ทำให้สามารถลดจำนวนของอุปกรณ์ใช้งานลงไปได้มาก และ มีความคล่องตัวในการใช้งานมาก

หลักการทำงานของวงจรควบคุมนี้จะประกอบด้วยสองส่วนใหญ่ๆ คือ หลักการทางด้านวงจร กับหลักการทำงานด้านโปรแกรม ซึ่งทั้งสองส่วนนี้จะต้องมีความสัมพันธ์กันเสมอ

2.3.1 หลักการทำงานของวงจรควบคุม

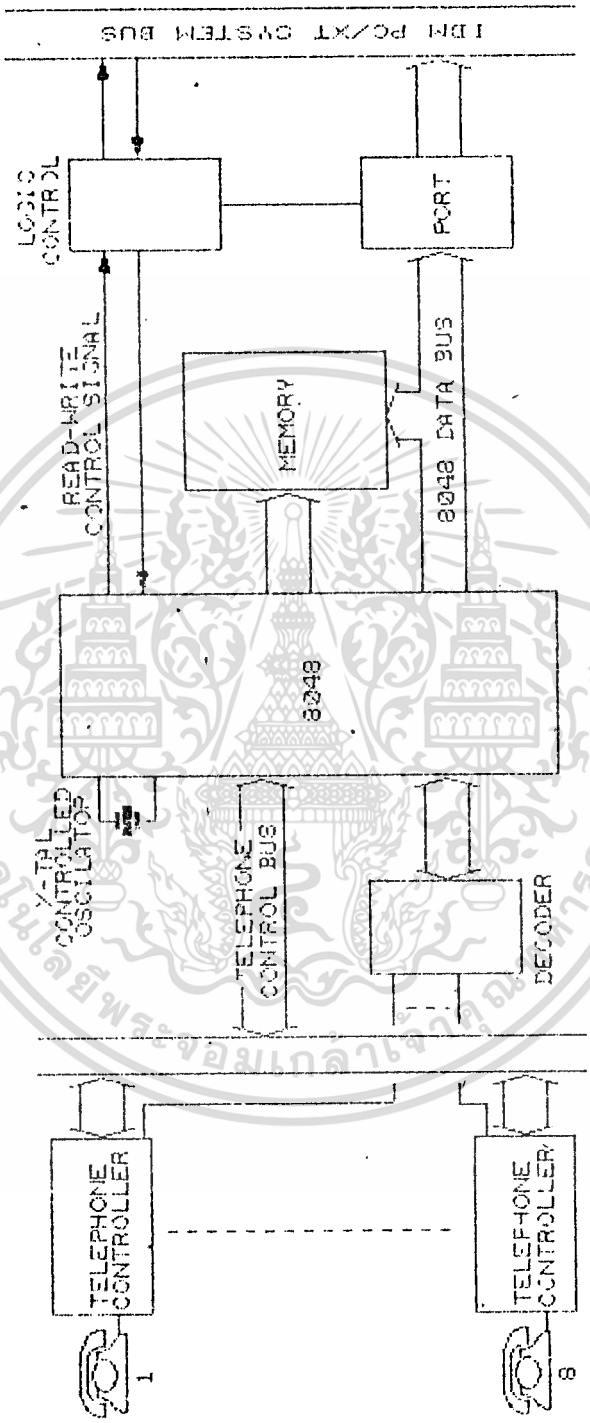
วงจรควบคุมจะประกอบด้วยส่วนต่างๆดังรูปที่ 2.2

เพื่อให้เกิดความสดวกในการต่อเพิ่มจำนวนวงจรส่วนหน้า ให้สามารถใช้งานกับจำนวนคู่สายโทรศัพท์มากขึ้น จึงออกแบบวงจรควบคุมให้สามารถติดต่อกับวงจรส่วนหน้าได้โดยใช้ระบบบัส เช่นเดียวกับระบบคอมพิวเตอร์ทั่วไป กล่าวคือ ขาสัญญาณต่างๆของวงจรส่วนหน้า ไม่ว่าจะเป็นอินพุทหรือเอาท์พุทจะต่อขนานกันหมด เมื่อ 8048 ต้องการติดต่อกับวงจรคู่สายใด ก็จะส่งสัญญาณไปยังวงจรเลือก แล้ววงจรเลือกจึงจะส่งสัญญาณสโตรบ (STROBE) ไปยังวงจรส่วนหน้า เพื่อบอกให้รู้ว่า 8048 ต้องการติดต่อกับวงจรคู่สายนั้น ส่วนวงจรคู่สายอื่นๆจะไม่สนใจเลย

สำหรับส่วนที่ต่อเป็นระบบไมโครคอมพิวเตอร์ 8048 นั้นจะประกอบด้วย ส่วนของหน่วยความจำโปรแกรมสำหรับเก็บโปรแกรมควบคุมการทำงาน และ หน่วยความจำข้อมูลสำหรับเก็บข้อมูลที่ได้รับทางโทรศัพท์ และ ตัวแลทช์ซึ่งทำหน้าที่แยกสัญญาณแอดเดรสที่มัลติเพล็กซ์มากับสัญญาณดาต้าออกจากดาต้าบัส

วงจรอีกส่วนหนึ่งทำหน้าที่เป็นบัฟเฟอร์เก็บข้อมูลชั่วคราว โดยข้อมูลที่ 8048 ได้รับจากโทรศัพท์จะถูกส่งไปยังเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ผ่านวงจรส่วนนี้

เพื่อให้การรับส่งข้อมูลมีความถูกต้องมากขึ้น จึงเพิ่มเติมส่วนของวงจรควบคุมการรับส่งข้อมูลขึ้นมา หน้าที่ของวงจรควบคุมนี้คือ ส่งสัญญาณบอกให้ เครื่องไมโครคอมพิวเตอร์ IBM PC/XT รู้เมื่อ 8048 ส่งข้อมูลมารอแล้ว และ ส่งสัญญาณบอกให้ 8048 ส่งข้อมูลไบท์ใหม่ เมื่อเครื่องไมโครคอมพิวเตอร์ IBM PC/XT อ่านข้อมูลไบท์เดิมไปแล้ว ซึ่งเรียกการติดต่อในลักษณะนี้ว่า เป็นการติดต่อแบบ HANDSHAKING



รูปที่ 2.2 แสดงส่วนประกอบของวงจรควบคุมการรับข้อมูลจากโทรศัพท์

2.3.2 หลักการทำงานของโปรแกรม

หน้าที่การทำงานอย่างคร่าวๆ ของระบบไมโครคอมพิวเตอร์ 8048 นี้ ได้แก่ การตรวจสอบสถานะต่างๆของคู่สาย และ การรับข้อมูลทางโทรศัพท์ โดยอาศัยการแสกนทีละคู่สายอย่างรวดเร็ว

หน้าที่การตรวจสอบสถานะของคู่สายนั้น ได้แก่ การตรวจสอบสัญญาณกระดิ่ง เพื่อดูว่ามีใครติดต่อเข้ามาหรือไม่ ถ้าหากมีผู้โทรศัพท์เข้ามา ก็จะรอให้สัญญาณกระดิ่งหมดลงเสียก่อน จึงจะรับสาย แล้วส่งสัญญาณตอบรับกลับไป เป็นการบอกให้ทราบว่า ขณะนี้เครื่องไมโครคอมพิวเตอร์พร้อมจะรับข้อมูลแล้ว เมื่อผู้ที่ติดต่อมา ก็จะป้อนข้อมูลผ่านทางคีย์โทรศัพท์ ได้เป็นสัญญาณ DTMF ออกมา แล้ว 8048 ก็จะอ่านข้อมูลเข้ามาเก็บไว้ในหน่วยความจำข้อมูล จนกระทั่งได้รับข้อมูลครบแล้ว ก็จะวางสายโทรศัพท์

ข้อมูลที่ส่งเข้ามานั้น จะถูกเก็บไว้ในหน่วยความจำ รอจนเมื่อเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ว่างจากงานอื่น ก็จะมาอ่านข้อมูลจาก 8048 ไปใช้งาน คือ ส่งข้อมูลดังกล่าวไปยังเครื่องส่งสัญญาณวิทยุ และบันทึกข้อมูลลงในระบบข้อมูล

การที่ 8048 จะสามารถติดต่อกับโทรศัพท์ได้หลายๆคู่สายนั้น จะต้องกระทำในลักษณะสแกนทีละคู่สายอย่างรวดเร็ว ซึ่งการสแกนนี้จะทำให้เหมือนกับว่า ทำงานหลายๆงานพร้อมๆกัน แต่ถ้ามองในแง่ของการเขียนโปรแกรมควบคุมนั้น จะพบว่าเป็นการทำงานที่ไม่ต่อเนื่อง กล่าวคือ ในขณะที่กำลังควบคุมโทรศัพท์เครื่องหนึ่งอยู่ ก็จะต้องเปลี่ยนไปควบคุมเครื่องอื่นๆต่อไป จนกระทั่งวนมาถึงเครื่องเดิม เพื่อให้การควบคุมสามารถดำเนินได้อย่างต่อเนื่อง จึงต้องแบ่งงานออกเป็นหลายขั้นตอน และมีการเก็บข้อมูลบางอย่างที่บอกให้รู้ว่า ในรอบที่แล้วมาเราควบคุมโทรศัพท์เครื่องนี้ถึงขั้นตอนใดแล้ว จะได้ควบคุมต่อไปได้ถูกต้อง ข้อมูลดังกล่าวนี้ขอเรียกว่าเป็น STATUS ของแต่ละคู่สาย

ข้อมูลใน STATUS จะบอกให้รู้ขั้นตอนที่จะใช้ควบคุมโทรศัพท์ เช่น ให้รับสายโทรศัพท์ หรือส่งสัญญาณตอบรับ หรือวางสายโทรศัพท์ หรือส่งข้อมูลไปให้เครื่อง IBM เป็นต้น

การควบคุมโทรศัพท์แบ่งออกเป็นขั้นตอนต่างๆ ดังนี้

1.ตรวจสอบดูว่ามีสัญญาณกระดิ่งเรียกเข้ามาหรือไม่ ถ้ายังไม่มีสัญญาณกระดิ่งก็จะผ่านไปตรวจสอบคู่สายอื่นๆต่อไป แต่ถ้าหากมีสัญญาณกระดิ่งเข้ามา 8048 ก็ยังสั่งให้รับสายนั้นไม่ได้ เนื่องจากสัญญาณกระดิ่ง มีแรงดันสูง อาจทำให้วงจรส่วนหน้าเสียหายได้

2. ตรวจสอบดูว่าเมื่อสัญญาณกระดิ่งเงียบไปแล้ว ให้รับสายโทรศัพท์

3. เมื่อรับสายโทรศัพท์แล้ว ก็จะส่งสัญญาณตอบรับกลับไป

4. เมื่อสัญญาณตอบรับจบแล้ว เครื่องก็พร้อมจะรับข้อมูลจากผู้ใช้โทรศัพท์ผ่านทางคีย์โทรศัพท์ โดยมีวงจรตรวจสอบสัญญาณ DTMF ทำหน้าที่แปลงสัญญาณ DTMF เป็นรหัสไบนารี่

ปัญหาอย่างหนึ่งที่อาจเกิดขึ้นได้ คือ การวางสายโทรศัพท์ไป ก่อนที่จะป้อนข้อมูลให้คอมพิวเตอร์จนครบ ซึ่ง 8048 ไม่มีทางรู้ได้เลยว่า ผู้ใช้โทรศัพท์วางสายไปก่อนแล้ว ก็คงจะรอสัญญาณ DTMF อยู่เรื่อยๆ ไม่ยอมวางสายนั้น การแก้ปัญหาข้อนี้ ทำได้สองทาง คือ

การเพิ่มวงจรภายนอก เพื่อตรวจจับการวางสายโทรศัพท์ กับอีกวิธีหนึ่งคือ การให้ 8048 หน่วงเวลารอประมาณ 10 วินาที ถ้าหากในเวลา 10 วินาทีนี้ ไม่มีสัญญาณ DTMF เข้ามา 8048 ก็จะสั่งให้วางสายนี้ วิธีการหน่วงเวลานี้ ทำให้ประหยัดอุปกรณ์ที่ต่อภายนอก แต่สามารถใช้ได้ผลดี สำหรับเวลา 10 วินาทีที่หน่วงเอาไว้นี้ ก็นับว่ามากพอสำหรับเตรียมกดคีย์ต่อไป

5. เมื่อ 8048 ได้รับข้อมูลครบแล้ว ก็จะสั่งให้วางสายโทรศัพท์คู่นั้น

6. ส่งสัญญาณออกมาเพื่อแสดงว่า ตอนนี้มีข้อมูลครบแล้ว สามารถให้เครื่องไมโครคอมพิวเตอร์ IBM PC/XT มาอ่านข้อมูลไปได้ ถ้าหากเครื่อง IBM ยังไม่พร้อมจะอ่านข้อมูล 8048 ก็จะใช้งานโทรศัพท์สายนี้ไม่ได้ เพราะจะทำให้มีการส่งข้อมูลใหม่มาทับข้อมูลเก่า แต่เมื่อเครื่อง IBM อ่านข้อมูลไปแล้ว ก็สามารถใช้งานคู่สายนั้นได้อีกครั้ง

การส่งข้อมูลไปให้แก่เครื่องไมโครคอมพิวเตอร์ IBM PC/XT นั้นจะต้องดูว่าข้อมูลถูกอ่านไปหรือยัง ถ้ายังไม่มีการอ่านข้อมูลก็จะยังส่งข้อมูลใหม่มาไม่ได้ ต้องรอจนกว่าเครื่องไมโครคอมพิวเตอร์ IBM PC/XT จะอ่านข้อมูลไปแล้ว จึงจะส่งข้อมูลไบท์ต่อไปมาได้

เมื่อ 8048 ส่งข้อมูลครบทั้งหมดแล้ว ก็จะต้องส่งรหัสหยุดไปให้เครื่องไมโครคอมพิวเตอร์ IBM PC/XT ด้วยเพื่อให้เครื่องไมโครคอมพิวเตอร์ IBM PC/XT หยุดการติดต่อ แล้วกลับไปทำงานหลักของมันต่อไป

2.4 หลักการโปรแกรมบนเครื่องไมโครคอมพิวเตอร์

โปรแกรมควบคุมการทำงานนี้จะต้องควบคุมให้ไมโครคอมพิวเตอร์สามารถรับข้อมูลจากโอเปอเรเตอร์ผ่านทางคีย์บอร์ด, จัดเก็บข้อมูล ,อ่านข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์เข้ามาในช่วงเวลาที่เหมาะสม และส่งข้อมูลไปยังวงจรสร้างคลื่นพาหะรองซึ่ง

การรับและส่งข้อมูลนี้จะต้องมีการติดต่อกับไอโอพอร์ท (I/O Port) โดยตรงตามที่กล่าวไว้ในหัวข้อ 2.2 ดังนั้นภาษาที่เลือกใช้ควรจะเป็นภาษาโครงสร้างเพื่อให้การเขียนโปรแกรมขนาดใหญ่ทำได้สะดวกและจะต้องสามารถติดต่อกับ หน่วยความจำ, พอร์ท และ รีจิสเตอร์ และเขียนโปรแกรมจัดการฐานข้อมูลได้อีกด้วย ทำให้การบันทึกข้อมูลการใช้งานสะดวกและแน่นอนขึ้น

ดังที่ได้กล่าวมาแล้วว่าไมโครคอมพิวเตอร์จะต้องอ่านข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์เป็นระยะๆด้วยช่วงเวลาที่เหมาะสม สลับไปกับการรับข้อมูลจากคีย์บอร์ด และการค้นหาหรือบันทึกข้อมูล ช่วงเวลาระหว่างการอ่านการอ่านข้อมูลแต่ละครั้งนั้นจะต้องไม่นานเกินไปจนข้อมูลมีจำนวนมากซึ่งอาจจะทำให้หน่วยความจำของเครื่องรับข้อมูลทางโทรศัพท์ไม่เพียงพอ และจะต้องไม่เร็วเกินไปจนบางครั้งไม่มีข้อมูลชุดใดครบถ้วนพร้อมที่จะอ่านได้ เราสามารถแบ่งโปรแกรมเป็นส่วนย่อยๆได้ดังนี้

1. โปรแกรมหลัก (Main Program) ทำหน้าที่เรียกโปรแกรมย่อยอื่นๆ ผังงาน (flow chart) ของโปรแกรมนี้แสดงในรูปที่ 2.3

2. โปรแกรมการรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์ เมื่อถูกเรียกจากโปรแกรมหลักโปรแกรมนี้จะอ่านข้อมูลจากพอร์ทที่แสดงสถานะของเครื่องรับข้อมูลทางโทรศัพท์ และตรวจสอบบิทที่ 7 ของข้อมูล ซึ่งจะแสดงความพร้อมที่จะส่งข้อมูลและจะทำการอ่านซ้ำจนกว่าบิทที่ 7 จะมีค่าเป็นหนึ่งแสดงว่าในขณะนั้น เครื่องรับข้อมูลทางโทรศัพท์พร้อมที่จะทำการส่งข้อมูลแล้ว ต่อไปโปรแกรมจะเริ่มอ่านพอร์ทข้อมูล การอ่านข้อมูลครั้งแรกจะเป็นการอ่านทิ้งเพื่อเคลียร์วงจรรับส่งข้อมูล จากนั้นจะอ่านข้อมูลจริงมาเก็บไว้แบบอะเรย์ (Array) ข้อมูลที่เข้ามาก่อนจะเป็นหมายเลขผู้ใช้เข้ามาทีละหลัก จะถูกเก็บไว้ในอะเรย์ของหมายเลขผู้ใช้ ขณะที่รับข้อมูลจะต้องตรวจสอบตัวคั่นที่กำหนดค่าให้พิเศษไปจากตัวเลขธรรมดา ตัวคั่นนี้จะคั่นระหว่างหมายเลขผู้ใช้กับหมายเลขเรียกกลับเมื่อพบตัวคั่นก็จะเก็บข้อมูลที่ตามมาไว้ในอะเรย์ของหมายเลขเรียกกลับ และในขณะที่อ่านข้อมูลจะต้องตรวจสอบตัวปิดท้ายที่กำหนดค่าให้เป็นค่าพิเศษอีกเช่นกัน ตัวปิดท้ายนี้จะเป็นตัวคั่นระหว่างข้อมูลแต่ละชุด เมื่อพบตัวปิดท้ายก็จะตรวจสอบตัวปิดท้ายการส่งที่กำหนดค่าเป็น FFH ถ้าไม่ใช่ก็จะบันทึกเป็นข้อมูลในชุดต่อไป เมื่อพบตัวปิดท้ายการส่งก็จะหยุดการติดต่อกับเครื่องรับข้อมูลทางโทรศัพท์ ต่อไปจะทำแปลงข้อมูลหมายเลขหลายๆหลักที่ส่งมาหนึ่งไบท์ต่อหนึ่งหลักเรียงกันมาและถูกเก็บไว้ในอะเรย์ ให้เป็นค่าเลขจำนวนจริง (แทนได้ด้วยเลขไบนารี่สองไบท์) เก็บลงในอะ

เรย์ของข้อมูลเอ๊าท์พุทเพื่อทำการส่งไปยังวงจรสร้างคลื่นพาหะ เมื่อส่งแล้วจะส่งข้อมูลให้โปรแกรมบันทึกข้อมูลทีละชุดเพื่อทำการบันทึกลงในไฟล์บันทึกการใช้งาน (บันทึกพร้อมกับวันที่และเวลา) ผังงานของโปรแกรมนี้แสดงอยู่ในรูปที่ 2.4

3. โปรแกรมการทำงานในส่วนของพนักงานรับโทรศัพท์ การทำงานของพนักงานรับโทรศัพท์มีการทำงานหลักเกี่ยวกับการส่งสัญญาณเรียกผู้ใช้ผ่านทางคีย์บอร์ด, การตรวจสอบบันทึกข้อมูลการใช้งาน, จัดการเกี่ยวกับข้อมูลของผู้ใช้แต่ละคนจากการทำงานหลักที่กล่าวมานี้ จะต้องใช้ไฟล์ข้อมูล 2 ไฟล์คือ

1) ไฟล์เก็บข้อมูลการใช้งาน จะเก็บข้อมูลคือ วันที่, เวลาที่ทำการส่ง, หมายเลขผู้ใช้, หมายเลขโทรศัพท์เรียกกลับ

2) ไฟล์เก็บข้อมูลเกี่ยวกับผู้ใช้ จะเก็บข้อมูลคือ หมายเลขผู้ใช้, ชื่อ, นามสกุล, ที่อยู่ และหมายเลขโทรศัพท์

รายการหลักสำหรับพนักงานรับโทรศัพท์ซึ่งจะสามารถเพิ่มเติมให้สมบูรณ์ภายหลังมีดังนี้

1) การส่งสัญญาณเรียกผู้ใช้โดยการป้อนชื่อ การทำงานในรายการนี้เริ่มด้วยการรับชื่อผู้ใช้ที่ต้องการเรียก ค้นหาข้อมูลในไฟล์เกี่ยวกับผู้ใช้ นำข้อมูลที่มีชื่อตรงกับชื่อที่ต้องการทั้งหมดมาแสดง บนจอภาพ จากนั้นโอเปอเรเตอร์ก็จะสามารถเลือกข้อมูลที่มี ชื่อ, นามสกุล และหมายเลข ตรงกับที่ต้องการโดยการเลื่อนตัวเลือก ไปบนจอภาพและเมื่อกดเลือกหมายเลขผู้ใช้ที่เลือกก็จะถูกส่งผ่านไอโอพอร์ทไปยังเครื่องส่งวิทยุ จากนั้นก็จะบันทึกเวลา, วันที่ และข้อมูลการส่งลงในไฟล์บันทึกข้อมูลการใช้งาน

2) รายงานการใช้โดยการป้อนชื่อ ข้อมูลในการส่งทั้งหมดที่ถูกบันทึกไว้จะต้องสามารถรายงานให้ผู้ใช้ทราบถึงการส่งสัญญาณเรียกทั้งหมดได้ การทำงานในรายการนี้จะเป็นการทำงานผ่านโปรแกรมจัดการฐานข้อมูลโดยเริ่มจากการรับชื่อจากคีย์บอร์ด ค้นหาหมายเลขผู้ใช้จากไฟล์ข้อมูลผู้ใช้ จากนั้นนำข้อมูลจากไฟล์บันทึกการใช้งานที่มีหมายเลขผู้ใช้ตรงตามที่ต้องการมาแสดงบนจอภาพ

3) การส่งสัญญาณเรียกผู้ใช้โดยป้อนหมายเลข เริ่มต้นโดยการรับหมายเลขที่ต้องการเรียกจากคีย์บอร์ด ส่งข้อมูลผ่านพอร์ทไปยังเครื่องส่งสัญญาณ เมื่อการส่งเรียบร้อยก็จะเรียกโปรแกรมจัดการข้อมูลเพื่อทำการเก็บข้อมูลและเวลาลงในไฟล์บันทึกการใช้งาน

4) การรายงานการใช้โดยการป้อนหมายเลข เริ่มต้นโดยการรับหมายเลขที่ต้อง

การตรวจสอบการใช้งานจากคีย์บอร์ด จากนั้นเรียกโปรแกรมจัดการฐานข้อมูลเพื่อทำการค้นหาข้อมูลในไฟล์บันทึกการใช้งานที่มีหมายเลขผู้ใช้ตามที่ต้องการและแสดงข้อมูลทั้งหมดทางจอภาพ

5) การเพิ่มเติมผู้ใช้ การทำงานในรายการนี้จะต้องใช้โปรแกรมที่เรียกว่าฟิล์ดอีดิท(field edit)ซึ่งจะทำหน้าที่รับข้อมูลผู้ใช้รายใหม่แต่ขณะที่ป้อนข้อมูลจะสามารถแก้ไขได้เมื่อรับข้อมูลเรียบร้อยจะทำการเรียกโปรแกรมจัดการฐานข้อมูลเพื่อที่จะเก็บข้อมูลที่มีลักษณะเป็นโครงสร้างลงไฟล์

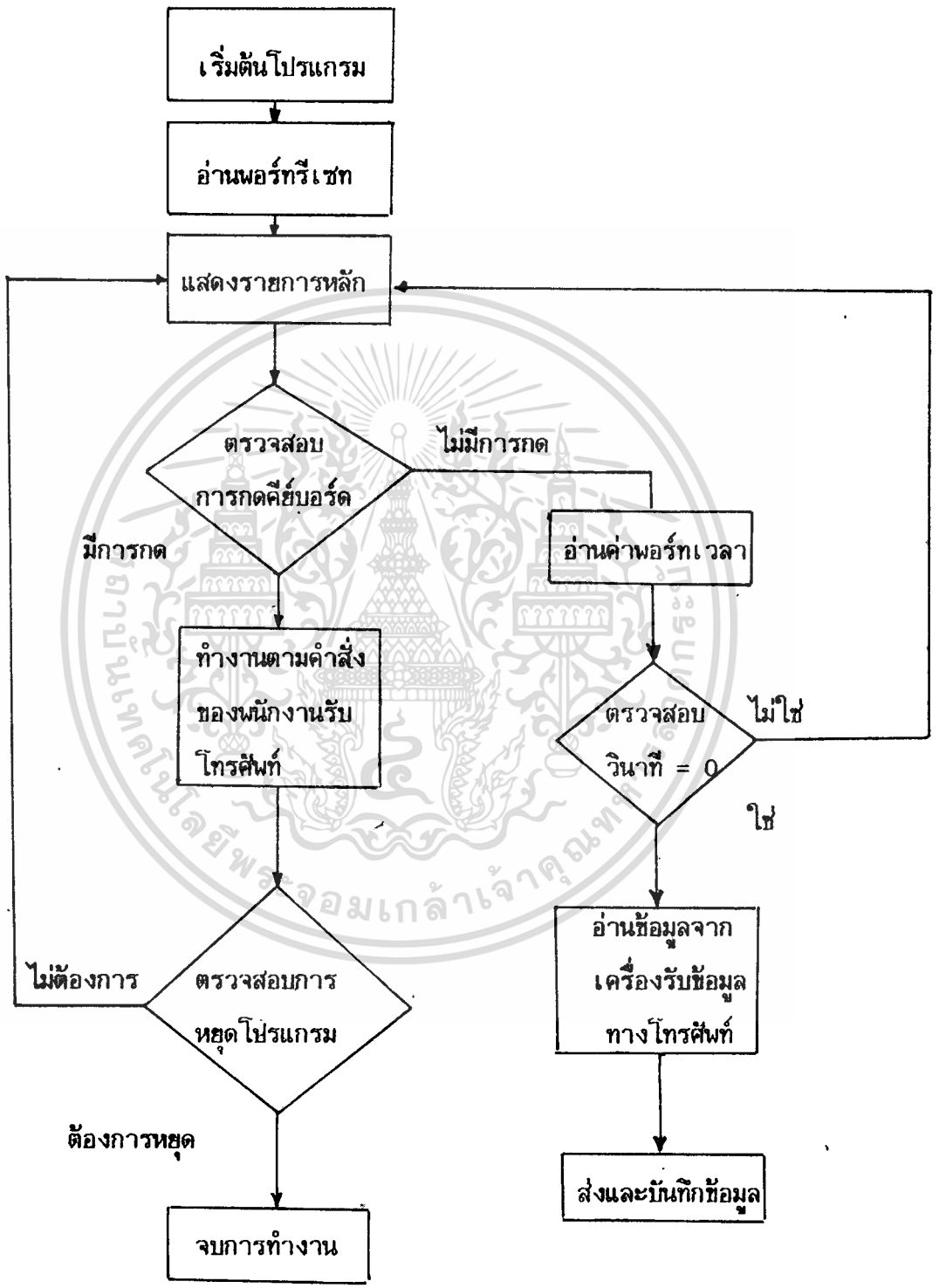
6) การลบข้อมูลของผู้ใช้ รายการนี้จะใช้งานเมื่อมีผู้ใช้เลิกเช่าเครื่อง โดยเริ่มจากการรับ ชื่อผู้ใช้ที่ต้องการยกเลิก จากนั้นเรียกโปรแกรมจัดการฐานข้อมูลเพื่อทำการลบข้อมูลจากไฟล์เกี่ยวกับผู้ใช้ตามชื่อที่ต้องการ

7) การตั้งนาฬิกา เนื่องจากมีการบันทึกเวลาการส่งสัญญาณดังนั้นจึงจำเป็นที่จะต้องเวลาของนาฬิกาให้ถูกต้อง จะกล่าวถึงโปรแกรมนี้อีกครั้งในหัวข้อ3.2.2

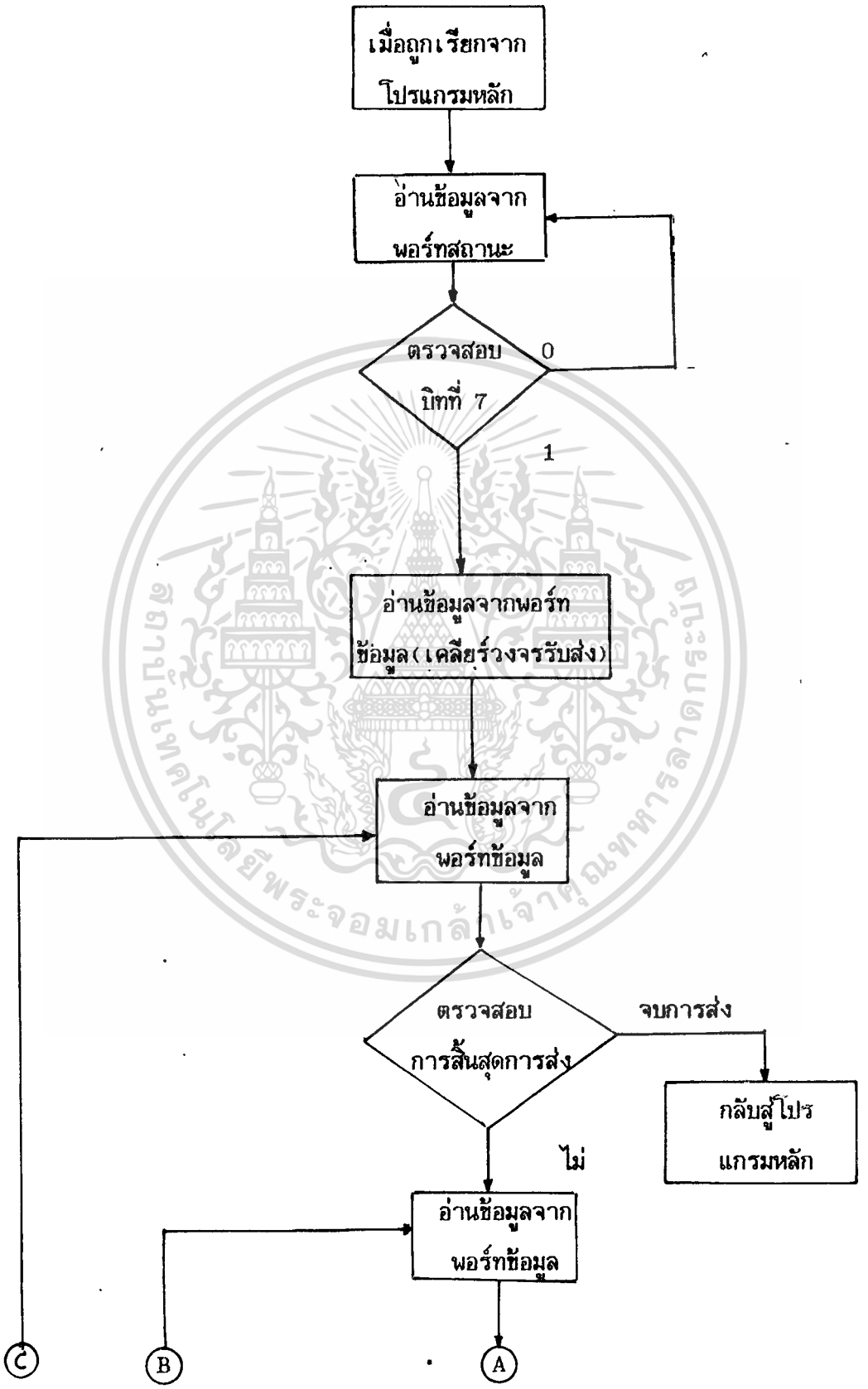
8) การจบโปรแกรม รายการนี้จะทำงานเมื่อโอเปอเรเตอร์ต้องการจบการทำงานทั้งหมด โดยจะทำหน้าที่ปิดไฟล์ข้อมูลทั้งสองไฟล์และกลับสู่การทำงานของดอส

4. โปรแกรมจัดการและบันทึกข้อมูล ทำหน้าที่หลักคือการบันทึกข้อมูลและวันเวลาของการส่งลงในหน่วยความจำถาวร อาจเพิ่มเติมโปรแกรมจัดการฐานข้อมูลเข้าไว้ด้วยภายหลัง

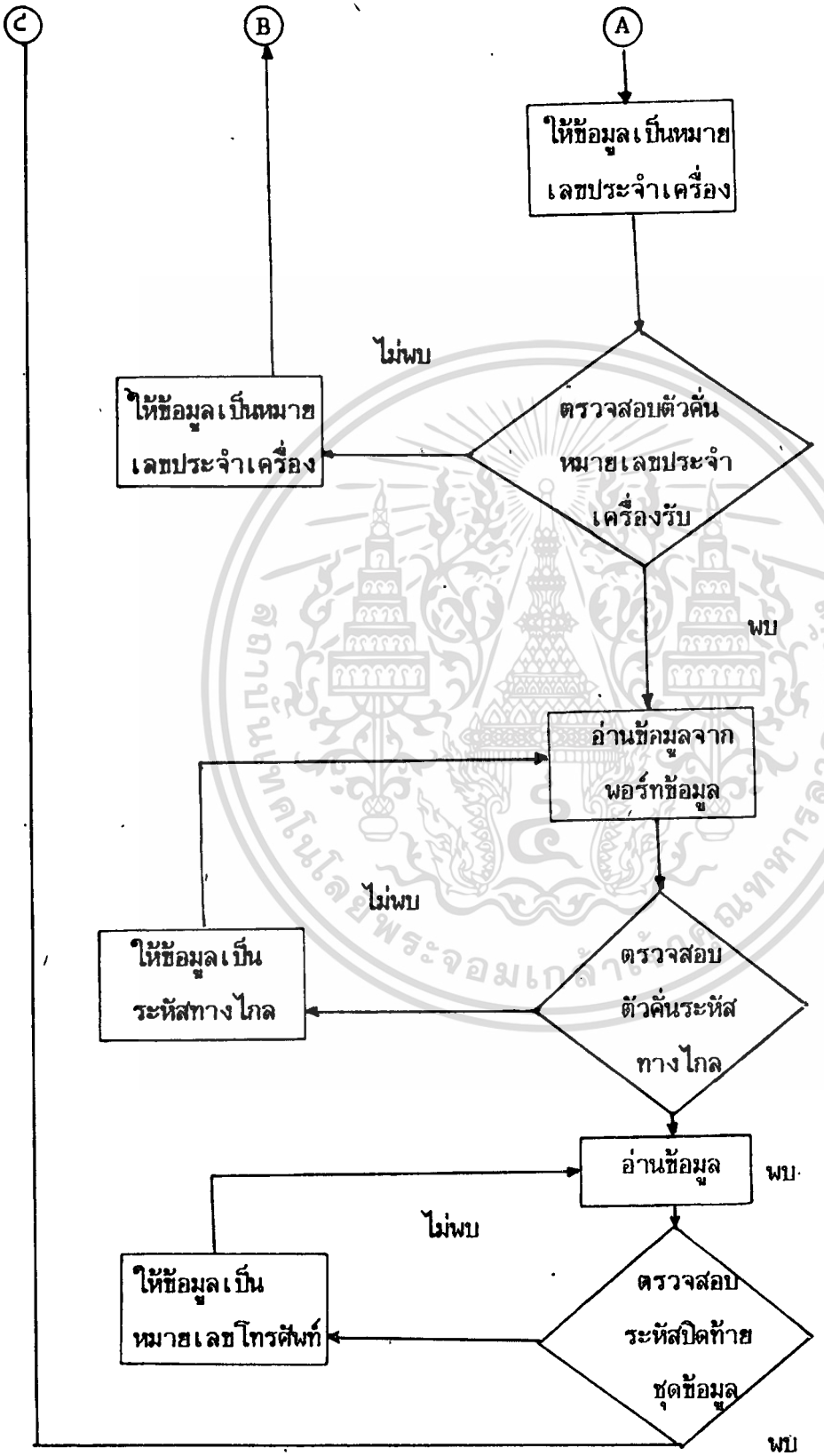
5. โปรแกรมจัดการการส่งข้อมูล ทำหน้าที่ตรวจสอบความพร้อมของวงจรส่ง เมื่อวงจรส่งพร้อมแล้วจะทำการส่งข้อมูลโดยเริ่มต้นด้วยไบท์เริ่มต้นซึ่งมีค่าพิเศษตามที่ตกลงกันไว้แล้วจึงส่งหมายเลขประจำเครื่องของเครื่องรับซึ่งเป็นเลขจำนวนจริงห้าหลักแทนด้วยเลขไบนารี่สองไบท์ตามด้วยคอมโทรลไบท์ที่บอกจำนวนหลักของระหัสทางไกลและหมายเลขโทรศัพท์โดยที่สี่บิทแรกบอกจำนวนหลักของระหัสทางไกลสี่บิทหลังบอกจำนวนหลักของหมายเลขโทรศัพท์จากนั้นจึงส่งหมายเลขโทกลับทั้งหมดที่เก็บไว้ในอะเรย์ออกไปที่ละชุดโดยส่งที่ละสองหลักแทนด้วยเลขไบนารี่หนึ่งไบท์ โดยแยกแต่ละหลักแสดงด้วยสี่บิท การส่งข้อมูลทั้งหมดนี้จะส่งไบท์ละสามครั้งเพื่อให้เครื่องรับสามารถตรวจสอบหาข้อมูลที่ถูกต้องได้ ผังงานของโปรแกรมนี้แสดงได้ดังรูปที่ 2.5



รูปที่ ที่ 2.3 ผังการทำงานของโปรแกรมหลัก



รูปที่ 2.4 ก ผังการทำงานของโปรแกรมรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์

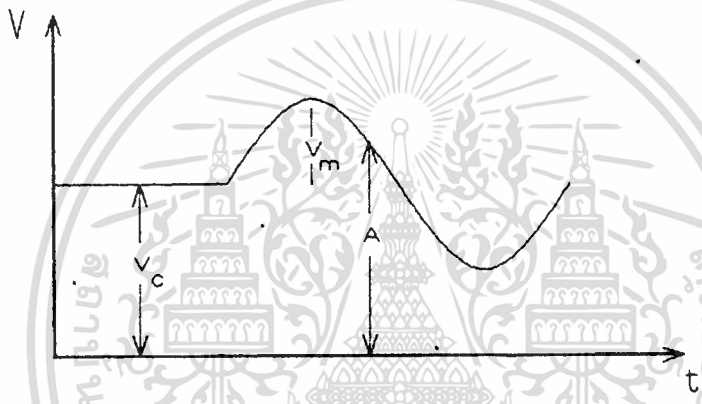


รูปที่ 2.4 ข ผังการทำงานของโปรแกรมรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์

รูปที่ 2.5 ผังงานการส่งข้อมูล

2.5 หลักการส่งคลื่นพาหะรอง

ก่อนจะกล่าวถึงการส่งคลื่นพาหะรองจะขออธิบายถึงการส่งสัญญาณในระบบ เอ.เอ็ม.อย่างคร่าวๆเสียก่อน การส่งสัญญาณในระบบเอ.เอ็ม.สามารถทำได้โดยการเปลี่ยนแปลงขนาดของคลื่นพาหะ (Carrier) ไปตามสัญญาณข้อมูลจะทำให้สัญญาณดังรูป 2.6



รูปที่ 2.6 สัญญาณที่ถูกมอดูเลทแบบ เอ.เอ็ม.

สัญญาณนี้เมื่อพิจารณาสเปกตรัมของความถี่ (Frequency Spectrum)จะพบว่าประกอบด้วยความถี่ของคลื่นพาหะและความถี่ไซด์แบนด์ (Sideband) ซึ่งสามารถพิสูจน์ได้จากสมการต่อไปนี้

เมื่อคลื่นพาหะ $v = V \sin w t$

คลื่นสัญญาณ $v = V \sin w t$

จะหามอดูเลชั่นอินเด็กซ์(m)ได้จาก $m = V / V$

และจะได้ขนาดของแรงดันมอดูเลท

$$A = V + v$$
$$= V + V \sin w t$$
$$= V + mV \sin w t$$
$$= V (1 + m \sin w t)$$

แต่เนื่องจาก V มีการเปลี่ยนแปลงแบบไซนูซอยดัล(sinusoidal) ดังนั้นค่าแรงดันชั่วขณะ (Instantaneous Voltage) จะสามารถหาได้จากการแทนค่า

$$v = A \sin w t$$

$$= V (1+m \sin w t) \sin w t$$

โดยใช้สมการทางตรีโกณมิติ

$$\sin x \sin y = 1/2[\cos(x-y) - \cos(x+y)]$$

เราสามารถจัดสมการได้ใหม่เป็น

$$v = V \sin w t + mV \cos(w - w)t/2 - mV \cos (w - w)t/2$$

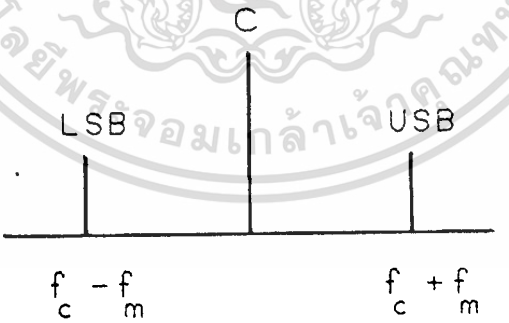
จะเห็นได้ว่าแรงดันนี้ประกอบด้วยสามความถี่คือ

f เป็นความถี่ของคลื่นพาหะ

$f - f$ เป็นความถี่โลเวอร์ไซด์แบนด์(Lower sideband:LSB)

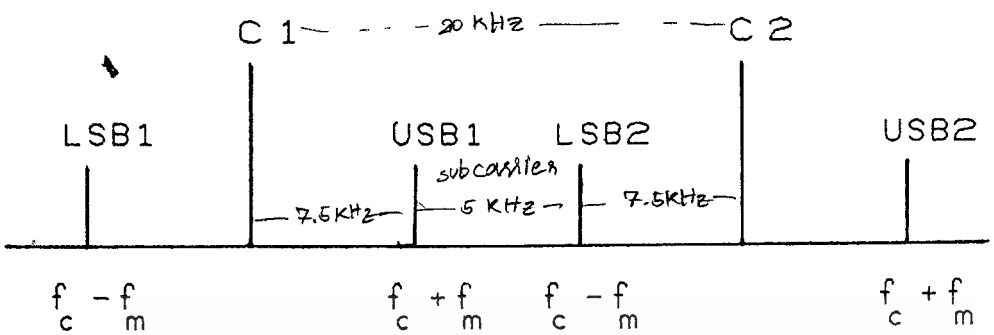
$f + f$ เป็นความถี่อัปเปอร์ไซด์แบนด์(Upper sideband:USB)

จากการวิเคราะห์สเปกตรัมของความถี่ข้างบนนี้จะเห็นได้ว่าช่วงกว้างของความถี่ (Frequency Band) ของสถานีวิทยุเอ.เอ็ม.จะมีค่าเป็นสองเท่าของความถี่สูงสุดของคลื่นที่มอดูเลท โดยปรกติแล้วสถานีวิทยุกระจายเสียงระบบเอ.เอ็ม.จะจำกัดความถี่สูงสุดของสัญญาณเสียงที่ต้องการส่งไว้ที่ 7.5 KHz เพื่อไม่เกิดการรบกวนสถานีข้างเคียง



รูปที่ 2.7 สเปกตรัมความถี่ของคลื่นเอ.เอ็ม.

สำหรับวิทยุกระจายเสียงในระบบนี้จะมีช่วงกว้างของความถี่ประมาณ 15 KHz แต่สถานีวิทยุแต่ละสถานีนั้นจะมีความถี่คลื่นพาหะห่างกันประมาณ 20 KHz ทำให้มีช่วงว่างของความถี่ระหว่างสถานีอยู่ดังรูปที่ 2.8 ทำให้สามารถส่งสัญญาณคลื่นความถี่รองที่มีความถี่ใกล้เคียงกับ 7.5 KHz (เช่น 8 KHz) ไปพร้อมกับสัญญาณเสียงได้โดยไม่มีการรบกวนสถานีข้างเคียงเราเรียกสัญญาณนี้ว่าคลื่นพาหะรอง (Subcarrier)

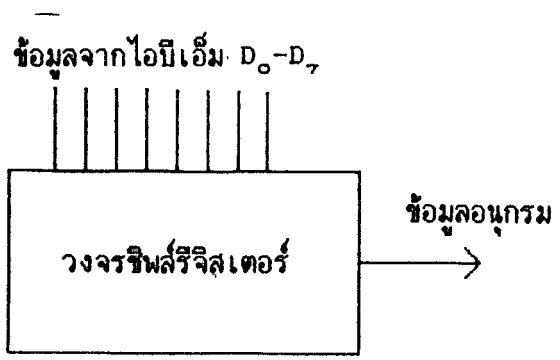


รูปที่ 2.8 สเปกตรัมของสถานีวิทยุที่อยู่ใกล้เคียงกัน

เราสามารถส่งข้อมูลไปกับคลื่นพาหะรองโดยใช้การมอดูเลทแบบเอ็ฟ. เอ็ม โดยให้ช่วงความถี่ที่เปลี่ยนแปลงมีค่าไม่กว้างนัก เพื่อไม่ให้เกิดการรบกวนสถานีข้างเคียง คลื่นพาหะนี้จะไม่ทำเกิดเสียงรบกวนในเครื่องรับวิทยุธรรมดามากนักเนื่องจากเครื่องรับวิทยุมีวงจรกรองความถี่ต่ำอยู่ทำให้ลดทอนคลื่นพาหะรองลง

2.6 หลักการส่งข้อมูลจากเครื่องไมโครคอมพิวเตอร์

เมื่อเครื่องไมโครคอมพิวเตอร์ไอบีเอ็ม (IBM) รับข้อมูลมาจากไมโครโปรเซสเซอร์ 8048 ก็จะทำการส่งข้อมูลไปยังระบบวิทยุ ซึ่งข้อมูลของเครื่องไมโครคอมพิวเตอร์ไอบีเอ็มนั้นเป็นแบบขนาน (Paralel) คือมีขาข้อมูล 8 เส้น จะต้องนำมาทำการแปลงให้เป็นแบบอนุกรมก่อน จึงจำทำการส่งข้อมูลไปยังระบบวิทยุได้ สำหรับการแปลงข้อมูลแบบขนานให้เป็นแบบอนุกรมนี้ใช้วงจรชิฟท์รีจิสเตอร์ (Shift Register) โดยอัตราการส่งข้อมูลออกของข้อมูลแบบอนุกรมขึ้นอยู่กับ สัญญาณนาฬิกาที่มาทริก (Trig)



รูปที่ 2.9 การแปลงข้อมูลแบบขนานเป็นแบบอนุกรม

2.7 หลักการวงจรสร้างคลื่นพาหะรองและการมอดูเลทแบบความถี่

ก่อนที่จะส่งข้อมูลเข้าตัวส่งวิทยุ จะต้องทำการมอดูเลทข้อมูลเสียก่อน เพื่อที่จะทำให้ตัวรับตรวจจับข้อมูลที่ส่งไปได้อย่างถูกต้อง การมอดูเลทนี้จะทำการมอดูเลทแบบความถี่ คือให้ข้อมูลที่มีโลจิก 0 มีความถี่สูงกว่าข้อมูลที่มีโลจิก 1 คือให้โลจิก 0 มีความถี่ประมาณ 8.4 กิโลเฮิร์ซ (KHz) และให้โลจิก 1 มีความถี่ประมาณ 7.6 กิโลเฮิร์ซ (KHz) ซึ่งหลักการที่ใช้ในการมอดูเลทนี้มี 2 แบบ คือ

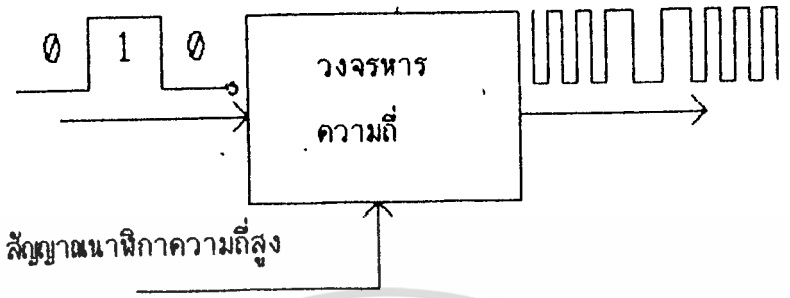
2.7.1 หลักการมอดูเลท โดยใช้วงจรโวลท์เตจคอนโทรลออสซิเลเตอร์ (Voltage Controlled Oscillator)

การมอดูเลทแบบนี้เป็นการมอดูเลทที่ใช้ความต่างศักย์ไปควบคุมความถี่ที่เกิดขึ้น เนื่องจากโลจิก 0 และ 1 มีระดับความต่างศักย์ 0 โวลต์ และ 5 โวลต์ ตามลำดับ เมื่อป้อนเข้าวงจรโวลต์เตจคอนโทรลออสซิเลเตอร์แล้ว แต่ละความถี่จะให้ความถี่ออกมาต่างกัน 2 ความถี่ ดังรูปที่ 2.10

รูปที่ 2.10 แสดงการมอดูเลทด้วยโวลต์เตจคอนโทรลออสซิเลเตอร์

2.7.2 หลักการมอดูเลทโดยใช้วงจรหารความถี่

การมอดูเลทแบบนี้เป็นการมอดูเลท โดยใช้ข้อมูลที่ต้อการส่งซึ่งเป็นโลจิก 0,1 ไปควบคุมการหารของวงจรความถี่จากความถี่ของสัญญาณนาฬิกาที่ตั้งไว้ โดยถ้าโลจิกเป็น 0 จะทำให้ตัวหารมีค่ามาก ทำให้ความถี่ที่ถูกหารมีค่าน้อย และถ้าโลจิกเป็น 1 จะทำให้ตัวหารมีค่าน้อย และทำให้ความถี่ที่ถูกหารมีค่ามาก



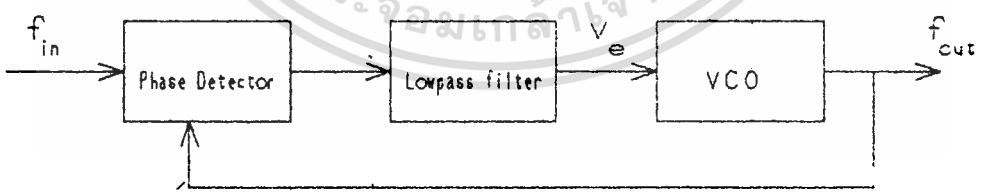
รูปที่ 2.11 แสดงการมอดูเลทด้วยหารความถี่

2.8 หลักการดีมอดูเลทสัญญาณออกจากคลื่นพาหะรอง

2.8.1 หลักการดีมอดูเลทโดยใช้เฟสล็อคลูป (Phase Lock Loop)

เฟสล็อคลูปเป็นระบบป้อนกลับทางอิเลคทรอนิคส์มีส่วนประกอบที่สำคัญสามส่วน คือ

1. วงเปรียบเทียบเฟส (Phase comparater)
2. วงจรกรองผ่านความถี่ต่ำ (Low-Pass Filter)
3. วงจรวี.ซี.โอ. (VCO:Voltage control oscilator)



รูปที่ 2.12 ส่วนประกอบของเฟสล็อคลูป

วงจรเปรียบเทียบเฟสจะทำหน้าที่เปรียบเทียบสัญญาณอินพุทกับสัญญาณที่ป้อนกลับมาจากวี.ซี.โอ.และจะให้เอ๊าท์พุทเป็นแรงดันความแตกต่าง (Error Voltage) ซึ่งจะมีแรงดันเฉลี่ยแปรผันตามความแตกต่างของความถี่และเฟสระหว่างสัญญาณอินพุทกับสัญญาณของ วี.ซี.โอ. แรงดันนี้จะถูกกรองโดยวงจรกรองผ่านความถี่ต่ำเพื่อกำจัดสัญญาณ

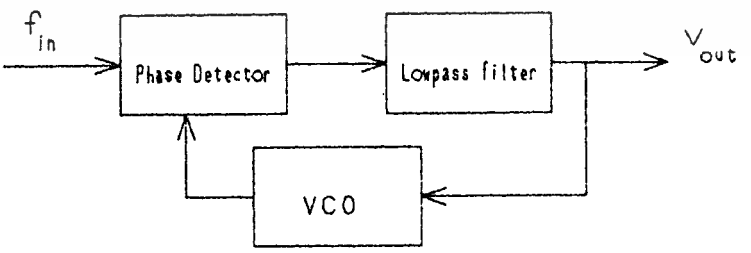
ความถี่สูงทำให้การควบคุมเป็นไปอย่างราบเรียบ สัญญาณนี้จะป้อนให้กับวงจรวี.ซี.โอ. เพื่อควบคุมให้ความถี่ของวี.ซี.โอ. เปลี่ยนแปลงไปในทิศทางที่จะลดความแตกต่างของความถี่ลง ทำให้เอ๊าท์พุทมีความถี่ใกล้กับอินพุทมากขึ้นลักษณะนี้เรียกว่าเฟสล็อคลูปทำงานอยู่ในสภาวะแค๊ปเจอร์ (Capture) จนกระทั่งเอ๊าท์พุทมีความถี่เท่ากับสัญญาณอินพุทแต่จะมีเฟสต่างกันเพื่อให้เกิดแรงดันความแตกต่างไปควบคุมความถี่ของวี.ซี.โอ. ในลักษณะนี้เรียกว่าอยู่ในสภาวะเฟสล็อค

ในขณะที่ไม่มีสัญญาณอินพุททำให้แรงดันความแตกต่างมีค่าเป็นศูนย์ดังนั้น วี.ซี.โอ. จะให้ความถี่คงที่ค่าหนึ่งเรียกว่าความถี่ฟรีรันนิ่ง (Free-Running) เฟสล็อคลูปจะเริ่มทำงานเมื่อสัญญาณอินพุทมีความถี่อยู่ในการทำงาน พิสัยการทำงานของเฟสล็อคลูปมีสองค่าคือ

1. พิสัยการเข้าสู่สภาวะล็อค (Capture Range) เป็นช่วงที่เฟสล็อคลูปเริ่มมีการเปลี่ยนแปลงตามสัญญาณอินพุท (เปลี่ยนจากสภาวะฟรีรันนิ่ง)

2. พิสัยการล็อค (Lock Range) เป็นช่วงความถี่ที่เฟสล็อคลูปยังสามารถเปลี่ยนแปลงตามอินพุทอยู่ได้ (และจะกลับสู่สภาวะฟรีรันนิ่ง) พิสัยนี้จะมีช่วงกว้างกว่าพิสัยแรก. ความกว้างของพิสัยการล็อคจะขึ้นกับความสามารถของวี.ซี.โอ. ที่จะเปลี่ยนแปลงความถี่ได้

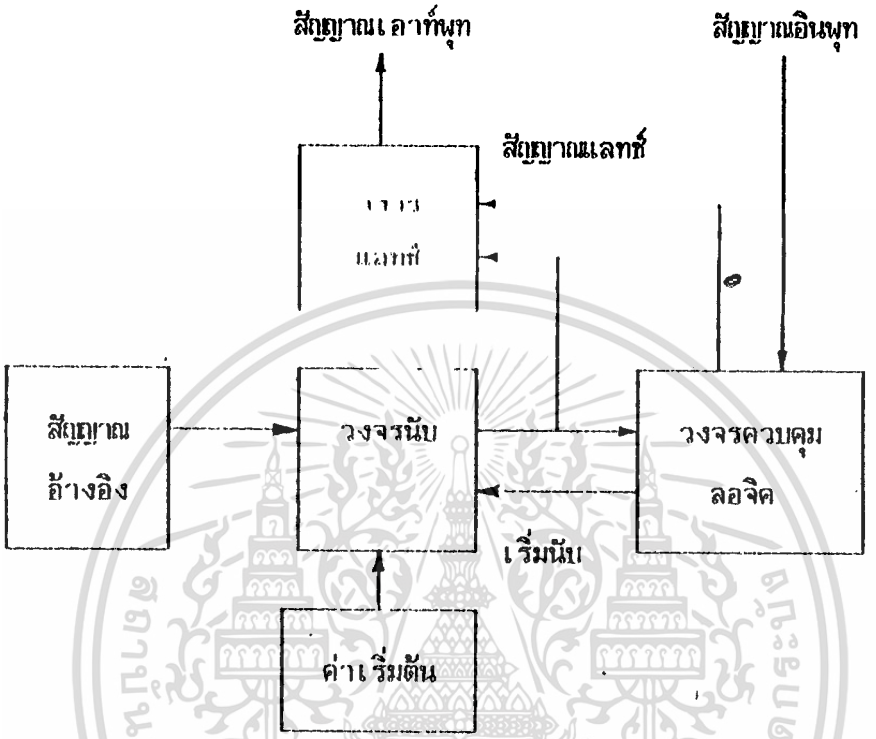
เมื่อเราพิจารณาแรงดันความแตกต่างที่ได้จากวงจรกรองผ่านความถี่ต่ำจะเห็นได้ว่ามีค่าเปลี่ยนแปลงตามความถี่ของสัญญาณอินพุท ซึ่งสามารถนำไปใช้เป็นเอฟ.เอ็ม. ดีมอดูเลเตอร์ได้ (มีคุณสมบัติเปลี่ยนแปลงความถี่เป็นแรงดัน)



รูปที่ 2.13 การใช้เฟสล็อคลูปเป็นเอฟ.เอ็ม.ดีมอดูเลเตอร์

2.8.2 หลักการดีมอดูเลทโดยใช้วงจรนับ

การดีมอดูเลทด้วยวิธีนี้จะอาศัยอุปกรณ์ต่างๆ เป็นส่วนประกอบดังนี้



รูปที่ 2.14 แสดงส่วนประกอบของวงจรดีมอดูเลเตอร์โดยใช้วงจรนับ

การทำงานของระบบจะเริ่มขึ้นเมื่อมีสัญญาณพัลซ์ลูกแรกเข้ามาที่อินพุทของวงจรควบคุม จะทำให้วงจรควบคุมส่งสัญญาณไปสั่งงานให้วงจรนับเริ่มต้นทำงาน วงจรนับก็จะนับสัญญาณนาฬิกาอ้างอิง ซึ่งมีความถี่คงที่แน่นอน ขณะที่วงจรนับกำลังทำงานอยู่นั้น ถ้าหากมีสัญญาณพัลซ์ลูกต่อมาเกิดขึ้นที่อินพุทของวงจรควบคุม ก็จะทำให้วงจรควบคุมส่งสัญญาณไปสั่งให้วงจรแลทช์ เก็บค่าของวงจรนับให้ค้างไว้ที่เอาท์พุท ขณะเดียวกันวงจรควบคุมก็จะสั่งให้วงจรนับเริ่มต้นทำงานใหม่อีกครั้ง การทำงานจะวนอยู่เช่นนี้เรื่อยไป

ค่าของวงจรนับจะมีค่าแตกต่างกันไปตามคาบเวลาระหว่างพัลซ์สองลูกที่อินพุทของวงจรควบคุม โดยถ้าหากสัญญาณนี้เป็นสัญญาณที่มีความถี่สูง คาบเวลาระหว่างพัลซ์ก็จะมีค่าน้อย ค่าที่ได้จากวงจรนับก็จะมีค่าน้อยด้วย แต่ถ้าหากสัญญาณอินพุทมีความถี่ต่ำ ค่าที่ได้จากวงจรนับก็จะมีค่ามากขึ้น

ในการใช้งานจริงนั้น เราต้องการเปลี่ยนสัญญาณที่มีความถี่ต่างกันสองค่า ให้

เป็นสัญญาณลอจิค ซึ่งมีเพียงสองระดับ โดยกำหนดให้

f1 เป็นค่าความถี่ของสัญญาณอินพุทที่แทนลอจิคศูนย์

f2 เป็นค่าความถี่ของสัญญาณอินพุทที่แทนลอจิคหนึ่ง

fr เป็นค่าความถี่ของสัญญาณนาฬิกาอ้างอิง

n1 เป็นค่าของวงจรนับที่ได้เมื่อป้อนสัญญาณความถี่ f1

n2 เป็นค่าของวงจรนับที่ได้เมื่อป้อนสัญญาณความถี่ f2

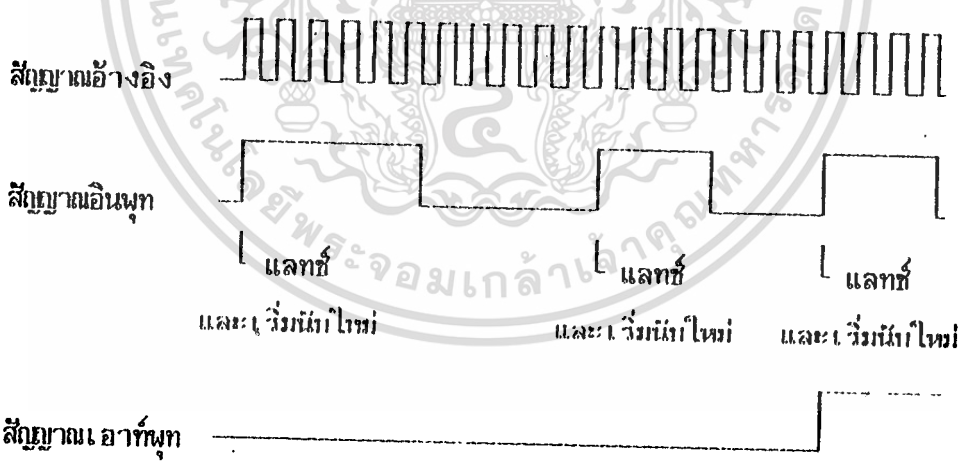
ดังนั้น จะได้ว่า $n1 = fr/f1$ และ $n2 = fr/f2$

$nc = (n1+n2)/2$ และ $fc = fr/nc$

และเมื่อกำหนดให้ $f1<f2$ จะได้ว่า

$f1<fc<f2$ และ $n2<nc<n1$

จากความสัมพันธ์เหล่านี้จะเห็นว่าเราสามารถออกแบบวงจรได้ง่ายขึ้น โดยออกแบบให้วงจรนับส่งสัญญาณลอจิคหนึ่งออกมาเมื่อวงจรนับสัญญาณนาฬิกาอ้างอิงได้น้อยกว่า nc ลูก และส่งสัญญาณลอจิคศูนย์ออกมาเมื่อวงจรนับสัญญาณนาฬิกาอ้างอิงได้มากกว่า nc ลูก



รูปที่ 2.15 รูปแสดงสัญญาณอินพุทและเอาท์พุทของวงจรดีมดูเลท

จากรูปนี้จะเห็นว่า n1 = 11 , n2 = 7 , nc = 9

การดีมอดูเลทด้วยวิธีนี้จะลาดเลี่ยงการทำงานของวงจรทางดิจิตอลเป็นส่วนมาก และลาดเลี่ยงสัญญาณอ้างอิงเพียงสัญญาณเดียว ซึ่งถ้าหากใช้วงจรออสซิเลเตอร์ที่สามารถผลิตสัญญาณนาฬิกาที่มีความถี่เที่ยงตรง แน่นอนแล้ว ก็จะทำให้วงจรมีความเที่ยงตรงไปด้วย การทำงานของวงจรไม่ขึ้นกับค่า DUTY CYCLE ของสัญญาณอินพุทเลย และเนื่องจากเป็น

วงจรทางดิจิตอล ดังนั้นจึงสามารถปรับแต่งได้ง่ายด้วย

2.9 หลักการของวงจรประมวลผลและแสดงผลข้อมูล

ข้อมูลที่ถูกส่งมาจากเครื่องส่งวิทยุ เมื่อผ่านวงจรส่วนดีมอดูเลทแล้ว ก็จะเป็นข้อมูลทางดิจิตอลแบบอนุกรม ซึ่งประกอบด้วยลอจิคหนึ่งและศูนย์เท่านั้น ข้อมูลดังกล่าวนี้จะถูกป้อนเข้าสู่วงจรส่วนประมวลผลและแสดงผลนี้

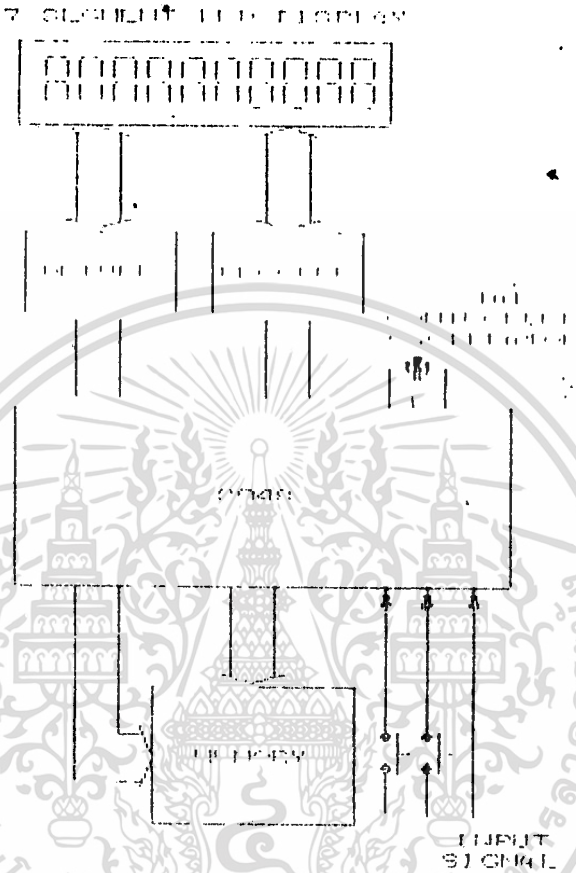
หน้าที่หลักของวงจรส่วนนี้ได้แก่

1. รับข้อมูลแบบอนุกรม แล้วเปลี่ยนเป็นข้อมูลแบบขนาน
2. ตรวจสอบว่าข้อมูลชุดนี้ เป็นของเพจเจอร์ของเราหรือไม่ ถ้าหากเป็นข้อมูลสำหรับเพจเจอร์อื่นๆ ก็จะไม่สนใจข้อมูลชุดนั้น แต่ถ้าหากเป็นข้อมูลของมันเอง ก็จะรับข้อมูลต่อไป ซึ่งข้อมูลที่ตามหลังมานั้นจะเป็นหมายเลขโทรศัพท์และรหัสทางไกลรวมกัน
4. ส่งสัญญาณเรียกออกมา เพื่อให้ผู้ใช้งานทราบ เมื่อได้รับข้อมูลครบทั้งชุดแล้ว
5. ตรวจสอบดูว่ามีการกดสวิทช์หรือไม่ ถ้าหากสวิทช์ถูกกด ก็จะแสดงข้อมูลออกมาทางส่วนแสดงผล
6. นำข้อมูลออกมาแสดงผลผ่านทาง 7 SEGMENT LED DISPLAY จำนวน 10 หลัก โดยใช้หลักการมัลติเพล็กซ์ (MULTIPLEX)

จากที่กล่าวมาข้างต้นแล้วถึงหน้าที่ต่างๆของวงจรนี้ ซึ่งจะเห็นว่าเป็นหน้าที่ซึ่งสลับซับซ้อนและยุ่งยาก ถ้าหากใช้วงจรแยกส่วนตามปกติก็จะได้วงจรที่มีขนาดใหญ่มาก จึงนำไอซีไมโครคอมพิวเตอร์ชิพเดี่ยว 8048 มาใช้ในวงจรนี้ ซึ่งทำให้ได้วงจรที่มีส่วนประกอบต่างๆดังรูปที่ 2.16

ในส่วนของระบบไมโครคอมพิวเตอร์ 8048 นั้นจะต้องประกอบด้วย ส่วนสร้างสัญญาณนาฬิกาสำหรับระบบ และส่วนของหน่วยความจำอันเป็นที่เก็บของโปรแกรมควบคุมการทำงานของวงจรส่วนนี้

ในส่วนของวงจรที่ติดต่อกับระบบไมโครคอมพิวเตอร์ 8048 ก็จะประกอบด้วย สัญญาณอินพุทข้อมูลแบบอนุกรม ,วงจรสวิทช์ ,วงจรขับ (DRIVER) ส่วนแสดงผล และวงจรเลือกหลักตัวแสดงผล



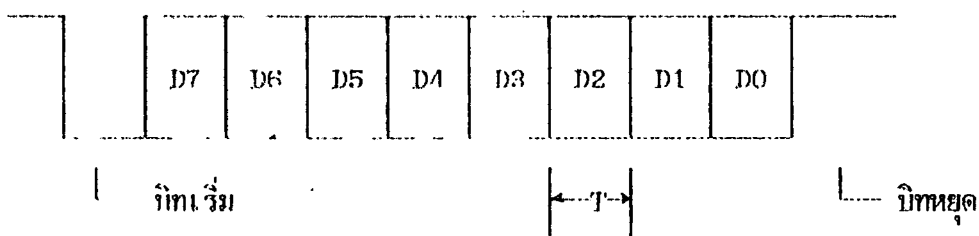
รูปที่ 2.16 แสดงส่วนประกอบของวงจรประมวลผลและแสดงผลข้อมูล

การควบคุมการทำงานของวงจรให้เป็นไปตามที่เราต้องการนั้น จะขึ้นอยู่กับตัวโปรแกรมเป็นส่วนใหญ่ โดยเราสามารถจะเขียนโปรแกรมสั่งงานให้วงจรส่วนอื่นๆที่ต่ออยู่กับ 8048 ทำงานอย่างไรก็ได้ นอกจากนี้การกำหนดว่าจะต่อวงจรอะไรเข้ากับส่วนไหนของ 8048 ก็ขึ้นอยู่กับความเหมาะสมของโปรแกรมด้วยเช่นกัน ตัวโปรแกรมทั้งหมดจะประกอบด้วยโปรแกรมย่อยๆหลายๆโปรแกรม ซึ่งทำหน้าที่ต่างกันไปดังนี้

1. โปรแกรมย่อยสำหรับอ่านข้อมูลแบบอนุกรม

โปรแกรมย่อยนี้จะอ่านข้อมูลจากวงจรดีมอดดูเลททีละบิท จนครบหนึ่งไบท์ (8 บิท) ในการออกแบบโปรแกรมเพื่ออ่านข้อมูลเข้ามานั้นจะต้องคำนึงถึงปัจจัยต่างๆดังรูปที่

2.17



รูปที่ 2.17 แสดงลักษณะของข้อมูลแบบอนุกรม

โดยเวลาข้อมูลแต่ละไบท์หรือแต่ละชุดนั้น ไม่ได้มีกำหนดเวลาที่แน่นอน ดังนั้น เมื่อฝั่งภาครับจะทราบได้ว่าเวลาส่งข้อมูลมาเมื่อใดจากเครื่องส่ง ก็จะพบว่าบิทเริ่มของข้อมูลถูกกำหนดให้เป็นลอจิกศูนย์เสมอ ดังนั้นถ้าเราต่อสัญญาณข้อมูลป้อนเข้าสู่ 8048 ทางขา INT เมื่อมีข้อมูลเข้ามา บิทเริ่มของข้อมูลแต่ละไบท์ก็จะไปอินเตอร์รัพท์ 8048 ทำให้ 8048 หยุดจากงานอื่นๆแล้วเตรียมตัวรับข้อมูลบิทต่อไปได้

ในการส่งข้อมูลแบบอนุกรมนี้ คาบเวลาของข้อมูลแต่ละบิทจะมีค่าเท่ากันหมด ซึ่งขึ้นอยู่กับอัตราการส่งข้อมูลของเครื่องส่ง ทางด้านเครื่องรับก็จะต้องรับข้อมูลด้วยอัตราเท่าๆกัน หรือใกล้เคียงกันมาก ซึ่งจะเห็นว่าระยะเวลาในการอ่านข้อมูลแต่ละบิทนั้นต้องมีความเที่ยงตรงแม่นยำ มิฉะนั้นจะทำให้การอ่านข้อมูลเลื่อนไปหรือผิดพลาดได้ เพื่อให้ได้คาบเวลาที่แน่นอน เราจึงใช้วงจรนับที่มีอยู่ภายใน 8048 มาทำหน้าที่กำหนดคาบเวลาการอ่านนี้ ตัววงจรนับนี้จะอาศัยสัญญาณนาฬิกาของระบบภายในมาเป็นฐานเวลาในการนับ ซึ่งสัญญาณนาฬิกานี้ผลิตจากวงจรออสซิลเลเตอร์ที่ควบคุมความถี่โดยคริสตอล ดังนั้นจึงได้คาบเวลาที่เที่ยงตรง และแน่นอนมาให้

เมื่อ 8048 ถูกอินเตอร์รัพท์โดยบิทเริ่มของข้อมูลแล้ว 8048 จะเตรียมตัวสำหรับอ่านข้อมูลอีก 8 บิทต่อไป โดยเซทค่ารีจิสเตอร์สำหรับนับจำนวนบิท ,ตั้งโปรแกรมให้วงจรนับส่งสัญญาณอินเตอร์รัพท์ทุกๆคาบเวลาของข้อมูลแต่ละบิท

และเมื่อวงจรนับส่งสัญญาณอินเตอร์รัพท์ออกมา ก็จะหมายความว่าถึงเวลาที่จะอ่านข้อมูลบิทถัดไปแล้ว 8048 ก็จะอ่านลอจิกที่ขา INT เข้ามาเป็นข้อมูล 1 บิทไปเก็บไว้ในรีจิสเตอร์ จนกระทั่งครบหนึ่งไบท์ ก็จะเตรียมตัวสำหรับอ่านข้อมูลไบท์ใหม่ต่อไป

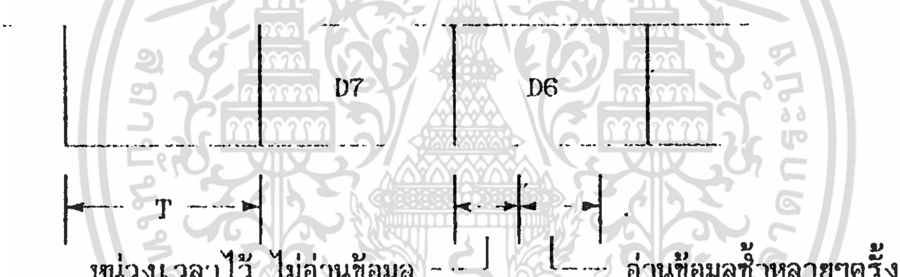
เพื่อให้การอ่านข้อมูลมีความถูกต้องมากขึ้น จำเป็นต้องมีวิธีการลดความผิดพลาดที่อาจจะเกิดขึ้นได้ ดังนี้

1.1 ความผิดพลาดที่เกิดจากความล่าช้าของสัญญาณจากวงจรดีมอดูเลท ใน

ช่วงที่สัญญาณข้อมูลมีการเปลี่ยนแปลงจากความถี่หนึ่งไปยังอีกความถี่หนึ่ง วงจรดีมอดูเลทไม่สามารถให้สัญญาณเอาท์พุทเปลี่ยนแปลงตามได้ทันที ทำให้เกิดการล่าช้าขึ้น และค่าเวลาหน่วงนี้ (DELAY TIME) ในขณะที่สัญญาณข้อมูลเปลี่ยนจากลอจิคหนึ่งเป็นลอจิคศูนย์ จะมีค่าแตกต่างจากค่าเวลาหน่วงในขณะที่สัญญาณข้อมูลเปลี่ยนจากลอจิคศูนย์เป็นลอจิคหนึ่ง

ดังนั้นในตอนเริ่มอ่านข้อมูลจึงต้องหน่วงเวลาไว้ชั่วขณะ ยังไม่อ่านข้อมูลในช่วงแรกของคาบเวลาทันที

1.2 [illegible] จะทำให้อ่านได้ข้อมูลที่ผิดไป [illegible] แล้วจึงดูผล [illegible] ข้อมูลว่าเป็นลอจิคศูนย์หรือลอจิคหนึ่ง เช่น เมื่อเราอ่านข้อมูลบิทนั้นซ้ำสามครั้ง ถ้าข้อมูลบิทนั้นมีค่าเหมือนกันสองครั้งขึ้นไป ก็จะถือว่าข้อมูลนั้นถูกต้อง



รูปที่ 2.18 แสดงเวลาที่อ่านข้อมูล

2. โปรแกรมย่อยสำหรับตรวจสอบข้อมูล

เพื่อให้เครื่องรับมีโอกาสรับข้อมูลที่ถูกต้องได้มากขึ้น จึงออกแบบให้เครื่องส่งส่งข้อมูลซ้ำๆกันไบท์ละสามครั้ง เครื่องรับจะอ่านข้อมูลทั้งสามไบท์นี้เข้าไป แล้วอาศัยหลักการเปรียบเทียบตัวเลขแต่ละบิทระหว่างข้อมูลสามไบท์ ถ้าบิทนั้นๆมีลอจิคเหมือนกันสองบิทขึ้นไป ก็จะถือว่าข้อมูลที่เหมือนกันนั้นเป็นข้อมูลที่ถูกต้อง

กระบวนการตรวจสอบข้อมูลนี้ มีลักษณะคล้ายกับวงจรถอดรหัสวงจรหนึ่ง จึงสามารถใช้หลักในการออกแบบวงจรคอมบิเนชั่น (COMBINATION) มาประยุกต์ได้ดังนี้

ข้อมูลชุดที่			เอาท์พุท
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Y = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$= AB + C(A \leftrightarrow B)$$

$$= (A \text{ and } B) \text{ or } [C \text{ and } (A \text{ xor } B)]$$

หลังจากเสร็จกระบวนการนี้แล้ว ก็จะได้ข้อมูลเพียงไบท์เดียวที่จะนำไปใช้

3. โปรแกรมย่อยในการแสดงผล

ตัวแสดงผลที่ใช้เป็น 7 SEGMENT COMMOND CATHODE LED DISPLAY ขนาด 10 หลัก การแสดงผลตัวแสดงผลนี้ต้องอาศัยหลักการมัลติเพล็กซ์เข้าช่วย โดย 8048 จะนำข้อมูลส่งออกมาทางวงจรขับ ซึ่งวงจรขับนี้ก็จะต่ออยู่กับแต่ละเซกเมนต์ (SEGMENT) ของตัวแสดงผล ข้อมูลที่ส่งออกมานี้จะทำให้แต่ละเซกเมนต์ติดสว่างได้ตามต้องการ จากนั้น 8048 ก็จะเอาท์พุทข้อมูลอีกชุดหนึ่งออกมาทางตัวเลือกหลักแสดงผล เพื่อระบุว่า ต้องการให้หลักใดติดสว่าง ซึ่งจะมีเพียงหลักเดียวเท่านั้นที่ติดสว่าง หลักอื่นๆจะดับ

ถ้าหากเราต้องการแสดงข้อมูลทั้ง 10 หลัก เราก็จะอาศัยหลักการข้างต้นทำให้ตัวแสดงผลแต่ละหลักติดสว่าง ด้วยความเร็วสูง ก็จะทำให้สายตาคนไม่สามารถตรวจจับการเปลี่ยนแปลงนั้นได้

4. โปรแกรมหลักสำหรับประมวลผลข้อมูล

โดยอาศัยโปรแกรมรับข้อมูลข้างต้น เมื่อได้รับข้อมูลมาแล้วก็จะนำข้อมูลนั้นมาประมวลผล การนำข้อมูลไปใช้ประโยชน์อะไร ขึ้นกับลำดับขั้นของการประมวลผล ซึ่งเมื่อพิจารณารูปแบบการส่งข้อมูลแต่ละชุด ก็จะทราบว่าข้อมูลแต่ละไบท์ในหนึ่งชุดนั้นเป็นข้อมูลอะไรบ้าง เรียงลำดับอย่างไร ดังนี้

4.1 รหัสซิงโครนัส เป็นรหัสนำหน้าข้อมูลแต่ละชุด

4.2 หมายเลขของเพจเจอร์ที่ต้องการติดต่อ

4.3 รหัสทางไกลและหมายเลขโทรศัพท์

โดยเมื่อมีข้อมูลชุดใหม่ โปรแกรมจะตรวจสอบว่าข้อมูลที่ได้รับเป็นรหัสซิงโครนัสหรือไม่ เมื่อตรวจพบรหัสซิงโครนัสแล้ว ก็จะอ่านข้อมูลต่อไปเข้ามา แล้วเปรียบเทียบกับหมายเลขเพจเจอร์ของตนเอง ถ้าหากหมายเลขตรงกัน ก็จะหมายความว่าข้อมูลชุดนี้สำหรับเพจเจอร์ของตน แต่ถ้าหากหมายเลขที่ได้รับไม่ตรงกับหมายเลขประจำเพจเจอร์ของตน โปรแกรมก็จะไม่สนใจข้อมูลที่ตามมา แต่จะคอยรับข้อมูลชุดใหม่แทน โดยการตรวจจับรหัสซิงโครนัส

เมื่อรู้ว่าข้อมูลที่ตามมาเป็นของตน โปรแกรมก็จะอ่านข้อมูลที่เป็นรหัสทางไกลและหมายเลขโทรศัพท์เข้าไปเก็บไว้ในหน่วยความจำ จนเมื่อข้อมูลครบแล้วก็จะส่งสัญญาณเตือนออกมา จากนั้นจะตรวจสอบการกดสวิทช์ เมื่อมีการกดสวิทช์ก็จะนำข้อมูลในหน่วยความจำไปแสดงผลให้ผู้ใช้ทราบ

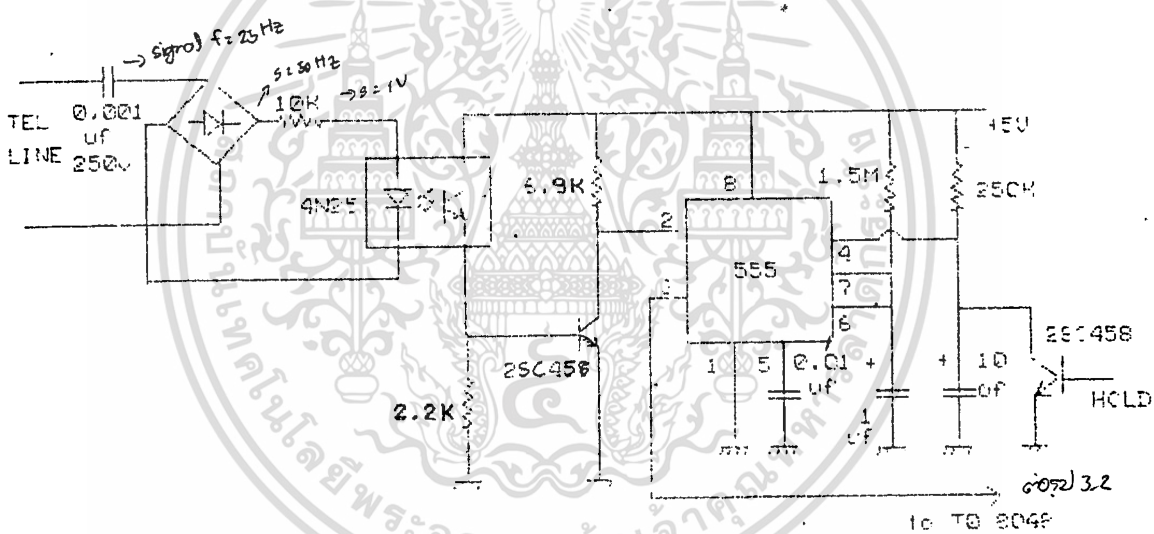
/

บทที่ 3 การออกแบบและการสร้าง

3.1 การสร้างวงจรรับข้อมูลโทรศัพท์

3.1.1 วงจรตรวจสอบสัญญาณเรียก (Ringing Detector)

วงจรตรวจสอบสัญญาณเรียกนี้ ใช้บริดจ์ไดโอด ซึ่งมีข้อดีคือ สามารถต่อสายโทรศัพท์เข้าที่ขั้วใดก็ได้ (กลับขั้วได้) นอกจากนี้ยังใช้ออปโตคัปเปอร์ (Opto Coupler) เพื่อแยกกราวน์(Ground)ของระบบโทรศัพท์ ซึ่งเป็นสัญญาณแรงดันสูงออกจากระบบของศูนย์ติดต่อ ซึ่งใช้แรงดันสัญญาณต่ำ เพื่อป้องกันการเสียหายของระบบศูนย์ติดต่อ จากระบบสัญญาณไฟสูง เมื่อมีการติดต่อเท่านั้น



รูปที่ 3.1 แสดงวงจรตรวจสอบสัญญาณเรียก

เมื่อมีสัญญาณกระดิ่ง(Ringing) เข้ามาจะผ่านตัวเก็บประจุจะกั้นสัญญาณไฟตรงไว้เหลือเพียงสัญญาณความถี่ 25 เฮิทร์ซ และเมื่อผ่านบริดจ์ไดโอดก็จะทำให้สัญญาณ เป็นสัญญาณซีกบวกแบบฟลูเวฟ(Full Wave) และเมื่อผ่านความต้านทาน 10 กิโลโอห์ม จะทำให้สัญญาณเหลือเพียงประมาณ 1 โวลต์ ซึ่งสามารถไบอัส LED ในตัวออปโตคัปเปอร์ และแสงจากตัว LED จะไบอัสที่เบสของทรานซิสเตอร์ ซึ่งอยู่ในตัวออปโตคัปเปอร์เช่นกัน หลังจากนั้นจะต่อเข้ากับทรานซิสเตอร์ภายนอก 2SC 458 เป็นอินเวอร์เตอร์(Inverter) จากขาคอลเล็คเตอร์ (Collecter) ของ 2SC 458 ซึ่งจะไป

ทริก (trig) วงจรโมโนสเตเบิล(Monostable) ที่สร้างจากไอซี 555 โดยตั้งเวลาของวงจรโมโนสเตเบิล ซึ่งมีเอาท์พุท 5 โวลต์เมื่อเกิดการทริกให้มีลอจิค 1ไปยังขา To ของไมโครโปรเซสเซอร์ (Microprocess) 8048 ต่อไป เพื่อทำให้ 8048 รู้ว่ามีการติดต่อเข้ามาที่คู่สายใด

3.1.2 วงจรพักสาย (Hold Line)

เมื่อไมโครโปรเซสเซอร์ 8048 รู้ว่าคู่สายใดทำการติดต่อมาก็ส่งสัญญาณเพื่อมาโฮลด(Hold)สายไว้เพื่อเรียกติดต่อกับคู่สายที่ต้องการจะโฮลดสัญญาณไว้ด้วยการทริกทรานซิสเตอร์ที่มาจากพอร์ตเพื่อให้สวิทซ์รีเลย์(Relay)ต่อความต้านทาน 600 โอห์ม คร่อมคู่สายซึ่งเป็นการเปรียบเสมือนการรับโทรศัพท์ และใช้งานรีเลย์ ใช้ไฟ 6 โวลต์และขดลวดมีความต้านทาน 150 โอหม์ และระบบจ่ายไฟของวงจรทั้งหมดใช้ไฟ 5 โวลต์ จึงต้อง ดึงไฟ ณ จุดที่ก่อนเข้าไอซีเรกกูเลท(IC Regulate) ที่จ่ายไฟให้ระบบซึ่งจะได้แรงดันไฟตรงประมาณ 8.4 โวลต์ จึงต้องใช้ความต้านทาน 51 โอหม์มาต่อกับขดลวดของรีเลย์ เพื่อให้แรงดันตกคร่อมรีเลย์ประมาณ 6 โวลต์ ซึ่งจะไม่ทำให้กระแสไหลผ่านขดลวดมากเกินไป

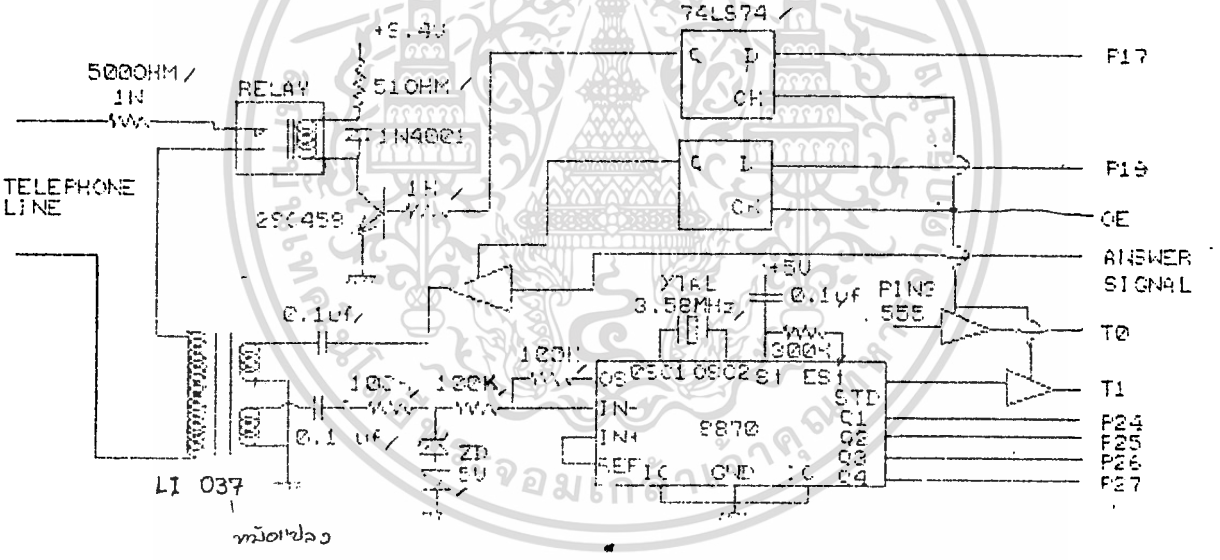
สำหรับปัญหาในการตัดต่อเพื่อพักสาย โดยใช้รีเลย์นี้ คือ เมื่อเลิกการติดต่อสวิทซ์ของรีเลย์ก็จะเปิดออกซึ่งจะทำให้เกิดสัญญาณรบกวน ไปทริกวงจรตรวจสอบสัญญาณเรียก ซึ่งจะทำให้วงจรโมโนสเตเบิลส่งสัญญาณเข้ามาอีก ซึ่งจะทำให้เกิดความผิดพลาดในการติดต่อขึ้น

การแก้ไขก็ทำได้โดย จะต่อสัญญาณที่จะไปทริกรีเลย์เข้ากับวงจรทรานซิสเตอร์ที่ทำหน้าที่เป็นอินเวอร์เตอร์ โดยต่อขาคอลเล็คเตอร์ของทรานซิสเตอร์เข้ากับขา 4 รีเซ็ท(Reset) ของ 555 โดยมีตัวเก็บประจุ 100 ไมโครฟารัด(UF) ต่อคร่อม ณ จุดนี้ด้วย ดังรูปข้างล่างนี้ โดยตัวเก็บประจุนี้จะเป็นตัวหน่วงแรงดันที่ขา 4 ของ 555 ให้ถึงระดับ 5 โวลท์ ช้าลงซึ่งเป็นการทำให้วงจรโมโนสเตเบิล เริ่มทำงานช้าลงคือหลังจากที่สวิทซ์รีเลย์เปิดออกเรียบร้อยแล้วทำให้สัญญาณรบกวนที่เกิดขึ้น จากการตัดต่อไม่เข้าไปรบกวนระบบอีกต่อไป

3.1.3 วงจรส่วนตอบรับสัญญาณและวงจรรับข้อมูลหมายเลข

สำหรับส่วนส่งสัญญาณตอบรับเพื่อให้ผู้ติดต่อทราบว่าสามารถเริ่มกดหมายเลขได้และส่วนรับข้อมูลที่ส่งผ่านมาทางสายโทรศัพท์ ซึ่งเป็นความถี่ 2 ความถี่ ทั้ง 2 ส่วนนี้จะ

ส่งผ่านสายโทรศัพท์ โดยให้การเหนี่ยวนำผ่านหม้อแปลงเป็นการแยกกราวนด์ของระบบทั้งสองออก โดยสัญญาณตอบรับนั้นจะส่งไปยังผู้ติดต่อ เมื่อสัญญาณอีนาเบิลการตอบรับ (ANSWER ENABLE) สัญญาณตอบรับก็จะเหนี่ยวนำผ่านหม้อแปลงไปยังคู่สายโทรศัพท์ โดยสัญญาณตอบรับนี้จะสร้างจากวงจรอะสเตเบิ้ล (Ast.able Crecuit.) โดยใช้ไอซี 555 สร้างความถี่ประมาณ 500 เฮิทร์ซ เมื่อผู้ติดต่อได้รับสัญญาณตอบรับก็จะกดหมายเลขวิทยุที่ต้องการติดต่อและหมายเลขโทรศัพท์ที่ต้องการให้โทรกลับไป ซึ่งสัญญาณนี้จะเป็นสัญญาณ [illegible] (Dual Tone Multi Frequency) ซึ่งจะถูกแปลงโดยไอซี 8870 ให้เป็นสัญญาณดิจิตอล 4 บิท ตามหมายเลขที่ส่งเข้ามา เพื่อส่งไปยังไมโครโปรเซสเซอร์ 8048 ต่อไป



รูปที่ 3.2 แสดงวงจรพักสาย และวงจรส่งสัญญาณตอบรับ และวงจรรับถ้อยมูลหมายเลข

โดยวงจรทั้งหมดที่กล่าวมานี้เป็นวงจรสำหรับหมายเลขโทรศัพท์ 1 หมายเลข ซึ่งระบบนี้สามารถติดต่อได้ 8 คู่สาย ซึ่งก็ต้องมีวงจรที่กล่าวมาทั้งหมด 8 ชุด โดยการควบคุมการติดต่อใช้ไมโครโปรเซสเซอร์ 8048 ซึ่งการติดต่อ วงจรทั้ง 8 ชุด จะใช้ลักษณะต่อแบบบัส โดยเมื่อ 8048 ต้องการติดต่อกับคู่สายใดก็จะส่งสัญญาณมาที่ขา OE เพื่อเป็นการอีนาเบิลให้คู่สายนั้นสามารถติดต่อกับ 8048 ได้

3.4.4 ส่วนควบคุมการรับ-ส่งข้อมูลทางโทรศัพท์

ตามที่ได้กล่าวถึงไปแล้วว่า วงจรควบคุมการรับ-ส่งข้อมูลทางโทรศัพท์นี้ทำหน้าที่ควบคุมการทำงาน และรับข้อมูลของวงจรส่วนหน้าหลายๆวงจรพร้อมกัน และส่งข้อมูลที่รวบรวมได้จากวงจรส่วนหน้าทั้งหมดส่งต่อไปให้เครื่องไมโครคอมพิวเตอร์ IBM PC/XT เพื่อนำไปใช้งานต่อไป

<u>รายละเอียดเกี่ยวกับวงจร</u>

วงจรควบคุมการรับ-ส่งข้อมูลทางโทรศัพท์ จะมีไมโครคอมพิวเตอร์ 8048 เป็นตัวประมวลผลหลัก และเป็นศูนย์กลางการควบคุม อุปกรณ์ต่างๆ จะต่ออยู่กับไมโครคอมพิวเตอร์ทั้งหมด ดังรูปที่ 3.3

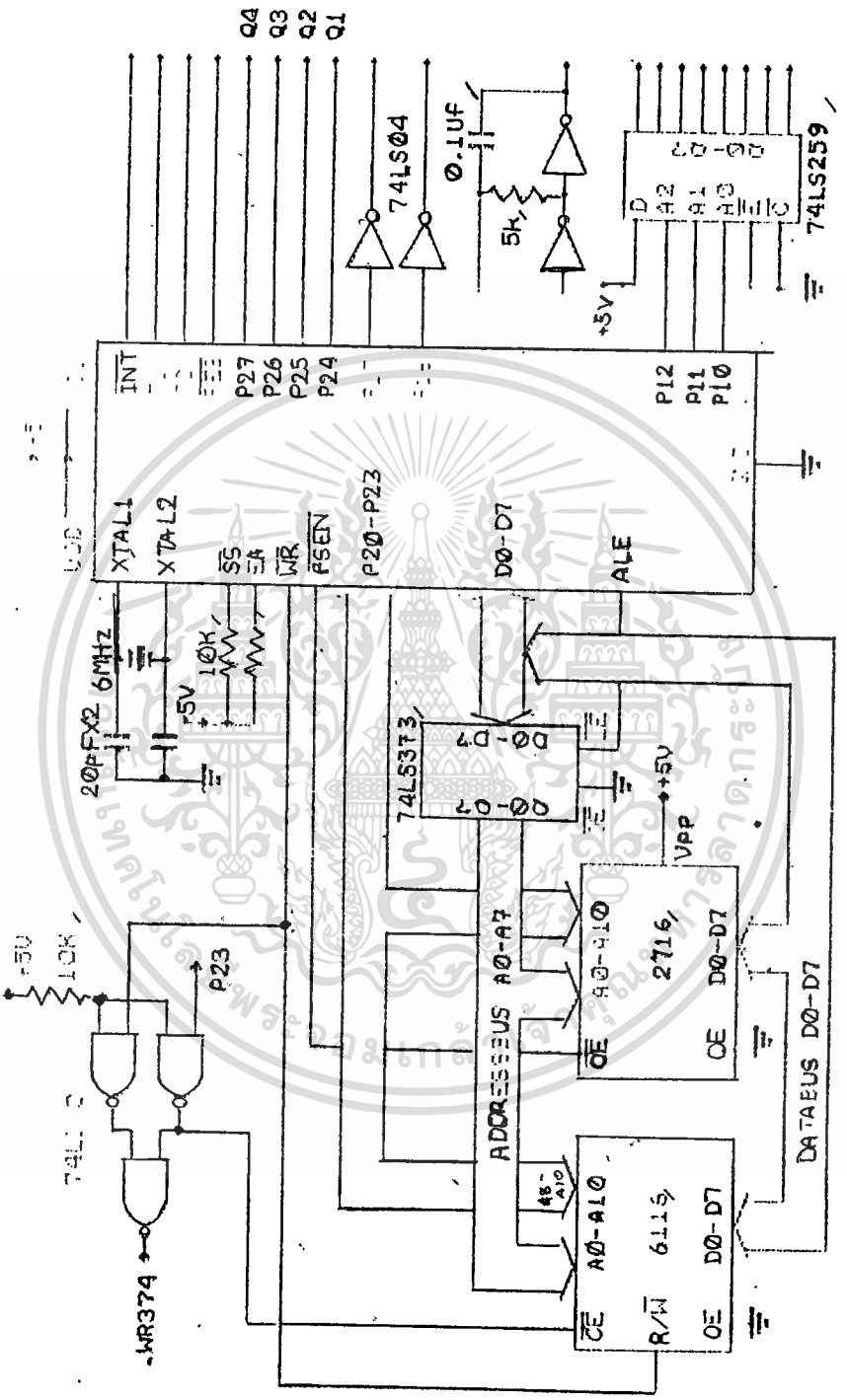
จากวงจรนี้จะเห็นว่า เราใช้คริสตอลขนาด 6 MHz ต่ออยู่ที่ขา XTAL1 และ XTAL2 สัญญาณนาฬิกาที่วงจรออสซิเลเตอร์ภายในผลิตได้ จะถูกหารความถี่ด้วยสาม ได้เป็นสัญญาณนาฬิกาความถี่ 2 MHz เพื่อใช้เป็นฐานเวลาของระบบภายใน 8048 ต่อไป

เราต่อขาควบคุม <u>EA ให้มีลอจิคหนึ่ง</u> <u>เพื่อให้ 8048 รันโปรแกรมในหน่วยความจำโปรแกรมที่ต่ออยู่ภายนอก</u> ส่วนการต่อ <u>SS ให้มีลอจิคหนึ่งนั้น</u> เพื่อให้ 8048 ทำงานในโหมดปกติ <u>ไม่ใช่โหมด SINGLE STEP</u>

สัญญาณจาก ดาต้าบัส D0-D7 จะถูกต่อผ่าน 74LS373 ซึ่งมันจะใช้สัญญาณ ALE มาเป็น STROBE ในการแลทช์ค่าแอดเดรส A0-A7 เอาไว้ เพื่อใช้อ้างถึงตำแหน่งของหน่วยความจำภายนอกได้ โดยการต่อเอาท์พุทจาก 74LS373 เข้ากับ ขาแอดเดรส A0-A7 ของ EPROM 2716 และ RAM 6116

ส่วนขาแอดเดรส 3 บิทสูงของหน่วยความจำ A8-A10 จะต่ออยู่กับขาสัญญาณ P20-P22 ของ 8048

จะสังเกตุเห็นว่า ขา CE (CHIP ENABLE) ของ EPROM 2716 นั้น ต่ออยู่กับขา PSEN นั่นคือ EPROM 2716 นี้จะทำงานได้เมื่อ 8048 เฟทช์คำสั่งจากหน่วยความจำโปรแกรมภายนอก หรือ กล่าวอีกนัยหนึ่งก็คือ EPROM 2716 นี้เป็นหน่วยความจำโปรแกรมภายนอกนั่นเอง



รูปที่ 3.3 แสดงวงจรของระบบควบคุม

ในส่วนของขาสัญญาณ P23 นั้นต่อลงกับขา CE และ ต่อขาสัญญาณ WR เข้ากับ ขา R/W ของ RAM 6116 เพื่อใช้ RAM 6116 นี้เป็นหน่วยความจำข้อมูลภายนอก การจะเขียน หรือ อ่านข้อมูลกับ หน่วยความจำส่วนนี้ จำเป็นต้องส่งลอจิคหนึ่งออกมาที่ขา P23 เสียก่อน เพื่อดึงนาเบิ้ล RAM ให้ทำงาน จากนั้นก็สามารถจะใช้คำสั่ง MOVX เพื่อเคลื่อนย้ายข้อมูลกับ RAM ได้แล้ว และ การเปลี่ยนข้อมูลที่เอาท์พุทพอร์ท 2 (P20-P22) ก็เท่ากับเป็นการเปลี่ยนแปลง แอดเดรส 3 บิทสูง (A8-A10) ของ RAM ด้วย เป็นการขยายหน่วยความจำข้อมูลภายนอก จากเดิม 256 ไบท์ ให้เป็น 2 KBYTE

แต่ถ้าหากเราให้ลอจิคศูนย์ ออกมาทางขา P23 และ เขียนข้อมูลออกมาด้วยคำสั่ง MOVX ข้อมูลนั้นก็จะถูกแลทช์ค้างไว้โดย 74LS374 ซึ่งเราใช้ส่วนนี้ เพื่อส่งข้อมูลจาก 8048 ไปให้แก่เครื่อง IBM PC/XT

ส่วนอื่นที่เหลือกันได้แก่ พอร์ท 1 (P10-P17), พอร์ท 2 (P24-P27), ขาอินพุททดสอบ T0, T1 และ INT จะเชื่อมต่อลงอยู่กับวงจรส่วนหน้า ที่ติดต่อกับโทรศัพท์ หน้าที่ของสัญญาณต่างๆเหล่านี้ และวิธีการใช้งาน จะสรุปไว้ข้างล่างดังนี้

ขาสัญญาณ P10-P12 ต่อลงอยู่กับ ขา A0-A2 ของไอซี 74LS259 ซึ่งทำหน้าที่เป็นวงจรถอดรหัส โดยจะส่งลอจิคหนึ่งออกมาทางเอาท์พุทที่กำหนดโดยขา A0-A2 เพื่อเลือกคู่สายที่จะตรวจสอบสถานะ

ขาสัญญาณ P16 ให้ควบคุมสัญญาณตอบรับ ให้เงียบ หรือให้ดังขึ้นมา โดยเมื่อต้องการให้ส่งสัญญาณตอบรับกลับไป จะต้องให้ลอจิคศูนย์ออกมาที่ขา P16 และเมื่อให้ลอจิคหนึ่งที่ขานี้ ก็จะทำให้สัญญาณตอบรับเงียบลงไป ตัวโปรแกรมที่ทำหน้าที่ควบคุม จะกำหนดให้สัญญาณตอบรับ ดังอยู่นานประมาณ 3 วินาที แล้วเงียบหายไป

ขาสัญญาณ P17 ใช้ควบคุมการรับสาย และ วางสายโทรศัพท์ ทำได้โดยเลือกคู่สายที่จะรับ และส่งสัญญาณ ออกมาทางขา P17 ถ้าหากต้องการให้รับสาย ก็จะส่งลอจิคศูนย์ออกมา ในทางตรงกันข้าม เมื่อต้องการวางสาย ก็จะต้องส่งลอจิคหนึ่ง ออกมาที่ขา P17 นี้แทน

ขาสัญญาณ T0 ต่อลงอยู่กับวงจรตรวจสอบสัญญาณกระดิ่ง เมื่อตรวจสอบลอจิคที่ขา T0 พบว่า มีลอจิคหนึ่ง ก็แสดงว่าขณะนี้มีคนโทรศัพท์เข้ามาแล้ว และสัญญาณกระดิ่งยังดังอยู่ จะต้องรอจนกว่าลอจิคหนึ่งที่ขา T0 นี้หมดลง จึงจะสามารถรับสายได้

ขาสัญญาณ T1 ต่อลงอยู่กับขา SID ของไอซี 8870 การตรวจสอบสัญญาณ SID

โดยดูจากลอจิคที่ขา T1 จะทำให้รู้ได้ว่ามีการส่งสัญญาณ DTMF เข้ามาหรือไม่ หรือ สัญญาณ DTMF เดิมหมดลงหรือยัง

พอร์ท 2 (P24-P27) ต่อกับขา Q1-Q4 ของไอซี 8870 ตามลำดับ เมื่อโปรแกรมตรวจสอบพบว่ามีสัญญาณ STD เกิดขึ้นแล้ว ก็สามารถจะอ่านข้อมูลได้จากพอร์ท 2 P24-P27 นี้

ขาสัญญาณ INT ต่อคู่กับวงจรควบคุมลอจิคระหว่างเครื่องไมโคร IBM PC/XT โดยใช้สัญญาณนี้เป็น STROBE ในการส่งข้อมูลจาก 8048 ไปยังเครื่อง IBM PC/XT แบบ HANDSHAKING กล่าวคือ เมื่อ 8048 รับข้อมูลจากโทรศัพท์เรียบร้อยแล้ว ก็จะทำให้ขา INT มีลอจิคหนึ่ง เป็นการส่งสัญญาณไปบอกเครื่องไมโคร IBM PC/XT เมื่อเครื่องไมโคร IBM PC/XT อ่านข้อมูลไปแล้ว วงจรควบคุมลอจิคจะทำให้ขา INT แอคทีฟ และเป็นการส่งสัญญาณบอกให้ 8048 ส่งข้อมูลใหม่ไบท์ต่อไปออกมา สรุปแล้ว ทั้ง 8048 และเครื่องไมโคร IBM PC/XT ต่างจะต้องตรวจสอบความพร้อมในการรับส่งข้อมูล ซึ่งกันและกันเสียก่อน โดยในส่วนของ 8048 จะตรวจสอบลอจิคที่ขา INT นี้

รายละเอียดในการออกแบบโปรแกรม

ตามที่ได้ทราบถึงหลักการทำงานของโปรแกรมคร่าวๆแล้ว ในการออกแบบโปรแกรมจริงๆนั้น เราจะต้องมีการกำหนดรายละเอียดต่างๆลงไปด้วย อันได้แก่

1. พื้นที่หน่วยความจำข้อมูลภายใน ที่ใช้งานในโปรแกรม

เราใช้หน่วยความจำข้อมูลภายใน ตั้งแต่แอดเดรส 20-2F เพื่อใช้เป็นตัวนับลูปในการหน่วงเวลา 10 วินาที โดยแต่ละคู่สายจะใช้หน่วยความจำ 2 ไบท์สำหรับการนับลูป 65536 (10000_{16} / FFFF) ครั้ง สาเหตุที่ใช้หน่วยความจำข้อมูลภายใน เนื่องจาก มีคำสั่งสำหรับเพิ่มค่าข้อมูล ในหน่วยความจำนี้โดยตรง ไม่ต้องดำเนินการผ่านแอคคิวมูเลเตอร์

หน่วยความจำแอดเดรสใด ใช้กับคู่สายใดนั้น กำหนดไว้ดังนี้

แอดเดรส 20,21 เป็นตัวนับลูปของคู่สาย 0

แอดเดรส 22,23 เป็นตัวนับลูปของคู่สาย 1

แอดเดรส 24,25 เป็นตัวนับลูปของคู่สาย 2

แอดเดรส 26,27 เป็นตัวนับลูปของคู่สาย 3

แอดเดรส 28,29 เป็นตัวนับลูปของคู่สาย 4

แอดเดรส 2A,2B เป็นตัวนับลูปของคู่สาย 5

แอดเดรส 2C, 2D เป็นตัวนับลูปของคู่สาย 6

แอดเดรส 2E, 2F เป็นตัวนับลูปของคู่สาย 7

นอกจากนี้เรายังใช้หน่วยความจำแอดเดรส 30-37 ไว้เก็บ STATUS ของแต่ละคู่สาย และใช้หน่วยความจำแอดเดรส 38-3F สำหรับเก็บค่าของ POINTER สำหรับชี้ไปยังแอดเดรสของหน่วยความจำภายนอก ที่ใช้เก็บข้อมูลจากโทรศัพท์

2. พื้นที่หน่วยความจำข้อมูลภายนอก ที่ใช้งานในโปรแกรม

เราใช้หน่วยความจำส่วนนี้ สำหรับเก็บข้อมูลที่ได้รับจากโทรศัพท์ ซึ่งได้แก่ หมายเลขของ PAGER และหมายเลขโทรศัพท์ที่โทรเข้ามาต่อไป

MEMORY MAP ของหน่วยความจำข้อมูลภายนอก เป็นดังนี้

แอดเดรส 10-27 เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 0

แอดเดรส 28-3F เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 1

แอดเดรส 40-57 เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 2

แอดเดรส 58-6F เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 3

แอดเดรส 70-87 เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 4

แอดเดรส 88-9F เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 5

แอดเดรส A0-B7 เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 6

แอดเดรส B8-CF เป็นบัฟเฟอร์ที่เก็บข้อมูลของคู่สาย 7

3. พื้นที่หน่วยความจำโปรแกรมภายนอก ที่ใช้งานในโปรแกรม

<u>แอดเดรส 000</u> เป็นแอดเดรสเริ่มต้นของโปรแกรม เมื่อ 8048 ได้รับการรีเซท จะเริ่มทำงานที่นี่ ณ. ตำแหน่งนี้ จะมีคำสั่งกระโดดไปยังแอดเดรส 100 อีกต่อหนึ่ง

<u>แอดเดรส 003</u> เป็นแอดเดรสเริ่มต้นของ INTERUPT SERVICE ROUTINE กล่าวคือ เมื่อมีการอินเตอร์รัพท์จากภายนอก มาทางขา INT จะทำให้โปรแกรมหยุดชั่วคราว และกระโดดมาทำโปรแกรม ที่แอดเดรส 003 ก่อน เมื่อเสร็จแล้วจึงจะกลับไปทำงานเดิมต่อไป

8048 จะถูกอินเตอร์รัพท์เมื่อมีข้อมูลอยู่บัฟเฟอร์เต็มแล้ว และเมื่อเครื่องคอมพิวเตอร์ IBM PC/XT ต้องการล่านข้อมูลจาก 8048 ดังนั้น ใน INTERUPT SERVICE ROUTINE จึงต้องมีคำสั่งสำหรับเคลื่อนย้ายข้อมูลระหว่าง 8048 กับเครื่อง IBM อยู่ด้วย

<u>แอดเดรส 007</u> เป็นแอดเดรสเริ่มต้นของ TIMER INTERUPT SERVICE

ROUTINE นั่นคือ เมื่อ วงจรนับถึง 00 แล้ว จะส่งสัญญาณไปอินเตอร์รัพท์ซีพียู ซีพียูจะกระโดดมาทำงานตามโปรแกรมที่แอดเดรสนี้

เราใช้วงจรนับนี้ ร่วมกับการนับลูปทอง รีจีสเตอร์ R6 สำหรับตั้งเวลาให้สัญญาณตอบรับดังคงอยู่นาน 3 วินาที ตามปกติแล้ววงจรนับสามารถตั้งเวลาได้มากที่สุดเพียง 20 mS ดังนั้น ถ้าหากเราใช้ R6 ในการนับจำนวนครั้งที่ถูกอินเตอร์รัพท์ เราจะต้องนับลูปถึง 200 ครั้ง จึงจะได้เวลานาน 3 วินาที

แอดเดรส 100 เป็นต้นแหน่งเริ่มต้นของโปรแกรมควบคุมนี้

4. ความหมายของปุ่ม # และ * ในการป้อนข้อมูลผ่านทางโทรศัพท์นั้น กำหนดให้สามารถใช้ปุ่ม * เพื่อยกเลิกข้อมูลเดิมที่ป้อนเข้าไปแล้ว โดยเมื่อกดปุ่ม * แล้วจะมีสัญญาณตอบรับกลับมา จึงเริ่มป้อนข้อมูลใหม่อีกครั้ง

ในการป้อนข้อมูลนั้นจะต้องป้อนหมายเลขพจเจอร์ที่ต้องการติดต่อเข้าไปก่อน แล้วกดปุ่ม # จากนั้นจึงใส่รหัสทางไกล แล้วกดปุ่ม # อีกจากนั้นจึงกดหมายเลขโทรศัพท์เข้าไป และจบด้วยปุ่ม # อีกเช่นกัน นั่นคือ เราใช้ปุ่ม # เป็นสัญญาณเพื่อแยกข้อมูลแต่ละอย่างออกจากกัน ในกรณีที่ป้อนข้อมูลเพียงสองอย่าง โปรแกรมจะถือว่าข้อมูลแรกเป็นหมายเลขเพจเจอร์ และข้อมูลที่สองเป็นหมายเลขโทรศัพท์

5. ในการส่งข้อมูลจากระบบไมโครคอมพิวเตอร์ 8048 ไปยังเครื่องไมโครคอมพิวเตอร์ IBM PC/XT จะส่งข้อมูลเป็นชุดๆติดต่อกัน โดยข้อมูลแต่ละชุดนั้นจะประกอบด้วยหมายเลขเพจเจอร์, ตัวกั้นที่มีค่าเป็น &H12, รหัสทางไกล, ตัวกั้นที่มีค่าเป็น &H14, หมายเลขโทรศัพท์ และตัวปิดท้ายข้อมูลแต่ละชุดที่มีค่าเป็น &H16

เมื่อข้อมูลทั้งหมดถูกส่งออกไปแล้ว 8048 ก็จะส่งรหัสหยุดออกมา ซึ่งมีค่าเป็น &HFF เป็นการจบกระบวนการรับ-ส่งข้อมูล

<u>การออกแบบโปรแกรม</u>

โปรแกรมทั้งหมดจะถูกแบ่งออกเป็นส่วนๆตาม STATUS หรือขั้นตอนการทำงาน โดยมีโปรแกรมหลักเป็นตัวควบคุม

ตัวโปรแกรมหลักจะมีลักษณะดังนี้

```
MAIN        MOV     A,#0C0H
            OUTL    P1,A
            MOV     A,#0F8H
            OUTL    P2,A
            MOV     R2,#FF
            MOV     R0,#37
            MOV     R7,#08
LOOP        CLR     A
            MOV     @R0,A
            CALL    INITPT
            CALL    INTIME
            DEC     R0
            DJNZ    R7,LOOP
            DIS     I
            EN      TCNTI
```

ส่วนต้นของโปรแกรมหลักนี้ จะเตรียมตัวสำหรับเริ่มทำงาน เช่นตั้งค่า STATUS เริ่มต้นให้เป็นศูนย์ , เซทค่ารีจีสเตอร์ต่างๆ , อีนาเบิ้ลอินเตอร์รัพท์ , กำหนดค่าเริ่มต้นของ POINTER ให้ชี้ไปที่แอดเดรสเริ่มต้นที่เก็บข้อมูล , เคลียร์ค่าตัวนับลูปเวลา 10 วินาที เป็นต้น

ในส่วนที่เหลือของโปรแกรมหลักนี้ ทำหน้าที่ตรวจค่า STATUS ของแต่ละคู่สาย แล้วกระโดดไปทำงานตาม STATUS นั้นๆ โดยการกระโดดของโปรแกรมนั้นใช้หลักการของ JUMP TABLE ซึ่งตารางกระโดดนี้เริ่มที่แอดเดรส OFFSET

```
MAIN2       MOV     R0,#30
MAIN3       MOV     A,@R0
            RL      A
            RL      A
            ADD     A,#OFFSET
```

```
                JMPP    @A
OFFSET          CALL    STAT0
                JMP     PASS
                CALL    STAT1
                JMP     PASS
                CALL    STAT2
                JMP     PASS
                CALL    STAT3
                JMP     PASS
                CALL    STAT4
                JMP     PASS
                CALL    STAT3
                JMP     PASS
                CALL    STAT4
                JMP     PASS
                CALL    STAT3
                JMP     PASS
                CALL    STAT4
                JMP     PASS
                CALL    STAT9
                JMP     PASS
                CALL    STAT10
                JMP     PASS
                CALL    STAT11
                JMP     PASS
                CALL    STAT12
                JMP     PASS
PASS            INC     R0
```

```
        MOV     A,R0
        XRL     A,#38
        JNZ     MAIN3
        JZ      MAIN2
```

เมื่อโปรแกรมกระโดดมาที่ STAT0 ซึ่งเป็น STATUS เริ่มต้นของแต่ละคู่สาย โดยจะส่งสัญญาณไปยังวงจรส่วนหน้าที่ต้องการติดต่อด้วย แล้วตรวจสอบสัญญาณกระดิ่งทางขา T0 ถ้าหากไม่มีสัญญาณกระดิ่งเข้ามา ก็จะออกไปตรวจสอบคู่สายถัดไป

แต่เมื่อมีสัญญาณกระดิ่งเข้ามา (มีลอจิคหนึ่งที่ขา T0) ก็จะเปลี่ยน STATUS เป็นหนึ่งทันที

```
STAT0   MOV     A,#C0H
        CALL    OUTP1
        JNT0    ENDS0
        INC     @R0
ENDS0   RET
```

เมื่อมีสัญญาณกระดิ่งเข้ามาแล้ว การทำงานใน STAT1 จะตรวจสอบดูว่าสัญญาณกระดิ่งเงียบลงหรือยัง และ วงจรส่งสัญญาณตอบรับว่างอยู่หรือเปล่า ถ้าหากยังไม่พร้อม ก็จะยังไม่รับสาย และค่า STATUS ยังคงเป็นหนึ่งอยู่

แต่เมื่อสัญญาณกระดิ่งเงียบลงแล้ว และวงจรตอบรับว่างแล้ว ก็จะสั่งให้รีเลย์ต่อวงจรเป็นการรับสายโทรศัพท์ แล้วเพิ่มค่า STATUS เป็นสอง

การตรวจสอบว่าวงจรส่งสัญญาณตอบรับว่างหรือไม่นั้น ดูได้จากค่ารีจีสเตอร์ R2 ถ้าค่าใน R2 เป็น FF แสดงว่าว่างอยู่ แต่ถ้าเป็นค่าตัวเลขอื่น ก็จะเป็นหมายเลขของคู่สายที่กำลังส่งสัญญาณตอบรับอยู่ จากลักษณะนี้จะเห็นว่าโปรแกรมจะควบคุมให้มีการตอบรับได้ครั้งละหนึ่งคู่สายเท่านั้น

```
STAT1   MOV     A,#C0H
        CALL    OUTP1
        JT0     ENDS1
        MOV     A,R2
        INC     A
```

```
        JNZ     ENDS1
        MOV     A,#40H
        CALL    OUTP1
        INC     @R0
        MOV     A,R0
        ANL     A,#07
        MOV     R2,A
        MOV     R6,#TC1
        CALL    CLOCK
ENDS1   RET
```

ในขณะที่กำลังส่งสัญญาณตอบรับ (STATUS=2) อยู่นั้น เป็นหน้าที่ของวงจรตั้งเวลาภายใน 8048 ทำหน้าที่ควบคุมให้สัญญาณตอบรับดังนานประมาณ 3 วินาที แล้วจะเงียบไป รวมทั้งจะเพิ่มค่า STATUS ให้เป็นสามด้วย ดังนั้นในส่วนของโปรแกรม STAT2 จึงมีแต่คำสั่ง RET เพียงคำสั่งเดียว

```
STAT2   RET
```

ส่วนที่โปรแกรม TIMER INTERUPT SERVICE RUOTINE จะมีโปรแกรมต่อไปนี้ ทำหน้าที่จับเวลาให้สัญญาณตอบรับดังนาน 3 วินาที

```
TCNTI   DJNZ    R6,ENDCT1
        MOV     R4,A
        MOV     A,R2
        JB7     QUIET
        ORL     A,#80
        MOV     R2,A
        MOV     R6,#TC2
        CALL    CLOCK
        CALL    OUTP1
        MOV     A,R4
        REITR
```

```
QUIET   MOV   A,#40
        CALL  OUTP1
        INC   @R0
        MOV   R2,#FF
        STOP  T
        MOV   A,R4
        RETR
```

เมื่อค่า STATUS เป็นสามแล้ว โปรแกรมจะเตรียมตัวรับข้อมูลทันที โดยจะตรวจสอบสัญญาณ STD ที่ขา T1 ในระหว่างที่รอรับข้อมูลอยู่นั้น โปรแกรมจะตรวจสอบเวลาอยู่เสมอ ถ้าหากโปรแกรมรอรับข้อมูลอยู่นานเกินกว่า 10 วินาที โปรแกรมจะถือว่าผู้ใช้วางสายโทรศัพท์ไปแล้ว ส่วนของโปรแกรมที่ตรวจสอบเวลาคือส่วนโปรแกรม WAIT

แต่ถ้าหากมีสัญญาณ STD เข้ามาก่อนเวลา 10 วินาที โปรแกรมย่อย INTIME จะเคลียร์ค่าตัวหน่วงเวลาให้เริ่มจับเวลาใหม่ แล้วอ่านข้อมูลเข้ามาทางพอร์ท 2 P24-P27

ข้อมูลที่ได้มานี้จะถูกตรวจสอบว่าเป็นปุ่ม * โปรแกรมจะยกเลิกข้อมูลเก่าแล้วรอรับข้อมูลใหม่ โดยเปลี่ยนค่า STATUS ให้เป็นสาม แต่ถ้าหากผู้ใช้กดปุ่ม # โปรแกรมก็จะใส่ตัวกั้นลงในข้อมูล แล้วเพิ่มค่า STATUS อีกสาม

เมื่อค่า STATUS เพิ่มจากสามเป็นหก ก็หมายความว่าผู้ใช้ได้ใส่หมายเลขเพจเจอร์ไปแล้ว กำลังจะป้อนรหัสทางไกล และหมายเลขโทรศัพท์ต่อไป โปรแกรมใน STATUS 5 และ 7 นั้นจะมีหลักการทำงานเหมือนกัน ต่างกันตรงที่ เมื่อผู้ใช้กดปุ่ม # ตามท้ายข้อมูล โปรแกรมจะต้องใส่ตัวกั้นด้วย ค่าของตัวกั้นนี้จะแยกข้อมูลแต่ละส่วนออกจากกัน และบอกให้รู้ว่าตัวกั้นนี้กั้นข้อมูลอะไรอยู่ เช่น ตัวกั้นค่า 12 จะกั้นอยู่หลังหมายเลขเพจเจอร์, ตัวกั้นค่า 14 จะกั้นอยู่หลังรหัสทางไกล และ ตัวกั้นค่า 16 จะกั้นอยู่หลังหมายเลขโทรศัพท์ เราสามารถดัดแปลงโปรแกรม STAT3 ให้สามารถใช้ได้กับ STATUS 5 และ 7 ด้วย

```
STAT3   MOV   A,#40H
        CALL  OUTP1
        JNT1  WAIT
```

```
        CALL    INITME
        ORL     P2,#E0
        IN      A,P2
        SWAP    A
        ANL     A,#0F
        MOV     R3,A
        XRL     A,#RESET
        JZ      RESTART
        MOV     A,R3
        XRL     A,#ENTER
        JZ      SKIP
        INC     @R0
STORE   CALL    INCPT
        MOV     A,R3
        MOVX    @R1,A
        MOV     A,#1F
        ANL     A,R1
        XRL     A,#12
        JZ      OUTRNG
        RET
WAIT    MOV     A,R0
        ANL     A,#1F
        RL      A
        MOV     R1,A
        INC     @R1
        MOV     A,@R1
        JNZ     ENDS3
        INC     R1
```

```
                INC     @R1
                MOV     A,@R1
                JNZ     ENDS3
                MOV     A,@R0
                ADD     A,#FB
                JNC     OUTRNG
                MOV     @R0,#07
                JMP     SKIP
OUTRNG          CLR     A
                MOV     @R0,A   — clear ค่า status
                CALL    INITPT
ENDS3           RET
RESTART         CALL    INITPT
                MOV     @R0,#04
                RET
SKIP            MOV     A,@R0
                ADD     A,#0F
                MOV     R3,A
                INC     @R0
                INC     @R0
                INC     @R0
                JMP     STORE
```

ถ้าหากข้อมูลที่ป้อนเข้ามาเป็นตัวเลข 0-9 ข้อมูลเหล่านั้นก็จะถูกเก็บลงในหน่วยความจำที่ POINTER ชี้อยู่ แล้วค่า STATUS จะเพิ่มขึ้นอีกหนึ่ง STATUS กลายเป็น 4 และที่ STAT4 นี้ก็จะรอให้ผู้ใช้ปล่อยมือออกจากปุ่มที่กด เพื่อป้องกันไม่ให้โปรแกรมกลับไปอ่านข้อมูลเดิมซ้ำอีก เมื่อผู้ใช้ปล่อยมือแล้ว สัญญาณ STD ที่ขา T1 ก็จะหมดลงด้วย โปรแกรมก็จะลดค่า STATUS ลงหนึ่งกลายเป็น STATUS3 เช่นเดิม

ส่วนของโปรแกรม STAT4 นี้สามารถใช้ได้กับ STATUS 6 และ 8 ด้วย

```
STAT4       MOV   A,#40
            CALL  OUTP1
            JT1   ENDS4
            MOV   A,@R0
            DEC   A
            MOV   @R0,A
ENDS4       RET
```

เมื่อผู้ใช้ป้อนข้อมูลเข้ามาครบหมดแล้ว ค่า STATUS จะเปลี่ยนเป็น 10 โปรแกรม STAT10 ก็จะสั่งให้วางสายโทรศัพท์ และส่งข้อมูลค่า FE ออกไปให้เครื่อง IBM ทางพอร์ท 74LS373 ข้อมูลที่ส่งออกไปนี้จะเซทขา INT ของ 8048 ให้เป็นลอจิคหนึ่งขณะเดียวกันก็เป็นการบอกให้เครื่อง IBM สามารถมาอ่านข้อมูลได้แล้ว

```
STAT10      MOV   A,#40
            CALL  OUTP1
            JT1   ENDS10
            MOV   A,#C0
            CALL  OUTP1
            CALL  INITPT
            INC   @R0
            MOV   A,#FE
            CALL  OUTIBM
            EN    I
ENDS10      RET
```

ในขณะที่โปรแกรมทำงานถึง STAT11 โปรแกรมจะยังไม่สามารถใช้โทรศัพท์คู่สายนั้น ในการรับข้อมูลได้อีก เพราะจะทำให้ข้อมูลเดิมหายไป จะต้องรอจนกระทั่งเครื่อง IBM มาอ่านข้อมูลไปหมดแล้ว

```
STAT11      RET
```

เมื่อเครื่อง IBM มาอ่านข้อมูลที่พอร์ทไปแล้ว จะทำให้เกิดสัญญาณอินเตอร์รัพท์ที่ 8048 ที่โปรแกรม INTERUPT SERVICE RUOTINE นี้จะเป็นโปรแกรมสำหรับส่ง

ข้อมูล โดยจะมีการตรวจสอบความพร้อมในการรับส่งข้อมูลของทั้งสองฝ่ายอยู่เสมอ ในการส่งข้อมูลแต่ละไบท์

โปรแกรมจะดูจาก STATUS ของแต่ละคู่สาย คู่สายใดที่มี STATUS เป็น 11 ก็แสดงว่ามีข้อมูลพร้อมแล้ว 8048 ก็จะนำข้อมูลนั้นส่งออกไป แล้วเปลี่ยน STATUS ให้เป็นศูนย์ เพื่อให้โทรศัพท์คู่สายนั้นพร้อมที่จะทำงานใหม่ได้ เมื่อส่งข้อมูลทั้งหมดแล้ว 8048 ก็จะส่งตัวเลข FF ออกไปเป็นการจบการส่งข้อมูล

```
INIT1           SEL.    RB1
                MOV     R7, A
                MOV     R0, #30
NEXTLN          MOV     A,@R0
                XRL     A,#11
                JNZ     NOTRDY
                CALL    INITPT
                MOV     R1,A
NEXTCH          MOVX    A,@R1
                CALL    OUTIBM
WAITI           JNI     CONT
                JMP     WAITI
CONT            INC     R1
                XRL     A,#16
                JNZ     NEXTCH
                MOV     @R0,#00
NOTRDY          INC     R0
                MOV     A,#38
                XRL     A,R0
                JNZ     NEXTLN
                DIS     I
                MOV     A,#FF
```

```
        CALL   OUTIBM
        MOV    A,R7
        RETR
```

นอกจากโปรแกรมหลักแล้ว ยังมีโปรแกรมย่อยอีกหลายๆโปรแกรมด้วยได้แก่

โปรแกรม OUTP1 เป็นโปรแกรมส่งข้อมูลออกไปทางพอร์ท P1 ข้อมูลที่ส่งออกไปจะเป็นรหัสควบคุมให้วางสาย รับสาย ควบคุมสัญญาณตอบรับ เป็นต้น

```
OUTP1   ORL    A,R0
        ANL    A,#07
        OUTL   P1,A
        RET
```

โปรแกรม CLOCK ทำหน้าที่ตั้งโปรแกรมให้วงจรนับภายใน ตั้งเวลาให้นานที่สุด

```
CLOCK   STOP   T
        CLR    A
        MOV    T,A
        STRT   T
        RET
```

โปรแกรม INITPT ทำหน้าที่กำหนดค่าเริ่มต้นให้กับ POINTER ให้ชี้ที่แอดเดรสเริ่มต้นของบัฟเฟอร์เก็บข้อมูล

```
INITPT  MOV    A,R0
        ORL    A,#08
        MOV    R1,A
        ORL    A,#F0
        MOVP   A,@A
        MOV    @R1,A
        RET
```

โปรแกรม OUTIBM ทำหน้าที่ส่งข้อมูลจาก 8048 ไปยังเครื่อง IBM

```
OUTIBM    ANL   P2,#F7
          MOVX  @R1,A
          ORL   P2,#08
          RET
```

โปรแกรม INTIME ทำหน้าที่เคลียร์ตัวนับเวลา 10 วินาที เมื่อผู้ใช้ป้อนข้อมูลที่ถูกต้องภายใน 10 วินาที

```
INTIME    MOV   A,R0
          ANL   A,#11
          RL    A
          MOV   R1,A
          CLR   A
          MOV   @R1,A
          INC   R1
          MOV   @R1,A
          RET
```

โปรแกรม INCPT ทำหน้าที่เพิ่มค่าของ POINTER ให้ชี้ไปที่แอดเดรสถัดไปสำหรับข้อมูลไบท์ใหม่ที่จะป้อนเข้ามา

```
INCPT     MOV   A,R0
          ORL   A,#08
          MOV   R1,A
          MOV   A,@R1
          INC   @R1
          MOV   R1,A
          RET
```

3.1.5 การออกแบบพอร์ทสำหรับเครื่องไมโครคอมพิวเตอร์ IBM PC/XT

ตามที่กล่าวถึงมาแล้วในตอนต้นว่า การจัดการเกี่ยวกับข้อมูลต่างๆ ที่ผู้ใช้โทรศัพท์ส่งมาให้นั้น จะเป็นหน้าที่หลักของเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ไม่ว่าจะเป็นการเก็บข้อมูลการใช้ PAGER ,รายละเอียดเกี่ยวกับผู้ถือ PAGER เป็นต้น ซึ่งหน้าที่ต่างๆเหล่านี้จะถูกกล่าวถึงโดยละเอียดอีกครั้งในตอนต่อไป

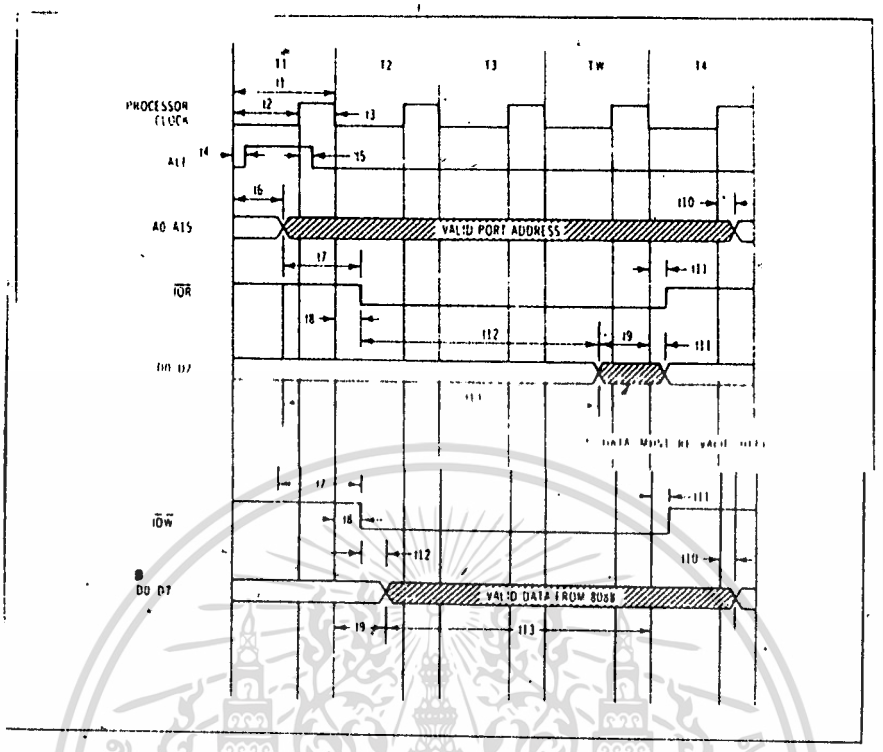
ไอซี 8088 เป็นซีพียูที่ใช้ในเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ลักษณะโดยทั่วไปของซีพียูจะมีดังนี้ ในเรื่องการติดต่อกับพอร์ทนั้นจะมีลักษณะดังนี้ คือ ซีพียูจะใช้ดาต้าบัสเป็นเส้นทางเดียวกัน ทั้งข้อมูลที่จะส่งไปยังพอร์ท หรือ อ่านข้อมูลจากพอร์ท และ จะต้องมีสัญญาณที่แสดงแอดเดรสบัส เป็นตัวบ่งชี้หมายเลขพอร์ทที่ซีพียูต้องการติดต่อ และ จะต้องมีสัญญาณอีกหนึ่งชุดสำหรับควบคุมการอ่านหรือเขียนข้อมูลกับพอร์ท

สำหรับเครื่อง IBM แล้ว จะรับ-ส่งข้อมูลผ่านทางดาต้าบัสขนาด 8 บิท และใช้แอดเดรสบัสเพียง 10 เส้น(จากทั้งหมด 16 เส้น) คือ A0-A9 เพื่อใช้ในการกำหนดหมายเลขพอร์ท สำหรับสัญญาณควบคุมที่ใช้ในการติดต่อกับพอร์ท มีอยู่ 3 เส้นด้วยกัน คือ IOR ,IOW และ AEN

สัญญาณ IOR (I/O READ) และ IOW (I/O WRITE) เป็นสัญญาณเอาท์พุทที่แอคทีฟศูนย์ และจะแอคทีฟเมื่อ มีการอ่าน หรือ เขียนข้อมูลกับพอร์ท ดังไดอะแกรมในรูปที่ 3.4

สำหรับสัญญาณ AEN (ADDRESS ENABLE) เป็นสัญญาณที่บอกให้รู้ว่าข้อมูลต่างๆบนบัสได้มาจากซีพียูใช่หรือไม่ ทั้งนี้เนื่องจากในระบบคอมพิวเตอร์ IBM นี้ มีชิพที่ทำหน้าที่เป็น DMA CONTROLLER อยู่ด้วย ซึ่ง DMA CONTROLLER นี้สามารถจะส่งสัญญาณออกมาทางแอดเดรสบัส ดาต้าบัสหรือบัสควบคุมได้ ดังนั้นเพื่อให้วงจรภายนอกสามารถรู้ได้ว่าขณะนั้นซีพียูหรือ DMA CONTROLLER กำลังทำงานอยู่บนบัส โดยดูจากลอจิคของสัญญาณ AEN

ในที่นี้เราต้องจำไว้ว่า เมื่อสัญญาณ AEN มีลอจิคศูนย์ แสดงว่า ซีพียูกำลังควบคุมระบบบัสต่างๆอยู่ เราสามารถจะติดต่อกับซีพียูผ่านทางพอร์ทได้

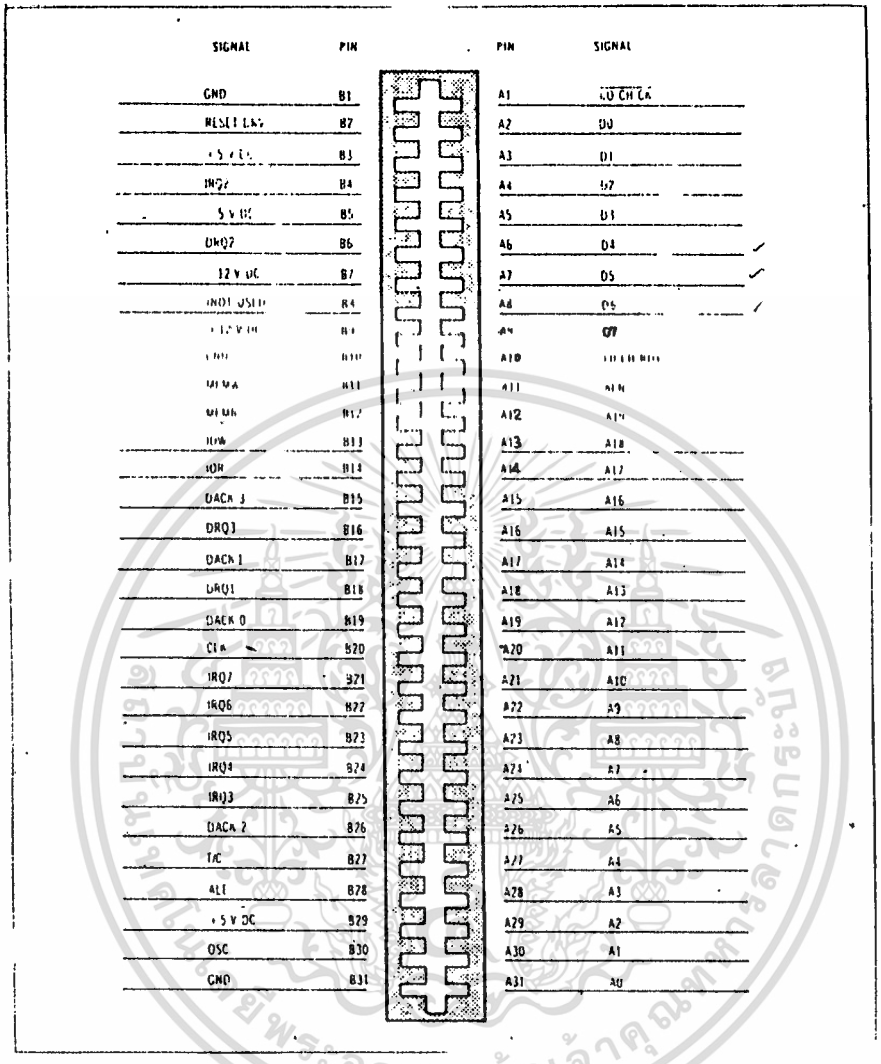


รูปที่ 3.4 แสดง TIMING DIAGRAM การเขียน อ่านข้อมูลกับพอร์ท

สายสัญญาณต่างๆที่ใช้ในการต่อพอร์ทนั้นได้ถูกเตรียมไว้ โดยเราสามารถนำสัญญาณเหล่านั้นมาใช้ได้ เพียงแต่เสียบแผ่นวงจรพิมพ์สองหน้า เข้ากับ CARD EDGED CONNECTOR หรือที่เรียกสั้นๆว่า สล็อต ตำแหน่งของสล็อตนั้น จะวางอยู่ภายในเครื่องทางด้านหลัง สัญญาณต่างๆที่มีอยู่บนสล็อตนั้น แสดงไว้ในรูปที่ 3.5 ซึ่งรวมสัญญาณต่างๆในการต่อพอร์ทเอาไว้ด้วย

สายสัญญาณแอดเดรสบัส ทำหน้าที่กำหนดหมายเลขพอร์ทว่า ซีพียูต้องการติดต่อกับพอร์ทใด ซึ่งในที่นี้สามารถใช้แอดเดรสบัส A0-A9 รวม 10 เส้น เท่ากับว่า เครื่อง IBM สามารถติดต่อกับพอร์ทต่างๆ ได้ถึง 1024 พอร์ท

หมายเลขทั้ง 1024 หมายเลขนี้แบ่งออกเป็น 2 ส่วน คือ พอร์ทหมายเลข 0-511 จะเป็นพอร์ทที่อยู่บนเมนบอร์ดของเครื่อง IBM ส่วนพอร์ทตั้งแต่หมายเลข 512-1023 นั้น เป็นพอร์ทที่ติดต่อกับซีพียูผ่านทางสล็อต

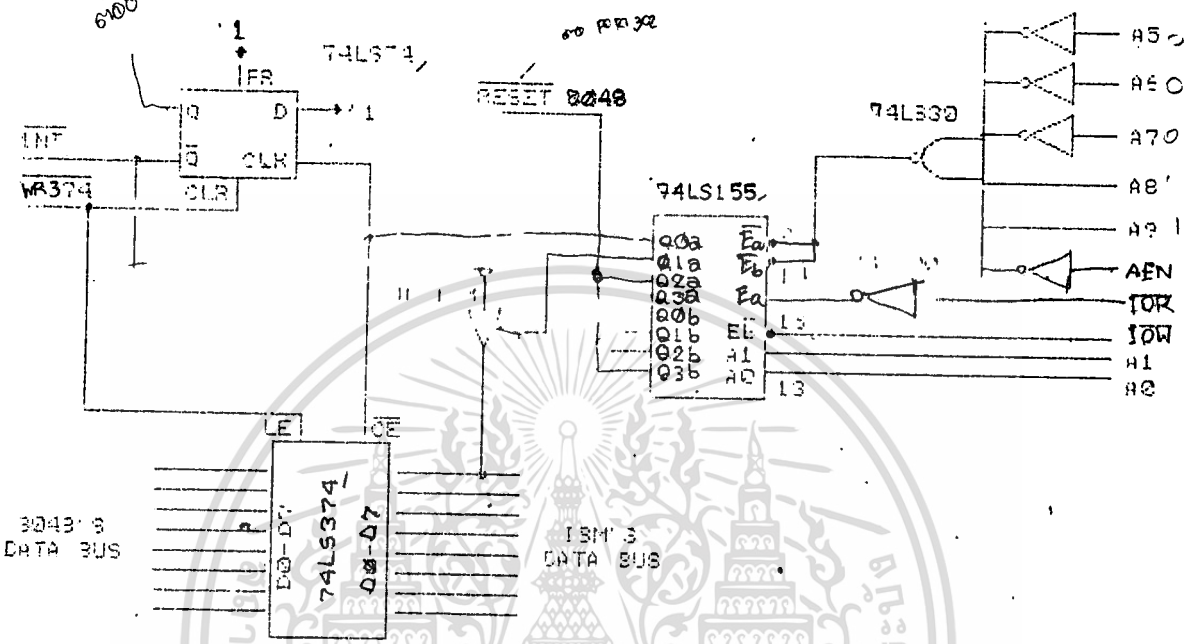


SIGNAL	PIN		PIN	SIGNAL
GND	B1		A1	I/O CH CK
RESET DRV	B2		A2	D7
+5 V DC	B3		A3	D6
IRQ2	B4		A4	D5
-5 V DC	B5		A5	D4
DRQ2	B6		A6	D3
-12 V DC	B7		A7	D2
(NOT USED)	B8		A8	D1
+12 V DC	B9		A9	D0
GND	B10		A10	I/O CH RDY
MEMW	B11		A11	AEN
MEMR	B12		A12	A19
IOW	B13		A13	A18
IOR	B14		A14	A17
DACK 3	B15		A15	A16
DRQ3	B16		A16	A15
DACK 1	B17		A17	A14
DRQ1	B18		A18	A13
DACK 0	B19		A19	A12
CLK	B20		A20	A11
IRQ7	B21		A21	A10
IRQ6	B22		A22	A9
IRQ5	B23		A23	A8
IRQ4	B24		A24	A7
IRQ3	B25		A25	A6
DACK 2	B26		A26	A5
TC	B27		A27	A4
ALE	B28		A28	A3
+5 V DC	B29		A29	A2
OSC	B30		A30	A1
GND	B31		A31	A0

รูปที่ 3.5 แสดงสัญญาณต่างๆบนสล็อต

เมื่อเราต่อพอร์ทกับเครื่อง IBM เราก็จำเป็นจะต้องกำหนดหมายเลขของพอร์ทเสียก่อน หลักในการกำหนดหมายเลขพอร์ทมีเพียง 2 ข้อหลัก คือ หมายเลขพอร์ทต้องอยู่ในช่วง 512-1023 และ หมายเลขพอร์ทนั้น ต้องไม่ไปซ้ำกับพอร์ทที่มีอยู่เดิม ในการออกแบบนี้ เลือกกำหนดหมายเลขพอร์ทเป็น 300-31F(ฐาน 16)

เมื่อเรากำหนดหมายเลขพอร์ทแล้ว ก็สามารถออกแบบวงจรส่วนอินเตอร์เฟสระหว่าง 8048 กับเครื่อง IBM ได้ดังรูปที่ 3.6



รูปที่ 3.6 แสดงวงจรพอร์ท

เมื่อเครื่อง IBM จะติดต่อกับพอร์ทหมายเลข 300-31F ก็จะทำให้เอาท์พุท ลง 74LS30 เป็นลอจิคศูนย์ และเมื่อสัญญาณ IOR หรือ IOW แอคทีฟ ก็จะทำให้ 74LS155 ทำงานได้ จะมีสัญญาณเอาท์พุทเป็นลอจิคศูนย์ไปปรากฏที่ขาเอาท์พุท Q0-Q3 ขาใดขาหนึ่ง โดยมีสายแอดเดรส A0,A1 เป็นตัวเลือก เช่น เมื่อเครื่อง IBM ต้องการอ่านข้อมูลจากพอร์ทหมายเลข 300 ก็จะมีลอจิคศูนย์ที่ขาเอาท์พุท Q0 ของ 74LS155 ส่วนขาเอาท์พุทอื่นๆจะเป็นลอจิคหนึ่ง

ในการทำงานของวงจรจริงๆนั้น จะมีการถ่ายเทข้อมูลจาก 8048 ผ่าน 74LS374 ไปยังดาต้าบัสของเครื่อง IBM เมื่อมีการส่งข้อมูลจากดาต้าบัสของ 8048 มาที่ 74LS374 ข้อมูลนั้นจะถูกแลทช์ด้วยสัญญาณ WR374 ทำให้มีข้อมูลปรากฏที่เอาท์พุทของ 74LS374 ขณะเดียวกัน สัญญาณ WR374 จะเคลียร์ฟลิปฟลอปให้เอาท์พุท Q เป็นลอจิคศูนย์ จึงไม่มีสัญญาณไปอินเตอร์รัพท์ 8048

สัญญาณเอาท์พุทของฟลิปฟลอปจะบอกให้รู้ว่า เครื่อง IBM อ่านข้อมูลไปจาก

พอร์ทหรือยัง ถ้าหากยังไม่มีการอ่านข้อมูลนั้น 8048 จะส่งข้อมูลตัวต่อมาไม่ได้ เพราะข้อมูลใหม่จะทับข้อมูลเก่าให้หายไป แต่เมื่อเครื่อง IBM อ่านข้อมูลไปแล้ว ก็จะทำให้เอาท์พุท Q ของฟลิปฟลอปเป็นลอจิคหนึ่ง เป็นการบอกให้ 8048 ส่งข้อมูลตัวใหม่ออกมาได้แล้ว

ทางฝ่ายเครื่อง IBM ก็เช่นกัน ก่อนที่จะมาอ่านข้อมูลที่พอร์ท 300 จะต้องอ่านข้อมูลที่พอร์ท 301 เสียก่อน แล้วดูที่บิท 7 ของข้อมูลว่าเอาท์พุทของฟลิปฟลอปมีลอจิคใด ถ้าหากเป็นลอจิคศูนย์ ก็แสดงว่าขณะนั้นข้อมูลใน 74LS374 เป็นข้อมูลเก่าที่อ่านไปแล้ว แต่ถ้าหากเอาท์พุทของฟลิปฟลอปเป็นลอจิคหนึ่ง แสดงว่า 8048 ได้ส่งข้อมูลตัวใหม่มาที่ 74LS374 แล้ว เครื่อง IBM สามารถอ่านข้อมูลนั้นได้จากพอร์ทหมายเลข 300

นอกจากพอร์ท 300, 301 แล้วยังมีพอร์ท 302 อีกพอร์ทหนึ่งซึ่งต่ออยู่กับขารีเซทของ 8048 ดังนั้น เมื่อมีการอ่านข้อมูลจากพอร์ท 302 นี้ก็จะเท่ากับเป็นการรีเซท 8048 นั่นเอง

ถ้าหากพิจารณาโปรแกรมของ 8048 ในตอนที่แล้ว ประกอบกับวงจรนี้ จะเห็นว่า โดยปกติ 8048 จะดิสเอเบิ้ลอินเตอร์รัพท์ จนเมื่อมันได้รับข้อมูลมาเก็บไว้ในบัฟเฟอร์เต็มแล้ว มันจะส่งค่า FF ออกมาที่ 74LS374 ค่า FF นี้ ไม่มีความหมายอะไร แต่ส่งออกมาเพื่อเคลียร์เอาท์พุทของฟลิปฟลอป หรือ ถอนสัญญาณอินเตอร์รัพท์ออกนั่นเอง แล้วจึงอีนาเบิ้ลอินเตอร์รัพท์

ทางฝ่ายเครื่อง IBM นั้น เมื่ออ่านค่าจากพอร์ท 301 แล้วพบว่า มีข้อมูลอยู่ใน 74LS374 แล้ว ก็จะมาอ่านข้อมูลที่พอร์ท 300 การอ่านครั้งนี้จะไม่สนใจข้อมูลที่อ่านแต่ทำไปเพื่ออินเตอร์รัพท์ 8048 เป็นการเริ่มต้นการอ่านข้อมูล

✳ เมื่อ 8048 ถูกอินเตอร์รัพท์ มันจะกระโดดไปรันโปรแกรม INTERUPT SERVICE ROUTINE ขณะที่อยู่ใน INTERUPT SERVICE ROUTINE นี้ 8048 จะดิสเอเบิ้ลอินเตอร์รัพท์โดยอัตโนมัติ ดังนั้น มันจึงสามารถใช้ขา INT เป็นขาอินพุทตอบธรรมดาได้ โดยตรวจสอบลอจิคที่ขา INT ถ้าหากเป็นลอจิคศูนย์ 8048 ก็จะส่งข้อมูลใหม่ออกไป แต่ถ้าเป็นลอจิคหนึ่งก็จะรอก่อน

เมื่ออ่านข้อมูลไปหมดแล้ว 8048 จะเอาท์พุทค่า FF ออกไปจากนั้นก็จะดิสเอเบิ้ลอินเตอร์รัพท์ แล้วออกจากโปรแกรม INTERUPT SERVICE ROUTINE

จากหลักการทำงานที่กล่าวมาข้างต้นนี้ จะเป็นตัวกำหนดการทำงานของโปรแกรมของเครื่อง IBM ในส่วนที่ทำหน้าที่ติดต่อกับพอร์ท ซึ่งจะกล่าวถึงในตอนต่อไป

3.2 การเขียนโปรแกรมควบคุมข้อมูล

ดังที่ได้กล่าวไว้ในหัวข้อ 2.4 ว่าภาษาที่เลือกใช้ในการเขียนโปรแกรมควบคุมข้อมูลควรจะเป็นภาษาโครงสร้างเพื่อให้การเขียนโปรแกรมขนาดใหญ่ทำได้ง่าย และจะต้องสามารถติดต่อกับหน่วยความจำ, พอร์ท และรีจิสเตอร์ได้สะดวก ในโครงงานนี้เลือกใช้ภาษา ซี(C Programing Language)ในการเขียนโปรแกรมควบคุมการทำงานบนเครื่องไมโครคอมพิวเตอร์ไอบีเอ็ม พีซี โดยใช้โปรแกรมตัวแปลภาษาเทอร์โบซี (Turbo C Complier) ของบริษัทบอร์แลนด์ (Borland International, Inc) การเขียนโปรแกรมนี้เขียนขึ้นเฉพาะโปรแกรมส่วนที่จำเป็นต่อการใช้งานจริงๆเท่านั้นคือ

1. โปรแกรมหลักทำหน้าที่ควบคุมการทำงานทั้งหมดและเรียกโปรแกรมย่อยอื่นๆซึ่งเขียนไว้เป็นฟังก์ชั่น(Function)จะขอนำโปรแกรมหลักเฉพาะในส่วนที่สำคัญมาอธิบายดังต่อไปนี้

```
#include <conio.h>     โปรแกรมสามบรรทัดแรกนี้เป็นการสั่งให้นำไฟล์ช่วยของเทอร์
#include <dos.h>       โบ ซีมาคอมไพล์ร่วมกับโปรแกรมนี้
#include <fcntl.h>
#include <screen.c>
#include <call8048.c>
#include <olport.c>
#include <recorder.c>
```

โปรแกรมในช่วงแรกนี้เป็นการสั่งให้นำไฟล์ช่วยของเทอร์โบ ซีและไฟล์ของโปรแกรมย่อยมาคอมไพล์ร่วมกับโปรแกรมนี้ โดยสามไฟล์แรกเป็นไฟล์ของเทอร์โบซี ไฟล์ screen.c เป็นไฟล์โปรแกรมย่อยที่จัดการเกี่ยวกับการแสดงบนจอภาพ ไฟล์ call8048.c เป็นไฟล์โปรแกรมเกี่ยวกับการอ่านข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์ ไฟล์ olport.c เป็นโปรแกรมเกี่ยวกับการจัดส่งข้อมูลไปยังเครื่องส่ง และไฟล์ recorder.c เป็นไฟล์เกี่ยวกับการบันทึกข้อมูลลงในดิสเกตท์ ไฟล์ของโปรแกรมย่อยเหล่านี้จะกล่าวถึงอีกครั้งหนึ่งภายหลัง ส่วนต่อไปเป็นส่วนกำหนดค่าคงที่

```
#define     HOURIPORT    0x2c4
#define     MINIPORT     0x2c3
#define     SECIPORT     0x2c2
```

สามบรรทัดนี้เป็นการกำหนดค่าของพอร์ทเวลาบนการ์ดมัลติฟังก์ชั่น (Multi function Card) ของเครื่องไอบีเค็มจะกล่าวถึงอีกครั้งในหัวข้อ 3.2.1 และ3.2.2

```
#define     RESETIPORT  0x302
```

บรรทัดนี้เป็นการกำหนดค่าพอร์ทซึ่งทำหน้าที่รีเซทการทำงานของเครื่องรับข้อมูลทางโทรศัพท์ในขณะเริ่มต้นการทำงาน เมื่อมีการส่งค่าจากพอร์ทหมายเลข 302H(ฐาน 16)สัญญาณจากแอดเดรสบัสจะไปรีเซทการทำงานของไมโครคอมพิวเตอร์ 8048 ดังที่ได้กล่าวมาแล้วในหัวข้อ 3.1

ส่วนต่อไปนี้เป็นการกำหนดตัวแปรภายนอก (Eternal Variable) การที่กำหนดเป็นตัวแปรภายนอกนี้ก็เนื่องจากต้องการให้ทุกๆฟังก์ชั่นสามารถใช้ตัวแปรเหล่านี้ร่วมกันได้

```
struct USER{
            char name[20];
            char surname[20];
            char address[40];
            char tel[10];
            char number[5];
            }user;
struct CALL{
            int pager;
            char area[4];
            char tel[8];
            char time[9];
            char date[9];
            }calling[100];
```

ข้อมูลแบบโครงสร้างสองชุดนี้เป็นโครงสร้างที่เก็บข้อมูลเกี่ยวกับการส่งเพื่อทำการบันทึกหรือค้นหาต่อไป

```
int  ttx = 0; /*  times of transmission */
int areallen,tellen;
```

ตัวแปร ttx นี้จะเก็บค่าจำนวนชุดของข้อมูลที่รับมาจากเครื่องรับข้อมูลทางโทรศัพท์หรือชี้ข้อมูลที่จะต้องส่ง

```
/*-------------------------------------------------------------*/
main()
  {
   int choicenumber = 0;
   int stop = 0;
   int loop = 1;
   inportb(RESETIPORT);
```

โปรแกรมบรรทัดนี้จะทำการรีเซทเครื่องรับข้อมูลทางโทรศัพท์

```
  while (loop != stop)
    {
     choicenumber = menu();
```

เป็นการเรียกฟังก์ชั่นชื่อ menu() ซึ่งทำหน้าที่แสดงรายการหลักของพนักงานรับโทรศัพท์และส่งค่าการเลือกของพนักงานรับโทรศัพท์มาเก็บไว้ที่ตัวแปรchoicenumber

```
     loop = call(choicenumber);
    }
  }
choice(columnlimit,width,space,ylimit,yspace,c0y,c0x)
  {

    while(kbhit() == 0)              /* CHECK KEYBOARD HIT */
        import8048();
```

ฟังก์ชั่น choice นี้เป็นฟังก์ชันเลื่อนตัวเลือกไปในรายการหลักแต่จะมีการทำงานที่สำคัญอีกอย่างหนึ่งคือการตรวจสอบการกดคีย์บอร์ดของพนักงานรับโทรศัพท์โดยใช้ฟังก์ชั่น kbhit() ของเทอร์โบ ซี ถ้าไม่มีการกดคีย์บอร์ดก็จะเรียกฟังก์ชั่น import8048() เพื่อทำการอ่านข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์(อ่านรายละเอียดในหัวข้อ 3.2.1

นอกจากส่วนที่กล่าวมาแล้วนี้โปรแกรมหลักยังมีส่วนย่อยอื่นๆอีกเช่นฟังก์ชั่น

Call()ทำหน้าที่เรียกฟังก์ชั่นการทำงานของพนักงานรับโทรศัพท์, ฟังก์ชั่น set.clock() สำหรับตั้งเวลา และโปรแกรมอื่นๆซึ่งจะไม่ขอกล่าวถึงในที่นี้

3.2.1 โปรแกรมรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์

ดังที่ได้กล่าวมาแล้วว่าไมโครคอมพิวเตอร์จะต้องอ่านข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์เป็นระยะๆด้วยช่วงเวลาที่เหมาะสม สลับไปกับการรับข้อมูลจากคีย์บอร์ด และการค้นหาหรือบันทึกข้อมูล การออกแบบโปรแกรมเพื่อให้ทำงานได้ตามนี้จะใช้นาฬิกาที่มีอยู่ในมัลติฟังก์ชั่นการ์ด(Multifunction card)ของเครื่องไอบีเอ็ม พีซี เป็นฐานเวลา การอ่านค่าเวลาจากมัลติฟังก์ชั่นการ์ดนี้ทำได้อย่างง่ายๆ โดยการอ่านพอร์ทหมายเลข2C4H จะได้เวลาหลักชั่วโมงและอ่านพอร์ทหมายเลข 2C3H และ2C2Hก็จะได้เวลาหลักนาทีและวินาทีตามลำดับสำหรับเวลาในการอ่านการอ่านข้อมูลนั้นจะต้องไม่นานเกินไปจนข้อมูลมีจำนวนมากซึ่งอาจจะทำให้หน่วยความจำของเครื่องรับข้อมูลทางโทรศัพท์ไม่เพียงพอ และจะต้องไม่เร็วเกินไปจนบางครั้งไม่มีข้อมูลชุดใดครบถ้วนพร้อมที่จะอ่านได้ ในที่นี้เลือกช่วงเวลานาน 1 นาที ซึ่งการหาเวลาในการอ่านทำโดยการตรวจสอบการเป็นศูนย์ของเวลาหลักวินาที จากหลักการสามารถนำมาใช้เขียนโปรแกรมได้ดังนี้

ในส่วนแรกนี้เป็นการกำหนดค่าคงที่หมายเลขพอร์ทต่าง โดยพอร์ทตรวจสอบความพร้อมที่จะส่งของเครื่องรับข้อมูลทางโทรศัพท์มีหมายเลขเป็น 301H พอร์ทข้อมูลมีหมายเลขเป็น 300H

```
# define RXJSTAT  0x301
# define RXJPORT  0x300
# define SECJPORT   0X2c2
```

```
import8048()
{
    int second,n;
int. inJpage [100][10];          /* data[frame][byte]*/
int frame,i,stop,max;
```

```
    extern CALL calling;
    extern  int ttx;
    extern  int areallen,telllen;
    second = inportb(SECIPORT);              /* inport second */
  if (second == 0)
```

เงื่อนไขนี้จะตรวจสอบการเป็นศูนย์ของหลักวินาทีของเวลาเมื่อเป็นศูนย์จะทำงานในส่วนต่อไป

```
    {
       gotoxy(20,20);
       printf(" IMPORT DATA FROM 8048   PLEASE WAIT!");
      do
         {
          promt = inportb(RXISTAT);
         } while ( promt < 128 );          /* check 7th bit */
```

โปรแกรมในลูปนี้จะตรวจสอบความพร้อมที่จะส่งข้อมูลของเครื่องรับข้อมูลทางโทรศัพท์โดยจะตรวจสอบบิทที่7ของข้อมูลที่อ่านเข้ามาได้

```
          inportb(RXIPORT);                  /* clear */
          frame = 0;
          stop = 0;
```

จากนั้นก็จะอ่านข้อมูลทิ้งหนึ่งครั้งเพื่อเคลียร์วงจรส่งข้อมูลและให้ค่าตัวแปรต่างๆให้เป็นศูนย์

```
          do{
             frame ++;
```

ลูปนี้เป็นลูปเพื่ออ่านข้อมูลเข้ามาทีละชุด ตัวแปรframeเป็นหมายเลขชุดของข้อมูล

```
             inIpage[frame][1] = inportb(RXIPORT);
             if(inIpage[frame][1] != 0xFF)  /* check EOT*/
```

จากนั้นอ่านข้อมูลไบท์แรกของชุดเข้ามาตรวจสอบว่าเป็นรหัสสิ้นสุดการส่ง (ffH) หรือไม่ ถ้าใช่ก็จะหลุดออกจากลูปนี้และทำงานตามคำสั่งelseที่ตอนท้ายโปรแกรม

```
             {
```

```
i = 1;
do
 {
   i++;
   in]page[frame][i] = inportb(RX]PORT);
 }while (in]page[frame][i] != 0x12);
 i--;
 max = i;
 for(;i <= 10;i++)
    in]page[frame][i]=0;
 calling[frame].pager = 0;
 for(i = 1;i <= 10;i++)
    {
       calling[frame].pager += in]page[fram][i]
       *power(10,max-1);
       max--;
    }
```

ข้อมูลที่ส่งมาตอนแรกนี้จะเป็นหมายเลขประจำเครื่องวิทยุติดตามตัวที่ต้องการติดต่อด้วยจะถูกเก็บไว้ในลักษณะอะเรย์จนกระทั่งพบเครื่องหมาย แล้วจึงแปลงตัวเลขเหล่านั้นเป็นเลขจำนวนจริง(long int.eger)เก็บลงในตัวแปรภายนอกเพื่อทำการส่งและบันทึกต่อไป

```
 i = 0;
 do
  {
    i++;
    calling[frame].area[i] = inportb(RX]PORT);
 }while (calling[frame].area[i] != 0x14)
 area]len = i - 1;
```

ข้อมูลที่ตามมาจะเป็นหมายเลขระหัสทางไกล(area code)จนกระทั่งพบตัวคั่นที่มีค่าเป็น 14H ข้อมูลต่อไปจะเป็นหมายเลขโทรศัพท์จบด้วยระหัสปิดท้ายชุดข้อมูลซึ่งมีค่าเป็น 16H เราจะต้องเก็บค่าจำนวนหลักของระหัสทางไกลและหมายเลขโทรศัพท์ไว้ในตัวแปรภาขนอกเพื่อใช้ในการส่งข้อมูลต่อไป

```
                i = 0;
                do
                 {
                   i++;
                   calling[frame].tel[i] = inportb(RX_PORT);
                 }while (calling[frame].area[i] != 0x16)
                 tel_len = i - 1;
            }
            else
             {
             stop = 1;
             ttx = num-1;
             }
           transmitt();
```

หลังจากรับข้อมูลจนครบ(จบด้วยระหัสจบการส่งมีค่าเป็น ffH)จะต้องบันทึกจำนวนชุดของข้อมูลลงในตัวแปร ttx และเรียกโปรแกรมส่งต่อไป

```
    }
 }
```

3.2.2โปรแกรมจัดการและบันทึกข้อมูล

โปรแกรมต่อไปนี้จะทำหน้าที่บันทึกข้อมูลที่,ส่งเวลาและวันที่ส่งลงในดิสเกทโปรแกรมนี้จะถูกเรียกทุกครั้งที่มีการส่งข้อมูลโดยการเรียกไม่ต้องส่งค่าใดๆให้แก่โปรแกรมเลยโปรแกรมนี้จะบันทึกข้อมูลที่อยู่ในตัวแปรภายนอก (External variable)ที่มีลักษณะเป็นข้อมูลโครงสร้างชื่อ calling และจะวนลูปการทำงานตามค่าจำนวนชุดของข้อมูลที่ส่งซึ่งเก็บอยู่ในแปรภายนอกชื่อ ttx

```
recorder()
{
        char ch;
        int n;
        extern int ttx;
        extern CALL calling;
        int handle, status,hour,minute;
        FILE *fp;
        struct date today;
        getdate(&today);
        hour   = inportb(HOURIPORT);
        minute = inportb(MINIPORT);
```

ที่จุดเริ่มต้นของโปรแกรมนี้จะอ่านค่าวันเดือนปีจากเอ็มเอสดอส(MS DOS) ด้วยฟังก์ชั่นภาษา ซีที่ชื่อ getdate และอ่านค่าเวลาจากพอร์ทชั่วโมงและนาทีเพื่อเป็นข้อมูลในการบันทึก

```
        handle = open("B:CALLING.DAT",O|APPEND);
        fp = fdopen(handle,"a");
      if (fp == NULL)
        printf("fdopen failed\n");
```

จากนั้นจึงเปิดไฟล์ข้อมูลชื่อ CALL.DAT ที่ไดรฟ์ B ตามข้อกำหนดของเทอร์โบซี

```
      else
       {
        for (n=1;n<=ttx;n++)
          {
           fprintf(fp,"%d%-4s%-8s",calling[n].pager
           ,calling[n].area,calling[n].tel);
           fprintf(fp,"%d%d%d%d%d",hour,minute
```

```
,today.dalmon,today.dalday,today.dalyear);
```

เมื่อการเปิดไฟล์เรียบร้อยก็จะเขียนข้อมูลลงในไฟล์โดยเรียงลำดับดังนี้

1 หมายเลขวิทยุติดตามตัว

2 ระหัสทางไกลของหมายเลขโทรกลับ

3 หมายเลขโทรกลับ

4 เวลาของการส่งสัญญาณ (ชั่วโมง,นาที)

5 วันที่ (เดือน,วัน,ปี)

จากนั้นก็จะปิดไฟล์เพื่อเก็บข้อมูลในหน่วยความจำบัฟเฟอร์ลงในดิสเกท

```
        }
      fclose(fp);
    }
}
```

3.2.3 โปรแกรมการส่งข้อมูล

โปรแกรมนี้จะเป็นโปรแกรมย่อยที่จะถูกเรียกจากโปรแกรมการทำงานของพนักงานรับโทรศัพท์หรือโปรแกรมรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์ โดยจะทำหน้าที่ส่งข้อมูลไปยังวงจรสร้างคลื่นพาหะรองโดยมีการเรียงลำดับดังนี้

บิทเริ่มต้น2ไบท์	หมายเลขเครื่องรับ	ระหัสข้อมูล	ระหัสทางไกลและหมายเลขโทรศัพท์
มีค่าเป็น ffH	ความยาว2ไบท์	1ไบท์	ความยาว 5 ไบท์

ระหัสข้อมูลที่มีความยาวหนึ่งไบท์นี้เป็นค่าบอกวงจรรับสัญญาณว่าหมายเลขโทรศัพท์ที่ส่งมานี้เป็นระหัสทางไกลกี่หลักและหมายเลขโทรศัพท์กี่หลักโดยสี่บิทแรกแสดงจำนวนหลักของระหัสทางไกลสี่บิทหลังแสดงจำนวนหลักของหมายเลขโทรศัพท์

ฟังก์ชั่น transmitt นี้สามารถเรียกได้โดยไม่ต้องส่งค่าใดๆมาทั้งสิ้นฟังก์ชั่นนี้จะทำงานโดยการส่งข้อมูลในตัวแปรภายนอกชื่อ calling เป็นจำนวนชุดเท่ากับค่าในตัวแปรภายนอกชื่อ ttx ข้อมูลในตัวแปร calling นี้สามารถมาได้จากสองทางคือ ทางโปรแกรมรับข้อมูลจากเครื่องรับข้อมูลทางโทรศัพท์ในกรณีนี้จำนวนชุดของข้อมูลจะถูกเก็บไว้ใน

ttx และอีกทางหนึ่งคือการส่งสัญญาณเรียกโดยพนักงานรับโทรศัพท์ ในกรณีนี้ ttx จะถูกให้ค่าเป็นหนึ่ง

```
transmitt()
    {
    extern CALL calling;
    extern int ttx;
    extern areallen,tellllen;
    int  i, data[10],codellen,n,m;
    int  high,low;

    for(i = 1;i <= ttx ;i++)
     {
      data[1] = calling[i].pager/256;
      data[2] = calling[i].pager%256;
```

ข้อมูล data[1]และdata[2]เป็นค่าของหมายเลขประจำเครื่องรับโดยที่ทำการแยกค่าออกเป็นไบท์สูงและไบท์ต่ำ

```
      data[3] = areallen * 16 + tellllen;
```

ข้อมูล data[3] นี้เป็นระหัสข้อมูลแสดงจำนวนหลัก

```
      strcat(calling[i].area,calling[i].tel);
      codellen = areallen + tellllen;
     if(codellen % 2 == 1)
       {
        calling[i].area[codellen] = 48 ;
        codellen++;
       }
     m = 4;
     for (n = 0;n <= codellen-1;n+=2)
       {
```

```
        high = calling[i].area[n];
        low  = calling[i].area[n+1];
        data[m] = 16*(high - 48) + (low - 48);
        m++;
       }
```

ส่วนที่เหลือนี้เป็นการนำระหัสทางไกลและหมายเลขโทรศัพท์มาเรียงต่อกันแล้วแปลงจากค่าตังอักษรให้เป็นเลขจำนวนจริงจากนั้นจึงแยกออกเป็นชุดละสองหลัก

```
    for ( i = 1 ;i <= 6;i++)
       outlserial(0xFF);
```

สองบรรทัดนี้เป็นการส่งบิทเริ่มต้นยาวสองไบท์ออกไปบิทละสามครั้งก่อนที่จะส่งข้อมูล

```
     for (n = 1;n <= codeLlen/2 + 4;n++)
       outlserial(data[n]);
     }
    recorder();
 }
   outlserial(data)
```

ฟังก์ชั่น outlserial นี้จะทำหน้าที่ส่งข้อมูลออกไปยังพอร์ทหมายเลข 300H โดยที่ก่อนการส่งข้อมูลจะต้องมีการตรวจสอบความพร้อมที่จะส่งข้อมูลของวงจรส่งโดยการอ่านข้อมูลจากพอร์ทหมายเลข 303H เมื่อบิทที่ 7 ของข้อมูลที่อ่านได้เป็นหนึ่งแสดงว่าวงจรพร้อมที่จะส่งข้อมูลแล้ว การส่งข้อมูลนี้จะส่งบิทละสามครั้งเพื่อให้เครื่องรับสามารถเลือกรับข้อมูลที่ถูกต้องได้

```
    {
     int status,i;
     do
        {
          status = inportb(TXISTAT);
        } while (status < 128);
    for (i = 1;i <= 3;i++)
```

```
out.portb(TXIPORT,dat.a);  /* out data */
}
```

3.3 การคอกแบกและการสร้างวงจรส่งข้อมูล

3.3.1 วงจรแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม

การส่งข้อมูลจากเครื่องไมโครคอมพิวเตอร์ไอบีเอ็ม ซึ่งมีข้อมูลเป็นแบบขนาน 8 เส้น จะต้องทำการแปลงเป็นข้อมูลของอนุกรมก่อน เพื่อที่จะสามารถนำข้อมูลไปส่งผ่านระบบวิทยุได้ ในภาวะปกติเมื่อไม่มีการส่งข้อมูล ขาข้อมูล D_0-D_7 ของไอบีเอ็ม จะมีสถานะโลจิกเป็น 1 ทั้ง 8 ขา ดังนั้นจึงจะกำหนดให้บิทเริ่มต้น (START BIT) ของการส่งข้อมูลให้มีโลจิกเป็น 0 เพื่อทำให้ตัวรับข้อมูลรู้ว่าเป็นการเริ่มส่งข้อมูล และให้บิทสุดท้าย (STOP BIT) เป็นโลจิก 1 ซึ่งเป็นการบอกการสิ้นสุดของข้อมูล ซึ่งระบบที่สร้างขึ้นนี้จะทำให้การส่งข้อมูลออกไปทีละตัว ซึ่งข้อมูล 1 ตัว มี 8 บิท ดังนั้นในการส่ง 1 ครั้ง จะมีข้อมูล 10 บิท ดังรูปที่ 3.7

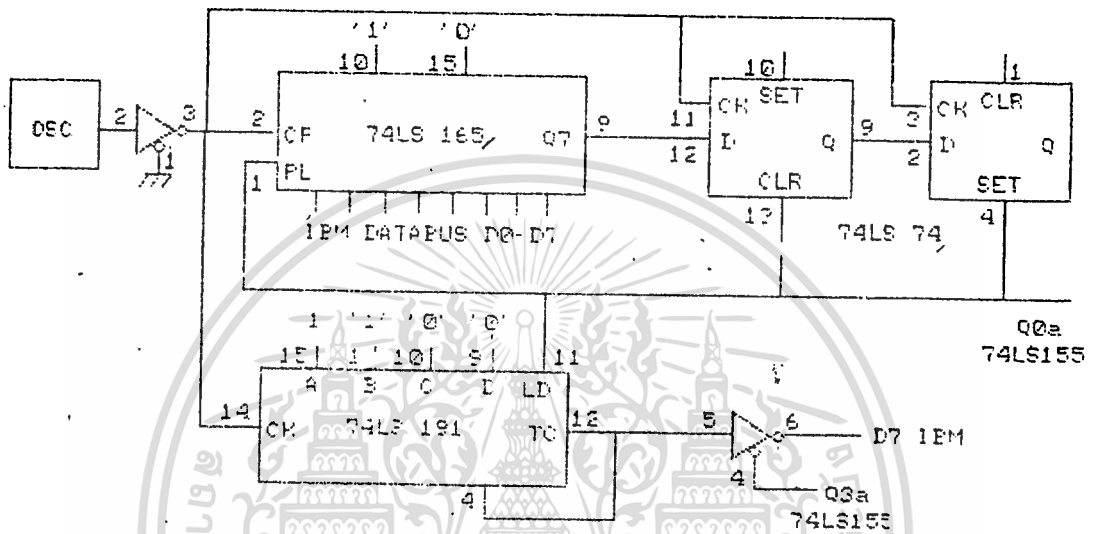
บิทสุดท้าย บิทเริ่มต้น

รูปที่ 3.7 แสดงลักษณะข้อมูลแบบอนุกรม

สำหรับวงจรที่ใช้แปลงข้อมูลแบบขนาน เป็นข้อมูลแบบอนุกรม และมีการเพิ่มบิทเริ่มต้นและบิทสุดท้ายนั้นใช้ไอซีชิฟต์รีจิสเตอร์ 8 บิท เบอร์ 74LS165 เพื่อรับข้อมูล D_0-D_7 ของไอบีเอ็มเข้ามา โดยต่อขา 10 ซีเรียลอินพุท (Serial Input) ให้มีโลจิกเป็น 1 ซึ่งเป็นการกำหนดบิทสุดท้าย สำหรับบิทเริ่มต้นนั้นใช้ไอซีดีฟลิปฟลอป (D FLIPFLOP) เบอร์ 74LS74 ต่ออยู่หน้าไอซี 74LS165 เพื่อเป็นการเพิ่มบิทข้อมูลอีก 1 บิท ซึ่งเป็นบิทเริ่มต้นโดยต่อในลักษณะ เมื่อมีการชิพ (Shift) ข้อมูลออกไป จะให้บิทเริ่มต้นเป็น 0 ตามที่ต้องการ

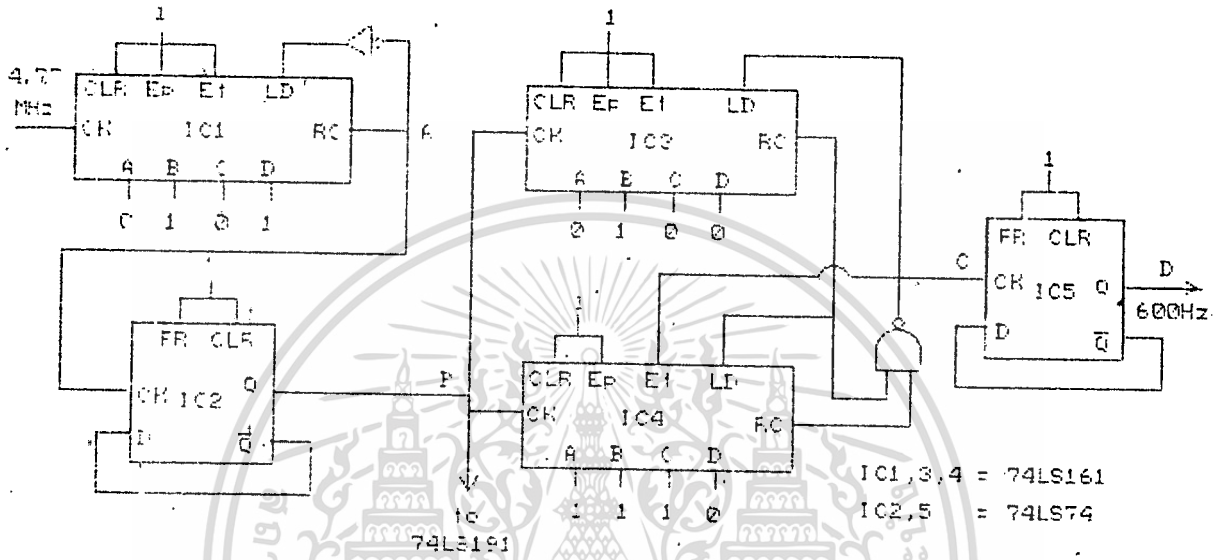
เนื่องจากเราต้องการที่จะให้คาบเวลาของข้อมูลที่ส่งไปมีค่าคงที่ตลอด เพื่อทำให้การรับข้อมูลเป็นไปอย่างถูกต้อง ถ้าต่อวงจรในลักษณะที่ไม่มีดีฟลิปฟลอปอีกตัวหนึ่งอยู่หน้าดีฟลิปฟลอปที่กำหนดบิทเริ่มต้น จะทำให้บิทเริ่มต้นนี้มีคาบเวลาไม่แน่นอนและจะสั้น

กว่าคาบเวลาปกติ จึงต้องนำดีฟลิปฟลอบอีกตัวหนึ่งต่อเพิ่มไปอีกหนึ่งบิท และกำหนดให้มีโลจิก 1 เมื่อมีการรีเซ็ตข้อมูลออก ซึ่งบิทนี้จะไม่มีความสำคัญต่อข้อมูล



รูปที่ 3.8 แสดงวงจรแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม

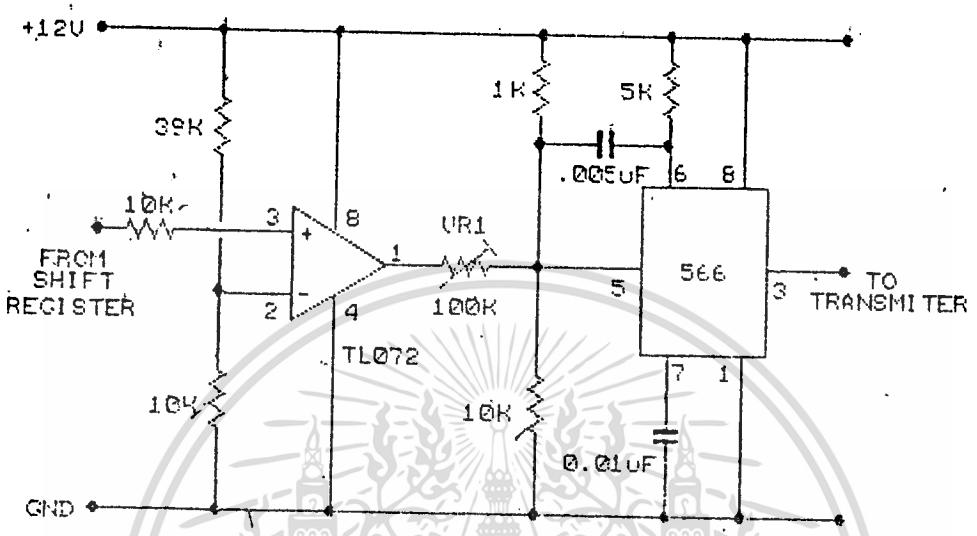
วงจรนี้จะมีการใช้ไอซีเคาน์เตอร์ (COUNTER) ทำการนับ Clock เพื่อเป็นการเช็คเมื่อข้อมูลถูกส่งไปครบ 1 ตัว ขา Tc ก็จะเปลี่ยนโลจิกจาก 0 เป็น 1 แล้วป้อนเข้าขาอีนาเบิล (Enable) เป็นการหยุดการนับของเคาน์เตอร์และทำให้ Tc มีโลจิกเป็น 1 ค้างไว้ ซึ่งจะทำให้ไอบีเอ็มสามารถตรวจสอบข้อมูลจากขา D_7 ของไอบีเอ็มว่ามีโลจิกเป็น 1 หรือยัง ถ้าเป็น 1 แล้วแสดงว่าข้อมูลถูกส่งไปครบ 1 ตัวแล้ว ทำให้สามารถส่งข้อมูลตัวต่อไปได้ ซึ่งการควบคุมการส่งข้อมูล และการตรวจเช็คการส่งข้อมูลทำได้โดยการเรียกพอร์ท (Port) จากไอบีเอ็ม โดยอัตราความเร็วในการส่ง จะถูกกำหนดโดยวงจรออสซิเลเตอร์ ซึ่งผลิตสัญญาณนาฬิกาในการควบคุมการทำงานของวงจรทั้งหมด ซึ่งในระบบนี้กำหนดให้ความเร็วในการส่งเท่ากับ 600 บิทต่อวินาที ซึ่งวงจรออสซิเลเตอร์ที่ใช้นี้จะใช้วงจรหารความถี่จากสัญญาณนาฬิกาของไอบีเอ็ม 4.7 เมกกะเฮิทร์ซ (MHz) มาเป็นความถี่ประมาณ 600 เฮิทร์ซ ซึ่งมีวงจรในรูปที่ 3.9



รูปที่ 3.9 แสดงวงจรสร้างสัญญาณนาฬิกาในการส่งข้อมูล

3.3.2 วงจรมอดูเลทโดยใช้โวลท์เตจคอนโทรลออสซิเลเตอร์

วงจรมอดูเลทโดยใช้โวลท์เตจคอนโทรลออสซิเลเตอร์แสดงไว้ดังรูปที่ 3.10 เนื่องจากสัญญาณข้อมูลจากไอบีเอ็ม ซึ่งมีโลจิก 0,1 จะมีระดับแรงดัน 0 กับ 5 โวทล์ ตามลำดับ แต่วงจรโวลท์เตจคอนโทรลออสซิเลเตอร์ ซึ่งใช้ไอซี 566 โดยจะออกแบบให้ใช้ความถี่ 7.6 และ 8.4 กิโลเฮิร์ซนั้น จะต้องใช้ไฟเลี้ยงประมาณ 9 โวลท์ขึ้นไป และเนื่องจากวงจรทั้งหมดนี้ต่ออยู่บนการ์ดของไอบีเอ็ม ซึ่งมีไฟเลี้ยง 12 โวลท์อยู่ด้วย จึงใช้ไฟเลี้ยง 12 โวลท์ สำหรับวงจรไอซี 566 นี้ จะต้องทำการยกระดับสัญญาณ 0,5 โวลท์ ให้เป็น 0,12 โวลท์ โดยใช้คอมพาราเตอร์ (Comparator) ซึ่งใช้ออปแอมป์ (Op-Amp) เบอร์ TL072 มายกระดับสัญญาณแล้วจึงป้อนเข้า 566



รูปที่ 3.10 แสดงวงจรมอดูเลทโดยใช้โวลท์เตจคอนโทรลออสซิเลเตอร์

สูตรในการคำนวณความถี่ที่ออกมาของไอซี 566 คือ

ความถี่ที่ออกมา (fo) = $2(Vcc - Vin) / (R_1C_1 * Vcc)$

โดย

Vcc คือ แรงดันไฟเลี้ยงของวงจร

Vin คือ แรงดันที่ป้อนเข้าขา 5 ของไอซี 566

กำหนดให้ค่า R1 = 5 กิโลโอห์ม

C1 = 0.01 ไมโครฟาร์ด

ถ้าต้องการความถี่ประมาณ 7.6 กิโลเฮิทร์ซ จะต้องป้อนแรงดันเข้าที่ขา 5 ของไอซี 566

$$Vin = 12 - (fo * R_1C_1 * Vcc)/2$$

$$= 12 - (7.6*10^{3} * 5*10^{3} * 0.01*10^{-6} * 12)/2 \quad \text{โวลท์}$$

$$= 9.72 \quad \text{โวลท์}$$

ถ้าต้องการความถี่ประมาณ 8.4 กิโลเฮิร์ซ จะต้องป้อนแรงดันเข้าที่ขา 5 ของไอซี 566

$$Vin = 12 - (8.4*10^{3} *5*10^{3} *0.01*10^{-5} *12/2 \quad \text{โวลท์}$$
$$= 9.48 \quad \text{โวลท์}$$

แต่เนื่องจากระดับสัญญาณจากคอมพาราเตอร์เมื่อป้อนโลจิก 0,1 จะมีระดับแรงดัน 0,12 โวลท์ ตามลำดับ ซึ่งเราจะต้องทำการลดทอนความแตกต่างของระดับสัญญาณโดยการต่อความต้านทาน 1 กิโลโอห์ม และ 10 กิโลโอห์มเป็นการลดทอนความแตกต่างของระดับแรงดันทั้งสอง และมีความต้านทานปรับค่าได้ (VR_1) 100 กิโลโอห์ม ต่ออยู่ เพื่อใช้ปรับให้แรงดันที่ป้อนเข้าขา 5 ไอซี 566 ให้มีระดับความแตกต่างที่เหมาะสมตามค่าที่คำนวณไว้ เพื่อจะให้ได้ว่า เมื่อมีโลจิก 0 จะทำให้ไอซี 566 ผลิตความถี่ค่าประมาณ 8.4 กิโลเฮิร์ซ และเมื่อมีโลจิก 1 จะทำให้ไอซี 566 ผลิตความถี่ค่าประมาณ 7.6 กิโลเฮิร์ซได้จริงตามความต้องการโดยในการต่อวงจรจริง จะใช้ความต้านทานปรับค่าได้ 10 กิโลโอห์ม แทนค่าความต้านทาน (R_1) 5 กิโลโอห์ม เพื่อใช้ช่วยปรับให้ได้ความถี่ที่ต้องการอย่างแท้จริง

3.3.3 วงจรส่งข้อมูลโดยใช้วงจรหารความถี่

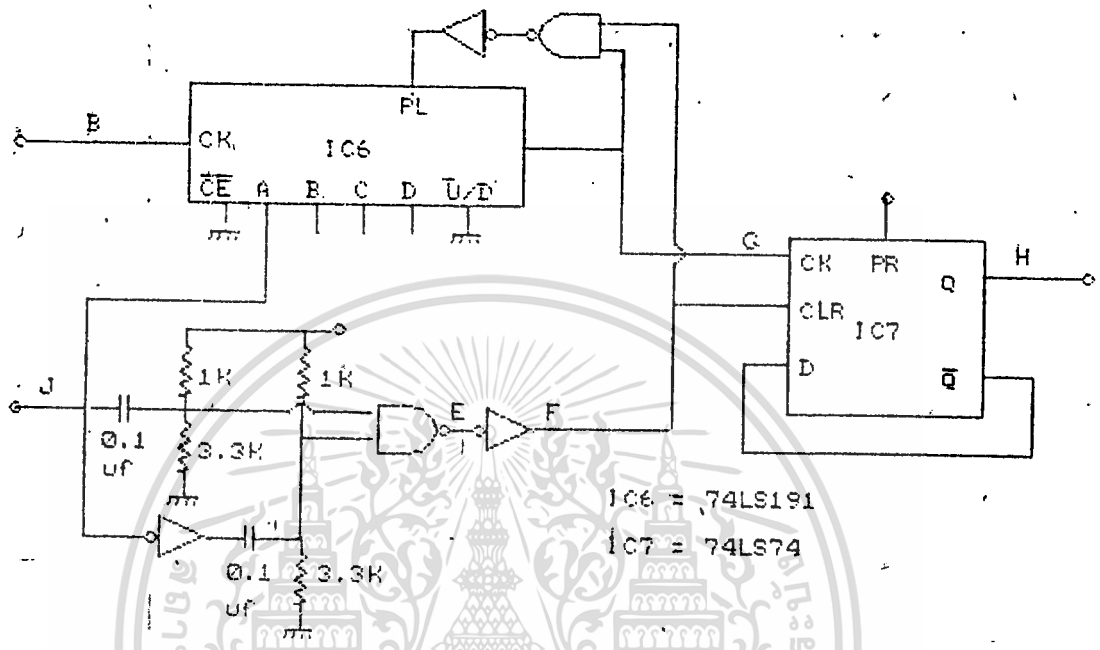
วงจรส่งข้อมูลโดยใช้วงจรหารความถี่แสดงไว้ในรูปที่ 3.11

จากรูป วงจรหารความถี่ จะเห็นว่าเราจะใช้ไอซีเคาน์เตอร์ (COUNTER) เบอร์ 74LS161 3 ตัวกับเบอร์ 74LS191 1 ตัว และใช้ไอซีดีฟลิปฟลอปอีก 3 ตัว ใช้ในการหารโดยไอซี 74LS161 ตัวที่ 1 นั้นจะหารความถี่สัญญาณนาฬิกา 4.77 เมกกะเฮิร์ซ (MHz) ของเครื่องไมโครคอมพิวเตอร์ไอบีเอ็มลง 14 เท่า นั้นคือ

ความถี่ที่จุด A = $4.77x10^{6}/14$ = 340.714 กิโลเฮิร์ซ

และไอซีตัวที่ 2 ซึ่งเป็นไอซีดีฟลิปฟลอปซึ่งต่อในลักษณะที่จะทำให้เป็นวงจรหารความถี่ลงครึ่งหนึ่ง และคาบเวลาของสัญญาณนาฬิกาในช่วงโลจิก 0 กับ 1 จะมีค่าเท่ากัน ดังนั้น

ความถี่ที่จุด B = $340.714x1^{03}/2$ = 170.357 กิโลเฮิร์ซ



รูปที่ 3.11 แสดงวงจรส่งข้อมูลโดยใช้วงจรหารความถี่

ซึ่งความถี่ที่จุด B จะเป็นสัญญาณนาฬิกาป้อนไอซี 3,4 และ 6 ซึ่งไอซี 3,4 เป็นไอซี 74LS161 ซึ่งต่อร่วมกันในลักษณะเป็นซิงโคนัสเคาน์เตอร์ (Synchronous Counter) ซึ่งจะทำให้หารความถี่ได้มากขึ้น ซึ่งในวงจรนี้ตั้งให้ไอซีทั้งสองทำการหารได้ 142 เท่า ดังนั้น

ความถี่ที่จุด C = $170 \times 10^{3} / 142$ = 1199.698 เฮิร์ซ

และไอซีตัวที่ 5 ซึ่งเป็นไอซีดีฟลิปฟลอป ซึ่งทำการหารความถี่เป็นครึ่งหนึ่ง ซึ่งจะได้ว่า

ความถี่ที่จุด D = 1199.698/2 = 599.85 เฮิร์ซ

ซึ่งความถี่ 600 Hz ใช้เป็นสัญญาณนาฬิกาในการแปลงข้อมูลแบบขนานเป็นข้อมูลอนุกรม ดังได้กล่าวมาแล้ว

ใช้สำหรับไอซีตัวที่ 6 ซึ่งเป็นไอซี 74LS191 นั้น ขา A จะต่อกับข้อมูลซึ่งถูกแปลงจากแบบขนานมาเป็นแบบอนุกรมแล้ว ซึ่งจะเห็นว่าเป็นข้อมูลมีโลจิกเป็น 1 จะทำให้ไอซี 74LS191 หารความถี่ด้วย 10 ถ้าข้อมูลที่เข้ามามีโลจิกเป็น 0 จะทำให้ไอซี 74LS191 หารความถี่ด้วย 11 จะได้ว่า

ความถี่ที่จุด G เมื่อข้อมูลมีโลจิก 0 = $170 \times 10^3 / 11 = 15454.5$ เฮิร์ซ

เมื่อข้อมูลมีโลจิก 1 = $170 \times 10^3 / 10 = 17000$ เฮิร์ซ

โดยจะมีวงจรสร้างสัญญาณ พัลส์ (Pulse) แคบ ๆ เมื่อมีโลจิกที่จุด J มีการเปลี่ยนแปลงจากโลจิก 0 เป็น 1 หรือจาก 1 เป็น 0 จะเกิดพัลส์แคบ ๆ ที่จุด E คือมีคาบของสถานะสูงหรือโลจิก 1 แคบ ๆ เมื่อผ่านอินเวอร์เตอร์ (Inverter) แล้วที่จุด F จะเกิดพัลส์แคบ ๆ ในลักษณะเป็นสถานะต่ำหรือโลจิก 0 แคบ ๆ เมื่อมีการเปลี่ยนสถานะที่จุด J ซึ่งพัลส์แคบ ๆ นี้จะไปทำการกำหนดให้ไอซี 74LS191 ทำการเริ่มนับสัญญาณนาฬิกาใหม่ เมื่อเปลี่ยนโลจิกที่จุด J ซึ่งจะทำให้ตัวหารเปลี่ยนไป ดังนั้นสัญญาณที่ได้ที่จุด H จะเป็นสัญญาณที่มีความถี่ ดังนี้คือ

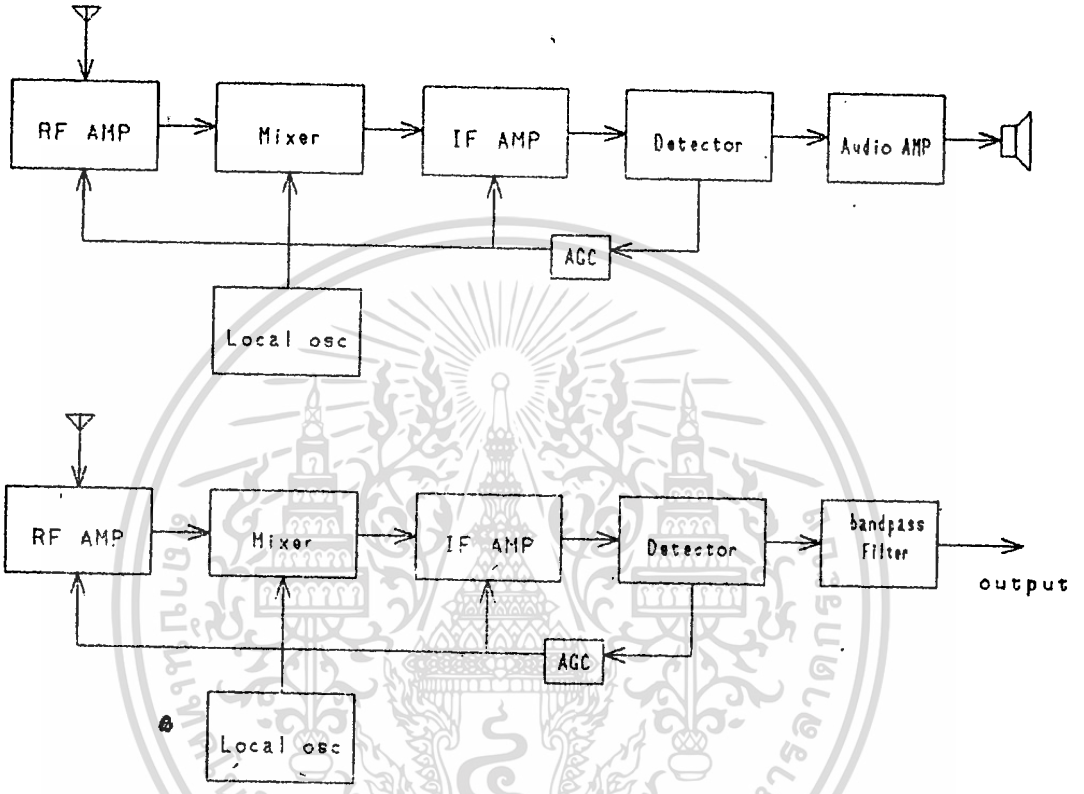
ความถี่ที่จุด H เมื่อข้อมูลมีโลจิก 0 = 15454.5/2 = 7.7 กิโลเฮิร์ซ

เมื่อข้อมูลมีโลจิก 1 = 17000/2 = 8.5 กิโลเฮิร์ซ

3.4 การออกแบบและการสร้างวงจรรับข้อมูล

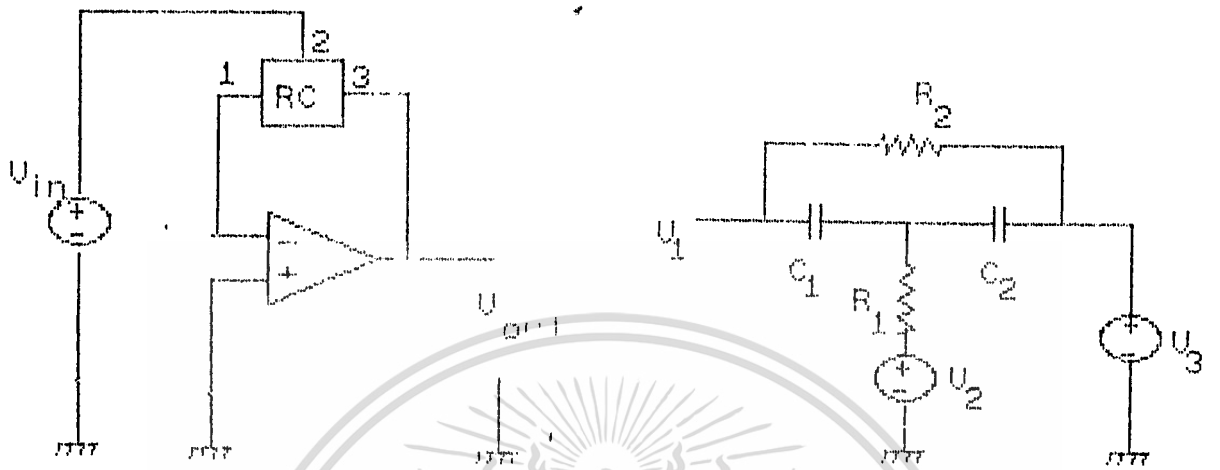
3.4.1 วงจรรับสัญญาณระบบเอ.เอ็ม.

ดังที่ได้กล่าวมาแล้วในหัวข้อ 2.5 ว่าเราสามารถดัดแปลงวงจรเครื่องรับวิทยุทั่วไปให้เป็นวงจรรับสัญญาณคลื่นพาหะรองได้โดยการเปลี่ยนวงจรกรองผ่านความถี่ต่ำในวงจรดีเทคเตอร์เป็นวงจรกรองผ่านช่วงความถี่ ดังรูปที่ 3.12



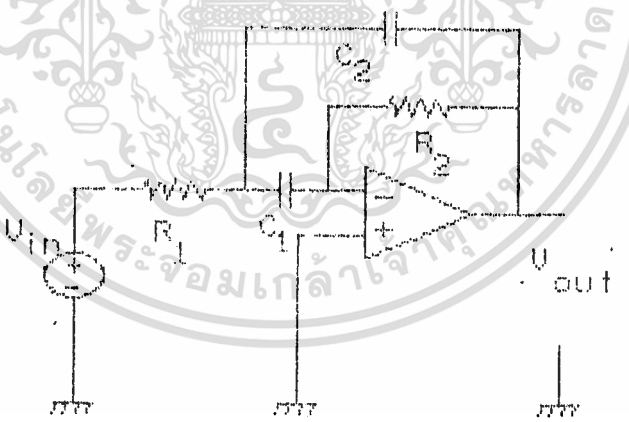
รูปที่ 3.12 วงจรรับสัญญาณคลื่นเฟาะรอง

วงจรที่เลือกใช้เป็นวงจรกรองผ่านช่วงความถี่เป็นวงจรแอคทีฟฟิลเตอร์ (Active filter) ออเดอร์สอง (2 nd order) ที่มีการป้อนกลับแบบลบ (Negative Feedback) ดังรูปที่ 3.13 ก วงจรที่เป็นส่วนป้อนกลับเป็นบริดจ์ที อาร์ซีเน็ทเวอร์ค (Bridge T RC Network) ดังรูปที่ 3.13 ข



(ก) (ข)

รูปที่ 3.13 ก.แอคทีฟเน็ทเวอร์ที่ใช้การป้อนกลับแบบลบ ข.บริดจ์ที อาร์ซีเน็ทเวอร์ค



รูปที่ 3.14 วงจรกรองผ่านช่วงความถี่แบบแอคทีฟ

ทรานสเฟอร์ฟังชั่นของวงจรในรูปที่ 3.14 หาได้จากการใช้โนดอีเควชั่น (Node Equation) ดังนี้

$$\begin{bmatrix} sC_1 + sC_2 + 1/R_1 & -sC_1 \\ sC_1 & sC_1 + 1/R_2 \end{bmatrix} \begin{bmatrix} V_x \\ V_1 \end{bmatrix} = \begin{bmatrix} 1/R_1 & sC_2 \\ 0 & 1/R_2 \end{bmatrix} \begin{bmatrix} V_2 \\ V_3 \end{bmatrix}$$

เนื่องจาก $T_{BP} = - T_{FF}/T_{FB}$

โดยที่ T_{BP} เป็น ทรานสเฟอร์ฟังชั่นของวงจรกรองผ่านช่วงความถี่

T_{FF} เป็น ทรานสเฟอร์ฟังชั่นของฟีดฟอร์เวอร์ดพาร์ท(Feedforward part.)

T_{FB} เป็น ทรานสเฟอร์ฟังชั่นของฟีดแบคพาร์ท (Feedback part.)

$$T_{FB} = V_1/V_3 \,|V_2=0 = \frac{s^2 + s(1/R_2C_1 + 1/R_2C_2) + 1/R_1R_2C_1C_2}{s^2 + s(1/R_2C_1 + 1/R_2C_2 + 1/R_1C_2) + 1/R_1R_2C_1C_2}$$

$$T_{FF} = V_1/V_2 \,|V_3=0 = \frac{s/R_1C_2}{s^2 + s(1/R_2C_1 + 1/R_2C_2 + 1/R_1C_2) + 1/R_1R_2C_1C_2}$$

$$T_{BP} = - T_{FF}/T_{FB} = \frac{s/R_1C_2}{s^2 + s(1/R_2C_1 + 1/R_2C_2) + 1/R_1R_2C_1C_2}$$

เมื่อพิจารณาเทียบกับทรานสเฟอร์ฟังชั่นของวงจรกรองช่วงความถี่ออเดอร์สอง

$$T_{BP} = K_1 \frac{s}{s^2 + w_p s/Q_p + w_p^2}$$

จะหาค่าแสดงคุณสมบัติของวงจร (Filter Characteristic) ได้ดังนี้

ความถี่กลางของช่วงผ่าน $f_o = 1/2\pi (R_1R_2C_1C_2)^{1/2}$

โดยที่ Q_p เป็นโพลคิว(Pole Q)ของวงจร = f_o/f ; f เป็นความกว้างของช่วงผ่าน (Pass band)

$$Q_p = \frac{(R_2/R_1)^{1/2}}{(C_2/C_1)^{1/2} + (C_1/C_2)^{1/2}}$$

และอัตราขยายในช่วงผ่าน (Passband Gain)

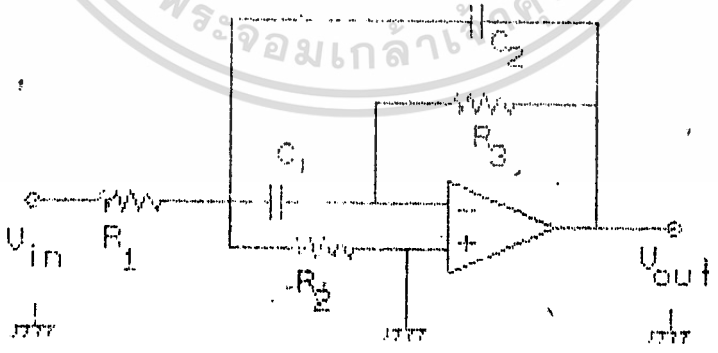
$$K = -1/R_1C_2$$

จากสมการของคุณสมบัติเหล่านี้จะเห็นได้ว่าถ้าต้องการเปลี่ยนแปลงความถี่กลางของช่วงผ่าน(f_o) จะทำได้ยากเนื่องจากการปรับค่าอุปกรณ์ใดๆในวงจรจะทำให้คุณสมบัติอื่นๆของวงจรเปลี่ยนแปลงไปด้วย ดังนั้นจึงได้ใช้วงจรที่ดัดแปลงมาจากวงจรนี้เรียกว่า วงจรมัลติเพิ่ลฟีดแบค (Multiple Feedback Bandpass Filter) ในรูปที่ 3.15 ซึ่งวงจรนี้มีข้อดีเหมาะสมกับการเลือกใช้คือ

1. ใช้ออพแอมป์ (Operational Amplify)เพียงตัวเดียว

2. ค่าความถี่กลางของช่วงผ่านสามารถปรับได้โดยเปลี่ยนค่าความต้านทาน R_2 โดยมีผลกระทบต่อคุณสมบัติอื่นๆเพียงเล็กน้อย

3. ในกรณีที่โพลดิวมีค่าน้อยกว่า 10 การเปลี่ยนแปลงของ Q_p และf_o เนื่องจากการเปลี่ยนไปของค่าอุปกรณ์มีค่าไม่สูง



รูปที่ 3.15 วงจรกรองผ่านช่วงความถี่แบบมัลติเพิ่ลฟีดแบค

ทรานสเฟอร์ฟังชั่นของวงจรมัลติเพิ่ลฟีดแบคสามารถหาได้ด้วยวิธีเดียวกัน โดยจะมีค่าเป็น

$$T_{BP} = \frac{s/R_1C_2}{s^2 + s(1/R_3C_1 + 1/R_3C_2) + (1/R_1+1/R_2)/R_3C_1C_2}$$

ความถี่กลางของช่วงผ่าน $f_o = \frac{1}{2\Pi}\left[\frac{1}{R_3C_1C_2}\left[\frac{1}{R_1}+\frac{1}{R_2}\right]\right]^{1/2}$

$$Q_p = \frac{[R_3(1/R_1+1/R_2)]^{1/2}}{(C_2/C_1)^{1/2} + (C_1/C_2)^{1/2}}$$

$$K = R_3C_2 / R_1(C_1 + C_2)$$

เมื่อหาค่าของอุปกรณ์ในวงจรในเทอมของคุณสมบัติโดยให้ $C_1=C_2=C$ จะได้

$R_1 = 1/(2\Pi fKC)$

$R_2 = [2\Pi C(2f_o^2/f - fK]^{1/2}$

$R_3 = 1/\Pi fC$

สำหรับระบบวิทยุติดตามตัวนี้เราใช้สัญญาณคลื่นพาหะรองที่มีความถี่ 7.7และ 8.5 KHz ดังนั้นเจ้งต้องการวงจรกรองช่วงความถี่ที่มีค่าความถี่กลางของช่วงผ่านประมาณ 8.1 KHz มีช่วงผ่านของความถี่ตั้งแต่ ประมาณ 7.5 ถึง 8.9 KHz ซึ่งจะต้องการค่าโพลคิวประมาณ 5 และใช้อัตราขยายในช่วงผ่านเป็น 10 กำหนดค่าตาปาซิเตอร์ C = .001 uF จากคุณสมบัติต่อเหล่านี้สามารถ

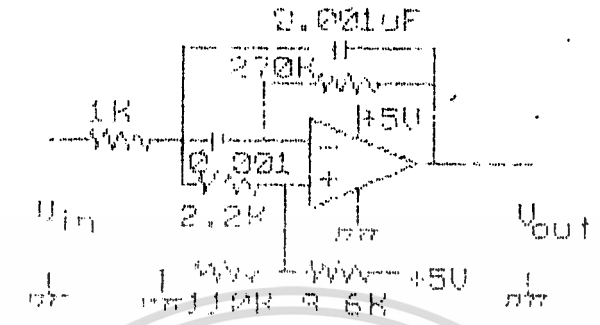
คำนวณหาค่าของอุปกรณ์ได้เป็น

R_1 = 12,242 ohm ใช้ค่าที่หาได้ง่ายเป็น 1 K ohm

R_2 = 2,456 ohm ใช้ค่าที่หาได้ง่ายเป็น 2.2 K ohm

R_3 = 244,853 ohm ใช้ค่าที่หาได้ง่ายเป็น 270 K ohm

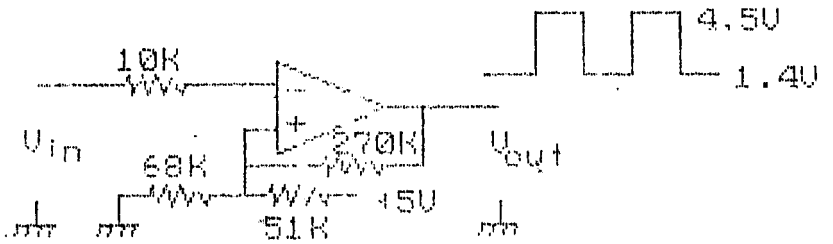
นำค่าเหล่านี้ไปสร้างเป็นวงจรได้ตามรูปที่ 3.15



รูปที่ 3.15 วงจรกรองช่วงความถี่สำหรับคลื่นพาหะรอง

เนื่องจากวงจรนี้นำไปใช้โดยมีไฟเลี้ยงเพียงชุดเดียว (Single Supply) ดังนั้นจึงจะต้องมีการไบอัสแรงดันประมาณครึ่งหนึ่งของไฟเลี้ยงให้กับขาบวก (Non Inverting input) ของ ออพแอมป์ ซึ่งทำได้โดยการต่อความต้านทาน R_4และ R_5 ดังรูป 3.15

อย่างไรก็ตามสัญญาณที่ต้องการในการป้อนให้กับวงจรดีมอดูเลเตอร์ไม่ว่าจะเป็นดิจิตอลเฟสล็อคลูปหรือวงจรนับจะต้องมีขนาดใหญ่และมีลักษณะเป็นคลื่นสี่เหลี่ยม (Square Wave)ดังนั้นเอาท์พุทที่ได้จากวงจรกรองช่วงความถี่จะต้องถูกขยายให้มีขนาดขึ้นจนทำให้เป็นคลื่นสี่เหลี่ยมการขยายในลักษณะนี้สามารถทำได้โดยใช้วงจรเปรียบเทียบแรงดัน (Voltage Comparater) แต่เนื่องจากถึงแม้จะผ่านวงจรกรองช่วงความถี่มาแล้วก็ตามก็อาจมีสัญญาณรบกวนจากสัญญาณเสียงหรือสัญญาณอื่นๆผ่านเข้ามาได้บ้างโดยจะถูกลดทอนให้มีขนาดเล็กลงการที่ใช้วงจรเปรียบเทียบแรงดันธรรมดาจะทำให้สัญญาณรบกวนเหล่านี้ถูกขยายขึ้นและไปรบกวนการทำงานของวงจรดีมอดูเลเตอร์ ดังนั้นจึงเลือกวงจรเปรียบเทียบแรงดันแบบที่มีการป้อนกลับของเอาท์พุททำให้เกิดฮิสเทอรีซิส(Hysteresis)ของแรงดันอ้างอิงหรือเรียกอีกอย่างว่าวงจรชมิททริกเกอร์ (Shmittriger circuit) ดังรูปที่ 3.16



รูปที่ 3.16 วงจรชมิททริกเกอร์

เนื่องจากวงจรนี้นำไปใช้ขยายสัญญาณจากวงจรกรองช่วงความถี่ดังนั้นจึงจะต้องไบอัส(Bias)ให้แรงดันอ้างอิงมีค่าอยู่ที่ระดับเดียวกับระดับไฟตรงที่ออกมาจากวงจรกรองช่วงสัญญาณทำได้โดยใช้ความต้านทาน R_3, R_4 เป็นวงจรแบ่งแรงดัน (Voltage Devider) วงจรนี้จะไม่ขยายสัญญาณที่มีแรงดันพีคทูพีค (Peak to Peak Voltage)ต่ำกว่าค่าฮิสเทอรีซิสของแรงดันอ้างอิง ดังนั้นค่าแรงดันพีคทูพีคของสัญญาณรบกวนจะเป็นค่าที่กำหนดค่าฮิสเทอรีซิสจากสมการ

$$V_{noisep-p} = R_t V_{outp-p} / R_f$$

โดยที่ V_{outp-p} เป็นค่าแรงดันพีคทูพีคของแรงดันเอาท์พุท

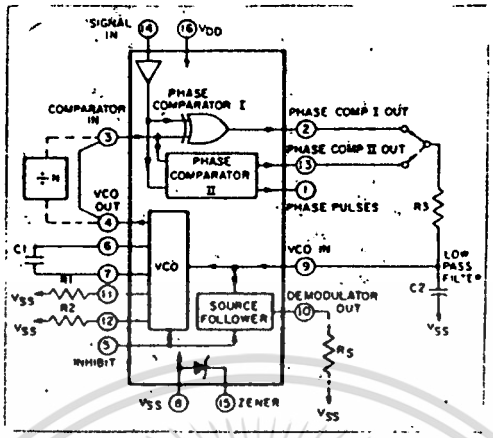
R_t เป็นค่าความต้านทานระหว่างขาบวกกับกราวนด์สามารถหาได้จาก

$$R_t = R_1 R_2 / (R_1 + R_2)$$

จากการทดลองไอซีออพแอมป์เบอร์ TL 074 มีค่าแรงดันเอ้าท์พุทพีคทูพีคสูงสุดประมาณ 3 โวล์ทที่แรงดันไฟเลี้ยง 5โวล์ท จากการคำนวณ R_t = 29 K ohm และกำหนดค่าของแรงดันพีคทูพีคของสัญญาณรบกวนไว้ที่ 0.3 V ดังนั้นจะค่าของ R_f ได้ประมาณ 291 K ohm เลือกค่าที่ใกล้เคียงคือ 270 K ohm

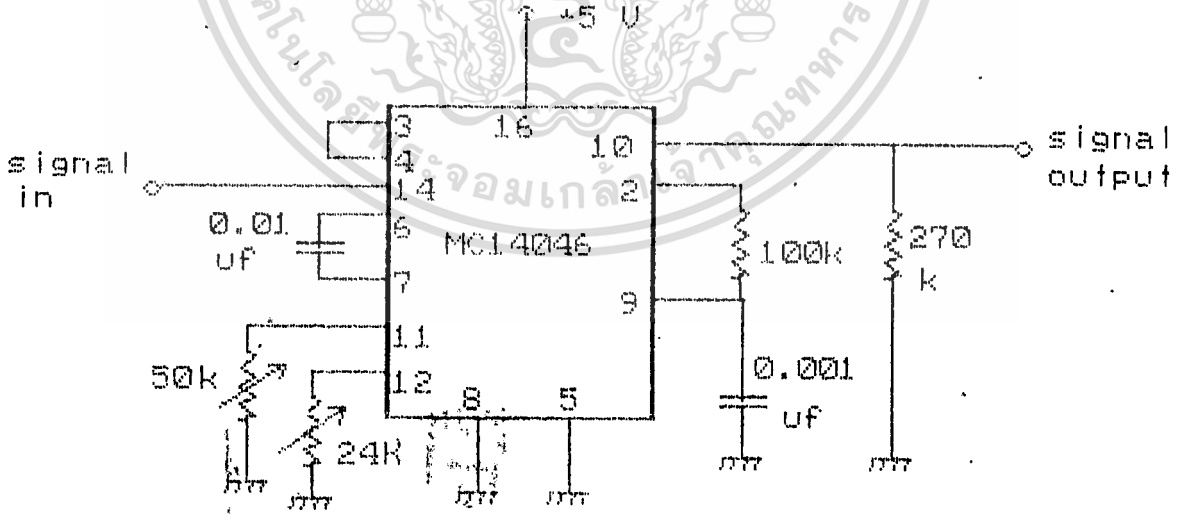
3.4.2 วงจรดีมอดูเลทโดยใช้เฟสล็อคลูป

ในการดีมอดูเลทคลื่นพาหะรองนี้เลือกใช้ไอซีเฟสล็อคลูป เบอร์ MC14046 เพราะว่านิยมใช้กันอย่างแพร่หลายหาซื้อง่าย, ใช้งานได้ง่ายและราคาถูก วงจรภายในของ MC14046 ประกอบด้วยส่วนต่างๆดังรูปที่ 3.17



รูปที่ 3.17 วงจรภายในของ MC14046

ดังที่ได้กล่าวมาแล้วในหัวข้อ2.8.1 ว่าพิสัยการล็อคของเฟสล็อคลูปขึ้นอยู่กับช่วงความถี่ที่วี.ซี.โอ.ทำงานได้ สำหรับวี.ซี.โอ ของMC14046นี้สามารถควบคุมความถี่การทำงานได้ด้วยความต้านทานและตัวเก็บประจุภายนอกดังรูปที่ 3.18 ค่าอุปกรณ์ต่างหาได้จากกราฟในเอกสารประกอบ(Data sheet.)และจากการทดลองปรับค่าโดยที่



รูปที่ 3.18 วงจร MC14046 เฟสล็อกลูป

R_1C_1 จะเป็นตัวแปรที่กำหนดค่าของความถี่สูงของวี.ซี.โอ.ทำการปรับได้โดยเลือกค่า C_1จากกราฟ ให้C_1 = 0.001uF จะได้R_1 อยู่ในช่วง 10^2 K ohm ให้แรงดันอินพุท

ของวี.ซี.โอ.(ขา9) เท่ากับไฟเลี้ยง จากนั้นทำการปรับค่า R_1 ให้ความถี่ของวี.ซี.โอ.มีค่าประมาณ 8.6 KHz ได้ค่าประมาณ 24 K ohm

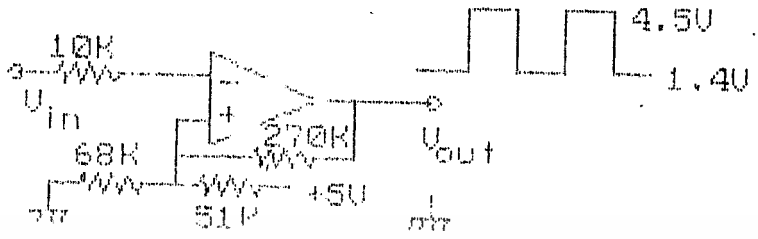
R_2C_1 จะเป็นตัวแปรที่กำหนดค่าขอบความถี่ต่ำของวี.ซี.โอ. การปรับทำโดยให้แรงดันอินพุทของวี.ซี.โอ. เท่ากับ 0 โวล์ทและปรับR_2 จนได้ความถี่ประมาณ 7.6 KHz ได้ค่าประมาณ 15 KHz

ลูปฟิลเตอร์ R_3C_2 เป็นตัวแปรกำหนดพิสัยการเข้าสู่สภาวะล็อค สามารถหาได้จากสมการ $2f_c = 1/\Pi (2\Pi f_1/R_3C_2)^{1/2}$

เมื่อกำหนดให้ C_2= 0.001uF และกำหนดพิสัยของการเข้าสู่สภาวะล็อค จะได้R_1 มีค่าประมาณ 500 K ohm

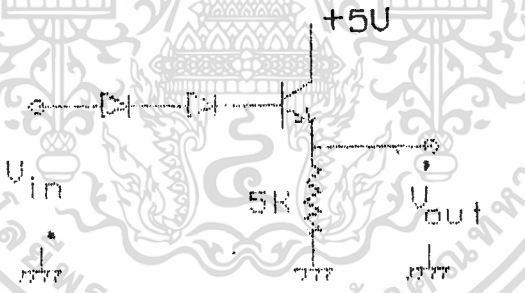
อย่างไรก็ตามค่าที่ได้เป็นเพียงค่าประมาณเท่านั้นการนำไปใช้งานจะต้องมีการปรับค่าต่างๆให้ได้สัญญาณเอ๊าท์พุทที่ดีที่สุด

อย่างไรก็ตามสัญญาณที่ต้องการในการป้อนให้กับวงจรภาคต่อไปซึ่งเป็นวงจรทางดิจิตอล จึงจะต้องมีขนาดใหญ่และมีลักษณะเป็นคลื่นสี่เหลี่ยม (Square Wave)ดังนั้นเอาท์พุทที่ได้จากวงจรกรองช่วงความถี่จะต้องถูกขยายให้มีขนาดขึ้นจนทำให้เป็นคลื่นสี่เหลี่ยมการขยายในลักษณะนี้สามารถทำได้โดยใช้วงจรเปรียบเทียบแรงดัน (Voltage Comparater) แต่เนื่องจากเอ๊าท์พุทของเฟสล็อคลูปอาจมีผลจากสัญญาณรบกวนอยู่บ้างการที่ใช้วงจรเปรียบเทียบแรงดันธรรมดาจะทำให้สัญญาณเอ๊าท์พุทมีค่าผิดพลาดไปดังนั้นจึงเลือกวงจรเปรียบเทียบแรงดันแบบที่มีการป้อนกลับของเอาท์พุททำให้เกิดฮิสเทอรีซิส (Hysteresis) ของแรงดันล้างอิงหรือเรียกอีกอย่างว่าวงจรชมิททริกเกอร์ (Shmittriger circuit) ดังรูปที่ 3.19



รูปที่ 3.19 แสดงวงจรชมิททริกเกอร์

สัญญาณที่ได้จากวงจรชมิททริกเกอร์นี้มีค่าแรงดันออฟเซท (Offset Voltage) ประมาณ 1.5โวล์ทซึ่งมากเกินไปที่จะป้อนให้กับวงจรภาคต่อไปที่เป็นไอซีดิจิตอลทีทีแอลดังนั้นจึงจะต้องใช้วงจรที่จะตัดแรงดันออฟเซทดังรูปที่3.20

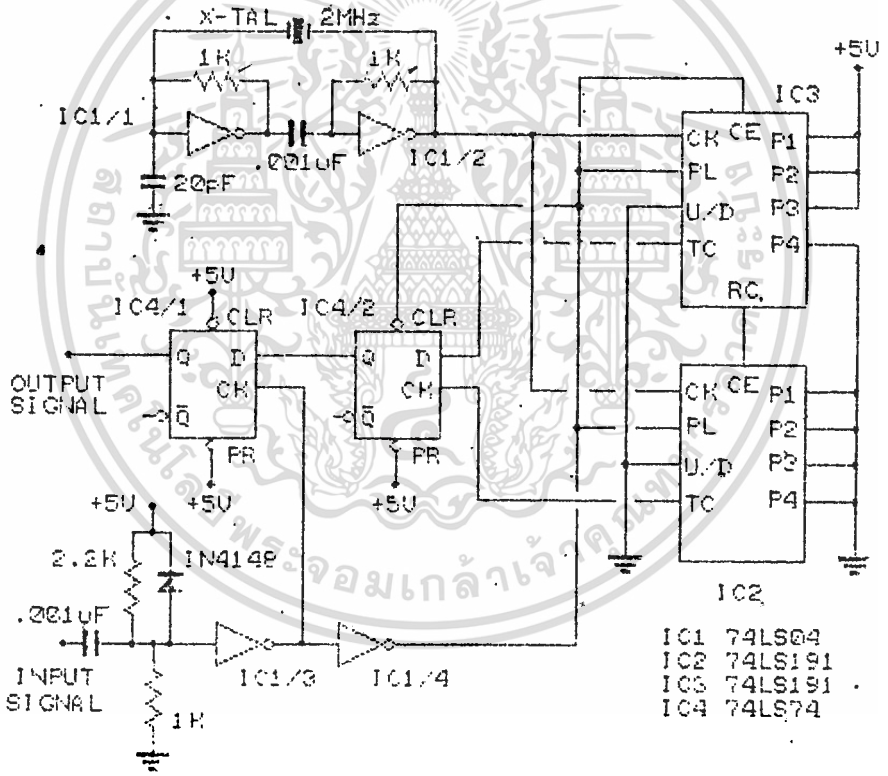


รูปที่ 3.20 วงจรลดแรงดันออฟเซท

3.4.3 การออกแบบวงจรดีมอดูเลทโดยใช้วงจรนับ

จากบล็อกไดอะแกรมในบทที่แล้ว จะพบว่าวงจรดีมอดูเลทนี้ประกอบด้วยวงจรต่างๆดังนี้

1. วงจรสร้างสัญญาณความถี่อ้างอิง
2. วงจรนับ
3. วงจรแลทช์เอาท์พุท
4. วงจรควบคุมลอจิค



รูปที่ 3.21 แสดงวงจรดีมอดูเลทโดยใช้วงจรนับ

จากวงจรในรูป 3.21 ส่วนของอินเวอร์ทเตอร์ (INVERTER) IC1 ทำหน้าที่เป็นวงจรออสซิเลเตอร์ผลิตสัญญาณนาฬิกาอ้างอิงขนาด 2 MHz โดยมีคริสตอลเป็นตัวควบคุมค่าความถี่ให้มีค่าคงที่

สัญญาณนาฬิกาอ้างอิงนี้จะป้อนเข้าสู่วงจรนับ ซึ่งประกอบด้วยวงจรนับขนาด 4 บิทสองตัวต่อให้นับร่วมกันเป็นวงจรนับขนาด 8 บิทและนับแบบซิงโครนัส เราสามารถจะกำหนดให้วงจรนับนี้เริ่มนับที่ค่าเท่าใดก็ได้ตั้งแต่ 0-254 โดยการให้ลอจิคที่ขา P1-P4 ของวงจรนับทั้งสองตัว ค่าเริ่มต้นของการนับนี้สามารถคำนวนได้จาก

$$nc = 0.5fr(1/f1 + 1/f2)$$

fr = ความถี่ของสัญญาณนาฬิกาอ้างอิง = 2 MHz

f1 = ความถี่แทนลอจิคศูนย์ = 7.7 KHz

f2 = ความถี่แทนลอจิคหนึ่ง = 8.5 KHz

$$nc = 247$$

แต่เนื่องจากเราออกแบบให้วงจรนับขึ้น ดังนั้นค่าเริ่มต้นของการนับที่จะตั้งให้กับวงจรนับจะมีค่าเท่ากับ 255-247 = 8

เอาท์พุทของวงจรนับได้จากขา TC (TERNIMAL COUNT) ของวงจรนับทั้งสองตัว เมื่อวงจรนับทั้งสองนับถึงเลข 15 ขา TC ก็จะเปลี่ยนจากลอจิคศูนย์เป็นลอจิคหนึ่ง ทำให้เอาท์พุท Q ของฟลิปฟลอป IC4/2 มีลอจิคหนึ่ง ซึ่งจะทำให้วงจรนับทั้งสองตัวหยุดทำงาน ดังนั้นขา TC ก็จะคงค้างลอจิคหนึ่งเอาไว้ได้

วงจรควบคุมลอจิคจะส่งลอจิคศูนย์ออกไปที่ขา CLR ของ IC4/2 ซึ่งจะเป็นการสั่งให้วงจรนับโหลดค่าเริ่มต้นเข้าไป และเป็นการเคลียร์เอาท์พุท Q ให้เป็นลอจิคศูนย์ ทำให้วงจรนับสามารถเริ่มนับได้ทันทีที่สัญญาณลอจิคศูนย์จากวงจรควบคุมหมดไป

ส่วนของฟลิปฟลอบ IC 4/1 ทำหน้าที่แลทช์ค่าเอาท์พุทของวงจรนับให้ค้างไว้ โดยอาศัยสัญญาณทริกจากวงจรควบคุมลอจิค

ส่วนของวงจรควบคุมลอจิค ประกอบด้วย IC1/3 และ IC1/4 ต่อเป็นวงจรโมโนสเตเบิ้ล โดยอาศัยช่วงขอบขาลงของสัญญาณอินพุทมาทริกให้วงจรโมโนสเตเบิ้ลทำงาน สร้างพัลซ์ขนาดแคบขึ้นมาหนึ่งลูก พัลซ์นี้จะออกมาทางเอท์พุทของ IC1/3 ซึ่งเราใช้ในการแลทช์ค่าเอาท์พุท ขณะเดียวกันสัญญาณพัลซ์นี้จะเข้าสู่ IC1/4 เพื่อสร้างเป็นสัญญาณโหลด และเพื่อให้สัญญาณโหลดมาตามหลังสัญญาณแลทช์ ตามที่ต้องการ

เมื่อมีสัญญาณอินพุทความถี่ f1 เข้าสู่วงจรจะทำให้เกิดสัญญาณไปโหลดให้วงจรนับเริ่มนับใหม่ เนื่องจากสัญญาณอินพุทมีความถี่สูง ดังนั้นในขณะที่วงจรนับยังนับไม่ถึง 255 วงจรควบคุมลอจิคก็จะสั่งให้ IC4/1 แลทช์ค่าเอาท์พุทเอาไว้ ซึ่งจะแลทช์ได้ค่าลอจิค

ศูนย์ ในขณะเดียวกันก็จะสั่งให้วงจรนับเริ่มนับใหม่

แต่เมื่อมีสัญญาณอินพุทความถี่ f2 เข้ามา ซึ่งสัญญาณนี้มีความถี่ต่ำ วงจรนับจึงสามารถนับถึงค่า 255 ได้ทำให้ได้เอาท์พุทของวงจรนับเป็นลอจิคหนึ่ง และวงจรนับหยุดทำงาน ดังนั้นเมื่อวงจรควบคุมส่งสัญญาณแลทช์มาให้ IC4/1 ก็จะทำให้แลทช์ได้เอาท์พุทเป็นลอจิคหนึ่ง

3.5 การออกแบบส่วนประมวลผล และ แสดงผลข้อมูล

ส่วนที่ทำหน้าที่ประมวลผลและแสดงผลข้อมูล อาศัยระบบไมโครคอมพิวเตอร์ 8048 เป็นหัวใจสำคัญ หน้าที่สำคัญของวงจรส่วนนี้ได้แก่ การรับข้อมูล ,การประมวลผล, การแสดงผล เป็นต้น

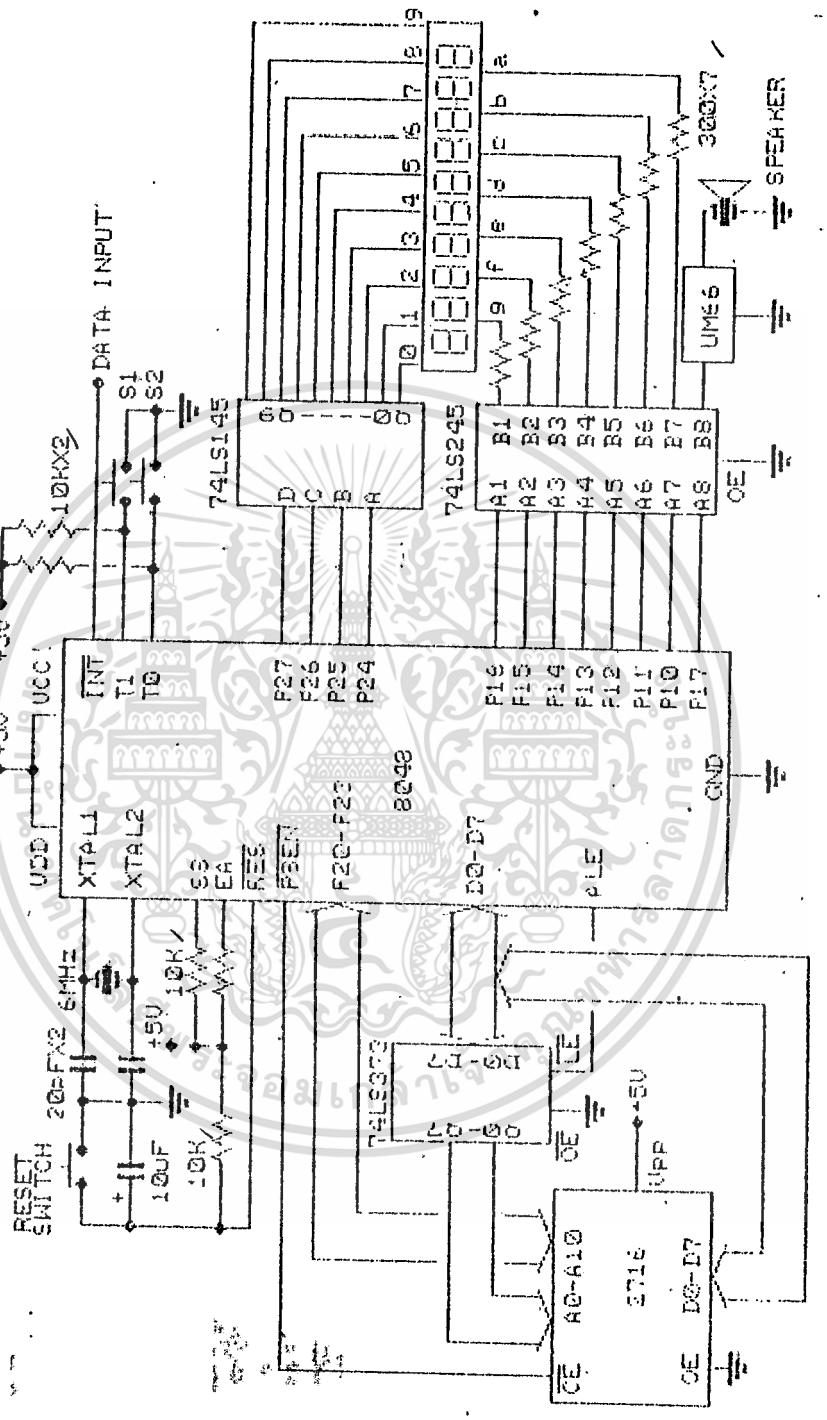
<u>1.รายละเอียดเกี่ยวกับวงจร</u>

วงจรของส่วนประมวลผล และ แสดงผลข้อมูลนี้แสดงไว้ในรูปที่ 3.22

ที่พอร์ท P1 ต่ออยู่กับอินพุทของบัฟเฟอร์ 74LS245 ส่วนที่เอาท์พุทของบัฟเฟอร์ต่ออยู่กับ ส่วนแสดงผล 7 SEGMENT COMMON CATHODE LED DISPLAY โดยบิท P10 ต่ออยู่กับเซกเมนต์ A ของ LED ,บิท P11-P16 ต่ออยู่กับเซกเมนต์ B-G ตามลำดับ ดังนั้น เมื่อเราสั่งให้เอาท์พุทลอจิคหนึ่งออกมาที่บิทใดของพอร์ท P1 ก็จะทำให้เซกเมนต์นั้นติดสว่างขึ้น

ที่บิท P17 ต่ออยู่กับไอซี UM66 ถ้าเราส่งลอจิคหนึ่งออกมาทางขานี้ ก็จะทำให้ไอซี UM66 ทำงานส่งสัญญาณเตือนออกมา เราจะใช้สัญญาณเตือนนี้ เพื่อบอกให้ผู้ใช้ทราบว่ามีข้อมูลชุดใหม่เข้ามาแล้ว เมื่อเราใช้ลอจิคศูนย์แก่ขานี้ เสียงก็จะเงียบไป

ที่พอร์ท P2 ส่วนที่เป็น P24-P27 ต่ออยู่กับไอซีถอดรหัส 1-OF-10 DECODER/DRIVER หมายเลข 74LS145 ที่เอาท์พุทของไอซีนี้ต่อกับขาร่วม (COMMON) ของ LED DISPLAY เมื่อเราเอาท์พุทค่าตัวเลข 0-9 ออกไปทางพอร์ท P24-P27 ไอซี 74LS145 จะถอดรหัสตัวเลขนั้น ทำให้ได้เอาท์พุทของไอซี 74LS145 มีลอจิคศูนย์เป็นการเลือกหลักตัวเลขที่ต้องการ



รูปที่ 3.22 แสดงวงจรประมวลผล และ แสดงผลข้อมูล

ในการแสดงผลด้วย 7 SEGMENT LED DISPLAY ทั้ง 10 หลักนั้นสามารถทำได้โดยใช้หลักการมัลติเพล็กซ์ คือ 8048 จะเอาท์พุทข้อมูลของตัวเลขหลักแรกออกมาทางพอร์ท P1 แล้วจึงเอาท์พุทเลข $00 ออกมาทาง P24-P27 จะทำให้ 7 SEGMENT LED หลักแรกติดสว่างขึ้น จากนั้น 8048 ก็จะเอาท์พุทข้อมูลของหลักที่สองออกมาทางพอร์ท P1 อีก และเอาท์พุทเลข $01 ไปทาง P24-P27 ซึ่งจะทำให้ตัวเลขหลักที่สองติดสว่างแทนหลักแรก ต่อจากนั้นก็จะแสดงผลหลักที่ 3,4 ติดต่อกันไปจนถึงหลักสุดท้ายด้วยวิธีเดียวกัน การแสดงผลนี้ให้ความเร็วในการสแกนสูงจนประสาทตาไม่สามารถสังเกตุเห็นการเปลี่ยนแปลงได้ทัน

ในส่วนอื่นๆที่เหลือนั้น เป็นส่วนประกอบพื้นฐานของระบบไมโครคอมพิวเตอร์ 8048 คือ ประกอบด้วยหน่วยความจำโปรแกรม และ ตัวแลทช์ค่าแอดเดรสแยกออกจากดาต้าบัส

สัญญาณข้อมูลแบบอนุกรมที่ต่ออยู่กับขา INT ของ 8048 จะทำให้ประสิทธิภาพในการรับข้อมูลสูงขึ้น และสามารถออกแบบโปรแกรมได้ง่ายขึ้นด้วย ทั้งนี้เนื่องจาก การส่งข้อมูลแต่ละไบท์ไม่มีการระบุเวลาที่แน่นอน เมื่อมีข้อมูลแบบอนุกรมเข้ามา บิทแรกของข้อมูลที่เราเรียกว่า START BIT (มีลอจิคศูนย์) จะไปอินเตอร์รัพท์ซีพียู ให้หยุดการทำงานอย่างอื่น แล้วเตรียมตัวอ่านข้อมูลแบบอนุกรมอีก 8 บิทที่ส่งตามมา

เมื่อ 8048 ถูกอินเตอร์รัพท์โดยบิทเริ่มของข้อมูล จะทำให้ซีพียูภายใน 8048 หยุดทำงานตามโปรแกรมปกติ แล้วกระโดดไปยังแอดเดรส $003 หน่วยความจำที่แอดเดรสนี้ถูกกำหนดให้เป็นจุดเริ่มต้นของโปรแกรมตอบสนองอินเตอร์รัพท์จากภายนอก (EXTERNAL INTERUPT SERVICE ROUTINE) เมื่อซีพียูทำงานในโปรแกรมตอบสนองอินเตอร์รัพท์เสร็จแล้ว ก็จะย้อนกลับไปทำงานเดิมที่ค้างอยู่ต่อไป

เราสามารถใช้คำสั่งทางซอฟท์แวร์เพื่อสั่งให้อีนาเบิ้ล หรือดิสเอเบิ้ลอินเตอร์รัพท์ได้ โดยใช้คำสั่ง EN I หรือ DIS I ตามลำดับ ในขณะที่ดิสเอเบิ้ลอินเตอร์รัพท์อยู่ เราสามารถใช้ขา INT เป็นขาอินพุทธรรมดาได้ ซึ่งมีคำสั่ง JNI สำหรับตรวจสอบลอจิคที่ขา INT

จากวงจรข้างต้น จะเห็นว่ามีการต่อสวิทช์กดสองตัวเข้ากับขาอินพุท T0,T1 ในขณะที่ยังไม่มีการกดสวิทช์ จะมีลอจิคหนึ่ง ที่ขาอินพุทนี้ จนเมื่อผู้ใช้กดสวิทช์นี้ จะทำให้ลอจิคเปลี่ยนเป็นลอจิคศูนย์ เราสามารถใช้คำสั่ง JNT0 หรือ JT0 ในการตรวจสอบลอจิคที่

ขา T0 และ ใช้คำสั่ง JNT1 หรือ JT1 ในการตรวจสอบลอจิคที่ขา T1

สวิทช์กดทั้งสองตัวนี้ มีไว้สำหรับให้ผู้ใช้กด เพื่อดูหมายเลขโทรศัพท์ในหน่วยความจำ ซึ่งมีหน่วยความจำอยู่สองชุดด้วยกัน เมื่อเรากดสวิทช์ที่ขา T1 โปรแกรมจะสั่งให้แสดงหมายเลขโทรศัพท์ของครั้งที่แล้ว แต่เมื่อกดสวิทช์ที่ขา T0 โปรแกรมจะสั่งให้แสดงหมายเลขโทรศัพท์ของครั้งล่าสุด ทั้งนี้ขึ้นอยู่กับการทำงานของโปรแกรม

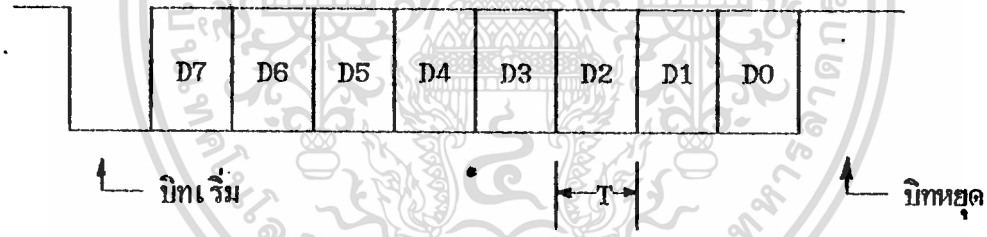
2. รายละเอียดเกี่ยวกับโปรแกรม

ก่อนที่จะออกแบบโปรแกรมจริงนั้น จำเป็นต้องทราบถึงข้อกำหนดบางประการเสียก่อน อันได้แก่

2.1 ข้อกำหนดในการออกแบบ

2.1.1 ความเร็วในการส่งข้อมูลของเครื่องส่ง (BAUD RATE) มีค่าเท่ากับ 600 BUAD (BIT PER SECOND)

2.1.2 ฟอร์แมทของข้อมูล 1 ไบท์จะมีลำดับดังต่อไปนี้



บิทแรกที่ส่งออกมาจากเครื่องส่งเรียกว่า บิทเริ่มต้น ซึ่งจะเป็นลอจิคศูนย์ จากนั้นก็จะตามด้วยข้อมูลหนึ่งไบท์ เริ่มที่บิท 7 และจบที่บิท 0 หลังจากที่ส่งข้อมูลครบหนึ่งไบท์แล้ว สัญญาณก็จะกลับเป็นลอจิคหนึ่งตามเดิม จนกว่าจะมีข้อมูลไบท์ต่อไปเข้ามา ก็จะกลับเป็นลอจิคศูนย์อีกครั้ง

ตามที่กำหนดไว้แล้วว่า อัตราการส่งข้อมูลมีค่าเป็น 600 BAUD ดังนั้น คาบเวลาของแต่ละบิทจะมีค่าประมาณ 1666 uS

2.1.3 ฟอร์แมทของข้อมูล 1 ชุด จะประกอบด้วยส่วนต่างๆดังนี้

2.1.3.1 รหัสซิงโครนัส (SYNCRONOUS CODE) มีค่าเป็น $FF ใช้เป็นสัญญาลักษณ์แทนการเริ่มต้นของการส่งข้อมูลแต่ละชุด ในตอนเริ่มต้นส่งข้อมูลแต่ละชุด จะต้องส่งรหัสซิงโครนัสออกมาก่อนกี่ไบท์ก็ได้ติดๆกัน แต่อย่างน้อยที่สุดจะต้องมีรหัสซิงโครนัสอยู่

หนึ่งไบท์ หลังจากรหัสชิงโครนัสแล้ว ก็จะเป็นข้อมูลอื่นๆ

2.1.3.2 หมายเลขของเพจเจอร์ (PAGER NUMBER) หลังจากที่ได้รับรหัสชิงโครนัส แล้วข้อมูลที่ตามมาอีกสองไบท์ จะเป็นค่าหมายเลขของเพจเจอร์ที่ต้องการติดต่อด้วย หลังจากที่เปรียบเทียบหมายเลขเพจเจอร์ของตนกับหมายเลขที่ได้รับมาแล้ว ถ้าหากหมายเลขตรงกันก็จะอ่านข้อมูลไบท์อื่นๆต่อไป แต่ถ้าหากหมายเลขไม่ตรงกัน มันก็จะไม่สนใจข้อมูลที่ตามมา แต่จะคอยตรวจสอบข้อมูลที่เป็นรหัสชิงโครนัสใหม่

หมายเลขของเพจเจอร์ที่เป็นไบท์สูง (HIGH ORDER BYTE) จะถูกส่งออกมาก่อน ไบท์ต่ำ (LOW ORDER BYTE)

2.1.3.3 ข้อมูลเกี่ยวกับจำนวนหลักของหมายเลขโทรศัพท์ ข้อมูลไบท์นี้จะประกอบด้วยข้อมูลสองส่วน คือ ส่วนที่เป็นสี่บิทสูง (D4-D7) จะบอกถึงจำนวนหลักของรหัสท้องถิ่น (AREA CODE) ส่วนบิท (D0-D3) นั้นจะบอกถึงจำนวนหลักของหมายเลขโทรศัพท์

หลังจากที่รู้จำนวนหลักของหมายเลขโทรศัพท์แล้ว ก็จะรู้ว่าจะต้องอ่านข้อมูลต่อไปอีกกี่ไบท์ และข้อมูลเหล่านั้นแบ่งเป็นหมายเลขโทรศัพท์และรหัสท้องถิ่นอย่างไร

2.1.3.4 รหัสท้องถิ่นและหมายเลขโทรศัพท์ รหัสท้องถิ่นจะถูกส่งออกมาก่อนแล้วจึงต่อด้วยหมายเลขโทรศัพท์ ข้อมูลแต่ละไบท์จะถูกแบ่งออกเป็นข้อมูลสองหลักๆละ 4 บิท ข้อมูลที่อยู่ใน D4-D7 จะเป็นหมายเลขตัวแรก ส่วนข้อมูลใน D0-D3 จะเป็นหมายเลขตัวถัดมา

เพื่อให้เครื่องรับสามารถรับข้อมูลได้ถูกต้องมากขึ้น จึงได้ใช้วิธีส่งข้อมูลซ้ำหลายๆครั้ง แล้วนำข้อมูลเหล่านั้นมาตรวจสอบ ก็จะได้ข้อมูลที่มีโอกาสผิดพลาดน้อยลง ในที่นี้ได้กำหนดให้ส่งข้อมูลแต่ละไบท์ซ้ำกันสามครั้ง ยกเว้นแต่รหัสชิงโครนัสเท่านั้น ที่จะมีกี่ไบท์ก็ได้

ตัวอย่างเช่น เมื่อเราต้องการให้เจ้าของเพจเจอร์หมายเลข 4185 ติดต่อกลับมาที่หมายเลข 2820704 ก็จะต้องส่งข้อมูลหนึ่งชุดออกไปดังนี้

$FF $10 $10 $10 $59 $59 $59 $07 $07 $07 $28 $28 $28 $20 $20 $20 $70 $70 $70 $40 $40 $40

ข้อมูลแต่ละชุดจะเริ่มต้นด้วยรหัสชิงโครนัส จำนวนกี่ไบท์ก็ได้ (อย่างน้อยที่สุดหนึ่งไบท์) ส่วนตัวเลขหกไบท์ต่อมาคือหมายเลขของเพจเจอร์ไบท์สูงและไบท์ต่ำตามลำดับ ($1059 เป็นตัวเลขฐานสิบหก ซึ่งจะเท่ากับ 4185 เลขฐานสิบ) ข้อมูลสามไบท์ต่อมาจะ

บอกให้รู้ว่า ข้อมูลที่ตามมาไม่มีรหัสท้องถิ่น มีแต่หมายเลขโทรศัพท์ 7 หลัก ส่วนข้อมูล 12 ไบท์สุดท้ายจะเป็นหมายเลขโทรศัพท์เพียงอย่างเดียว

2.1.4 การแบ่งส่วนพื้นที่หน่วยความจำข้อมูลภายใน เพื่อให้มีพื้นที่เป็นสัดส่วนในการเก็บข้อมูลต่างๆ ได้แก่

2.1.4.1 หน่วยความจำแอดเดรส 20-29 (BUFF1) เป็นที่เก็บข้อมูลในการแสดงผลเมื่อมีการกดสวิทช์ที่ขา T0 เก็บหมายเลขโทรศัพท์ครั้งล่าสุด

2.1.4.2 หน่วยความจำแอดเดรส 2A-33 (BUFF2) เป็นที่เก็บข้อมูลในการแสดงผลเมื่อมีการกดสวิทช์ที่ขา T1 เก็บหมายเลขโทรศัพท์ครั้งที่แล้ว

2.1.4.3 หน่วยความจำแอดเดรส 34-3D (BUFF4) เป็นที่เก็บข้อมูลที่ได้รับเข้ามา ก่อนที่จะเคลื่อนย้ายข้อมูลไปเก็บไว้ในหน่วยความจำ BUFF1

2.1.4.4 หน่วยความจำแอดเดรส 3E-3F (TD1) เป็นค่าตัวเลขที่ใช้ในการหน่วงเวลาตอนแสดงผลให้นานประมาณ 3 วินาที หลังจากที่ปล่อยสวิทช์ที่ขา T0 แล้ว

2.1.4.5 หน่วยความจำแอดเดรส 1B-1C (TD2) เช่นเดียวกับ TD1 แต่จะหน่วงเวลาหลังจากกดสวิทช์ที่ขา T1

2.1.4.6 หน่วยความจำแอดเดรส 1D-1F (BUFF3) เก็บข้อมูลสามไบท์ที่รับเข้ามา เพื่อให้โปรแกรม DECODE นำไปตรวจสอบให้ได้ข้อมูลถูกต้องเพียงไบท์เดียว

2.2 การออกแบบโปรแกรม

เพื่อให้สามารถเข้าใจหลักการของโปรแกรมได้ง่ายขึ้น ก็จะแยกอธิบายโปรแกรมออกเป็นส่วนๆ ดังนี้

2.2.1 โปรแกรมอ่านข้อมูลแบบอนุกรม

ส่วนของโปรแกรมที่ทำหน้าที่อ่านข้อมูลแบบอนุกรม ที่เข้ามาทางขา INT นั้นจะประกอบด้วยโปรแกรมย่อยหลายโปรแกรมทำงานร่วมกัน ดังนี้

เมื่อเริ่มมีข้อมูลเข้ามา บิทเริ่มต้นของข้อมูลแต่ละไบท์จะอินเตอร์รัพท์ 8048 ให้กระโดดไปยังโปรแกรมตอบสนองอินเตอร์รัพท์ที่แอดเดรส 003 ซึ่งในโปรแกรมตอบสนองอินเตอร์รัพท์นี้จะกำหนดค่าพารามิเตอร์เริ่มต้นต่างๆสำหรับอ่านข้อมูลหนึ่งไบท์ จากนั้นก็จะดิสเอเบิ้ลอินเตอร์รัพท์ แล้วสั่งให้วงจรนับภายใน 8048 ทำงานส่งสัญญาณอินเตอร์รัพท์ทุกๆคาบเวลา 1666uS (เนื่องจากเรากำหนดให้อัตราการส่งข้อมูลเป็น 600 BAUD จะได้ว่าคาบเวลาของข้อมูลแต่ละบิทจะนานเท่ากับ 1666 uS) เมื่อวงจรนับส่งสัญญาณอิน

เตอร์รัพท์ 8048 ก็หมายความว่าถึงเวลาที่จะอ่านข้อมูลบิทต่อไปแล้ว โดยจะอาศัยโปรแกรมย่อยชื่อ RDBIT สำหรับอ่านข้อมูลเข้าไป

เมื่ออ่านข้อมูลครบทั้ง 8 บิทแล้วก็จะเก็บข้อมูลนั้นไว้ในหน่วยความจำ แต่เนื่องจากในการส่งข้อมูลแต่ละไบท์จะส่งออกมาซ้ำๆกันสามครั้ง จึงต้องเขียนโปรแกรมให้อ่านข้อมูลทั้ง 3 ครั้ง ไปเก็บในหน่วยความจำ แล้วจึงตรวจสอบข้อมูลจนได้ข้อมูลที่ถูกต้องออกมาเพียงไบท์เดียว

รายละเอียดของโปรแกรม จะแยกอธิบายเป็นส่วนๆ ดังนี้

2.2.1.1 โปรแกรม RDBIT เป็นโปรแกรมย่อยสำหรับอ่านข้อมูลที่เข้ามา โดยในการออกแบบโปรแกรมนี้ ต้องการให้สามารถอ่านข้อมูลได้ถูกต้องมากขึ้น จึงอาศัยหลักการอ่านข้อมูลซ้ำหลายๆครั้ง แล้วดูแนวโน้มของข้อมูลที่ได้ว่าควรจะเป็นลอจิคหนึ่ง หรือลอจิคศูนย์

ตัวโปรแกรม RDBIT เป็นดังนี้

```
RDBIT   MOV   R3,#00
        MOV   R7,#29H
RB1     JNI   ZERO
        INC   R3
        NOP
RB2     DJNZ  R7,RB1
        MOV   A,R3
        ADD   A,#0EBH
        RET
ZERO    JMP   RB2
```

เมื่อเรากำหนดให้อัตราการส่งข้อมูลเท่ากับ 600 BUAD หรือ มีคาบเวลาของแต่ละบิทข้อมูลเป็น 1666 uS ดังนั้นเวลาทั้งหมดที่โปรแกรม RDBIT ใช้ในการอ่านข้อมูลแต่ละบิทนั้น กำหนดให้มีค่าประมาณ 20%-60% ของคาบเวลาแต่ละบิท ในการคำนวนค่าเวลาที่โปรแกรมใช้ไป จะอาศัยสมการต่อไปนี้

เวลา = 2.5uS x (6n+10) ; n = จำนวนครั้งในการอ่านข้อมูล กำหนดไว้ใน R7

ผลลัพธ์ที่ได้จากโปรแกรมนี้ จะอยู่แฟลกตัวทด C โดยเมื่อแฟลกตัวทดเป็นศูนย์ หมายถึงข้อมูลที่เข้ามาเป็นลอจิคศูนย์ แต่ถ้าแฟลกตัวทดเป็นหนึ่ง แสดงว่าข้อมูลที่เข้ามาเป็นลอจิคหนึ่ง

2.2.1.2 โปรแกรมตอบสนองอินเตอร์รัพท์จากภายนอก จะเป็นโปรแกรมย่อยทำหน้าที่เตรียมพร้อมสำหรับอ่านข้อมูล ซึ่งมีรายละเอียดดังนี้

```
INTI        SEL  RB0
            MOV  R5,A
            SEL  RB1
            MOV  R5,#TC4
DELAY2      DJNZ R5,DELAY2
            MOV  A,#TCT
            MOV  T,A
            STRT T
            CALL RDBIT
            JC   RETURN
            MOV  R5,#08H
            DIS  I
            SEL  RB0
            MOV  A,R5
            RETR
RETURN      STOP TCNT
            SEL  RB0
            MOV  A,R5
            RETR
```

เมื่อเริ่มเข้าสู่โปรแกรมตอบสนองอินเตอร์รัพท์ จำเป็นต้องเก็บค่าเดิมของแอคคิวมูเลเตอร์ไว้ใน R5 เสียก่อน และเปลี่ยนไปใช้รีจีสเตอร์แบงค์หนึ่ง จากนั้นจะหน่วงเวลาประมาณ 20% ของคาบเวลา 1666uS และส่งค่าเริ่มต้นให้แก่วงจรนับภายใน สั่งให้วงจรนับเริ่มทำงาน แล้วจึงอ่านข้อมูลที่ขา INT เพื่อตรวจสอบให้แน่นอนว่า การอินเตอร์

รัพท์นั้นไม่ได้เกิดจากสัญญาณรบกวน ถ้าหากเป็นบิทเริ่มต้นจริงก็จะดิสเอเบิ้ลอินเตอร์รัพท์แล้วจึงออกจากโปรแกรมตอบสนองอินเตอร์รัพท์นี้

2.2.1.3 โปรแกรมตอบสนองอินเตอร์รัพท์จากวงจรนับ จะทำงานเมื่อวงจรนับนับเวลาครบ 1666uS ทำให้ 8048 อ่านข้อมูลเข้ามาหนึ่งบิท แล้วเก็บข้อมูลไว้ในรีจีสเตอร์จนครบสามไบท์ จากนั้นจึงนำข้อมูลนั้นมาตรวจสอบด้วยโปรแกรมย่อย DECODE จนได้ข้อมูลที่ถูกต้อง จึงจะเซทแฟลก F1 ให้เป็นหนึ่ง เพื่อให้โปรแกรมภายนอกสามารถตรวจสอบได้ว่า ได้รับข้อมูลเรียบร้อยแล้วหรือไม่

```
TCNTI       MOV   R7,A
            STOP  TCNT
            MOV   A,#TCI
            MOV   T,A
            MOV   R3,#TC3
TDELAY      DJNZ  R3,TDELAY
            STRT  T
            CALL  RDBIT
            MOV   A,R2
            RLC   A
            MOV   R2,A
            DJNZ  R5,OUT1
            MOV   @R0,A
            INC   A
            JZ    OUT3
            INC   R0
            DJNZ  R6,OUT
            CALL  DECODE
OUT3        MOV   R6,#03
            MOV   R0,#BUFF3
            CPL   F1
```

```
                MOV   A,R2
                SEL   RB0
                MOV   R2,A
OUT             STOP  TCNT
OUT2            JNI   OUT2
                EN    I
OUT1            SEL   RB0
                MOV   A,R7
                RETR
```

ในส่วนของโปรแกรมตอบสนองอินเตอร์รัพท์ จะเห็นว่ามีค่าตัวแปรหลายๆตัว ซึ่งการกำหนดค่าตัวแปรเหล่านี้เกี่ยวข้องกับอัตราการส่งข้อมูล เช่น TC3,TC4,TCT เป็นต้น ค่าตัวแปรเหล่านี้สามารถคำนวนได้จากสมการต่อไปนี้

TCT = ค่าจำนวนเต็มของ 1/(RATE x 80uS)

TC3 = [(เศษที่ได้จาก TCT x 32)-11]/2

TC4 = [(80000/RATE)-11]/2

ในการทดลองนี้ กำหนดอัตราการส่งเป็น 600 BUAD ดังนั้นจึงสามารถค่าตัวแปรเหล่านี้ได้ดังนี้

TCT = 1/(600 x 80uS) = 20.833 เอาเฉพาะจำนวนเต็มจะได้ TCT = 20

TC3 = [(.8333 x 32)-11]/2 = 7.833 ปัดขึ้นเป็นจำนวนเต็มจะได้ TC3 = 8

TC4 = [(80000/600)-11]/2 = 61.16 ปัดเป็นจำนวนเต็มจะได้ TC4 = 61

2.2.1.4 โปรแกรมย่อย DECODE ทำหน้าที่ตรวจสอบข้อมูลที่ส่งมาซ้ำกันสามครั้ง ให้ได้ข้อมูลที่ถูกต้องข้อมูลเดียว โดยอาศัยหลักการเปรียบเทียบตัวเลขแต่ละบิทระหว่างข้อมูลสามไบท์ ถ้าบิทนั้นๆมีลอจิคเหมือนกันสองบิทขึ้นไป ก็จะถือว่าข้อมูลที่เหมือนกันนั้นเป็นข้อมูลที่ถูกต้อง

```
DECODE          MOV   R0,#BUFF3
```

```
        MOV   A,@R0
        MOV   R2,A
        INC   R0
        MOV   A,@R0
        MOV   R6,A
        XRL   A,R2
        INC   R0
        ANL   A,@R0
        MOV   R5,A
        MOV   A,R2
        ANL   A,R6
        ORL   A,R5
        MOV   R2,A
        RET
```

2.2.2 โปรแกรมแสดงผลข้อมูล ตรวจสอบสวิทช์

หน้าที่หลักของโปรแกรมส่วนนี้ ได้แก่ การตรวจสอบแฟก F1 ว่ามีข้อมูลเข้ามาหรือไม่ ,การตรวจสอบสวิทช์ซึ่งต่ออยู่กับขา T0 และ T1 ว่ามีการกดสวิทช์หรือไม่ เมื่อมีการกดสวิทช์ก็จะนำข้อมูลในหน่วยความจำไปแสดงผลทาง 7 SEGMENT LED DISPLAY จำนวน 10 หลัก โดยใช้หลักการมัลติเพล็กซ์ และเมื่อผู้ใช้ปล่อยมือจากสวิทช์แล้ว โปรแกรมจะยังคงยืดเวลาการแสดงผลออกไปอีกประมาณ 3 วินาที

โปรแกรมที่ทำหน้าที่เหล่านี้จะจะประกอบด้วยโปรแกมย่อยๆต่างๆดังนี้

2.2.2.1 โปรแกรม DISP เป็นโปรแกรมที่นำข้อมูลจากหน่วยความจำไปแสดงผลทาง 7 SEGMENT LED DISPLAY โดยใช้หลักการมัลติเพล็กซ์ที่กล่าวไปแล้ว ตัวโปรแกรม DISP จะเป็นดังนี้

```
DISP        MOV   R0,#BUFF1+9
LOOP        MOV   R6,#0AH
LOOP2       MOV   A,@R0
            MOVP3 A,@A
```

```
                OUTL P1,A
                MOV  A,R6
                DEC  A
                SWAP A
                OUTL P2,A
                MOV  A,#20H
DELAY           JF1  END
                DEC  A
                JNZ  DELAY
                DEC  R0
                DJNZ R6,LOOP2
END             CLR  A
                OUTL P1,A
                RET
```

ส่วนที่มีคำสั่ง JF1 ทำหน้าที่ตรวจสอบว่ามีข้อมูลเข้ามาหรือไม่ ถ้าหากมีข้อมูลเข้ามาก็จะออกจากโปรแกรมย่อยนี้ทันที

เราสามารถกำหนดแอดเดรสของหน่วยความจำที่เก็บข้อมูลได้ โดยกำหนดไว้ในรีจีสเตอร์ R0 เช่นเมื่อผู้ใช้กดสวิทช์ที่ขา T0 โปรแกรมก็จะแสดงข้อมูลที่อยู่ในแอดเดรส BUFF1 แต่ถ้าหากผู้ใช้กดสวิทช์ที่ขา T1 โปรแกรมก็จะแสดงข้อมูลที่อยู่ในแอดเดรส BUFF2 แทน โดยใส่ค่าแอดเดรสของ BUFF2 ลงในรีจีสเตอร์ R0

คำสั่ง MOVP3 A,@A เป็นคำสั่งนำรหัสตัวเลขไปเปิดตารางที่หน่วยความจำโปรแกรมเพจสาม เพื่อให้ได้เป็นรหัสของเซกเมนท์ในการแสดงผลออกมา

2.2.2.2 โปรแกรม WAITCH ทำหน้าที่ตรวจสอบแฟลก F1 ว่ามีข้อมูลเข้ามาหรือไม่ ถ้าหากมีข้อมูลเข้ามา ก็จะนำข้อมูลนั้นไปเก็บไว้ในแอคคิวมูเลเตอร์ และเคลียร์แฟลก F1 เพื่อแสดงว่าได้อ่านข้อมูลไปแล้ว จากนั้นจึงออกจากโปรแกรม WAITCH นี้

แต่ถ้าหากยังไม่มีข้อมูลใหม่ก็จะตรวจสอบการกดสวิทช์ ถ้าหากมีการกดสวิทช์ก็จะต้องดูว่าสวิทช์ที่กดนั้นเป็นสวิทช์ตัวใด แล้วจึงนำข้อมูลในหน่วยความจำแต่ละส่วนไปแสดงผลด้วยโปรแกรมย่อย DISP และเมื่อผู้ใช้ปล่อยมือจากสวิทช์แล้ว โปรแกรมจะหน่วง

เวลาการแสดงผลให้ค้างต่อไปอีกนานประมาณ 3 วินาที

ถ้าหากไม่มีการกดสวิทช์เลย โปรแกรมก็จะย้อนกลับไปตรวจสอบแฟลก F1 อีกวนอยู่เช่นนี้ไม่สามารถออกจากโปรแกรม WAITCH ได้ จนกว่าจะมีข้อมูลเข้ามา

ตัวโปรแกรม WAITCH จะเป็นดังนี้

```
WAITCH      JF1   DATAIN
            SEL   RB1
            MOV   A,@R1
            JNZ   NTDLO
            INC   R1
            MOV   A,@R1
            JNZ   NTDHI
            DEC   R1
            SEL   RB0
            JMP   W2
NTDHI       INC   @R1
            DEC   R1
NTDLO       INC   @R1
            MOV   A,#TD1
            XRL   A,R1
            SEL   RB0
            JZ    DISPB1
            JNZ   DISPB2
W2          JNT0  KEY
            JT1   WAITCH
            SEL   RB1
            MOV   R1,#TD2+1
            JMP   SETCK
DISPB2      MOV   R0,#BUFF2+9
```

```
                CALL LOOP
                JMP  WAITCH
KEY             SEL  RB1
                MOV  R1,#TD1+1
SETCK           MOV  @R1,#HITD
                DEC  R1
                MOV  @R1,#LOTD
                SEL  RB0
                JMP  WAITCH
DISPB1          CALL DISP
                JMP  WAITCH
DATAIN          MOV  A,R2
                CLR  F1
                RET
```

2.2.2.3 โปรแกรม PUTBF ทำหน้าที่นำข้อมูลที่รับเข้ามา ซึ่งเป็นตัวเลขรหัสทางไกล หรือหมายเลขโทรศัพท์ ไปเก็บไว้ในหน่วยความจำ โดยจะต้องแยกข้อมูลที่ส่งเข้ามาหนึ่งไบท์นั้น ออกเป็นตัวเลขสองหลักเสียก่อน แล้วตรวจสอบค่าตัวเลขว่าเป็นเลข 0-9 หรือไม่ ถ้าหากไม่ใช่ก็แสดงว่าคาจมีการผิดพลาดจากการรับข้อมูล ข้อมูลทั้งชุดนี้ไม่สามารถนำมาแสดงผลได้ ต้องยกเลิกข้อมูลทั้งหมด แล้วเตรียมตัวรับข้อมูลชุดใหม่

ตัวโปรแกรม PUTBF จะเป็นดังนี้

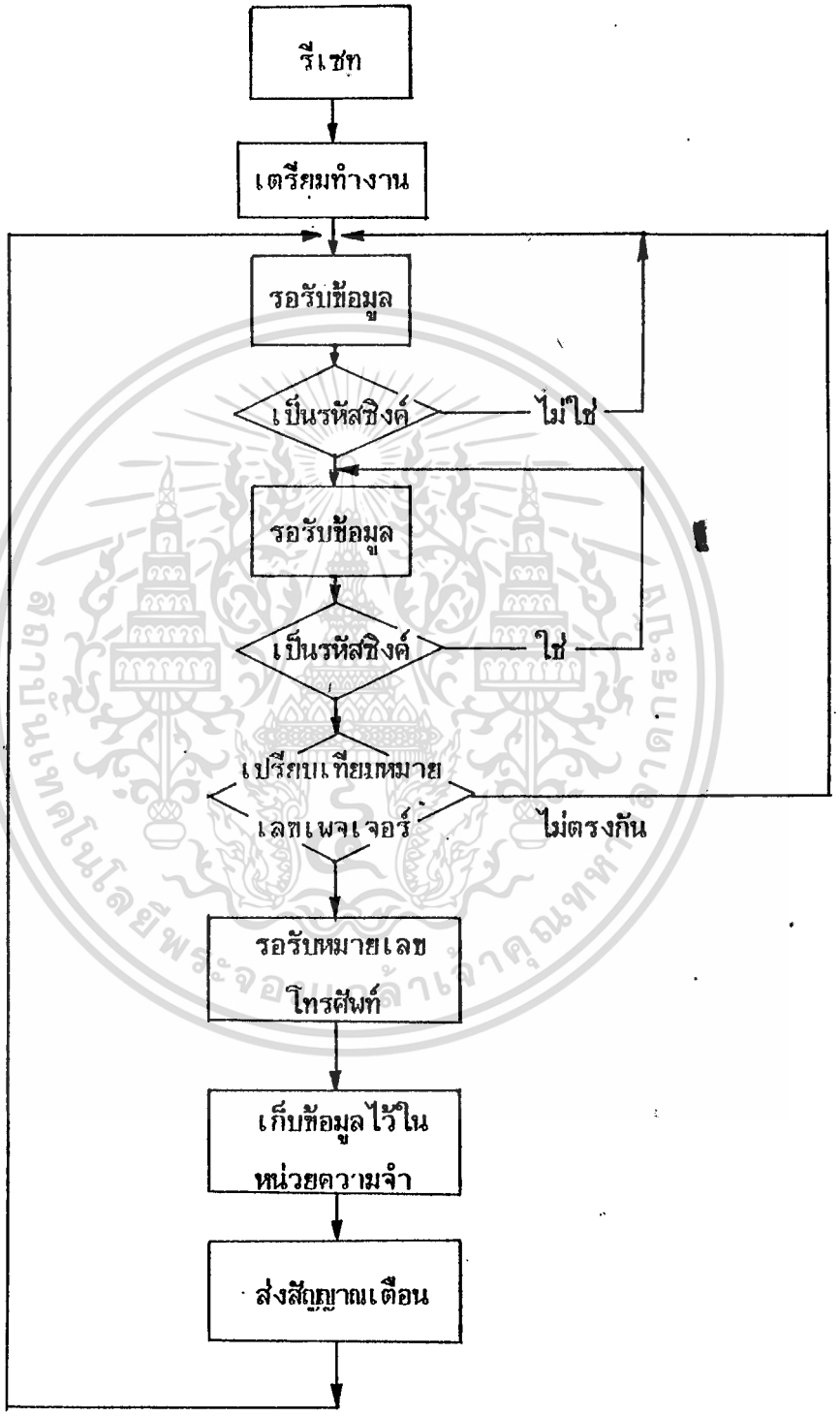
```
PUTBF           CLR  F0
                MOV  A,R3
                JZ   BACK
AGAIN           CALL WAITCH
                INC  A
                JZ   BACK
                MOV  A,R2
                SWAP A
```

```
                ANL   A,#0FH
                MOV   R6,A
                ADD   A,#0F6H
                JC    NDIGIT
                MOV   A,#0FH
                ANL   A,R2
                MOV   R2,A
                ADD   A,#0F6H
                JC    NDIGIT
                MOV   A,R6
                MOV   @R1,A
                INC   R1
                DJNZ  R3,NZERO
                RET
NZERO           MOV   A,R2
                MOV   @R1,A
                INC   R1
                DJNZ  R3,AGAIN
BACK            CPL   F0
                RET
NDIGIT          MOV   R2,#0FEH
                RET
```

ในกรณีที่มีการผิดพลาดเกิดขึ้น ค่าของรีจีสเตอร์ R2 จะมีค่าเป็น FE โปรแกรมหลักที่เรียกใช้งานโปรแกรมย่อย PUTBF นี้จะต้องตรวจสอบค่าในรีจีสเตอร์ R2 ก่อน

2.2.3 โปรแกรมหลัก จะควบคุมการทำงานของวงจรส่วนประมวลผล และแสดงผลนี้ทั้งหมด โดยจะอาศัยโปรแกรมย่อยต่างๆที่กล่าวถึงไปแล้ว

ลำดับขั้นการทำงานของโปรแกรมหลัก จะเป็นไปตาม FLOWCHART ต่อไปนี้



รูปที่ 3.23 แสดงลำดับขั้นการทำงานของโปรแกรมหลัก

ในตอนเริ่มต้น หลังจากที่ 8048 ถูกรีเซท จะต้องเตรียมตัวสำหรับทำงานต่างๆ เช่น เซทค่ารีจีสเตอร์ต่างๆ เซทแฟลก อีนาเบิ้ลอินเตอร์รัพท์ เป็นต้น แล้วจึงรอรับข้อมูลที่จะเข้ามา เมื่อมีข้อมูลเข้ามาก็จะตรวจสอบดูว่าข้อมูลนั้นเป็นรหัสชิงโครนัสหรือไม่ แล้วจะรอรับข้อมูลตัวถัดไป ถ้าหากข้อมูลที่ตามหลังไม่ใช่รหัสชิงโครนัสก็แสดงว่าเป็นหมายเลขเพจเพจอร์ โปรแกรมก็จะนำตัวเลขนี้ไปเปรียบเทียบกับหมายเลขของตน เมื่อหมายเลขตรงกันก็จะรอรับข้อมูลที่เป็นหมายเลขโทรศัพท์ต่อไป จนเมื่อได้รับข้อมูลครบแล้วก็จะส่งสัญญาณเตือนผู้ใช้ ผู้ใช้สามารถดูข้อมูลได้โดยการกดสวิทช์ที่ขา TO หรือ ที่ขา T1

โปรแกรมหลักจะมีลักษณะดังนี้

```
MAIN        MOV   R0,#BUFF1
            MOV   R1,#BUFF2
            MOV   R7,#READY
            MOV   R3,#0AH
NEXT        MOV   A,R7
            MOVP  A,@A
            MOV   @R0,A
            MOV   @R1,A
            INC   R0
            INC   R1
            INC   R7
            DJNZ  R3,NEXT
            MOV   R2,#TC1
NEXT2       MOV   R3,#TC2
NEXT3       CALL  DISP
            DJNZ  R3,NEXT3
            DJNZ  R2,NEXT2
CLEAR       CLR   A
            OUTL  P1,A
            CLR   F1
```

```
                EN   I
                EN   CNTI
                SEL  RB1
                MOV  R1,#TD1+1
                MOV  @R1,#HITD
                DEC  R1
                MOV  @R1,#LOTD
                MOV  R6,#03H
                MOV  R0,#BUFF3
                SEL  RB0
MAIN3           CLR  F0
MAIN2           CALL WAITCH
                JF0  SYNCIN
                INC  A
                JNZ  MAIN2
                CPL  F0
                JMP  MAIN2
SYNCIN          INC  A
                JZ   MAIN
                MOV  A,R2
                XRL  A,#HIBYTE
                JZ   PAGER
                CALL WAITCH
                JMP  MAIN3
PAGER           CALL WAITCH
                XRL  A,#LOBYTE
                JNZ  MAIN3
                CALL WAITCH
```

```
INC  A
JZ   MAIN2
MOV  A,R2
SWAP A
ANL  A,#0FH
MOV  R3,A
MOV  R3,A
MOV  A,R2
ANL  A,#0FH
MOV  R4,A
ADD  A,R3
ADD  A,#0F6H
JC   MAIN3
MOV  R1,#BUFF4
CALL PUTBF
MOV  A,R2
INC  A
JZ   TSET1
INC  A
JZ   MAIN3
MOV  A,#HIFEN
MOV  @R1,A
INC  R1
JF0  SKIP
MOV  A,R2
MOV  @R1,A
INC  R1
DEC  R4
```

```
SKIP        MOV   A,R4
            MOV   R3,A
            CALL  PUTBF
            MOV   A,R2
            INC   A
TEST1       CLR   F0
            JNZ   TEST2
            CPL   F0
            JMP   MAIN2
            INC   A
            JZ    MAIN3
EMPTY       MOV   A,#BUFF4+10
            XRL   A,R1
            JZ    FINISH
            MOV   A,#BLANK
            MOV   @R1,A
            INC   R1
            JMP   EMPTY
FINISH      ORL   P1,#80H
            MOV   R0,#BUFF1
            MOV   R1,#BUFF2
            MOV   R6,#0AH
MOVE1       MOV   A,@R0
            MOV   @R1,A
            INC   R0
            INC   R1
            DJNZ  R6,MOVE1
            MOV   R0,#BUFF1
```

```
                MOV   R1,#UFF4
                MOV   R6,#0AH
MOVE2           MOV   A,@R1
                MOV   @R0,A
                INC   R0
                INC   R1
                DJNZ  R6,MOVE2
                JMP   MAIN3
```

บทที่ 4 การทดลอง และผลการทดลอง

หลังจากที่ได้ออกแบบวงจร และ โปรแกรมต่างๆเป็นที่เรียบร้อยแล้ว ในการทดลองวงจรแต่ละส่วนจะกระทำบนแผ่นทดลองวงจรโปรโตบอร์ดก่อน เมื่อได้ผลเป็นที่พอใจแล้วจึงประกอบลงบนแผ่นวงจรพิมพ์

การทดลองจะแยกเป็นส่วนย่อยๆ เพื่อตรวจสอบการทำงานเฉพาะส่วนก่อน จากนั้นจึงจะประกอบระบบทั้งหมดเข้าด้วยกัน แล้วตรวจสอบระบบทั้งหมดอีกครั้ง ในการทดลองสำหรับแต่ละวงจรนั้นจะใช้วิธีการทดลองแตกต่างกันไป ดังต่อไปนี้

4.1 การทดลองเกี่ยวกับวงจรรับข้อมูลทางโทรศัพท์

4.1.1 การทดลองวงจรตรวจสอบสัญญาณเรียก

เป็นการทดลองใช้ทดลองต่อสายโทรศัพท์เข้ากับวงจรตรวจสอบสัญญาณเรียก โดยกำหนดให้สัญญาณ HOLD เป็นลอจิคศูนย์ เพื่อให้รีเลย์หยุดทำงานและขารีเซทของไอซี 555 ในวงจรนี้จะมีมีลอจิคหนึ่ง จากนั้นใช้โทรศัพท์ภายนอกติดต่อเข้ามายังวงจรตรวจสอบสัญญาณเรียก แล้ววัดแรงดันเอาท์พุทที่ขา 3 ของไอซี 555

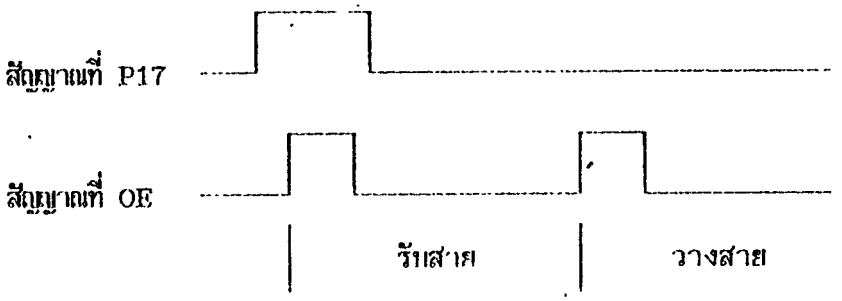
เมื่อมีสัญญาณกระดิ่งเข้ามา แรงดันเอาท์พุทที่ขา 3 ของไอซี 555 เปลี่ยนจากเดิมที่เป็นลอจิคศูนย์ กลายเป็นลอจิคหนึ่ง และเมื่อสัญญาณกระดิ่งหมดลง แรงดันที่ขา 3 ก็ยังคงค้างอยู่ประมาณ 0.5 วินาที แล้วจึงเปลี่ยนเป็นลอจิคศูนย์ดังเดิม จนเมื่อมีสัญญาณกระดิ่งดังอีกครั้งหนึ่ง ขาเอาท์พุทก็กลับมีลอจิคหนึ่งอีก

วงจรสามารถทำงานได้ตามปกติ และ ผลที่ได้เป็นไปตามที่ต้องการ

4.1.2 การทดลองวงจรพักสาย

เป็นการทดลองการรับโทรศัพท์เมื่อมีสัญญาณไฟฟ้ามากระตุ้นวงจร โดยได้ทำการทดลองต่อวงจรพักสายเข้ากับสายโทรศัพท์ แล้วใช้โทรศัพท์ภายนอกติดต่อเข้ามา เมื่อมีสัญญาณกระดิ่งดังขึ้น ก็จะส่งลอจิคศูนย์ออกมาทางขา P17 ของ 8048 แล้วส่งสัญญาณ OE ออกมาทาง 74LS259 ทำให้วงจรพักสายทำงาน หน้าสัมผัสของรีเลย์ต่อวงจรทางด้านสายโทรศัพท์ได้

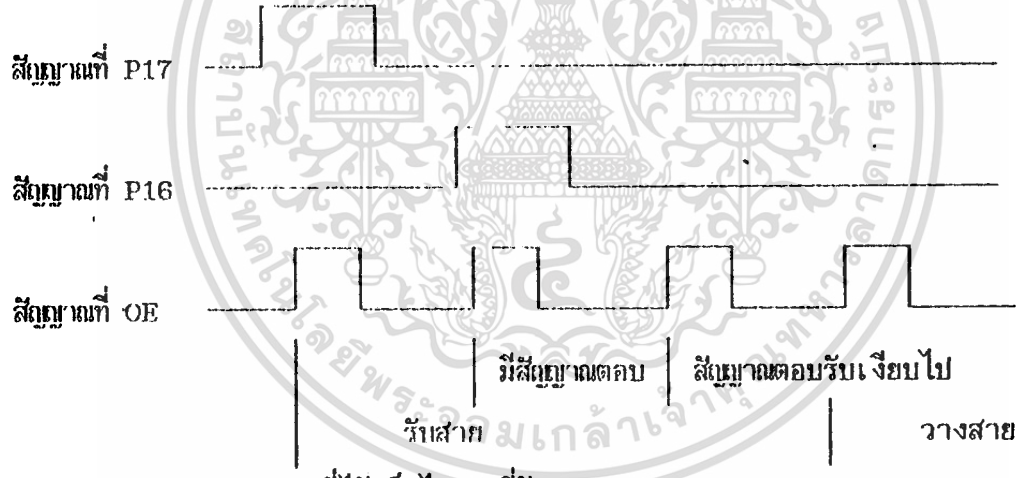
และเมื่อยกเลิกลอจิคที่ขา P17 และ OE ให้เป็นลอจิคศูนย์ทั้งคู่ ไอซีฟลิปฟลอปภายในวงจรพักสาย ก็ยังคงทำงานค้างอยู่ จนเมื่อให้ลอจิคที่ขา P17 เป็นศูนย์และลอจิคที่ขา OE เปลี่ยนจากศูนย์เป็นหนึ่ง วงจรพักสายก็หยุดทำงาน



ผลการทดลองที่ได้เป็นไปตามหลักการทำงานที่ออกแบบไว้

4.1.3 การทดลองวงจรส่งสัญญาณตอบรับ

หลังจากที่ทดลองได้ว่าวงจรพักสายสามารถทำงานได้แล้ว จึงทดลองส่งสัญญาณตอบรับ โดยการใช้ลอจิคที่ขา P16 ของ 8048 เป็นตัวควบคุมว่าจะให้สัญญาณตอบรับเงียบหรือดังขึ้น แล้วให้ผู้ทดลองที่ติดต่อเข้ามา ลองฟังเสียงสัญญาณตอบรับด้วย ในการทดลองได้ให้ลอจิคแก่ขา P16 ,P17 และ OE สัมพันธ์กันดังนี้ แล้วได้ผลออกมาดังรูป



ผลการทลองที่ได้เป็นไปตามที่ต้องการ

4.1.4 การทดลองวงจรรับหมายเลขโทรศัพท์

เป็นการทดลองส่งสัญญาณ DTMF ผ่ายทางสายโทรศัพท์ แล้วให้ไอซี 8870 ภายในวงจรนี้แปลงสัญญาณ DTMF ออกมาเป็นรหัสไบนารี่ แล้วทดลองตรวจจับสัญญาณ STD ด้วย

ในการทดลองได้ต่อไอซี 8870 ให้รับสัญญาณ DTMF ผ่านทางหม้อแปลงไฟฟ้า LI037 แล้ววัดลอจิคเอาท์พุทที่ ขา Q1-Q4 ของไอซี 8870 ส่วนการทดลองเกี่ยวกับสัญญาณ SID จากไอซี 8870 นั้นเนื่องจากสัญญาณ SID เป็นสัญญาณที่เกิดขึ้นชั่วขณะ และมี

คาบเวลาน้อยๆ ประกอบกับเขาดแคลนเครื่องมือที่เหมาะสมจึงไม่สามารถวัดคาบเวลาของสัญญาณ SID ออกมาได้ สามารถทดลองได้เพียงว่า มีสัญญาณ SID ออกมาจาก 8870 หรือไม่เท่านั้น โดยการต่อวงจรโมโนสเตเบิ้ลเข้าที่ขา SID ทำให้สามารถทราบได้ว่ามีสัญญาณเข้ามาทริกวงจรโมโนสเตเบิ้ลหรือไม่

เมื่อผู้ใช้โทรศัพท์กดคีย์โทรศัพท์ เข้ามาหนึ่งครั้ง สังเกตุเห็นการเปลี่ยนแปลงลอจิคที่ขา SID และที่เอาท์พุทของโมโนสเตเบิ้ล ส่วนที่ขาเอาท์พุท Q1-Q4 นั้นสามารถอ่านได้ค่าตัวเลขไบนารี่ได้ตรงกับคีย์ที่กด และลอจิคที่เอาท์พุทยังคงค้างอยู่จนกระทั่งผู้ทดลองกดคีย์อื่นๆต่อมา

ในระหว่างการทดลองได้มีการปรับค่าอัตราขยายของ 8870 ให้สามารถทำงานได้ เนื่องจากในตอนออกแบบนั้นไม่ทราบขนาดของสัญญาณ DTMF ที่ผ่านสายโทรศัพท์เข้ามา และไม่ทราบขอบเขตที่แน่นอนของขนาดสัญญาณอินพุทที่ไอซี 8870 จึงจำเป็นต้องอาสัยการทดลองปรับช่วย

4.1.5 การทดลองวงจรควบคุมการรับเข้าดูมูล และ โปรแกรมควบคุม

ในการทดลองเกี่ยวกับวงจรควบคุมนี้ อาศัยอุปกรณ์อีกสองชิ้นเข้ามาช่วยคือ

1 เครื่องไมโครคอมพิวเตอร์แอปเปิ้ลทู (APPLE IIE) และโปรแกรมแอสเซมเบลอร์ของ 8048 ที่ดโปรแกรม SC MACRO 8048 สำหรับแปลงภาษาแอสเซมเบลี้ของ 8048 ให้เป็นภาษาเครื่อง

2 วงจรแรมสองทาง ใช้สำหรับเก็บโปรแกรมภาษาเครื่องที่แปลงได้ และนำมาให้รันด้วย 8048 โดยในการใช้งานแรมสองทางนี้ เมื่อต้องการเขียนโปรแกรมให้กับ 8048 ก็จะเขียนโดยใช้เครื่องแอปเปิ้ลทู และเมื่อเขียนเสร็จแล้วต้องการจะทดสอบโปรแกรม ก็สามารถเลื่อนสวิทช์เพื่อให้หน่วยความจำแรมสองทาง หันไปติดต่อกับ 8048 ได้

การทดสอบโปรแกรมโดยใช้อุปกรณ์ทั้งสองชนิดนี้ทำให้มีความสดวกขึ้นมาก แต่ถ้ามีโปรแกรม หรืออุปกรณ์อื่นๆที่ช่วยในการติดตามโปรแกรม และตรวจหาข้อผิดพลาดในโปรแกรมด้วย ก็จะทำให้สามารถทำงานได้รวดเร็วขึ้น

ในการทดลองนั้นจะต้องตรวจสอบการทำงานของวงจรควบคุมนี้เสียก่อน โดยเขียนโปรแกรมสำหรับตรวจสอบวงจรที่ติดต่อกับ 8048 เช่นโปรแกรมให้รับรับสายโทรศัพท์ ส่งสัญญาณตอบรับ รับสัญญาณจากไอซี 8870 เป็นต้น เมื่อตรวจสอบอุปกรณ์ภายนอกครบหมดแล้ว ก็สามารถเขียนโปรแกรมควบคุม แล้วทดลองรันโปรแกรมควบคุมนั้นได้

จากผลการทดลองที่ได้นั้นพบว่า โปรแกรมสามารถทำงานได้ตามที่ออกแบบไว้ แต่ก็มีบางครั้งซึ่งไม่เป็นไปตามการทดลองบ้าง

4.1.6 การทดลองวงจรพอร์ทสำหรับเครื่องไมโครคอมพิวเตอร์ IBM PC/XT

การทดลองนี้เป็นการทดลองส่งข้อมูลจาก 8048 ไปยังเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ซึ่งสามารถรับส่งข้อมูลได้ถูกต้องทุกประการ

4.1.7 การทดลองโปรแกรมจัดการฐานข้อมูล และควบคุมการส่งข้อมูล

การทดลองนี้เป็นการทดลองการเก็บข้อมูลของการส่งลงในดิสเกทและการส่งข้อมูลไปยังเครื่องส่งการทดลองนี้จะต้องทำไปพร้อมกับการทดลองส่งข้อมูลแบบขนานในหัวข้อต่อไปแต่ใช้โปรแกรมที่มีการบันทึกข้อมูลลงในไฟล์ซึ่งสามารถบันทึกข้อมูลได้ถูกต้องทุกประการ

4.2 การทดลองเกี่ยวกับการส่งข้อมูล

4.2.1 การทดลองวงจรส่งข้อมูลแบบอนุกรม

เป็นการทดลองส่งข้อมูลแบบอนุกรมออกจากเครื่องไมโครคอมพิวเตอร์ IBM PC/XT ให้มีรูปฟอร์มตามที่ต้องการ คือมีบิทเริ่มเป็นลอจิคศูนย์ ตามด้วยข้อมูล 8 บิทและบิทหยุดอีกสองบิท

ในการส่งข้อมูลตามที่ออกแบบไว้นั้น จะส่งด้วยอัตราเร็ว 600 บิทต่อวินาที แต่เนื่องจากไม่มีเครื่องที่เหมาะสมสำหรับวัดข้อมูลอนุกรมที่ส่งออกได้ จึงต้องเปลี่ยนความเร็วในการส่งเป็น 0.5 บิทต่อวินาที เพื่อให้สามารถตรวจสอบข้อมูลที่ส่งออกมาได้

ในการทดลองจะตรวจสอบการทำงานของส่วนชิฟท์รีจีสเตอร์ และส่วนวงจรนับจำนวนบิทว่าทำงานเป็นปกติหรือไม่ โดยการใช้เครื่องไมโครคอมพิวเตอร์ IBM PC/XT ส่งข้อมูลไปยังชิฟท์รีจีสเตอร์ แล้วดูเอาท์พุทจากวงจร ซึ่งข้อมูลที่ได้จากการทดลองวัดลอจิคที่เอาท์พุทจะเริ่มที่ลอจิคศูนย์ของบิทเริ่ม แล้วตามด้วยข้อมูลบิทที่ 8 จนถึงบิทที่ 1 แล้วมีบิทหยุดที่เป็นลอจิคหนึ่งอีกสองบิท

สำหรับเอาท์พุทจากวงจรนับจำนวนบิทนั้น จะมีลอจิคศูนย์ จนกระทั่งส่งข้อมูลออกไปครบ 11 บิทแล้ว (บิทเริ่ม 1 บิท , ข้อมูล 8 บิท , บิทหยุด 2 บิท) ก็จะเปลี่ยนเป็นลอจิคหนึ่ง

4.2.2 การทดลองวงจรมอดูเลทสัญญาณข้อมูลกับคลื่นพาหะรอง

เป็นการทดลองวัดคุณสมบัติของวงจรมอดูเลท เกี่ยวกับลักษณะของสัญญาณ

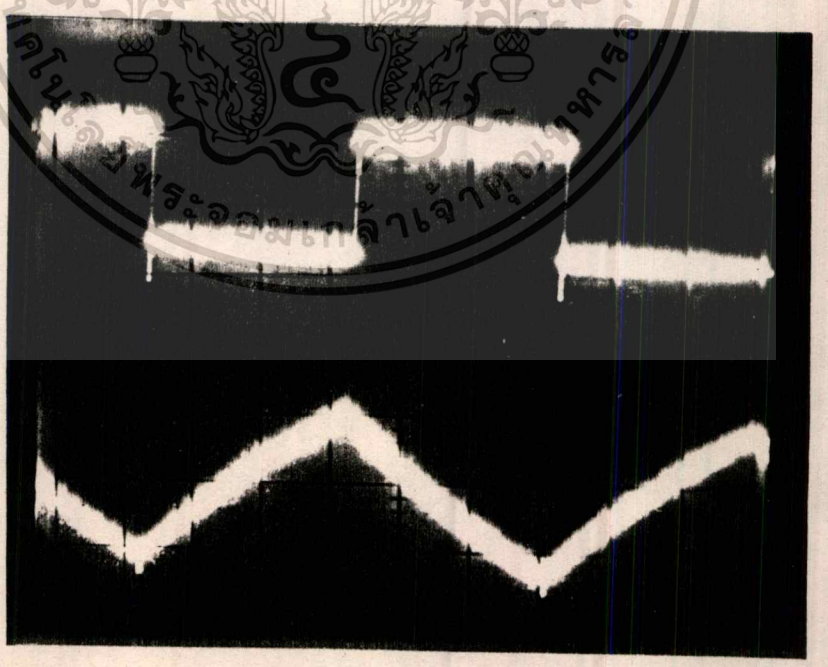
เอาท์พุทที่ได้ในสภาวะที่อินพุทมีค่าคงที่ กับในช่วงที่อินพุทมีการเปลี่ยนแปลง ,ความแน่นอนของค่าความถี่

4.2.2.1 โดยใช้วงจรออสซิเลเตอร์ควบคุมด้วยแรงดัน

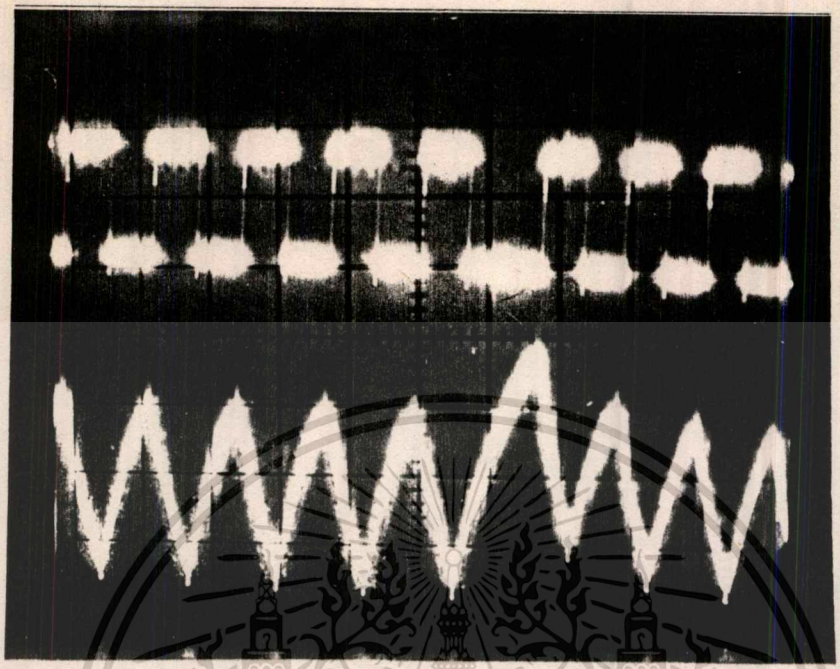
เป็นการทดลองวงจรออสซิเลเตอร์ควบคุมแรงดันที่ใช้ ไอซี 566 ซึ่งเมื่อป้อนโลจิก 0 เข้าไอซี 566 จะได้ความถี่ต่ำกว่า 8.4 กิโลเฮิทร์ซ (ขณะเริ่มเปิดเครื่อง) จนเวลาผ่านไป10นาทีจึงจะได้ความถี่ 8.4 กิโลเฮิทร์ซ และเมื่อป้อนโลจิก 1 ก็จะได้ผลเช่นเดียวกันคือต้องรอสักครู่จึงจะได้ความถี่ 7.6 กิโลเฮิทร์ซ ซึ่งแสดงว่าความถี่ในการออสซิเลทของไอซี 566 แปรตามอุณหภูมิทำให้การใช้งานไม่แน่นอนจึงเปลี่ยนไปทดลองใช้วงจรหารความถี่ในหัวข้อถัดไป

4.2.2.1 โดยใช้วงจรหารความถี่

การทดลองนี้เป็นการทดลองใช้วงจรมอดูเลทที่ใช้วงจรหารความถี่ โดยใช้ข้อมูลที่มีลอจิค 0,1 เป็นตัวควบคุมการหารความถี่จากสัญญาณนาฬิกาความถี่ 4.77 MHz จากไอบีเอ็ม ซึ่งเมื่อทดลองป้อนลอจิค0 จะได้ความถี่ที่ออกมาประมาณ 8.5 KHz และเมื่อป้อนลอจิค1 ก็จะได้ความถี่ประมาณ 7.6 KHz ซึ่งจากการทดลองพบว่า อุณหภูมิมีผลต่อความถี่ที่ออกมาน้อยมาก



รูปที่ 4.1 แสดงสัญญาณที่ได้จากวงจรมอดูเลททั้งรูปสี่เหลี่ยม และสามเหลี่ยม

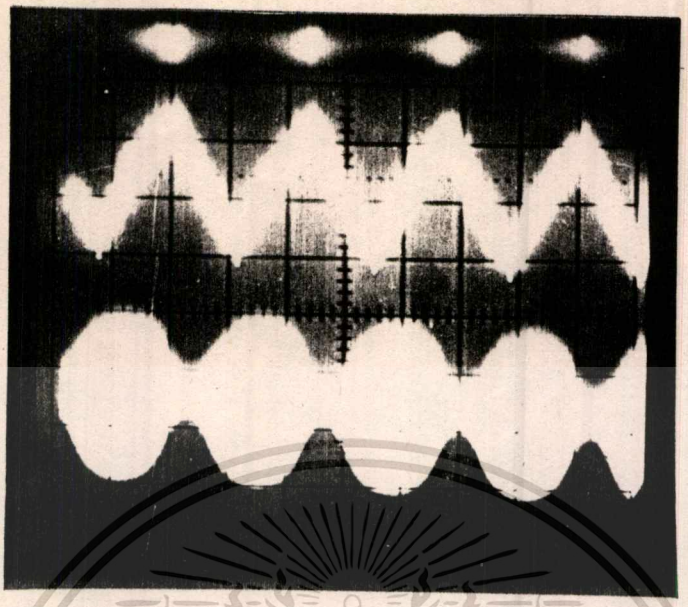


รูปที่ 4.2 แสดงสัญญาณจากวงจรมคดูเลทเมื่ออินพุทเป็นสัญญาณนาฬิกา 600 Hz

4.2.3 การทดลองวงจรส่งคลื่นวิทยุ AM

การทดลองนี้เป็นการทดลองส่งสัญญาณคลื่นผาหะรองผ่านวงจรเครื่องส่งวิทยุ เอ. เอ็ม. ที่สร้างขึ้นอย่างง่ายๆ โดยใช้วงจรทรานซิสเตอร์คอมมอนเบสออสซิลเลเตอร์(common base Oscillator) ที่มีความถี่การออสซิลเลทประมาณ 1.6 MHz ทำได้โดยการส่งความถี่ประมาณ 600 Hz ให้กับวงจรหารความถี่ในหัวข้อ4.2.2.1ป้อนสัญญาณที่ได้ให้กับเครื่องส่ง วัดสัญญาณที่ได้จากเครื่องรับวิทยุ เอ. เอ็ม. ที่ทำการดัดแปลงตามหัวข้อ 3.4.1 และปรับความถี่ไว้ประมาณ 1.6 MHz จากนั้นทำการปรับความถี่ของเครื่องส่งจนกระทั่งเครื่องรับสามารถรับสัญญาณได้

จากการทดลองพบว่าการที่ป้อนสัญญาณจากวงจรหารความถี่โดยตรง(อินพุทของเครื่องส่งเป็นสัญญาณสี่เหลี่ยม) เครื่องส่งจะให้เอาท์พุทที่ไม่ดีนักจึงได้เพิ่มวงจรอินทีเกรท(Integrater Circuit)ระหว่างวงจรหารความถี่กับเครื่องส่งเพื่อให้อินพุทของเครื่องส่งเป็นคลื่นสามเหลี่ยมในกรณีนี้เครื่องส่งจะให้เอาท์พุทที่ดีกว่าดังรูปที่ 4.3

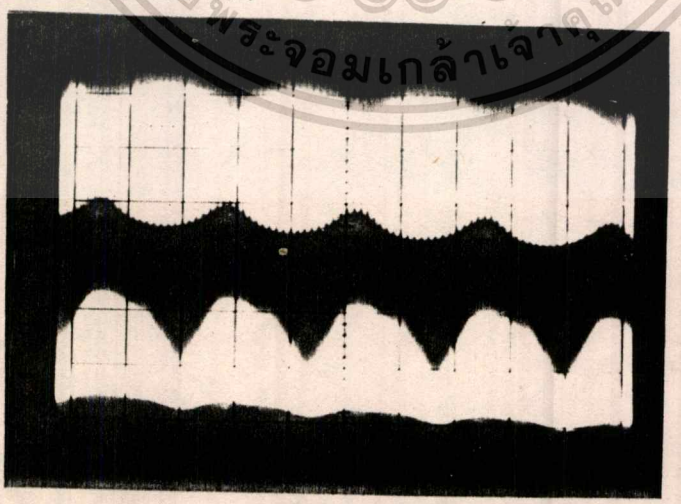


รูปที่ 4.3 อินพุทของเครื่องส่ง(บน) เอาท์พุทของเครื่องส่ง(ล่าง)

4.3 การทดลองเกี่ยวกับการรับโทรมูล

4.3.1 การทดลองวงจรรับสัญญาณวิทยุ AM

การทดลองนี้ทำไปพร้อมๆกับการทดลองที่4.2.3พบว่าการรับส่งสัญญาณทำได้ไม่ดีนักเนื่องจากการส่งสัญญาณวิทยุความถี่ 1.6 MHz ต้องการเครื่องส่งที่มีสายอากาศยาวมากๆและเสถียรภาพของความถี่เครื่องส่งที่สร้างขึ้นนี้ไม่ดีนักทำให้ความถี่ที่ส่งเปลี่ยนแปลงอยู่เรื่อยๆต้องมีการปรับแต่งตลอดเวลา เมื่อวัดสัญญานเอาท์พุทของเครื่องส่งเทียบกับเอาท์พุทของเครื่องรับจะได้ดังภาพที่ 4.4



รูปที่ 9 สัญญาณของเครื่องส่ง(บน) สัญญาณของเครื่องรับ(ล่าง)

4.3.2 การทดลองวงจรกรองความถี่

การทดลองนี้เป็นการทดลองเพื่อหาคุณสมบัติของวงจรกรองช่วงความถี่ที่ออกแบบไว้ในหัวข้อ 3.4.1 ว่าได้คุณสมบัติตามที่ต้องการหรือไม่ ทำได้โดยการป้อนสัญญาณชายน์ที่ดินพุทของวงจรทำการเปลี่ยนความถี่ของสัญญาณอินพุทไปทีละค่าของสัญญาณเอาท์พุทกับความถี่ของสัญญาณอินพุทนำค่าที่ได้นี้มาหาค่าคุณสมบัติของวงจรกรองความถี่ได้ผลการทดลองดังนี้

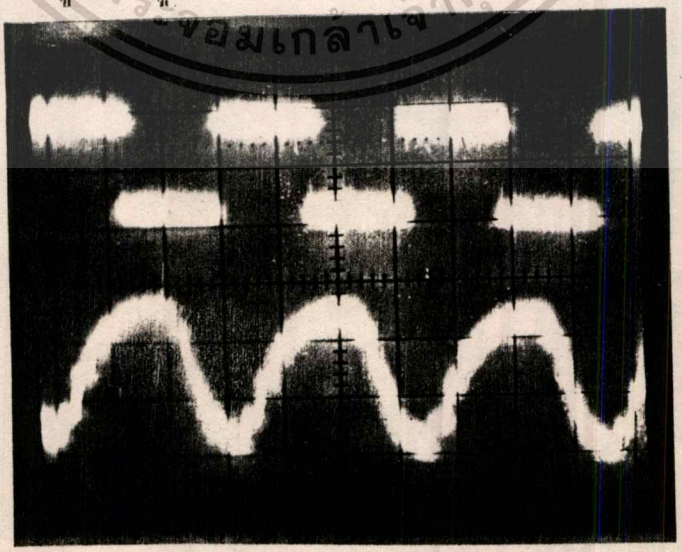
ความถี่กลางของช่วงผ่าน	= 8.19 KHz
อัตราขยายที่ความถี่กลาง	= 11
ความถี่คัทออฟด้านต่ำ(Low cut. off frequency)	= 7.75 KHz
ความถี่คัทออฟด้านสูง(High cut. off frequency)	= 8.92 KHz
โพลคิว(Pole Q)	= 7

จากการทดลองพบว่าวงจรที่สร้างขึ้นนี้มีคุณสมบัติตามที่ต้องการ จากนั้นได้นำวงจรนี้ไปกรองสัญญาณที่ได้จากเครื่องรับวิทยุ(สัญญาณในรูปที่ 4.9) ได้สัญญาณเอาท์พุทที่มีลักษณะเป็นคลื่นชายน์ แสดงว่าวงจรนี้สามารถทำงานได้ถูกต้องตามต้องการ

4.3.3 การทดลองวงจรดีมอดูเลทสัญญาณข้อมูลออกจากคลื่นพาหะรอง

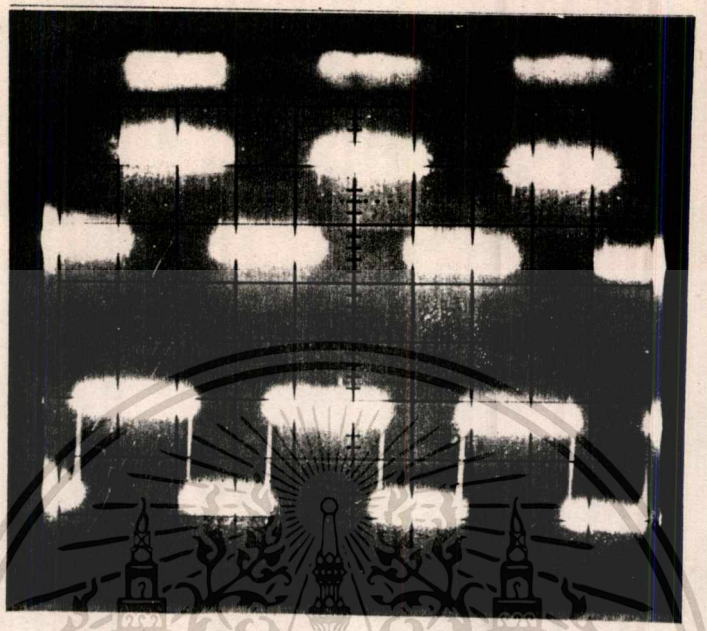
4.3.3.1 โดยใช้วงจรเฟสล็อกลูป

การทดลองทำโดยการปรับความถี่ประมาณ 600 Hz ให้กับวงจรมอดูเลทแล้วนำสัญญาณจากวงจรมอดูเลท ป้อนให้กับวงจรดีมอดูเลทแบบเฟสล็อคลูป จากการทดลองได้เอาท์พุทของวงจรดีมอดูเลทดังรูปที่ 4.5



รูปที่ 4.5 สัญญาณอินพุทของวงจรมอดูเลท(บน) สัญญาณเอาท์พุทของวงจรดีมอดูเลท

และเมื่อนำสัญญาณที่ได้ผ่านวงจรเปรียบเทียบแรงดันจะได้ผลดังรูปที่ 4.6

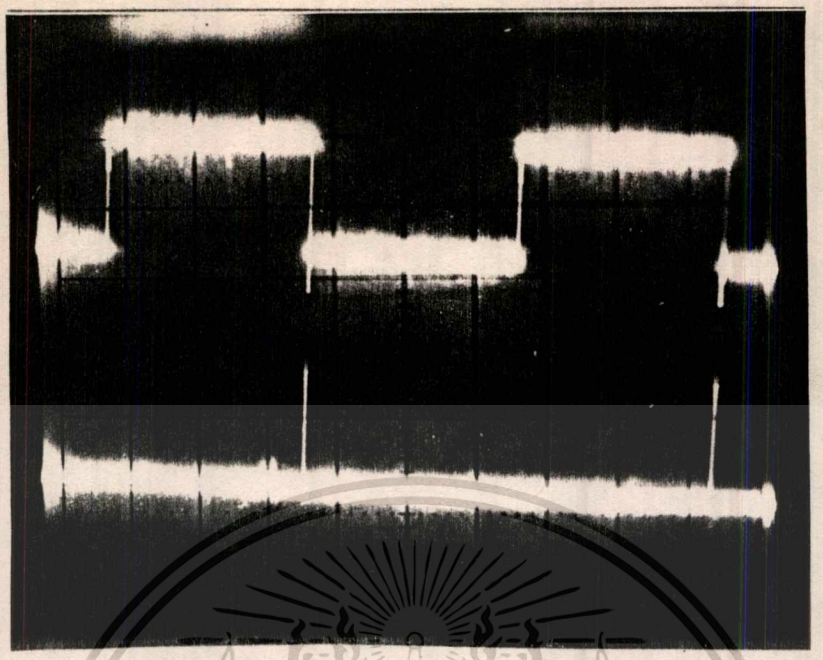


รูปที่ 4.6 อินพุท (บน) เอาท์พุท(ล่าง)

จากภาพนี้ในระหว่างการทดลองพบว่าการทำงานของไอซีเฟสล็อคลูปเบอร์ MC14046มีการเปลี่ยนแปลงตามคุณสมบัติ ผลนี้จะทำให้ดิวตี้ไซเคิล(Duty cycle)ของสัญญาณเอาท์พุทผิดไปได้ ซึ่งจะทำให้การรับส่งข้อมูลผิดพลาด ดังนั้นจึงได้ออกแบบวงจรดีมอดูเลทที่มีการทำงานที่แน่นอนกว่า โดยใช้หลักการของวงจรนับ แต่อย่างไรก็ตามวงจรที่ใช้เฟสล็อคลูปมีข้อดีคือต้องการพลังงานต่ำมาก(35 mW)

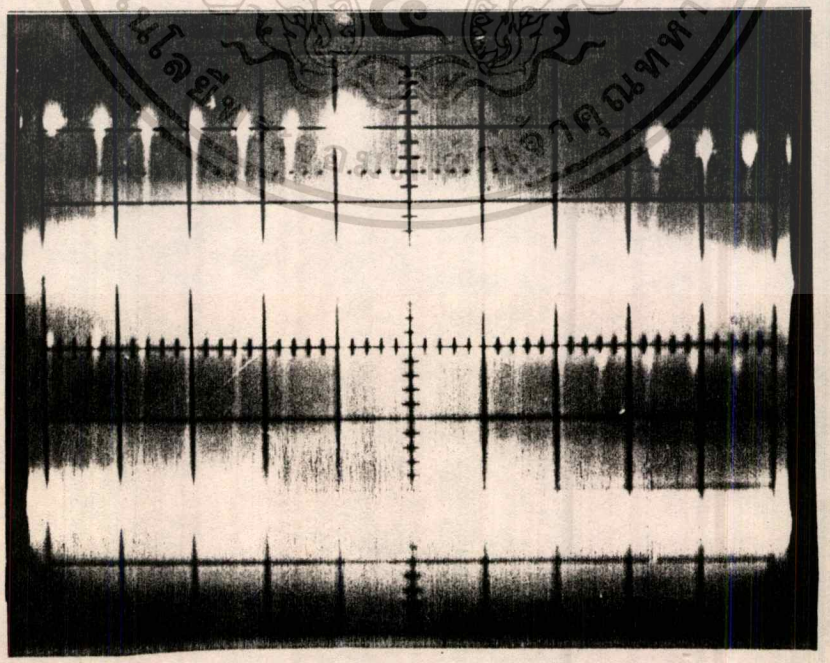
4.3.3.2 โดยใช้วงจรนับความถี่

ในการทดลองได้นำสัญญาณนาฬิกาขนาด 600 Hz ป้อนเข้าสู่วงจรมอดูเลท แล้วนำสัญญาณจากวงจรมอดูเลท เข้าสู่วงจรดีมอดูเลทนี้ เมื่อใส่สัญญาณอินพุทเข้าไปวงจรดีมอดูเลทจะทำให้เกิดสัญญาณพัลซ์แคบขึ้นในช่วงขอบขาลงของสัญญาณอินพุทดังแสดงไว้ในรูปที่ 4.7

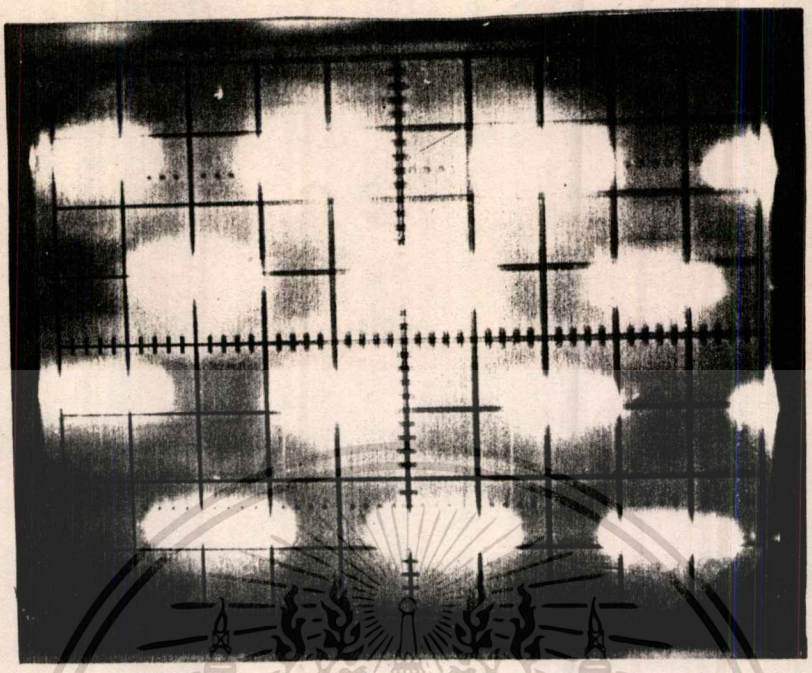


รูปที่ 4.7 แสดงสัญญาณแลทช์กับสัญญาณอินพุท
ความกว้างของพัลซ์นั้นมีค่าประมาณ 1 uS

เมื่อทดลองวัดสัญญาณจากวงจรนับ เปรียบเทียบกับสัญญาณแลทช์นี้จะเห็นว่า เมื่อสัญญาณอินพุทมีความถี่ต่ำ วงจรนับสามารถนับได้ถึง 255 จึงมีเอาท์พุทเป็นลอจิคหนึ่งออกมา ส่วนในช่วงท้ายซึ่งสัญญาณมีความถี่สูง ทำให้วงจรนับยังไม่ทันนับถึง 255 ก็จะถูกสัญญาณแลทช์ทริกให้เริ่มนับใหม่ สัญญาณเอาท์พุทจากวงจรนับจึงมีลอจิคศูนย์



รูปที่ 4.8 แสดงสัญญาณแลทช์กับสัญญาณจากวงจรนับ



รูปที่ 4.9 แสดงสัญญาณก่อนมอดูเลท และสัญญาณจากการดีมอดูเลท

จากรูปข้างต้นจะเห็นว่าในช่วงที่สัญญาณเปลี่ยนจากลอจิคหนึ่งเป็นลอจิคศูนย์นั้น สัญญาณจากวงจรดีมอดูเลทจะมาช้ากว่าสัญญาณจริงประมาณ 250 uS ในขณะที่ช่วงเปลี่ยนลอจิคศูนย์เป็นลอจิคหนึ่งนั้น สัญญาณจากวงจรดีมอดูเลทจะมาช้ากว่าสัญญาณจริงประมาณ 135 uS

4.3.4 การทดลองวงจรประมวลผลและแสดงผลข้อมูล และโปรแกรม

การทดลองเกี่ยวกับวงจรนี้ จำเป็นต้องใช้เครื่องไมโครคอมพิวเตอร์แอปเปิ้ลทู โปรแกรมแอสเซมเบลอร์ของ 8048 และวงจรแรมสองทางช่วยในการเขียน และทดสอบโปรแกรม

ในขั้นแรกได้ทดสอบวงจรภายนอกที่ต่ออยู่กับ 8048 ทั้งหมด ได้แก่ ส่วนแสดงผล สวิทช์กด เป็นต้น โดยเขียนโปรแกรมง่ายๆสำหรับทดสอบ จากนั้นจึงเขียนโปรแกรมควบคุมจริงด้วยเครื่องแอปเปิ้ลทู แล้วทดลองรันโปรแกรม

ข้อมูลอนุกรมที่ใช้ทดสอบนั้น ในขั้นแรกได้ต่อเอาท์พุทจากวงจรส่งข้อมูลอนุกรมเข้ากับขา INT ของ 8048 โดยแล้วทดลองใช้โปรแกรมภาษาเบสิกง่ายๆ จำลองการส่งข้อมูลออกมา โปรแกรมที่ใช้ทดสอบเป็นดังนี้

```
10 FOR I =1 TO 10
20 READ A
30 FOR J=1 TO 3
40 OUT &H300,A
43 IF INP(&H303)<128 THEN 43
45 PRINT A;
50 NEXT J
55 PRINT
60 NEXT I
70 DATA &hff,&hff,&h10,&h59,&h27,&h02,&h28,&h20,&h70,&h46
```

เมื่อส่งข้อมูลออกมาแล้ว วงจรส่วนประมวลผลสามารถรับข้อมูลได้ แล้วส่งสัญญาณเตือนออกมา เมื่อกดสวิทช์เสียงเตือนก็จะเงียบไปแล้วปรากฏหมายเลขโทรศัพท์ขึ้นมา หลังจากปล่อยมือจากสวิทช์ ตัวเลขยังคงแสดงผลอยู่

เมื่อมีการส่งข้อมูลอีกชุดหนึ่งเข้ามา เครื่องก็ส่งสัญญาณเตือนอีกครั้ง ข้อมูลเก่าถูกย้ายไปอยู่ในหน่วยความจำอีกชุดหนึ่ง และข้อมูลชุดใหม่เข้าแทนที่ข้อมูลเดิม แต่ก็ยังสามารถดูข้อมูลทั้งสองได้ โดยเลือกกดสวิทช์ตัวใดตัวหนึ่ง

บทที่ 5 บทวิจารณ์และสรุป

จากผลการทดลองสร้างระบบวิทยุติดตามตัวนี้จะพบว่า ระบบที่สร้างขึ้นนี้สามารถจัดการการติดต่อทางโทรศัพท์กับผู้ที่ต้องการติดต่อเพจเจอร์ได้จริง และเครื่องไมโครคอมพิวเตอร์ไอบีเอ็มสามารถบันทึกข้อมูลการติดต่อในแต่ละครั้งได้ทั้งยังสามารถส่งข้อมูลเพื่อไปมอดูเลทกับคลื่นพาหะรองได้ถูกต้อง

สำหรับการมอดูเลทคลื่นพาหะรองนั้นได้ทำการทดลองสองระบบคือระบบที่ใช้วงจรออสซิลเลเตอร์ควบคุมด้วยแรงดันและระบบที่ใช้วงจรหารความถี่ซึ่งพบว่าระบบที่ใช้วงจรหารความถี่จะมีเสถียรภาพของความถี่ที่ดีกว่า เมื่อทดลองส่งกับเครื่องส่งพบว่าเครื่องส่งที่สร้างขึ้นนี้เป็นแบบง่ายๆ มีความถี่ในการออสซิเลทไม่คงที่แปรตามอุณหภูมิ ซึ่งต้องทำการปรับความถี่อยู่เรื่อยๆ ซึ่งถ้านำระบบนี้ไปใช้กับสถานีวิทยุ AM ก็จะไม่มีปัญหาในการส่ง

ส่วนภาคดีมอดูเลทสัญญาณที่ส่งมาก็ได้ทำการทดลอง 2 ระบบ คือ ใช้วงจรเฟสล็อคลูป กับใช้วงจรนับความถี่ พบว่า วงจรนับความถี่ไม่ต้องมีการปรับแต่งมากเหมือนวงจรเฟสล็อคลูป และวงจรเฟสล็อคลูปจะมีผลแปรตามอุณหภูมิอยู่บ้าง แต่กินไฟน้อยกว่าวงจรนับความถี่

จากการทดลอง เมื่อต่อทั้งระบบแล้ว ตัวเพจเจอร์สามารถแสดงผลได้ตามหมายเลขที่ส่งมา ไม่ว่าจะใช้วงจรออสซิเลเตอร์ที่ควบคุมด้วยแรงดัน หรือวงจรนับความถี่ในการมอดูเลท และไม่ว่าจะใช้วงจรเฟสล็อกลูป หรือวงจรหารความถี่ในการดีมอดูเลทสัญญาณที่ส่งมา แต่พบว่า เสถียรภาพของระบบที่ใช้วงจรหารความถี่ในการมอดูเลท และวงจรนับความถี่ในการดีมอดูเลท จะมีเสถียรภาพที่ดีกว่า แต่อาจจะกินไฟมากกว่า

จากการทดลองทั้งหมด จะเห็นว่าระบบวิทยุติดตามตัวนี้สามารถนำไปใช้งานได้จริงโดยนำไปติดตั้งกับสถานีวิทยุ AM ทั่วไปได้ และสามารถปรับปรุงให้ตัวเพจเจอร์นั้นมีขนาดเล็กลง และกินไฟน้อยลงได้ โดยใช้เครื่องรับวิทยุ AM ขนาดเล็กจิ๋ว และทำวงจรตัวรับให้อยู่ในชิพไอซีเดียว และจอแสดงผลแบบ LCD แทนแบบ LED

ภาคผนวก

1. รายละเอียดของไอซี TL072, TL074

2. รายละเอียดของไอซี 4046

3. รายละเอียดของไอซี 8870

4. สรุปชุดคำสั่งของ 8048

5. โปรแกรมภาษาแคสเซมบลี้ สำหรับ ส่วนควบคุมการรับข้อมูลทางโทรศัพท์

6. โปรแกรมภาษาแอสเซมบลี้ สำหรับ ส่วนประมวลผลและแสดงผลข้อมูล

7. โปรแกรมภาษาซี สำหรับ ควบคุมและส่งผ่านข้อมูล

1. รายละเอียดของไอซี TL072, TL074

LINEAR INTEGRATED CIRCUITS

TYPES TL070, TL070A, TL071, TL071A, TL071B, TL072, TL072A, TL072B, TL074, TL074A, TL074B, TL075 LOW-NOISE JFET-INPUT OPERATIONAL AMPLIFIERS

D2393, SEPTEMBER 1978—REVISED SEPTEMBER 1983

19 DEVICES COVER COMMERCIAL, INDUSTRIAL, AND MILITARY TEMPERATURE RANGES

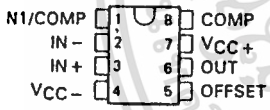
- Low Power Consumption
- Wide Common-Mode and Differential Voltage Ranges
- Low Input Bias and Offset Currents
- Output Short-Circuit Protection
- Low Total Harmonic Distortion 0.003% Typ
- Low Noise . . . Vn = 18 nV/√Hz Typ
- High Input Impedance . . . JFET-Input Stage
- Internal Frequency Compensation (Except TL070, TL070A)
- Latch-Up-Free Operation
- High Slew Rate . . . 13 V/μs Typ

description

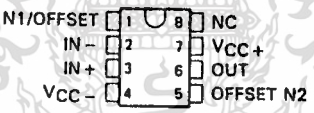
The JFET-input operational amplifiers on the TL07_ series are designed as low-noise versions of the TL08_ series amplifiers with low input bias and offset currents and fast slew rate. The low harmonic distortion and low noise make the TL07_ series ideally suited as amplifiers for high-fidelity and audio preamplifier applications. Each amplifier features JFET-inputs (for high input impedance) coupled with bipolar output stages all integrated on a single monolithic chip.

Device types with an "M" suffix are characterized for operation over the full military temperature range of −55°C to 125°C, those with an "I" suffix are characterized for operation from −25°C to 85°C, and those with a "C" suffix are characterized for operation from 0°C to 70°C.

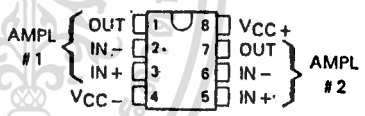
TL070, TL070A
D, JG, OR P DUAL-IN-LINE PACKAGE
(TOP VIEW)



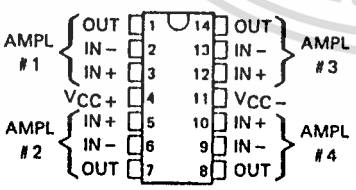
TL071, TL071A, TL071B
D, JG, OR P DUAL-IN-LINE PACKAGE
(TOP VIEW)



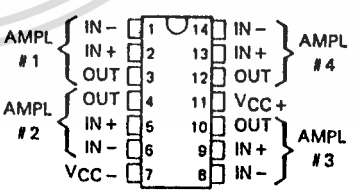
TL072, TL072A, TL072B
D, JG, OR P DUAL-IN-LINE PACKAGE
(TOP VIEW)



TL074, TL074A, TL074B
D, J, OR N DUAL-IN-LINE
OR W FLAT PACKAGE
(TOP VIEW)

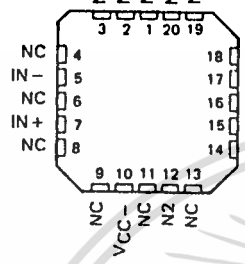


TL075
N DUAL-IN-LINE PACKAGE
(TOP VIEW)

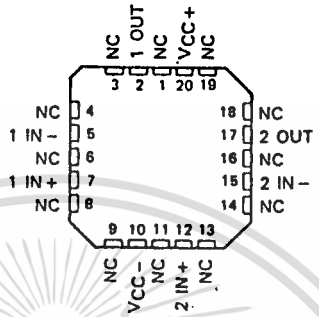


NC—No internal connection

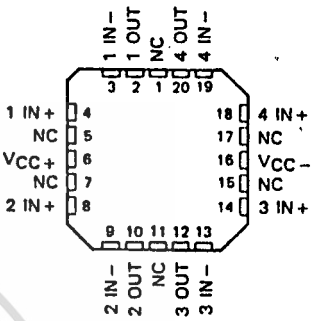
TL071
FH OR FK CHIP-CARRIER PACKAGE
(TOP VIEW)



TL072
FH OR FK CHIP-CARRIER PACKAGE
(TOP VIEW)

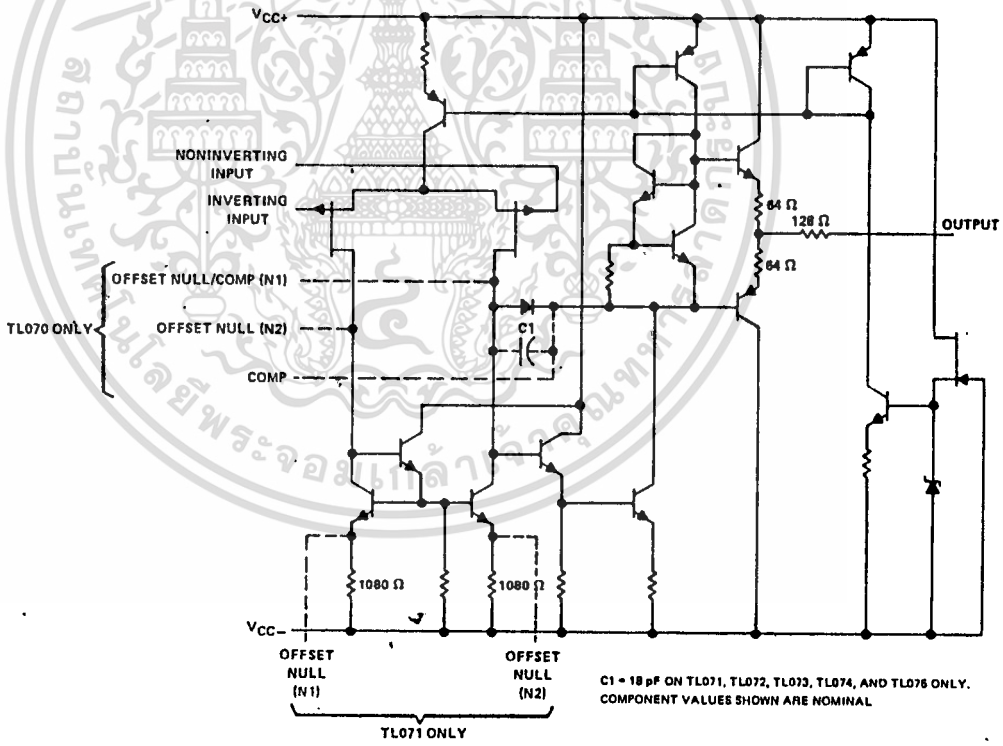


TL074
FH OR FK CHIP-CARRIER PACKAGE
(TOP VIEW)



NC—No internal connection

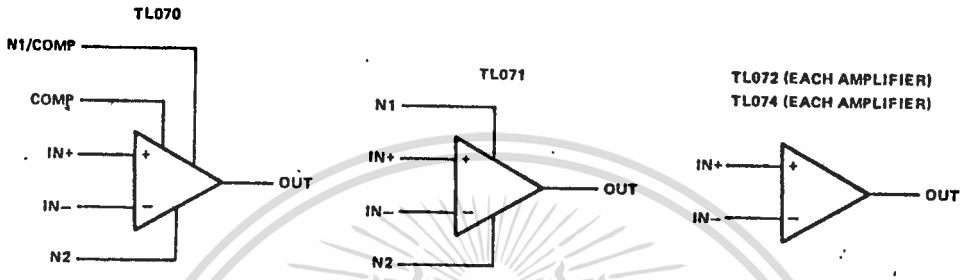
schematic (each amplifier)



C1 = 18 pF ON TL071, TL072, TL073, TL074, AND TL075 ONLY.
COMPONENT VALUES SHOWN ARE NOMINAL

TEXAS INSTRUMENTS
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

symbols



DEVICE TYPES, SUFFIX VERSIONS, AND PACKAGES					
	TL070	TL071	TL072	TL074	TL075
TL07_M	*	FH, FK, JG	FH, FK, JG	FH, FK, J, W	*
TL07_I	D, JG, P	D, JG, P	D, JG, P	D, J, N	*
TL07_C	D, JG, P	D, JG, P	D, JG, P	D, J, N	N
TL07_AC	D, JG, P	D, JG, P	D, JG, P	D, J, N	*
TL07_BC	*	D, JG, P	D, JG, P	D, J, N	*

*These combinations are not defined by this data sheet.

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

		TL07_M	TL07_I	TL07_C TL07_AC TL07_BC	UNIT
Supply voltage, V_{CC+} (see Note 1)		18	18	18	V
Supply voltage, V_{CC-} (see Note 1)		−18	−18	−18	V
Differential input voltage (see Note 2)		±30	±30	±30	V
Input voltage (see Notes 1 and 3)		±15	±15	±15	V
Duration of output short circuit (see Note 4)		unlimited	unlimited	unlimited	
Continuous total dissipation at (or below) 25°C free-air temperature (see Note 5)		680	680	680	mW
Operating free-air temperature range		−55 to 125	−25 to 85	0 to 70	°C
Storage temperature range		−65 to 150	−65 to 150	−65 to 150	°C
Lead temperature 1,6 mm (1/16 inch) from case for 60 seconds	J, JG, JH, FK, or W package	300	300	300	°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	D, N, or P package		260	260	°C

NOTES: 1. All voltage values, except differential voltages, are with respect to the midpoint between V_{CC+} and V_{CC-}.
2. Differential voltages are at the noninverting input terminal with respect to the inverting input terminal.
3. The magnitude of the input voltage must never exceed the magnitude of the supply voltage or 15 volts, whichever is less.
4. The output may be shorted to ground or to either supply. Temperature and/or supply voltages must be limited to ensure that the dissipation rating is not exceeded.
5. For operation above 25°C free-air temperature, refer to Dissipation Derating Curves, Section 2. In the J and JG packages, TL07_M chips are alloy-mounted; TL07_I, TL07_C, TL07_AC, and TL07_BC chips are glass mounted.

TYPES TL071M, TL072M, TL074M
LOW-NOISE JFET-INPUT OPERATIONAL AMPLIFIERS

electrical characteristics, $V_{CC\pm}$ = ±15 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		TL071M, TL072M			TL074M			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
V_{IO}	Input offset voltage	V_O = 0, R_S = 50 Ω	T_A = 25°C		3	6		3	9	mV
			T_A = −55°C to 125°C			9			15	
α_{VIO}	Temperature coefficient of input offset voltage	V_O = 0, T_A = −55° to 125°C	R_S = 50 Ω,		10			10		μV/°C
I_{IO}	Input offset current‡	V_O = 0	T_A = 25°C		5	100		5	100	pA
			T_A = −55°C to 125°C			20			20	nA
I_{IB}	Input bias current‡	V_O = 0	T_A = 25°C		30	200		30	200	pA
			T_A = −55°C to 125°C			50			20	nA
V_{ICR}	Common-mode input voltage range	T_A = 25°C		±11	±12		±11	±12		V
V_{OM}	Maximum peak output voltage swing	T_A = 25°C,	R_L = 10 kΩ	±12	±13.5		±12	±13.5		V
		T_A = −55°C to 125°C	R_L ≥ 10 kΩ	±12			±12			
			R_L ≥ 2 kΩ	±10	±12		±10	±12		
A_{VD}	Large-signal differential voltage amplification	V_O = ±10 V, T_A = 25°C	R_L ≥ 2 kΩ,	35	200		35	200		V/mV
		V_O = ±10 V, T_A = −55°C to 125°C	R_L ≥ 2 kΩ,	15			15			
B_1	Unity-gain bandwidth	T_A = 25°C			3			3		MHz
r_i	Input resistance	T_A = 25°C			10^{12}			10^{12}		Ω
CMRR	Common-mode rejection ratio	V_{IC} = V_{ICR} min, R_S = 50 Ω,	V_O = 0, T_A = 25°C	80	86		80	86		dB
k_{SVR}	Supply voltage rejection ratio ($\Delta V_{CC\pm}/\Delta V_{IO}$)	V_{CC} = ±15 V to ±9 V, R_S = 50 Ω,	V_O = 0, T_A = 25°C	80	86		80	86		dB
I_{CC}	Supply current (per amplifier)	No load, T_A = 25°C	V_O = 0,		1.4	2.5		1.4	2.5	mA
V_{o1}/V_{o2}	Crosstalk attenuation	A_{VD} = 100,	T_A = 25°C		120			120		dB

†All characteristics are measured under open-loop conditions with zero common-mode input voltage unless otherwise specified.

‡Input bias currents of a FET-input operational amplifier are normal junction reverse currents, which are temperature sensitive as shown in Figure 18. Pulse techniques must be used that will maintain the junction temperatures as close to the ambient temperature as is possible.

TEXAS INSTRUMENTS
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

electrical characteristics, $V_{CC\pm}$ = ±15 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		TL070I TL071I TL072I TL074I			TL070C TL071C TL072C TL074C TL075C			TL070AC TL071AC TL072AC TL074AC			TL071BC TL072BC TL074BC			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
V_{IO}	Input offset voltage	V_O = 0, R_S = 50 Ω	T_A = 25°C		3	6		3	10		3	6		2	3	mV
			T_A = full range			8			13			7.5			5	
α_{VIO}	Temperature coefficient of input offset voltage	V_O = 0, R_S = 50 Ω, T_A = full range			10			10			10			10		μV/°C
I_{IO}	Input offset current‡	V_O = 0	T_A = 25°C		5	100		5	100		5	100		5	100	pA
			T_A = full range			10			2			2			2	nA
I_{IB}	Input bias current‡	V_O = 0	T_A = 25°C		30	200		30	200		30	200		30	200	pA
			T_A = full range			20			7			7			7	nA
V_{ICR}	Common-mode input voltage range	T_A = 25°C		±11	±12		±11	±12		±11	±12		±11	±12		V
V_{OM}	Maximum peak output voltage swing	T_A = 25°C,	R_L = 10 kΩ	±12	±13.5		±12	±13.5		±12	±13.5		±12	±13.5		V
		T_A = full range	R_L ≥ 10 kΩ	±12			±12			±12			±12			
			R_L ≥ 2 kΩ	±10	±12		±10	±12		±10	±12		±10	±12		
A_{VD}	Large-signal differential voltage amplification	V_O = ±10 V, R_L ≥ 2 kΩ, T_A = 25°C		50	200		25	200		50	200		50	200		V/mV
		V_O = ±10 V, R_L ≥ 2 kΩ, T_A = full range		25			15			25			25			
B_1	Unity-gain bandwidth	T_A = 25°C			3			3			3			3		MHz
r_i	Input resistance	T_A = 25°C			10^{12}			10^{12}			10^{12}			10^{12}		Ω
CMRR	Common-mode rejection ratio	V_{IC} = V_{ICR} min, V_O = 0, R_S = 50 Ω, T_A = 25°C		80	86		70	86		80	86		80	86		dB
k_{SVR}	Supply voltage rejection ratio ($\Delta V_{CC\pm}/\Delta V_{IO}$)	V_{CC} = ±15 V to ±9 V, V_O = 0, R_S = 50 Ω, T_A = 25°C		80	86		70	86		80	86		80	86		dB
I_{CC}	Supply current (per amplifier)	No load, V_O = 0, T_A = 25°C			1.4	2.5		1.4	2.5		1.4	2.5		1.4	2.5	mA
V_{o1}/V_{o2}	Crosstalk attenuation	A_{VD} = 100, T_A = 25°C			120			120			120			120		dB

†All characteristics are measured under open-loop conditions with zero common-mode input voltage unless otherwise specified. Full range for T_A is 25°C to 85°C for TL07_I and 0°C to 70°C for TL07_C, TL07_AC, and TL07_BC.

‡Input bias currents of a FET-input operational amplifier are normal junction reverse currents, which are temperature sensitive as shown in Figure 18. Pulse techniques must be used that will maintain the junction temperatures as close to the ambient temperature as is possible.

operating characteristics, $V_{CC\pm}$ = ±15 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		TL07_M			ALL OTHERS			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
SR	Slew rate at unity gain	V_I = 10 V, C_L = 100 pF,	R_L = 2 kΩ, See Figure 1	10	13		8	13		V/μs
t_r	Rise time	V_I = 20 mV,	R_L = 2 kΩ,		0.1			0.1		μs
	Overshoot factor	C_L = 100 pF,	See Figure 1		10			10		%
V_n	Equivalent input noise voltage	R_S = 100 Ω	f = 1 kHz		18			18		nV/√Hz
			f = 10 Hz to 10 kHz		4			4		μV
I_n	Equivalent input noise current	R_S = 100 Ω,	f = 1 kHz		0.01			0.01		pA/√Hz
THD	Total harmonic distortion	$V_{O(rms)}$ = 10 V, R_L ≥ 2 kΩ,	R_S ≤ 1 kΩ, f = 1 kHz		0.003			0.003		%

PARAMETER MEASUREMENT INFORMATION

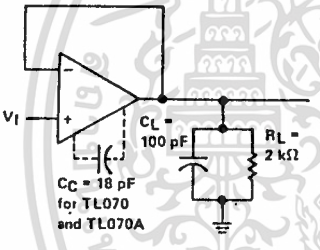


FIGURE 1–UNITY-GAIN AMPLIFIER

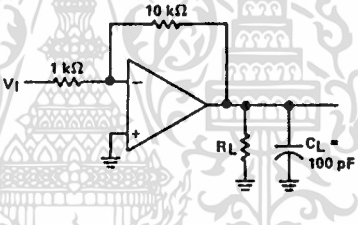


FIGURE 2–GAIN-OF-10 INVERTING AMPLIFIER

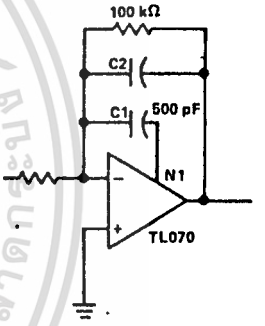


FIGURE 3–FEED-FORWARD COMPENSATION

INPUT OFFSET VOLTAGE NULL CIRCUITS

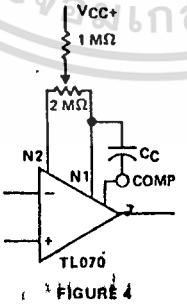


FIGURE 4

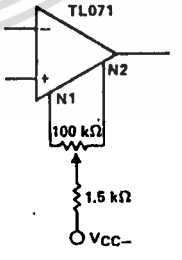


FIGURE 5

TEXAS INSTRUMENTS
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

TYPICAL CHARACTERISTICS†

MAXIMUM PEAK OUTPUT VOLTAGE vs FREQUENCY

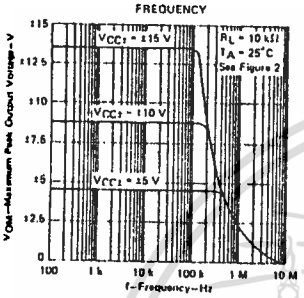


FIGURE 6

MAXIMUM PEAK OUTPUT VOLTAGE vs FREQUENCY

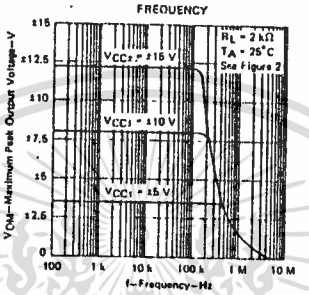


FIGURE 7

MAXIMUM PEAK OUTPUT VOLTAGE vs FREQUENCY

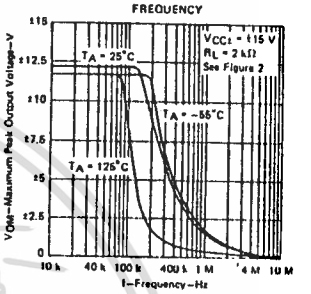


FIGURE 8

MAXIMUM PEAK OUTPUT VOLTAGE vs FREE-AIR TEMPERATURE

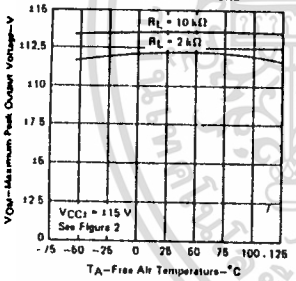


FIGURE 9

MAXIMUM PEAK OUTPUT VOLTAGE vs LOAD RESISTANCE

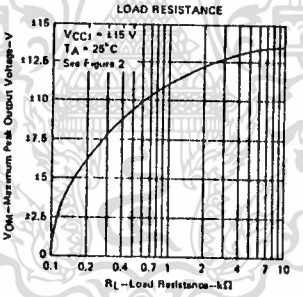


FIGURE 10

MAXIMUM PEAK OUTPUT VOLTAGE vs SUPPLY VOLTAGE

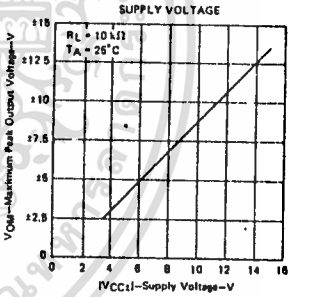


FIGURE 11

TOTAL POWER DISSIPATED vs FREE-AIR TEMPERATURE

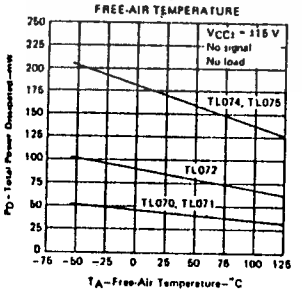


FIGURE 12

SUPPLY CURRENT PER AMPLIFIER vs FREE AIR TEMPERATURE

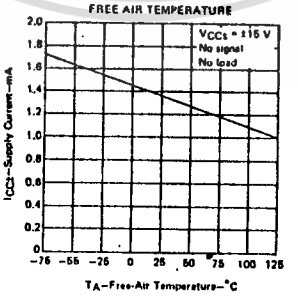


FIGURE 13

SUPPLY CURRENT vs SUPPLY VOLTAGE

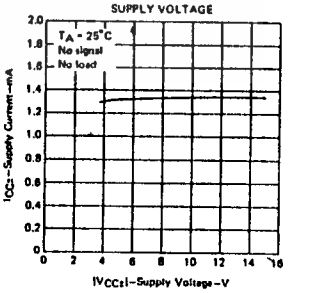


FIGURE 14

† Data at high and low temperatures are applicable only within the rated operating free-air temperature ranges of the various devices. A 18-pF compensation capacitor is used with TL070 and TL070A.

SCL4046B
SCL4446B

CMOS PHASE-LOCKED LOOPS

FEATURES

- Very low power consumption — 70 μW (typ) @ f_o = 10kHz, 5Vdc
- Operating frequency range (no offset) —
  Up to 3MHz (typ) @ 10Vdc (SCL4046B)
  Up to 4MHz (typ) @ 10Vdc (SCL4446B)
- Low frequency drift — 0.04%/°C (typ) @ 10Vdc
- Choice of two phase comparators:
  1. Exclusive-OR network
  2. Edge-controlled memory network with phase-pulse output for lock indication
- VCO Inhibit control for ON-OFF keying and ultra-low standby power consumption
- High VCO linearity 1% (typ)
- Source-follower output of VCO control input (Demodulator Output)
- Zener Diode to assist Supply Regulation
- Balanced Output Drive Current Specifications

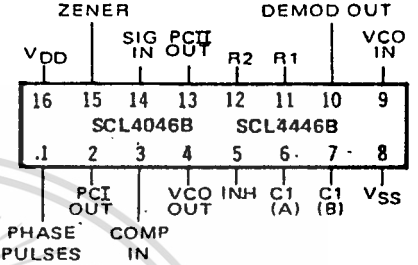
APPLICATIONS

- FM demodulator and modulator
- Frequency synthesis and multiplication
- Frequency discriminator
- Data synchronization
- Voltage-to-frequency conversion
- Tone decoding
- FSK Modems
- Signal conditioning

DESCRIPTION

The SCL4046B and SCL4446B phase-locked loops contain two phase comparators, a voltage-controlled oscillator (VCO), source follower, and zener diode. The comparators have two common inputs. The Signal Input can be used directly coupled to large voltage signals, or indirectly coupled (with a series capacitor) to small voltage signals. The self-bias circuit adjusts small voltage signals in the linear region of the amplifier. Phase comparator I (an exclusive-OR gate) provides a digital error signal PCI_{out}, and maintains 90° phase shift at the center frequency between Signal and Comparator inputs (both at 50% duty cycle). Phase comparator II (with leading edge sensing logic) provides digital error signals $PCII_{out}$ and Phase Pulses, and maintains a 0° phase shift between input signals (duty cycle is immaterial). The linear VCO produces an output signal VCO_{out} whose frequency is determined by the voltage of input VCO_{in} and the capacitor and resistors connected to pins $C1_A$, $C1_B$, R1, and R2. The source follower output, Demod Out, with an external resistor is used where the VCO_{in} signal is needed but no loading can be tolerated. The inhibit input Inh, when high, disables the VCO and source follower to minimize standby power consumption. The zener diode can be used to assist in power supply regulation.

CONNECTION DIAGRAM
(all packages)



Add suffix for package:

C	16-pin Cerdip
D	16-pin Ceramic
E	16-pin Epoxy
F	16-pin Flat
H	Chip

RECOMMENDED OPERATING CONDITIONS

For maximum reliability:

DC Supply Voltage	$V_{DD} - V_{SS}$	3 to 15	Vdc
Operating Temperature	T_A		
C, D, F, H Device		-55 to +125	°C
E Device		-40 to +85	°C

BLOCK DIAGRAM

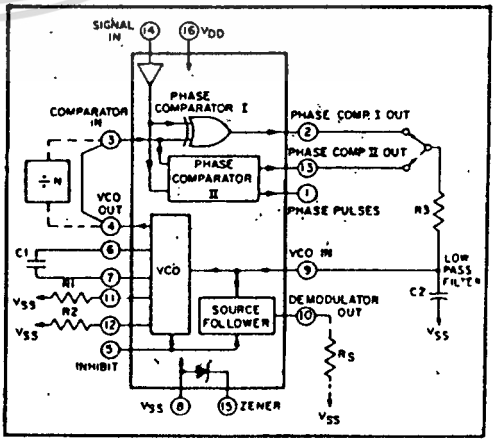


Fig. 1

VCO SECTION

The VCO requires one external capacitor (C1) and one to two external resistors (R1 or R1 and R2). Resistor R1 and capacitor C1 determine the frequency range of the VCO and resistor R2 enables the VCO to have a frequency offset if required. The high input impedance ($10^{12}\Omega$) of the VCO simplifies the design of low-pass filters by permitting the designer a wide choice of resistor-to-capacitor ratios. In order not to load the low-pass filter, a source-follower output of the VCO input voltage is provided at terminal 10 (DEMODULATOR OUTPUT). If this terminal is used, a load resistor (R_S) of 50kΩ or more should be connected from this terminal to V_{SS}. If unused, this terminal should be left open. The VCO can be connected directly or through frequency dividers to the comparator input of the phase comparators. A full CMOS logic swing is available at the output of the VCO. A logic 0 on the INHIBIT input "enables" the VCO and the source follower, while a logic 1 "turns off" both to minimize stand-by power consumption.

PHASE COMPARATORS

The phase-comparator signal input (terminal 14) can be direct-coupled provided the signal swing is within CMOS logic levels [logic "0" $\leq$ 30% (V_{DD}-V_{SS}), logic "1" $\geq$ 70% (V_{DD}-V_{SS})]. For smaller swings the signal must be capacitively coupled to the self-biasing amplifier at the signal input.

Phase comparator I is an exclusive-OR network; it operates analogously to an over-driven balanced mixer. To maximize the lock range, the signal and comparator-input frequencies must have a 50% duty cycle. With no signal or noise on the signal input, this phase comparator has an average output voltage equal to $V_{DD}/2$. The low-pass filter connected to the output of phase comparator I supplies the averaged voltage to the VCO input, and causes the VCO to oscillate at the center frequency (f_o).

The frequency range of input signals on which the PLL will lock, if it was initially out of lock, is defined as the frequency capture range ($2f_c$).

The frequency range of input signals on which the loop will stay locked if it was initially in lock is defined as the frequency lock range ($2f_L$). The capture range can not exceed the lock range.

With phase comparator I, the range of frequencies over which the PLL can acquire lock (capture range) is dependent on the low-pass-filter characteristics, and can be made as large as the lock range. Phase-comparator I enables a PLL system to remain in lock in spite of high amounts of noise in the input signal.

One characteristic of this type of phase comparator is that it may lock onto input frequencies that are close to harmonics of the VCO center-frequency. A second characteristic is that the phase angle between the signal and the comparator input varies between 0° and 180°, and is 90° at the center frequency. Figure 2 shows the (typical) triangular phase-to-output response characteristic of phase-comparator I. Typical waveforms for a CMOS phase-locked-loop employing phase comparator I in locked condition is shown in Figure 3.

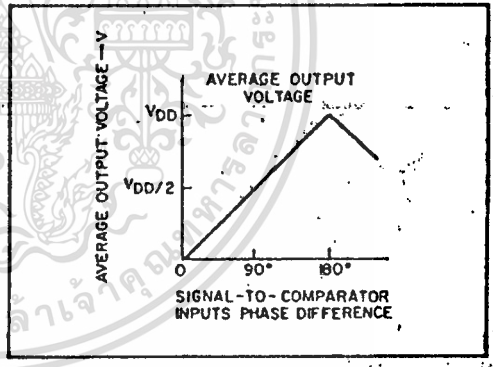


Fig. 2 — Phase-comparator I characteristics at low-pass filter output.

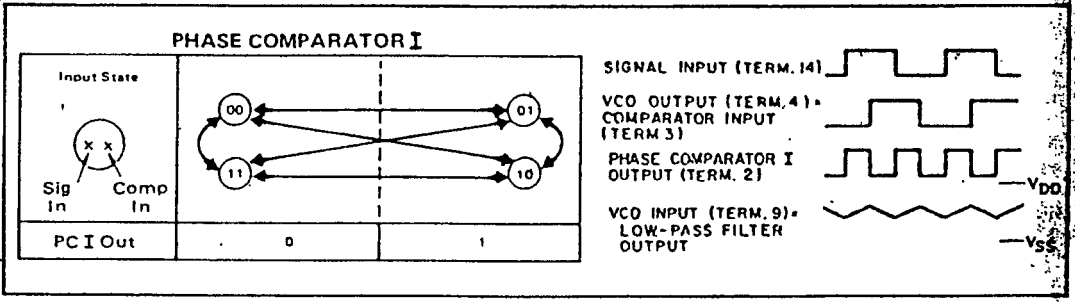


Fig. 3 — Typical waveforms employing phase comparator I in locked condition

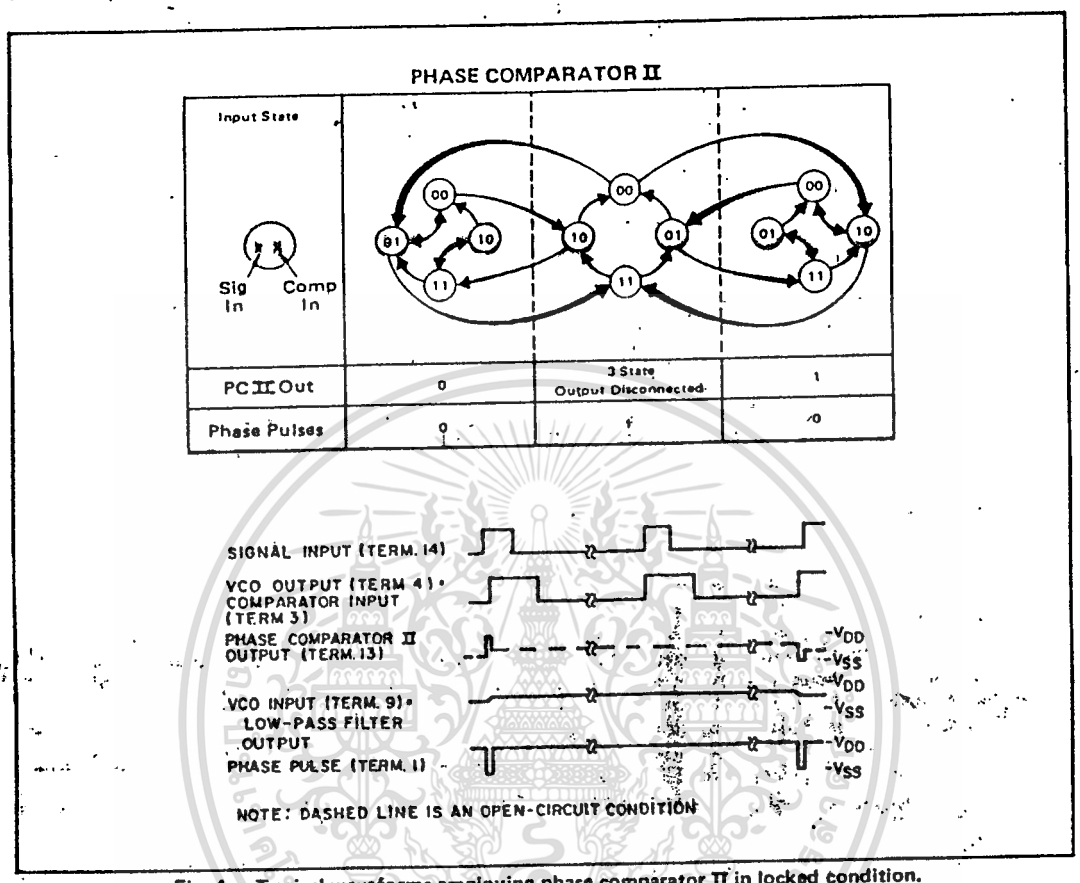


Fig. 4 — Typical waveforms employing phase comparator II in locked condition.

Phase-comparator II is an edge-controlled digital memory network. It consists of several flip-flop stages, control gating, and a three state output circuit comprising p- and n-type drivers having a common output node. When the p-MOS or n-MOS drivers are ON, they pull the output up to V_{DD} or down to V_{SS}, respectively. This type of phase comparator acts only on the positive edges of the signal and comparator inputs. The duty cycles of the signal and comparator inputs are not important since positive transitions control the PLL system utilizing this type of comparator. If the signal lags the comparator input in phase, the n-type output driver is maintained ON for a time corresponding to the phase difference. If the comparator input lags the signal in phase, the p-type output driver is maintained ON for a time corresponding to the phase difference. Subsequently, the capacitor voltage of the low-pass filter connected to this phase comparator is adjusted until the signal and comparator inputs are equal in both phase and frequency. At this stable point, both p- and n-type output drivers remain OFF. Thus, the phase comparator output becomes an open circuit and holds the voltage on the capacitor of the low-pass filter constant. Moreover, the signal at the "phase pulses" output is a high level which can be used for indicating a locked condition. Thus, for phase comparator II, no phase difference exists between signal and comparator input over the full VCO frequency range. Moreover, the power dissipation due to the low-pass filter is reduced when this type of phase comparator is used because both the p- and n-type output drivers are OFF for most of the signal input cycle.

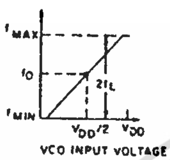
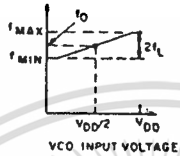
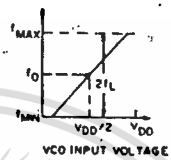
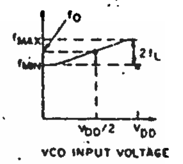
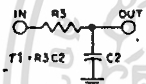
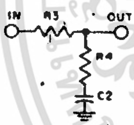
It should be noted that the PLL lock range for this type of phase comparator is equal to the capture range, independent of the low-pass filter. With no signal present at the signal input, the VCO is adjusted to its lowest frequency for phase comparator II. Figure 4 shows typical waveforms for a CMOS PLL employing phase comparator II in a locked condition.

DESIGN INFORMATION

This information is a guide for approximating the values of external components for the SCL4046B and SCL4446B in a Phase-Locked Loop system. The selected external components must be within the following ranges:

$R1, R2 \geq 2k\Omega, R_S \geq 10k\Omega$
$C1 \geq 15pF$

In addition to the given design information refer to Figure 5 for R1, R2, and C1 component selections.

CHARACTERISTICS	USING PHASE COMPARATOR I: VCO WITHOUT OFFSET $R_2 = \infty$	USING PHASE COMPARATOR I: VCO WITH OFFSET	USING PHASE COMPARATOR II: VCO WITHOUT OFFSET $R_2 = \infty$	USING PHASE COMPARATOR II: VCO WITH OFFSET
VCO Frequency				
For No Signal Input	VCO in PLL system will adjust to center frequency, f_o		VCO in PLL system will adjust to lowest operating frequency, f_{min}	
Frequency Lock Range, $2f_L$	$2f_L$ = full VCO frequency range $2f_L = f_{max} - f_{min}$			
Frequency Capture Range, $2f_C$	 $2f_C \approx \frac{1}{\pi}\sqrt{\frac{2\pi f_L}{\tau 1}}$		$f_C = f_L$	
Loop Filter Component Selection	 For $2f_C$, see Ref.			
Phase Angle between Signal and Comparator	90° at center frequency (f_o), approximating 0° and 180° at ends of lock range ($2f_L$)		Always 0° in lock	
Locks on Harmonics of Center Frequency	Yes		No	
Signal Input Noise Rejection	High		Low	
VCO Component Selection	– Given: f_o – Use f_o with Fig.5a to determine R1 and C1	– Given: f_o and f_L – Calculate f_{min} from the equation $f_{min} = f_o - f_L$ – Use f_{min} with Fig. 5b to determine R2 and C1 – Calculate $\frac{f_{max}}{f_{min}}$ from the equation $\frac{f_{max}}{f_{min}} = \frac{f_o + f_L}{f_o - f_L}$ – Use $\frac{f_{max}}{f_{min}}$ with Fig.5c to determine ratio R2/R1 to obtain R1	– Given: f_{max} – Calculate f_o from the equation $f_o = \frac{f_{max}}{2}$ – Use f_o with Fig.5a to determine R1 and C1	– Given: f_{min} & f_{max} – Use f_{min} with Fig.5b to determine R2 and C1 – Calculate $\frac{f_{max}}{f_{min}}$ – Use $\frac{f_{max}}{f_{min}}$ with Fig.5c to determine ratio R2/R1 to obtain R1

REF. G. S. Moschytz, "Miniaturized RC Filters Using Phase-Locked Loop", BSTJ, May, 1965.

ELECTRICAL CHARACTERISTICS [1, 4]

PARAMETER		V_{DD} (Vdc)	CONDITIONS	T_{LOW} [2]		+25°C			T_{HIGH} [2]		Units
				Min.	Max.	Min.	Typ.	Max.	Min.	Max.	
QUIESCENT DEVICE CURRENT	I_{DD}	5	Inhibit = V_{DD}	–	5	–	0.05	5	–	150	μAdc
		10	Signal Input =	–	10	–	0.01	10	–	300	
		15	V_{DD}	–	20	–	0.2	20	–	600	
TOTAL POWER DISSIPATION	P_T		Inh = V_{SS}, $VCO_{IN} = \frac{V_{DD}}{2}$, f_o = 10kHz,								
		5	C_L = 15pF	–	–	–	0.07	–	–	–	mW
		10	R1 = 1MΩ,	–	–	–	0.6	–	–	–	
		15	R2 = R_S = ∞	–	–	–	2.4	–	–	–	

NOTES: [1] Remaining Static Electrical Characteristics are listed under "SCL4000B Series Family Specifications".
[2] T_{LOW} = -55°C for C, D, F, H device.
= -40°C for E device.
T_{HIGH} = +125°C for C, D, F, H device.
= + 85°C for E device.
[3] VCO output (pin 4) and Phase Comparator Outputs (pins 2 and 13) have been designed for balanced output drive current specifications. Consult Family Specifications.

PARAMETER		CONDITIONS		V_{DD}	25°C Min.	25°C Typ.	25°C Max.	UNIT
VCO SECTION								
MAXIMUM OPERATING FREQUENCY	f_{max}							
SCL4046B			R1 C1					
		R2 = ∞, VCO_{IN} = V_{DD}	10k 50pF	5	0.5	0.8	–	MHz
				10	1.0	1.5	–	
				15	1.3	1.9	–	
			5k 50pF	5	0.6	1.0	–	MHz
				10	1.4	2.1	–	
				15	1.8	2.7	–	
			2k 50pF	5	–	1.3	–	MHz
				10	–	2.9	–	
				15	–	3.8	–	
SCL4446B		R2 = ∞, VCO_{IN} = V_{DD}	R1 C1					
			10k 50pF	5	0.7	1.0	–	MHz
				10	1.3	2.0	–	
				15	1.9	2.8	–	
			5k 50pF	5	0.9	1.3	–	MHz
				10	1.9	2.9	–	
				15	2.6	3.9	–	
			2k 50pF	5	–	1.8	–	MHz
				10	–	3.9	–	
				15	–	5.4	–	
LINEARITY		R2 = ∞, VCO_{IN} = 2.5±0.3V, R1>10kΩ		5	–	1	–	%
		VCO_{IN} = 5.0±2.5V, R1>400kΩ		10	–	1	–	
		VCO_{IN} = 7.5±5.0V, R1>1MΩ		15	–	1	–	

ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		CONDITIONS	V_{DD}	+25°C			UNIT
				Min.	Typ.	Max.	
VCO SECTION (Continued)							
TEMPERATURE-FREQUENCY STABILITY							
No Offset		R2 = ∞	5	–	0.12-0.24	–	%/°C
			10	–	0.04-0.08	–	
			15	–	0.015-0.03	–	
With Offset		R2 < 10X R1	5	–	0.06-0.12	–	%/°C
			10	–	0.05-0.1	–	
			15	–	0.03-0.06	–	
INPUT RESISTANCE (VCO_{IN})	R_{IN}		5, 10, 15	–	10^6	–	MΩ
OUTPUT DUTY CYCLE		All valid input combinations and voltages		–	50	–	%
OUTPUT TRANSITION TIME	t_{TLH}, t_{THL}	C_L = 50pF	5	–	100	200	ns
			10	–	50	100	
			15	–	40	80	
PHASE COMPARATORS							
INPUT RESISTANCE Signal Input	R_{IN}		5	1	3	–	MΩ
			10	0.2	0.7	–	
			15	0.1	0.3	–	
Comparator Input	R_{IN}		5, 10, 15	–	10^6	–	MΩ
AC-COUPLED INPUT SENSITIVITY Signal Input	V_{IN}		5	–	200	400	mV
			10	–	400	800	
			15	–	700	1400	
OUTPUT TRANSITION TIME PCI, PCII Outputs	t_{TLH}, t_{THL}	C_L = 50pF	5	–	100	200	ns
			10	–	50	100	
			15	–	40	80	
Phase Pulses Output	t_{TLH}, t_{THL}		5	–	130	260	ns
			10	–	65	130	
			15	–	50	100	
DEMODULATOR OUTPUT							
OFFSET VOLTAGE	VCO_{IN}-V_{DEM}	$R_S \geq 50k\Omega$	5	–	1.4	2.2	Vdc
			10	–	1.6	2.2	
			15	–	1.8	2.2	
LINEARITY		$R_S \geq 50k\Omega$					
		VCO_{IN} = 2.5±0.3V	5	–	0.1	–	%
		VCO_{IN} = 5.0±2.5V	10	–	0.6	–	
		VCO_{IN} = 7.5±5.0V	15	–	0.8	–	
ZENER DIODE							
ZENER VOLTAGE	V_Z	I_Z = 50μA	–	6.3	7.0	7.7	V
DYNAMIC RESISTANCE	R_Z	I_Z = 1mA	–	–	100	–	Ω

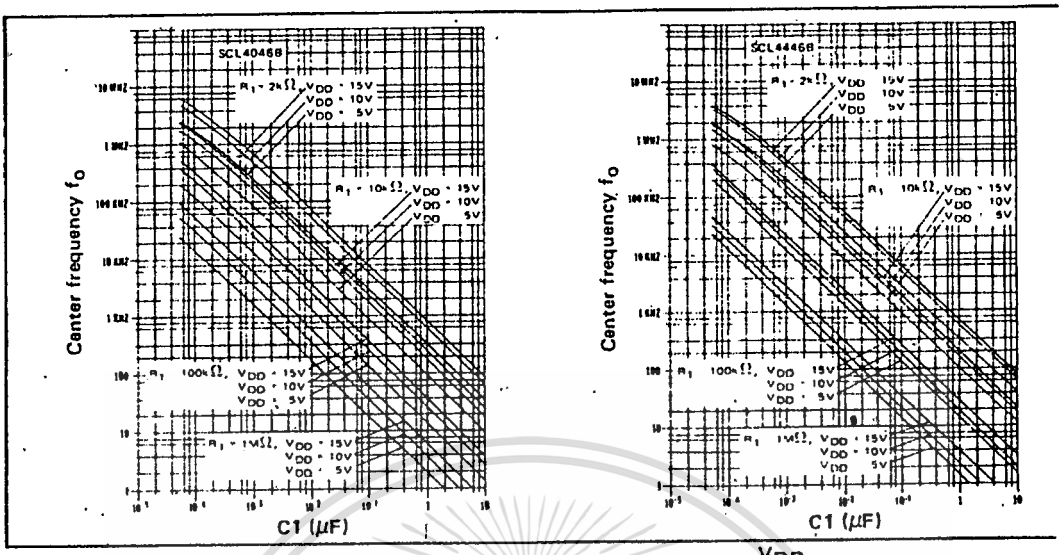


Fig. 5 (a) Typical center frequency (f_o) vs C1 (R2 = ∞, $VCO_{IN} = \frac{V_{DD}}{2}$, T_A = 25°C)

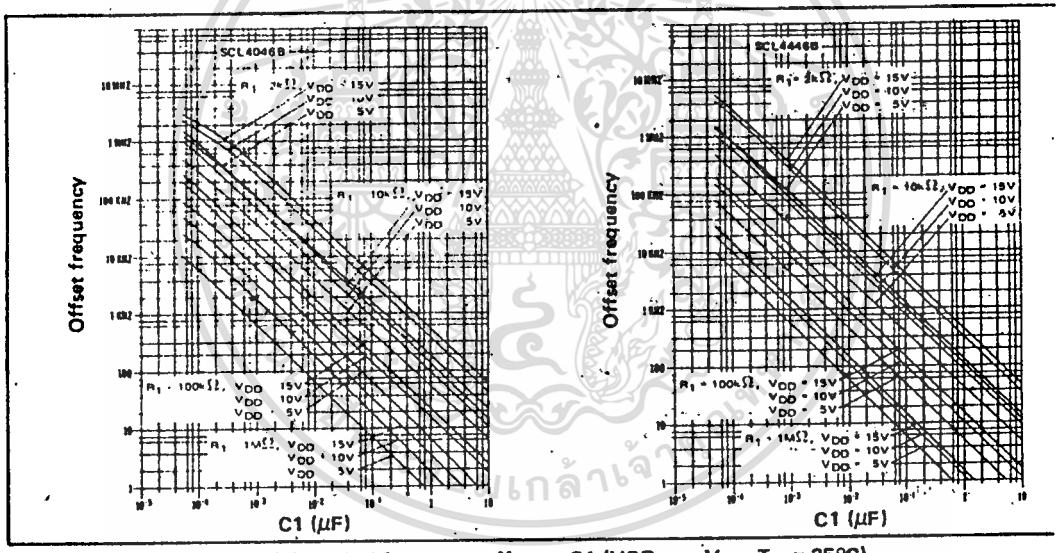


Fig. 5 (b) Typical frequency offset vs C1 ($VCO_{IN} = V_{SS}$, T_A = 25°C)

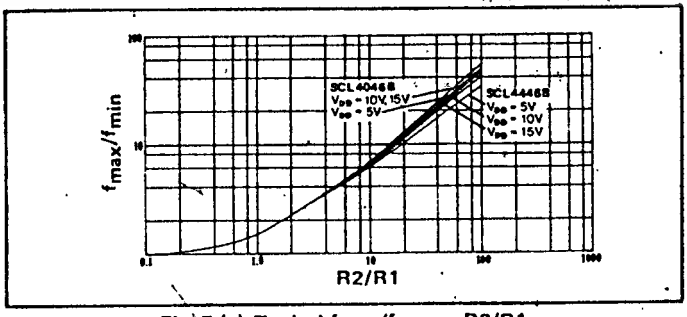


Fig. 5 (c) Typical f_{max}/f_{min} vs R2/R1

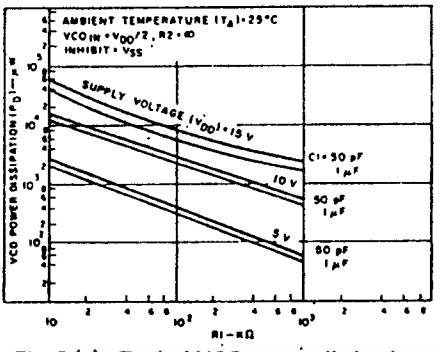


Fig. 6 (a) - Typical VCO power dissipation at center frequency vs R1.

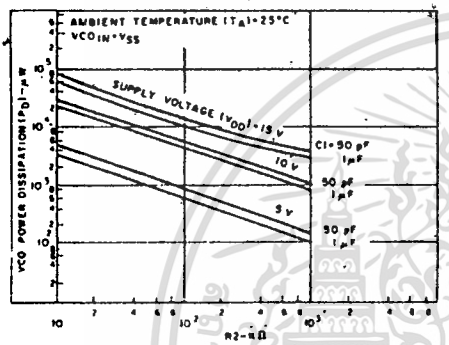


Fig. 6 (b) - Typical VCO power dissipation at f_{min} vs R2.

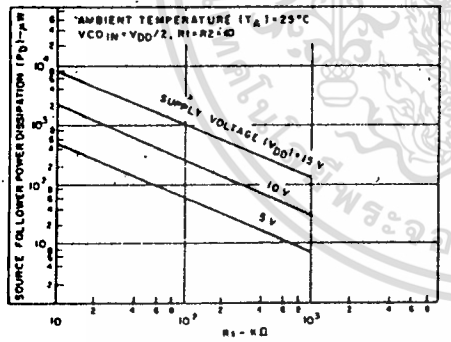


Fig. 6 (c) - Typical source follower power dissipation vs R_S.

NOTE: To obtain approximate total power dissipation of PLL system for no-signal input

P_D (Total) = P_D (f_o) + P_D (f_{MIN}) + P_D (R_S)
— Phase Comparator I

P_D (Total) = R_D (f_{MIN})
— Phase Comparator II

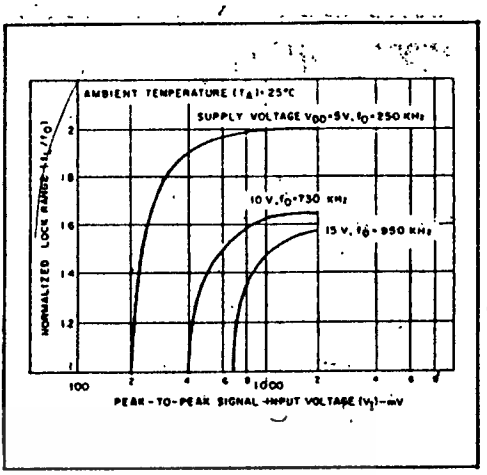


Fig. 7 — Typical lock range vs signal input amplitude

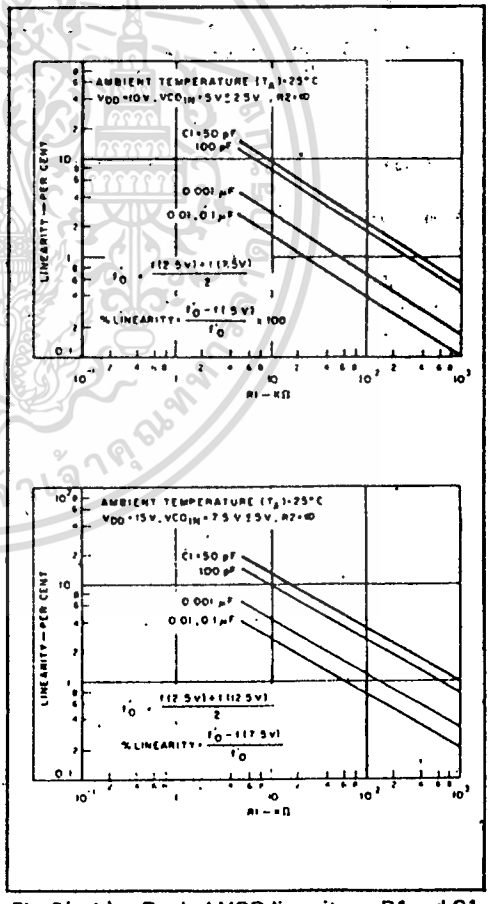


Fig. 8(a, b) — Typical VCO linearity vs R1 and C1

MITEL

ISO²-CMOS™ MT8870A
Integrated DTMF Receiver
Preliminary Information

APRIL 1983

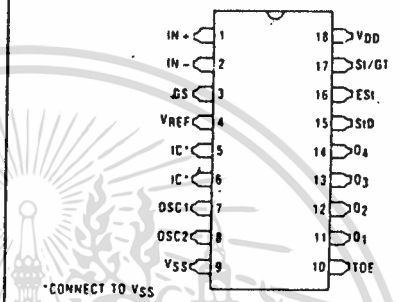
Features

- Full receiver in single 18-pin package
- Central Office quality
- Low power consumption
- Adjustable acquisition and release times

Applications

- PABX
- Central Office
- Key Systems
- Mobile Radio
- Remote Control
- Remote Data Entry

Pin Connections



Ordering Information

MT8870AC 18-PIN CERDIP

Description

The MT8870 is a complete DTMF receiver integrating both the bandsplit filter and digital decoder functions, fabricated in Mitel's double-poly ISO²-CMOS™ technology. The filter section uses switched capacitor techniques for high- and low-group filters and dial-tone rejection; the decoder uses digital counting techniques to detect and decode all 16 DTMF tone-pairs into a 4-bit code. External component count is minimized by on-chip provision of a differential input amplifier, clock oscillator and latched 3-state bus interface.

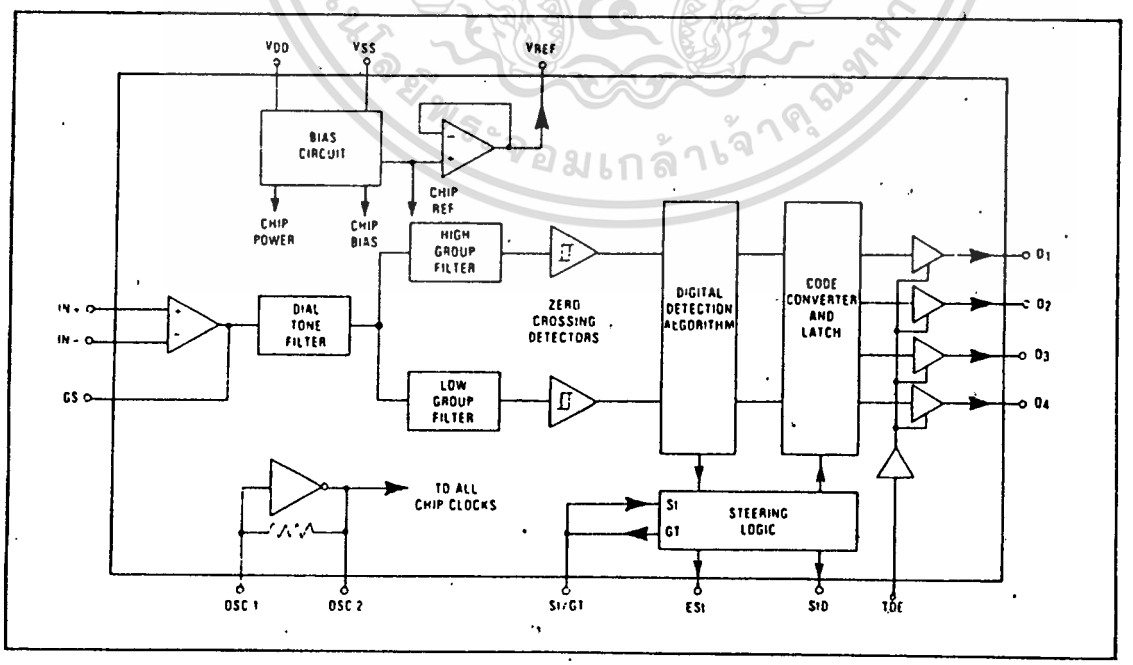


Fig. 1 Functional Block Diagram

Absolute Maximum Ratings[1]

Parameters	Min.	Max.	Units
Power Supply Voltage $V_{DD} - V_{SS}$		6	V
Voltage on any pin	V_{SS} -0.3	V_{DD} +0.3	V
Current at any pin		10	mA
Operating temperature	-40	+85	°C
Storage temperature	-65	+150	°C
Package power dissipation[2]		1000	mW

[1] Exceeding these ratings may cause permanent damage. Functional operation under these conditions is not implied.
[2] Derate above 75°C at 16 mW/°C. All leads soldered to board.

DC Electrical Characteristics

		Characteristics		Symbol	Min.	Typ.	Max.	Units	Test Conditions†
1	SUPPLY	Operating Supply Voltage		V_{DD}	4.75		5.25	V	
2		Operating Supply Current		I_{DD}		3.0	7	mA	
3		Power Consumption		P_C		15	35	mW	f = 3.579 MHz; V_{DD} = 5V
4	INPUTS	Low Level Input Voltage		V_{IL}			1.5	V	
5		High Level Input Voltage		V_{IH}	3.5			V	
6		Input Leakage Current		I_{IH}/I_{IL}		0.1		uA	V_{IN} = V_{SS} or V_{DD}
7		Pull Up (Source) Current		I_{SO}					TOE (Pin 10) = O V
8		Input Impedance	Signal Inputs 1,2	R_{IN}		10		Meg Ω	@ 1 kHz
9		Steering Threshold Voltage		V_{TSI}		2.45		V	
10	OUTPUTS	Low Level Output Voltage		V_{OL}		0.03		V	No Load
11		High Level Output Voltage		V_{OH}		4.97		V	No Load
12		Output Low (Sink) Current		I_{OL}	1.0	2.5		mA	V_{OUT} = 0.4 V
13		Output High (Source) Current		I_{OH}	0.4	0.8		mA	V_{OUT} = 4.6 V
14		Output Voltage	V_{REF}	V_{REF}	2.4		2.7	V	No Load
15		Output Resistance	V_{REF}	R_{OR}		10		KΩ	

† V_{DD} = 5 V, V_{SS} = 0 V, T_A = 25°C

Operating Characteristics

Gain Setting Amplifier

	Characteristics	Symbol	Min.	Typ.	Max.	Units	Test Conditions †
1	Input Leakage Current	I_{IN}		±100		nA	$V_{SS} < V_{IN} < V_{DD}$
2	Input Resistance	R_{IN}		10		MΩ	
3	Input Offset Voltage	V_{os}		±25		mV	
4	Power Supply Rejection	PSRR		60		dB	1 kHz
5	Common Mode Rejection	CMRR		60		dB	$-3.0\ V < V_{IN} < 3.0V$
6	DC Open Loop Voltage Gain	A_{VOL}		65		dB	
7	Open Loop Unity Gain Bandwidth	f_c		1.5		MHz	
8	Output Voltage Swing	V_o		4.5		V_{pp}	$R_L \geq 100\ k\Omega$ to V_{SS}
9	Tolerable capacitive load (GS)	C_L		100		pF	
10	Tolerable resistive load (GS)	R_L		50		KΩ	
11	Common Mode Range	V_{cm}		3.0		V_{pp}	No Load

† $V_{DD} = 5\ V$, $V_{SS} = 0\ V$, $T_A = 25°C$

AC Characteristics

	Characteristics			Symbol	Min.	Typ.	Max.	Units	Notes
SIGNAL CONDITIONS	Valid Input Signal Levels (each tone of composite signal)	MIN					-21	dBm	1,2,3,4,5,8
							69.1	mV_{RMS}	1,2,3,4,5,8
		MAX			+1			dBm	1,2,3,4,5,8
					883			mV_{RMS}	
	Twist Accept Limit		Positive			10		dB	2,3,4,8
			Negative			10		dB	
	Freq. Deviation Accept Limit						±2.5%	Nom.	2,3,5,8
	Freq. Deviation Reject Limit				±3.5%			Nom.	2,3,5

AC Characteristics

	Characteristics		Symbol	Min.	Typ.	Max.	Units	Notes
	Third Tone Tolerance				−16		dB	2,3,4,5,8,9
	Noise Tolerance				−12		dB	2,3,4,5,6,8,9
	Dial Tone Tolerance				− 18		dB	2,3,4,5,7,8,9
TIMING	Tone Present Detection Time		t_{DP}	6	11	14	mS	Refer to Fig. 4
	Tone Absent Detection Time		t_{DA}	1	4	8	mS	
	Tone Duration Accept		t_{REC}			40	mS	(User Adjustable) Times shown are obtained with circuit in Fig. 2
	Interdigit Pause Accept		t_{ID}			40	mS	
OUTPUTS	Propagation Delay (St to Q)		t_{PQ}		8	11	µS	TOE = V_{DD}
	Propagation Delay (St to StD)		t_{PSID}		12		µS	
	Output Data Set Up (Q to Std)		t_{QSID}		3.4		µS	
	Propagation Delay (TOE to Q)	ENABLE	t_{PTE}		50		nS	R_L = 10 kΩ C_L = 50pf
		DISABLE	t_{PTD}		300		nS	
CLOCK	Crystal/Clock Frequency		f_{CLK}	3.5759	3.5795	3.581	MHz	
	Clock Output (OSC2)	Capacitive Load	C_{LO}			30	pf	

† Unless noted V_{DD} = 5V, T_A = 25°C, f_{CLK} = 3.579545 MHz, using test circuit of Fig. 2.

NOTES

1. dBm = decibels above or below a reference power of 1 mW into a 600 ohm load.
2. Digit sequence consists of all 16 DTMF tones.
3. Tone duration = 40 mS. Tone pause = 40 mS.
4. Nominal DTMF frequencies are used.
5. Both tones in the composite signal have an equal amplitude.
6. Bandwidth limited (0 to 3 kHz) Gausian Noise.
7. The precise dial tone frequencies are (350 Hz and 440 Hz) ±2%.
8. For an error rate of better than 1 in 10,000.
9. Referenced to lowest level frequency component in DTMF signal.

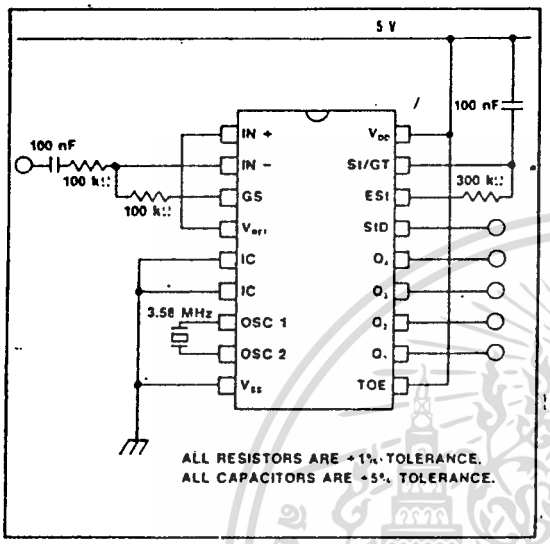


Fig. 2, Single Ended Input Configuration

F_{low}	F_{high}	KEY	TOE	Q_4	Q_3	Q_2	Q_1
697	1209	1	H	0	0	0	1
697	1336	2	H	0	0	1	0
697	1477	3	H	0	0	1	1
770	1209	4	H	0	1	0	0
770	1336	5	H	0	1	0	1
770	1477	6	H	0	1	1	0
852	1209	7	H	0	1	1	1
852	1336	8	H	1	0	0	0
852	1477	9	H	1	0	0	1
941	1336	0	H	1	0	1	0
941	1209	*	H	1	0	1	1
941	1477	#	H	1	1	0	0
697	1633	A	H	1	1	0	1
770	1633	B	H	1	1	1	0
852	1633	C	H	1	1	1	1
941	1633	D	H	0	0	0	0
–	–	ANY	L	Z	Z	Z	Z

L = LOGIC LOW, H = LOGIC HIGH, Z = HIGH IMPEDANCE

Fig. 3 Functional Decode Table

Functional Description

The MT8870 monolithic DTMF receiver offers small size, low power consumption and high performance. Its architecture consists of a bandsplit filter section, which separates the high and low tones of a receiver pair, followed by a digital counting section which verifies the frequency and duration of the received tones before passing the corresponding code to the output bus.

Filter Section

Separation of the low-group and high-group tones is achieved by applying the dual-tone signal to the inputs of two ninth-order switched-capacitor bandpass filters, the bandwidths of which correspond to the bands enclosing the low-group and high-group tones (see Fig. 5). The filter section also incorporates notches at 350 Hz and 440 Hz for exceptional dial-tone rejection. Each filter output is followed by a single-order switched-capacitor section which smooths the signals prior to limiting. Limiting is performed by high-gain comparators which are provided with hysteresis to prevent detection of unwanted low-level signals and noise; the outputs of the comparators provide full-rail logic swings at the frequencies of the incoming tones.

Decoder Section

The decoder uses digital counting techniques to determine the frequencies of the limited tones and to verify that they correspond to standard DTMF frequencies. A complex averaging algorithm protects against tone simulation by extraneous signals, such as voice, while providing tolerance to small frequency deviations and variations. This averaging algorithm has been developed to ensure an optimum combination of immunity to "talk-off" and tolerance to the presence of interfering signals ("third tones") and noise. When the detector recognizes the simultaneous presence of two valid tones (referred to as "signal condition", in some industry specifications), it raises the "early steering" flag (ESt). Any subsequent loss of signal-condition will cause ESt to fall.

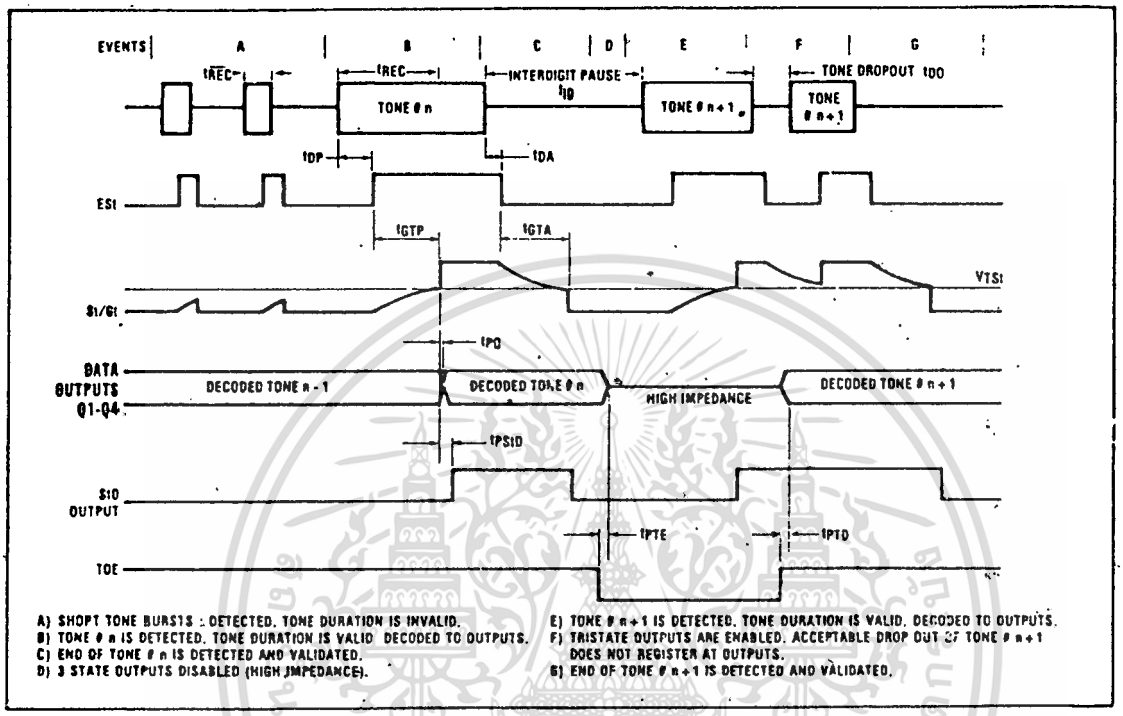


Fig. 4 Timing Diagram

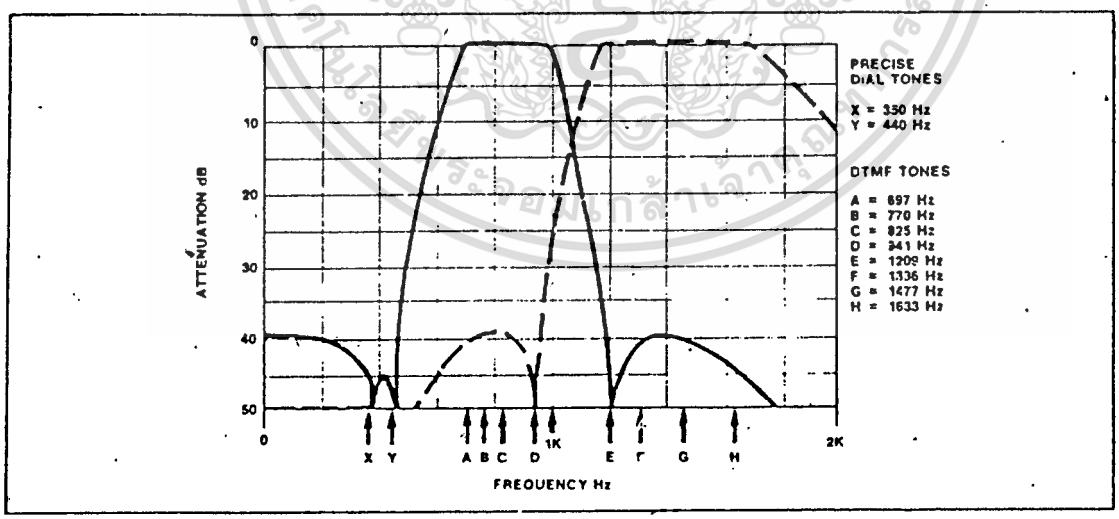


Fig. 5 Typical Filter Characteristic

INSTRUCTION SET SUMMARY 8748

Table 4-1. 48-Series Instruction Set Summary

MNEMONIC	FUNCTION	DESCRIPTION	CYCLES	BYTES	FLAGS C AC F0 F1
		ACCUMULATOR			
ADD A, #data	(A) ← (A) + data	Add immediate the specified Data to the Accumulator.	2	2	• •
ADD A, Rr	(A) ← (A) + (Rr) for r = 0 - 7	Add contents of designated register to the Accumulator.	1	1	• •
ADD A, @ Rr	(A) ← (A) + ((Rr)) for r = 0 - 1	Add indirect the contents the data memory location to the Accumulator	1	1	• •
ADDC A, #data	(A) ← (A) (C) + data	Add immediate with carry the specified data to the Accumulator.	2	2	• •
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0 - 7	Add with carry the contents of the designated register to the Accumulator	1	1	• •
ADDC A, @ Rr	(A) ← (A) + (C) + ((Rr)) for r = 0 - 1	Add indirect with carry the contents of data memory location to the Accumulator.	1	1	• •
ANL A, #data	(A) ← (A) AND data	Logical AND specified immediate Data with Accumulator.	2	2	
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0 - 7	Logical AND contents of designated register with Accumulator.	1	1	
ANL A, @ Rr	(A) ← (A) AND ((Rr)) for r = 0 - 1	Logical AND indirect the contents of data memory with Accumulator.	1	1	
CPL A	(A) ← NOT (A)	Complement the contents of the Accumulator.	1	1	
CLR A	(A) ← 0	CLEAR the contents of the Accumulator	1	1	
DA A		DECIMAL ADJUST the contents of the Accumulator.	1	1	•
DEC A	(A) ← (A) - 1	DECREMENT by 1 the accumulator's contents	1	1	
INC A	(A) ← (A) + 1	Increment by 1 the accumulator's contents	1	1	
ORL A, #data	(A) ← (A) OR data	Logical OR specified immediate data with Accumulator.	2	2	
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0 - 7	Logical OR contents of designated register with Accumulator.	1	1	
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0 - 1	Logical OR indirect the contents of data memory location with Accumulator.	1	1	
RL A	(An+1) ← (An) for n = 0 - 6 (A0) ← (A7)	Rotate Accumulator left by 1 bit without carry	1	1	
RLC A	(An+1) ← (An) n = 0 - 6 (A0) ← (C) (C) ← (A7)	Rotate Accumulator left by 1 bit through carry	1	1	
RR A	(An) ← (An+1) n = 0 - 6 (A7) ← (A0)	Rotate Accumulator right by 1 bit without carry	1	1	
RRC A	(An) ← (An+1) n = 0 - 6 (A7) ← (C) (C) ← (A0)	Rotate Accumulator right by 1 bit through carry	1	1	
SWAP A	(A4-A7) ↔ (A0 - A3)	Swap the 2, 4-bit nibbles in the Accumulator	1	1	
XRL A, #data	(A) ← (A) XOR data	Logical XOR immediate specified data with Accumulator	2	2	
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0 - 7	Logical XOR contents of designated register with Accumulator	1	1	
XRL A, @ Rr	(A) ← (A) XOR ((Rr)) for r = 0 - 1	Logical XOR indirect the contents of data memory location with Accumulator.	1	1	
		BRANCH			
DJNZ Rr, addr	(Rr) ← (Rr) - 1 if (Rr) ≠ 0 (PC 0-7) ← addr	Decrement the specified register and test contents	2	2	
JBb addr	(PC 0-7) ← addr if Bb = 1 (PC) ← (PC) + 2 if Bb = 0	Jump to specified address if Accumulator bit is set	2	2	
JC addr	(PC 0-7) ← addr if C = 1 (PC) ← (PC) +2 if C = 0	Jump to specified address if carry flag is set.	2	2	
JF0 addr	(PC 0-7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address if Flag F0 is set.	2	2	
JF1 addr	(PC 0-7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address if Flag F1 is set.	2	2	
JMP addr	(PC 8-10) ← addr 8-10 (PC 0-7) ← addr 0-7 (PC 11) ← DBF	Direct Jump to specified address within the 2K address block	2	2	
JMPP @ A	(PC 0-7) ← (A)	Jump indirect to specified address pointed to by the accumulator in current page.	2	1	
JNC addr	(PC 0-7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump to specified address if carry flag is low.	2	2	
JNI addr	(PC 0-7) ← addr if I = 0 (PC) ← (PC) + 2 if I = 1	Jump to specified address if interrupt is low.	2	2	
JNT0 addr	(PC 0-7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump to specified address if Test 0 is low	2	2	
JNT1 addr	(PC 0-7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump to specified address if Test 1 is low	2	2	
JNZ addr	(PC 0-7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump to specified address if accumulator is non-zero.	2	2	

INSTRUCTION SET SUMMARY 8748

Table 4-1. 48-Series Instruction Set Summary (Cont'd.)

MNEMONIC	FUNCTION	DESCRIPTION	CYCLES	BYTES	FLAGS C AC F0 F1
JTF addr	(PC 0-7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump to specified address if Timer Flag is set to 1.	2	2	
JT0 addr	(PC 0-7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump to specified address if Test 0 is a 1.	2	2	
JT1 addr	(PC 0-7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump to specified address if Test 1 is a 1.	2	2	
JZ addr	(PC 0-7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump to specified address if Accumulator is 0.	2	2	
		CONTROL			
EN I		Enables the External interrupt input.	1	1	
DIS I		Disable the External interrupt input.	1	1	
ENT0 CLK		Enable T0 as the Clock Output.	1	1	
SEL MB0	(DBF) ← 0	Select Bank 0 (addresses 0 - 2047) of Program Memory.	1	1	
SEL MB1	(DBF) ← 1	Select Bank 1 (addresses 2048 - 4095) of Program Memory.	1	1	
SEL RB0	(BS) ← 0	Select Bank 0 (addresses 0 - 7) of Data Memory.	1	1	
SEL RB1	(BS) ← 1	Select Bank 1 (addresses 24 - 31) of Data Memory.	1	1	
		DATA MOVES			
MOV A, #data	(A) ← #data	Move immediate the specified data into the Accumulator.	2	2	
MOV A, Rr	(A) ← (R(r)), r = 0 - 7	Move the contents of the designated registers into the Accumulator.	1	1	
MOV A, @ Rr	(A) ← ((R(r))), r = 0 - 1	Move (Indirect the contents of data memory) location into the Accumulator.	1	1	
		INPUT/OUTPUT (Cont'd.)			
MOVD A, Pp	(A 4-7) ← 0 (A 0-3) ← (Pp) p = 4,5,6,7	Move contents of designated port (4 - 7) into Accumulator.	2	1	
MOVD Pp, A	(Pp) ← (A) 4-7 p = 4,5,6,7	Move contents of Accumulator to designated port (4 - 7).	2	1	
ORL BUS, #data	(BUS) ← (BUS) OR data	Logical OR immediate specified data with contents of BUS.	2	2	
ORLD Pp, A	(Pp) ← (Pp) OR (A0 - 3), p = 4 - 7	Logical OR contents of Accumulator with designated port (4 - 7).	2	1	
ORL Pp, #data	(Pp) ← (Pp) OR data; p = 1, 2	Logical OR immediate specified data with designated port (1 - 2).	2	2	
OUTL BUS, A	(BUS) ← (A)	Output contents of Accumulator onto BUS.	2	1	
OUTL Pp, A	(Pp) ← (A), p = 1 - 2	Output contents of Accumulator to designated port (1 - 2).	2	1	
		REGISTERS			
DEC Rr	((Rr)) ← ((Rr)) - 1, r = 0-7	Decrement by 1 contents of designated register.	1	1	
INC Rr	(Rr) ← (Rr) + 1, r = 0-7	Increment by 1 contents of designated register.	1	1	
INC @ Rr	((Rr)) ← ((Rr)) + 1, r = 0-1	Increment indirect by 1 the contents of data memory location.	1	1	
		SUBROUTINE			
CALL addr	((SP)) ← (PC) ((SP)) ← (PSW 4-7) (SP) ← (SP) + 1 (PC8-10) ← addr 8-10 (PC 0-7) ← addr 0-7 (PC 11) ← DBF	Call designated Subroutine	2	2	
RET	(SP) ← (SP) - 1 (PC) ← ((SP))	Return from Subroutine without restoring Program Status Word	2	1	
RETR	(SP) ← (SP) - 1 (PC) ← ((SP)) (PSW 4-7) ← ((SP))	Return from Subroutine restoring Program Status Word.	2	1	
		FLAGS			
CPL C	(C) ← NOT (C)	Complement Content bi carry bit	1	1	
CPL F0	(F0) ← NOT (F0)	Complement Content of Flag F0	1	1	
CPL F1	(F1) ← NOT (F1)	Complement Content of Flag F1	1	1	
CLR C	(C) ← 0	Clear content of carry bit to 0	1	1	
CLR F0	(F0) ← 0	Clear content of Flag 0 to 0	1	1	
CLR F1	(F1) ← 0	Clear content of Flag 1 to 0	1	1	
		MISCELLANEOUS			
NOP		No operation	1	1	

5. โปรแกรมภาษาแอสเซมบลี้ สำหรับ ส่วนควบคุมการรับข้อมูลทางโทรศัพท์

```
INTI        SEL   RB1
            MOV   R7,A
            MOV   R0,#30
NEXTLN      MOV   A,@R0
            XRL   A,#11
            JNZ   NOTRDY
            CALL  INITPT
            MOV   R1,A
NEXTCH      MOVX  A,@R1
            CALL  OUTIBM
WAITI       JNI   CONT
            JMP   WAITI
CONT        INC   R1
            XRL   A,#16
            JNZ   NEXTCH
            MOV   @R0,#00
NOTRDY      INC   R0
            MOV   A,#38
            XRL   A,R0
            JNZ   NEXTLN
            DIS   I
            MOV   A,#FF
            CALL  OUTIBM
            MOV   A,R7
            RETR

TCNTI       DJNZ  R6,ENDCTI
            MOV   R4,A
            MOV   A,R2
            JB7   QUIET
            ORL   A,#80
            MOV   R2,A
            MOV   R6,#TC2
            CALL  CLOCK
            CALL  OUTP1
            MOV   A,R4
            RETR
QUIET       MOV   A,#40
            CALL  OUTP1
            INC   @R0
            MOV   R2,#FF
            STOP  T
            MOV   A,R4
            RETR

MAIN        MOV   A,#0C0H
            OUTL  P1,A
            MOV   A,#0F8H
            OUTL  P2,A
            MOV   R2,#FF
            MOV   R0,#37
            MOV   R7,#08
```

```
>type b:ct2
    LOOP        CLR   A
                MOV   @R0,A
                CALL  INITPT
                CALL  INITIME
                DEC   R0
                DJNZ  R7,LOOP
                DIS   I
                EN    TCNTI

    MAIN2       MOV   R0,#30
    MAIN3       MOV   A,@R0
                RL    A
                RL    A
                ADD   A,#OFFSET
                JMPP  @A
    OFFSET      CALL  STAT0
                JMP   PASS
                CALL  STAT1
                JMP   PASS
                CALL  STAT2
                JMP   PASS
                CALL  STAT3
                JMP   PASS
                CALL  STAT4
                JMP   PASS
                CALL  STAT3
                JMP   PASS
                CALL  STAT4
                JMP   PASS
                CALL  STAT3
                JMP   PASS
                CALL  STAT4
                JMP   PASS
                CALL  STAT9
                JMP   PASS
                CALL  STAT10
                JMP   PASS
                CALL  STAT11
                JMP   PASS
                CALL  STAT12
                JMP   PASS
    PASS        INC   R0
                MOV   A,R0
                XRL   A,#38
                JNZ   MAIN3
                JZ    MAIN2

    STAT0       MOV   A,#C0H
                CALL  OUTP1
                JNT0  ENDS0
                INC   @R0
    ENDS0       RET

A>
```

```
type b:ct2
STAT1           MOV   A,#C0H
                CALL  OUTP1
                JT0   ENDS1
                MOV   A,R2
                INC   A
                JNZ   ENDS1
                MOV   A,#40H
                CALL  OUTP1
                INC   @R0
                MOV   A,R0
                ANL   A,#07
                MOV   R2,A
                MOV   R6,#TC1
                CALL  CLOCK
ENDS1           RET

STAT2           RET

STAT3           MOV   A,#40H
                CALL  OUTP1
                JNT1  WAIT
                CALL  INTIME
                ORL   P2,#F0
                IN    A,P2
                SWAP  A
                ANL   A,#0F
                MOV   R3,A
                XRL   A,#RESET
                JZ    RESTART
                MOV   A,R3
                XRL   A,#ENTER
                JZ    SKIP
                INC   @R0
STORE           CALL  INCPT
                MOV   A,R3
                MOVX  @R1,A
                MOV   A,#1F
                ANL   A,R1
                XRL   A,#12
                JZ    OUTRNG
                RET
WAIT            MOV   A,R0
                ANL   A,#1F
                RL    A
                MOV   R1,A
                INC   @R1
                MOV   A,@R1
                JNZ   ENDS3
                INC   R1
                INC   @R1
                MOV   A,@R1
                JNZ   ENDS3
```

```
type b:ct8
                MOV     A,@R0
                ADD     A,#F8
                JNC     OUTRNG
                MOV     @R0,#07
                JMP     SKIP
OUTRNG          CLR     A
                MOV     @R0,A
                CALL    INITPT
ENDS3           RET
RESTART         CALL    INITPT
                MOV     @R0,#04
                RET
SKIP            MOV     A,@R0
                ADD     A,#0F
                MOV     R3,A
                INC     @R0
                INC     @R0
                INC     @R0
                JMP     STORE

STAT4           MOV     A,#40
                CALL    OUTP1
                JT1     ENDS4
                MOV     A,@R0
                DEC     A
                MOV     @R0,A
ENDS4           RET

STAT10          MOV     A,#40
                CALL    OUTP1
                JT1     ENDS10
                MOV     A,#C0
                CALL    OUTP1
                CALL    INITPT
                INC     @R0
                MOV     A,#FE
                CALL    OUTIBM
                EN      I
ENDS10          RET

STAT11          RET

OUTP1           ORL     A,R0
                ANL     A,#C7
                OUTL    P1,A
                RET

CLOCK           STOP    T
                CLR     A
                MOV     T,A
                STRT    T
                RET
```

```
type b:ct8
INITPT          MOV     A,R0
                ORL     A,#08
                MOV     R1,A
                ORL     A,#F0
                MOVP    A,@A
                MOV     @R1,A
                RET

OUTIBM          ANL     P2,#F7
                MOVX    @R1,A
                ORL     P2,#08
                RET

INTIME          MOV     A,R0
                ANL     A,#1F
                RL      A
                MOV     R1,A
                CLR     A
                MOV     @R1,A
                INC     R1
                MOV     @R1,A
                RET

INCPT           MOV     A,R0
                ORL     A,#08
                MOV     R1,A
                MOV     A,@R1
                INC     @R1
                MOV     R1,A
                RET
```

6. โปรแกรมภาษาแอสเซมบลี้ สำหรับ ส่วนประมวลผลและแสดงผลข้อมูล

```
TCNTI       MOV   R7,A          TIMER/COUNTER INTERUPT
            STOP  TCNT          SERVICE ROUTINE
            MOV   A,#TCT
            MOV   T,A
            MOV   R3,#TC3
TDELAY      DJNZ  R3,TDELAY
            STRT  T
            CALL  RDBIT
            MOV   A,R2
            RLC   A
            MOV   R2,A
            DJNZ  R5,OUT1
            MOV   @R0,A
            INC   A
            JZ    OUT3
            INC   R0
            DJNZ  R6,OUT
            CALL  DECODE
OUT3        MOV   R6,#03
            MOV   R0,#BUFF3
            CPL   F1
            MOV   A,R2
            SEL   RB0
            MOV   R2,A
OUT         STOP  TCNT
OUT2        JNI   OUT2
            EN    I
OUT1        SEL   RB0
            MOV   A,R7
            RETR

INTI        SEL   RB0           EXTERNAL INTERUPT
            MOV   R5,A          SERVICE ROUTINE
            SEL   RB1
            MOV   R5,#TC4
DELAY2      DJNZ  R5,DELAY2
            MOV   A,#TCT
            MOV   T,A
            STRT  T
            CALL  RDBIT
            JC    RETURN
            MOV   R5,#08H
            DIS   I
            SEL   RB0
            MOV   A,R5
            RETR
RETURN      STOP  TCNT
            SEL   RB0
            MOV   A,R5
            RETR
```

```
type b:pr2
   MAIN          MOV   R0,#BUFF1      MAIN PROGRAM
                 MOV   R1,#BUFF2      START UP
                 MOV   R7,#READY
                 MOV   R3,#0AH
   NEXT          MOV   A,R7
                 MOVP  A,@A
                 MOV   @R0,A
                 MOV   @R1,A
                 INC   R0
                 INC   R1
                 INC   R7
                 DJNZ  R3,NEXT
                 MOV   R2,#TC1
   NEXT2         MOV   R3,#TC2
   NEXT3         CALL  DISP
                 DJNZ  R3,NEXT3
                 DJNZ  R2,NEXT2
   CLEAR         CLR   A
                 OUTL  P1,A
                 CLR   F1
                 EN    I
                 EN    CNTI
                 SEL   RB1
                 MOV   R1,#TD1+1
                 MOV   @R1,#HITD
                 DEC   R1
                 MOV   @R1,#LOTD
                 MOV   R6,#03H
                 MOV   R0,#BUFF3
                 SEL   RB0

   MAIN3         CLR   F0             RESTART POINT
   MAIN2         CALL  WAITCH
                 JF0   SYNCIN
                 INC   A
                 JNZ   MAIN2          DETECT FOR SYNC
                 CPL   F0
                 JMP   MAIN2
   SYNCIN        INC   A
                 JZ    MAIN
                 MOV   A,R2
                 XRL   A,#HIBYTE      COMPARE WITH THE
                 JZ    PAGER          PAGER NUMBER
                 CALL  WAITCH
                 JMP   MAIN3
   PAGER         CALL  WAITCH
                 XRL   A,#LOBYTE
                 JNZ   MAIN3
                 CALL  WAITCH
                 INC   A
                 JZ    MAIN2
                 MOV   A,R2
```

```
type b:pr2
                SWAP  A
                ANL   A,#0FH
                MOV   R3,A
                MOV   R3,A
                MOV   A,R2
                ANL   A,#0FH
                MOV   R4,A
                ADD   A,R3
                ADD   A,#0F6H
                JC    MAIN3
                MOV   R1,#BUFF4
                CALL  PUTBF
                MOV   A,R2
                INC   A
                JZ    TSET1
                INC   A
                JZ    MAIN3
                MOV   A,#HIFEN
                MOV   @R1,A
                INC   R1
                JF0   SKIP
                MOV   A,R2
                MOV   @R1,A
                INC   R1
                DEC   R4
SKIP            MOV   A,R4
                MOV   R3,A
                CALL  PUTBF
                MOV   A,R2
                INC   A
TEST1           CLR   F0
                JNZ   TEST2
                CPL   F0
                JMP   MAIN2
                INC   A
                JZ    MAIN3
EMPTY           MOV   A,#BUFF4+10   FILL THE BUFFER
                XRL   A,R1          WITH BLANK
                JZ    FINISH
                MOV   A,#BLANK
                MOV   @R1,A
                INC   R1
                JMP   EMPTY
FINISH          ORL   P1,#80H       BEEP THE SPEAKER
                MOV   R0,#BUFF1
                MOV   R1,#BUFF2
                MOV   R6,#0AH
MOVE1           MOV   A,@R0
                MOV   @R1,A
                INC   R0
                INC   R1
                DJNZ  R6,MOVE1
```

```
type b:pr6
                MOV   R0,#BUFF1
                MOV   R1,#UFF4
                MOV   R6,#0AH
MOVE2           MOV   A,@R1
                MOV   @R0,A
                INC   R0
                INC   R1
                DJNZ  R6,MOVE2
                JMP   MAIN3

DISP            MOV   R0,#BUFF1+9     DISPLAY THE CONTENTS
LOOP            MOV   R6,#0AH         IN MEMORY
LOOP2           MOV   A,@R0
                MOVP3 A,@A
                OUTL  P1,A
                MOV   A,R6
                DEC   A
                SWAP  A
                OUTL  P2,A
                MOV   A,#20H
DELAY           JF1   END             IF THERE IS NEW DATA
                DEC   A               WILL JUMP TO END
                JNZ   DELAY
                DEC   R0
                DJNZ  R6,LOOP2
END             CLR   A
                OUTL  P1,A
                RET

WAITCH          JF1   DATAIN          WAIT FOR NEW DATA
                SEL   RB1             TEST THE KEY PRESSED
                MOV   A,@R1           AND DISPLAY DATA
                JNZ   NTDLO
                INC   R1
                MOV   A,@R1
                JNZ   NTDHI
                DEC   R1
                SEL   RB0
                JMP   W2
NTDHI           INC   @R1
                DEC   R1
NTDLO           INC   @R1
                MOV   A,#TD1
                XRL   A,R1
                SEL   RB0
                JZ    DISPB1
                JNZ   DISPB2
W2              JNTO  KEY
                JT1   WAITCH
                SEL   RB1
                MOV   R1,#TD2+1
                JMP   SETCK
```

```
type b:pr4
   DISPB2          MOV   R0,#BUFF2+9
                   CALL  LOOP
                   JMP   WAITCH
   KEY             SEL   RB1
                   MOV   R1,#TD1+1
   SETCK           MOV   @R1,#HITD
                   DEC   R1
                   MOV   @R1,#LOTD
                   SEL   RB0
                   JMP   WAITCH
   DISPB1          CALL  DISP
                   JMP   WAITCH
   DATAIN          MOV   A,R2
                   CLR   F1
                   RET

   PUTBF           CLR   F0              PUT DATA INTO
                   MOV   A,R3            THE BUFFER
                   JZ    BACK
   AGAIN           CALL  WAITCH
                   INC   A
                   JZ    BACK
                   MOV   A,R2
                   SWAP  A
                   ANL   A,#0FH
                   MOV   R6,A
                   ADD   A,#0F6H
                   JC    NDIGIT
                   MOV   A,#0FH
                   ANL   A,R2
                   MOV   R2,A
                   ADD   A,#0F6H
                   JC    NDIGIT
                   MOV   A,R6
                   MOV   @R1,A
                   INC   R1
                   DJNZ  R3,NZERO
                   RET
   NZERO           MOV   A,R2
                   MOV   @R1,A
                   INC   R1
                   DJNZ  R3,AGAIN
   BACK            CPL   F0
                   RET
   NDIGIT          MOV   R2,#0FEH
                   RET

   RDBIT           MOV   R3,#00          READ SERIAL DATA
                   MOV   R7,#29H         FROM INT LINE
   RB1             JNI   ZERO
                   INC   R3
                   NOP
```

```
type b:pr5
    RB2             DJNZ R7,RB1
                    MOV  A,R3
                    ADD  A,#0EBH
                    RET
    ZERO            JMP  RB2

    DECODE          MOV  R0,#BUFF3      GET THE CORRECT DATA
                    MOV  A,@R0
                    MOV  R2,A
                    INC  R0
                    MOV  A,@R0
                    MOV  R6,A
                    XRL  A,R2
                    INC  R0
                    ANL  A,@R0
                    MOV  R5,A
                    MOV  A,R2
                    ANL  A,R6
                    ORL  A,R5
                    MOV  R2,A
                    RET

A>
```

7. โปรแกรมภาษาซี สำหรับ ควบคุมและส่งผ่านข้อมูล

```
A>type b:p1
#include <conio.h>
#include <dos.h>
#include <fcntl.h>
#include <screen.c>
#include <call8048.c>
#include <o_port.c>
#include <recorde.c>

#define     HOUR_PORT    0x2c4
#define     MIN_PORT     0x2c3
#define     SEC_PORT     0x2c2
#define     RESET_PORT   0x302
#define     TX_PORT      0x300
#define     STAT_TX      0x303
#define     RX_PORT      0x300
#define     STAT_RX      0x301

struct USER{
            char name[20];
            char surname[20];
            char address[40];
            char tel[10];
            char number[5];
            }user;
struct CALL{
            int pager;
            char tel[10];
            char time[9];
            char date[9];
            }calling[100];
  int  ttx = 0; /*  times of transmission */

main()
  {
   int choice_number = 0;
   int stop = 0;
   int loop = 1;
   inportb(RESET_PORT);
  while (loop != stop)
    {
     choice_number = menu();
     loop = call(choice_number);
    }
  }
/*              FUNCTION RECIEVE KBD AND RETURN CHOICE
choice(columnlimit,width,space,ylimit,yspace,c0y,c0x)
  {
    int ch,cxx,cyy,cxstop,choiceno;
    int ch2 = 0;
    int column =1;
    int row = 1;
    int select = 13;
```

```
A>type b:p2
    cxx = c0x;
    cyy = c0y;
    fpoke (cyy,cxx,width,1);
   do
    {
     ch2=0;
     while(kbhit() == 0)                    /* CHECK KEYBOARD HIT */
        import8048();
     ch=getch();
   if(ch==0)
     ch2=getch();
   fpoke(cyy,cxx,width,0);          /* Erase old reverse block */
   switch(ch2)
    {
    case 'H' :if(row > 1)                              /* Move up */
                {
                cyy -= (1+yspace);
                 row--;
                 }
               break;
    case 'P' :if(row < ylimit)                         /* Move down */
                {
                cyy += (1+yspace);
                 row++;
                 }
               else
               { cyy = c0y;
                row = 1;}
               break;
    case 'M' :if(column < columnlimit)          /* Move to right */
                 {
                  cxx += (width + space);
                   column++;
                   }
              else
               {
               cxx =c0x;
                column = 1;
                }
             break;
    case 'K' :if(column > 1)                           /* Move to left */
                {
                cxx -= (width + space);
                 column--;
                 }
             break;
    }
    fpoke (cyy,cxx,width,1);
   } while (ch != select);
    fpoke (cyy,cxx,width,0);                          /*ERASE OLD BLOCK*/
    choiceno = (row-1)*columnlimit + column;
    return choiceno;
  }
```

```
A>type b:p2
/*          FUNCTION SHOW MAIN MENU
/*          CALL - menu()
menu()

{
clsr();
  printf("\n");
  printf("\n");
  printf("\n");
  printf("                                                MAIN MENU \n");
  printf("\n");
  printf("\n");
  printf("            1)Call user with his name              2)List a
record with user name\n");
  printf("\n");
  printf("\n");
  printf("            3)Call user with his number            4)List a
record with user number\n");
  printf("\n");
  printf("\n");
  printf("              5)Input new user data                  6) a
Delete user record\n");
  printf("\n");
  printf("\n");
  printf("              7)Set clock                            8) a
Exit\n");

  flame(7,1,79,17);                          /* Draw flame */
  draw(8,5,78,5,196);
  pokeb(0xb000,co_xy(7,5)-1,199);
  pokeb(0xb000,co_xy(79,5)-1,182);
  draw(43,6,43,16,179);
  pokeb(0xb000,co_xy(43,5)-1,194);
  pokeb(0xb000,co_xy(43,17)-1,207);
  gotoxy(9,16);
  return choice(2,30,7,4,2,6,10);
}
/*      FUNTION CALL OTHER FUNCTION FROM CHOICE            */
/*    CALL - call(choice_number)                           */
call(choice_no)
int loop = 1;
int n;
int line = 5;
int buf_no = 1;
int equal  = 1;
int select;

gotoxy(9,16);
switch(choice_no)
{
  case 1 :
       printf(" Get user name and search \n");
       break;
```

```
A>type b:p5
   case 2 :
        printf(" Get user name and list records \n");
        break;
   case 3 :
        printf(" Get pager number and transmitt \n");
        get_num();
        gotoxy(10,24);
        printf("Transmitt %s",calling.pager);
        break;
   case 4 :
        printf(" Get pager number and list records\n");
        break;
   case 5 :
        printf(" Get user Data\n");
        break;
   case 6 :
        printf(" Get user name and delete\n");
        break;
   case 7 :
        setclock();
        break;
   case 8 :
        loop = 0;
        fdclose(fp);
        printf(" EXIT !");
        break;

}
for(n=1;n<100;n++)
{
  gotoxy(65,16);
  printf("  working ");
}
return loop;
}
/*        FUNCTION SET CLOCK                                    */
/*      CALL - setclock()                                       */
setclock()
{
    int    i ;
     clsr();
     gotoxy(35,7);
     flame(33,4,59,14);
     printf("enter HOUR    (0-24)   ");
     gotoxy(56,7);
     scanf("%x",&i) ;
     outportb(HOUR_PORT,i) ;
     gotoxy(35,8);
     printf("enter MINUTE (0-59)   ") ;
     scanf("%x",&i) ;
     outportb(MIN_PORT,i) ;
     gotoxy(35,9);
     printf("enter SECOND (0-59)   ") ;
```

```
A>type b:p5
     scanf("%x",&i) ;
     outportb(SEC_PORT,i)  ;
     gotoxy(35,10);
     printf("time is already set") ;
}
get_num()
{
  clsr();
  gotoxy(8,3);
  printf("Input user number ");
  f_edit(3,32,20,calling.pager);
}
/*                    PROGRAM SCREEN TOOL                                  */
/* FUNCTION FOR CONVERT COORDINATE(X,Y) TO ADDRESS OF ATTRIBUTE */
/*  CALL - co_xy(cordinate_x,cordinate_y)
co_xy(cx,cy)
{
 int displace,xy;
 displace = 2*(cy*80+cx);
 xy = 0x0001 + displace;
 return(xy);
}
/* FUNCTION  POKE  IF CODE=1   ATTRIBUTE=REVERSE          */
/*                 IF CODE=2   ATTRIBUTE=BLINGING         */
/*                 IF CODE=0   ATTRIBUTE=NORMAL           */
/*      Call - fpoke(row,column,width,code)               */
fpoke(row,column,width,code)
{
int attribute,n;
int reverse = 0x70;
int blinging= 0xf0;
int normal  = 0x07;
int vdo_seg = 0xb000;
int start,stop;

start = co_xy(column,row);
stop  = co_xy(column+width,row);
switch(code)
  {
   case 0 :
        attribute = normal;
        break;
   case 1 :
        attribute = reverse;
        break;
   case 2 :
        attribute = blinging;
        break;
   }
 for (n = start ; n <= stop ; n += 2)
   pokeb(vdo_seg,n,attribute);
}
/*         FUNCTION DRAW TWIN LINE FRAME                  */
```

```
A>type b:p5
/*      CALL - flame(xstart,ystart,xstop,ystop)          */
flame(xstart,ystart,xstop,ystop)
{
int x,y;

pokeb(0xb000,co_xy(xstart,ystart)-1 ,201);
pokeb(0xb000,co_xy(xstop,ystart)-1 ,187);
pokeb(0xb000,co_xy(xstart,ystop)-1 ,200);
pokeb(0xb000,co_xy(xstop,ystop)-1 ,188);
for(x=xstart+1 ; x <= xstop-1 ;x++)
   {
    pokeb(0xb000,co_xy(x,ystart)-1,205);
    pokeb(0xb000,co_xy(x,ystop)-1 ,205);
   }
for(y=ystart+1 ; y <= ystop-1 ;y++)
   {
    pokeb(0xb000,co_xy(xstart,y)-1 ,186);
    pokeb(0xb000,co_xy(xstop,y)-1 ,186);
   }
}
/*       FUNCTION DRAW  A LINE                          */
/*     CALL - draw(xstart,ystart,xstop,ystop,ascii)     */
draw(xstart,ystart,xstop,ystop,ascii)
{
int x,y;

if (xstart == xstop)
    {
    for(y = ystart ; y <= ystop ;y++)
     pokeb(0xb000,co_xy(xstart,y)-1 ,ascii);
    }
 else
    {
     if (ystart == ystop)
      {
       for(x = xstart ; x<= xstop ;x++)
        pokeb(0xb000,co_xy(x,ystart)-1,ascii);
      }
      else
       printf(" Invalid coordinate in function draw ");
    }
}
/*            FUNCTION FIELD EDIT                                */
/*      CALL - f_edit(int row,int column,int width,char i[]) */

#define MAX_STRINGS 80
f_edit(int row,int column,int width,char i[])
{
  int n= 0;
  char ch,ch2;

  width--;
  fpoke(row,column,width,1);
```

```
A>type b:p8
  gotoxy(column,row);
  do
  {
  ch = getch();
   if (ch == 0)
   {
      ch2 = getch ();
      switch (ch2)
      {
        case 'K' : if(n>0)
                   {
                   n--;
                   break;
                   }
        case 'M' : if(n<width)
                   {
                   n++;
                   break;
                   }
        case 'H' : {
                   fpoke(row,column,width,0);
                   return (-1);
                   }
        case 'P' : {
                   fpoke(row,column,width,0);
                   return (1);
                   }
        case 13  : {
                   fpoke(row,column,width);
                   return (1);
                   }
        default  :{
                    i[n]   = 0;
                   putch(ch);
                    i[n+1] = ch2;
                    putch(ch2);
                    n += 2;
                  }
      }
   }
   else
   {
     if (ch == 13)
        {
        for (;n <= width;n++)
           i[n]  =  0;
         }
     if (ch != 13 && ch !=8 )  /* 13 is select code */
        {
          i[n] = ch;
          putch(ch);
          n++;
        }
```

```
A>type b:p8
    }
gotoxy(column+n, row);
} while (n <= width && ch != 13);
if (ch == 13 || n> width)
  {
  fpoke(row,column,width,0);
  return (1);
  }
}
/*       FUNCTION GOTO COORDINATE  X - Y                    */
/*     CALL - gotoxy(x,y)                                   */
gotoxy(x,y)
{
union REGS regs;
int VIDEO = 0x10;

regs.h.ah = 2;
regs.h.dh = y;
regs.h.dl = x;
regs.h.bh = 0;
int86(VIDEO,&regs,&regs);
}
/*       FUNCTION CLEAR SCREEN                              */
/*         CALL - clsr()                                    */
clsr()
{
union REGS regs;
int VIDEO = 0x10;

regs.h.ah = 15;
int86(VIDEO,&regs,&regs);
regs.h.ah = 0;
int86(VIDEO,&regs,&regs);
regs.h.ah = 0x06;
regs.h.al = 0;
regs.h.ch = 0;
regs.h.cl = 0;
regs.h.dh = 0x18;
regs.h.dl = 0x4f;
regs.h.bh = 7;
int86(VIDEO,&regs,&regs);
regs.h.ah = 2;
regs.h.dh = 0;
regs.h.dl = 0;
regs.h.bh = 0;
int86(VIDEO,&regs,&regs);
}
/*-------------------------------------------------------*/
/*       FUNCTION IMPORT DATA FROM 8048                  */
/*     CALL - import8048()                               */
/*-------------------------------------------------------*/

# define RX_STAT  0x301
```

```
A>type b:p8
# define RX_PORT  0x300
# define HOUR_PORT  0x2c4
# define MIN_PORT   0x2c3
# define SEC_PORT   0X2c2

import8048()

{
    int second,n;
    int in_page [100][10];          /* data[frame][byte]*/
    int frame,i,stop,max;
    extern CALL calling;
    extern ttx;

    second = inportb(SEC_PORT);                 /* inport second */
  if (second == 0)
    {
       gotoxy(20,20);
       printf(" IMPORT DATA FROM 8048   PLEASE WAIT!");
      do
        {
          promt = inportb(RX_STAT);
        } while ( promt < 128 );              /* check 7th bit */
       inportb(RX_PORT);                      /* clear */
       frame = 0;
       stop = 0;
       do
         {
          frame ++;
          in_page[frame][1] = inportb(RX_PORT);
         if(in_page[frame][1] != 0xFF)  /* check EOT*/
            {
              i = 1;
              do
               {
                 i++;
                 in_page[frame][i] = inportb(RX_PORT);
               }while (in_page[frame][i] != 0x12);
               i--;
               max = i;
               for(;i <= 10;i++)
                  in_page[frame][i]=0;
               calling[frame].pager = 0;
               for(i = 1;i <= 10;i++)           /*translate to 2by
                  {
                     calling[frame].pager += in_page[fram][i]*pow
                     max--;
                  }
               i = 0;
               do
                {
                  i++;
                  calling[frame].area[i] = inportb(RX_PORT);
```

```
A>type b:pl0
                }while (calling[frame].area[i] != 0x16)
               i = 0;
               do
                {
                  i++;
                  calling[frame].tel[i] = inportb(RX_PORT);
                }while (calling[frame].area[i] != 0x14)
          }
          else
           {
           stop = 1;
           ttx = num-1;
        }
   }
/*-------------------------------------------------------------*/
/*      FUNCTION      RECCORD DATA                             */
recorder()
{
        char ch;
        int n;
        extern int ttx;
        extern CALL calling;
        int handle, status,hour,minute;
        FILE *fp;
        struc date today;

        getdate(&today);
        hour   = inportb(HOUR_PORT);
        minute = inportb(MIN_PORT);
        handle = open("B:CALLING.DAT",O_APPEND);
        fp = fdopen(handle,"a");
      if (fp == NULL)
        printf("fdopen failed\n");
      else
       {
         for (n=1;n<=ttx;n++)
          {
           fprintf(fp,"%d%-4s%-8s",calling[n].pager
           ,calling[n].area,calling[n].tel);
           fprintf(fp,"%d%d%d%d%d",minute,hour
           ,today.da_mon,today.da_day,today.da_year);
          }
         fclose(fp);
      }
 }
 /*-------------------------------------------------------------
 /*    FUNCTION TRANSMITT DATA TO CALL PAGER
 /*     call transmitt ()
 /*-------------------------------------------------------------
 transmitt()
     {
     extern CALL calling;
     extern int ttx;
```

```
A>type b:pl0
    extern area_len,tel_len;
    int  i, data[10],code_len,n,m;
    int  high,low;

    for(i = 1;i <= ttx ;i++).
     {
      data[1] = calling[i].pager/256;
      data[2] = calling[i],pager%256;
      data[3] = area_len * 16 + tel_len;
      strcat(calling[i].area,calling[i].tel);
      code_len = area_len + tel_len;
     if(code_len % 2 == 1)
       {
        calling[i].area[code_len] = 48 ;
        code_len++;
       }
     m = 4;
     for (n = 0;n <= code_len-1;n+=2)
       {
        high = calling[i].area[n];
        low  = calling[i].area[n+1];
        data[m] = 16*(high - 48) + (low - 48);
        m++;
       }
     for (i = 1; i <= 6; i++)
       out_serial(0xFF);
     for (n = 1;n <= code_len/2 + 4;n++)
       out_serial(data[n]);
     }
 }
   out_serial(data)
    {
     int status,i;
     do
        {
          status = inportb(TX_STAT);
        } while (status < 128);
     for (i = 1;i <= 3;i++)
      outportb(TX_PORT,data);  /* out data */
      }
```

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จลุล่วงไปด้วยดี ด้วยความช่วยเหลือและสนับสนุน เงินทุนวิจัยจากภาควิชาอิเลคทรอนิคส์ โดยมี รศ.ดร.สิทธิชัย โภไคยอุดม อาจารย์ประจำภาควิชาอิเลคทรอนิคส์เป็นผู้ดูแล และให้คำแนะนำปรึกษาเกี่ยวกับการทำโครงงานนี้มาโดยตลอด

เอกสารอ้างอิง

1.David F.Stout / Milton Kaufman,editor, " Handbook of operational amplifier circuit design ", Mc Graw-Hill Book company, 415 p.,1976

2.Gobind Daryanani, " Principle of active network synthesis and design " , John Wiley & Sons, 495 p.,1976

3.Kennedy, " Electronic communication system " ,Mc Graw-Hill Book company , 741 p.,1984

4.Lewis C. Eggebrecht, " Interfacing to the IBM Personal Computer " , Howard W.Sams & Co.,Ltd , 246p.,1983

5.Howard M. Berlin, " Design of phase lock loop circuits with experiments ", Howard W.Sams & Co.,Ltd , 254p.,1981

6.Brian W. Kernigham / Dennis M. Ritchie, " The C programming language ",Prentice-Hall,inc. ,228p.

7." TURBO C refference guide and user guide " ,Borland International ,inc. ,298p.

8.MOTOROLA TTL LOW POWER SCHOTTKY DATA BOOK, Motorola Semiconductor product inc.,374p.

9.บุญเลิศ เอี่ยมทัศนา ,ยืน ภู่วรวรรณ ,สมนึก ดีรีโต ," โปรแกรมคอมพิวเตอร์ ภาษาซี " ,บริษัท ซีเอ็ดยูเคชั่น จำกัด ,299p., 2529