การสื่อสารข้อมูลด้วยระบบแพกเกตเรดิโอ

PACKET RADIO DATA COMMUNICATION

โดย

นายบัญชา เจริญมาก

นายสุรพงศ์ ธาดาสีห์

นางสาวเอมอร สืบสนิท

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การสื่อสารข้อมูลด้วยระบบแพกเกตเรดิโอ

PACKET RADIO DATA COMMUNICATION

โดย

นายบัญชา เจริญมาก 38013018

นายสุรพงศ์ ธาดาสีห์ 38013036

นางสาวเอมอร สืบสนิท 38013045

อาจารย์ที่ปรึกษา

ดร. สุทธิชัย นพนาคีพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมโทรคมนาคม

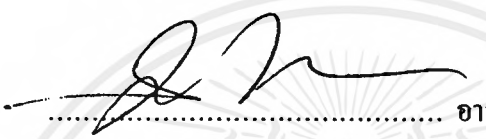
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสื่อสารข้อมูลด้วยระบบแพกเกตเรดิโอ

PACKET RADIO DATA COMMUNICATION

ผู้จัดทำ

1. นายบัญชา เจริญมาก 38013018
2. นายสุรพงศ์ ธาคาสีห์ 38013036
3. นางสาวเอมอร สืบสนิท 38013045



..... อาจารย์ที่ปรึกษา

(ดร. สุทธิชัย นพนาคีพงษ์)

การสื่อสารข้อมูลด้วยระบบแพกเกตเรดิโอ
PACKET RADIO DATA COMMUNICATION

โดย นายบัญชา เจริญมาก 38013018
นายสุรพงศ์ ธาดาสีห์ 38013036
นางสาวเอมอร สืบสนิท 38013045

อาจารย์ที่ปรึกษา ดร. สุทธิชัย นพนาคีพงษ์

บทคัดย่อ

โครงงานนี้เป็นการศึกษา การสร้างเครื่องแพกเกตเรดิโอ ระบบแพกเกตเรดิโอเป็นระบบการสื่อสารที่นิยมใช้ในกิจการวิทยุสมัครเล่น และกำลังมีบทบาทสำคัญในระบบสื่อสารโทรคมนาคมในปัจจุบัน ต้นแบบของโครงงานนี้เป็นการสร้างเทอร์มินอล ซึ่งมีโมเด็มอยู่ภายในตัวเอง โมเด็มใช้คลื่นวิทยุความถี่ 144-148 เมกะเฮิรตซ์ ระบบ ไบนารีเอฟเอสเค และใช้เฟิร์มแวร์ TNC-2 ของ TAPR

ABSTRACT

This project is a study of packet radio system. The packet radios are normally used in armature radio and important in the telecommunication field. This project is a phototype of packet radio terminal including with RF modem, 144-148 MHz binary FSK modulation and TNC-2 firmware from TAPR.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

การใช้ประโยชน์จากเทคโนโลยีของวิทยุสมัครเล่นเพื่อการศึกษาด้านวิทยาศาสตร์และเทคโนโลยี โดยเฉพาะในสาขาวิศวกรรมโทรคมนาคมเป็นสิ่งที่น่าสนใจ ปัจจุบันเทคโนโลยีวิทยุสมัครเล่นมีการพัฒนาไปมากจากอดีตซึ่งเป็นการติดต่อสื่อสารโดยใช้ระบบวิทยุโทรเลข จนในปัจจุบันสามารถทำการสื่อสารกันด้วยระบบดิจิตัลผ่านดาวเทียม

จะเห็นได้ว่าได้มีสถาบันการศึกษาชั้นนำของโลกหลายแห่งได้ทำการส่งดาวเทียมวิทยุสมัครเล่นเพื่อใช้ในการเรียนการสอน ขณะเดียวกันนักวิทยุสมัครเล่นและนักวิจัยทั่วโลกยังสามารถใช้ประโยชน์จากดาวเทียมเหล่านั้นได้อีกด้วย

ระบบแพกเกจเรดิโอนับว่ามีบทบาทสำคัญในการสื่อสารข้อมูลดิจิตอลในกิจการวิทยุสมัครเล่น แม้กระทั่งดาวเทียมวิทยุสมัครเล่นในปัจจุบันได้มีการบรรจุระบบแพกเกจเรดิโอให้เป็นส่วนหนึ่งของตัวดาวเทียม ด้วยจุดเด่นที่เป็นระบบที่มีความยืดหยุ่นสูง สามารถใช้งานได้หลายรูปแบบและเป็นระบบเปิดที่สามารถเรียนรู้ทำความความเข้าใจได้ง่าย

ระบบแพกเกจเรดิโอที่ใช้ในกิจการวิทยุสมัครเล่นพัฒนามาจากระบบอโลฮา มีการกำหนดโปรโตคอลที่ใช้ในการเชื่อมต่อเรียกว่าโปรโตคอล AX.25 แม้ว่าในปัจจุบันได้มีการพัฒนาระบบแพกแกจเรดิโอไปในหลายรูปแบบเช่น ROSE, NOS, NET หรือ TCP/IP แต่ก็ยังคงใช้โครงสร้างแบบ AX.25

โครงงานนี้เป็นการนำเสนอการสร้างเครื่องเทอร์มินอลที่ใช้สำหรับเชื่อมต่อเครื่องไมโครคอมพิวเตอร์กับระบบแพกเกตเรดิโอ โดยพัฒนามาจากวงจรเทอร์มินอลโหนดคอนโทรลเลอร์รุ่น TNC-2 ของสมาคม TAPR (Tucsan Amateur Packet Radio) และใช้เฟิร์มแวร์แพกเกจเรดิโอคอนโทรลเลอร์ของ TAPR เวอร์ชั่น 2.18a

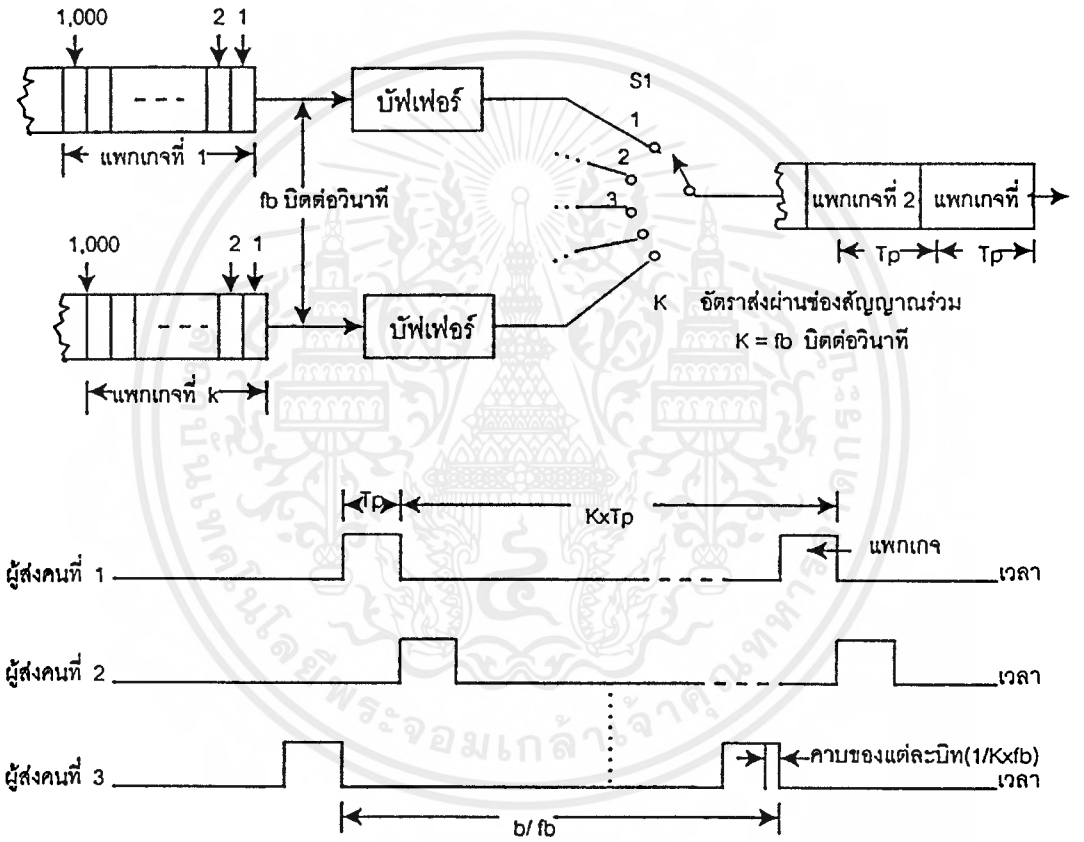
เพื่อให้สามารถใช้เฟิร์มแวร์ของ TAPR ได้ ดังนั้นโครงงานนี้จึงใช้ชิปไมโครคอนโทรลเลอร์และการถอดรหัสพอร์ทเช่นเดียวกับเครื่อง TNC-2 โดยส่วนที่ได้ทำการพัฒนาขึ้นคือ วงจรโมเด็มอินเตอร์เฟส และวงจรอาร์เอฟโมเด็มแบบไบนารีเอฟเอสเคซึ่งทำงานในย่านความถี่วิทยุ 144 -148 เมกะเฮิรตซ์

บทที่ 2
ทฤษฎีพื้นฐาน

2.1 หลักการของแพกเกตเรดิโอ

แพกเกตเรดิโอเป็นการส่งข้อความสื่อสารระหว่างจุดต่อจุด โดยอาศัยคลื่นวิทยุ ตั้งแต่ย่านความถี่ HF (1-30 เมกะเฮิรตซ์) จนถึงย่านความถี่ EHF (3-10 กิกะเฮิรตซ์) สำหรับรูปแบบของการส่งข้อมูลมีอยู่หลายวิธี ที่นิยมกันมากได้แก่ แบบแบ่งเวลา (Time Division Multiple Access), แบบแบ่งความถี่ (Frequency Division Multiple Access), แบบอโลฮา (ALOHA) และยังมีแบบอื่น ๆ อีกมากมาย

2.1.1 การส่งแบบแบ่งเวลา



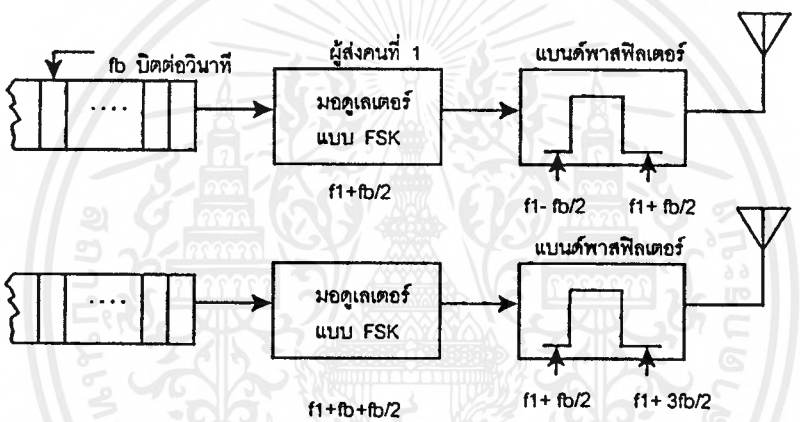
รูปที่ 2.1 แสดงการส่งแพกเกตแบบแบ่งเวลา

จากรูปที่ 2.1 เป็นวิธีการส่งแพกเกตแบบแบ่งเวลา สำหรับวิธีการนี้เป็นลักษณะที่ผู้ใช้งานแต่ละคนผลัดกันส่งข้อมูล สมมุติว่ามีผู้ทำการส่งข้อมูลออกอากาศในความถี่เดียวกันอยู่ K คน แต่ละคนถูกกำหนดให้ส่งข้อมูลออกมาเป็นแพกเกต ๆ ละ 1,000 บิต (ซึ่งเป็นขนาดมาตรฐานที่ใช้กันทั่วไป) ข้อมูลที่ส่งออกมาจะถูกเก็บไว้ในบัฟเฟอร์ของแต่ละช่องข้อมูล จนกว่าสวิทช์ตัดต่อ S ซึ่งวิ่งวนจากช่องข้อมูลที่ I ถึงช่องที่ K เคลื่อนมาถึงช่องข้อมูลดังกล่าว ข้อมูลที่เก็บไว้ในบัฟเฟอร์จะถูกส่งผ่านออกไปยังช่องสัญญาณร่วม และบัฟเฟอร์จะถูกทำให้ว่างเพื่อรอรับแพกเกตใหม่ต่อไป หากผู้ใช้งานแต่ละคนส่งข้อมูล

ออกด้วยอัตรา fb บิตต่อวินาทีเท่า ๆ กัน จะเห็นได้ว่าอัตราส่งผ่านข้อมูลผ่านช่องสัญญาณร่วมจะมีค่าเป็นผลคูณของ K×fb บิตต่อวินาที ซึ่งหากพิจารณาถึงข้อมูลที่วิ่งผ่านช่องสัญญาณร่วม จะเห็นว่ามีการส่งแพกเกตของช่องสัญญาณที่ 1 ตามด้วยช่องสัญญาณที่ 2 ไปจนถึงช่องสัญญาณที่ K แล้ววกกลับมาเป็นช่องสัญญาณที่ 1 อีก เช่นนี้เรื่อยไป

ถ้ากำหนดว่าข้อมูลแต่ละแพกเกตมีขนาด b บิต ความกว้างของแต่ละแพกเกตที่ผ่านช่องสัญญาณร่วมจะเป็น b/(K×fb) ดังนั้นสวิทช์ตัดต่อ S จะต้องหมุนด้วยคาบ b/fb เพื่อที่จะได้รับข้อมูลที่ออกจากผู้ส่งแต่ละช่องได้ทันเวลา ซึ่งการส่งแบบแบ่งเวลานี้จะมีประสิทธิภาพสูงสุดก็ต่อเมื่อผู้ส่งข้อมูลแต่ละคนส่งข้อมูลอยู่ตลอดเวลา หากมีบางช่องข้อมูลไม่ทำการส่งแล้ว ผู้ส่งในช่องอื่นๆ ที่มีข้อมูลอยู่จะต้องเสียเวลารอจนกว่าสวิตช์ตัดต่อเคลื่อนที่มาถึงช่องของตน ยิ่งอัตราการรอสูงขึ้นมากเท่าใด ประสิทธิภาพของการส่งแบบนี้ก็ยิ่งลดลงมากขึ้นเท่านั้น

2.1.2 การส่งแบบแบ่งความถี่



รูปที่ 2.2 แสดงการส่งแพกเกตแบบแบ่งความถี่

การส่งแบบแบ่งความถี่นี้ต่างกับแบบแบ่งเวลา ตรงที่ผู้ส่งแบบแบ่งเวลาจะผลัดกันส่งข้อมูล แต่แบบแบ่งความถี่ อนุญาตให้แต่ละคนทำการส่งข้อมูลได้ตลอดเวลา โดยข้อมูลของแต่ละคนจะถูกส่งไปคนละความถี่ซึ่งมีแบนด์วิดธ์เท่ากับ Bu ภายในขอบเขตแบนด์วิดธ์รวมเท่ากับ B ซึ่งแท้จริงแล้วอัตราการส่งข้อมูลแบบแบ่งความถี่ จะมีค่าเท่ากับแบบแบ่งเวลาคือ K×fb บิตต่อวินาที

จากรูปที่ 2.2 ได้แสดงให้เห็นถึงรูปแบบการส่งแพกเกตแบบแบ่งความถี่ ข้อมูลจากแต่ละช่องจะถูกส่งไปยังมอดูเลเตอร์แบบ FSK (Frequency Shift Keying) โดยความถี่พาหะของแต่ละช่องสัญญาณจะถูกกำหนดให้ห่างกันเท่ากับ fb และแต่ละช่องข้อมูลใช้แบนด์วิดธ์เท่ากับ fb แล้วจึงทำการส่งออกอากาศ ที่เครื่องรับแต่ละช่องถูกปรับให้รับคลื่นความถี่ที่ตรงกันกับช่องสัญญาร และมีแบนด์วิดธ์ fb เช่นเดียวกัน ข้อดีของการส่งแบบแบ่งความถี่ ก็คือการที่ผู้ส่งไม่ต้องเสียเวลารอคอยช่องสัญญาณในการส่ง แต่ละช่องสามารถส่งข้อมูลได้ตามอิสระ อีกทั้งยังเป็นการกำจัดบัฟเฟอร์ออกไป (บัฟเฟอร์เป็นหน่วยความจำ ซึ่งมีราคาสูง) ทำให้ลดต้นทุนลงได้ ส่วนข้อเสียคือ การที่แบนด์วิดธ์ของระบบโดยทั่วไปมีไม่กว้างมากนัก ทำให้จำนวนผู้ใช้งานมีได้จำกัด และในทางปฏิบัติพบว่าการส่งแบบแบ่งเวลา เป็นแบบที่

นิยมมากกว่าแบบแบ่งความถี่ เนื่องจากผู้ผลิตอุปกรณ์ส่วนใหญ่ให้การสนับสนุนพัฒนาประสิทธิภาพของการส่งแบบแบ่งเวลาให้ดีขึ้นอยู่ตลอดเวลา

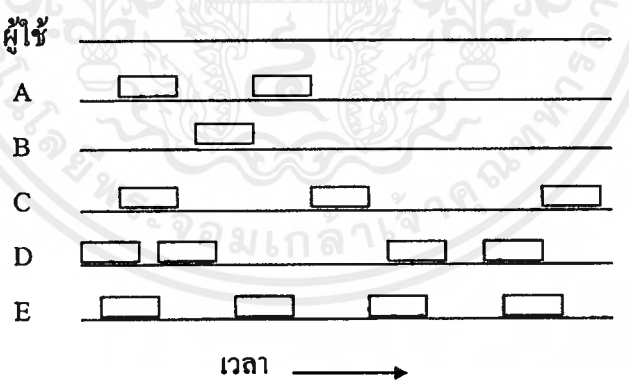
2.1.3 การส่งแบบอโลฮา

จากข้อจำกัดของการส่งแบบแบ่งเวลา ในเรื่องของการรอช่องสัญญาณส่งข้อมูล และรูปแบบของการส่งแบบแบ่งความถี่ ในเรื่องแบนด์วิดธ์ที่มีจำกัด ศาสตราจารย์อับราสันจากมหาวิทยาลัยฮาวายได้พัฒนาการส่งข้อมูลซึ่งใช้การเข้าถึงช่องสัญญาณแบบสุ่มขึ้นโดยตั้งชื่อว่าการส่งแบบอโลฮา ซึ่งเป็นแบบที่นิยมกันมากในปัจจุบันอีกวิธีหนึ่งจะได้อธิบายถึงรายละเอียดในเรื่องของการส่งแบบอโลฮาในหัวข้อต่อไป

2.2 หลักการพื้นฐานของระบบอโลฮา

2.2.1 ระบบช่องสัญญาณอโลฮา

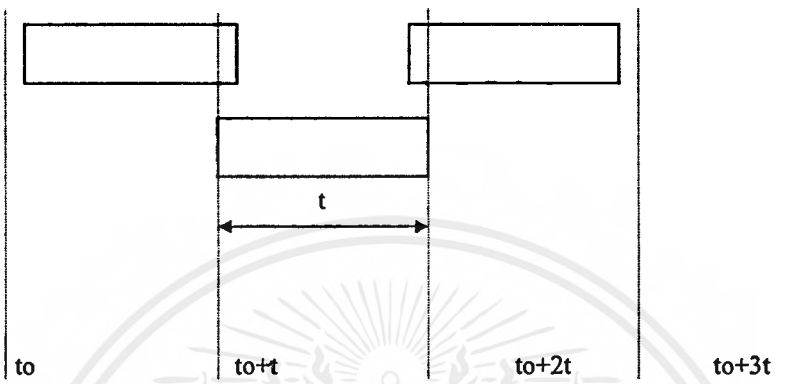
แนวคิดพื้นฐานของระบบอโลฮา เกิดจากการตั้งสมมุติฐานว่าโดยธรรมชาติแล้ว ผู้ใช้งานช่องสัญญาณมักจะไม่ส่งข้อมูลตลอดเวลา และทุกคนก็อาจจะไม่ส่งข้อมูลพร้อมกันก็ได้ ผู้ใช้งานช่องสัญญาณจะใช้งานเมื่อมีข้อมูลที่จะส่งเท่านั้น ปัญหาที่เกิดขึ้นคือในกรณีที่มีการส่งข้อมูลจากผู้ใช้งานหลาย ๆ ช่องพร้อมกัน กรณีนี้เครื่องรับจะรับแพกเกตหลายแพกเกตในเวลาเดียวกัน โดยไม่สามารถแยกแพกเกตเหล่านี้ออกจากกันได้เรียกว่า เกิด "การชน" ซึ่งความน่าจะเป็นที่ข้อมูลจะเกิดการชนกันขึ้นอยู่กับพฤติกรรมการส่งข้อมูลของบรรดาผู้ใช้ช่องสัญญาณนั่นเอง เมื่อการชนกันของข้อมูลเกิดขึ้น ข้อมูลเหล่านี้จะกลายเป็นขยะและจะถูกลบออกจากช่องสัญญาณร่วม และเครื่องรับจะส่งสัญญาณให้ผู้ส่งทำการส่งข้อมูลใหม่ จนกว่าจะไม่เกิดการชนอีก การขอให้ส่งใหม่จะเกิดขึ้นเพียงไม่กี่ครั้งเท่านั้น เพราะความน่าจะเป็นของการส่งข้อมูลออกมาพร้อมกันในครั้งหลังจะต่ำกว่าในครั้งแรก ๆ



รูปที่ 2.3 ระบบการส่งแพกเกตแบบอโลฮา เมื่อเวลาใด ๆ

จากรูปที่ 2.3 แต่ละแพกเกตจะมีขนาดเท่ากัน ถ้าเวลาใดมีแพกเกตที่จะส่ง 2 แพกเกต แพกเกตนั้น ก็จะถูกทำลายและจะถูกทำการส่งใหม่ เมื่อผู้ใช้เตรียมข้อมูลที่จะส่ง สมมุติว่าต้องรอ R วินาที ถึงจะทำสารส่งเสร็จเรียบร้อย แต่ถ้ายังไม่เสร็จก็จะต้องรอส่งไปเรื่อย ๆ ให้แพกเกตไทม์ แทนเวลาทั้งหมดที่จะส่งเสร็จ มีค่าเท่ากับความยาวแพกเกตหารด้วยบิทเรต โดยให้มีความยาวแพกเกตที่แน่นอน ถ้าแพกเกตที่จะส่งมีจำนวนที่จะส่งมากๆ แล้วจะต้องทำการแยกโดย S แทนจำนวนแพกเกตทั้งหมดหารด้วยแพกเกตไทม์ ถ้า S มีค่ามากกว่า 1 แสดงถึงว่าผู้ใช้ส่งตอนนี้จะต้องเกิดการชนกันของแพกเกตแน่ ๆ ดังนั้นจะต้องมีค่า S อยู่ระหว่าง 0 ถึง 1 ดังนั้นแล้วแพกเกตใหม่ที่จะทำการส่งไป ถ้าเกิดส่งไปแล้วเสีย

หายโดยการชนกันก็ต้องส่งใหม่ สมมุติความน่าจะเป็น k ของความพยายามในการส่งต่อแพกเกตไทม์ คือ G ต่อแพกเกตไทม์ ถ้า G มีค่ามากกว่า S มาก ๆ ในสภาวะที่มีข้อมูลน้อย นั่นคือ S มีค่าประมาณศูนย์ การชนกันก็จะเกิดขึ้นน้อยมาก และการส่งใหม่จะยิ่งน้อยไปอีก เมื่อ G มีค่าประมาณเท่ากับ S แต่เมื่อมีข้อมูลที่จะส่งมากมายการชนกันก็จะมากขึ้น ในสภาวะนี้คือ G มากกว่า S ดังนั้นความน่าจะเป็นของการส่งไปสำเร็จคือ S = GPo เมื่อ Po คือความน่าจะเป็นซึ่งแพกเกตนั้นจะไม่เสียหายจากการชนกัน



รูปที่ 2.4 แสดงความเสียหายเนื่องจากการชนกัน

แพกเกตจะไม่เสียหายเนื่องจากการชนกัน ถ้ามีการส่งแพกเกต 2 แพกเกตซ้อนกันกับบิตแรกของแพกเกตก่อนหน้านั้นแสดงให้เห็นได้ดังรูป 2.4 จากรูปแพกเกตอันล่างจะไม่เสียหาย ถ้าผู้ใช้ให้แพกเกตอยู่ระหว่าง to และ to+t ของแพกเกตที่ถูกทับหรือโอเวอร์แลป(overlap) ที่บิตแรก และให้แพกเกตที่ถูกโอเวอร์แลปที่บิตสุดท้ายเลื่อนไปอยู่ที่ to+2t และ to+3t

ความน่าจะเป็นของแพกเกต k จะถูกให้มีอยู่ในแพกเกตไทม์ โดยแยกออกเนื่องจากการเสียหาย

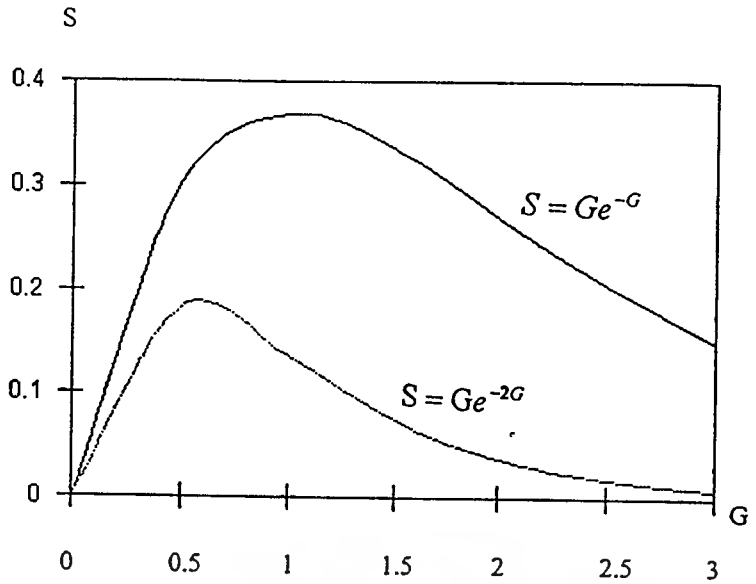
$$\Pr[k] = \frac{G^k e^{-G}}{k!} \qquad \text{..... 1}$$

ถ้าความน่าจะเป็นของแพกเกตนั้นมี 0 แพกเกตให้ใช้ e^{-G}

ถ้าเวลาผ่านไป 2 แพกเกตไทม์ แพกเกตที่ส่งได้ในเวลาแพกเกตไทม์คือ 2G ความน่าจะเป็นที่จะมีการเสียหายจากการซ้ำกัน $P_o = e^{-2G}$ ถ้าใช้ S = GPo ดังนั้นจะได้ว่า $S = Ge^{-2G}$ การส่งข้อมูลไปได้แค่ไหนกับค่า ความพยายามในการส่งแสดงให้เห็นในรูป 2.4 การส่งได้สูงสุดเกิดขึ้นเมื่อ G = 0.5 จะได้ S=1/2e หรือมีค่าประมาณ 0.184 แสดงว่ามีความสามารถในการส่งประมาณ 18% แสดงว่ายังไม่ดีเท่าไร ถ้าจะให้ดีมาก ๆ ต้องมีค่า 100%

ในปี 1972 โรเบิร์ตได้ประกาศวิธีบรรจุของระบบอโลฮา ถูกเรียกว่าช่องสัญญาณอโลฮา (Slot ALOHA) แต่ละช่องจะมีความน่าจะเป็นของการชนกันคือ

$$S = Ge^{-G} \qquad \text{..... 2}$$



รูปที่ 2.5 แสดงอัตราการส่งข้อมูลจริงของระบบอโลฮา

ในรูปที่ 2.5 จะเห็นได้ว่าจากเส้นสมการ $S = Ge^{-G}$ นั้นทำให้มี G = 1 และการส่งข้อมูลได้มากสุดคือ S = 1/e หรือ 0.368 หรือคิดเป็นเปอร์เซ็นต์ เท่ากับ 37%

2.2.2 จำนวนผู้ใช้ช่องสัญญาณอโลฮาที่มีจำกัด

ถ้าให้ Si แทนความน่าจะเป็นของการส่งได้สำเร็จของผู้ใช้ i และ ให้ Gi เป็นผลรวมความน่าจะเป็นของการส่ง (ต่อช่องสัญญาณ) ของผู้ใช้ i รวมทั้งแพกเกตใหม่และแพกเกตที่ถูกส่งไปแล้วและต้องส่งใหม่เพราะการชนกัน ดังนั้นถ้า Si มีค่าน้อยกว่า Gi มาก ๆ แล้ว ความน่าจะเป็นที่ช่องสัญญาณจะส่งได้สำเร็จโดยผู้ใช้ i คือ ความน่าจะเป็นที่ผู้ใช้ i ส่งแพกเกต

$$S = Gi\prod_{j\neq i}(1-G_j) \qquad \text{..... 3}$$

จากสมการที่ 3 ชี้ให้เห็นว่าจำนวนผู้ใช้ N คน สามารถส่งได้ Si =S/N แพกเกต/ช่องสัญญาณ และผลรวมของอัตราการส่งทั้งหมด Gi = G/N แพกเกต/ช่องสัญญาณ ซึ่ง $G = \sum Gi$ แทนลงในสมการ 3 จะได้ว่า $S = G[1-\frac{G}{N}]^{N-1}$ ถ้า $N \to \infty$ จะทำให้ได้สมการใหม่ดังนี้ $lim_{x\to\infty}[1+\frac{x}{k}]^x = e^x$ ก็จะเหมือนกับสมการที่เป็นของผู้ใช้จำนวนไม่จำกัดไป จากรูป 2.4 จะเห็นว่าการส่งข้อมูลได้สูงที่สุดสำหรับจำนวนไม่จำกัดของช่องสัญญาณระบบอโลฮา เกิดขึ้นที่ G=1 แอบแรมสันได้แสดงไว้ว่าสำหรับการส่งข้อมูลได้สูงสุดคือ

$$\sum_{i=1}^{\infty} Gi = 1 \qquad \text{..... 4}$$

ถ้าให้ N_1 เป็นผู้ใช้ช่องสัญญาณที่ต้องการส่งข้อมูลครั้งแรก และ N_2 เป็นผู้ใช้ช่องสัญญาณที่ต้องการส่งข้อมูลครั้งที่ 2 จะได้ว่า

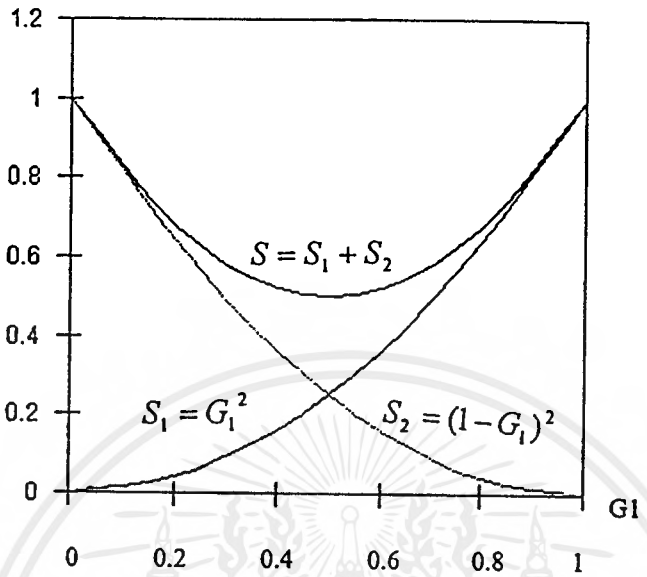
$$S_1 = G_1(1-G_1)^{N_1-1}(1-G_2)^{N_2} \qquad \text{..... 5}$$

$$S_2 = G_2(1-G_2)^{N_2-1}(1-G_1)^{N_1} \qquad \text{..... 6}$$

การส่งข้อมูลสูงสุดจากสมการ 6 จะได้ว่า

$$N_1G_1 + N_2G_2 = 1 \qquad \text{..... 7}$$

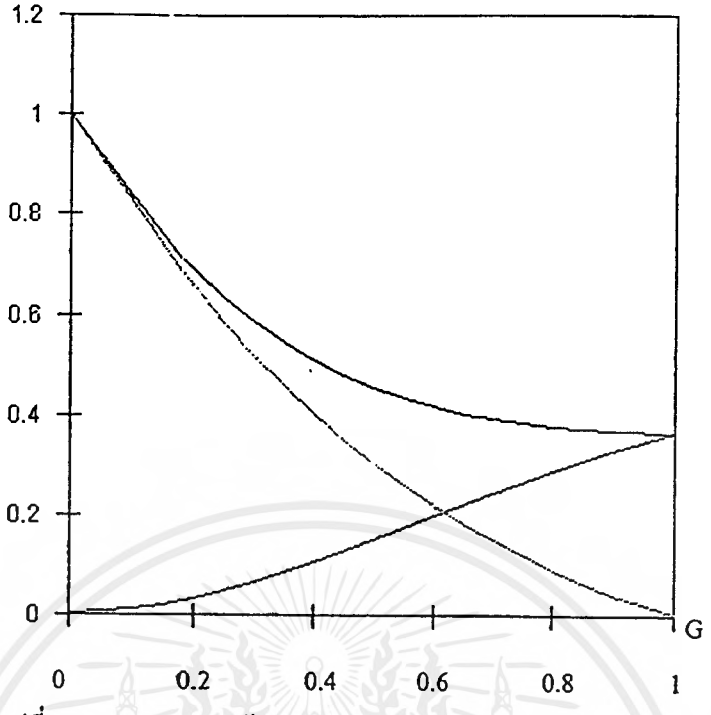
จากสามสมการมีตัวแปร 4 ตัวคือ S_1, S_2, G_1, G_2 และสมการ 8 ถ้าแทน G_2 จากสมการ 5 และ 6 จะได้สมการการส่งถ่ายข้อมูลที่มีแต่ G_1 ตัวอย่างเช่น $N_1 = 1$ และ $N_2 = 1$ จะได้ $S_1 = G_1^{\,2}$ และ $S_2 = (1 - G_1)^2$ และ $S = S_1 + S_2$ สามารถแสดงให้เห็นได้ในรูปที่ 2.6



รูปที่ 2.6 อัตราการส่งข้อมูลจริงของผู้ใช้ 2 คนในช่องสัญญาณระบบอโลฮา

จากรูปที่ 2.6 เมื่อ G1 เข้าใกล้ศูนย์มากที่สุด ผู้ใช้ช่อง 1 พยายามที่จะส่งมาก แต่ผู้ใช้ช่อง 2 จะสามารถส่งได้เกือบทุกช่องสัญญาณ แต่จะเกิดปัญหาขึ้นเมื่อผู้ใช้ช่อง 1 กับผู้ใช้ช่อง 2 ต้องการที่จะส่งข้อมูลพร้อมกัน ความน่าจะเป็นที่แต่ละคนจะส่งได้ก็คือ 0.5 และโอกาสที่จะสามารถส่งได้สำเร็จคือ 25% จากสมการที่ 5 และ 6 กรณีนี้ $N_1 = 1$ และ $N_2 = \infty$ ผู้ใช้ช่อง 1 กำลังพยายามส่งไฟล์ที่ใหญ่มาก G2 มีค่าเข้าสู่ศูนย์ จะเห็นได้ว่า $G = N_2 G_2$ และ $S = N_2 S_2$ จะทำให้ได้ $S_1 = G_1 e^{-G}$ และ $S = G e^{-G}(1 - G_1)$ เงื่อนไขนี้การส่งข้อมูลสูงสุดจะอยู่ที่ $G_1 + G = 1$ สามารถพล็อต S_1, S และการส่งข้อมูล $S_1 + S$ เป็นฟังก์ชันของ G ดังแสดงให้เห็นในรูปที่ 2.7

อัตราการส่งข้อมูลจริงต่อช่องสัญญาณ



รูปที่ 2.7 อัตราการส่งข้อมูลจริงของระบบขนาดใหญ่มีผู้ใช้หลายคน

จากรูป 2.7 จะเห็นได้ว่าหากผู้ใช้ช่อง 1 มีความต้องการที่จะส่งข้อมูลจำนวนมาก ๆ เขาก็สามารถส่งได้โดยไม่ต้องกลัวว่าจะเกิดการชนกัน ถึงแม้ว่าจะมีผู้ส่งหลายคนก็ตาม

2.2.3 การหน่วงและอัตราการส่งจริงของช่องสัญญาณอโลฮา

กระบวนการในการทำงานนี้จะมีความสำคัญในเรื่องของการหน่วงมาก ในระบบอโลฮานี้ การหน่วงมาจากการชนกัน ซึ่งต้องทำการส่งใหม่ซ้ำไปเรื่อย ๆ ทำให้ค่า R ซึ่งเป็นเวลาในการส่งมีค่าเพิ่มมากขึ้นเรื่อย ๆ สมมุติว่าการส่งปกติสามารถทำได้ R วินาที แต่เมื่อเกิดการชนกันและต้องส่งใหม่ เวลาทั้งสิ้นอาจเพิ่มเป็นอย่างน้อย 4R วินาที ขึ้นอยู่กับว่ามีการชนกันมากน้อยแค่ไหน

ในการแบ่งแยกเวลาหน่วงนี้จะใช้รูปแบบดังนี้ สมมุติมีการแบ่งแยกโดยผู้ใช้ N ที่ต้องทำแพกเกตใหม่เนื่องจากเกิดการเสียหายจากการชนกัน ดังนั้นแสดงว่า p มีค่าน้อยกว่า 1 มาก ๆ แพกเกตต่อช่องสัญญาณที่ไม่สามารถทำบล็อกได้ การส่งใหม่จะไม่นับ p ถ้าขณะนั้นมีจำนวนเวลาของการคิดคือ T1 และจำนวนช่องสัญญาณ T2 ดังนั้น p=T2/T1 เมื่อผู้ใช้ทำบล็อกแล้ว จะมีบัฟเฟอร์ ตัวหนึ่งไว้รับคำสั่งเพื่อทำแพกเกต นั้นคือจะต้องมีตัวบอกว่ามีจำนวนแพกเกตที่พร้อมแล้วจึงทำการส่ง

ตัวแปรที่ทำหน้าที่ในการตรวจสอบการชนกันจะต้องมีการรันไปเรื่อย ๆ นั่นคือจะมีตัวหนึ่ง เป็นตัวบรรจุข้อมูลถ้าเมื่อมีน้อยก็แสดงว่าเกิดการชนกันมากขึ้น ดังนั้นจะต้องมีช่องสัญญาณ L ที่คอยทำการส่งข้อมูลใหม่แทนความน่าจะเป็นของแพกเกตที่ชนกันด้วยแบ็กล็อค และความน่าจะเป็นของช่องสัญญาณในการต้องส่งใหม่คือ α ความน่าจะเป็นของจำนวนการหน่วงสล็อต k โดย $\alpha(1-\alpha)^{k-1}$ จะได้ผลรวมการหน่วงก่อนการส่งใหม่คือ $\sum_{k=1}^{\infty}\alpha k(1-\alpha)^{k-1}=\frac{1}{\alpha}$ วิธีการแก้ปัญหาคือใช้วิธีสมมุติการหน่วงระหว่างความพยายามในการส่งใหม่โดยจะได้

$$\frac{1}{\alpha} = R + \frac{L+1}{2} \quad \text{..... 8}$$

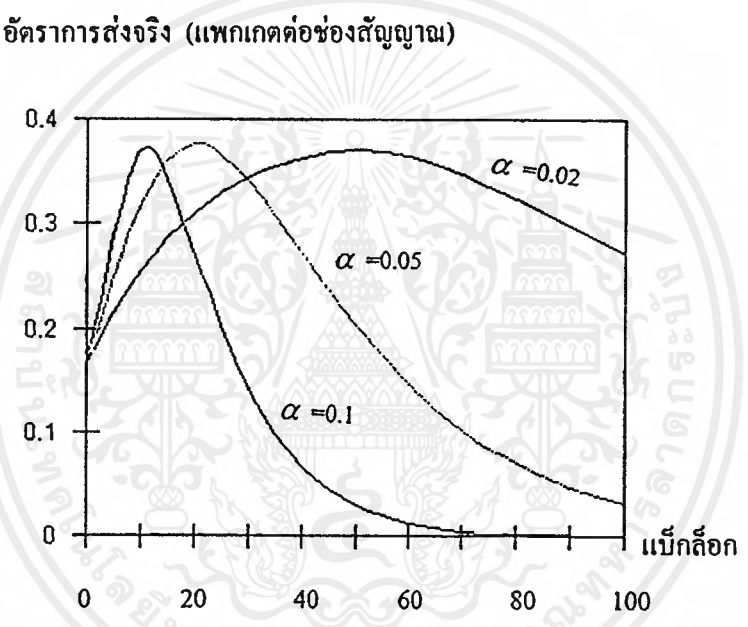
จะสามารถบอกได้ว่ามีกี่สถานีที่จะทำบล็อกได้บ้าง โดยมีจำนวนแพกเกต k ที่เป็นแบ็กล็อคอยู่ แพกเกตที่เพิ่มขึ้นใหม่ถ้าทำงานยังไม่เสร็จก็จะต้องกลับมาเพิ่มจำนวนแบ็กล็อกขึ้นอีก และก็จะถูกส่งเป็นตัวสุดท้าย

2.2.4 เสถียรภาพของช่องสัญญาณอโลฮา

ระบบอโลฮานั้น บางระบบก็เสถียรภาพบางระบบก็ไม่เสถียรภาพ ขึ้นอยู่กับจำนวนข้อมูลที่ต้องการจะส่งมีมากน้อยเพียงใด ถ้ามี N ช่องสัญญาณที่ส่งแล้วเป็นแบ็กล็อกจะได้ว่า

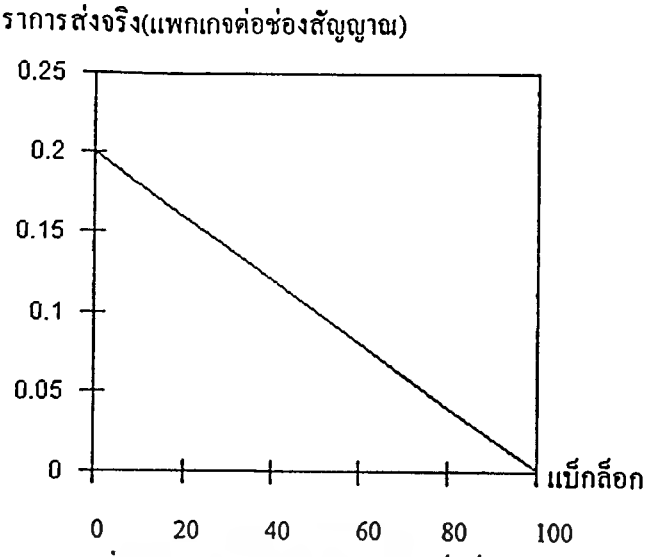
$$\text{อัตราการส่งข้อมูลจริง} = N\alpha(1-\alpha)^{N-1} \quad \text{..... 9}$$

จากสมการ 9 ค่าของ α จะมีค่าน้อยมาก และทำให้ตัดทิ้งไปได้ และถ้ามีการต้องส่งใหม่ช่วงเวลาที่รอจะมีค่าน้อยมาก ยิ่งค่า α ยิ่งน้อยเท่าไร ประสิทธิภาพก็จะดีเท่านั้น



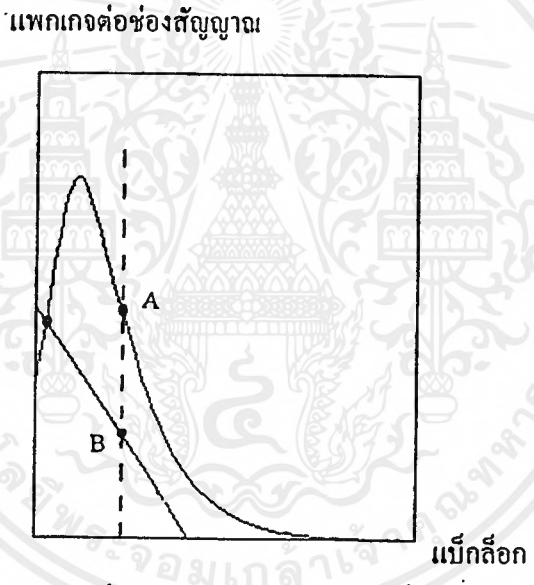
รูปที่ 2.8 อัตราการส่งจริงกับจำนวนแบ็กล็อก

ในการหาประสิทธิภาพของช่องสัญญาณอโลฮา จะใช้ความสัมพันธ์ของแบ็กล็อกและอัตราการส่งจริง ดังแสดงให้เห็นในรูป 2.8 เมื่อแบ็กล็อกมีค่ามากถึง 60 แพกเกต α = 0.1 การพยายามส่งใหม่ 6 แพกเกตต่อช่องสัญญาณ และถ้าสมมุติให้มีแบ็กล็อก 60 แพกเกต α =0.02 ความพยายามในการส่งใหม่ จะมีค่าเพียง 1.2 แพกเกตต่อช่องสัญญาณเท่านั้น



รูปที่ 2.9 อัตราอินพุทกับจำนวนแบ็กล็อก

เมื่อดูแบ็กล็อกถ้าให้มีสถานีอยู่ k สถานีที่เป็นแบ็กล็อก อัตราอินพุท คือ (N-k)p ดังแสดงใน รูป 2.9 สำหรับ p=0.002 และ N = 100

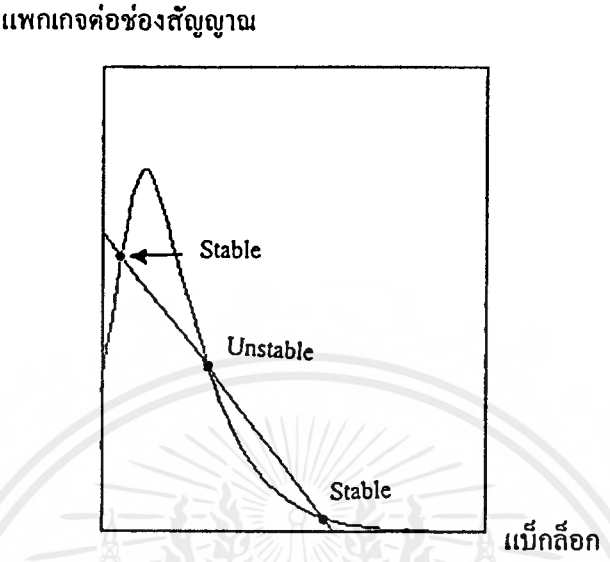


รูปที่ 2.10 จุดสมดุล การหน่วงต่ำ

ในสภาพที่สมดุลอัตราการส่งจริงต้องเท่ากับอัตราอินพุท สามารถแสดงได้ดังรูป 2.10 กับเส้นโหลด แบ่งเป็น 4 กรณี ในภาพ 2.10 N มีค่าน้อยมาก เส้นความสมดุลจะอยู่ที่แบ็กล็อกต่ำ ๆ แต่ถ้าสังเกตดูจะเห็นว่ามีอยู่อีก 1 จุดคือที่เป็นเส้นประในแนวตั้ง ตรงจุดนี้จะเห็นว่าเส้นอัตราส่งจริง A ซึ่งมีค่ามากกว่าเส้นอัตราอินพุท ถึงจะมีการเพิ่มขึ้นหรือลดลงอย่างทันใด ค่าของอัตราอินพุตก็ยังมีค่าน้อยกว่าอัตราส่งจริงดังนั้นทำให้จุดนี้สมดุล

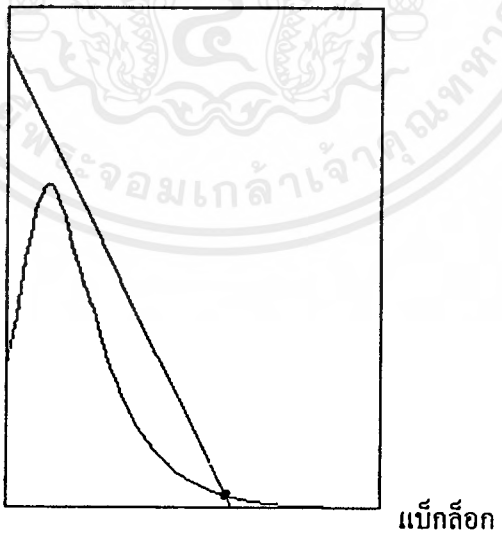
ถ้ามีผู้ใช้หลายคน เส้นโหลดก็จะสูงขึ้นเป็นดังรูปที่ 2.11 ซึ่งจะเห็นว่าเส้นโหลดตัดกับเส้นอัตราการส่งจริงอยู่ 3 จุด ซึ่ง ณ 3 จุดนี้ อัตราอินพุทเท่ากับอัตราการส่งจริง ทั้ง 3 จุดนี้เป็นจุดสมดุล แต่มีเพียง 2 จุดเท่านั้นที่เสถียรภาพ ณ จุดแรก แบ็กล็อกมีค่าน้อย จุดนี้เสถียรภาพ แต่ ณ จุดกึ่งกลาง จุดนี้แบ็กล็อกมีค่าเพิ่มขึ้นทันที จะทำให้อัตราส่งจริงน้อยกว่าอัตราอินพุทอย่างทันทีทันใด แบ็กล็อกก็มีค่าเพิ่ม

ขึ้นอย่างมาก ทำให้ ณ จุดนี้ ไม่เสถียรภาพ เมื่อถึงจุดที่ 3 แบ็กล็อกมีค่ามากก็จริงแต่มีค่าน้อยกว่าอัตราส่งจริง ทำให้จุดนี้มีเสถียรภาพอีกครั้ง



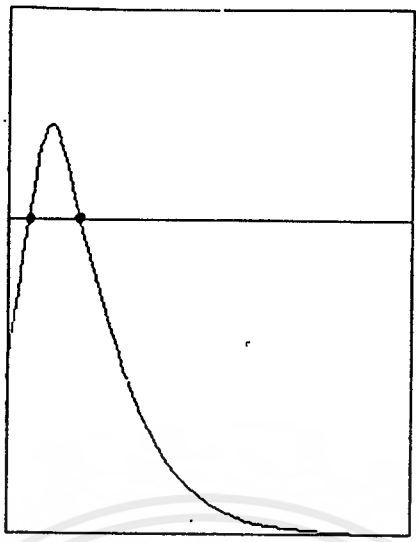
รูปที่ 2.11 จุดสมดุล ที่มีเสถียรภาพ 2 จุด

จากรูปที่ 2.12 แสดงให้เห็นว่า ช่องสัญญาณขณะนี้เกิดโอเวอร์โหลด มีจุดสมดุลอยู่จุดเดียวเท่านั้น ซึ่งจุดนี้ทั้งอัตราอินพุทและอัตราการส่งจริงมีค่าเข้าใกล้ศูนย์ ซึ่งจากรูปนี้แสดงให้เห็นว่ามีผู้ใช้จำนวนมากมายนับไม่ถ้วน อัตราอินพุททำให้แบ็กล็อกมีค่าเพิ่มขึ้นเรื่อย ๆ



รูปที่ 2.12 จุดสมุดล 1 จุดที่เสถียรภาพ การหน่วงสูง

แพกเกจต่อช่องสัญญาณ



แบ็กล็อก

รูปที่ 2.13 จุดสมดุล ไม่เสถียรภาพ

จากรูป 2.13 ภาพนี้แสดงว่าระบบไม่มีเสถียรภาพที่จุดสมดุล เพราะเมื่ออัตราอินพุทเพิ่มขึ้น แบ็กล็อกจะมีค่าเพิ่มขึ้นอย่างไม่มีขีดจำกัด

2.3 ทฤษฎีวงจรขยายสัญญาณความถี่สูงแบบเลือกความถี่

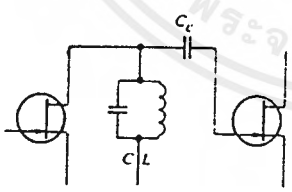
2.3.1 ชนิดของวงจรที่ใช้คัปปลิ้งสัญญาณ

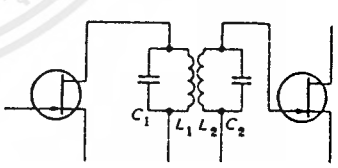
วงจรคัปปลิ้งที่ใช้ในวงจรขยายสัญญาณความถี่สูงแบบเลือกความถี่ แสดงไว้ดังรูปที่ 2.14 โดยที่

(ก) แบบเลือกสัญญาณด้านเดียวโดยใช้ค่าความจุส่งผ่านสัญญาณ

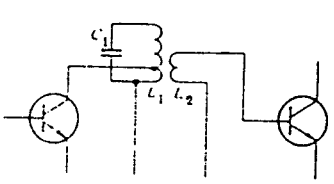
(ข) แบบเลือกสัญญาณคู่

(ค) แบบเลือกสัญญาณด้านเดียว โดยอาศัยผลของอิเล็กโตรแมกเนติกส่งผ่านสัญญาณ





(ก) แบบเลือกสัญญาณด้านเดียวโดยใช้ค่าความจุส่งผ่านสัญญาณ (ข) แบบเลือกสัญญาณคู่

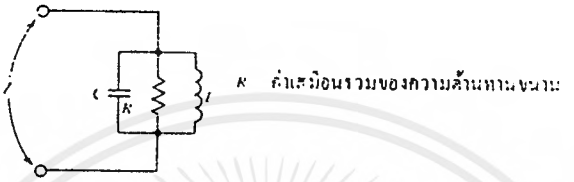


(ค) แบบเลือกสัญญาณด้านเดียว โดยอาศัยผลของอิเล็กโตรแมกเนติกส่งผ่านสัญญาณ

รูปที่ 2.14 แสดงชนิดของวงจรเลือกสัญญาณ

วงจรเลือกสัญญาณชนิดใช้ค่าความจุเป็นตัวส่งผ่านสัญญาณตามรูปที่ 2.14(ก) จะเป็นแบบง่ายที่สุด แต่ให้ค่าอิมพีแดนซ์ของวงจรสูง จึงมักถูกเลือกใช้ในวงจรเฟทบ่อย ๆ เมื่อดัดแปลงให้มีรูปแบบผิดไปเล็กน้อยดังรูปที่ 2.14(ข) จะเป็นวงจรที่มักจะนำมาใช้กับวงจรเฟทเช่นกัน เพราะว่าตัวมันคุณสมบัติให้ค่าแบนด์วิดธ์กว้าง และการปรับเลือกค่าความถี่ทำได้ดีมาก มีการลดทอนสัญญาณภายนอกช่วงความถี่ที่เลือกไว้มาก แต่การออกแบบวงจรเลือกสัญญาณชนิดนี้ยุ่งยากพอสมควรส่วนรูปที่ 2.14(ค) เป็นวงจรที่ง่ายต่อการทำให้เป็นกลางและให้ความสมพงษ์กันของค่าอิมพิแดนซ์ได้ดี จึงนำมาใช้ในวงจรทรานซิสเตอร์เป็นส่วนมาก

2.3.2 คุณสมบัติของวงจรเลือกสัญญาณแบบขนาน



รูปที่ 2.15 แสดงวงจรเลือกความถี่แบบขนาน

จากรูปที่ 2.15 ค่าอิมพิแดนซ์แบบขนาน $\dot{Z}$ จะมีค่าเป็น

$$\dot{Z}=\frac{1}{j\omega C+\frac{1}{R}+\frac{1}{j\omega L}}=\frac{R}{j\omega CR-j\frac{R}{\omega L}+1} \qquad \text{..... 10}$$

ทำการจัดรูปสมการที่ 10 ใหม่จะได้เป็น

$$\dot{Z}=\frac{R}{jQ(\frac{\omega}{\omega_o}-\frac{\omega_o}{\omega})+1} \qquad \text{..... 11}$$

เมื่อ

$$\omega_o^2=\frac{1}{LC}$$

$$Q=\omega_o CR=\frac{R}{\omega_o L} \qquad \text{..... 12}$$

ในสมการที่ 11 ที่ความถี่เชิงมุม ω มีค่าเข้าใกล้ความถี่กำธร ω_o จะได้ค่าโดยประมาณดังต่อไปนี้

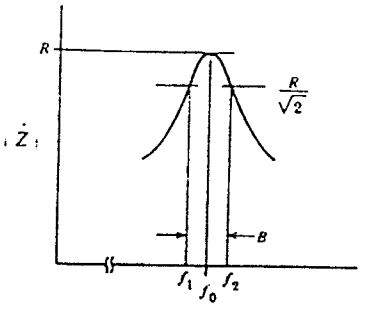
$$\frac{\omega}{\omega_o}-\frac{\omega_o}{\omega}=\frac{\omega^2-\omega_o^2}{\omega_o\omega}=2\frac{\omega-\omega_o}{\omega_o}=2\frac{f-f_o}{f_o} \qquad \text{..... 13}$$

เมื่อความถี่ f_o เป็นความถี่กำธร ดังนั้นสมการที่ 11 จะเป็น

$$\dot{Z}=\frac{R}{j2Q\frac{f-f_o}{f_o}+1} \qquad \text{..... 14}$$

เมื่อ $f=f_o$ ค่า $\dot{Z}_o$ จะเป็น $\dot{Z}_o=R$

นั่นคือ ค่าอิมพิแดนซ์ $\dot{Z}_o$ ที่เกิดจากการกำธร จะเป็นจำนวนจริงและมีค่าเท่ากับความต้านทาน R ที่ประกอบขนานอยู่ ค่า $|\dot{Z}|$ ที่ความถี่ f เข้าใกล้ f_o จะแสดงได้ดังรูปที่ 2.16



รูปที่ 2.16 แสดงคุณสมบัติของ $|\dot{Z}|$

ทำนองเดียวกัน เมื่อค่า $\{2Q(f-f_o)/f_o\}^2 = 1$ สมการที่ 14 ค่า $\dot{Z}$ จะกลายเป็น $|\dot{Z}| = R/\sqrt{2}$ และถ้าความถี่ f_1 และ f_2 เป็นความถี่ที่จุดนี้แล้ว จะได้ว่า

$$f_1 = f_o(1 - \frac{1}{2Q}) \qquad \text{..... 15}$$

$$f_2 = f_o(1 + \frac{1}{2Q})$$

ช่วงความถี่จากความถี่ f_1 ถึงความถี่ f_2 เรียกว่าค่าแบนวิดธ์ของความถี่ที่ยอมให้ผ่านไปได้ B

โดยที่ $$B = f_2 - f_1 = \frac{f_o}{Q}$$

หรือ $$Q = \frac{f_o}{B} \qquad \text{..... 16}$$

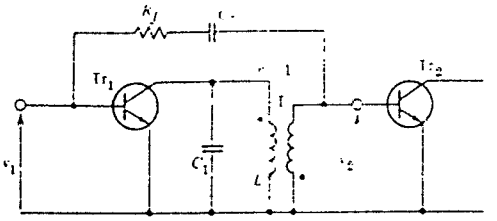
เมื่อค่า f_o และ B เป็นไปตามที่แสดงไว้แล้ว ค่า Q ในสมการที่ 16 จะต้องนำมาพิจารณาดู ถ้าค่า Q ของวงจรมีค่ามากก็ทำให้ค่าความต้านทานที่ขนานอยู่ต่ำลงเช่นกัน ค่าความต้านทานนี้เรียกว่าค่าความต้านทานแดมปิ้ง Q ในทางกลับกัน เมื่อใช้อุปกรณ์ที่มีค่าอิมพิแดนซ์ต่ำต่อเข้ากับวงจร จะไม่สามารถทำให้ค่า Q สูงขึ้นได้ ค่า Q สามารถทำให้เพิ่มขึ้นได้ โดยต่ออุปกรณ์เข้าที่จุดที่แทป ออกมาบริเวณกลางของขดลวดอย่างเหมาะสม วิธีการนี้เรียกว่าการเพิ่มค่า Q โดยการแทป ตัวอย่างของการเพิ่มค่า Q โดยการแท็ปทางด้านขดปฐมภูมิของขดลวด แสดงไว้ดังรูปที่ 2.14(ค)

2.3.3 วงจรขยายสัญญาณแบบใช้วงจรเลือกสัญญาณแบบเดี่ยว

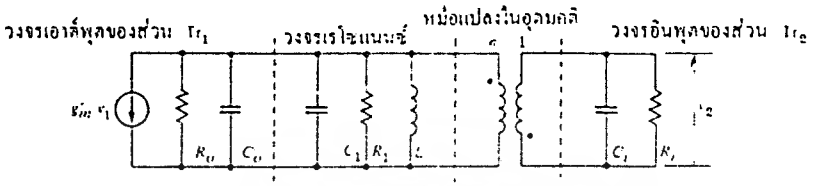
โดยทั่วๆ ไป ค่าอัตราการขยายสัญญาณ A_v ของวงจรขยายสัญญาณ แสดงได้ดังสมการที่ 17

$$A_v = g_m Z \qquad \text{..... 17}$$

เมื่อ Z คือค่าอิมพิแดนซ์ของโหลด ถ้าสมมุติว่าคุณสมบัติทางด้านความถี่ของ $|Z|$ เป็นดังรูปที่ 2.16 คุณสมบัติทางด้านความถี่ของ $|A_v|$ จะมีลักษณะเช่นเดียวกัน จึงเป็นไปได้ว่าจะสามารถเลือกและขยายสัญญาณตามช่วงความถี่ที่กำหนดได้



(ก) วงจรทรานซิสเตอร์ที่ใช้ขยายสัญญาณโดยใช้วงจรเลือกสัญญาณแบบเดี่ยว



(ข) วงจรทัดเทียมของรูป (ก)

รูปที่ 2.17 แสดงวงจรเลือกสัญญาณแบบเดี่ยวที่ใช้ในวงจรขยายสัญญาณ

ตามรูปที่ 2.17(ก) เมื่อมีสัญญาณป้อนเข้าไปทางด้านขดปฐมภูมิของทรานฟอเมอร์ T ด้านทุติยภูมิของมันจะได้รับสัญญาณออกมาเหมือนกับที่ป้อนเข้าไปทางด้านขดปฐมภูมิ ลักษณะกระทำเช่นนี้เรียกว่าวงจรคัปปลิ้ง วงจรคัปปลิ้งนี้สามารถถือได้ว่าประกอบด้วยค่าความเหนี่ยวนำ L และทรานฟอเมอร์ทางอุดมคติมีอัตราส่วน n:1 และถือได้ว่าฟังก์ชันของมันแยกกันโดยอิสระ ด้วยเหตุนี้จึงได้วงจรทัดเทียมออกมาตามรูปที่ 2.17(ข)

ค่า $g_{m'}, R_o$ และ C_o ของวงจรทัดเทียมรูป 2.17(ข) คล้ายคลึงกับค่า y_{fe} และ y_{oe} (โดยถือว่า $y_{re} = 0$) ดังนั้น

$$g_{m'} = y_f = \frac{g_m}{j\omega C_e r_{bb'} + 1}$$

$$R_o = \frac{C_e}{C_c g_m} \qquad \text{..... 18}$$

$$C_o = C_c$$

ทำนองเดียวกันค่า C_i และ R_i จะสอดคล้องกับ y_{ie} จึงได้ว่า

$$C_i = \frac{C_e}{1 + (\omega C_e r_{bb'})^2}$$

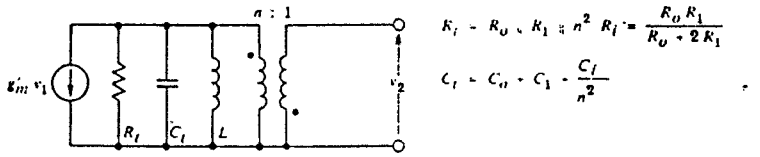
$$R_i = \frac{1 + (\omega C_e r_{bb'})^2}{(\omega C_e)^2 r_{bb'}} \qquad \text{..... 19}$$

ค่าความต้านทาน R_i เป็นองค์ประกอบด้านความต้านทานของวงจรกำธร เนื่องจากเป็นค่าความต้านทานอันเกิดจากตัวขดลวดเอง ดังนั้นจึงควรต่ออนุกรมอยู่กับขดลวด แต่ในวงจรกำธรแบบขนานจะทำให้พิจารณาได้สะดวกกว่าเมื่อเขียนค่าความต้านทานขนานไว้ในวงจรทัดเทียม กำลังที่สูญเสียไปโดยความต้านทาน R_i เรียกว่า การสูญเสียแบบอินเสิทชั่น (insertion loss) ค่าความต้านทาน R_i มีค่าน้อยลง ค่า Q ของขดลวดจะมีค่ามากขึ้น ค่า Q_o เป็นค่า Q ในขณะที่ไม่มีโหลด

จากสมการที่ 12 ค่าความต้านทาน R_i และ Q_o จะมีความสัมพันธ์กันดังนี้

$$R_i = Q_o \omega_o L \qquad \text{..... 20}$$

เพราะว่าค่าความต้านทาน หรือ ค่ารีแอคแตนซ์ ที่ต่ออยู่ทางด้านขดปฐมภูมิ หรือ ขดทุติยภูมิของหม้อแปลงในอุดมคติ สามารถจะเปลี่ยนกลับไปมาทั้งสองด้านได้ โดยมีความสัมพันธ์กันในลักษณะกำลังสองของอัตราจำนวนรอบที่พันของทั้งสองขด ซึ่งในที่นี้คือ n^2 ดังนั้นค่า C_i และ R_i ของด้านสัญญาณเข้าของทรานซิสเตอร์ Tr_2 จะถูกเปลี่ยนให้ไปอยู่ทางด้านขดปฐมภูมิ และจัดรูปใหม่จะได้ดังรูปที่ 2.18



รูปที่ 2.18 แสดงวงจรทัดเทียมที่ถูกเปลี่ยนรูปแบบใหม่

เพื่อที่จะให้เกิดความสมพงษ์กันของค่าอิมพิแดนซ์ อัตราส่วนของขดลวดของทรานสฟอเมอร์จะหาได้จาก

$$R_o = n^2 R_i \quad \text{..... 21}$$

$$\therefore \quad n = \sqrt{\frac{R_o}{R_i}} \quad \text{..... 22}$$

เนื่องจากค่าความต้านทาน R_t ที่ต่อขนานอยู่ได้มาจากการรวมกันของ R_o, R_1 และ $n^2 R_i$ ดังนั้นสมการที่ 21 จะเป็น

$$R_t = \frac{(R_o/2)\cdot R_1}{(R_o/2)+R_1} = \frac{R_O R_1}{R_O + 2R_1} \quad \text{..... 23}$$

ในทางกลับกัน ค่า Q ขณะที่ใช้งานเรียก Q_t จะหาได้จากค่าแบนวิดธ์ B กับความถี่ที่เกิดกำธร f_o ฉะนั้นจากสมการที่ 12 ค่า L และ Ct จะหาค่าได้ดังนี้

$$L = \frac{R_t}{Q_t \omega_O} = \frac{R_O R_1}{Q_t \omega_O (R_O + 2R_1)} \quad \text{..... 24}$$

$$C_t = \frac{Q_t}{\omega_O R_t} = \frac{Q_t (R_O + 2R_1)}{\omega_O R_O R_1} = \frac{1}{\omega_O^2 L} \quad \text{สมการที่ 25}$$

และตัวเก็บประจุ C_1 ที่ใช้เลือกความถี่จะมีค่าเป็น

$$C_1 = \frac{1}{\omega_O^2 L} - C_O - \frac{C_i}{n^2} \quad \text{..... 26}$$

จากสมการที่ 13 ค่าอัตราการขยายศักดาสัญญาณ A_v ที่จุดกึ่งกลางของย่านความถี่จะหาค่าได้ดังสมการที่ 27 โดยที่ค่าอิมพิแดนซ์ของจุดกึ่งกลางความถี่คือ $Z_o = R_t$

$$A_V = \frac{v_2}{v_1} = \frac{g_m \cdot v_1 R_t}{v_1} \cdot \frac{1}{n} = \frac{g_m \cdot R_t}{n} \quad \text{..... 27}$$

จากสมการที่ 14 ค่าอัตราการขยายศักดาสัญญาณ A_m ซึ่งอยู่ที่ความถี่ใด ๆ นอกจุดกึ่งกลางของย่านความถี่จะเป็น

$$A_{vt} = \frac{g_{m'}}{n} \cdot \frac{R_t}{j2Q_t \frac{f - f_o}{f_o} + 1}$$

หรือ

$$|A_{vt}| = \frac{g_{m'} R_t}{n} \cdot \frac{1}{\sqrt{(2Q_t \frac{f - f_o}{f_o})^2 + 1}} \quad \text{..... 28}$$

2.3.4 วงจรขยายสัญญาณที่ใช้เฟท

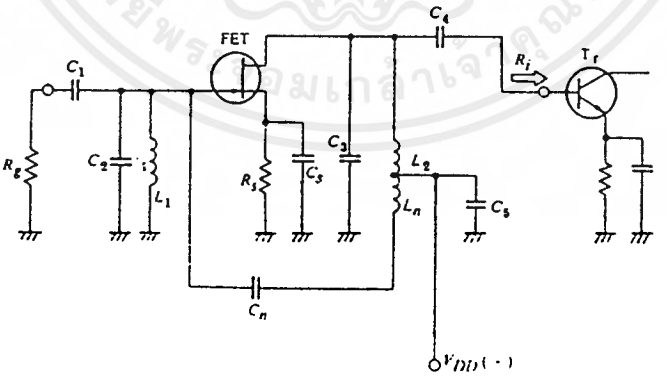
เนื่องจากค่า gm ของเฟทมีค่าต่ำ ดังนั้นเมื่อเปรียบเทียบกับทรานซิสเตอร์แล้ว ค่าอัตราการขยายสัญญาณของวงจรจะต่ำกว่า แต่การใช้เฟทก็มีข้อดีกว่าทรานซิสเตอร์หลายประการดังนี้

1) คุณสมบัติเกี่ยวกับการเกิดผสมสัญญาณขึ้นภายในตัวดีมาก : การเกิดผสมสัญญาณขึ้นภายในเป็นปรากฏการณ์ของการเกิดผสมสัญญาณของคลื่นที่สอดแทรกเข้ามาภายใน และส่งสัญญาณนี้ออกไปปรากฏที่ด้านสัญญาณออก ถ้าต้องการดูสัญญาณนี้ จะทำได้โดยป้อนสัญญาณเลียนแบบการสอดแทรกเข้าไปที่จุดสัญญาณเข้า สาเหตุของการเกิดการผสมสัญญาณภายในเนื่องมาจากคุณสมบัติที่ไม่เป็นเชิงเส้นของตัวอุปกรณ์นั่นเอง แต่เนื่องจากเฟทมีคุณสมบัติในรูปของกำลังสอง ดังนั้นการกำเนิดสัญญาณที่เป็นการผสมสัญญาณภายในเกือบจะตัดทิ้งออกไปได้ทั้งหมด ด้วยเหตุนี้เอง เฟทจึงมักจะถูกนำไปใช้ในวงจรเครื่องรับเอฟเอ็ม หรือ เครื่องรับโทรทัศน์

2) ค่าอิมพิแดนซ์ทางด้านสัญญาณเข้าจะมีค่าสูง นั่นหมายความว่า loaded Q ของวงจรเลือกสัญญาณจะทำให้มีค่าสูงได้ ยังผลให้การเลือกความถี่ที่ใช้งานทำได้ดีขึ้นด้วย

3) สัญญาณรบกวนที่เกิดจากตัวเฟทเองมีค่าต่ำมาก ดังนั้นวงจรขยายสัญญาณที่ใช้เฟท จึงมีค่าอัตราส่วนของสัญญาณต่อสัญญาณรบกวนดีมาก

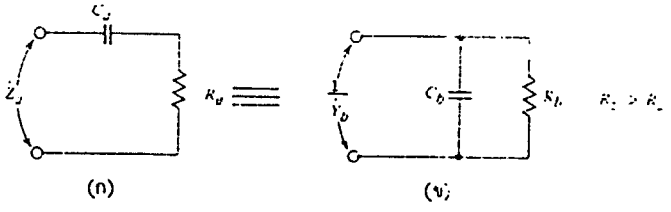
จากที่อธิบายมาแล้วทั้งเฟทและทรานซิสเตอร์จะมีข้อดีข้อเสียต่างกันไป ดังนั้นจึงขึ้นอยู่กับการเลือกใช้วงจรเพื่อที่ได้คุณสมบัติครอบคลุมข้อดีไว้ได้มาก โดยมีข้อเสียน้อยที่สุดนั่นเอง



รูปที่ 2.19 วงจรขยายสัญญาณความถี่สูงโดยใช้เฟท

จากรูปที่ 2.19 จะเป็นตัวอย่างวงจรขยายสัญญาณความถี่สูงที่ใช้เฟท ที่มีวงจรเลือกสัญญาณด้านสัญญาณเข้าประกอบด้วย L_1 และ C_2 และวงจรเลือกสัญญาณด้านขาออกประกอบด้วย L_2 และ C_3 ส่วน L_n และ C_n จะทำให้ผลของการป้อนกลับแบบลบภายในกลายเป็นกลางไปตัวเก็บประจุ C_1 และ C_4 จะเป็นตัวเก็บประจุคัปปลิ้งที่ทำหน้าที่เป็นตัวเปลี่ยนค่าอิมพิแดนซ์ ค่าความต้านทาน R_g จะเป็นค่าอิม

พิแดนซ์ของสายอากาศและมีค่าต่ำมาก ค่าความต้านทาน R_i จะเป็นค่าอิมพิแดนซ์ด้านสัญญาณเข้าของวงจรในภาคถัดมา ซึ่งเป็นวงจรของทรานซิสเตอร์ และจะมีค่าต่ำไม่เหมาะสมกับวงจรของเฟท ดังนั้นตัวเก็บประจุ C_1 และ C_4 จะเป็นฟังก์ชันที่ใช้เปลี่ยนค่าอิมพิแดนซ์เพื่อให้สมพงษ์กัน โดยอาศัยหลักการตามรูปที่ 2.20



รูปที่ 2.20 การเปลี่ยนค่าอิมพิแดนซ์โดยการใช้ตัวเก็บประจุ C_a เข้ามาอนุกรม

เมื่อนำตัวเก็บประจุ C_a ไปอนุกรมกับ R_a ยังผลให้ได้วงจรทัดเทียมแบบขนานที่มีความต้านทาน R_b ใหญ่กว่าความต้านทาน R_a อาศัยผลข้อนี้ช่วยทำการเพิ่มค่าอิมพิแดนซ์ให้สูงขึ้น ถ้าค่าอิมพิแดนซ์ของรูป 2.50(ก) เป็น Z_a แล้ว มันจะมีค่าเป็น

$$\dot{Z}_a = \frac{1}{j\omega C_a} + R_a \qquad \text{..... 29}$$

และค่าส่วนกลับของ Z_a จะเป็น

$$\frac{1}{\dot{Z}_a} = \frac{1}{(1/j\omega C_a) + R_a} = \frac{(\omega C_a)^2 R_a}{1+(\omega C_a R_a)^2} + j\frac{\omega C_a}{1+(\omega C_a R_a)^2} \qquad \text{..... 30}$$

แต่จาก $1/\dot{Z}_a = \dot{Y}_b$ ค่า R_b และ C_b จะได้เป็น

$$R_b = \frac{1}{(\omega C_a)^2 R_a} + R_a \qquad \text{..... 31}$$

$$C_b = \frac{C_a}{1+(\omega C_a R_a)^2} \qquad \text{..... 32}$$

ตามสมการที่ 31 เทอมแรกด้านขวามือจะเป็นค่าความต้านทานที่เพิ่มขึ้น (ในกรณีที่ใช้ตัวเก็บประจุคัปปลิ้งแบบธรรมดาที่มีค่ามาก จะทำให้เทอมนี้ตัดทิ้งไปได้ ผลที่ได้คือ $R_b = R_a$) และจากสมการที่ 32 ทำให้เข้าใจได้ว่า ค่าความจุทัดเทียมของ C_b มีค่าน้อยกว่า C_a ประสิทธิภาพของวงจรขยายสัญญาณแบบเลือกความถี่ สามารถอธิบายได้โดยผลคูณของอัตราการขยายศักดาสัญญารกับค่าของแบนด์วิดธ์หรือ GB ดังนี้

$$GB = A_v B = g_m R_t B$$

แต่จากสมการที่ 25 และ 15

$$R_t = \frac{Q_t}{C_t \omega_O}, \qquad B = \frac{f_O}{Q_t}$$

ดังค่า GB จะเป็น

$$GB = \frac{1}{2\pi} \cdot \frac{g_m}{C_t} \qquad \text{..... 33}$$

จากสมการนี้จะเห็นได้ว่า ค่า C_t ยังมีค่าน้อยเท่าไร จะทำให้วงจรมีประสิทธิภาพมากขึ้นเท่านั้น กลับมาพิจารณาในรูปที่ 2.49 ต่อไป เพราะว่าคุณสมบัติของสัญญาณรบกวนของเฟท จะเปลี่ยนแปรขึ้น

กับค่าอิมพิแดนซ์ของแหล่งจ่ายสัญญาณ ค่าความจุของ C_1 ที่ใส่เข้าไปจะทำให้สัญญาณรบกวนที่เกิดขึ้นมีผลน้อยที่สุดเท่าที่เป็นไปได้ ส่วนค่าความจุของ C_4 จะทำให้เกิดการสมพงษ์กันของค่าอิมพิแดนซ์ ด้านสัญญาณออกของเฟทกับค่าอิมพิแดนซ์ด้านสัญญาณเข้าของวงจรภาคถัดมา

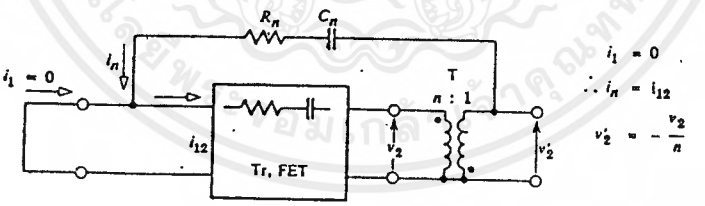
ค่า L_1 และ L_2 สามารหาได้จากการประกอบกันอย่างขนานของความต้านทาน R_t และ loaded Q(Q_t) ตามสมการที่ 24 และค่า C_2 และ C_3 หาได้จากสมการที่ 26 ส่วนค่า C_n และ L_n จะหาได้เช่นกัน

2.3.5 วงจรที่ช่วยทำให้เป็นกลางหรือวงจรสะเทินสัญญาณ

ผลของการป้อนกลับเนื่องมาจากตัวเก็บประจุภายใน C_c และ C_d ของวงจรทรานซิสเตอร์แบบอิมิตเตอร์ร่วมและของวงจรเฟทแบบต่อซอร์สลงกราวด์ จะไม่สามารถตัดทิ้งไปได้ในย่านความถี่สูง เมื่อการป้อนกลับภายในมีค่ามากขึ้น จะทำให้วงจรไม่มีเสถียรภาพและผลที่เกิดกับด้านสัญญาณออกจะส่งผลไปด้านสัญญาณเข้าด้วย ทั้งยังทำให้มีปัญหาบางประการเกี่ยวกับการเลื่อนออกไปของความถี่ที่เลือกไว้แล้วด้วย

วงจรที่ทำให้ผลเหล่านี้เป็นกลางได้อย่างสมบูรณ์แบบแล้วสัญญาณจะส่งผ่านโดยตรงจากด้านสัญญาณเข้า (เบสหรือเกท) ตรงไปยังด้านสัญญาณออก (คอลเลคเตอร์หรือเดรน) เท่านั้น สัญญาณที่ย้อนกลับทางจะมีค่าเป็นศูนย์ ด้วยเหตุนี้ วงจรที่ทำให้เป็นกลางจึงเรียกได้อีกอย่างหนึ่งว่าวงจรทิศทางเดียว

หลักการเบื้องต้นของการทำให้เป็นกลางคือ ป้อนสัญญาณจากภายนอกเข้าไปให้มีขนาดเดียวกับสัญญาณที่ต้องการทำให้เป็นกลาง แต่มีทิศทางตรงข้ามกับสัญญาณที่เกิดการป้อนกลับอยู่ภายในหรือสัญญาณที่ต้องการทำให้เป็นกลางนั่นเอง ด้วยหลักการนี้จะทำให้ผลของการป้อนกลับที่อยู่ภายในหมดไป ป้องกันไม่ให้สัญญาณจากด้านสัญญาณออกส่งผลมายังด้านสัญญาณเข้า หลักการนี้แสดงได้ดังรูปที่ 2.21



รูปที่ 2.21 หลักของการทำให้เป็นกลางหรือสะเทินสัญญาณป้อนกลับภายใน

ตามรูปที่ 2.21 โดยอาศัยหลักความจริงที่ว่า เมื่อเกิดการเป็นกลางขึ้นจริงแล้ว ค่ากระแส $i_1 = 0$ นั่นคือ

$$i_n = i_{12}$$

และนี้คือข้อกำหนดของการสะเทินสัญญาณ จากคำจำกัดความขององค์ประกอบแบบวาย i_{12} จะเป็น

$$i_{12} = y_r v_2$$

จากรูปที่ 2.21 i_n มีค่าเป็น

$$i_n = \frac{v_2'}{R_n + (1/j\omega C_n)} = -\frac{v_2}{n} \cdot \frac{1}{R_n + (1/j\omega C_n)}$$

นั่นคือ $$v_2' = -\frac{v_2}{n}$$

ดังนั้น ค่า y_r จะกลายเป็น

$$y_r = -\frac{1}{n(R_n + (1/j\omega C_n)} \quad \text{..... 34}$$

นี่คือค่า y_r ในรูปของไฮบริด-พาย ฉะนั้นค่า y_{re} จะเป็น

$$y_{re} = -\frac{j\omega C_c}{j\omega C_e r_{bb'} + 1} = -\frac{1}{(C_e r_{bb'}/C_c) + (1/j\omega C_c)} \quad \text{..... 35}$$

นั่นคือจากสมการที่ 34 และ 35 ค่า R_n และ C_n จะหาได้ดังนี้

$$R_n = \frac{1}{n} \cdot \frac{C_e r_{bb'}}{C_C}$$

$$C_n = nC_c$$

และในกรณีของเฟท ค่า y_{rs} จะเป็น

$$y_{rs} = -j\omega C_{dg} = -\frac{1}{1/j\omega C_{dg}}$$

นั่นคือ ค่า R_n และ C_n จะเป็น

$$R_n = 0 \qquad C_n = nC_{dg}$$

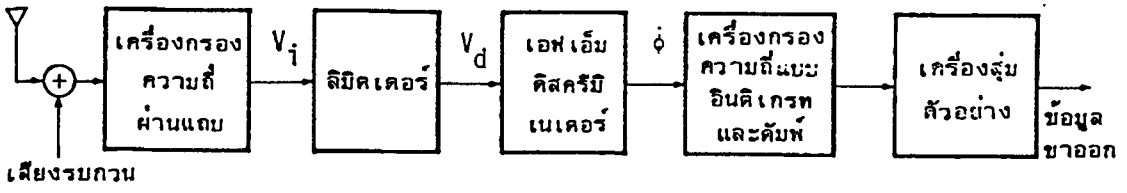
เมื่อค่า n คืออัตราส่วนของจำนวนรอบของขดลวดหม้อแปลง แต่ในรูปที่ 2.19 ซึ่งเป็นวงจรขยายสัญญาณที่ใช้เฟท ค่า n จะเป็นอัตราส่วนของจำนวนของจำนวนรอบของขดลวด L_2 และ L_n

2.4 ดิจิตัลเอฟเอ็ม

การสื่อสารดิจิตัลแบบง่ายๆ ที่ดัดแปลงระบบเอฟเอ็มมาใช้ในการโมดูเลทและส่งสัญญาณดิจิตัลไบนารี '1' หรือ '0' นี้ เรียกว่า ดิจิตัลเอฟเอ็ม (Digital FM) หรือไบนารีเอฟเอสเค (Binary FSK) การสื่อสารดิจิตัลเอฟเอ็มนี้ นิยมใช้กันในสมัยเริ่มแรกของการพัฒนาระบบสื่อสารดิจิตัล ในปัจจุบันก็ยังคงใช้กันอยู่ในวงการสื่อสารดิจิตัลที่ต้องการส่งข้อมูลในอัตราความเร็ว (Bit Rate) ต่ำกว่า 1200 บิตต่อวินาที และประหยัดค่าใช้จ่าย ความแตกต่างระหว่างอะนาล็อกเอฟเอ็มและดิจิตัลเอฟเอ็ม กล่าวคือ อะนาล็อกเอฟเอ็ม ส่วนมากเป็นระบบเอฟเอ็มที่มีแถบความถี่กว้าง (Wideband FM) ส่วนดิจิตัลเอฟเอ็ม ส่วนมากเป็นระบบเอฟเอ็มที่มีแถบความถี่แคบ (Narrowband FM)

เครื่องรับของระบบดิจิตัลเอฟเอ็มส่นมากใช้ลิมิตเตอร์ดิสคริมิเนเตอร์ตามด้วยเครื่องกรองความถี่แบบอินทีเกรทและดัมพ์ (integrate-and-dump filter) ดังรูปที่ 2.22 สัญญาณดิจิตัลเอฟเอ็มที่ขาเข้าของเครื่องรับจะถูกรบกวนด้วยเสียงรบกวนแบบเก๊าเซียน และผ่านเครื่องกรองความถี่กลาง (IF Filter) ที่มีความถี่ศูนย์กลางเท่ากับ f_0 และมีแถบความถี่กว้าง 2B และจะถูกดีโมดูเลทโดยเครื่องลิมิตเตอร์ดิสคริมิเนเตอร์ ในที่นี้เราสมมติว่าแถบความถี่ของเครื่องกรองความถี่แถบกลางกว้างพอที่จะให้สัญญาณดิจิตัลเอฟเอ็มผ่านโดยไม่มีความผิดเพี้ยน สัญญาณขาออกจากเครื่องลิมิตเตอร์ดิสคริมิเนเตอร์จะผ่านเครื่องกรองความถี่แบบอินทีเกรทและดัมพ์ และเครื่องสุ่มตัวอย่างที่ตัดสินขนาดของพัลซ์ผสมกับสัญญาณที่มีค่ามาก

กว่าค่าเทรสโฮลด์ ซึ่งอาจตั้งไว้ที่ค่าศูนย์ให้เป็นพัลซ์บวกและที่มีค่าน้อยกว่าค่าเทรสโฮลด์เป็นพัลซ์ลบ ดังรูปที่ 2.22



รูปที่ 2.22 แผนภาพของเครื่องรับดิจิตัลเอฟเอ็ม

สัญญาณดิจิตัลเอฟเอ็มที่ขาเข้าของเครื่องรับคือ

$$S(t) = A \cdot \cos(\omega_c t + D(t) + \theta) \quad \text{..... 36}$$

ในที่นี้ A คือ อแมพลิจูดของตัวพา

ω_c คือ ความถี่เชิงมุมของตัวพา

θ คือ เฟสเริ่มต้นของตัวพา

$D(t)$ คือ สัญญาณโมดูเลตติงดิจิตัลเอฟเอ็ม

$$D(t) = \omega_d \int_0^t x(t')\, dt' \quad \text{..... 37}$$

และสัญญาณข้อมูลเบสแบนด์ x(t) เป็นพัลซ์ซีเควนชนิดไบนารี g(t) ที่มีขนาดเป็นบวกหรือลบ a_n g(t) นี้จะมีแรงดันเปลี่ยนตามความถี่เบี่ยงเบนของสัญญาณดิจิตัลเอฟเอ็ม ดังสมการข้างล่างนี้

$$\chi(t) = \sum_{n=-\infty}^{\infty} a_n\, g(t - nT) \quad \text{..... 38}$$

เสียงรบกวนที่ขาเข้าของเครื่องรับ คือ

$$\begin{aligned} n(t) &= \alpha(t) \cos(\omega_c t + \lambda(t)) \\ &= n_1(t) \cos \omega_c t - n_2(t) \sin \omega_c t \end{aligned} \quad \text{..... 39}$$

ในที่นี้ $\alpha(t)$ เป็นเอ็นเวลลอปของเสียงรบกวนแบบเก๊าเซียน $n(t)$ ซึ่งมีการกระจายแบบเรย์เล่ย์ (Rayleigh distribution) $\lambda(t)$ เป็นเฟสของเสียงรบกวนที่มีการกระจายแบบสม่ำเสมอระหว่างค่า 0 และ 2π n_1 และ n_2 เป็นส่วนประกอบอินเฟสและควอดราเจอร์ของ $n(t)$ เทียบกับความถี่ของตัวพา

สัญญาณผสมเสียงรบกวนที่ขาเข้าของเครื่องลิมิตเตอร์ เขียนเป็นสมการได้ดังนี้

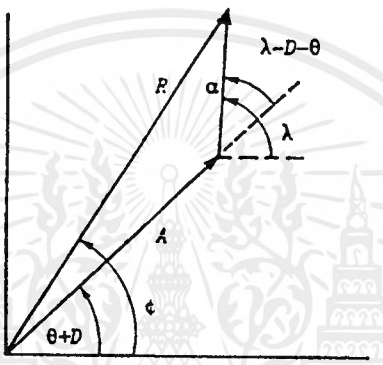
$$v(t) = s(t) + n(t) = R(t)\cos(\omega_c t + \phi(t)) \quad \text{..... 40}$$

ในที่นี้

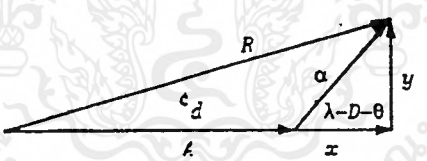
$$R(t) = \sqrt{[A+x]^2 + y^2} \quad \text{..... 41}$$

$$\phi(t) = D + \theta + \tan^{-1}\frac{y}{A+x} \quad \text{..... 42}$$

ดังแสดงในรูปที่ 2.23



(ก) การเปรียบเทียบอัตราความผิดพลาดเทียบกับระบบเอฟเอสเคแบบไบนารี



(ข) การเปรียบเทียบเครื่องดิสคริมิเนเตอร์เอฟเอ็มและเครื่องกรองความถี่แบบแมช

รูปที่ 2.23 ภาพแสดงความสัมพันธ์ทางเฟสเซอร์ระหว่างสัญญาณและเสียงรบกวน

ในที่นี้เสียงรบกวนถูกแยกออกเป็นส่วนประกอบอินเฟสและควิดราเจอร์กับสัญญาณดังนี้

$$x \equiv \alpha\cos(\lambda - D - \theta)$$
$$y \equiv \alpha\sin(\lambda - D - \theta) \quad \text{..... 43}$$

เครื่องลิมิตเตอร์จะตัดเอ็นเวลลอปของสัญญาณผสมระหว่างสัญญาณและเสียงรบกวนให้เรียบที่ขาออกของเครื่องลิมิตเตอร์ที่ตามด้วยเครื่องกรองความถี่ผ่านแถบ (Bandpass Filter) จะได้

$$V_d(t) = \cos(\omega_c t + \phi(t))$$

ที่ขาออกของเครื่องดิสคริมิเนเตอร์ คือ

$$\phi = D + \phi_u \quad \text{.....} \quad 44$$

ในที่นี้ D คือสัญญาณที่ถูกดีโมดูเลทตามต้องการ และ ϕ_u คือเสียงรบกวนขาออก พฤติกรรมของเสียงรบกวนในระบบดิจิตัลเอฟเอ็มก็มีลักษณะเดียวกับพฤติกรรมของเสียงรบกวนในระบบอะนาล็อกเอฟเอ็ม กล่าวคือ ที่ขาออกของเครื่องรับเอฟเอ็ม เสียงรบกวนจะประกอบด้วยเสียงรบกวน 2 ชนิด ชนิดแรกเป็นเสียงรบกวนแบบเก๊าเซียน และชนิดหลังเป็นเสียงรบกวนแบบอิมพัลซ์ หรือเรียกว่า เสียงรบกวนคลิ๊ค เพราะเวลาเกิดเสียงรบกวนคลิ๊กจะได้ยินเสียงคลิ๊คๆ จากเครื่องรับ และเสียงรบกวนแบบเก๊าเซียนจะมีอิทธิพลต่อสัญญาณดิจิตัลเอฟเอ็ม ที่ค่าของอัตราส่วนของกำลังของตัวพาต่อกำลังของเสียงรบกวน (CNR) สูง ส่วนที่ค่าของอัตราส่วนของกำลังของตัวพาต่อกำลังของเสียงรบกวนต่ำของเสียงรบกวนแบบคลิ๊คจะมีอิทธิพลต่อสัญญาณดิจิตัลเอฟเอ็มมากกว่าเสียงรบกวนแบบเก๊าเซียน

ในขณะที่ขนาดของ x และ y สั่นกระเพื่อมอย่างแรนดัม $\phi_u(t)$ ก็จะสั่นกระเพื่อมตามด้วย ที่ค่าของอัตราส่วนของกำลังของสัญญาณต่อกำลังของเสียงรบกวน ρ มาก นั่นคือ $\alpha(t) << A$ การสั่นกระเพื่อมของ $\phi_u(t)$ จะน้อย ดังนั้น เราสามารถแทน $\phi_u(t)$ ด้วยสมการข้างล่างนี้

$$\begin{aligned} \phi_u(t) &= \tan^{-1} \frac{\alpha \sin(\lambda - D - \theta)}{A + \alpha \cos(\lambda - D - \theta)} \\ &\cong \frac{\alpha \sin(\lambda - D - \theta)}{A} = \frac{y}{A} \end{aligned} \quad \text{.....} \quad 45$$

เสียงรบกวนที่ขาออกของเครื่องดิสคริมิเนเตอร์ คือ

$$\phi_u(t) = \frac{y}{A} \quad \text{.....} \quad 46$$

เนื่องจาก y เป็นตัวแปรแรนดัมแบบเก๊าเซียน การแปลงเชิงเส้นของตัวแปรแรนดัมแบบเก๊าเซียนก็ยังคงเป็นตัวแปรแบบเก๊าเซียนอยู่ ฉะนั้น $\phi_u(t)$ จึงเป็นเก๊าเซียนด้วย

ในขณะที่สัญญาณมีกำลังลดลงในบางเวลา ขนาดของเสียงรบกวนจะมีขนาดมากกว่าขนาดของสัญญาณ และมีโอกาสที่ R(t) ซึ่งเป็นเอ็นเวลลอปรวมของสัญญาณดิจิตัลเอฟเอ็มกับเสียงรบกวนจะหมุนไปทางซ้ายมือล้อมรอบจุดศูนย์กลางอย่างรวดเร็ว เกิดการเปลี่ยนมุมเป็น 2π เครื่องดิสคริมิเนเตอร์จะเปลี่ยนจากมุม 2π เป็นพัลซ์แหลมเล็กที่มีพื้นที่ $+2\pi$ และมีความกว้างเล็กกว่า $1/(2B)$ เมื่อผ่านเครื่องกรองความถี่เบสแบนด์ที่ติดกับเครื่องดิคริมิเนเตอร์ ซึ่งมีแถบความถี่ที่ยอมให้เสียงรบกวนแบบคลิ๊คผ่านได้สะดวก เสียงรบกวนแบบคลิ๊คจึงสามารถแทนด้วยอิมพัลซ์ที่มีพื้นที่ 2π ได้ดังนี้

$$\phi u(t) \equiv Z_+ - Z_- = 2\pi\left[\sum_{k=-\infty}^{\infty}\delta(t-t_k) - \sum_{l=-\infty}^{\infty}\delta(t-t_l)\right] \quad \text{..... 47}$$

ในที่นี้ Z_+ และ Z_- เป็นเสียงรบกวนแบบคลิ๊คบวกและลบ t_k และ t_l เป็นเวลาที่เสียงรบกวนแบบคลิ๊คบวกและลบเกิด

เสียงรบกวนแบบคลิ๊คจะมีการกระจายแบบพอยซัน (Poisson distribution) ซึ่งเราจำเป็นต้องทราบอัตราการเกิดของเสียงรบกวนแบบคลิ๊คเพื่ออธิบายลักษณะสมบัติเชิงสถิติของเสียงรบกวนแบบคลิ๊คได้ สมมติว่าความเข้มของสเปคตรัลของเสียงรบกวนขาเข้าสมมาตรกับความถี่ศูนย์กลางของสัญญาณเอฟเอ็ม fc จำนวนเฉลี่ยของสัญญาณรบกวนแบบคลิ๊คบวกและลบใน 1 วินที วิศวกรสื่อสารชาวอเมริกันชื่อไรซ์ (Rice) ได้คำนวณค่าไว้ดังนี้

$$N_+ = \frac{1}{2}\left\{\sqrt{\Gamma^2 + f_d^2}\left[1 - erf\left(\sqrt{\rho + \rho f_d^2/\Gamma}\right)\right] - f_d \exp(-\rho)\left[1 - erf\left(f\sqrt{\rho/\Gamma}\right)\right]\right\}$$

ในขณะที่จำนวนเฉลี่ยของเสียงรบกวนแบบคลิ๊คใน 1 วินาทีคือ

$$N_- = N_+ + f_d \exp(-\rho) \quad \text{..... 49}$$

ในขณะที่ Γ คือแถบความถี่ที่อาร์เอสเอ็ม (rsm bandwidth) ของเสียงรบกวนขาเข้ามีหน่วยเป็นเฮิรตซ์ต่อวินาที ดังนั้น

$$\Gamma = \frac{1}{2\pi}\sqrt{\frac{b_2}{b_0}} \quad \text{..... 50}$$

$$b_0 = \sigma^2 = \int_0^{\infty} G_n(f)\, df$$

$$b_2 = (2\pi)^2 \int_0^{\infty} (f - f_c)\, G_n(f)\, df \quad \text{..... 51}$$

$$\rho = \frac{A^2}{2\sigma_n^2} \quad \text{..... 52}$$

ในที่นี้ ρ คือ CNR , Gn(f) คือความเข้มข้นของสเปคตรัลของกำลังข้างหนึ่ง (one-side power spectral density) ของเสียงรบกวนขาเข้า, $f_d = \omega_d/2\pi$ และ erfc(x) คือฟังก์ความผิดพลาดเสริม (complimentary error function)

จะเห็นได้ว่าอัตราการเกิดของเสียงรบกวนแบบคลิ๊คเป็นฟังก์ชันของ ρ, D(t) หรือ f_d และแถบความถี่อาร์เอ็มเอสของเสียงรบกวนขาเข้า Γ

$$h(r) = \begin{cases} 1/T, & 0 \le \Gamma \le T \\ 0, & other \end{cases} \quad \text{..... 58}$$

สมมติว่าช่วงเวลาอินทีเกรท T มีค่าเท่ากับช่วงเวลาของพัลซ์บิตพอดีซึ่งทำให้ไม่เกิดการรบกวนระหว่างบิตที่เวลาของการสุ่มตัวอย่าง

สัญญาณขาออกของเครื่องกรองความถี่แบบอินทีเกรทและดัมพ์ที่ถูกสุ่มตัวอย่างตรงปลายบิต ณ เวลา $t_s = (n+1)\,T$ คือ

$$V_0 = D_0 + u_0 + Z_0 \quad \text{..... 59}$$

สัญญาณดิจิตัลเอฟเอ็มขาออก ณ เวลาสุ่มตัวอย่าง (ดูรูปที่ 2.25 (ก)) จะได้

$$D_0 = \begin{cases} \omega_d & 'mark' \\ -\omega_d & 'space' \end{cases} \quad \text{..... 60}$$

ในที่นี้ ω_d คือ ความถี่เชิงมุมเบี่ยงเบนที่ค่ามากที่สุด
เสียงรบกวนเก๊าเซียนขาออก u_0 จะได้

$$u_0 = a_0 \frac{\cos\theta}{A} - b_0 \frac{\sin\theta}{A} \quad \text{..... 61}$$
$$a_0 = [\,a(t) * h(t)]_{t=t_s}\,,\ [\,b(t) * h(t)\,]_{t=t_s}$$

และเครื่องหมายดอกจันแสดงการคอนโวลูททางแกนเวลา (time convolution)
เสียงรบกวนแบบคลิ๊คขาออกจะได้

$$Z_0 = \frac{1}{T}\int_{nT}^{(n+1)T} \sum_{k=-\infty}^{\infty} 2\pi\delta\,(t - t_k) - \frac{1}{T}\int_{nT}^{(n+1)T} \sum_{l=-\infty}^{\infty} 2\pi\delta\,(t - t_l)\,dt \quad \text{..... 62}$$

ให้ x และ y เป็นจำนวนคลิ๊คบวกและคลิ๊คลบที่เกิดในช่วงเวลาของพัลซ์บิต n จะได้

$$Z_0 = \frac{2\pi\,(x-y)}{T}$$

พรอบอะบิลิตีเดนซิตีฟังก์ชัน (pdf) ของ Z_0

$$p(Z_0) = \sum_{\mu=-\infty}^{\infty} \exp[-(\lambda_+ + \lambda_-)](\frac{\lambda_+}{\lambda_-})^{\mu/2} I_\mu [2\sqrt{\lambda_+\lambda_-}] * \delta(Z_0 - \frac{2\pi\mu}{T}) \quad \text{.... 63}$$

ในที่นี้ $\lambda_+ = N_+T$ และ $\lambda_- = N_-$ คือจำนวนเฉลี่ยของคลิ๊คในช่วงเวลาของพัลซ์บิตใดๆ ที่สนใจ จะเห็นได้ว่า pdf ของผลต่างของกระบวนการพอยซอน x และ y จะไม่ได้กระบวนการพอยซอนอีก

pdf $p(\upsilon 0 \cong D0 + u0)$ จากสมการที่ 60 และ 61 จะได้

$$p(v_0 \Big| D_0 = \pm\omega_d) = \frac{1}{\sqrt{2\pi M_0}} \exp(-\frac{(v_0 \mp \omega_d)^2}{2M_0^2}) \quad \text{..... 64}$$

$$M_0^2 = E[u_0^2] = E[a_0^2] / A^2 \quad \text{..... 65}$$

และ $E(a_0^2)$ คือกำลังของเสียงรบกวน ณ เวลาสุ่มตัวอย่าง

$$E(a_0^2) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \omega^2 G_n(\omega)[H(\omega)]^2 d\omega$$

$$= \frac{1}{2\pi}\int_{-2\pi B}^{2\pi B} \omega^2 \frac{\sigma^2 n}{2B}[\frac{\sin(\omega T/2)}{(\omega T/2)}]^2 d\omega$$

$$2B = 2(\beta+1) f_m$$

$$f_m = 1/2T, \quad \beta = \Delta f / f_m$$

$$E(a_0^2) = \frac{2\delta_n^2}{T^2} f_0(\beta)$$

$$f_0(\beta) = 1 - \frac{\sin\pi(\beta+1)}{\pi(\beta+1)}$$

สมมติว่าสัญญาณมาร์คและสัญญาณสเปสมีโอกาสเกิดเท่ากัน ถ้าเราตั้งเทรสโฮลด์ไว้ที่ 0 โวลท์ เครื่องรับเอฟเอ็มจะตัดสินว่า $D_0 = \omega_d$ ถ้า $V_0 > 0$ และ $D_0 = -\omega_d$ ถ้า $V_0 < 0$

$p(V_0)$ ได้มาจากการคอนโวลูทสมการที่ 63 และสมการที่ 64 ได้ดังนี้

$$p(V_0 \mid D_0 = \pm\omega_d) = \sum_{\mu=-\infty}^{\infty} \exp[-(\lambda_+ + \lambda_-)](\frac{\lambda_+}{\lambda_-})^{\mu 2/} I_\mu[2\sqrt{\lambda_+\lambda_-}]$$

$$\frac{1}{\sqrt{2\pi M_0}} \exp(-\frac{(V_0 \mp \omega_d - \frac{2\pi\mu}{T})^2}{2M_0^2}) \quad \text{..... 66}$$

พรอบอะบิลิตีความผิดพลาดคือ พรอบอะบิลิตีที่ส่งสัญญาณมาร์ค $D_0 = \omega_d$ แต่เครื่องรับจะตัดสินเป็นสัญญาณสเปสเพราะ $V_0 < 0$ และตัดสินเป็นสัญญาณมาร์คเพราะ $V_0 > 0$ ในขณะที่ส่งสัญญาณสเปส $D_0 = -\omega_d$ นั่นคือ

$$\left.\begin{aligned} P_{eM} &= P(V_0 < 0 \mid D_0 = \omega_d)\, dV_0 \\ P_{eS} &= P(V_0 > 0 \mid D_0 = -\omega_d)\, dV_0 \end{aligned}\right\} \quad \text{.....} \quad 67$$

แทนสมการที่ 66 ลงในสมการที่ 67 และอินทีเกรทเทียบกับ V_0 จะได้

$$\left.\begin{matrix} P_{eM} \\ P_{eS} \end{matrix}\right\} = \sum_{\mu=-\infty}^{\infty} \exp[-(\lambda_+ + \lambda_-)] \left(\frac{\lambda_+}{\lambda_-}\right)^{\mu/2} I_\mu[2\sqrt{\lambda_+\lambda_-}]\, \frac{1}{2}\, erfc\left(\frac{(\omega_d \pm \frac{2\pi\mu}{T})}{\sqrt{2M_0}}\right) \quad 68$$

ในที่นี้ผลของเสียงรบกวนต่อเนื่องขาออกที่มีต่อพรอบอะบิลิตีความผิดพลาดนั้นแสดงอยู่ในรูปของฟังก์ชันความผิดพลาดเสริมซึ่งมีส่วนที่ถูกย้ายไป $\pm 2\pi\mu / T$ ซึ่งเป็นผลสืบเนื่องมาจากการคอนโวลูทกับสัญญาณคลิ๊ค เป็นที่น่าสังเกตว่าจำนวนเฉลี่ยของคลิ๊คลบที่เกิดในช่วงเวลาของพัลซ์บิตบวก T เท่ากับจำนวนเฉลี่ยของคลิ๊คบวกที่เกิดในช่วงเวลลาของพัลซ์บิตลบ T ดังนั้นถ้าแทน μ ของ P_{es} ในสมการที่ 68 ด้วยค่า $-\mu$ ผลที่ได้จะเท่ากับ P_{eM} ฉะนั้นสำหรับโอกาสการเกิดของสัญญาณมาร์คและสัญญาณสเปสที่เท่ากัน พรอบอะบิลิตีความผิดพลาด P_e จะได้

$$P_e = (1/2)\, P_{eM} + (1/2)\, P_{es} = P_{eM} \quad \text{.....} \quad 69$$

แทนสมการที่ 69 ในสมการที่ 68 พรอบอะบิลิตความผิดพลาด

$$P_e = P_{eM} = \frac{1}{2} \sum_{\mu=-\infty}^{\infty} \exp[-(\lambda_+ + \lambda_-)] \left(\frac{\lambda_+}{\lambda_-}\right)^{\mu/2} I_\mu[2\sqrt{\lambda_+\lambda_-}]\, \frac{1}{2}\, erfc\left(\frac{(\omega_d \pm \frac{2\pi\mu}{T})}{\sqrt{2M_0}}\right)$$

ดูค่อนข้างยุ่งยากทางคณิตศาสตร์ แต่ความเป็นจริงแล้วสมการนี้แก้ได้ค่อนข้างง่าย ถ้าเราใช้หลักความจริงที่ว่า สัญญาณมาร์คคลิ๊คส่วนมากจะเกิดในทางลบ ดังนั้นเราสามารถตัดคลิ๊คบวกออกได้สมการที่ 48 และสมการที่ 49 จะถูกทอนให้เหลือ

$$N_+ \ll N_-$$
$$N_- \approx f_d \exp(-\rho) \quad \text{..... 71}$$
$$\lambda_+ = N_+ T \ll \lambda_-$$
$$\lambda_- = N_- T \approx (\beta/2) \exp(-\rho)$$
$$\beta = 2 f_d T \quad \text{..... 72}$$

นอกจากนี้ถ้าเราพิจารณาที่อัตราส่วนของกำลังของตัวพาต่อกำลังของเสียงรบกวน (CNR) ที่ค่ามากพอควร โอกาสที่คลิ๊กบวกและคลิ๊กลบจะเกิดมากกว่าหนึ่งคลิ๊กในหนึ่งช่วงเวลาของพัลซ์บิต T น้อยมาก นั่นคือ λ_+ และ λ_- << 1 จะได้

$$I\mu[2\sqrt{\lambda_+\lambda_-}] \approx (\lambda_+\lambda)^{|\mu|/2}, \quad \exp[-(\lambda_+\lambda_-)] \approx 1 \quad \text{..... 73}$$

สมมติว่าใน 1 ช่วงเวลาของพัลซ์บิต T จะมีคลิ๊กเกดมากที่สุดไม่เกิน 1 คลิ๊ก และแทนสมการที่ 71, 72 และ 73 ในสมการที่ 74 จะได้

$$P_e = \frac{1}{2}\sum_{\mu=0}^{1} \left(\frac{\beta}{2}\exp\left(-\frac{A^2}{2\sigma_n^2}\right)\right)^{\mu} erfc\left(\frac{(\omega_d - \frac{2\pi\mu}{T})A}{\sqrt{2a_0^2}}\right) \quad \text{..... 74}$$

$$P_e = \frac{1}{2} erfc\left(\frac{\pi\beta\rho_0^{1/2}}{\sqrt{2f_0(\beta)}}\right) + \frac{\beta}{4}\exp(-\rho_0)\, erfc\left(\frac{\pi\beta(1-\frac{2}{\beta})\rho_0^{1/2}}{\sqrt{2f_0(\beta)}}\right) \quad \text{..... 75}$$
$$\beta = \omega_d/\pi/T = \omega_d/\omega_m$$

ในที่นี้ β คือดัชนีโมดูเลชัน (modulation index)

fm = T/2 คือความถี่ของสัญญาณและแวเรียนซ์ของเสียงรบกวน

$$a_0^2 = \frac{2\sigma_n^2}{T^2} f_0(\beta)$$

เป็นที่น่าสังเกตว่าอัตราความผิดพลาด P_e ในสมการที่ 75 นั้นเป็นความผิดพลาดที่เกิดจากเสียงรบกวนแบบเก๊าเซียนที่ต่อเนื่องอย่างเดียว (เทอมแรกของสมการที่ 75 ที่ค่า μ = 0) บวกกับความผิดพลาดที่เกดจากเสียงรบกวนคลิ๊กและเสียงรบกวนเก๊าเซียนที่ต่อเนื่อง (เทอมที่สองของสมการที่ 75 ที่ค่า μ = 1) อัตราความผิดพลาดนี้แสดงในรูปที่ 2.25 (ข) เป็นฟังก์ชันของดัชนีโมดูเลชัน β ที่อัตราส่วน

ระหว่างพลังงานสัญญาณต่อกำลังสเปกตรัลของเสียงรบกวนเท่ากับ 13.4 dB ดัชนีโมดูเลชันที่อ๊อพติมัม ซึ่งให้อัตราความผิดพลาดน้อยที่สุดมีค่าประมาณ 0.6

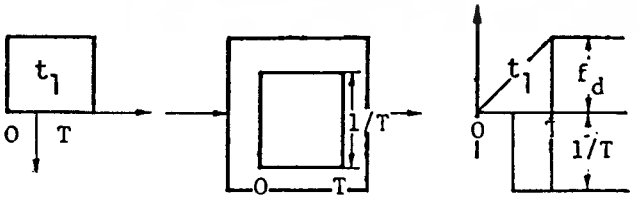
ต่อไปนี้เราจะอธิบายกลไกของความผิดพลาดที่เกิดจากคลิ๊กในช่วงเวลาของพัลซ์บิต T และผลของมันที่มีต่ออัตราความผิดพลาดโดยส่วนรวม ดังแสดงในรูปที่ 2.24 พัลซ์บิตบวกที่ถูกดีโมดูเลทแล้วและคลิ๊กลบหลังจากผ่านเครื่องกรองความถี่แบบอินทีเกรทและดัมพ์จะมีขนาด f_d และ 1/T ที่ขาออกของเครื่องกรองความถี่ ณ เวลาสุ่มตัวอย่างตามลำดับ เนื่องจากความสูงของพัลซ์บิตบวกที่ผ่านเครื่องกรองความถี่เป็นฟังก์ชันของดัชนีโมดูเลชัน $\beta = \omega_d/\pi/T$ ในกรณีพิเศษที่ค่า $\beta = 2$ นั่นคือ $f_d = 1/T$ ขนาดของสัญญาณดิจิตัลเอฟเอ็มที่ผ่านเครื่องกรองความถี่และขนาดของคลิ๊กจะมีค่าเท่ากัน ดังนั้น ณ เวลาสุ่มตัวอย่างขนาดของสัญญาณดิจิตัลเอฟเอ็มที่ผ่านเครื่องกรองความถี่มีแนวโน้มว่าจะถูกหักล้างโดยขนาดของคลิ๊กและเกิดปรากฏการณ์ดังนี้

1. สำหรับ β เนื่องจากอัตราการเกิดของคลิ๊กเป็นฟังก์ชันของ β การเพิ่มดัชนีโมดูเลชันก็เท่ากับการเพิ่มจำนวนคลิ๊กด้วย ถ้าเราพิจารณาถึงผลของการเกิดคลิ๊กมากกว่า 1 คลิ๊กแล้ว ความผิดพลาดที่เกิดขึ้นจะเกิดจากการกระทำร่วมกันของจำนวนคลิ๊กที่เพิ่มขึ้นและการกระเพื่อมของเสียงรบกวนเก๊าเซียนพร้อมกัน

2. ที่ค่าดัชนีโมดูเลชันต่ำ $\beta < 0.6$ เนื่องจาก λ_1 หรือ λ_2 เป็นฟังก์ชันของ β โอกาสที่คลิ๊กเกิดในช่วงเวลาของพัลซ์บิต T 1 บิตนั้นมีน้อยมากจนเกือบไม่มี ดังนั้นเสียงรบกวนเก๊าเซียนจึงเป็นตัวสำคัญที่ทำให้เกิดความผิดพลาดขึ้นในย่านที่ $\beta > 0.6$ นี้

3. เมื่อ β มีค่ามากกว่า 0.6 เช่น $\beta = 1$ ขนาดของคลิ๊กจะมีค่าเป็น 2 เท่าของขนาดของสัญญาณ ณ เวลาสุ่มตัวอย่าง ดังนั้นเราสามารถคาดหมายได้ว่าความผิดพลาดที่เกิดขึ้นนั้นเกิดจากคลิ๊กเพียงอย่างเดียว

4. สำหรับ β ที่มีค่าใกล้เคียงกับ 0.6 และ 2 ซึ่งเป็นย่านที่มีการเปลี่ยนแปลง คือ เปลี่ยนจากความผิดพลาดที่เกิดจากเสียงรบกวนเก๊าเซียนมาเป็นความผิดพลาดที่เกิดจากคลิ๊ก ในกรณีที่ β มีค่าใกล้เคียงกับ 0.6 และมีผลกลับกัน กรณีที่ β มีค่าใกล้เคียงกับ 2 ระหว่างค่า $\beta = 0.6$ และ $\beta = 2$ นั่นคือ $0.6 < \beta < 2$ ก็คือกรณีในหัวข้อที่ 3 ที่ได้กล่าวมาแล้ว



รูปที่ 2.24 กลไกของความผิดพลาดที่เกิดจากคลิ๊ก

ความผิดพลาดที่เกิดใน 3 ย่านนี้คือ $\beta < .06$, $0.6 < \beta < 2$ และ $\beta > 2$ สามารถเห็นได้ชัดในรูปที่ 2.25 (ข) ดังนั้น เราสามารถสรุปได้ว่าในการใช้เครื่องรับเอฟเอ็มธรรมดาเป็นเครื่องรับสัญญาณที่ตั้งฉากกัน (orthogonal signal) ในกรณีของเราคือเอฟเอสเค (FSK) การเพิ่มค่า β หรือพูด

อีกในหนึ่งว่าการเพิ่มแถบความถี่ของเครื่องกรองความถี่กลาง อาจจะไม่ต้องเพิ่มพิสัยความสามารถของระบบก็ได้ ทั้งนี้เพราะแถบความถี่ของเครื่องกรองความถี่กลางเพิ่มขึ้นก็จะรับเสียงรบกวนมากขึ้น

2.5 การเปรียบเทียบการทำงานของระบบ

ในหัวข้อนี้ จะเปรียบเทียบพรอบอะบิลิตีความผิดพลาดของระบบดิจิตัลเอฟเอ็มที่ใช้เครื่องดิสคริมิเนเตอร์ในการรับสัญญาณกับระบบเอฟเอสเคไบนารีอื่น

ในการเปรียบเทียบกับระบบเอฟเอสเคไบนารีอื่น เราควรใช้พารามิเตอร์อัตราส่วนพลังงานของสัญญาณในช่วงพัลซ์ 1 บิต T ต่อความเข้มข้นของกำลังของเสียงรบกวน ฉะนั้นถ้าเราใช้กฎของคาร์สันในการคำนวณแถบความถี่ ต้องมีแฟคเตอร์การแปลงในการคำนวณอัตราความผิดพลาดในสมการที่ 75 ดังนี้

$$\rho_0 = (E/N_0)(1/(\beta+1)) \qquad \text{..... 76}$$

พรอบอะบิลิตีความผิดพลาดในการใช้เครื่องกรองความถี่แบบแม็ช เพื่อรับสัญญาณเอฟเอสเคไบนารีคือ

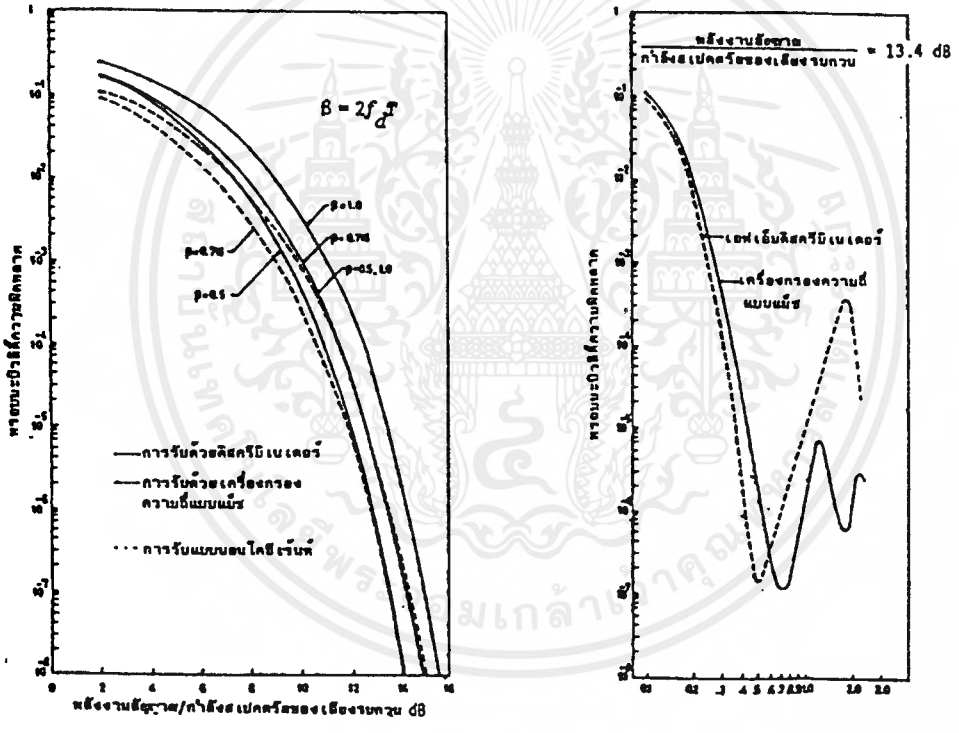
$$P_e = (\frac{1}{2})\, erfc\left\{\sqrt{0.5\,(E/N_0)\,(1-(\sin 2\pi\beta)/(2\pi\beta))}\ \right\} \qquad \text{..... 77}$$

ในที่นี้ค่าอ๊อพติมัม $\beta = 0.715$ เครื่องกรองความถี่แบบแม็ชจะเป็นเครื่องรับสัญญาณแบบโคฮีเร็นท์ (coherent detector) เพราะมีค่าพรอบอะบิลิตีความผิดพลาดต่ำสุด ส่วนพรอบอะบิลิตีความผิดพลาดสำหรับนอนโคฮีเร็นท์เอฟเอสเคคือ

$$P_e = \frac{1}{2}\exp(-\frac{E}{2N_0}) \qquad \text{..... 78}$$

รูปที่ 2.25 (ข) จะให้การเปรียบเทียบพรอบอะบิลิตีของเครื่องดิสคริมิเนเตอร์เอฟเอ็มเครื่องกรองความถี่แบบแม็ช และเครื่องรับแบบนอนโคฮีเรนท์เอฟเอสเคที่ค่าดัชนีโมดูเลชัน = 0.5, 0.715 และ 1 จะเห็นได้ว่าการรับสัญญาณแบบเครื่องดิสคริมิเนเตอร์นั้นจะให้พรอบอะบิลิตีน้อยที่สุดที่ดัชนีโมดูเลชันเท่ากับ 0.5 และมีพรอบอะบิลิตีความผิดพลาดเข้าใกล้โคฮีเรนท์เอฟเอสเคในย่านที่ค่า E/N_0 สูง ในขณะที่ถ้ารับสัญญาณที่มีดัชนีโมดูเลชันเท่ากับ 1 จะมีพรอบอะบิลิตีความผิดพลาดเกือบเท่ากับนอนโคฮีเรนท์เอฟเอสเค ในกรณีนี้นอนโคฮีเรนท์เอฟเอสเคและโคฮีเรนท์เอฟเอสเค (ที่ใช้เครื่องกรองความถี่แบบแม็ชที่มีดัชนีโมดูเลชัน $\beta = 0.715$) สามารถคิดเสมือนหนึ่งว่าเป็นขอบเขตบนและขอบเขตล่างของสัญญาณดิจิตัลเอฟเอ็มแบบแถบความถี่แคบที่มีดัชนีโมดูเลชันระหว่าง 0.5 ถึง 1 รูปที่ 2.25 จะแสดงพรอบอะบิลิตีความผิดพลาดเปรียบเทียบกับดัชนีโมดูเลชันสำหรับเครื่องดิสคริมิเนเตอร์เอฟเอ็มและเครื่องรับสัญญาณที่ใช้เครื่องกรองความถี่แบบแม็ช พรอบอะบิลิตีความผิดพลาดน้อยที่สุดสำหรับเครื่องดิสคริมิเนเตอร์เอฟ

เอ็มที่เกิดดัชนีโมดูเลชัน เท่ากับ 0.5 และสำหรับเครื่องรับสัญญาณที่ใช้เครื่องกรองความถี่แบบแม็ชที่ดัชนีโมดูเลชันเท่ากับ 0.715 (β = 0.715) ในทางปฏิบัติเราเปรียบเทียบพรอบอะบิลิตีความผิดพลาดที่ค่า E/N0 กำหนดโดย CCITT ซึ่งเท่ากับ 13.4 dB จะเห็นได้ว่าเครื่องดิสคริมิเนเตอร์เอฟเอ็มไม่เพียงแต่สนองความต้องการของ CCITT ที่ค่าพรอบอะบิลิตีความผิดพลาดเท่ากับ 106 ที่ E/N0 = 13.4 dB โดยมีดัชนีโมดูเลชัน = 0.715 แต่ยังมีค่าพรอบอะบิลิตีความผิดพลาดต่ำกว่าถ้าใช้ดัชนีโมดูเลชันเท่ากับ 0.5 (β = 0.5) เครื่องกรองความถี่แบบแม็ชมีวิสัยความสามารถเหนือกว่าเครื่องดิสคริมิเนเตอร์เอฟเอ็มเฉาะในย่านเสียงรบกวนคลิ๊ค สำหรับค่าโมดูเลชันที่ต่ำกว่า 0.6 หรือในย่านเสียงรบกวนคลิ๊ค เครื่องดิสคริมิเนเตอร์จะมีวิสัยความสามารถดีกว่าเครื่องกรองความถี่แบบแม็ชเล็กน้อย



(ก) การเปรียบเทียบอัตราความผิดพลาดเทียบกับระบบเอฟเอสเคแบบไบนารี

(ข) การเปรียบเทียบเครื่องดิสคริมิเนเตอร์เอฟเอ็มและเครื่องกรองความถี่แบบแม็ช

รูปที่ 2.25

บทที่ 3
การสร้างและการคำนวณ

3.1 บล๊อกไดอะแกรม

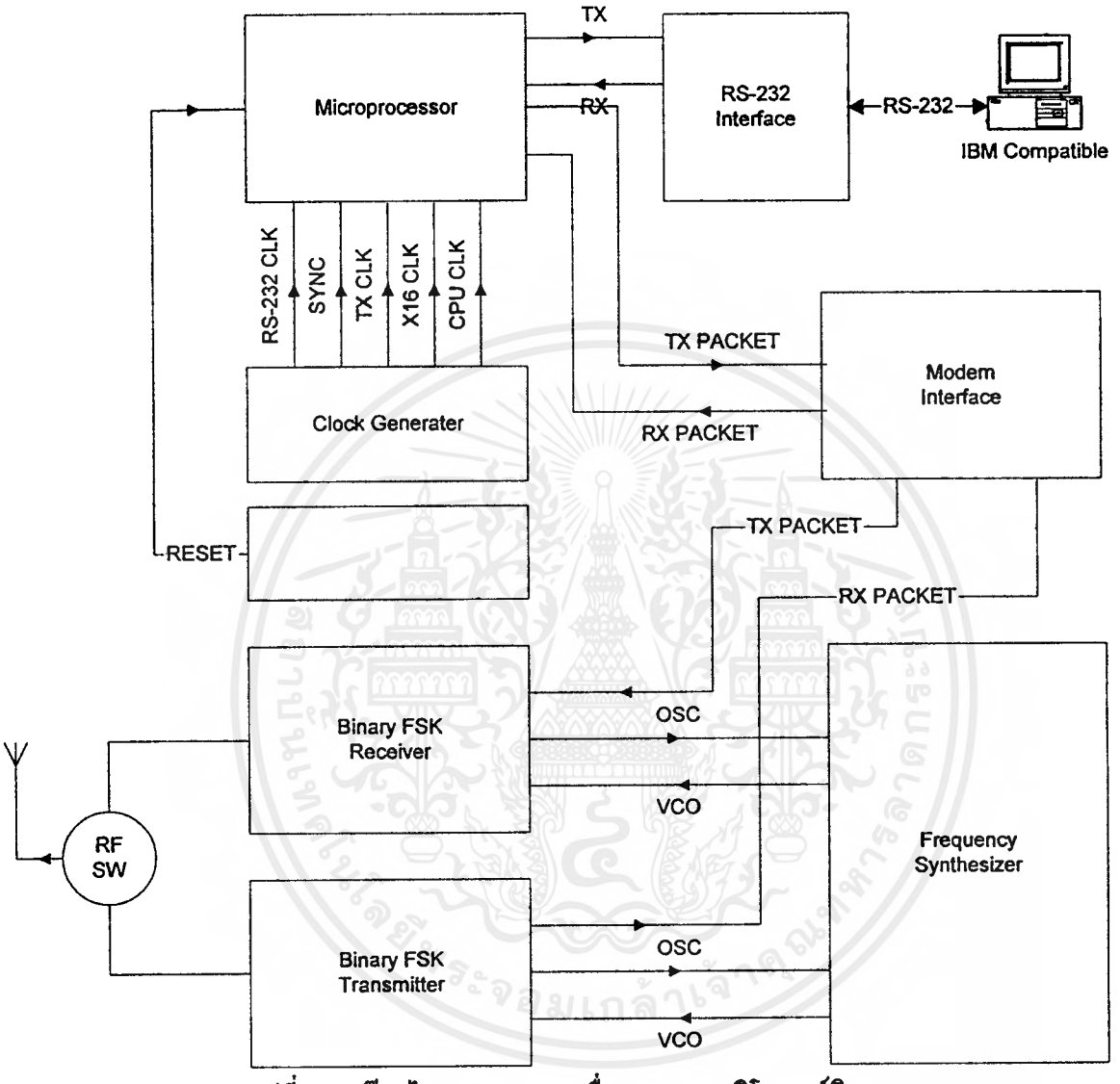
จากรูปที่ 3.1 แสดงบล๊อกไดอะแกรมของเครื่องแพกเกตเรดิโอเทอร์มินอลที่สร้างขึ้นในโครงงานนี้ วงจรควบคุมและประมวลผลทำหน้าที่ควบคุมจัดการกระบวนการสื่อสารให้เป็นไปตามข้อกำหนดหรือโปรโตรคอล ในโครงงานนี้ใช้เฟิร์มแวร์ TNC-2 ซึ่งเป็นโปรแกรมสำหรับควบคุมการบวนการสื่อสารแบบอะโลฮา โดยใช้โปรโตรคอล AX.25 โปรแกรมนี้ทำงานบนไมโครโปรเซสเซอร์เบอร์ Z80 ร่วมกับชิปสื่อสารอเนกประสงค์เบอร์ Z80/SIO สัญญาณนาฬิกาที่ใช้กำหนดความเร็วของการรับส่งข้อมูลสร้างฐานเวลาสำหรับแพกเกตไทม์ และสัญญาณนาฬิกาสำหรับระบบไมโครโปรเซสเซอร์ ทั้งหมดถูกสร้างมาจากวงจรสร้างสัญญาณนาฬิกาที่สามารถสร้างสัญญาณนาฬิกาได้หลายๆ ค่าเวลาได้ ข้อมูลหรือคำสั่งที่ส่งมาจากเครื่องโฮสคอมพิวเตอร์ถูกส่งออกมาตามมาตฐาน RS-232 ผ่านวงจรอินเตอร์เฟสพอร์ตอนุกรมเพื่อแปลงระดับสัญญาณให้สามารถใช้กับวงจรควบคุมและประมวลผลได้ สัญญาณข้อมูลแต่ละแพกเกตที่ถูกส่งออกไปจากวงจรควบคุมและประมวลผลจะผ่านเข้าวงจรโมเด็มอินเตอร์เฟสเพื่อทำการเข้ารหัสสัญญาณแบบ NRZI (Non Return to Zero Inverse) เพื่อลดส่วนประกอบไฟตรงป้องกันการจางหายของข้อมูล สำหรับสัญญาณแพกเกตข้อมูลที่รับเข้ามานั้นก่อนเข้าวงจรควบคุมและประมวลผลก็จะถูกวงจรโมเด็มอินเตอร์เฟสแปลงสัญญาณ NRZI กลับเป็นสัญญาณข้อมูลปกติ นอกจากนี้ยังทำการแยกสัญญาณนาฬิกาออกจากสัญญาณแพกเกต ในส่วนของโมเด็มเป็นอาร์เอฟโมเด็มแบบไบนารีเอฟเอสเค ทำงานในโหมดซิมเพล็กหรือผลัดรับผลัดส่งโดยรับคำสั่งมาจากวงจรควบคุมและประมวลผล ความถี่ของอาร์เอฟโมเด็มควบคุมโดยวงจรสังเคราะห์ความถี่แบบเฟสล๊อกลูป

รูปที่ 3.2 แสดงบล๊อกไดอะแกรมของเครื่องรับวิทยุแบบไบนารีเอฟเอสเค สัญญาณวิทยุที่รับเข้ามาทางสายอากาศจะถูกขยายด้วยวงจรอาร์เอฟแอมป์หลังจากนั้นก็จะถูกส่งเข้าไปผสมสัญญาณกับสัญญาณความถี่ที่สูงกว่าสัญญาณวิทยุอยู่ 10.7 เมกะเฮิรตซ์จากวงจรออสซิลเลเตอร์ชนิดควบคุมความถี่ด้วยแรงดันไฟฟ้าหรือวงจรวีซีโอซึ่งความถี่ของวงจรวีซีโอนี้จะถูกควบคุมโดยวงจรเฟสล๊อกลูปอีกชั้นหนึ่ง สัญญาณผลต่างจะถูกส่งผ่านเข้าวงจรแบนด์พาสฟิลเตอร์ 10.7 เมกะเฮิรตซ์ หลังจากนั้นสัญญาณดังกล่าวจะถูกผสมกับสัญญาณความถี่ 10.245 เมกะเฮิรตซ์จากวงจรออสซิลเลเตอร์ สัญญาณผลต่าง 455 กิโลเฮิรตซ์จะผ่านวงจรแบนด์พาสฟิลเตอร์ 455 กิโลเฮิรตซ์ไปวงจรดีเทกเตอร์แบบควอดดราเจอร์ได้เป็นสัญญาณไฟฟ้าที่เปลี่ยนแปลงไปตามความถี่ของสัญญาณวิทยุที่เปลี่ยนแปลงไป สัญญาณไฟฟ้าจะถูกจัดระดับสัญญาณโดยวงจรตอมพาราเตอร์แบบจุดตัดผ่านศูนย์ ให้เป็นระดับสัญญาณข้อมูลดิจิตัลหนึ่งหรือศูนย์ (“1” หรือ “0”)

รูปที่ 3.3 แสดงบล๊อกไดอะแกรมของเครื่องส่งวิทยุแบบไบนารีเอฟเอสเค วงจรวีซีโอจะถูกควบคุมความถี่ให้คงที่ที่ความถี่ใช้งานโดยวงจรเฟสล๊อกลูป สัญญาณข้อมูลดิจิตัลจะทำให้ความถี่ของวงจรวีซีโอเปลี่ยนแปลงไป ทำให้เกิดสัญญาณไบนารีเอฟเอสเค ดังที่ได้อธิบายในหัวข้อดิจิตัลเอฟเอ็ม ในบทที่ 2

รูปที่ 3.4 แสดงบล๊อกไดอะแกรมของวงจรเฟสล๊อกลูปแบบดูอัลโมดูลัสซึ่งเป็นหัวใจการงานของ

วงจรสังเคราะห์ความถี่แบบเฟสล็อกลูป ตัวเคาท์เตอร์เสริมจะเป็นตัวบังคับให้วงจรหาร 64/65 ทำการหารด้วยตัวหารตัวใดตัวหนึ่งระหว่าง 64 หรือ 65 เช่นหากตัวหารนี้กำลังทำการหารด้วย 64 เมื่อเคาท์เตอร์เสริม

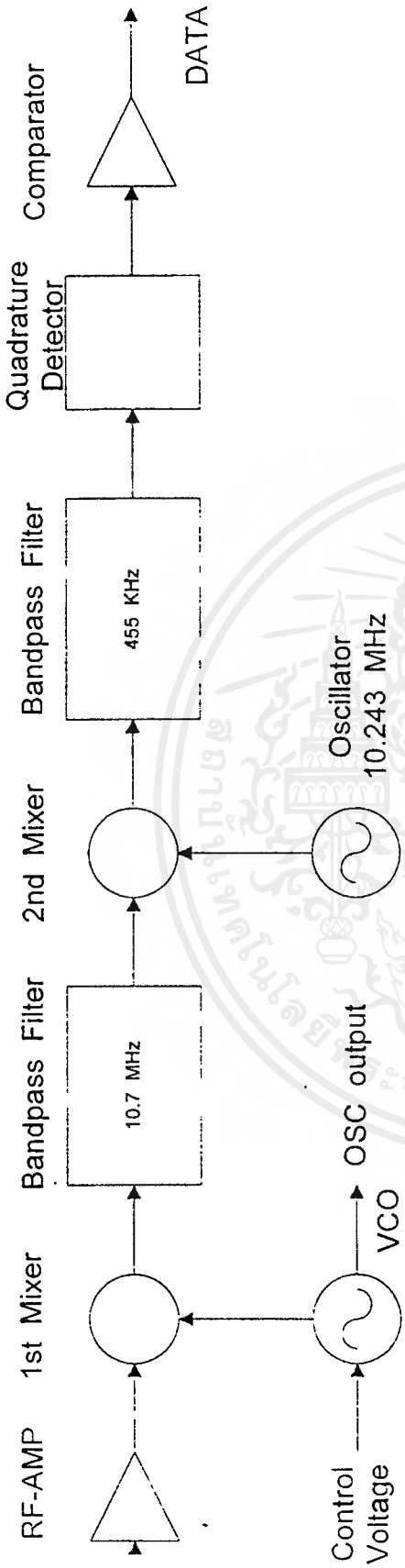


รูปที่ 3.1 บล็อกไดอะแกรมของเครื่องแพกเกตเรดิโอเทอร์มินอล

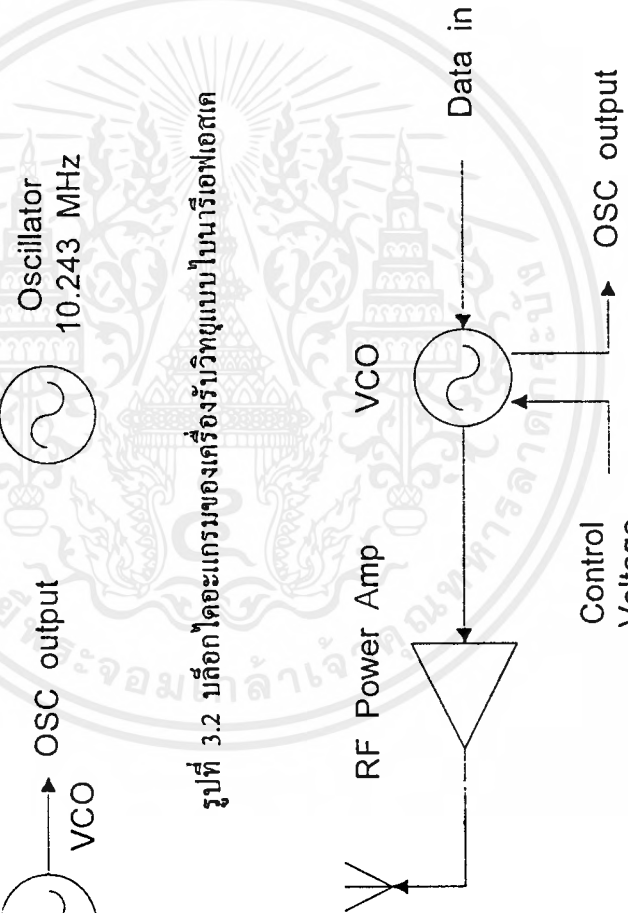
หยุดนับ ก็จะสั่งให้วงจรหาร 64/65 ทำการหารด้วย 65 ในทางกลับกันก็เช่นเดียวกัน ส่วนตัวเคาท์เตอร์หลักก็ขะทำการนับลงจากค่าที่ได้ตั้งไว้ เมื่อเคาท์เตอร์หลักและเคาท์เตอร์เสริมนับถึงศูนย์เมื่อใด วงจรเคาท์เตอร์ก็จะทำการโหลดค่าใส่เคาท์เตอร์ทั้งสองใหม่และทำการนับลงเช่นนี้เป็นวงรอบไป เนื่องจากวงจรเคาท์เตอร์เสริมจะต้องนับถึงศูนย์ก่อนดังนั้นค่าของตัวเคาท์เตอร์เสริมจะต้องน้อยกว่าเคาท์เตอร์หลักเสมอ ความถี่ที่สังเคราะห์ได้จากวงจรเฟสลือกลูปนี้จะมีค่าเป็น

$$Fsynth = Fref\ (64M + A) \quad \text{..... 79}$$

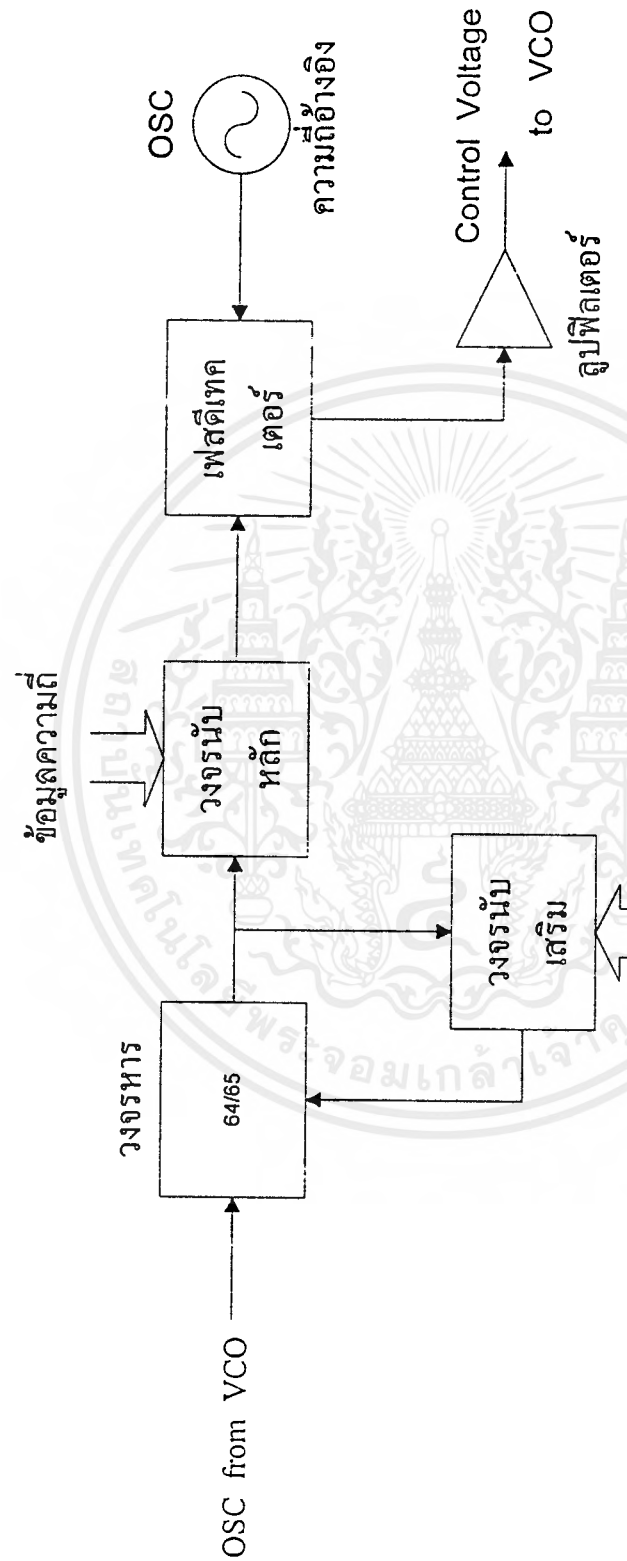
โดยที่ Fsynth คือความถี่ที่ได้จากวงจรสังเคราะห์ความถี่หรือจากวีซีโอนั่นเอง Fref คือความถี่อ้างอิงจากวงจรสร้างความถี่อ้างอิง ส่วน M และ A คือค่าตัวหารของวงจรนับหลักและวงจรนับเสริมตามลำดับ



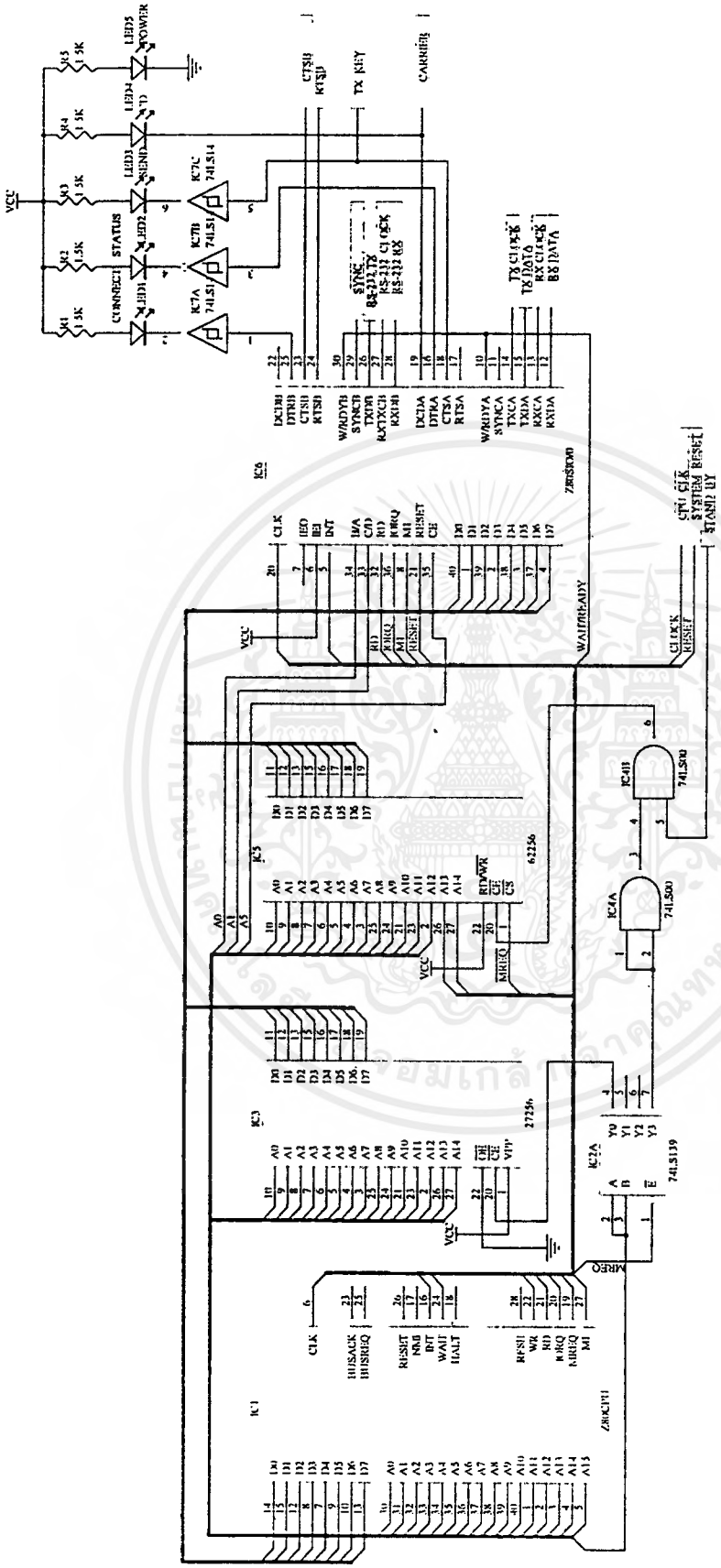
รูปที่ 3.2 บล็อกไดอะแกรมของเครื่องรับวิทยุแบบไบนารีเอฟเอสเค



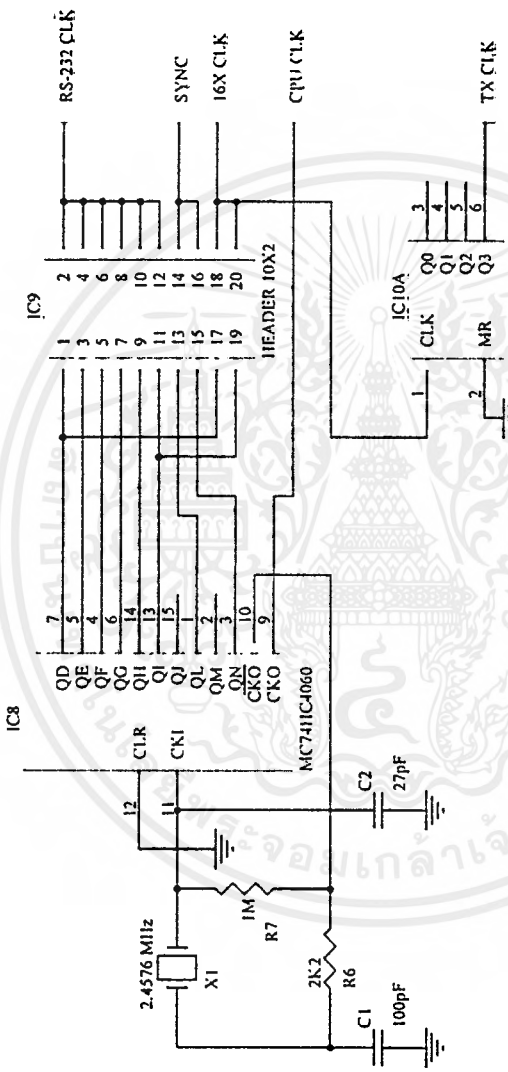
รูปที่ 3.3 บล็อกไดอะแกรมของเครื่องส่งวิทยุแบบไบนารีเอฟเอสเค



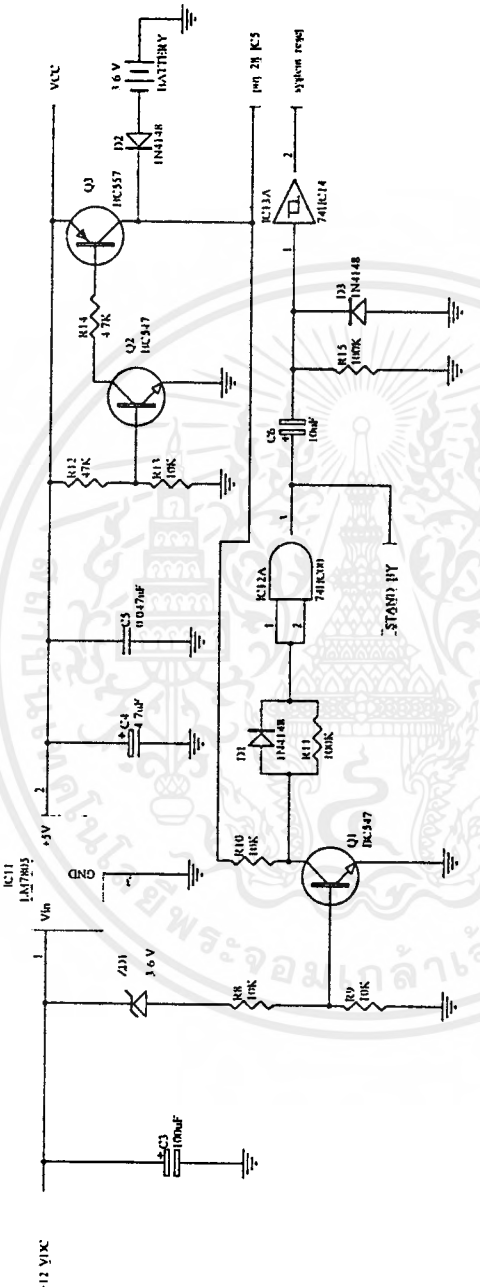
รูปที่ 3.4 บล็อกไดอะแกรมของวงจรเฟสล็อกลูปแบบดูอัลโมดูลัส



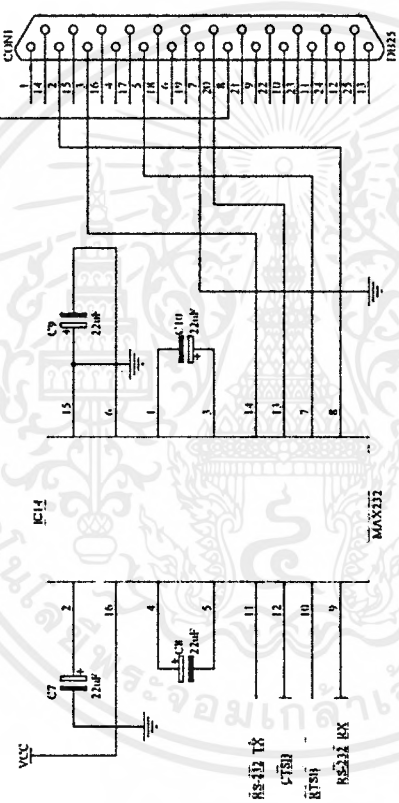
รูปที่ 3.5 วงจรควบคุมการทำงานของระบบรับ-ส่งแพกเกตข้อมูล



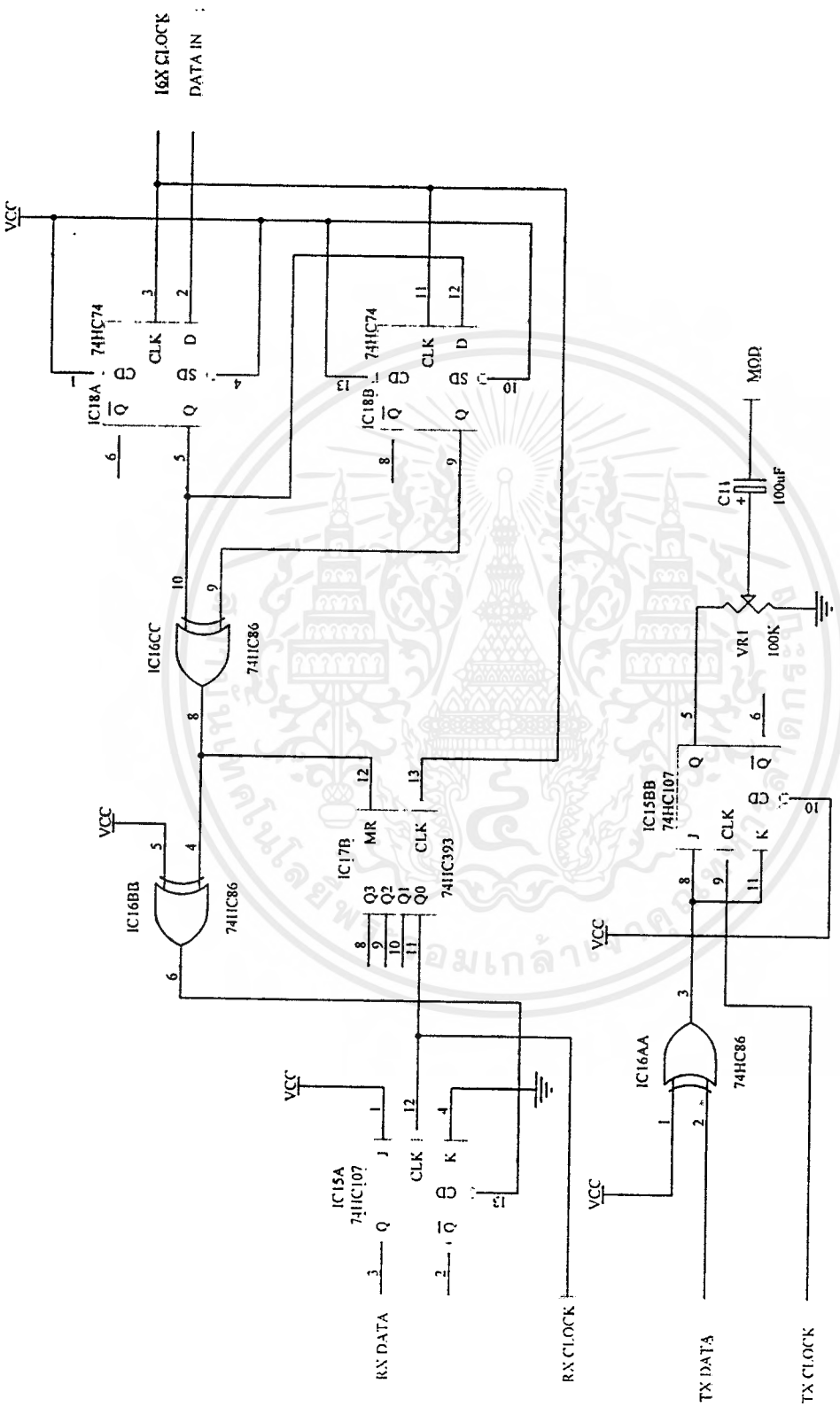
รูปที่ 3.6 วงจรสร้างสัญญาณนาฬิกาอ้างอิง



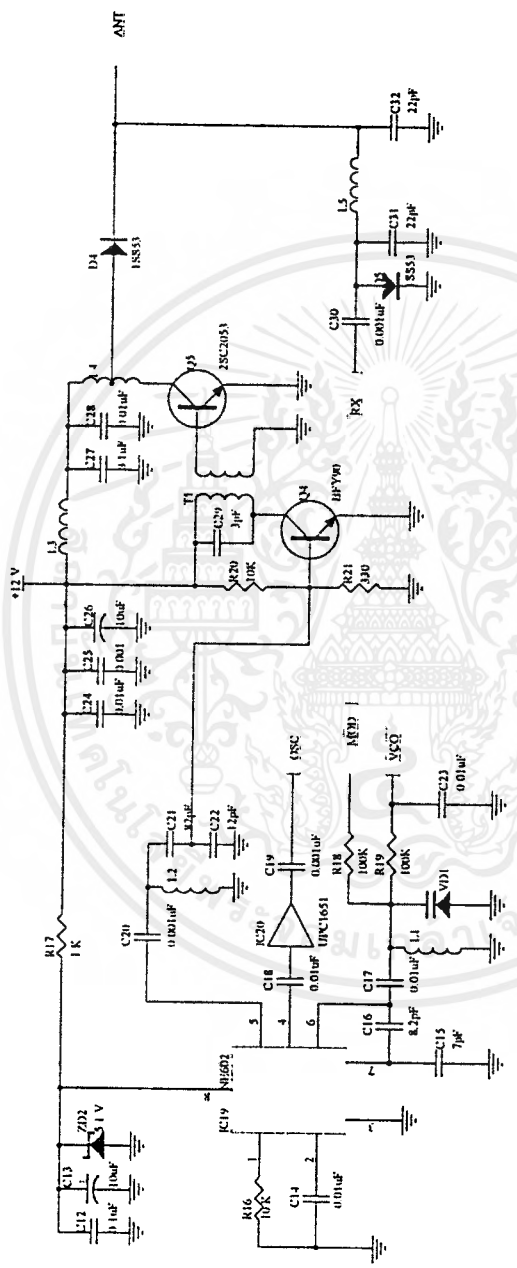
รูปที่ 3.7 วงจรภาคจ่ายไฟ



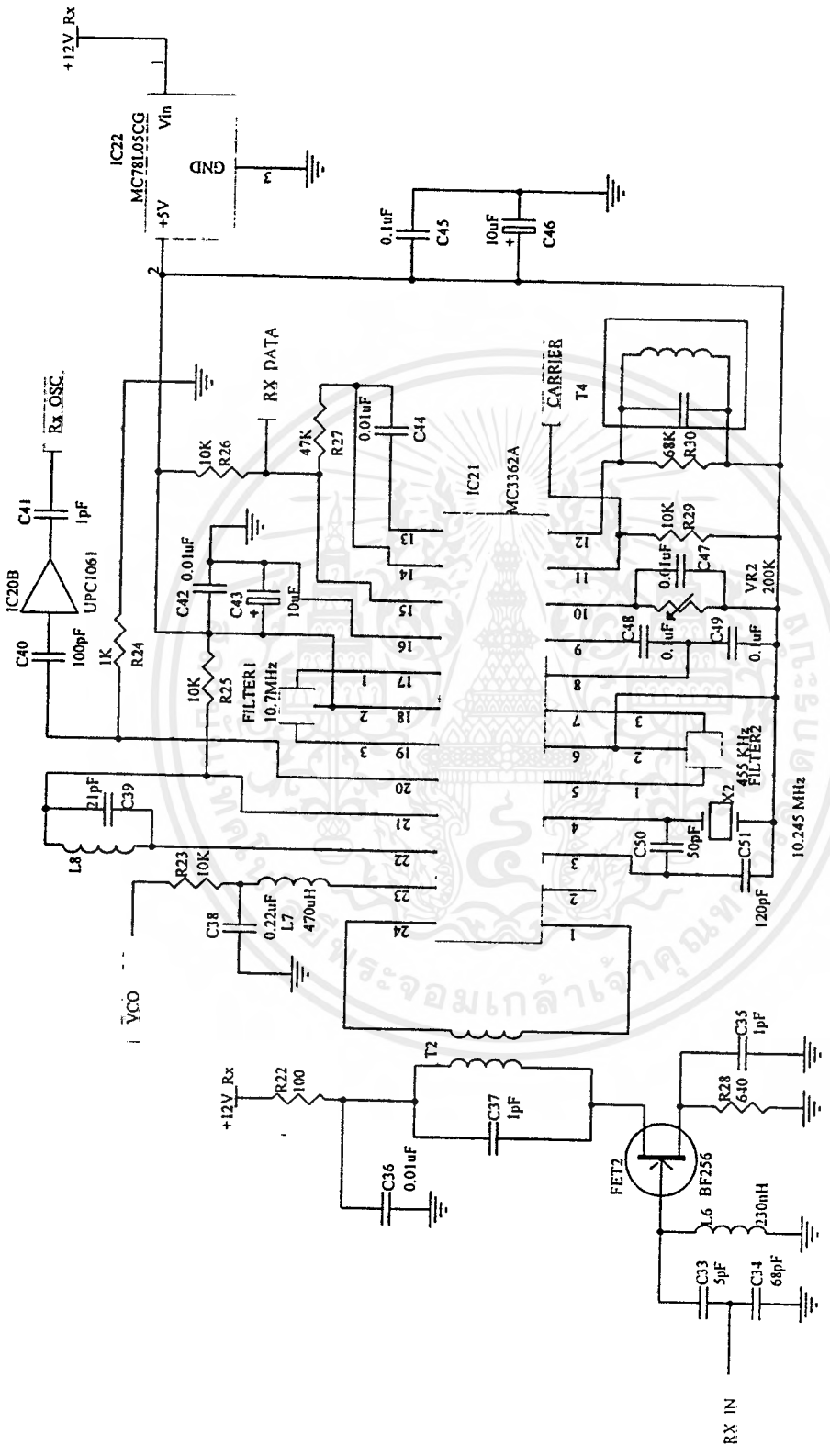
รูปที่ 3.8 วงจรอินเตอร์เฟสด้าน RS-232



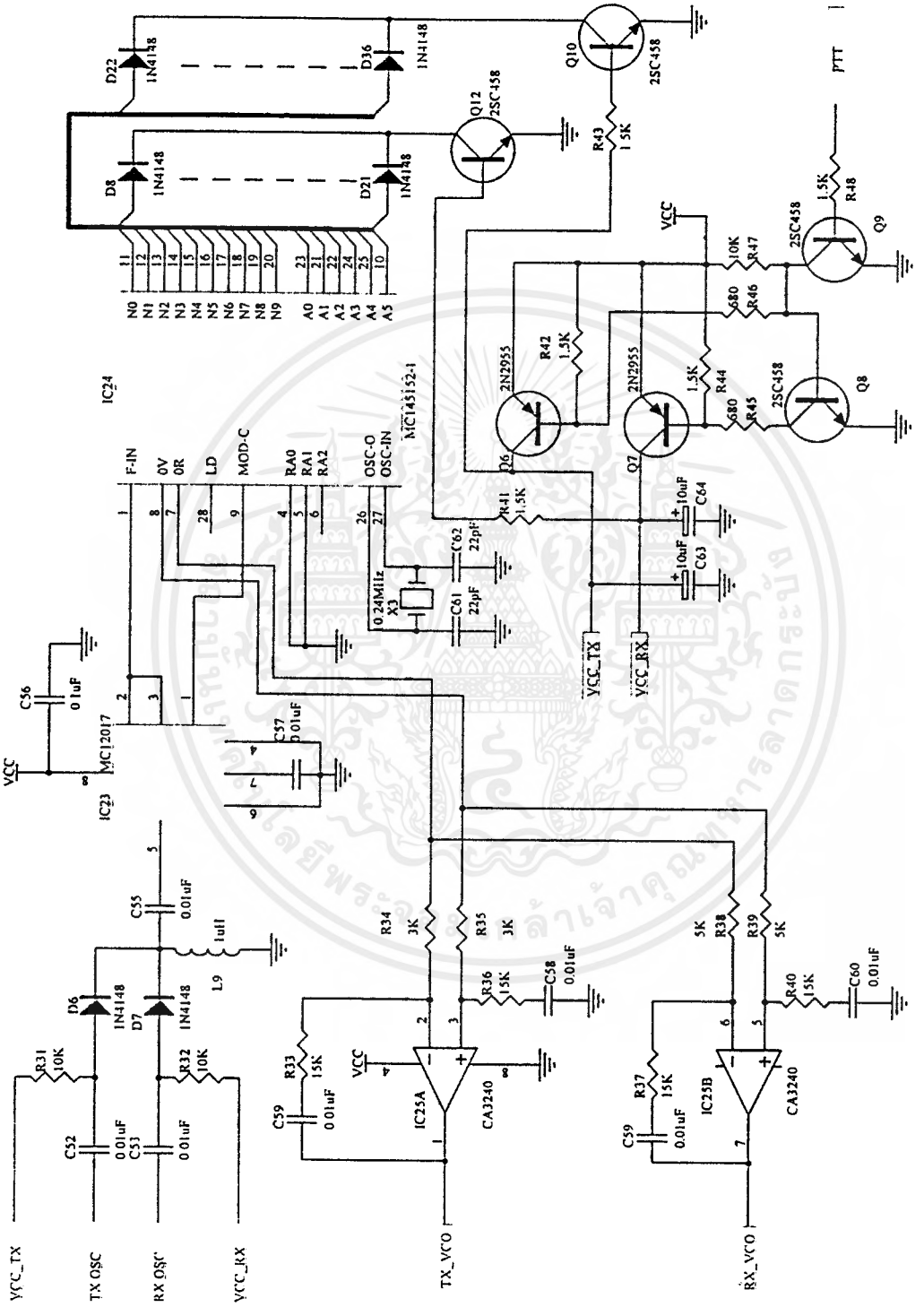
รูปที่ 3.9 วงจรเข้ารหัสและถอดรหัสข้อมูล



รูปที่ 3.10 วงจรภาคส่งวิทยุแบบไบเรียฟอสเค



รูปที่ 3.11 วงจรภาครับวิทยุแบบใช้งานวีเอฟเอสเค



รูปที่ 3.12 วงจรสังเคราะความถี่ เฟสล็อกลูป

3.2 การทำงานของวงจร

จากรูปที่ 3.5 เป็นวงจรควบคุมการทำงานของระบบรับส่งแพกเกตข้อมูลใช้ไมโครโปรเซสเซอร์เบอร์ Z-80 ทำงานร่วมกับหน่วยความจำถาวร (ROM) เบอร์ 27256 และหน่วยความจำชั่วคราว (RAM) เบอร์ 62256 เป็นวงจรไมโครคอมพิวเตอร์ขนาดเล็ก ซึ่งติดตั้งโปรแกรม TNC-2 และเชื่อมต่อกับหน่วยสื่อสารแบบเอนกประสงค์เบอร์ Z-80/SIO ทั้งหมดทำหน้าที่เป็นหน่วยควบคุมการสื่อสารระหว่างเครื่องเทอร์มินอลแพกเกตเรดิโอกับโฮสคอมพิวเตอร์ และระหว่างเครื่องเทอร์มินอลด้วยกันเอง ข้อมูลที่รับส่งมาจากเครื่องโฮสคอมพิวเตอร์ผ่านพอร์ตอนุกรมเข้าวงจรอินเตอร์เฟสแบบ RS-232 ในรูปที่ 3.8 ผ่านเข้าจุดต่อ RS-232 TX และ RS-232 RX ในวงจรรูปที่ 3.5 ข้อมูลที่รับส่งระหว่างเครื่องเทอร์มินอลจะผ่านเข้าจุดต่อ TX DATA และ RX DATA ของวงจรในรูปที่ 3.5 วงจรไมโครคอมพิวเตอร์ขนาดเล็กนี้มีหน่วยความจำทั้งหมด 64 กิโลไบท์ แบ่งเป็น 32 กิโลไบท์ สำหรับหน่วยความจำถาวรและอีก 32 กิโลไบท์ สำหรับหน่วยความจำชั่วคราวโดยมีไอซี 2A ถอดรหัสแอดเดรส หน่วยความจำชั่วคราวจะได้รับการสำรองข้อมูลโดยแบตเตอรี่ในวงจรที่ 3.7 เมื่อจ่ายไฟเข้าเครื่องเทอร์มินอลหากแรงดันไฟอินพุทสูงกว่า 3.6 โวลท์ Q1 จะทำงานทำให้ประจุที่ C6 ถูกประจุ เป็นผลให้ขา 2 ของไอซี 13A มีสถานะเป็นลอจิก "1" จนกระทั่ง C6 เก็บประจุจนเต็ม ขา 2 ของไอซี 13A จึงมีสถานะเป็น "0" สัญญาณดังกล่าวถูกนำไปรีเซตวงจรทั้งหมด Q2 และ Q3 ทำหน้าที่ตัดต่อแหล่งจ่ายไฟกับแบตเอตรี่เพื่อจ่ายให้หน่วยความจำชั่วคราว วงจรในรูปที่ 3.6 เป็นวงจรสร้างสัญญาณนาฬิกาอ้าวอิง สัญญาณ RS-232 CLK นำไปใช้กำหนดความเร็วการรับส่งข้อมูลระหว่างเครื่องเทอร์มินอลกับโฮสคอมพิวเตอร์ สัญญาณ SYNC นำไปใช้สร้างฐานเวลาสำหรับการรับส่งแบบอโลฮา ดังในบทที่ 2 สัญญาณ 16X CLK ส่งไปยังวงจรเข้ารหัสและถอดรหัสข้อมูลในรูปที่ 3.9 สัญญาณ CPU CLK ใช้สำหรับเป็นสัญญาณนาฬิกาของระบบไมโครคอมพิวเตอร์ สัญญาณ TX CLK ใช้สำหรับกำหนดความเร็วการส่งข้อมูลระหว่างเครื่องเทอร์มินอล วงจรในรูปที่ 3.9 เป็นวงจรเข้ารหัสและถอดรหัสข้อมูลแบบ NRZI ในส่วนของการเข้ารหัสสัญญาณข้อมูล สัญญาณ TX DATA ถูกต่อเข้าขา J และ K ของไอซี 15 B ซึ่งเป็น JK Flip-Flop ทำงานตามจังหวะของสัญญาณ TX CLK เมื่อสัญญาณ TX DATA มีสถานะเป็นลอจิก "0" ขา 5 ของไอซี 15B จะเกิดการเปลี่ยนแปลงตรงข้ามกับสถานะเดิมตามจังหวะของ TX CLK หากสัญญาณ TX DATA มีสถานะลอจิก "1" ขา 5 ของไอซี 15B จะไม่เกิดการเปลี่ยนแปลงจากสถานะเดิม สัญญาณที่ขา 5 นี้เป็นสัญญาณ NRZI ซึ่งผ่าน PORT1 เพื่อลดระดับสัญญาณให้มีความเหมาะสมก่อนผ่านเข้า C11 ไปโมดูเลตแบบไบนารีเอฟเอสเคต่อไป ในส่วนของการรับสัญญาณ DATA IN จะถูกสุ่มตามจังหวะของสัญญาณ 16X CLK เพื่อตรวจจับการเปลี่ยนแปลงของระดับสัญญาณโดยมีไอซี 18A ไอซี 18B และไอซี 16C ทำหน้าที่ดังกล่าว ไอซี 17B เป็นวงจรหารความถี่ลง 16 เท่า จากความถี่ของสัญญาณ 16X CLK เมื่อสัญญาณ DATA IN มีการเปลี่ยนแปลงระดับของสัญญาณจะทำให้วงจรหารความถี่ถูกรีเซต ดังนั้นเฟสของสัญญาณที่ได้จากวงจรหารความถี่หรือสัญญาณ RX CLK จึงเท่ากับเฟสของสัญญาณ DATA IN ไอซี 15A ทำหน้าที่ถอดรหัสสัญญาณ NRZI เมื่อสัญญาณ DATA IN เกิดการเปลี่ยนแปลงระดับสัญญาณหรืออาจกล่าวได้ว่าฝ่ายตรงข้ามได้ส่งลอจิก "0" มา จะทำให้ขา 13 ของไอซี 15A ถูกกระตุ้น ดังนั้นขา 3 ของไอซี 15A มีสถานะเป็น "0" หากระดับของสัญญาณ DATA IN ไม่เกิดการ

เปลี่ยนแปลงในจังหวะของสัญญาณ RX CLK จะทำให้ไอซี 15A ถูกเซตเป็น "1" กล่าวได้ว่าฝ่ายตรงข้ามได้ทำการส่งลอจิก "1" มา

วงจรในรูปที่ 3.10 เป็นวงจรเครื่องส่งวิทยุแบบไบนารีเอฟเอสเค IC19 ทำหน้าที่เป็นวงจรออสซิลเลเตอร์แบบเปลี่ยนความถี่ด้วยแรงดันไฟฟ้าโดยมี L1 และ VD1 เป็นวงจรเรโซแนนซ์แบบขนาน แรงดันควบคุมจากวงจรเฟสล็อกลูปผ่านเข้า R19 ไปทำรีเวิร์สไบอัสกับ VARACTOR ทำให้ความจุแฝงภายใน VARACTOR มีการเปลี่ยนแปลงค่าความจุผกผันกับแรงดันอินพุท ดังนั้นเมื่อแรงดันสูงขึ้นความถี่ของวงจรออสซิลเลเตอร์ก็จะสูงขึ้นเช่นกัน สัญญาณ MOD เป็นสัญญาณข้อมูลที่ถูกนำมามอดูเลต ทำให้ความถี่ของวงจรออสซิลเลเตอร์มีการเปลี่ยนแปลงตามสัญญาณ MOD L3 C21 และ C22 ทำหน้าที่คัปปลิงและปรับอิมพิแดนระหว่า IC19 และทรานซิสเตอร์ Q4 เพื่อขยายสัญญานให้มีความแรงในระดับหนึ่งก่อน หลังจากนั้นจึงนำไปขยายอีกครั้งโดยผ่าน T1 ไปเข้า Q5 เพื่อขยายสัญญานครั้งสุดท้าย สัญญานที่ได้จาก Q5 จะผ่าน L4 เพื่อปรับอิมพิแดน เมื่อภาคส่งทำงานแรงดันไฟฟ้าส่วนหนึ่งจะผ่าน D4 และ D5 ทำให้ไดโอดอยู่ในสภาวะฟอร์เวิร์ดไบอัสหรือเปรียบเสมือนว่าสวิทช์ถูกเปิดทำให้สัญญานจากภาคส่งสามารถผ่านสู่สายอากาศได้ ในขณะที่ C31 เปรียบเหมือนถูกลัดวงจรทำให้ L6 และ C30 เรโซแนนแบบขนานซึ่งจะมีอิมพิแดนสูงที่ความถี่เรโซแนน

ค่าของ L6 C30 และ C31 หาได้จากสมการ

$$C = \frac{1}{2\pi f Z_o}$$

$$L = \frac{Z_o}{2\pi f}$$

เมื่อแทนค่า Z_o เท่ากับ 50 โอห์มและความถี่กลางเป็น 146 เมกกาเฮิร์ต จะได้ค่า L6 เป็น 54.5 นาโนเฮนรี่ C30 และ C31 จะมีค่า 22 พิโคฟารัด

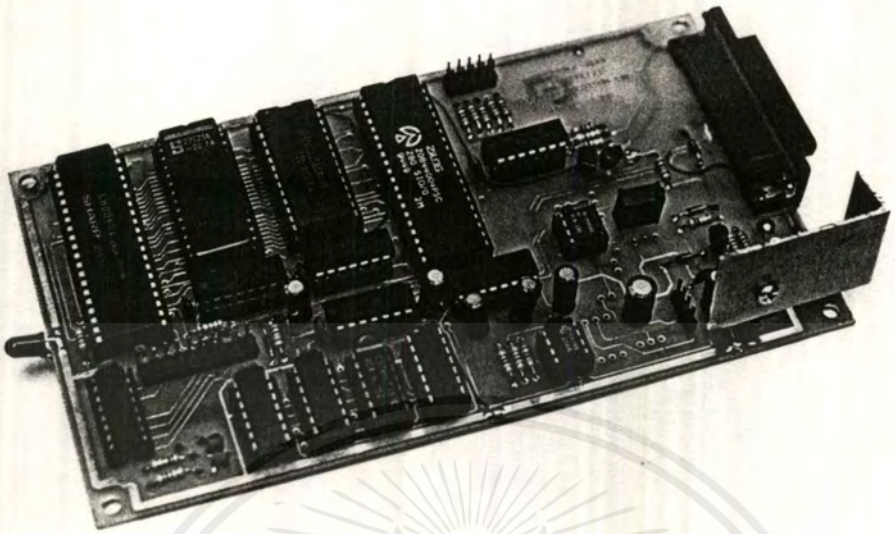
วงจรในรูปที่ 3.11 เป็นวงจรเครื่องรับแบบไบนารีเอฟเอสเค FET3 ทำหน้าที่เป็นพรีแอมปริไฟร์เพื่อขยายสัญญาณวิทยุให้แรงขึ้นก่อนเข้าไอซี 20 เบอร์ MC 3362 ซึ่งเป็นชิปเครื่องรับวิทยุเอฟเอ็มแบบแบนด์แคบ C33 C32 และ L6 ทำหน้าที่ปรับอิมพิแดนของเฟทกับสายอากาศส่วน C37 และ T2 ทำหน้าที่ปรับอิมพิแดนระหว่างเฟทและไอซีเบอร์ MC3362 แรงดันไฟควบคุมความถี่ของ VCO ภายใน IC21 จากวงจรเฟสลอกลูปจะผ่าน R23 และ L7 เข้าขา 23 วงจร VCO ภายในไอซีเบอร์ MC3362 สามารถกำหนดความถี่ได้จากวงจรเรโซแนนแบบขนาน L8 และ C39 ทำหน้าที่ดังกล่าว สัญญานความถี่จากวงจร VCO ส่วนหนึ่งจะถูกส่งออกมาที่ขา 21 ของ IC21 ผ่านเข้า IC 20 เพื่อเป็นบัฟเฟอร์ก่อนส่งไปเข้าวงจรเฟสลอกลูปต่อไป สัญญานตวามถี่จาก VCO จะถูกนำไปผสมกับสัญญานที่รับได้จากสายอากาศแล้วเลือกเอาสัญญานผลต่างความถี่ 10.7 เมกกาเฮิร์ตโดยใช้เซรามิคฟิลเตอร์ 10.7 เมกกาเฮิร์ต หลังจากนั้นความถี่กลางขนาด 10.7 เมกกาเฮิร์จากถูกนำไปผสมกับความถี่ 10.245 เมกกาเฮิร์ตที่ผลิตโดยแร่คริสตอล X2 คัดเลือกเอาสัญญานผลต่างขนาด 455 กิโลเฮิร์ตไปทำการดีเทกต่อไป สัญญานที่ผ่านกสรดีเทกแล้วจะได้สัญญานที่ถูกมอดูเลท มาออกที่ขา 13 ของ IC21 ก่อนผ่านเข้าวงจรคอมพาราเตอร์เพื่อปรับสัญญานที่เป็นอนาลอกให้อยู่ในรูปของสัญญานดิจิตอลโดยส่งผ่าน C44 เข้าขา 13 ออกที่ขา 14 เป็นสัญญานดิจิตอล

ในรูปที่ 3.12 เป็นวงจรเฟสลอกลูป สัญญานออสซิลเตอร์จาก VCO ของทั้งภาครับและภาคส่งจะผ่าน D6 หรือ D7 โดยมีสัญญาน PTT จากภาคควบคุมและประมวลผล กำหนดสภาวะรับหรือส่ง Q6 และ Q7 เป็นสวิตช์จ่ายแรงดันไฟแก่ภาครับหรือภาคส่ง สัญญานความถี่จากวงจร VCO จะผ่านเข้า IC23ซึ่งเป็นปรีสเกลเลอร์ 64/65 มีขาเลือกตัวนับที่ขา 1 เอาท์พุทที่ได้ที่ขา 2 จะถูกส่งไปที่ IC24 เป็นชิปเฟสลอกลูปความถี่อ้างอิงของวงจรเฟสลอกลูปผลิตจากตริสตอล 10.24 เมกกาเฮิร์ตแล้วทำการหารจนเหลือ 20 กิโลเฮิร์ตแล้วไปเปรียบเทียบกับความถี่ที่ได้จากวงจรหารความถี่แบบสองมอดูลัส N0 ถึง N9 เป็นค่าของตัวนับหลัก A0 ถึง A5 เป็นตัวนับรอง ไดโอด D8 ถึง D21 ทำหน้าที่กำหนดค่าตัวหารของภาครับ ส่วน D22 ถึง D36 ทำหน้าที่กำหนดค่าตัวหารของภาคส่ง หาต้องการให้บิทใดเป็นหนึ่งก็ให้ปลดไดโอดที่ขานั้นออก ความถี่ของภาครับและส่งจะต่างกันอยู่ 10.7 เมกกาเฮิร์ต ตัวอย่างเช่น ต้องการทำงานที่ความถี่ 146.02 เมกกาเฮิร์ต ต้องการหาค่าของตัวนับเสริมและตัวนับหลัก เมื่อนำความถี่ที่กำหนดมาหาค่าตัวหารจะได้ค่าตัวหารรวม 146.02 MHz/20 KHz = 7301 จากสมการที่ 79 หากให้ค่าตัวหารหลักเท่ากับ 144 ค่าของตัวหารรองจะเป็น 5 เราก็นำค่าที่ได้แปลงเป็นเลขฐานสองแล้วนำไปกำหนดตำแหน่งการใส่ไดโอดของภาคส่ง ในส่วนของภาครับเราต้องนำความถี่ที่เราต้องการมาบวกหรือลบด้วย 10.7 เมกกาเฮิร์ตเสียก่อนเช่น 146.02 เมกกาเฮิร์ต – 10.7เมกกาเฮิร์ต = 135.32 เมกกาเฮิร์ต หาค่าตัวหารรวมได้ 135.32 เมกกาเฮิร์ต / 20 กิโลเฮิร์ต เท่ากับ 6766 หากให้ค่าตัวหารหลักเป็น 105 ค่าของตัวหารรองจะเป็น 46 นำค่าที่ได้ไปกำหนดตำแหน่งของไดโอดในภาครับต่อไป ผลการเปรียบเทียบเฟสจะถูกส่งไป IC25A และ IC25B ไปควบคุมความถี่ภาครับและภาคส่งต่อไป ค่าของ C57-C60 และ R33-R39 หาได้จากสมการ

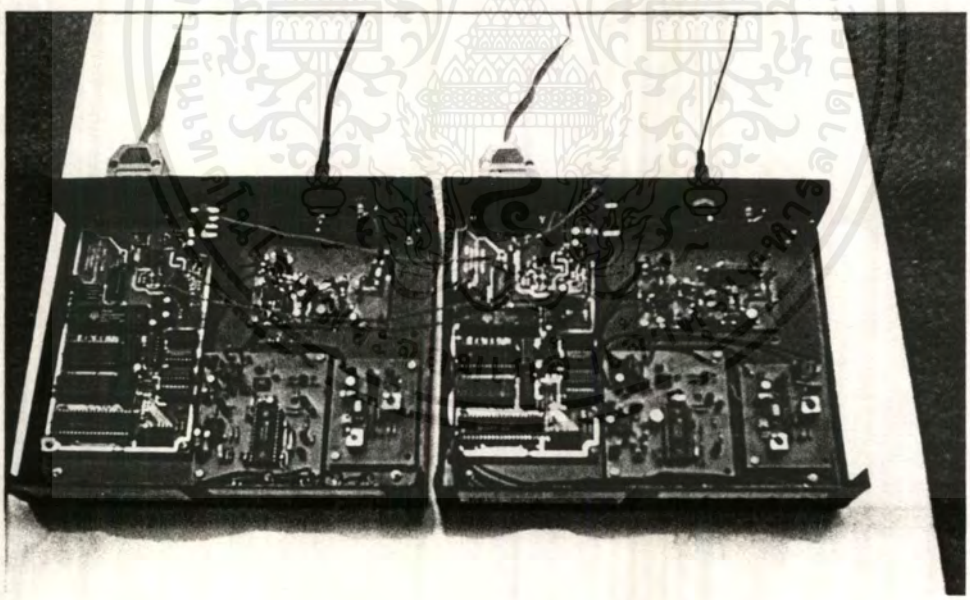
$$\omega_n = \sqrt{\frac{K_\theta K_{VCO}}{NR_1C}}$$

$$\zeta = \frac{\omega_n R_2 C}{2}$$

โดยที่ N คือค่าตัวหารจากตัวอย่างมีค่า 7301 สำหรับภาคส่งและ 6766 สำหรับภาครับ K_θ มีค่าเท่ากับ VDD/2π =5/2*3.14= 0.8 ค่า $K_{VCO} = 43.4\times10^6$ เรเดียน/วินาทีสำหรับภาคส่ง(จากการทดลอง) และ 79.8×10^6เรเดียน/วินาทีสำหรับภาครับ ค่า $\omega_n \cong (2\pi f_r)/10 \cong 12.6\times10^3$ เรเดียน/วินาที และให้ δ=1 แทนค่าทั้งหมดหาค่าอุปกรณ์สำหรับลูปฟิลเตอร์ได้ดังในวงจร



รูปที่ 3.13 ส่วนคอนโทรลเลอร์ที่สำเร็จแล้ว



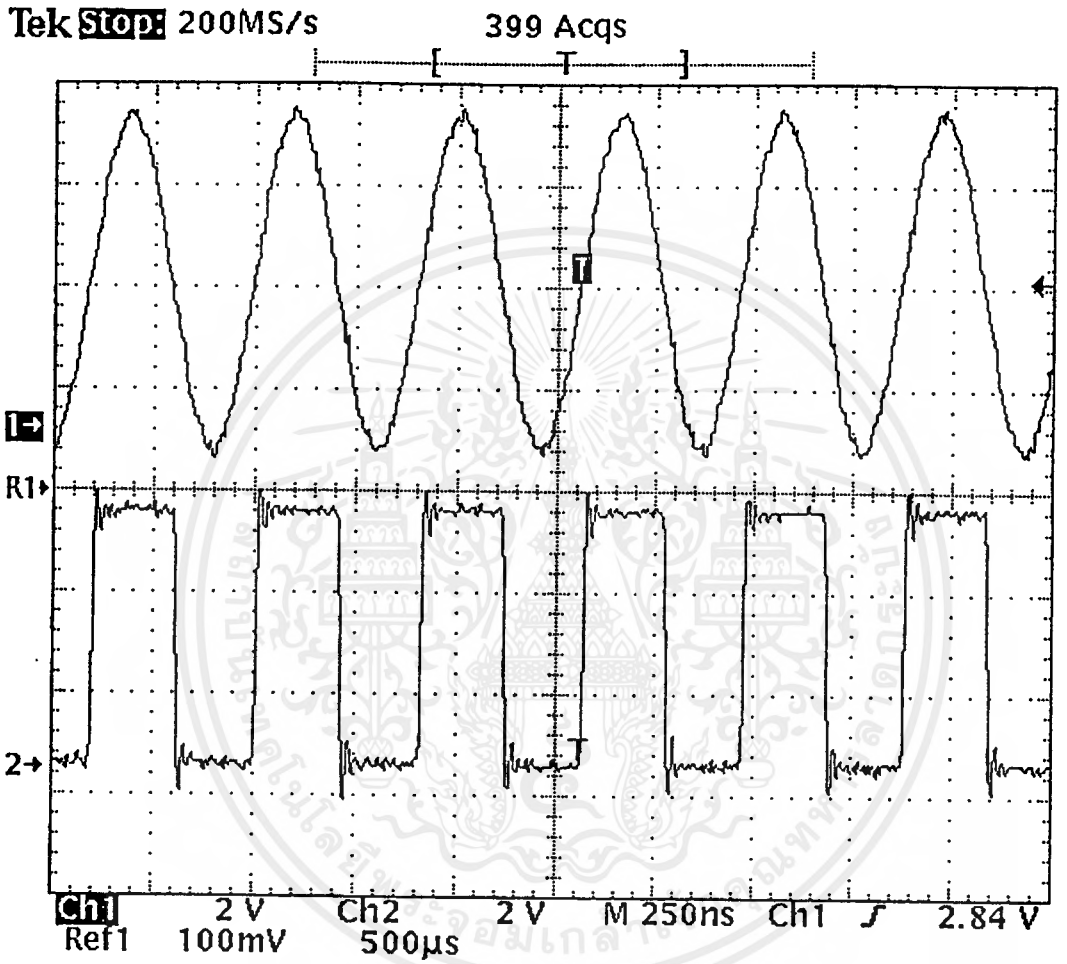
รูปที่ 3.14 รูปสำเร็จเมื่อได้ต่อวงจรส่วนต่าง ๆ เข้าด้วยกันแล้ว

บทที่ 4

การทดลองและผลการทดลอง

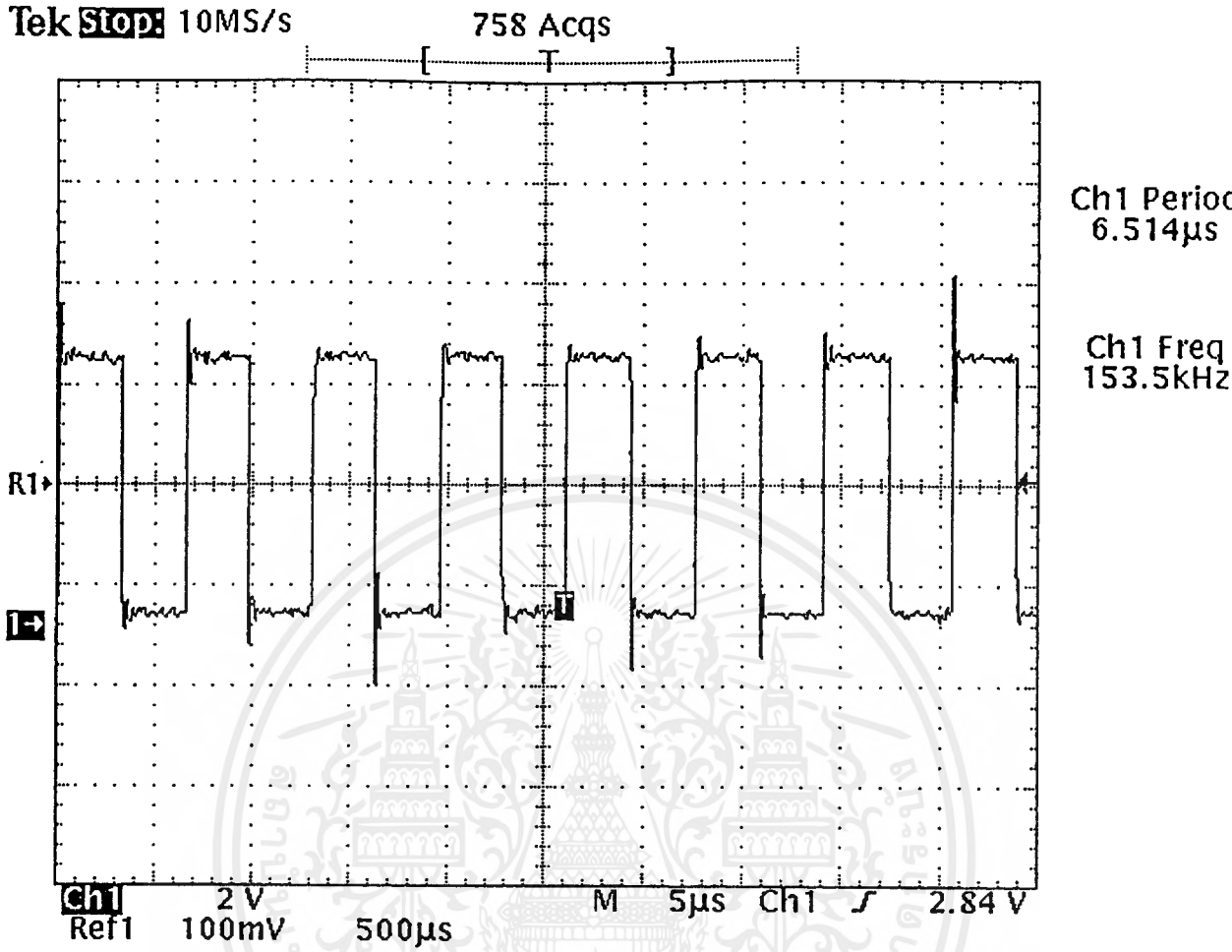
4.1 ส่วนควบคุม

เมื่อทำการป้อนไฟเข้าสู่วงจรส่วนควบคุม แล้วใช้ออสซิลโลสโคปทำการวัดสัญญาณตามจุดต่าง ๆ ของวงจรได้ดังนี้



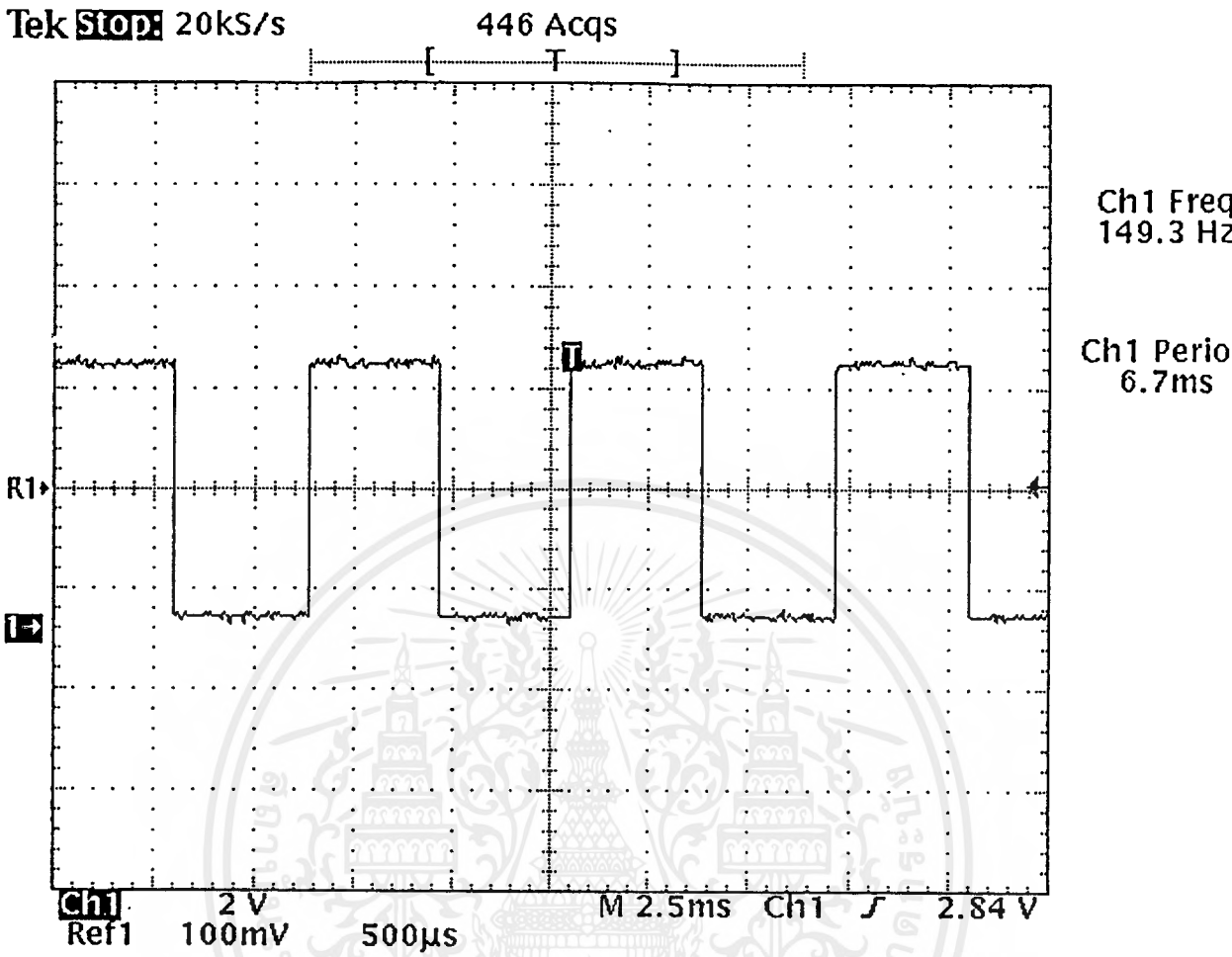
รูปที่ 4.1 สัญญาณที่ได้จากคริสตอลและสัญญาณนาฬิกาที่ป้อนเข้าระบบไมโครโปรเซสเซอร์

จากรูปที่ 4.1 เป็นการวัดสัญญาณที่ได้จากขา 11 ของ IC8 ซึ่งต่อกับคริสตอล 2.4576 MHz เทียบกับสัญญาณนาฬิกาที่ป้อนเข้าสู่ส่วนควบคุม จะเห็นว่า IC8 ทำหน้าที่ผลิตสัญญาณนาฬิกา โดยใช้คริสตอลดังกล่าว



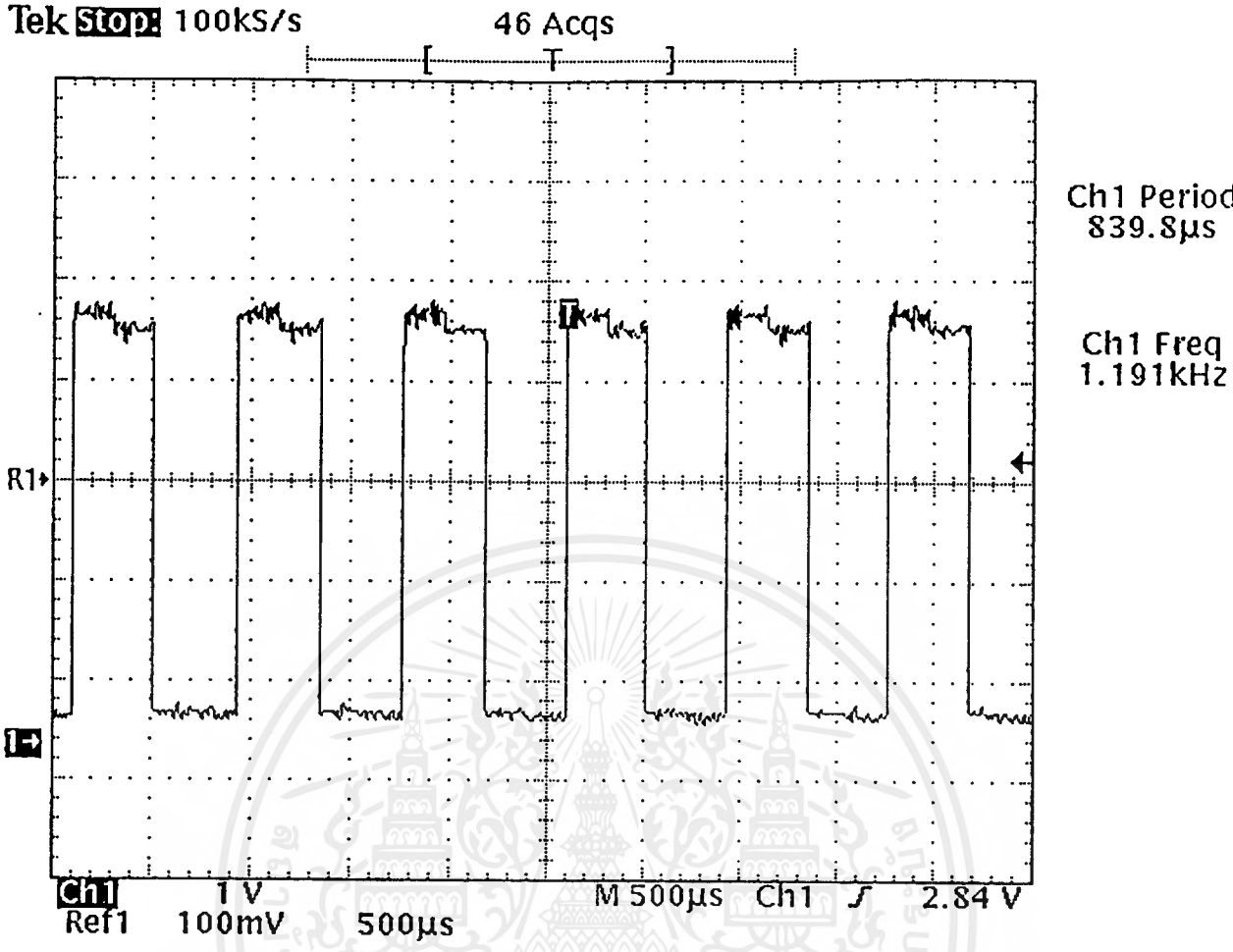
รูปที่ 4.2 สัญญาณนาฬิกาที่ใช้กำหนดความเร็วของอัตราการรับส่งของพอร์ท RS232

จากรูปที่ 4.2 เป็นการวัดสัญญาณนาฬิกาที่นำไปสู่ RS232 Clk ของรูปวงจรที่ 3.6 ซึ่งใช้กำหนดความเร็วของพอร์ท RS232 โดยวงจรหารภายใน IC6 จะทำหน้าที่หารความถี่ดังกล่าวให้ตรงกับอัตราการส่งข้อมูลที่แท้จริงต่อไป



รูปที่ 4.3 สัญญาณนาฬิกาที่ใช้กำหนดฐานเวลามาตรฐานของระบบบอโลฮา

จากรูปที่ 4.3 เป็นสัญญาณที่ใช้กำหนดเวลามาตรฐานของระบบบอโลฮา สัญญาณดังกล่าวจะเข้าไปทำการสร้างสัญญาณอินเตอร์รัพท์ทุก ๆ 6.7 ms เพื่อให้โปรแกรมทำการนับให้ครบ 1 วินาที โดยทำการนับ 150 ครั้ง จะได้ 1 วินาที

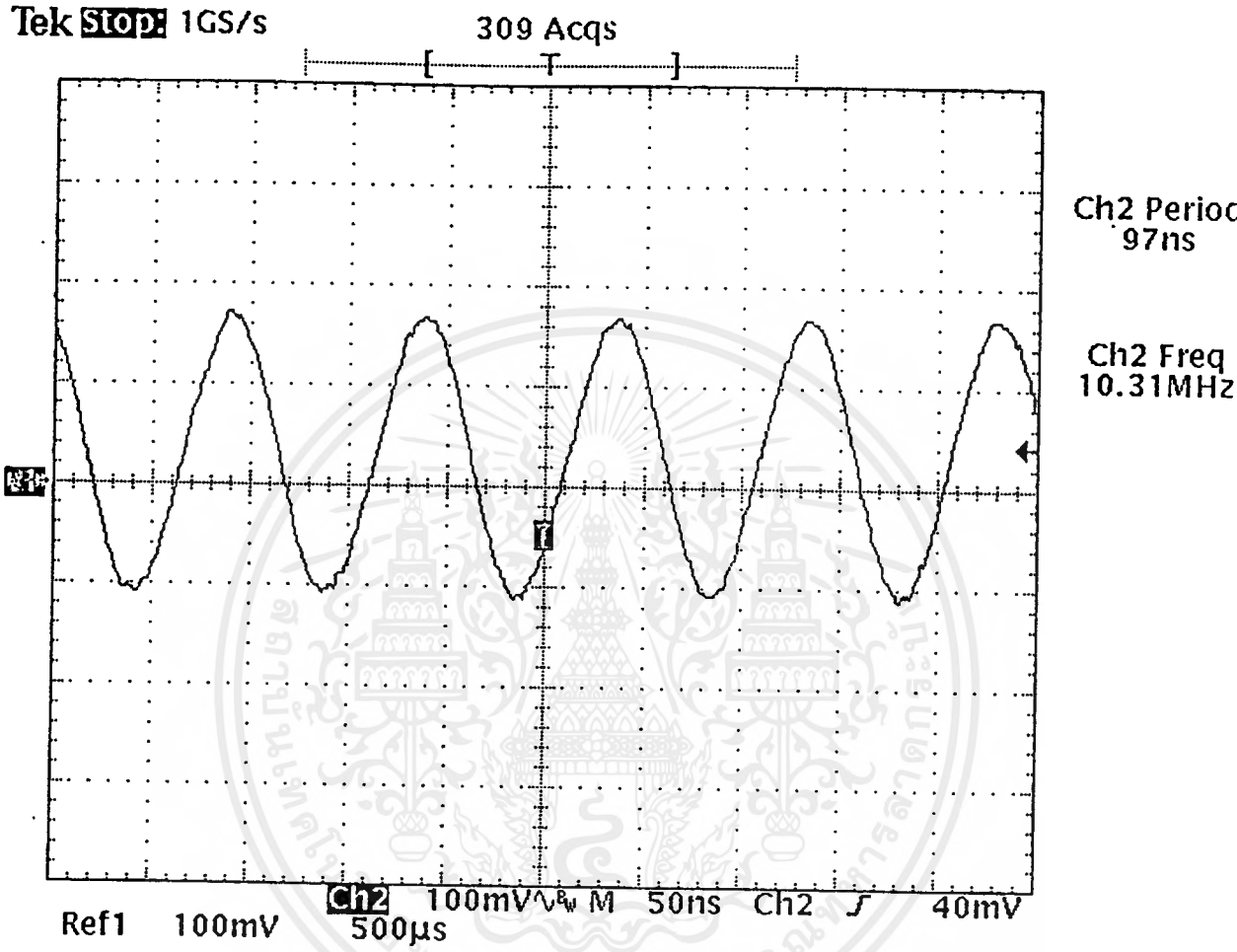


รูปที่ 4.4 สัญญาณนาฬิกาที่ใช้กำหนดอัตราการส่งแพ็กเกจ

จากรูปที่ 4.4 เป็นสัญญาณที่ใช้กำหนดความเร็วของการส่งข้อมูล ในแบบแพ็กเกจ ก่อนนำไปมอดูเลทที่ภาคเครื่องรับเครื่องส่งต่อไป

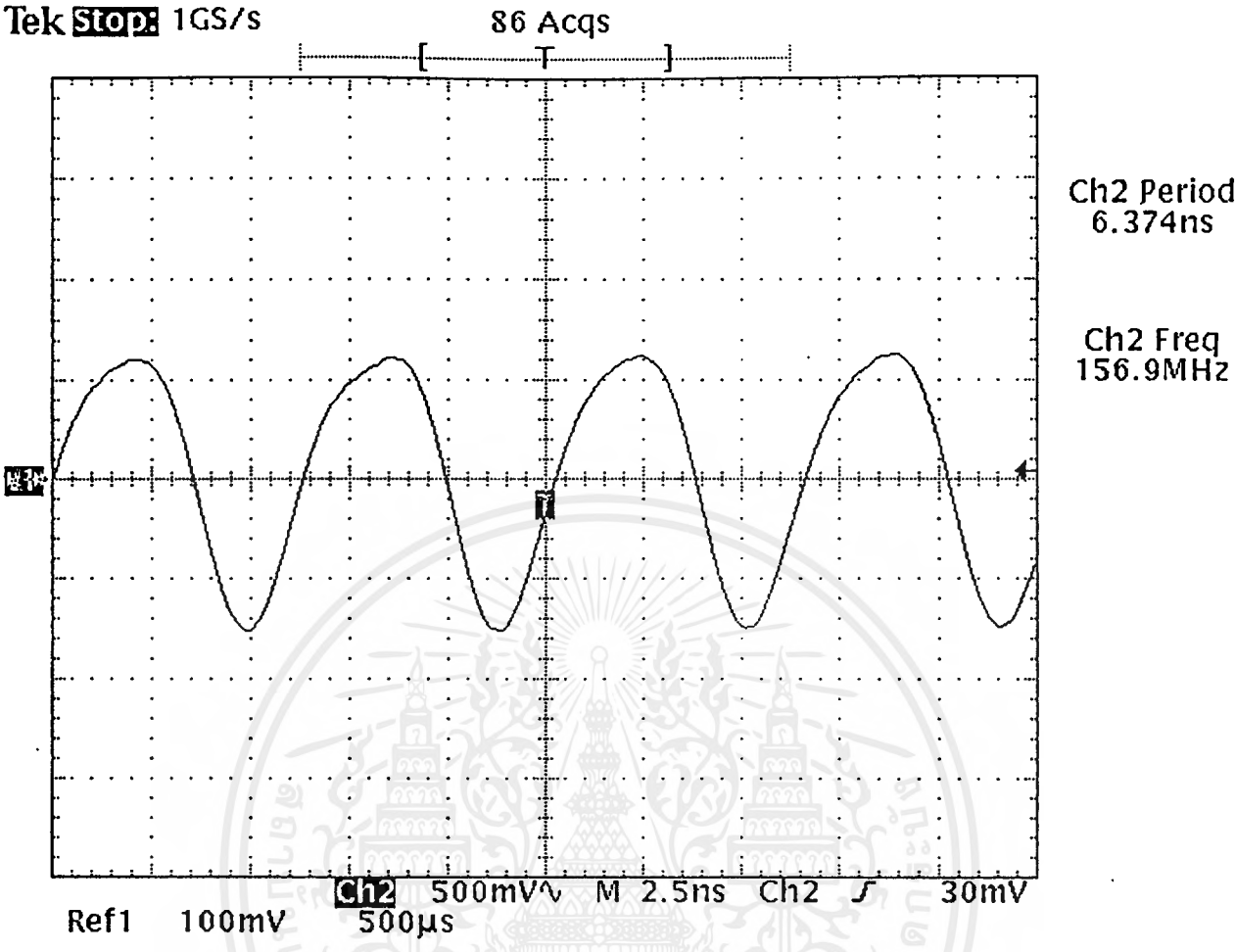
4.2 ส่วนเครื่องรับและเครื่องส่ง

ทำการเชื่อมต่อวงจรเครื่องรับและเครื่องส่ง พร้อมทั้งวงจรเฟสล็อกลูปเข้าด้วยกัน แล้วทำการวัดสัญญาณตามจุดต่าง ๆ ขณะที่ยังไม่ได้ทำการมอดูเลท



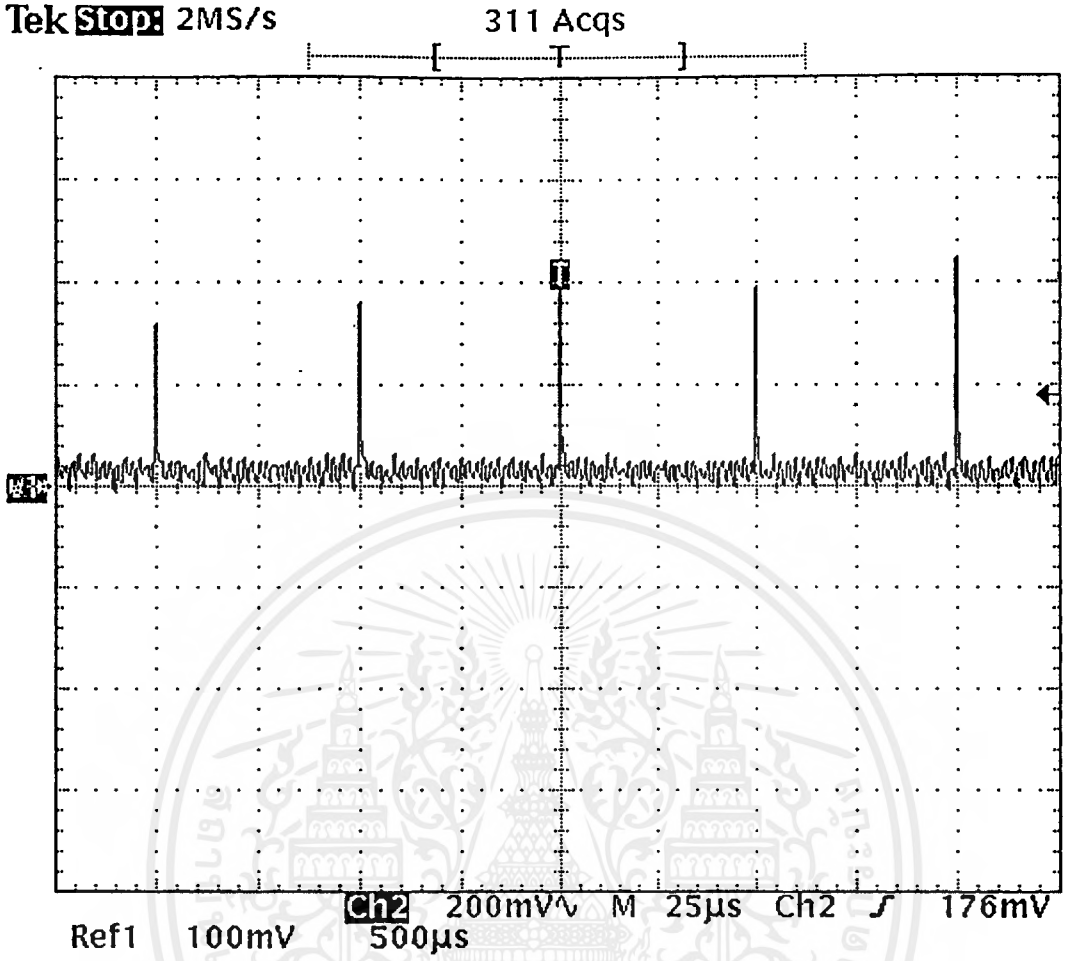
รูปที่ 4.5 สัญญาณจากคริสตอลที่ใช้เป็นฐานเวลาของระบบเฟสล็อกลูป

จากรูปที่ 4.5 เป็นสัญญาณที่ได้จากแร่คริสตอล 10.24 MHz ซึ่งใช้เป็นฐานเวลาของระบบเฟสล็อกลูป เพื่ออ้างอิงกับความถี่ที่ได้จากการหารความถี่จาก VCO แล้วนำไปเปรียบเทียบกันที่วงจรเฟสดีเท็กเตอร์



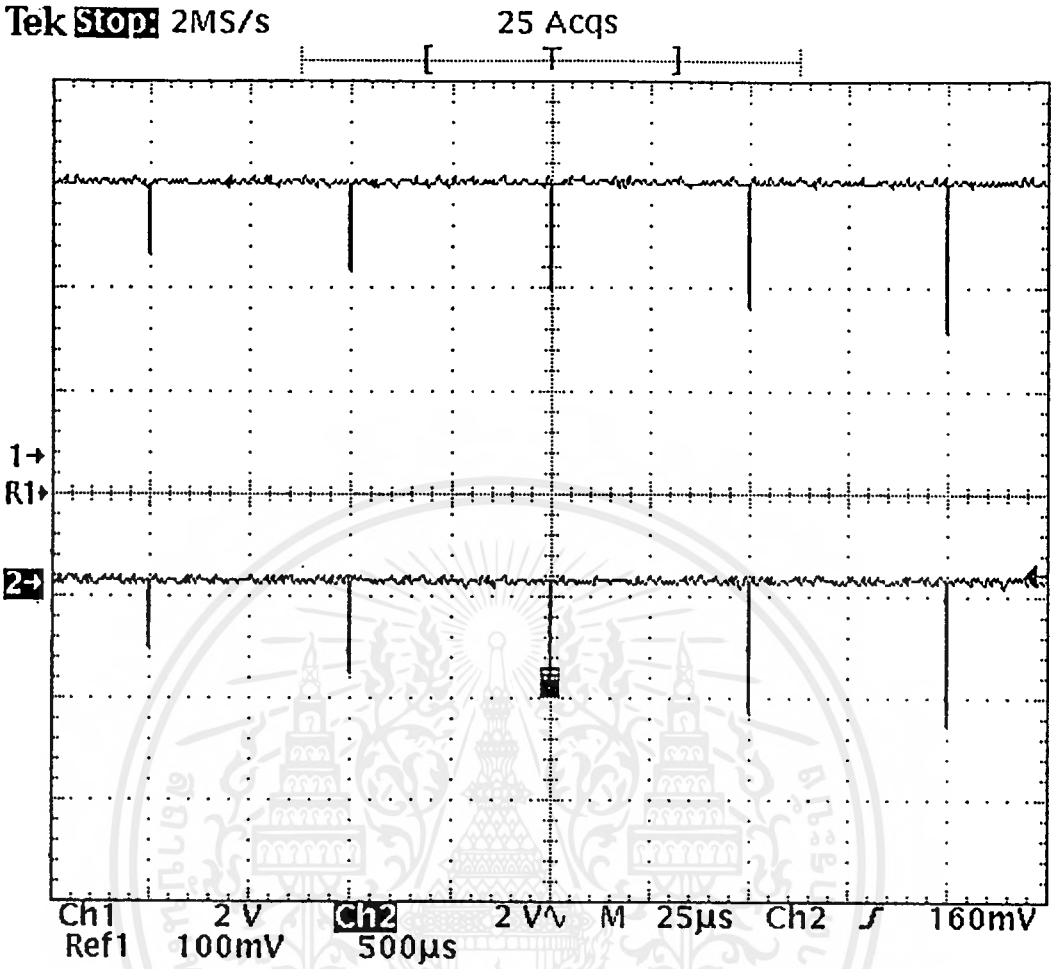
รูปที่ 4.6 สัญญาณความถี่ที่ป้อนเข้าสู่ภาครับ

จากรูปที่ 4.6 ความถี่ที่ได้จากวงจรเฟสล็อกลูปไปควบคุมวงจร VCO สำหรับภาครับ เพื่อผลิตความถี่ที่ต่างจากความถี่ส่งอยู่ 10.7 MHz ความถี่นี้จะนำไปผสมกับความถี่ที่รับเข้ามา ทำให้เป็นความถี่กลางขนาด 10.7 MHz เพื่อนำไปดีเทกเตอร์สัญญาณออกมา



รูปที่ 4.7 สัญญาณไฟตรงที่ไปควบคุมวงจร VCO

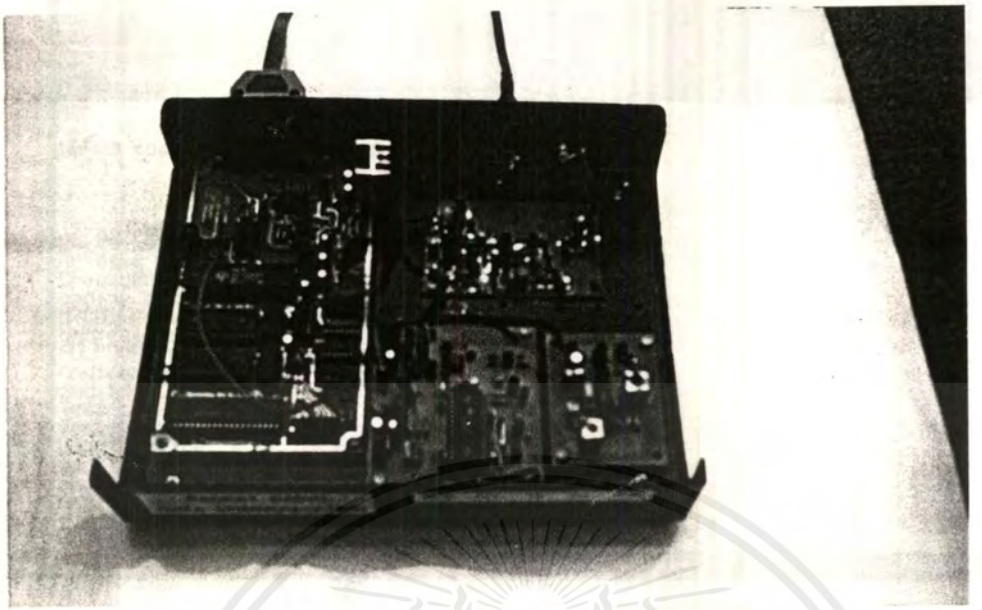
จากรูปที่ 4.7 ได้ทำการวัดที่ขา 7 หรือขา 1 ของ IC23 เป็นสัญญาณไฟที่นำไปควบคุมวงจร VCO ซึ่งได้จากลูปฟิลเตอร์ สัญญาณนี้แสดงให้เห็นเป็นรูปคลื่นที่ไม่พึงประสงค์



รูปที่ 4.8 สัญญาณที่แสดงให้เห็นว่าวงจรเฟสล็อกลูปทำการล็อก

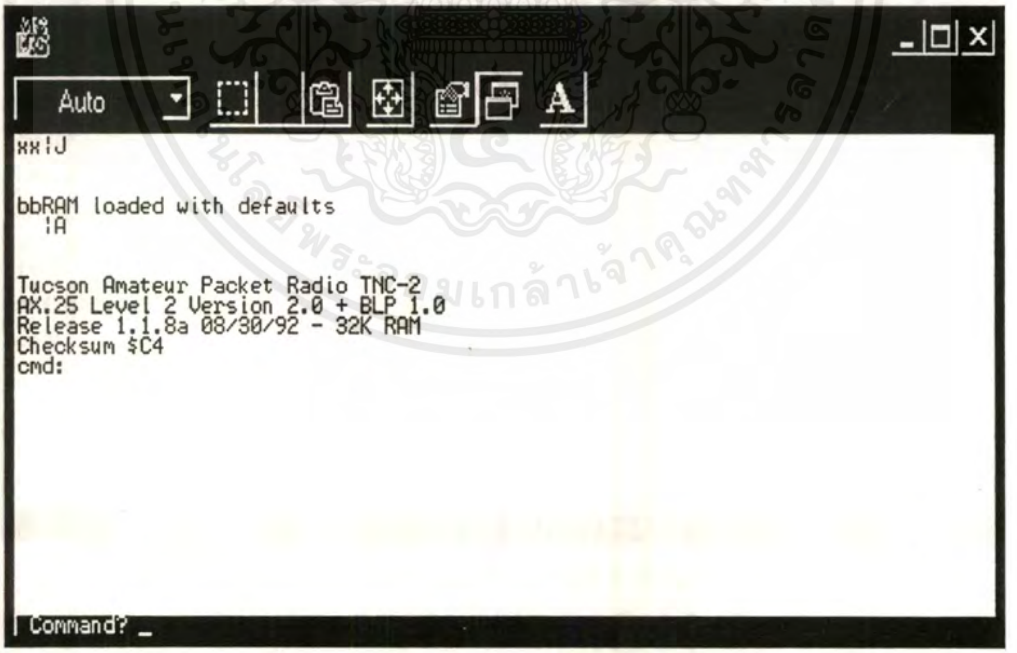
จากรูปที่ 4.8 ได้จากการวัดที่ขา 8 และ 9 ของ IC22 แสดงให้เห็นถึงสภาวะการล็อกของวงจรเฟสล็อกลูป ในสภาวะนี้เฟสและระดับสัญญาณทั้งสองจะต้องใกล้เคียงกันจึงจะถือว่าวงจรเฟสล็อกลูปได้ทำการล็อกโดยสมบูรณ์แล้ว

4.3 การทดลองรับส่งข้อมูล



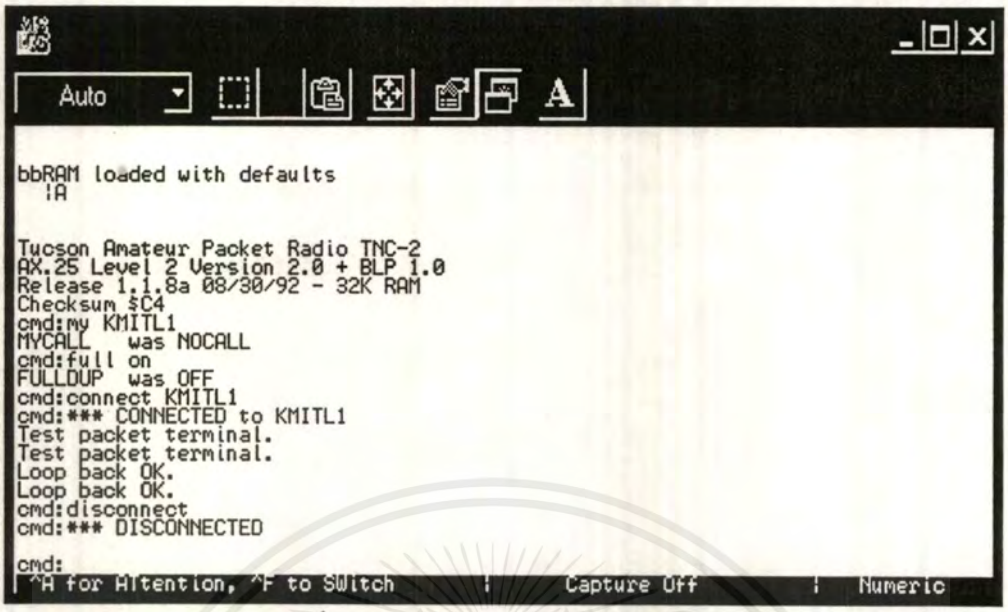
รูปที่ 4.9 การต่อวงจรทำการทดลองรับส่งข้อมูล

จากรูปที่ 4.9 ทำการต่อวงจรควบคุมเข้าแหล่งจ่ายไฟ 12 V ต่อสาย RS232 เข้าเครื่องคอมพิวเตอร์ โดยการทดลองแบบดิจิตอลลูปแบ็ค ซึ่งได้ทำการต่ออินพุทและเอาท์พุทบนแผ่นวงจรภาคควบคุม



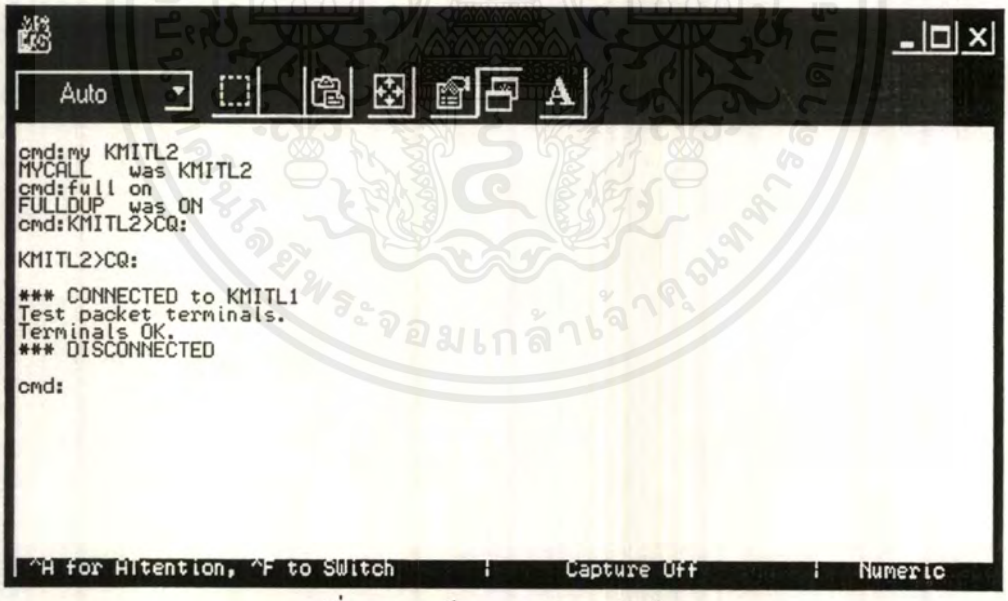
รูปที่ 4.10 หน้าจอเริ่มต้นการทำงานของโปรแกรม

จากรูปที่ 4.10 เมื่อได้ทำการเรียกโปรแกรม Xtalk เพื่อทำการเชื่อมต่อกับวงจรควบคุมแล้ว



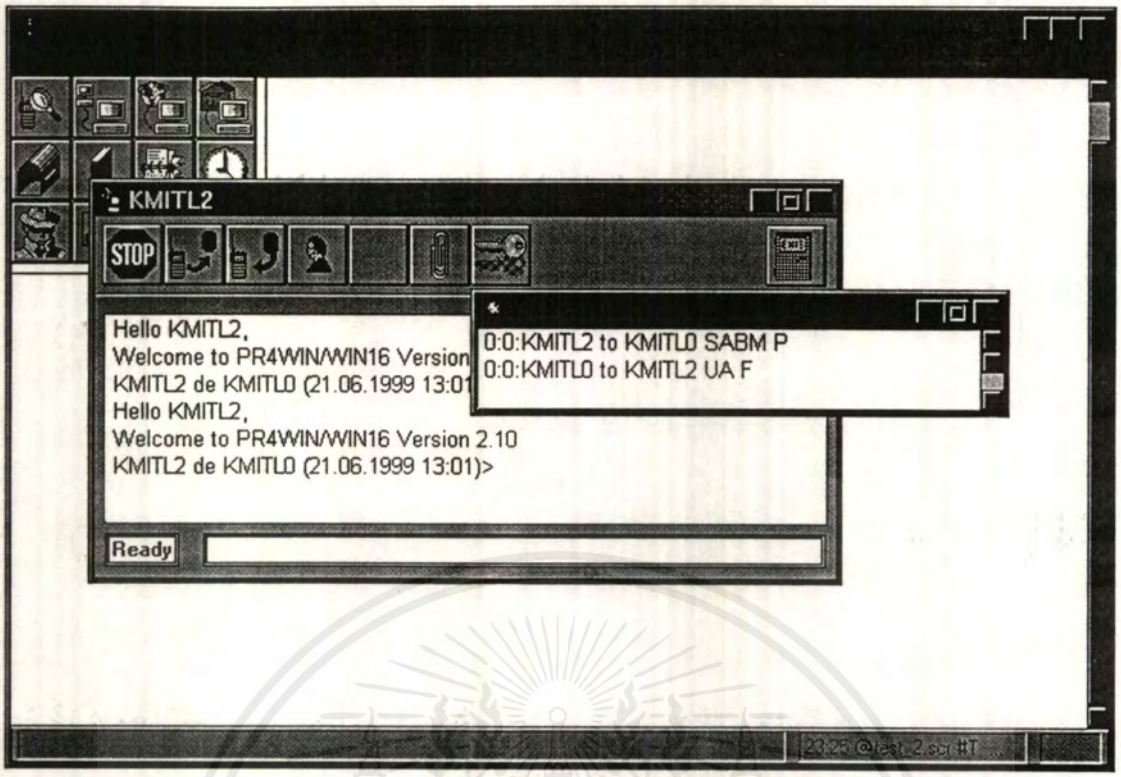
รูปที่ 4.11 หน้าจอของการเตรียมส่งข้อมูล

จากรูปที่ 4.11 เมื่อได้ทำการเรียกโปรแกรมตามรูปที่ 4.10 แล้ว ก็ได้ทำการตั้งค่าพารามิเตอร์ของการกำหนดชื่อของเครื่อง จากนั้นทำการกำหนดโหมดเป็นแบบฟูลดูเพล็กซ์ หลังจากนั้นทำการติดต่อโดยใช้ชื่อเครื่องเดียวกัน แล้วพิมพ์ข้อความที่จะทำการส่ง



รูปที่ 4.12 หน้าจอการส่งและรับข้อมูล

จากรูปที่ 4.12 เป็นรูปสมบูรณ์เมื่อได้ทำการส่งและรับข้อมูลเรียบร้อยแล้ว



รูปที่ 4.13 ผลการทดลองการรับส่งข้อมูลจากโปรแกรมที่เขียนขึ้น

จากรูปที่ 4.13 เป็นผลการทดลองเรียกโปรแกรม P4WIN เมื่อติดตั้งอีพรอมโปรแกรมที่ได้จากการคอมไพล์โปรแกรมที่แสดงไว้ในภาคผนวก เพื่อแสดงให้เห็นวิธีการรับส่งข้อมูลแบบอโลฮา

บทที่ 5
สรุปและวิจารณ์

จากผลการทดลองพบว่าการทำงานของส่วนควบคุมสามารถใช้งานได้ทั้งการส่งและการรับ ในการทดลองนี้เราได้ใช้ซอฟแวร์สำเร็จรูปชื่อ TNC2 ทำการส่งแพ็กเกจมาตรฐาน แล้วใช้โปรแกรมที่เขียนขึ้นแสดงให้เห็นในภาคผนวก แล้วพบว่าการรับส่งข้อมูลแบบแพ็กเกจที่ได้ทำการศึกษานั้นสามารถใช้งานได้จริง โปรแกรมที่เขียนขึ้นนั้นเป็นโปรแกรมที่ใช้จัดการ การเข้าถึงช่องสัญญาณในแบบอโลฮา ข้อมูลที่รับได้หรือส่งออกไปนั้นได้จากโปรโตคอลที่สูงขึ้นไปเช่น AX-25,NOS หรือ Ethernet โดยผ่านโปรโตคอลการเชื่อมต่อแบบ KISS โปรแกรมที่สนับสนุนโปรโตคอลเหล่านี้ได้แก่ WINPAC และ LINUX AX.25 Device Driver Pacth เป็นต้น

ในการทำงานของส่วนที่ยังไม่สมบูรณ์คือส่วนของภาครับและภาคส่ง ซึ่งปัญหาสามารถกล่าวได้ดังนี้

1. วงจรเฟสล็อกลูปในส่วนของลูปฟิลเตอร์ ค่าของ R42,R43,R45 และ R60 จะต้องมีค่ามาก ๆ เพื่อป้องกันการโหลด IC22 ซึ่ง IC22 สามารถจ่ายกระแสได้เพียง 1 mA เท่านั้น หากทำการจ่ายกระแสได้มากกว่านี้ก็จะเกิดสัญญาณรบกวนที่มีความถี่เท่ากับความถี่ที่ได้จากขา 9 ของตัว IC22 เอง แต่การเพิ่มค่าของความต้านทานดังกล่าวจะส่งผลให้ C9,C54,C59 และ C60 มีค่าน้อยมาก ๆ ซึ่งจะมีผลต่อเสถียรภาพของลูปฟิลเตอร์

2. วงจรเครื่องรับ FET3 จะต้องเลือกชนิดที่มีสัญญาณรบกวนต่ำมาก ๆ รวมทั้ง L7 และ T3 ควรเป็นแบบมีแผ่นโลหะปิดมิดชิด เพื่อป้องกันการออสซิลเลท

3. วงจรเครื่องส่ง IC19 ขาเอาท์พุท 4 และ 5 เป็นแบบคอมพลีเมนทารี่ ดังนั้น สัญญาณจากขาทั้งสองเมื่อเทียบกับกราวด์ จึงดูเหมือนว่ามีขนาดสัญญาณที่อ่อนมาก จึงควรแก้ไข โดยนำขา 4 หรือขา 5 ต่อลงกราวด์ แล้วใช้วงจร RF Devider ทำการแยกสัญญาณไปเข้า RF Amp และเฟสล็อกลูป จะดีกว่า

4. ในส่วนของวงจร RF สวิทช์ D4 และ D5 ควรเป็นแบบพินไดโอด ที่มีค่าความจุแฝงต่ำมาก ๆ เพราะเมื่อความถี่สูงขึ้นความจุแฝงดังกล่าวจะทำให้อิมพีแดนซ์ของไดโอดลดลง นอกจากนี้ L5 ควรออกแบบให้มีค่า Q สูงมาก ๆ เพื่อลดการสูญเสียของสัญญาณทั้งการรับและการส่ง เราสามารถนำเอา Quarter Wave Transmission line มาแทน C31 , C32 และ L5 แต่ก็จะทำให้วงจรมีขนาดใหญ่เกินไป

โปรแกรมที่ใช้ในการทดลอง

```
                cpu "z80.tbl"
                hof "int16"
                org     0000h
Free_RAM:   equ     8000h
TRUE:  equ   1
FALSE: equ   0
SIO:   equ   0dch       ;only A5 is used for select SIO
A_dat:  equ   SIO+0      ;modem data port
A_ctl:  equ   SIO+1      ;modem control port
B_dat:  equ   SIO+2      ;host data port
B_ctl:  equ   SIO+3      ;host control port
DCD:   equ   8          ;dcd bit in RR0, used in Ch A
CTS:   equ   32         ;cts bit in RR0, used in Ch A
TBE:   equ   4          ;TX Buffer Empty bit
RTS:   equ   2          ;Request To Send (PTT bit in WR5 of Chan A)
Framing_Error:  equ   40h   ;Bit in RR1 for async framing error
Break_Abort:   equ   80h   ;Bit in RR0 for async Break detection
Auto_Enable:   equ   0e1h
;frame character
FEND:   equ   300o      ;frame end character
FESC:   equ   333o      ;frame escape character
TFEND:  equ   334o      ;extend end
TFESC:  equ   335o      ;extend escape
;indicater values
ALEDon: equ   69h       ;bits for WR5 to turn on  STA LED
ALEDoff: equ   0e9h      ;bits for WR5 to turn off STA LED
ALED:   equ   80h       ;The DTR Bit in Ch A WR5, we will soon remove
                        ;previous 2 definitions & use the memory loc.
                        ;A_WR5 to hold Ch A WR5's value, because we
                        ;need to be aware when we are transmitting!
BLEDon: equ   6ah       ;bits for WR5 to turn on  CON LED
BLEDoff: equ   0eah      ;bits for WR5 to turn off CON LED
BLED:   equ   80h
start:
        jp    code_start    ;go around this data area
        org   10h
I_Vector:
        dwl   ib_tbe      ;ch B transmitter buffer empty interrupt/user
        dwl   ib_ext      ;ch B ext/status change/user
```

```
        dwl     ib_rca          ;ch B received char available/user
        dwl     ib_special      ;ch B special receive condition/user
        dwl     ia_tbe          ;ch A transmitter buffer empty interrupt/modem
        dwl     ia_ext          ;ch A ext/status change/modem
        dwl     ia_rca          ;ch A received char available/modem
        dwl     ia_special      ;ch A special receive condition/modem
code_start:
        di                      ;No interrupts for the moment...
;Init SIO.  This is required even if we wanna flash LEDs...
        in      a,(A_ctl)       ;assure we are talking to ch 0
        ld      c,A_ctl
        ld      b,a_size
        ld      hl,a_init
        otir                    ;init sync (modem) port
;Init Async port, also to allow flashing LEDs
        in      a,(B_ctl)       ;assure we are talking to ch 0
        ld      c,B_ctl
        ld      b,b_size
        ld      hl,b_init
        otir                    ;init async port & interrupt vector
; Figure out where top of stack is, set stack pointer.
; 32K RAM system.
        ld      sp,0            ;top of stack
;Clear out RAM.
        ld      bc,0ffffh-Free_RAM-1    ;get Byte Count into BC
        ld      hl,Free_RAM             ;
        ld      (hl),0                  ;
        ld      de,Free_RAM+1   ;get "source" address = Free_RAM
        ldir                    ;Zero memory.
;This sequence loads up our data area in RAM:
        ld      hl,data_init
        ld      de,TXdelay
        ld      bc,data_size
        ldir
; init free buffer list.
        ld      hl,Bottom       ;beginning of buffer space
                                        ;now it's also top of free list
        ld      b,{-1+{-100-Bottom}/128} & 0ffh  ;get buffers - 1
ibloop:
        push    hl
        ld      de,128
```

```
        add     hl,de           ;HL has "next" pointer
        ex      de,hl           ;DE has "next" pointer
        pop     hl              ;HL now has pointer to current buffer
        ld      (hl),e          ;low byte of "next" pointer first
        inc     hl
        ld      (hl),d          ;now hi byte
        inc     hl
        xor     a
        ld      (hl),a          ;zero out count field
        inc     hl
        ld      (hl),a          ;zero out # of bytes read field
        ex      de,hl           ;HL is now pointer to next buffer
        djnz    ibloop          ;and init all the available buffers
        xor     a
        ld      (hl),a          ;Last "next" address is 0
        inc     hl
        ld      (hl),a          ;
        inc     hl
        ld      (hl),a          ;zero out count field
        inc     hl
        ld      (hl),a          ;zero out # of bytes read field
;init regs for ib_ext interrupt
        exx
        ld      bc,0            ;set prev state of SYNC pin,for 1200hz
        ld      de,0            ;count of # of interrupts init
        exx
;Now have the CON and STA LEDs do a "dance".
        ld      b,6             ;Do it 6 times (arbitrary as hell, but should
                                ;be an even number so that the LEDs are off at
                                ;the end of this mess...)
        ld      hl,0            ;use HL as downcounter
dance0:
        call    CON_Flip
        call    STA_Flip
dance1:
        dec     hl
        ld      a,h
        or      l
        jr      nz,dance1
        djnz    dance0          ;do this 6 times  (3 "cycles")
        in      a,(A_ctl)       ;assure we are talking to ch 0
```

```
        ld    c,A_ctl
        ld    b,a_size
        ld    hl,a_init
        otir              ;init sync (modem) port
;Re-Init Async port.
        in    a,(B_ctl)     ;assure we are talking to ch 0
        ld    c,B_ctl
        ld    b,b_size
        ld    hl,b_init
        otir              ;init async port & interrupt vector
; Prepare to load hi bits of interrupt vector
        ld    a,I_Vector/256
        ld    i,a         ;set interrupt page for mode 2 ints
        im    2
        ei                ;let 'em rip!
Commutator_loop:
        call  TX_data
        call  Host_TX_data
        jp    Commutator_loop
;Now see if we need to start an output to RS-232 (host) port
Host_TX_data:
        ld    a,(out_started)
        or    a           ;also clears carry (see below)
        ret   nz          ;if output started, nothing to do
        in    a,(B_ctl)     ;look at RR0
        and   TBE         ;isolate the TBE bit
        ret   z
; else we should check to see if we need to start an output
        di
        call  CON_off           ;
        ld    hl,(out_head_cbuf)    ;grab current top of circ buf ptr
        ld    de,(out_tail_cbuf)    ;and where the next free buf ptr is
        ei
        or    a
        sbc   hl,de
        ret   z           ;if the same, nothing to do
;else we need to start an output
        di                ;interrupt protect this section,
        ld    hl,(out_head_cbuf)    ;get pointer to next cbuf to output
        ld    e,(hl)
        inc   hl
```

```
        ld    d,(hl)        ;DE has pointer to buffer chain
        ld    (out_chain_head),de    ;set in interrupt routine's place
        ld    a,TRUE
        ld    (out_started),a       ;yes. output started
        call  CON_on
        ld    a,FEND
        out   (B_dat),a     ;send FEND character (start txing)
        ei
        ret                 ;keep looking for new opportunity
TX_data:
        ld    a,(TX_State)
        or    a
        jp    z,txd0
        cp    1
        jp    z,txd1
        cp    2
        jp    z,txd2
        cp    3
        jp    z,txd3
        cp    4
        ret   z
;When tail timer times out, turn off the TX
        ld    a,(TX_Timer)
        or    a
        ret   nz
        ld    a,5           ;ready to write to WR5 of Ch A
        di                  ;must have atomic use of A_WR5 & SIO
        out   (A_ctl),a     ;Next char to A_ctl goes to WR5
        ld    a,(A_WR5)     ;grab A_WR5
        and   NOT RTS       ;turn off RTS bit there
        ld    (A_WR5),a     ;keep memory copy updated
        out   (A_ctl),a     ;and turn off TX now
        xor   a
        ld    (TX_State),a
        ei
        ret
txd0:
        ld    a,(TX_outstanding)    ;if there are no outstanding TX...
        or    a                     ;...frames, then we don't have to...
        ret   z                     ;...worry about Transmitter
; do persistence algorithm
```

```
        ld    a,r               ;grab the Z-80 refresh register
        add   a,a                ;double;now 0 <= A reg <= 254
        ld    b,a               ;B holds our "random" number
        ld    a,(Persistence)
        sub   b                 ;A reg = Persistence - Random #
        jp    c,No_PTT           ;if (P-r) < 0 then no PTT now
                                          ; Note that P=255 means ALWAYS key up
; else we've noticed that we've got some frame(s) to send.
; try to keyup TX
        ld    a,(Full_Duplex)
        or    a
        jp    nz,Key_Up          ;if Full Duplex, then there is no
                                          ;need to worry about all this silly
                                          ;slot time and persistence stuff!

        ld    a,(A_RR0)          ;A_RR0 is set in interrupt routine
        and   DCD
        jp    nz,No_PTT          ;If carrier active, wait it out
;OK, so we've won with the random number generator.  Keyup TX and start the
;TXdelay timer
Key_Up:
        ld    a,(TXdelay)
        ld    (TX_Timer),a       ;Get timer value into timer slot
        ld    a,5
        di                      ;we need quite time here.
        out   (A_ctl),a          ;Ready to write into WR5 of Ch A
        ld    a,(A_WR5)
        or    RTS               ;Turn on the PTT bit...
        ld    (A_WR5),a          ;...in the memory copy of WR5
        out   (A_ctl),a          ; Keyup transmitter
        ld    a,2
        ld    (TX_State),a
        ei
        ret                     ;That's all we do for now, we await
                                          ;TXdelay event
No_PTT: ;since we lost on Random #, wait SlotTime before trying again
        ld    a,(SlotTime)
        ld    (TX_Timer),a       ;Set up the timer value of this event
        ld    a,1
        ld    (TX_State),a
```

```
        ret
txd1:
        ld    a,(TX_Timer)
        or    a
        ret   nz
        xor   a
        ld    (TX_State),a
        ret
txd2:
        ld    a,(TX_Timer)
        or    a
        ret   nz
        ld    a,3
        ld    (TX_State),a
        ret
txd3:
        di
        ld    a,4
        ld    (TX_State),a
        call  TXnext_CBuf        ;gets HL to point to buffer chain, and
                                 ;sets TX_Chain_Head for the interrupt
                                 ;routine
        ld    a,80h
        out   (A_ctl),a          ;reset TX CRC
        call  getchar            ;getchar needs int. protection
        out   (A_dat),a          ;Ship this char to TX modem
        ld    a,TRUE
        ld    (TX_Started),a     ;and  we've started TX
        ld    a,0c0h
        out   (A_ctl),a          ;reset TX underrun/EOM latch
        ei
        ret
;-------------------------------------------------------------------------
ia_tbe:
        push  af
        push  hl
        ld    a,(TX_Started)
        or    a
        jp    z,ia_t2            ;previous frame finished
        ld    hl,(TX_Chain_Head)
        call  getchar
```

```
        ld    (TX_Chain_Head),hl    ;must keep this pointer updated
        jp    z,ia_t1               ;no more to send
        out   (A_dat),a             ;else ship this char out
ia_t9:
        pop   hl
        pop   af
        ei
        reti                        ;just return from these interrupts
ia_t1:
        halt                        ;if it gets here, halt
        xor   a
        ld    (TX_Started),a        ;TX is NOT started
        ld    hl,TX_Outstanding     ; make is so that one fewer frames
                                                ; NOT "(TX_Outstanding)" (!) 29 Sep
        dec   (hl)                  ; are outstanding
        ld    a,28h
        out   (A_ctl),a             ; reset TX interrupt pending
        jp    ia_t9
;previous frame is done, SIO now sending a flag.  More?
ia_t2:
        ld    a,(TX_Outstanding)
        or    a
        jp    nz,ia_t21             ;if more to send, go there
; else we're done here, clean up.
        ld    a,28h
        out   (A_ctl),a             ; Reset TX interrupt pending
;start Tail timer event
        ld    a,(TailTime)
        ld    (TX_Timer),a          ; wait for CRC to clear TX
        ld    a,5
        ld    (TX_State),a
        jp    ia_t9
ia_t21:          ;start up next frame
        call  TXnext_CBuf           ; get the next buffer chain pointer
                                                ; setup HL and TX_Chain_Head
        ld    a,80h
        out   (A_ctl),a             ; reset TX CRC generator
        call  getchar
        out   (A_dat),a             ;get 1st char of next frame
        ld    a,TRUE
        ld    (TX_Started),a  ; TX started again
```

```
; if no room, flush this frame (sigh)
ia_rc2:
        xor   a
        ld    (RX_Allocated_Buffer),a
        ld    hl,(RX_head)
        call  free_chain
ia_rc5:
        ld    a,TRUE
        ld    (RX_flushing),a ; we are in the midst of
                                        ; flushing this frame
        call  STA_on            ;ddd Note that we are in flushing
                                        ;state
        jp    ia_rc9
ia_special:
        push  af
        push  hl        ; regs we'll need
        ld    a,1
        out   (A_ctl),a     ; ready to read RR1
        in    a,(A_ctl)     ; OK, grab RR1
        bit   5,a       ; RX overrun?
        jp    nz,ia_sp8     ; If a problem, treat as bad CRC
                                ; That is, flush this frame....
;ia_sp0:
        bit   7,a       ; check state of End of Frame bit
        jp    z,ia_sp8      ; Else something weird happened - probably
                                ; RX overrun. In any case, flush this frame.
                                ; error reset & then exit
                                ; that is, treat like it was a CRC error
; If End of Frame, check CRC bit for valid.
ia_sp1:
        bit   6,a       ; Check CRC error bit
        jp    nz,ia_sp8     ; If CRC error bit is on, then was CRC error
; First ensure that we indeed have a buffer allocated...
        ld    a,(RX_Allocated_Buffer)
        or    a
        jp    z,ia_sp9      ; if no buffer allocated, ignore this.
        ld    hl,(RX_head)
        call  out_queue_insert      ; Shove this buffer string onto
                                        ; output queue
        xor   a
        ld    (RX_Allocated_Buffer),a ; We don't have a buffer
```

```
                                                    ; allocated for the next
                                                    ; frame...
        jp      ia_sp9
; get here if there was a bad CRC
ia_sp8:
        ld      a,(RX_Allocated_Buffer) ; If we don't have any
                                                    ; buffers allocated, then
        or      a               ;
        jp      z,ia_sp9        ; we MUST NOT "release" them
                                        ; if they are not allocated
        xor     a
        ld      (RX_Allocated_Buffer),a
                                        ; not receiving if we have bad CRC
        ld      hl,(RX_head)
        call    free_chain      ; free up all buffer(s)
ia_sp9:
        ld      a,30h           ; error reset
        out     (A_ctl),a
        in      a,(A_dat)       ; Avoid spurious RCA interrupt
        ld      a,03h           ; select WR3
        out     (A_ctl),a       ;
        ld      a,0D9h          ; enter hunt mode
        out     (A_ctl),a       ;
        xor     a
        ld      (RX_State),a    ; store sync/hunt state
        ld      (RX_flushing),a
        pop     hl
        pop     af
        ei
        reti
ia_ext:
        push    af
        ld      a,10h           ; reset ext/status interrupts
        out     (A_ctl),a
        in      a,(A_ctl)       ; grab RR0
        ld      (A_RR0),a
        bit     4,a             ; check sync/hunt bit
        jp      nz,ia_ex1       ; no need to worry, if not zero
        ld      a,(RX_State)    ; it is 0! Did it change?
        or      a               ;
        jp      nz,ia_ex9       ; no, this is a DCD,CTS or EOM-interrupt
```

```
        ld      a,TRUE          ; indeed, it changed!
        ld      (RX_State),a    ; next time, we'll know
        ld      a,(RX_Allocated_Buffer) ; if we are not in the
                                                        ; receiving state...
        or      a               ; then there are no allocated buffers and...
        jp      z,ia_ex9        ; we MUST NOT "release" them
                                        ; if no buffers allocated
        xor     a
        ld      (RX_Allocated_Buffer),a ; not receiving
        push    hl
        ld      hl,(RX_head)
        call    free_chain              ; free up all buffer(s)
        pop     hl
        jp      ia_ex9
ia_ex1:
        xor     a               ;  Prepare for next frame start
        ld      (RX_State),a    ;
        ld      (RX_flushing),a
ia_ex9:
        pop     af
        ei
        reti
sync_hunt       equ     10h
ib_ext:
        ex      af,af'
        exx                     ; we want the other registers
        ld      a,10h
        out     (B_ctl),a       ; reset ext/status interrupts
        in      a,(B_ctl)       ; grab RR0
        ld      d,a             ; Hold it for a moment...
        and     sync_hunt       ; isolate this bit
        jp      z,ib_s0
;else sync/hunt is a 1
        ld      a,c
        or      a
        jp      z,ib_s1 ; go here if state of sync/hunt changed
; Here if sync/hunt bit did NOT change - maybe something else did....
ib_s9:
        ld      a,d             ; retreive RRO from above
        and     Break_Abort     ; Check if we are doing a break/abort thing
        jp      z,ib_NBA        ; There if No break/abort
```

```
; Else Break/Abort bit on, note state change...
        ld    a,TRUE
        ld    (in_break),a    ; save in mem  (probably can use E reg...)
        in    a,(B_dat)       ; clear out any null character from buffer
        jp    ib_BOK  ; Break OK for now...
ib_NBA: ;if no break/abort, check if we are in break/abort state.
        ld    a,(in_break)
        or    a
        jp    z,ib_BOK        ; Nothing going on, Break OK
; Else we were in break mode, and this is the tail end of a break.
        xor   a
        ld    (in_break),a
        in    a,(B_dat)       ; discard the single extraneous null
ib_BOK:
ib_s99:
        ex    af,af'
        exx
        ei
        reti                  ; else something else & we don't care
ib_s0:          ; sync/hunt is a 0
        ld    a,c
        or    a
        jp    nz,ib_s1a       ; go here if sync/hunt changed
        jp    ib_s9   ; else not interested, forget it
;get here if state of sync/hunt changed
ib_s1:
        ld    c,1
        jp    ib_s1b
ib_s1a:                 ; first fix up C for next tick
        ld    c,0
ib_s1b:
; Here when we've seen a real "clock tick" & dealt with C reg
        inc   b
        ld    a,b
        cp    12
        jp    nz,ib_s99             ; we act on every 12th clock tick...
        ld    b,0                   ; so reload divisor. This give us an
                                            ; effective interrupt rate of 100 Hz

; Decrement all the timers
```

```
        ld    a,(TX_Timer)          ; Get value, and ...
        or    a
        jp    z,ib_s1c
        dec   a                     ; ... decrement it as required.
        ld    (TX_Timer),a
ib_s1c:
        jp    ib_s99
;
;---------------------------------------------------------------
ib_special:
        push  af
ib_sp9:             ; Normal exit
        ld    a,30h         ; error reset
        out   (B_ctl),a
        pop   af
        ei
        reti
;---------------------------------------------------------------
; The TX has become empty, shove a new character out
ib_tbe:
        halt
        push  af          ; new char will return in A
        push  hl
        ld    a,(Out_esc_mode)
        or    a
        jp    z,ib_t1 ; not escaped, so go here
; else we are escaped, so send escaped char
        ld    a,(Out_char)    ; char which follows escape
        or    a
        jp    z,ib_t2 ; special case if at end of frame, clean up
        out   (B_dat),a
        xor   a
        ld    (Out_esc_mode),a      ; get out of escaped mode
        jp    ib_t9         ; all for now...
ib_t1:
        ld    hl,(out_chain_head)    ; we are currently on this buffer,
        call  getchar               ; as getchar() needs to know
        ld    (out_chain_head),hl    ; maybe HL changed,so save it in case
        jp    z,ib_tdone     ; if no more chars, deal with this
        cp    FESC
        jp    z,ib_t1a       ; deal with FESC char in data stream
```

```
        cp    FEND
        jp    z,ib_t1b      ; deal with FEND char in data stream
; else this char is nothing special, so shove it out

        out   (B_dat),a     ; shove it out
        jp    ib_t9  ; if this is not last char, all for now
; else this is last char, send FEND
ib_tdone:
        ld    a,FEND
        out   (B_dat),a
        ld    a,TRUE
        ld    (Out_esc_mode),a      ; set special escaped mode by...
        xor   a
        ld    (Out_char),a          ;... making escaped char a 0
        jp    ib_t9         ; all till TX Buffer goes empty again.
; here if are completely done sending frame
ib_t2:
        push  de            ; need this for a moment
        ld    hl,(out_head_cbuf)
        inc   hl
        inc   hl
        ld    de,out_bottom
        or    a
        push  hl
        sbc   hl,de
        pop   hl                  ; this may be the one we want
        pop   de
        jp    nz,ib_t2a             ; yes it is!
        ld    hl,Out_Top            ; else, make a circular buffer
ib_t2a:
        ld    (out_head_cbuf),hl    ; we will work on this one next
        xor   a
        ld    (out_started),a       ; not doing outputs anymore
        ld    (Out_esc_mode),a      ; !! NOT IN ESCAPED MODE ANYMORE !!
        ld    a,28h                 ; NEEDED for ASYNC
        out   (B_ctl),a             ; reset TX interrupt pending
ib_t9:
        pop   hl
        pop   af
        ei
        reti                        ; now get our butts out of here...
```

```
; here is FESC in data stream
ib_t1a:
        out   (B_dat),a         ; Ship FESC character to port
        ld    a,TFESC           ; ready what will be next char
ib_t1z:
        ld    (Out_char),a      ; set char for next time
        ld    a,TRUE
        ld    (Out_esc_mode),a  ; we are in escaped mode
        jp    ib_t9             ; all for now
; here is FEND in data stream
ib_t1b:
        ld    a,FESC
        out   (B_dat),a
        ld    a,TFEND
        jp    ib_t1z            ; rest is same as FESC case
;-----------------------------------------------------------
; Got a char from the TTY port, deal with it.
ib_rca:
        push  af
        in    a,(B_ctl)         ; Read RR0: force reg pointer to be 0
        ld    a,1
        out   (B_ctl),a         ; ready to read RR1
        in    a,(B_ctl)         ; Grab RR1
        and   Framing_Error     ; Isolate the FE bit
        halt
        jp    z,ib_Rtop         ; No Framing Error, so process this char
; Else we have a Framing Error - Ignore this char & flush this frame...
        call  STA_off           ; Off with the LED!
        in    a,(B_dat)         ; Flush erroneous character
        xor   a
        ld    (In_state),a      ; Force receiver to look for FEND
        ld    a,(In_Allocated_Buffer)
        or    a
        jp    z,ib_rc9          ; If no buffer is allocated, done; Exit.
; Else we were receiving a data SLIP frame, so flush it.
        push  hl
        ld    hl,(In_head)
        call  free_chain        ; Dump these buffers back to free list
        pop   hl
        jp    ib_rc9 ; And get out of here!
ib_rTop:
```

```
        ld      a,(In_state)    ; get our state machine value
        or      a
        jp      z,ib_r0 ; in state 0, waiting for FEND
        cp      1
        jp      z,ib_r1 ; in state 1, saw FEND
        cp      2
        jp      z,ib_r2 ; in state 2, data to follow
        cp      3
        jp      z,ib_r3 ; saw FESC, expecting TFESC or TFEND
        cp      10
        jp      z,ib_r10        ; Expecting TXdelay
        cp      20
        jp      z,ib_r20        ; Expecting P value
        cp      30
        jp      z,ib_r30        ; Expecting SlotTime value
        cp      40
        jp      z,ib_r40        ; Expecting TailTime value
        cp      50
        jp      z,ib_r50        ; Expecting Full/Half duplex value
;else we don't know what happened, ignore it.
ib_rcjunk:
        in      a,(B_dat)
ib_rcFEND:
        xor     a
ib_rcSTATE:
        ld      (In_State),a    ;go into In_State 0, FEND hunt
ib_rc9:
        pop     af              ; throw it away, we don't need junk
        ei
        reti
; Here if we are hunting for FEND character
ib_r0:
        call    STA_off
        in      a,(B_dat)
        cp      FEND
        jp      nz,ib_rc9       ; if we didn't see an FEND, keep looking
; else is an FEND, change state
        ld      a,1
        jp      ib_rcSTATE
; Get here if we've seen FEND character; look for command byte
ib_r1:
```

```
        call   STA_off
        in     a,(B_dat)
        cp     FEND
        jp     z,ib_rc9       ; Just another FEND, keep looking for cmd
        call   STA_on          ;getting valid SLIP; show in STA LED
; Here if we DO NOT have an FEND (expecting command byte)
                cp     0ffh
                jp     z,kiss_exit
                and    0fh
        jp     z,ib_r1a        ; 0 command means data will follow
        cp     1
        jp     z,ib_r1b        ; 1 command means TXdelay will follow
        cp     2
        jp     z,ib_r1c        ; 2 command means P(Persistence) will follow
        cp     3
        jp     z,ib_r1d        ; 3 command means Slot Time will follow
        cp     4
        jp     z,ib_r1e        ; 4 command means TailTime to follow
        cp     5
        jp     z,ib_r1f        ; 5 command means Full/Half duplex to come
; Here if we receive bogus command byte, flush rest of frame
        call   STA_off         ;bogosity, so turn off STA LED
        jp     ib_rcFEND
; exit kiss mode.
kiss_exit:
                ld     hl,(000eh)
                res    4,(hl)
                ld     hl,0000h
                push   hl
                ld     hl,(0019h)
                jp     (hl)
; Data are expected, change state
ib_r1a:
        ld     a,2
        jp     ib_rcSTATE
; TXdelay to follow, change state
ib_r1b:
        ld     a,10
        jp     ib_rcSTATE
; P to follow, change state
ib_r1c:
```

```
        ld      a,20
        jp      ib_rcSTATE
; SlotTime to follow, change state
ib_r1d:
        ld      a,30
        jp      ib_rcSTATE
; TailTime to follow, change state
ib_r1e:
        ld      a,40
        jp      ib_rcSTATE
; Full/Half Duplex to follow, change state
ib_r1f:
        ld      a,50
        jp      ib_rcSTATE
; These bytes are data
ib_r2:
        in      a,(B_dat)
        cp      FEND
        jp      z,ib_r2b        ; FEND means to queue this buffer
        push    af              ; Save the char we read on stack for a bit..
        ld      a,(In_Allocated_Buffer)
        or      a
        jp      nz,ib_r2c       ; if we already allocated buffer
        push    hl
        call    allocate_buffer ; get our initial buffer to mess with
        jp      nz,ib_r22
; else no room, flush this frame
        pop     af
        pop     hl              ; keep stack tidy
        jp      ib_rcFEND
ib_r22:
        ld      a,TRUE
        ld      (In_Allocated_Buffer),a ; make ourselves active
        ld      (In_buffer),hl
        ld      (In_head),hl    ; save current & head of chain pointers
        pop     hl
ib_r2c:
        pop     af              ; Retreive the data char we just got...
        cp      FESC
        jp      z,ib_r2a        ; If FESC in data stream, switch state
        push    hl
```

```
        ld    hl,(In_buffer)
        call  putchar       ; shove this character into our buffer
        jp    nc,ib_r2ca
        xor   a
        ld    (In_Allocated_buffer),a
        ld    hl,(In_head)
        call  free_buffer
        pop   hl
        jp    ib_rcFEND
ib_r2ca:
        ld    (In_buffer),hl  ; save in case HL changed
        pop   hl
        jp    ib_rc9  ; done so far
; FESC character seen while grabbing data
ib_r2a:
        ld    a,3
        jp    ib_rcSTATE
; FEND character seen while grabbing data
ib_r2b:
        ld    a,(In_Allocated_Buffer)
        or    a
        jp    z,ib_r2z      ; No bytes accumulated, so is null frame
; else we must ship this frame to TX
        push  hl            ; This bug found 29 Sep (must save HL !!!)
        ld    hl,(In_Buffer)
        call  putchar       ; put a garbage character at the end of
                            ; last buffer because getchar() will strip
                            ; it. Hack needed because of RX use of
                            ; putchar/getchar.
        ld    hl,(In_head)
        jp    nc,ib_r2za
        call  free_chain
        jp    ib_r2zb
ib_r2za:
        call  TX_queue_insert
ib_r2zb:
        pop   hl
        xor   a
        ld    (In_Allocated_Buffer),a ; input no longer active
ib_r2z:             ; entry point for null frame
        call  STA_off       ;done getting this frame, turn STA LED off
```

```
        ld    a,1          ; Keep as was, FENDs only at end in v.32
        jp    ib_rcSTATE
; here if we've seen FESC in data stream
ib_r3:
        in    a,(B_dat)
        cp    TFESC
        jp    z,ib_r3a
        cp    TFEND
        jp    z,ib_r3b
        ld    a,2
        jp    ib_rcSTATE
ib_r3a:
        ld    a,FESC
ib_r3z:
        push  hl
        ld    hl,(In_buffer)
        call  putchar
        jp    nc,ib_r3za
        xor   a
        ld    (In_Allocated_buffer),a
        ld    hl,(In_head)
        call  free_buffer
        pop   hl
        jp    ib_rcFEND
ib_r3za:
        ld    (In_buffer),hl
        pop   hl
        ld    a,2
        jp    ib_rcSTATE
; Here if we've seen TFEND after FESC in data stream; write FEND
ib_r3b:
        ld    a,FEND
        jp    ib_r3z       ; rest is same as for TFESC case
; This character is interpreted as TXdelay
ib_r10:
        in    a,(B_dat)
        ld    (TXdelay),a
        jp    ib_rcFEND
; This charcter is P, Persistence value
ib_r20:
        in    a,(B_dat)
```

```
        ld      (Persistence),a
        jp      ib_rcFEND
; This character is SlotTime value
ib_r30:
        in      a,(B_dat)
        ld      (SlotTime),a
        jp      ib_rcFEND
ib_r40:
        in      a,(B_dat)
        ld      (TailTime),a
        jp      ib_rcFEND
; This character is Full/Half Duplex value
; 0 means Half Duplex, non-zero means Full Duplex
ib_r50:
        in      a,(B_dat)
        ld      (Full_Duplex),a
        jp      ib_rcFEND
;       CTS flow
ib_r60cts:
        sub     0feh
        ld      (CTS_Control),a
        jp      ib_rcFEND
;       Software DCD
ib_r60dcd
        sub     0f8h
        ld      (Soft_DCD),a
        jp      ib_rcFEND
** note!! This code not present memory management.
Original code has dynamic allocation memory.
```

PRODUCT SPECIFICATION

Z8440/1/2/4, Z84C40/1/2/3/4

SERIAL INPUT/OUTPUT CONTROLLER

FEATURES

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rate in the x1 clock mode of 0 to 2.0M bits/second with a 10 MHz clock.
- NMOS version for cost sensitive performance solutions, CMOS version for the designs requiring low power consumption
- NMOS Z0844x04 - 4 MHz Z0844x06 - 6.17 MHz (Where x is the designator for the bonding option; 0, 1, 2 or 4)
- CMOS Z84C4x06 - DC to 6.7 MHz, Z84C4x08 - DC to 8 MHz, Z84C4x10 - DC to 10 MHz (Where x is the designator for the bonding option; 0, 1, 2 , 3 or 4)
- 6 MHz version supports 6.144 MHz CPU clock operation.
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7, or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7, or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection, and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

GENERAL DESCRIPTION

The Z80 SIO (here in after referred to as the Z80 SIO or, SIO). Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs, and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast, or slow, peripheral devices. While designed primarily as a member of the Z80 family, its versatility makes it well suited to many other CPUs.

The Z80 SIO uses a single +5V power supply and the standard Z80 family single-phase clock. The SIO/0, SIO/1, and SIO/2 are packaged in a 40-pin DIP, the SIO/4 is packaged in a 44-pin PCC and the SIO/3 is packaged in a 44-pin QFP. Note that SIO/3 is only available in CMOS and in QFP package.

PIN DESCRIPTION

Figures 1 through 6 illustrate the three 40-pin configurations (bonding options) available in the Z80C SIO (hereafter referred to as SIO or Z80 SIO). The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together:

- Z80 SIO/2 lacks $\overline{SYNCB}$
- Z80 SIO/1 lacks $\overline{DTRB}$
- Z80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The 44-pin package, the Z80 SIO/4 for PLCC package, and Z80 SIO/3 for QFP, has all options (Figure 7a and 7b).

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data

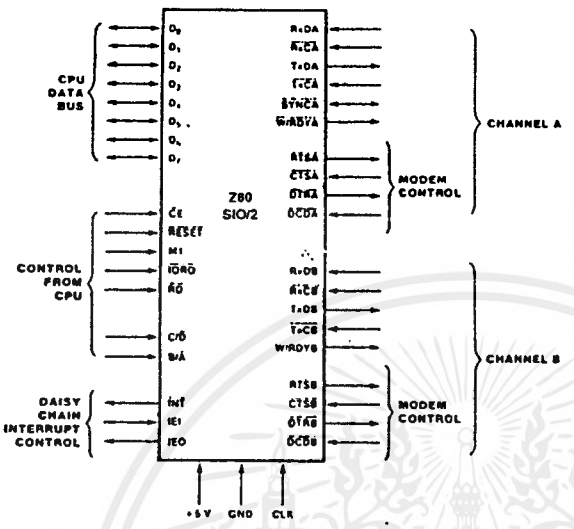


Figure 1. Pin Functions

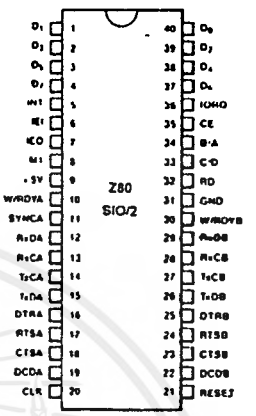


Figure 2. 40-pin Dual-In-Line Package (DIP). Pin Assignments

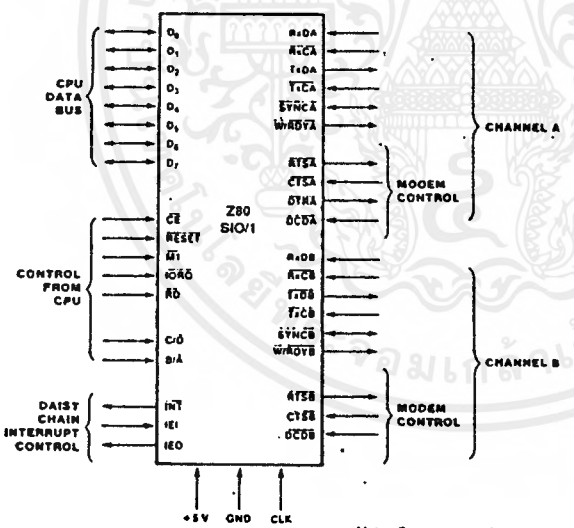


Figure 3. Pin Functions

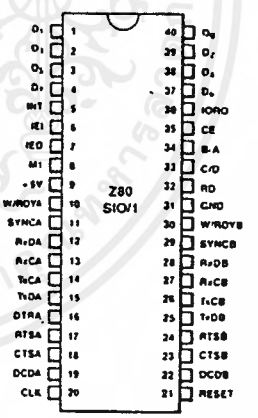


Figure 4. 40-pin Dual-In-Line Package (DIP), Pin Assignments

Note: Power connections follow conventional descriptions below:

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

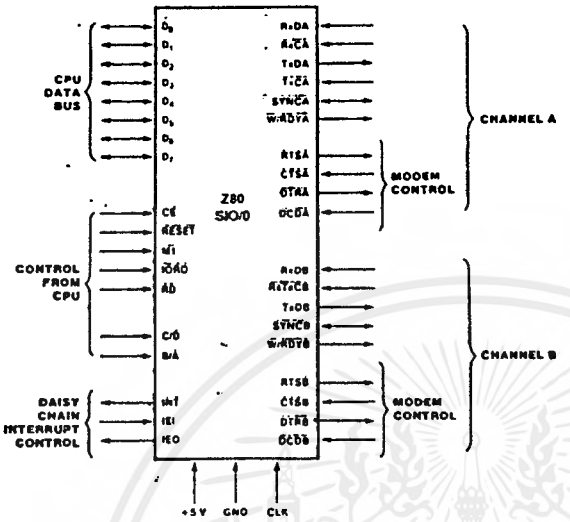


Figure 5. Pin Functions

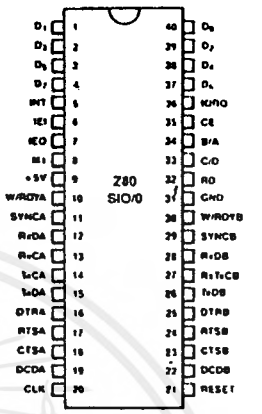


Figure 6. 40-pin Dual-In-Line Package (DIP), Pin Assignments

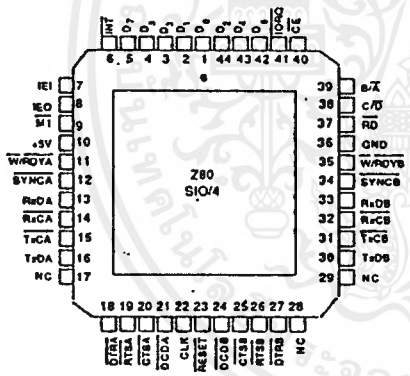


Figure 7a. 44-pin Chip Carrier, Pin Assignments

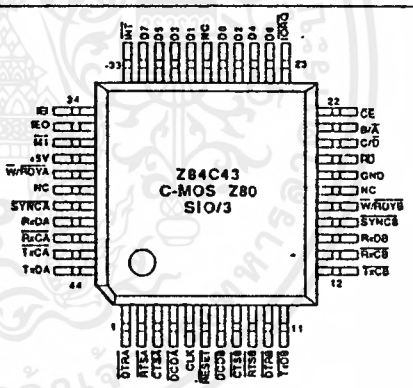


Figure 7b. 44-pin Quad Flat Pack Pin Assignments

transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z80 System Clock to synchronize internal signals. This is single-phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffering does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the Z80 SIO. They can also be programmed as general-purpose outputs.

In the Z80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. As mentioned previously, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle One* (input from Z80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32, or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered; no noise level margin is specified.

In the Z80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High, and disables all interrupts. The control registers must be rewritten after the SIO is reset and before data is transmitted or received.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (bidirectional, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. Once $\overline{SYNC}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{RxC}$ that immediately precedes the falling edge of $\overline{SYNC}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{RxC}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z80 SIO/2 bonding option, $\overline{SYNCB}$ is omitted.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32, or 64 times the data rate; however, the clock multiplier must be the same for the transmitter and the receiver. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements; no noise level margin is specified. Transmitter Clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z80 SIO/0 bonding option, $\overline{TxCB}$ is bonded together with $\overline{RxCB}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{TxC}$

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function; driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

FUNCTIONAL DESCRIPTION

The functional capabilities of the Z80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z80 family peripheral, it interacts with the Z80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling, and simple handshake capability. Figure 8 is a block diagram.

Figure 9 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

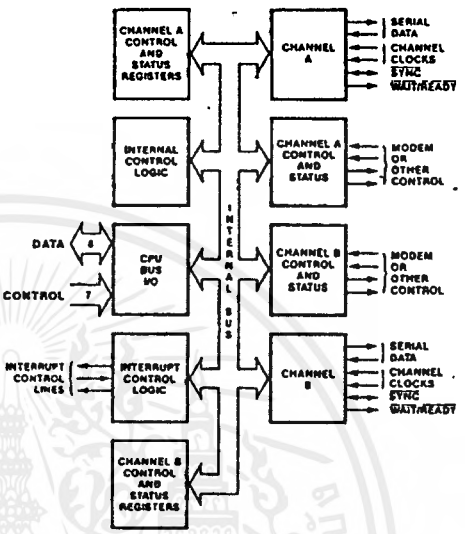


Figure 8. Block Diagram

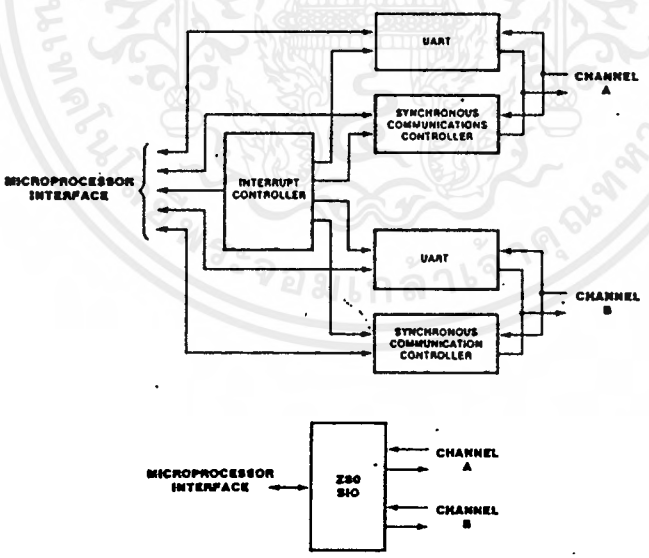


Figure 9. Conventional Devices Replaced by the Z80 SIO

DATA COMMUNICATION CAPABILITIES

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous, or synchronous data-communication protocol. Figure 10a illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z80 SIO Technical Manual* (03-3033-01).

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist, as in the case of a transient, the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit; a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals, a feature that allows it to be used with a Z80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{SYNC}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync characters can be removed without interrupting the CPU.

Five-, six-, or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple incoming sync characters, as shown in Figure 10b.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Figure 10a. Some Z80 SIO Protocols

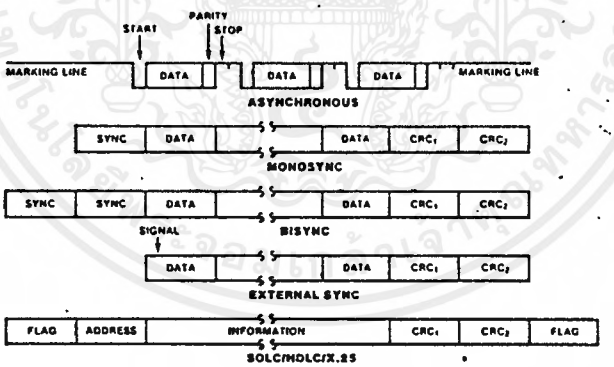


Figure 10b. Six-Bit Sync Character Recognition

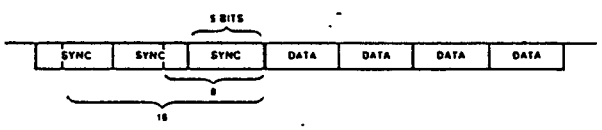


Figure 10. Data Communication

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0s; in SDLC modes, it is initialized to 1s. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disks, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin. An interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

I/O INTERFACE CAPABILITIES

The SIO offers the choice of polling, vectored or non-vectored interrupts and block-transfer modes to transfer data, status, and control information to, and from, the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts, and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit, and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example, receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{\text{CTS}}$), Data Carrier Detect ($\overline{\text{DCD}}$), and Synchronization ($\overline{\text{SYNC}}$) pins (Figures 1 through 7). In addition, an external/status

interrupt is also caused by a CRC-sending condition, or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

In a Z80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{\text{WAIT}}$ line in the CPU block-transfer mode or as a $\overline{\text{READY}}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to, or from, memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

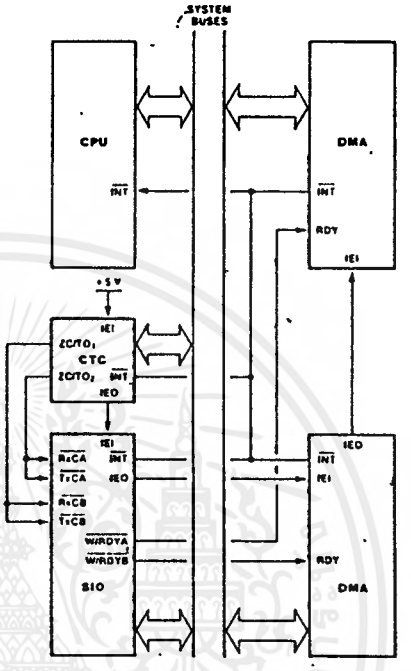


Figure 11. Typical Z80 Environment

INTERNAL STRUCTURE

The internal structure of the device includes a Z80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{\text{CTS}}$) and Data Carrier Detect ($\overline{\text{DCD}}$), are

Table 1. Register Functions

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)
Write Register Functions	
WR0	Register pointers; CRC initialize, and initialization commands for the various modes.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

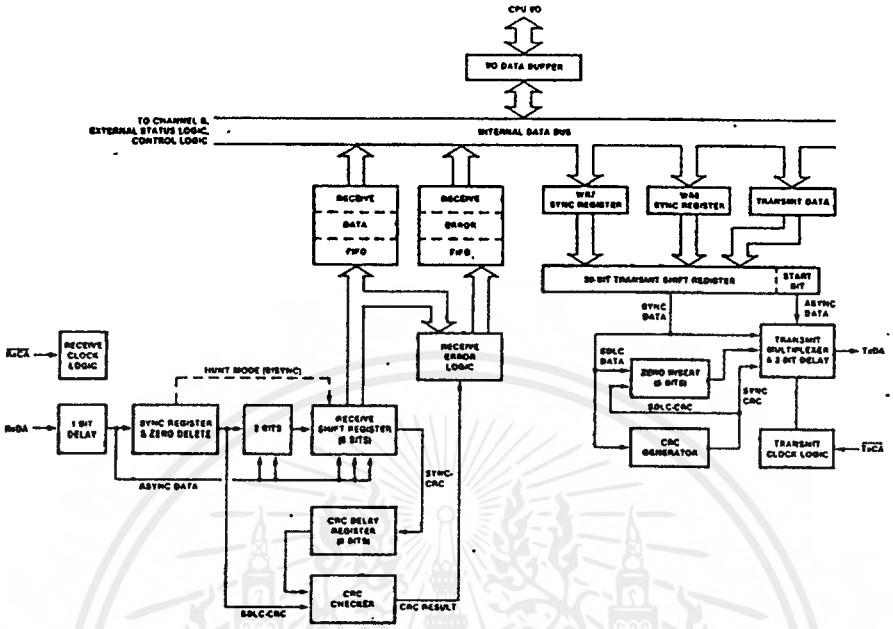


Figure 12. Transmit and Receive Data Path (Channel A)

PROGRAMMING

The system program first issues a series of commands that initialize the basic mode of operation and then issues other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information: RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector, and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels: WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

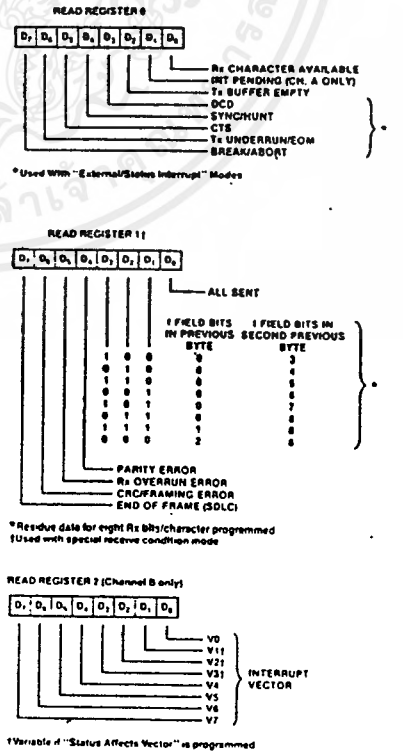


Figure 13. Read Register Bit Functions

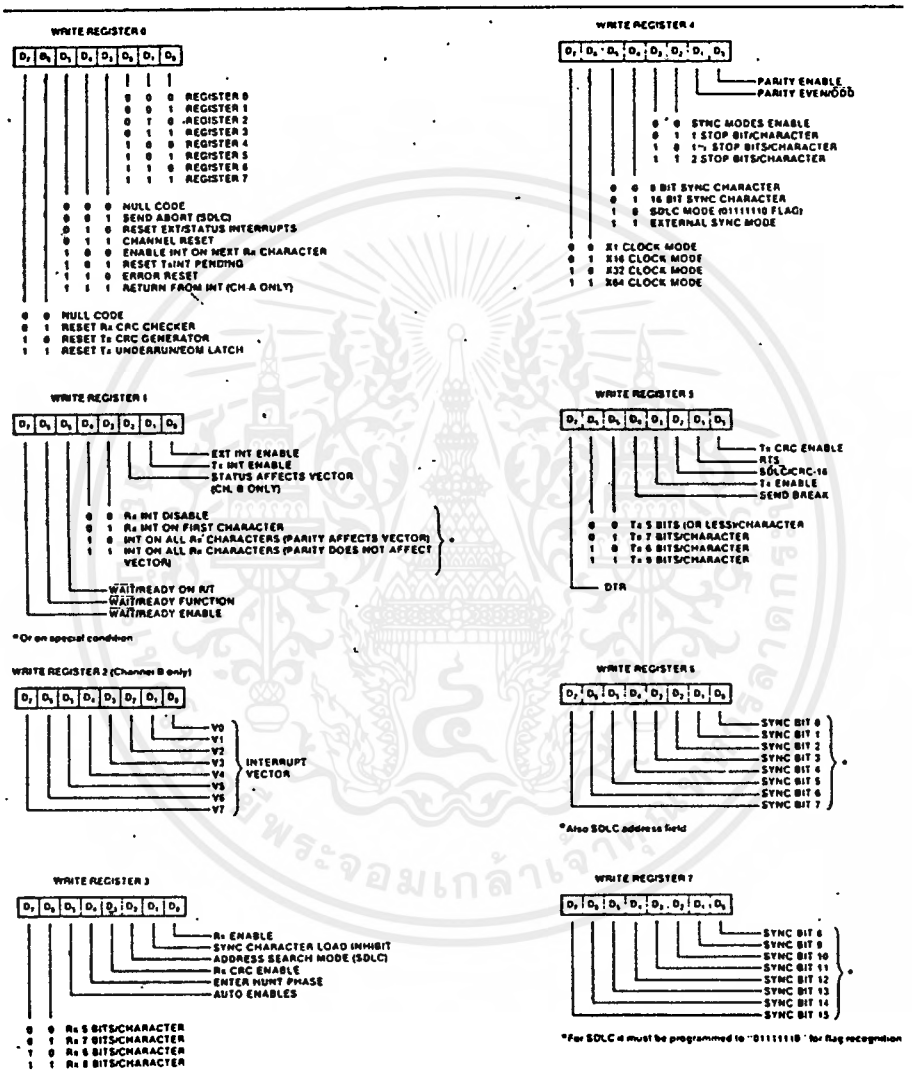


Figure 14. Write Register Bit Functions

TIMING

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z80 CPU sends an interrupt-acknowledge sequence. $\overline{M1}$ Low and $\overline{IORQ}$ Low, a few cycles later (Figure 17).

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z80 CPU issues a Return From Interrupt (RETI) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever ED is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is 4D, the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z8400 Z80 CPU Product Specification* (00-2001-04).

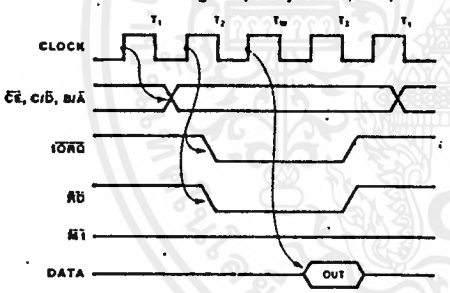


Figure 15. Read Cycle

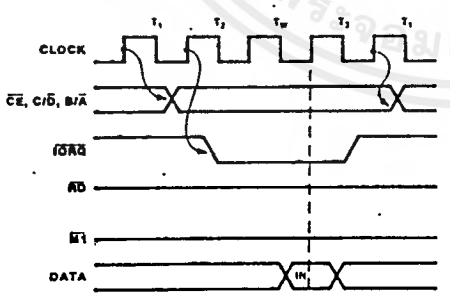


Figure 16. Write Cycle

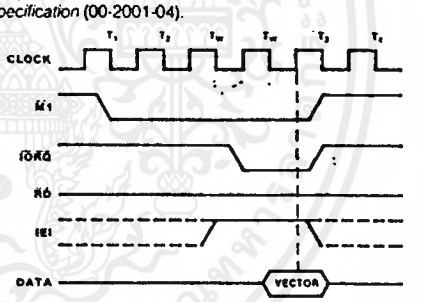


Figure 17. Interrupt Acknowledge Cycle

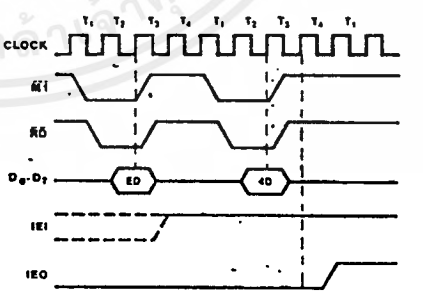


Figure 18. Return from Interrupt Cycle

ABSOLUTE MAXIMUM RATINGS

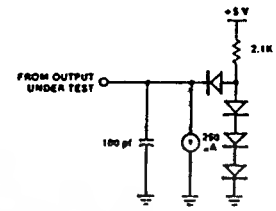
Voltages in V_{CC} with respect to V_{SS} . . . . . – 0.3V to + 0.7V
Voltages on all inputs with respect to V_{SS} . . . . . – 0.3V to V_{CC} + 0.3V
Storage Temperature . . . . . – 65°C to + 150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin. Available operating temperature range is:

- **S = 0°C to +70°C, V_{CC} Range**
  - **NMOS: +4.75V ≤ V_{CC} ≤ +5.25V**
  - **CMOS: +4.50V ≤ V_{CC} ≤ +5.50V**
- **E = -40°C to 100°C, =4.50V ≤ V_{CC} ≤ +5.50V**



DC CHARACTERISTICS

Z84C40 CMOS Z80 SIO, Z84C40/41/42/43/44 DC CHARACTERISTICS
V_{CC}=5.0V ± 10%, unless otherwise specified

Symbol	Parameter	Min	Max	Typ	Unit	Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45		V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	V_{CC}+0.3		V	
V_{IH}	Input High Voltage	2.2	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8		V	
V_{OL}	Output Low Voltage		0.4		V	I_{LO}=2.0mA
V_{OH1}	Output High Voltage	2.4			V	I_{OH}=-1.6mA
V_{OH2}	Output High Voltage	V_{CC}-0.8			V	I_{OH}=-250µA
I_{LI}	Input Leakage Current	-10	10		µA	V_{IN}=0.4V to V_{CC}
I_{LO}	3-state Output Leakage Current in Float	-10	10		µA	V_{OUT}=0.4V to V_{CC}
I_L(SY)	SYNC Pin Leakage Current	-40	10		µA	
I_{CC1}	Power Supply Current - 4MHz		10 [1]	7	mA	V_{CC}=5V
	- 6MHz		10 [1]	7	mA	CLK=4,6,8,10MHz
	- 8MHz		12 [1]	8	mA	V_{IH}=V_{CC}-0.2V
	- 10MHz		15 [1]	8	mA	V_{IL}=0.2V
I_{CC2}	Standby Supply Current		10		µA	V_{CC}=5V CLK=(0) V_{IH}=V_{CC}-0.2V V_{IL}=0.2V

Note:
[1] Measurements made with outputs floating.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C	Clock Capacitance		7	pf
C_{IN}	Input Capacitance		5	pf
C_{OUT}	Output Capacitance		10	pf

Over specified temperature range, f = 1 MHz.
Unmeasured pins returned to ground.

AC CHARACTERISTICS*

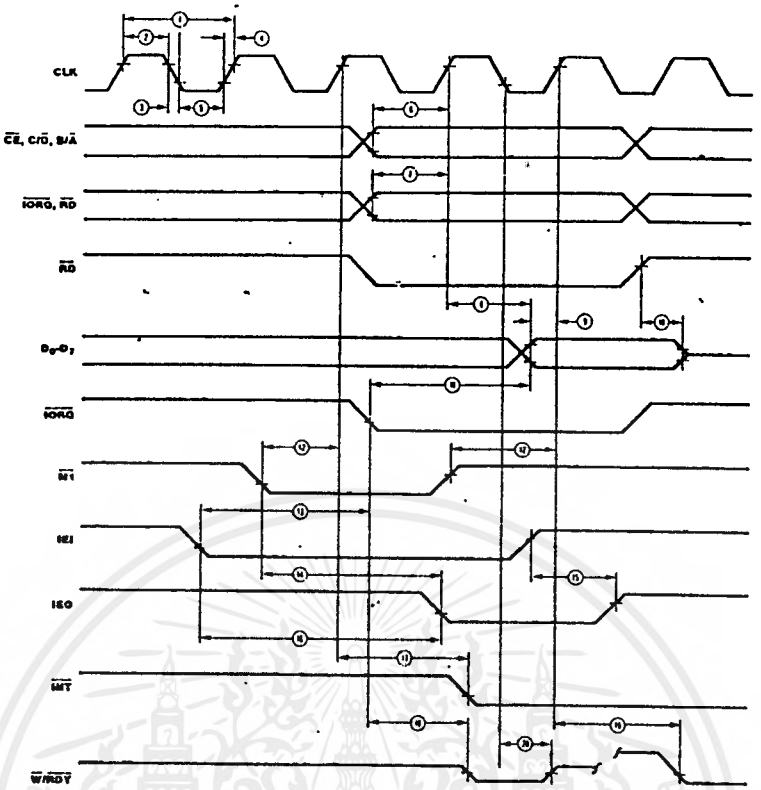
Z84C40/41/42/43/44 AC CHARACTERISTICS

No	Symbol	Parameter	Z84C4X04* Min	Z84C4X04* Max	Z84C4X06 Min	Z84C4X06 Max	Z84C4X08 Min	Z84C4X08 Max	Z84C4X10 Min	Z84C4X10 Max	Note
1	TcC	Clock Cycle Time	250	DC	162	DC	125	DC	100	DC	
2	TwCh	Clock Pulse Width (High)	105	DC	65	DC	55	DC	42	DC	
3	TfC	Clock Fall Time		30		20		10		10	
4	TrC	Clock Rise Time		30		20		10		10	
5	TwCl	Clock Pulse Width (Low)	105	DC	65	DC	55	DC	42	DC	
6	TsAD	/CE,B//A,C//D to Clock Rise Setup Time	145		60		40		35		
7	TsCS(C)	/IORQ, /RD to Clock Rise	115		60		40		35		
8	TdC(DO)	Clock Rise to Data Out Delay		220		150		100		85	
9	TsDI(C)	Data In to Clock Rise Setup Time (Write or /M1 Cycle)	50		30		20		20		
10	TdRD(DOz)	/RD Rise to Data Out Float Delay		110		90		75		65	
11	TdIO(DOI)	/IORQ Fall to Data Out Delay (/INTACK Cycle)		160		120		90		80	
12	TsM1(C)	/M1 to Clock Rise Setup Time	90		75		55		40		
13	TsIEI(IO)	IEI to /IORQ Fall Setup Time (/INTACK Cycle)	140		120		80		60		
14	TdM1(IEO)	M1 Fall to IEO Fall Delay (Interrupt Before /M1)		190		160		130		100	
15	TdIEI(IEOr)	IEI Rise to IEO Rise Delay (After ED Decode)		100		70		60		50	
16	TdIEI(IEOf)	/M1 Fall to IEO Fall Delay		100		70		60		50	
17	TdC(INT)	Clock Rise to /INT Fall Delay		200		150		120		100	
18	TdIO(W/RWf)	/IORQ or /CE Fall to /W//RDY Delay (Wait Mode)		210		175		130		110	
19	TdC(W/RR)	Clock Rise to /W//RDY Delay (Ready Mode)	120		100		90		85		
20	TdC(W/RWz)	Clock Fall to /W//RDY Float Delay (Wait Mode) When Setup Is Specified		130		110		90		80	
21	Th	Any Unspecified Hold	0		0		0		0		

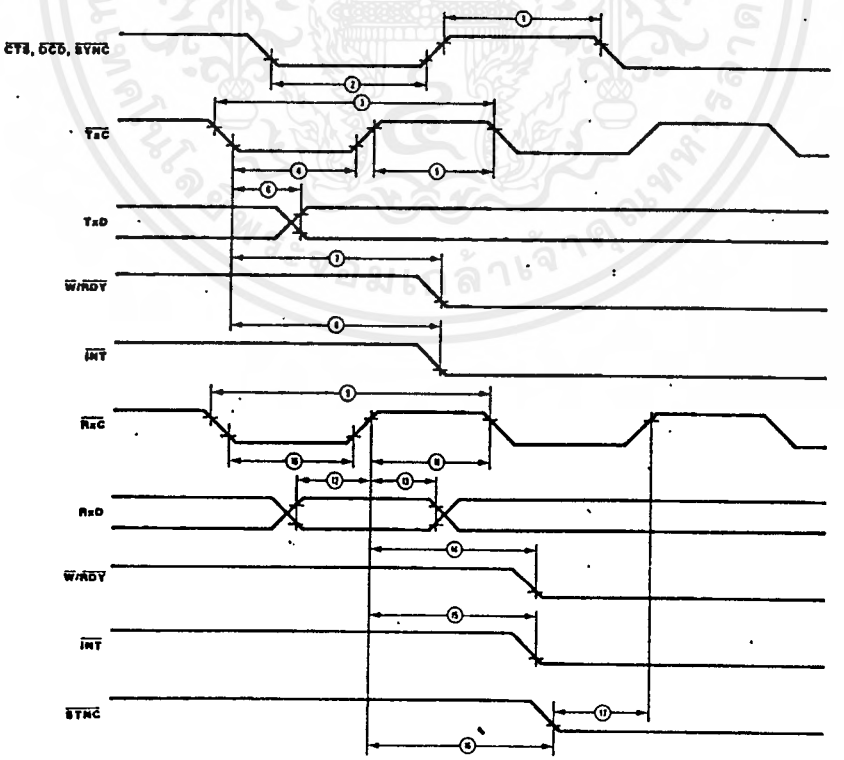
Note:
* Units in nanoseconds (nS).

* 4 MHz 84C4x is obsoleted and replaced by 6 MHz.

AC CHARACTERISTICS TIMING (Z84C4X CMOS Z80 SIO)



AC CHARACTERISTICS TIMING (Z84C4X CMOS Z80 SIO; Continued)



AC CHARACTERISTICS (Z84C4X CMOS Z80 SIO; Continued)

Z84C40/41/42/43/44 AC CHARACTERISTICS

No	Symbol	Parameter	Z84C4X04* Min	Z84C4X04* Max	Z84C4X06 Min	Z84C4X06 Max	Z84C4X08 Min	Z84C4X08 Max	Z84C4X10 Min	Z84C4X10 Max	Note
1	TwPh	Pulse Width (High)	200		200		150		150		[2]
2	TwPl	Pulse Width (Low)	200		200		150		150		[2]
3	TcTxC	/TxC Cycle Time	400		330		250		200		[2]
4	TwTxCl	/TxC Width (Low)	180		100		85		80		[2]
5	TwTxCh	/TxC Width (High)	180		100		85		80		[2]
6	TdTxC(TxD)	/TxC Fall to TxD Delay		300		220		160		120	[2]
7	TdTxC(W/RRI)	/TxC Fall to /W//RDY Fall Delay (Ready Mode)	5	9	5	9	5	9	5	9	[1]
8	TdTxC(INT)	/TxC Fall to /INT Fall Delay	5	9	5	9	5	9	5	9	[1]
9	TcRxC	/RxC Cycle Time	400		330		250		200		[2]
10	TwRxCl	/RxC Width (Low)	180		100		85		80		[2]
11	TwRxCh	/RxC Width (High)	180		100		85		80		[2]
12	TsRxD(RxC)	RxD to /RxC Setup Time (X1 Mode)	0		0		0		0		[2]
13	ThRxD(RxC)	/RxC Rise to RxD Hold Time (X1 Mode)	140		100		80		60		[2]
14	TdRxC(W/RRI)	/RxC Rise to /W//RDY Fall Delay (Ready Mode)	10	13	10	13	10	13	10	13	[1]
15	TdRxC(INT)	/RxC Rise to /INT Fall Delay	10	13	10	13	10	13	10	13	[1]
16	TdRxC(SYNC)	/RxC Rise to /SYNC Fall Delay (Output Modes)	4	7	4	7	4	7	4	7	[1]
17	TsSYNC(RxC)	/SYNC Fall to /RxC Rise Setup (External Sync Modes)	-100		-100		-100		-100		[2]

* In All Modes, the System Clock rate must be at least five times the maximum data rate.
/RESET must be active a minimum of one complete clock cycle.

Notes:
[1] Units equal to System Clock Periods.
[2] Units in nanoseconds (nS).

* 4 MHz 84C4x is obsoleted and replaced by 6 MHz.

DC CHARACTERISTICS (Z844X / NMOS Z80 SIO)

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} -0.6	V_{CC} +0.3	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	
V_{OH_1}	Output High Voltage	+2.4		V	I_{OL} = 2.0 mA
V_{OH_2}	Output High Voltage			V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		±10	µA	V_{IN} = 0.4 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10	µA	V_{OUT} = 0.4 to V_{CC}
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current		+10/-40	µA	$0 < V_{IN} < V_{CC}$
ICC_1	Power Supply Current		100	mA	

Over specified temperature and voltage range.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C	Clock Capacitance		40	pf
C_{IN}	Input Capacitance		5	pf
C_{OUT}	Output Capacitance		15	pf

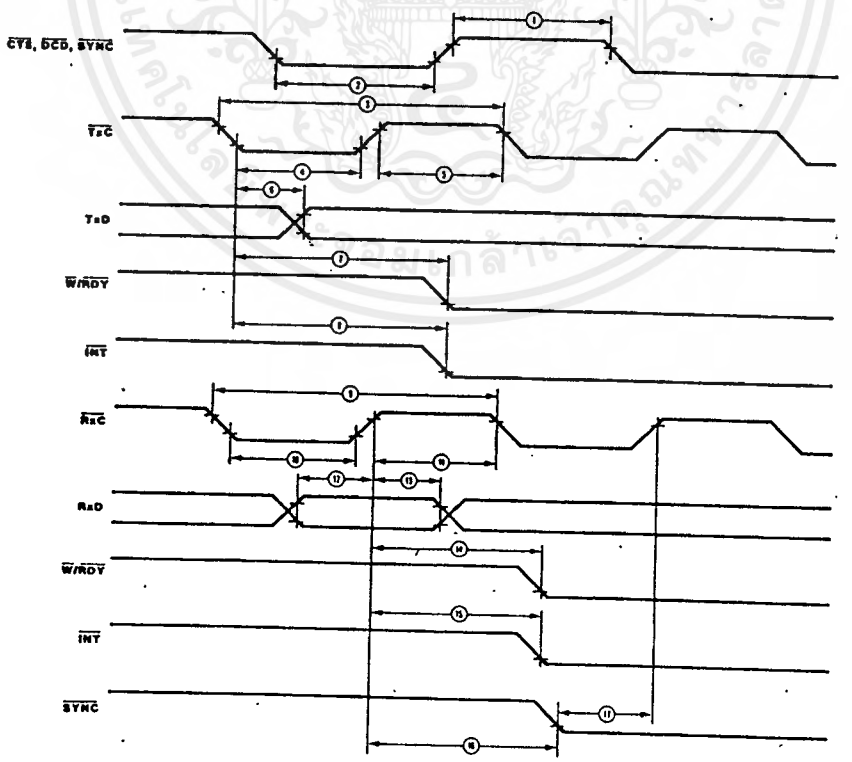
Over specified temperature range; f = 1 MHz.
Unmeasured pins returned to ground.

AC CHARACTERISTICS* (Z844X / NMOS Z80 SIO)

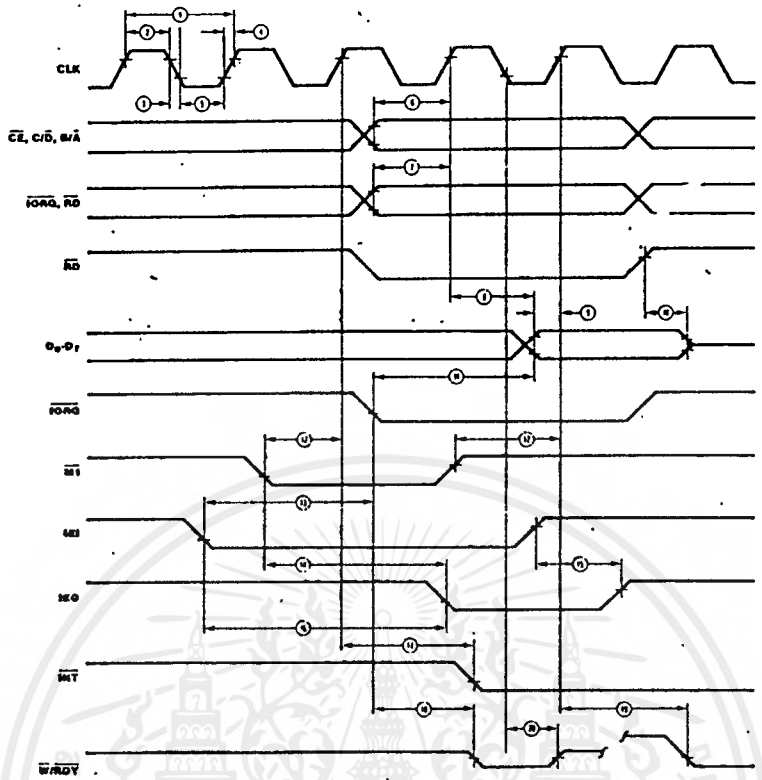
Number	Symbol	Parameter	Z0844X04 Min	Z0844X04 Max	Z0844X06 Min	Z0844X06 Max
1	TcC	Clock Cycle Time	250	4000	162	4000
2	TwCh	Clock Width (High)	105	2000	70	2000
3	TfC	Clock Fall Time		30		15
4	TrC	Clock Rise Time		30		15
5	TwCl	Clock Width (Low)	105	2000	70	2000
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	145		60	
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	115		60	
8	TdC(DO)	Clock ↑ to Data Out Delay		220		150
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		30	
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		110		90
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		160		120
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	90		75	
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	140		120	
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		190		160
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		100		70
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		100		70
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		150
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Wait Mode)		210		175
19	TdC(W/RRf)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		100
20	TdC(W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		130		110
21	Th	Any unspecified Hold when Setup is specified	0		0	

*Units in nanoseconds (ns).

AC CHARACTERISTICS TIMING (Z844X / NMOS Z80 SIO; Continued)



AC CHARACTERISTICS TIMING (Z844X / NMOS Z80 SIO)



AC CHARACTERISTICS (Z844X / NMOS Z80 SIO; Continued)

No.	Symbol	Parameter	Z0844X04 Min	Z0844X04 Max	Z0844X06 Min	Z0844X06 Max	Notes*
1	TwPh	Pulse Width (High)	200		200		2
2	TwPl	Pulse Width (Low)	200		200		2
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	330	∞	2
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	100	∞	2
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	100	∞	2
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay		300		220	2
7	TdTxC(W/RR)	$\overline{TxC}$ ↓ to W/$\overline{RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	1
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	1
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	330	∞	2
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	100	∞	2
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	100	∞	2
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		2
13	ThRxD(RxC)	$\overline{RxC}$ ↑ RxD Hold Time (x1 Mode)	140		100		2
14	TdRxC(W/RR)	$\overline{RxC}$ ↑ to W/$\overline{RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	1
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↑ Delay	10	13	10	13	1
16	TdRxC(SYNC)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ ↓ Delay (Output Modes)	4	7	4	7	1
17	TsSYNC(RxC)	$\overline{SYNC}$ ↓ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	-100		-100		2

*In all modes, the System Clock rate must be at least five times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete clock cycle.

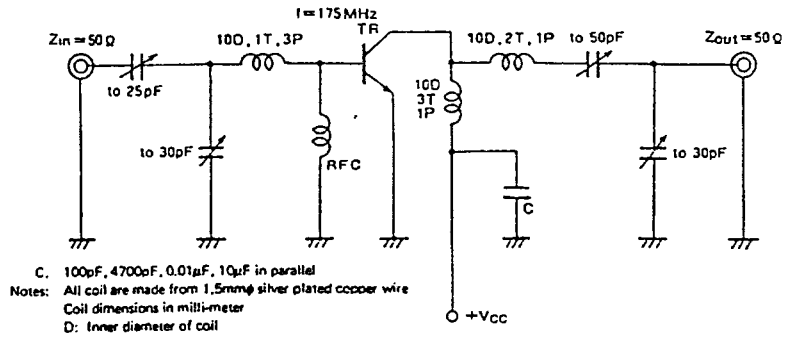
1. Units equal to System Clock Periods.

2. Units in nanoseconds (ns).

2SC1947

NPN EPITAXIAL PLANAR TYPE

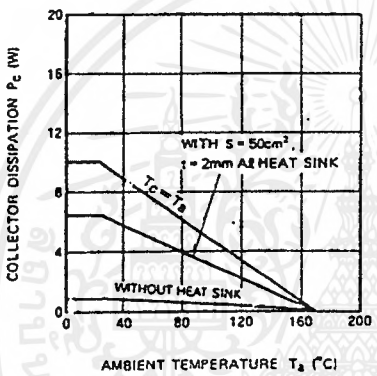
TEST CIRCUIT



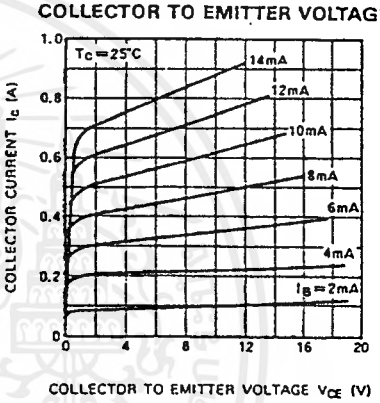
C. 100pF, 4700pF, 0.01μF, 10μF in parallel
Notes: All coil are made from 1.5mmφ silver plated copper wire
Coil dimensions in milli-meter
D: Inner diameter of coil
T Turn number of coil
P: Pitch of coil

TYPICAL PERFORMANCE DATA

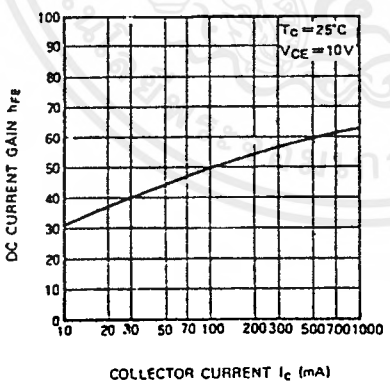
COLLECTOR DISSIPATION VS. AMBIENT TEMPERATURE



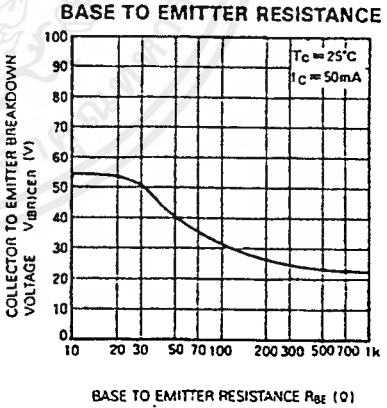
COLLECTOR CURRENT VS. COLLECTOR TO EMITTER VOLTAGE



DC CURRENT GAIN VS. COLLECTOR CURRENT

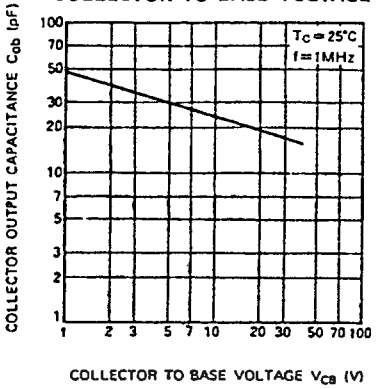


COLLECTOR TO EMITTER BREAKDOWN VOLTAGE VS. BASE TO EMITTER RESISTANCE

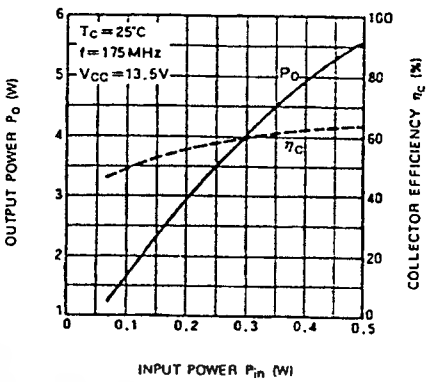


NPN EPITAXIAL PLANAR TYPE

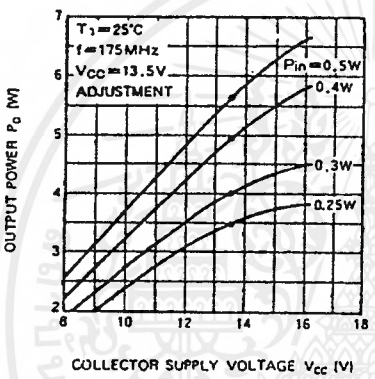
COLLECTOR OUTPUT CAPACITANCE VS. COLLECTOR TO BASE VOLTAGE



OUTPUT POWER, COLLECTOR EFFICIENCY VS. INPUT POWER



OUTPUT POWER VS. COLLECTOR SUPPLY VOLTAGE



MOTOROLA

MC12015
MC12016
MC12017

Dual Modulus Prescaler

The MC12015, MC12016 and MC12017 are dual modulus prescalers which will drive divide by 32 and 33, 40 and 41, and 64 and 65, respectively. An internal regulator is provided to allow these devices to be used over a wide range of power–supply voltages. The devices may be operated by applying a supply voltage of 5.0 Vdc ±10% at Pin 7, or by applying an unregulated voltage source from 5.5Vdc to 9.5 Vdc to Pin 8.

- 225 MHz Toggle Frequency
- Low–Power 7.5 mA Maximum at 6.8 V
- Control Input and Output Are Compatible With Standard CMOS
- Connecting Pins 2 and 3 Allows Driving One TTL Load
- Supply Voltage 4.5 V to 9.5 V

MECL PLL COMPONENTS
DUAL MODULUS PRESCALER

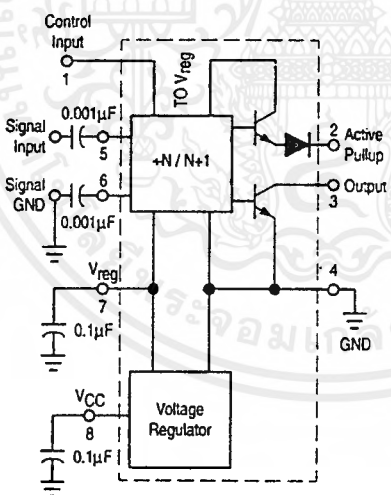
SEMICONDUCTOR
TECHNICAL DATA

D SUFFIX
PLASTIC PACKAGE
CASE 751
(SO–8)

ORDERING INFORMATION

Device	Operating Temperature Range	Package
MC12015D	T_A = –40 to 85°C	SO–8
MC12016D		
MC12017D		

SIMPLIFIED BLOCK DIAGRAM



1. V_{reg} at Pin 7 is not guaranteed to be between 4.5 and 5.5V when V_{CC} is being applied to Pin 8
2. Pin 7 is not to be used as a source of regulated output voltage

Rev 3

MAXIMUM RATINGS [tblhead]

Rating	Symbol	Value	Unit
Regulated Voltage, Pin 7	V_{reg}	8.0	Vdc
Power Supply Voltage, Pin 8	V_{CC}	10	Vdc
Operating Temperature Range	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−65 to +175	°C

NOTE: ESD data available upon request.

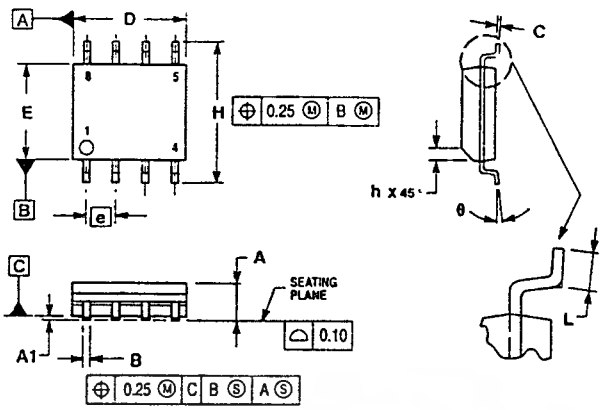
ELECTRICAL CHARACTERISTICS (V_{CC} = 5.5 to 9.5 V; V_{reg} = 4.5 to 5.5 V; T_A = −40 to 85°C, unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Toggle Frequency (Sine Wave Input)					MHz
	f_{max}	225	–	–	
	f_{min}	–	–	35	
Supply Current	I_{CC}	–	6.0	7.8	mA
Control Input HIGH (+32, 40 or 64)	V_{IH}	2.0	–	–	V
Control Input LOW (+33, 41 or 65)	V_{IL}	–	–	0.8	V
Output Voltage HIGH (I_{source} = 50μA) [Note 1]	V_{OH}	2.5	–	–	V
Output Voltage LOW (I_{sink} = 2mA) [Note 1]	V_{OL}	–	–	0.5	V
Input Voltage Sensitivity	V_{in}				mVpp
35 MHz		400	–	800	
50 to 225 MHz		200	–	800	
PLL Response Time [Notes 2 and 3]	t_{PLL}	–	–	t_{out} to 70	ns

NOTES: 1. Pin 2 connected to Pin 3.
2. t_{PLL} = the period of time the PLL has from the prescaler rising output transistion (50%) to the modulus control input edge transition (50%) to ensure proper modulus selection.
3. t_{out} = period of output waveform.

OUTLINE DIMENSIONS

D SUFFIX
PLASTIC PACKAGE
CASE 751-06
(SO–8)
ISSUE T



NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS ARE IN MILLIMETER.
3. DIMENSION D AND E DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.
5. DIMENSION B DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.127 TOTAL IN EXCESS OF THE B DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS	
DIM	MIN	MAX
A	1.35	1.75
A1	0.10	0.25
B	0.35	0.49
C	0.19	0.25
D	4.80	5.00
E	3.80	4.00
e	1.27 BSC	
H	5.80	6.20
h	0.25	0.50
L	0.40	1.25
θ	0°	7°

MC145152-2

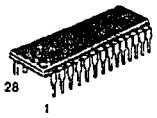
Parallel-Input PLL Frequency Synthesizer

Interfaces with Dual–Modulus Prescalers

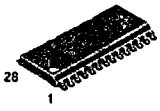
The MC145152–2 is programmed by sixteen parallel inputs for the N and A counters and three input lines for the R counter. The device features consist of a reference oscillator, selectable–reference divider, two–output phase detector, 10–bit programmable divide–by–N counter, and 6–bit programmable ÷ A counter.

The MC145152–2 is an improved–performance drop–in replacement for the MC145152–1. Power consumption has decreased and ESD and latch–up performance have improved.

- Operating Temperature Range: – 40 to 85°C
- Low Power Consumption Through Use of CMOS Technology
- 3.0 to 9.0 V Supply Range
- On– or Off–Chip Reference Oscillator Operation
- Lock Detect Signal
- Dual Modulus/Parallel Programming
- 8 User–Selectable ÷ R Values: 8, 64, 128, 256, 512, 1024, 1160, 2048
- ÷ N Range = 3 to 1023, ÷ A Range = 0 to 63
- Chip Complexity: 8000 FETs or 2000 Equivalent Gates
- See Application Note AN980



P SUFFIX
PLASTIC DIP
CASE 710

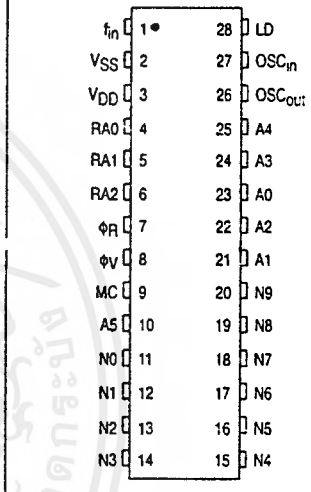


D'V SUFFIX
SOG PACKAGE
CASE 751F

ORDERING INFORMATION

MC145152P2	Plastic DIP
MC145152DW2	SOG Package

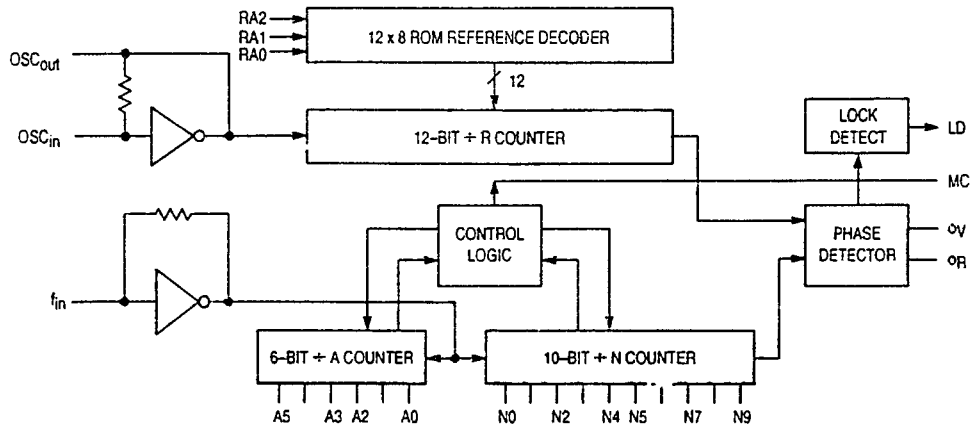
PIN ASSIGNMENT



REV 1
8/95

MOTOROLA

MC145152–2 BLOCK DIAGRAM



NOTE: N0 – N9, A0 – A5, and RA0 – RA2 have pull–up resistors that are not shown.

PIN DESCRIPTIONS

INPUT PINS

f_{in}
Frequency Input (Pin 1)

Input to the positive edge triggered ÷ N and ÷ A counters. f_{in} is typically derived from a dual–modulus prescaler and is ac coupled into the device. For larger amplitude signals (standard CMOS logic levels) dc coupling may be used.

RA0, RA1, RA2
Reference Address Inputs (Pins 4, 5, 6)

These three inputs establish a code defining one of eight possible divide values for the total reference divider. The total reference divide values are as follows:

Reference Address Code			Total Divide Value
RA2	RA1	RA0	
0	0	0	8
0	0	1	64
0	1	0	128
0	1	1	256
1	0	0	512
1	0	1	1024
1	1	0	1160
1	1	1	2048

N0 – N9
N Counter Programming Inputs (Pins 11 – 20)

The N inputs provide the data that is preset into the ÷ N counter when it reaches the count of 0. N0 is the least significant digit and N9 is the most significant. Pull–up resistors ensure that inputs left open remain at a logic 1 and require only a SPST switch to alter data to the zero state.

A0 – A5
A Counter Programming Inputs (Pins 23, 21, 22, 24, 25, 10)

The A inputs define the number of clock cycles of f_{in} that require a logic 0 on the MC output (see **Dual–Modulus Prescaling** section). The A inputs all have internal pull–up resistors that ensure that inputs left open will remain at a logic 1.

OSC_{in}, OSC_{out}
Reference Oscillator Input/Output (Pins 27, 26)

These pins form an on–chip reference oscillator when connected to terminals of an external parallel resonant crystal. Frequency setting capacitors of appropriate value must be connected from OSC_{in} to ground and OSC_{out} to ground. OSC_{in} may also serve as the input for an externally–generated reference signal. This signal is typically ac coupled to OSC_{in}, but for larger amplitude signals (standard CMOS logic levels) dc coupling may also be used. In the external reference mode, no connection is required to OSC_{out}.

OUTPUT PINS

ϕ_R, ϕ_V
Phase Detector B Outputs (Pins 7, 8)

These phase detector outputs can be combined externally for a loop–error signal.

If the frequency f_V is greater than f_R or if the phase of f_V is leading, then error information is provided by ϕ_V pulsing low. ϕ_R remains essentially high.

If the frequency f_V is less than f_R or if the phase of f_V is lagging, then error information is provided by ϕ_R pulsing low. ϕ_V remains essentially high.

If the frequency of $f_V = f_R$ and both are in phase, then both ϕ_V and ϕ_R remain high except for a small minimum time period when both pulse low in phase.

MC
Dual–Modulus Prescale Control Output (Pin 9)

Signal generated by the on–chip control logic circuitry for controlling an external dual–modulus prescaler. The MC level will be low at the beginning of a count cycle and will remain low until the ÷ A counter has counted down from its programmed value. At this time, MC goes high and remains high until the ÷ N counter has counted the rest of the way down from its programmed value (N – A additional counts since both ÷ N and ÷ A are counting down during the first

portion of the cycle). MC is then set back low, the counters preset to their respective programmed values, and the above sequence repeated. This provides for a total programmable divide value (N_T) = N • P + A where P and P + 1 represent the dual–modulus prescaler divide values respectively for high and low MC levels, N the number programmed into the ÷ N counter, and A the number programmed into the ÷ A counter.

LD
Lock Detector Output (Pin 28)

Essentially a high level when loop is locked (f_R, f_V of same phase and frequency). Pulses low when loop is out of lock.

POWER SUPPLY

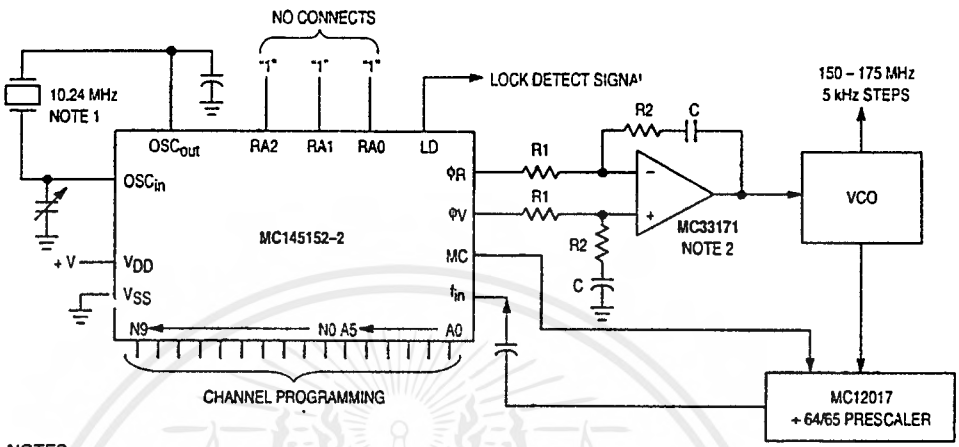
V_{DD}
Positive Power Supply (Pin 3)

The positive power supply potential. This pin may range from + 3 to + 9 V with respect to V_{SS}.

V_{SS}
Negative Power Supply (Pin 2)

The most negative supply potential. This pin is usually ground.

TYPICAL APPLICATIONS



NOTES:
1. Off–chip oscillator optional.
2. The ϕ_R and ϕ_V outputs are fed to an external combiner/loop filter. See the Phase–Locked Loop — Low–Pass Filter Design page for additional information. The ϕ_R and ϕ_V outputs swing rail–to–rail. Therefore, the user should be careful not to exceed the common mode input range of the op amp used in the combiner/loop filter.

Figure 1. Synthesizer for Land Mobile Radio VHF Bands

กิตติกรรมประกาศ

โครงงานเรื่องการสื่อสารข้อมูลด้วยระบบแพกเกตเรดิโอนี้สำเร็จลุล่วงได้ ด้วยความช่วยเหลือจากบุคคลหลาย ๆ ท่าน ในด้านความรู้ ประสบการณ์ เครื่องมืออุปกรณ์ ตลอดจนเป็นที่ปรึกษา และเป็นกำลังใจในการทำงาน ขอขอบคุณบุคคลต่อไปนี้

- อาจารย์สุทธิชัย นพนาคีพงษ์ อาจารย์ที่ปรึกษาโครงงาน ผู้ให้คำปรึกษาและแนะนำ
- กองอุปกรณ์อิเล็กทรอนิกส์ สำนักงานพลังงานปรมาณูเพื่อสันติ กระทรวงวิทยาศาสตร์เทคโนโลยีและสิ่งแวดล้อม ที่ให้ความสะดวกในการใช้อุปกรณ์การทดลอง และการใช้ห้องปฏิบัติการ

และขอขอบคุณ คุณพ่อ คุณแม่ ที่ให้กำลังใจอยู่เสมอ รวมถึงเพื่อนทุกๆ คนที่ให้ความร่วมมือช่วยเหลือทำให้โครงงานชิ้นนี้สำเร็จได้ด้วยดี

คณะผู้จัดทำ

20 มิถุนายน 2542

หนังสืออ้างอิง

ประสิทธิ์ ประพิณมงคลการ. หลักการระบบสื่อสาร. กรุงเทพฯ; ซีเอ็ดยูเคชั่น , 2537.

ถวิล พึ่งมา. การออกแบบวงจรทางโทรคมนาคม. กรุงเทพฯ; คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2534

Peter Vizmulle. RF Design Guide System, Circuits, and Equations. London; Artech House, Inc., 1995

Lawrence E. Larson. RF and Microwave Circuit Design For Wireless Communications. London; Artech House, Inc., 1996

Ulrich L. Rohde , T.T.N. Bucher. Communications Receivers. USA; McGraw-Hill, inc., 1988