การส่งข้อมูลดิจิตอลด้วยวิธี เอฟ. เอส. เค.ที่มีอัตราการส่งสูง

(HIGH TRANSFER RATE FOR DIGITAL DATA COMMUNICATION USING FREQUENCY SHIFT KEYING)

นายพรศักดิ์ ทับเที่ยง

Mr.PORNSAK TABTIENG

อาจารย์ที่ปรึกษา

รศ.ดร.จเร สุรวัฒน์ปัญญา

Assoc.Prof.Dr.CHARRAY SURAWATPUNYA

อาจารย์ที่ปรึกษาร่วม

รศ.ดร.สิทธิชัย โภไคยอุดม

Assoc.Prof.Dr.SITTHICHAI POOKAIYAUDOM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต

ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อวิทยานิพนธ์	การส่งข้อมูลดิจิตอลด้วยวิธี เอฟ.เอส.เค. ที่มีอัตราการส่งสูง
นักศึกษา	นายพรศักดิ์ ทับเที่ยง
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.จเร สุรวัฒน์ปัญญา
อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม	รศ.ดร.สิทธิชัย โภไคยอุดม
ระดับการศึกษา	วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ภาควิชา	โทรคมนาคม, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2536

บทคัดย่อ

การสื่อสารโดยส่งข้อมูลดิจิตอลด้วยวิธี เอฟ.เอส.เค.(FREQUENCY SHIFT KEYING, FSK) ผ่านสายโทรศัพท์ ในปัจจุบันนี้ถูกจำกัดด้วยแบนด์วิดท์ (BANDWIDTH) ของคู่สายโทรศัพท์ ทำให้อัตราการส่งข้อมูลมีความเร็วประมาณ 300 บิทต่อวินาที แต่วงจรของภาคส่งและภาครับแบบ FSK นี้สร้างได้ง่ายไม่สลับซับซ้อน และมีเสถียรภาพต่อสัญญาณรบกวนได้ค่อนข้างดี วิทยานิพนธ์นี้จัดทำเพื่อมีวัตถุประสงค์ที่จะหาวิธีการส่งข้อมูลดิจิตอลด้วยวิธี FSK แต่ให้มีอัตราการส่งข้อมูลได้ไม่ต่ำกว่า 1 กิโลบิทต่อวินาที โดยใช้แบนด์วิดท์ของตัวกลางระหว่างภาคส่งและภาครับอยู่ที่ 3.4 กิโลเฮิรตซ์ และยังคงข้อดีของการส่งแบบ FSK ที่อัตราการส่ง 300 บิทต่อวินาที

Thesis Title	HIGH TRANSFER RATE FOR DIGITAL DATA COMMUNICATION USING FREQUENCY SHIFT KEYING
Student	Mr.Pornsak Tabtieng
Thesis Advisor	Assoc.Prof.Dr.Charray Surawatpunya
Thesis Co-advisor	Assoc.Prof.Dr.Sitthichai Pookaiyaudom
Level of Study	Master of Engineering in Eletrical Engineering
Department	Telecommunication Engineering,King Mongkut's Institute of technology Ladkrabang
Year	1993

ABSTRACT

Digital data communication using Frequency Shift Keying (FSK) through the telephone line has at present, been limited by the bandwidth of the line and the capability of the transmitter and the receiver. This results in a data transfer rate approximately 300 bit/sec. However the designing technique for FSK is simple and it is able to tolerate noise interferrence. The objective of this thesis is to design a system for transferring digital data communication at the rate at least 1 kilobit/second. The bandwidth of the medium between the transmitter and the receiver is 3.4 kilohertz. The system is designed to maintain the advantage of FSK as proposed above.

กิตติกรรมประกาศ

ผู้เขียนขอขอบคุณ ดร.จเร สุรวัฒน์ปัญญา อาจารย์ที่ปรึกษา ดร.สิทธิชัย โภไคยอุดม อธิการบดี มหาวิทยาลัยเทคโนโลยีมหานคร อาจารย์ที่ปรึกษาร่วม ที่ได้กรุณาให้คำแนะนำและข้อเสนอแนะ ทั้งให้ยืมอุปกรณ์ต่าง ๆ และใช้อุปกรณ์ในห้อง Lab ที่มหาวิทยาลัยเทคโนโลยีมหานครในการทดลองจนโครงการนี้สำเร็จลุล่วง พร้อมนี้ขอขอบคุณ ดร.สุเจตน์ จันทรังษี นายวรพล ลีลาเกียรติสกุล และนายวรพงศ์ ศิลาพันธ์ ซึ่งทั้ง 3 ท่าน เป็นอาจารย์มหาวิทยาลัยเทคโนโลยีมหานคร และให้การสนับสนุนในการทำการทดลองโครงการนี้

นายพรศักดิ์ ทับเที่ยง

สารบัญ

บทคัดย่อ

กิตติกรรมประกาศ

บทที่ 1 บทนำ 1

บทที่ 2 ทฤษฎีและแนวความคิดของระบบเอฟ.เอส.เค.

2.1 กล่าวนำ 4

2.2 ต้วกำเนิดสัญญาณ เอฟ.เอส.เค. 6

2.3 เอฟ.เอส.เค. ดีมอดูเลเตอร์ 8

2.4 แนวความคิดการแก้ปัญหาระบบ เอฟ.เอส.เค.มอดูเลต ระบบเดิม 12

2.5 แนวการออกแบบภาคมอดูเลเตอร์ 15

2.6 แนวการออกแบบภาคดีมอดูเลเตอร์ 17

บทที่ 3 การออกแบบภาคมอดูเลตสัญญาณ เอฟ.เอส.เค

3.1 กล่าวนำ 20

3.2 การทำงานของภาคมอดูเลตสัญญาณ เอฟ.เอส.เค. 20

3.2.1 การทำงานของวงจรส่วนที่ 1 22

3.2.2 การทำงานของวงจรส่วนที่ 2 24

บทที่ 4 การออกแบบภาคดีมอดูเลตสัญญาณ เอฟ.เอส.เค

4.1 กล่าวนำ 36

4.2 ภาค เอฟ.เอส.เค.ดีมอดูเลต 36

4.2.1 การทำงานของวงจรสร้างสัญญาณนาฬิกา ทุกครึ่งคาบเวลา 37

4.2.2 การทำงานของวงจรสร้างสัญญาณเปรียบเทียบ 39

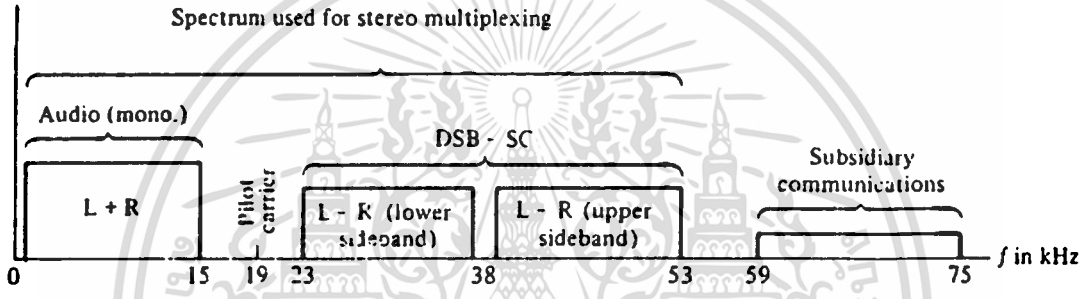
4.2.3 การทำงานของวงจรเปรียบเทียบคาบเวลา 41
4.2.4 การทำงานของวงจรแปลงข้อมูลอนุกรม
เป็นข้อมูลขนาน 42
4.2 วงจรทดลองภาคดีมอดูเลเตอร์ 43
บทที่ 5 ผลการทดลอง 48
บทที่ 6 สรุป 61
ภาคผนวก ก. 63
ภาคผนวก ข. 74
เอกสารอ้างอิง 85

บทที่ 1
บทนำ

การสื่อสารในปัจจุบันได้ก้าวหน้าไปอย่างไม่หยุดยั้ง ดังนั้น การสื่อสารในระยะไกลจึงได้เข้ามามีบทบาทสำคัญต่อชีวิตประจำวันมากขึ้น ไม่ว่าจะเป็นการสื่อสารผ่านตัวกลางโดยใช้สาย หรือกระจายคลื่นไปในอากาศ ดังเช่น ระบบโทรศัพท์ วิทยุกระจายเสียง วิทยุโทรทัศน์ และตลอดจนการสื่อสารผ่านดาวเทียม แต่จะเห็นว่าการสื่อสารที่สามารถตอบสนองความต้องการของทุกชนชั้น ได้มากกว่าระบบอื่น ๆ คือ วิทยุกระจายเสียง ทั้งนี้เนื่องจากว่าเครื่องรับวิทยุทั่วไปราคาถูก และหาได้ง่ายนั่นเองและวิทยุกระจายเสียงในปัจจุบันสามารถออกอากาศได้ครอบคลุมทั่วประเทศ แต่วิทยุกระจายเสียงก็มีความสามารถส่งได้แต่ระบบเสียงเท่านั้น ดังนั้น จึงมีการค้นคิดหาวิธีต่าง ๆ เพื่อที่จะให้ระบบวิทยุกระจายเสียงสามารถส่งข้อมูลอย่างอื่นผสมไปด้วยกับการออกอากาศรายการปกติ โดยไม่มีการรบกวนซึ่งกันและกัน ซึ่งเป็นการนำเอาความถี่ที่ได้รับมาใช้ให้เกิดประโยชน์มากที่สุดตามกฎเกณฑ์มาตรฐานวิทยุกระจายเสียงของ CCIR.

ดังนั้นเมื่อพิจารณาแถบคลื่นความถี่ทางวิทยุกระจายเสียง ระบบ เอฟ.เอ็ม. สเตอริโอ(FM Stereo) ซึ่งมี สเปคตรัม(Sepctrum) ของสัญญาณดังรูปที่ 1.1[1] จะเห็นว่าการส่งกระจายเสียงรายการปกตินั้นใช้ความถี่ในช่วง 0 ถึง 53 kHz ส่วนความถี่ที่อยู่ในช่วง 59 - 75 kHz เป็นส่วนที่เรียกว่า ซับซิเดรี่(Subsidiary) เป็นส่วนที่สามารถนำมาใช้งานในลักษณะพิเศษ ซึ่งจะเรียกว่า FM SCA (FM Subsidiary Communication Authorization) โดยมี Subcarrier ที่ 67 kHz และได้มีบางประเทศได้นำเอาความถี่นอกเหนือจากนี้โดยใช้ Subcarrier ที่ 57 kHz มาใช้ประโยชน์ในการส่งข่าวสารด้านการจราจรและแจ้งเหตุ จึงเรียกระบบการส่งนี้ว่าระบบ RDS (Radio Data System)

จากการที่ในปัจจุบันนี้ได้นำเอาระบบการสื่อสารข้อมูลดิจิตอล หรือการส่งดาต้า (Data Communication) ซึ่งเป็นการนำเอาสัญญาณแอนาลอกมาเปลี่ยนให้อยู่ในรูปของข้อมูลแบบไบนารี (Binary Form) แล้วนำมาใช้ติดต่อสื่อสารนั่นเอง การส่งข้อมูลดิจิตอลในระบบต่าง ๆ จึงถูกทำการค้นคว้าทดลองและเป็นที่รู้จักอย่างแพร่หลายในปัจจุบัน



รูปที่ 1.1 แสดงสเปคตรัมของคลื่นวิทยุ เอฟ.เอ็ม.สเตอริโอ

โครงการนี้เป็นการนำเสนอวิธีการสื่อสารข้อมูลดิจิตอลโดยใช้ ระบบวิทยุกระจายเสียงหรือสายโทรศัพท์เป็นตัวกลาง ด้วยวิธีการเปลี่ยนความถี่ตามระดับสัญญาณดิจิตอล ซึ่งเป็นดิจิตอลมอดูเลชันวิธี เอฟ.เอส.เค. (Frequency Shift Keying, FSK) แบบครึ่งคาบเวลาซึ่งใช้สัญญาณความถี่ครึ่งคาบเวลาต่อข้อมูลไบนารี 1 บิต เพื่อนำไปประยุกต์ใช้งานในด้านการศึกษาทางไกล หรือการสื่อสารข้อมูลผ่านทางสื่อวิทยุกระจายเสียงด้วยสัญญาณวิทยุแบบมีภาพ (Radio and Video System, RVS) โดยการนำเอาสัญญาณภาพหรือข้อมูลที่สร้างจากเครื่องคอมพิวเตอร์ ซึ่งเป็นสัญญาณดิจิตอลมาทำการมอดูเลตชัน(Modulation) ด้วยวิธีการแบบ เอฟ.เอส.เค.

ซึ่งจะได้เป็นสัญญาณแอนาลอก แล้วนำไปทำการมอดูเลตกับความถี่หลัก (Main Carrier) ของสถานีวิทยุกระจายเสียงในระบบมอดูเลชันทางความถี่(Frequency Modulation, FM) ในช่วงความถี่ซับชิเดรีแล้วนำส่งกระจายเสียงออกอากาศไปตามปกติ

การสื่อสารข้อมูลดิจิตอลผ่านตัวกลาง ต้องทำการเปลี่ยนข้อมูลดิจิตอลที่ต้องการส่งให้เป็นสัญญาณแอนาลอกก่อนจะทำการส่ง แล้วทำการแปลงสัญญาณแอนาลอกกลับมาเป็นข้อมูลดิจิตอลที่ภาครับปลายทาง ซึ่งใช้เทคนิคการมอดูเลตสัญญาณ แบบดิจิตอลมอดูเลชั่น(Digital Modulation) ซึ่งจะกล่าวรายละเอียดในบทต่อไป

บทที่ 2
ทฤษฎีและแนวคิดของระบบ
เอฟ. เอส. เค

2.1 กล่าวนำ

การสื่อสารในปัจจุบันได้นิยมนำเอาการมอดูเลตสัญญาณแบบดิจิตอลมอดูเลชันมาใช้กันอย่างแพร่หลาย เพราะว่าระบบดิจิตอลให้ค่าความแน่นอนน่าเชื่อถือสูงกว่าระบบแอนาลอก และมีการรบกวนอันเนื่องมาจากสัญญาณรบกวน (Noise) ต่ำ ซึ่งในปัจจุบันนี้ อุปกรณ์ต่าง ๆ ในด้านดิจิตอลได้มีการพัฒนาก้าวหน้าไปอย่างรวดเร็ว ทำให้ต้นทุนในการผลิตอุปกรณ์ต่ำลง และนอกจากนี้แล้วระบบดิจิตอลมอดูเลชันยังสามารถทำการเข้ารหัส(Encode) ก่อนทำการมอดูเลต แล้วทำการถอดรหัส(Decode) หลังการดีมอดูเลต ทำให้การส่งข้อมูลมีการผิดพลาดน้อยลง ในการมอดูเลชันสัญญาณแบบดิจิตอลมีอยู่ด้วยกัน 3 วิธีใหญ่ ๆ [1] คือ (1) วิธีการเปลี่ยนขนาดของสัญญาณตามสัญญาณดิจิตอล(Amplitude Shift Keying, ASK) (2) วิธีเปลี่ยนความถี่ตามสัญญาณดิจิตอล(Frequency Shift Keying, FSK) และ (3) วิธีการเปลี่ยนเฟสตามสัญญาณดิจิตอล(Phase Shift Keying, PSK) รูปแบบของสัญญาณแบบดิจิตอลมอดูเลชันทั้ง 3 วิธีแสดงไว้ดังรูปที่ 2.1

จากรูปที่ 2.1 เป็นรูปสัญญาณดิจิตอลมอดูเลชันแบบต่าง ๆ ซึ่งอธิบายรูปแบบของสัญญาณได้ดังนี้

วิธีที่ 1 การเปลี่ยนขนาดของสัญญาณตามสัญญาณดิจิตอล(เอ.เอส.เค.)รูปคลื่นสัญญาณที่ได้จากการมอดูเลตแบบสัญญาณดิจิตอลเปลี่ยนแปลงขนาดสัญญาณที่ได้ ตามระดับของสัญญาณดิจิตอลที่เปลี่ยนแปลงไป จากรูปที่ 2.1 ที่ระดับดิจิตอลมีสถานะลอจิก 0 สัญญาณที่ได้จะมีขนาดเป็น 0 และจะมีขนาดเปลี่ยนแปลงไปตามคลื่นพาห์ (Carrier) เมื่อระดับลอจิกมีสถานะเป็น 1 วิธีการนี้มีข้อดีคือ ทั้งภาคมอดูเลเตอร์

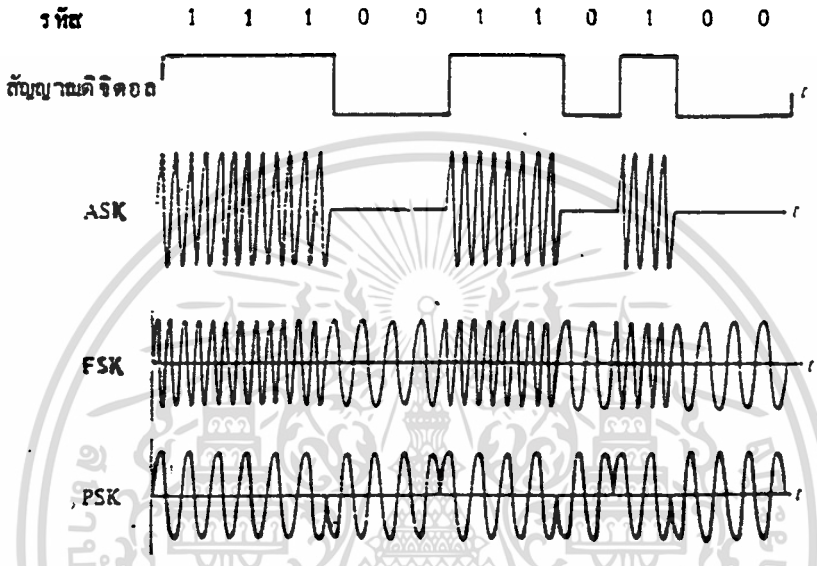
และดีมอดูเลเตอร์มีส่วนประกอบวงจรที่ง่าย ราคาถูก แต่มีข้อเสียคือข้อมูลที่รับเข้ามาภาคปลายทางผิดพลาดได้ง่าย อันเนื่องมาจากสัญญาณรบกวนที่จะมีผลต่อขนาดของสัญญาณ และทางภาครับมีวงจรขยายชดเชยการลดทอนสัญญาณในสายอัตโนมัติ อีกทั้งยังมีอัตราการส่งข้อมูลได้ไม่สูงมากนัก

วิธีที่ 2 การเปลี่ยนความถี่ตามสัญญาณดิจิตอล(เอฟ.เอส.เค.) รูปคลื่นสัญญาณที่ได้จากการมอดูเลตแบบนี้ สัญญาณดิจิตอลควบคุมความถี่ของสัญญาณที่จะส่งออกจากวงจรมอดูเลชัน โดยให้รูปคลื่นที่ได้มีความถี่สูง เมื่อระดับสัญญาณดิจิตอลเป็น 1 และมีความถี่ต่ำเมื่อเป็น 0 ซึ่งมีอัตราการส่งข้อมูลต่ำพอ ๆ กับวิธีการ เอ.เอส.เค. สำหรับกรณีที่ใช้ส่งทางสายที่มีแบนด์วิดท์ (Bandwidth) ไม่เกิน 3.4 กิโลเฮิรตซ์ อัตราการส่งข้อมูลสูงสุดจะไม่เกิน 1200 บิตต่อวินาที และวิธีการนี้มีข้อดีเหมือนกับวิธีการมอดูเลตแบบ เอ.เอส.เค. คือ มีส่วนประกอบของวงจรที่ง่ายไม่สลับซับซ้อน ราคาถูก และมีเสถียรภาพต่อสัญญาณรบกวนได้สูงกว่า

วิธีที่ 3 การเปลี่ยนเฟสตามสัญญาณดิจิตอล(พี.เอส.เค.) รูปคลื่นสัญญาณที่ได้จากการมอดูเลตแบบสัญญาณดิจิตอลควบคุมการเปลี่ยนเฟสของสัญญาณ จะเห็นได้ว่า เมื่อใช้ความถี่ในการส่งสัญญาณตามระดับของสัญญาณดิจิตอลด้วยความถี่เดียวกันตลอด แต่เมื่อมีการเปลี่ยนแปลงระดับของสัญญาณดิจิตอล ก็จะมีการเปลี่ยนเฟสของคลื่นพาห์เป็นตรงข้าม(180^{0}) วงจรของภาครับและส่งข้อมูลมีความยุ่งยากมาก ราคาสูง แต่สามารถส่งข้อมูลได้สูงกว่า 1200 บิต/วินาที

ดังนั้น เมื่อพิจารณาข้อดีข้อเสียของรูปแบบการมอดูเลตสัญญาณแบบดิจิตอลมอดูเลชันด้วยวิธีการต่าง ๆ ที่กล่าวมาแล้วจะเห็นว่าเอฟ.เอส.เค. จะมีเสถียรภาพต่อสัญญาณรบกวนได้ดีกว่าระบบ เอ.เอส.เค.และระบบ พี.เอส.เค.[2] และวงจรภาคมอดูเลเตอร์มีขั้นตอนการทำงานของวงจรไม่ซับซ้อนและราคาถูก ดังนั้นโครงการนี้จะนำเสนอวิธีการมอดูเลตและดีมอดูเลตด้วยวิธีการ เอฟ.เอส.เค. แบบใหม่ซึ่งทำ

การรับ-ส่ง ข้อมูลไบนารี 1 บิต ด้วยสัญญาณความถี่ครึ่งคาบเวลา ซึ่งจะทำให้อัตราบิต(Bit rate) ในการส่งสูงกว่าแบบเดิมที่ใช้อยู่ในปัจจุบัน และมีส่วนประกอบการทำงานของวงจรแบบง่ายๆ ไม่ซับซ้อน ราคาถูก

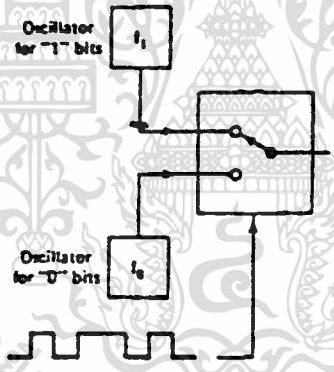


รูปที่ 2.1 รูปคลื่นของการมอดูเลตสัญญาณแบบดิจิตอลมอดูเลชัน

2.2 ตัวกำเนิดสัญญาณ เอฟ.เอส.เค.(FSK Modulator)

เนื่องจากการส่งข้อมูลด้วยวิธีการมอดูเลตแบบ เอฟ.เอส.เค. นี้ เป็นการเปลี่ยนแปลงความถี่ในการรับ-ส่งข้อมูลตามระดับของสัญญาณดิจิตอล เมื่อข้อมูลดิจิตอลแบบไบนารีได้มีการเปลี่ยนแปลงระดับที่ใช้ตามสถานะลอจิก 1 และ 0 สัญญาณ เอฟ.เอส.เค. ที่ได้จะมีการเปลี่ยนแปลงระหว่าง 2 ความถี่ด้วยกัน[3] คือให้ความถี่ที่สถานะลอจิก 1 เป็นความถี่มาร์ค(Mark Frequency) และความถี่ที่สถานะลอจิก 0 เป็นความถี่สเปซ(Space Frequency) ดังนั้น ที่เอาท์พุตจะมีการเปลี่ยนแปลงความถี่เมื่อข้อมูลเข้ามาที่อินพุทเปลี่ยนสถานะไป ในระบบดิจิตอลมอดูเลชันนั้น อัตรา

การเปลี่ยนแปลงของสัญญาณอินพุทเรียกว่าอัตราบิต โดยมีหน่วยเป็นบิตต่อวินาที(bps) ส่วนอัตราการเปลี่ยนแปลงของสัญญาณด้านเอาท์พุทเรียกว่า อัตราบอด(Baud Rate) ดังนั้น การส่งสัญญาณข้อมูลแบบระบบ เอฟ.เอส.เค. จะเห็นว่าค่าอัตราบิตและอัตราบอดจะเท่ากันเสมอ แผนภาพการทำงานของภาคมอดูเลชันของการส่งข้อมูลดิจิตอลด้วยวิธีการ เอฟ.เอส.เค. มอดูเลตชันในปัจจุบันมีการทำงานอย่างง่าย ๆ[4] ดังรูปที่ 2.2 โดยมีแหล่งกำเนิดสัญญาณความถี่หรือออสซิลเลเตอร์ (Oscillator) ความถี่ f_1 และ f_2 โดยจะใช้ข้อมูลดิจิตอลแบบไบนารีมาควบคุมการทำงานของสวิทช์ให้เลือกตัวกำเนิดสัญญาณ เอฟ.เอส.เค. ซึ่งทำหน้าที่คล้ายสวิทช์ 2 ทาง โดยเลื่อนไปเลือกอินพุทตามสัญญาณไบนารีที่ควบคุม



รูปที่ 2.2 หลักการทำงานอย่างง่าย ๆ ของ เอฟ.เอส.เค.มอดูเลเตอร์

ถ้าข้อมูลเข้ามามีลอจิกเป็น 0 จะต้องทำการเลื่อนสวิทช์ไปทางวงจร f_1 เพื่อให้สัญญาณ f_1 ออกที่เอาท์พุท และเมื่อลอจิกเป็น 1 เลื่อนสวิทช์ไปทางวงจร f_2 เพื่อให้ความถี่ f_2 ออกที่เอาท์พุท

สัญญาณ เอฟ.เอส.เค. ที่ได้จะมีสมการดังนี้[3]

$v(t) = A \cos(\omega_c \pm \Delta\omega)t$

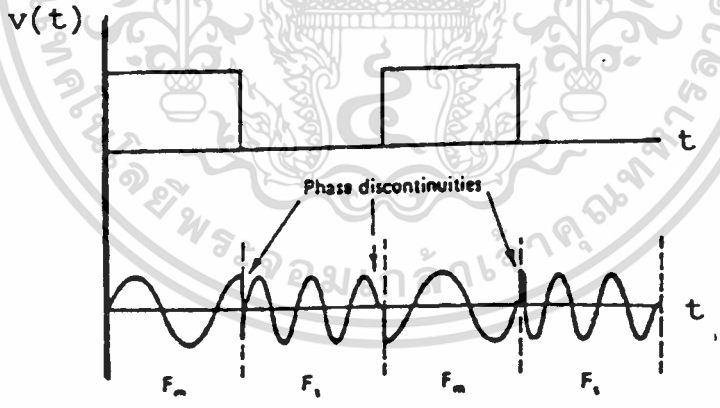
A : ขนาดของสัญญาณ FSK

$\omega_c = 2\pi f_c$

$\Delta\omega$: ความถี่เบี่ยงเบน

t : เวลา

จากวงจรรูปที่ 2.2 จะเห็นว่าการทำงานของวงจรใช้แหล่งกำเนิดสัญญาณความถี่แยกกัน 2 ชุด จึงเป็นสาเหตุทำให้สัญญาณ เอฟ.เอส.เค. ที่ได้ มีเฟสไม่ต่อเนื่อง(Discontinuous Phase)[2] ตรงที่มีการเปลี่ยนแปลงของระดับลอจิกในสัญญาณดิจิตอล ดังรูปที่ 2.3 และจะเห็นว่าสัญญาณที่ได้จากวิธีการแบบ เอฟ.เอส.เค. ที่ได้จะมีรูปคลื่นมากกว่า 1 คาบเวลาต่อสัญญาณดิจิตอล 1 บิต ซึ่งเป็นสาเหตุที่ทำให้อัตราบิตในการส่งข้อมูลได้ไม่สูง

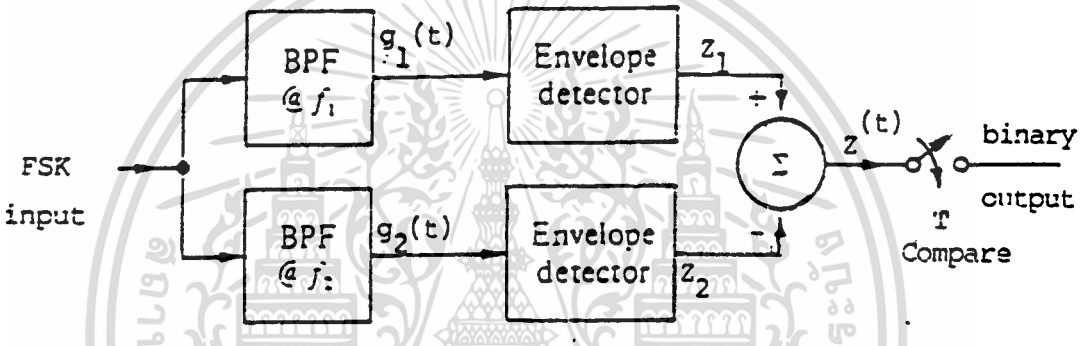


รูปที่ 2.3 สัญญาณ เอฟ.เอส.เค.ที่มีเฟสไม่ต่อเนื่อง

2.3 เอฟ.เอส.เค.ดีมอดูเลเตอร์ (FSK Demodulator)

แผนภาพของวงจร เอฟ.เอส.เค. ดีมอดูเลเตอร์[1]แสดงได้ดังรูปที่ 2.4

ซึ่งวงจรแบ่งออกเป็น 2 ส่วน คือ ส่วนที่ใช้สำหรับแยกความถี่ f_1 และ f_2 โดยใช้วงจรกรองแถบความถี่ผ่าน(Band Pass Filter,BPF) ทำงานร่วมกับวงจรเอนเวลโลปดีเทคเตอร์ (Envelope Detector) ซึ่งมีการทำงานเหมือนกับวิธีการดีมอดูเลตของเครื่องรับวิทยุ ระบบ เอ.เอ็ม. เมื่อให้สัญญาณที่ได้ผ่านเข้าวงจรรวมสัญญาณแบบลบ ก็จะได้สัญญาณข้อมูลดิจิตอลตามแบบสัญญาณอินพุทของภาคมอดูเลเตอร์

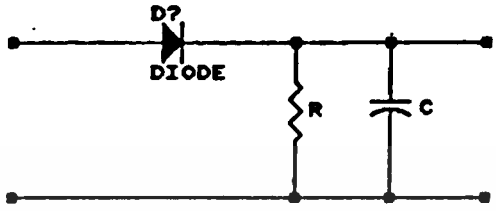


รูปที่ 2.4 การทำงานของภาค เอฟ.เอส.เค. ดีมอดูเลเตอร์

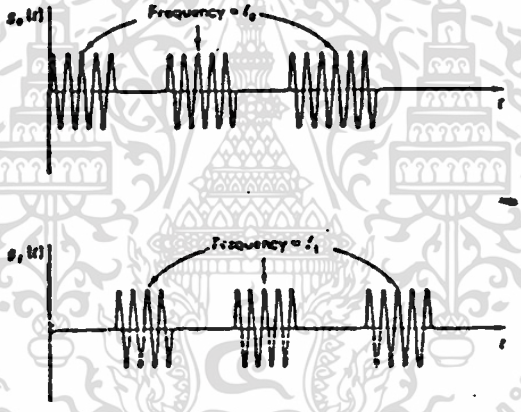
จากการพิจารณาวงจรกรองแถบความถี่ผ่าน ซึ่งเสมือนเป็นตัวกำหนดให้อัตราบิตในการส่งไม่สูง[3]เพราะว่า ถ้าให้ความถี่ f_1 และ f_2 มีค่าใกล้เคียงกันมากค่า Q ของวงจรกรองแถบความถี่ต้องสูงจึงสามารถที่จะแยกสัญญาณได้เด็ดขาด ดังนั้น ความถี่ f_1 และ f_2 จะต้องมีความถี่ห่างกันพอสมควรเพื่อป้องกันการผิดพลาดที่จะเกิดขึ้นในการแยกสัญญาณ

วงจรเอนเวลโลปดีเทคเตอร์ที่แสดงได้ดังรูปที่ 2.5 [1] มีการทำงานเหมือนกับการดีมอดูเลตใน เครื่องรับวิทยุ เอ. เอ็ม. หรือที่รู้จักกันในชื่อของวงจร

ไดโอดดีเทคเตอร์(Diode Detector) สัญญาณเอาท์พุทที่ออกจากวงจรกรองแถบความถี่ผ่านทั้งสองมีลักษณะดังรูป 2.6



รูปที่ 2.5 วงจรเอนเวลโลปดีเทคเตอร์

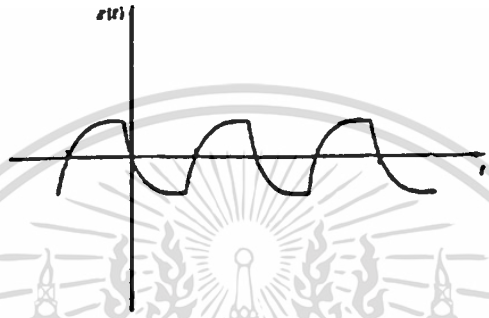


รูปที่ 2.6 สัญญาณอินพุตของวงจรเอนเวลโลปดีเทคเตอร์

หลักการทำงานของวงจรเอนเวลโลปดีเทคเตอร์ จะเลือกเอาสัญญาณชีกบวกของสัญญาณ $g_1(t)$ และ $g_2(t)$ เข้ามาประจุ (Charge) แรงดันให้แก่ตัวเก็บประจุ และคายประจุ(Discharge) จนกว่าแรงดันที่คร่อมตัวเก็บประจุมีค่าต่ำกว่าขนาดแรงดันที่เข้ามาใหม่ก็จะทำการเก็บประจุใหม่ จากการเลือกค่าของตัวต้านทานและตัวเก็บประจุที่ถูกต้อง จะได้แรงดันที่เอาท์พุตมีขนาดเปลี่ยนแปลงตามเอนเวลโลป

ของสัญญาณ

สัญญาณเอาท์พุทที่วงจรเอนเวลโลปดีเทคเตอร์ มีลักษณะเป็นแบบเอ็กโพเนนเชียล (Exponential) แสดงได้ดังรูป 2.7



รูปที่ 2.7 สัญญาณที่เอาท์พุตของวงจรเอนเวลโลปดีเทคเตอร์

ในการใช้วงจรกรองแถบความถี่ผ่านที่มีค่า Q ของวงจรสูงมาก ๆ จะทำให้การทำงานของวงจรเอนเวลโลปดีเทคเตอร์ทำงานผิดพลาดได้ง่าย ดังนั้น สัญญาณที่ออกจากวงจรเอนเวลโลปดีเทคเตอร์ Z_1 และ Z_2 จะถูกนำมาลบกันแล้วส่งให้วงจรเปรียบเทียบแรงดัน เพื่อให้ได้ข้อมูลกลับออกมาเป็นข้อมูลดิจิตอลต่อไป

จากหลักการมอดูเลตและดีมอดูเลตด้วยวิธี เอฟ.เอส.เค. นี้ได้นำมาใช้ในระบบโมเด็ม(Modem) ในการสื่อสารข้อมูลผ่านทางสายโทรศัพท์ จะเป็นโมเด็มที่มีอัตราความเร็วในการส่งต่ำ [2] เช่น ถ้าให้ความถี่สูงและความถี่ต่ำ มีค่า 2 กิโล-เฮิรตซ์ และ 1 กิโลเฮิรตซ์ตามลำดับ จะเห็นว่าในการมอดูเลตและดีมอดูเลตต้องใช้ 2 ไซเคิลต่อการส่งข้อมูล 1 บิต ที่การส่งความถี่สูงทำให้ส่งข้อมูลได้สูงสุดไม่เกิน 1200 บิตต่อวินาที หรือถ้าใช้ความถี่สูงและความถี่ต่ำต่างกันมาก ๆ เพื่อให้วงจรกรองแถบความถี่ผ่านไม่ต้องมีค่า Q สูง ๆ ก็จะเป็นตัวทำให้อัตราการส่ง ถูกจำกัด

ด้วยความถี่ต่ำสุดที่เลือกใช้ เพราะว่าความถี่ต่ำนี้จะเป็นตัวกำหนดอัตราส่งต่ำสุดที่ส่งได้ ดังนั้นจะเห็นว่าอัตราการส่ง-รับข้อมูล จะถูกกำหนดด้วยวงจรกรองแถบความถี่ผ่านของภาคดีมอดูเลเตอร์ด้วย[7]

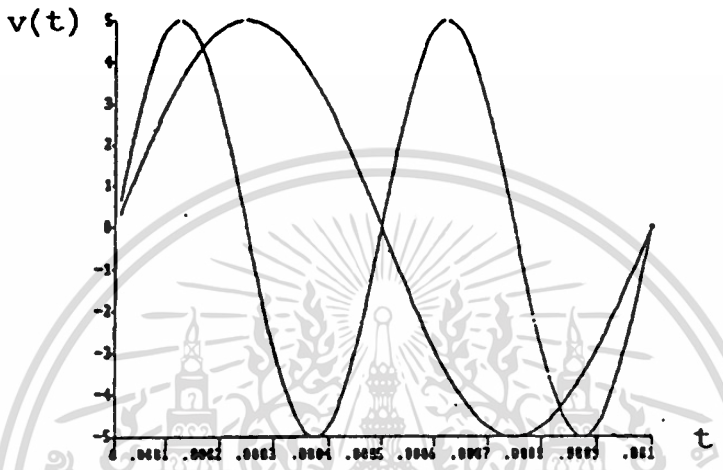
อีกสิ่งหนึ่งที่เป็นตัวจำกัดอัตราการส่ง - รับ ข้อมูลก็ คือ เฟสของสัญญาณไม่ต่อเนื่องตรงจุดที่การเปลี่ยนแปลงของสัญญาณข้อมูลจาก 0 เป็น 1 หรือจาก 1เป็น 0 ดังรูปที่ 2.3 เพราะจะเกิดสเปคตรัมของสัญญาณความถี่สูง และจะทำให้การทำงานของภาคดีมอดูเลเตอร์ทำงานผิดพลาด โมเด็มที่ใช้วิธี เอฟ.เอส.เค. ที่จัดอยู่ในกลุ่มความเร็วต่ำมีค่าอัตราบิตไม่เกิน 1200 บิตต่อวินาที เช่น Bell 103, Bell 202, CCITT V21 และ CCITT V23 เป็นต้น

2.4 แนวความคิดในการแก้ปัญหาระบบ เอฟ.เอส.เค.มอดูเลต ระบบเดิม

จากการทำงานของระบบ เอฟ.เอส.เค.มอดูเลต จากที่กล่าวมา จะเห็นว่าค่าอัตราบิตสูงสุดถูกจำกัดอยู่ที่ความถี่ต่ำที่ใช้งาน เช่นถ้ากำหนดให้ f_1 เป็นความถี่ด้านต่ำ มีความถี่เท่ากับ 1 กิโลเฮิรตซ์ แทนสัญญาณไบนารีที่สภาวะลอจิก "0" และความถี่ f_2 เป็นความถี่สูงมีความถี่เท่ากับ 2 กิโลเฮิรตซ์ แทนสภาวะลอจิก "1" ดังนั้น ในการทำงานเมื่อสภาวะลอจิก "1" จะมีคาบเวลา 1 ไซเคิลเท่ากันเท่ากับ 0.0005 วินาทีและถ้าสภาวะลอจิก "0" จะมีคาบเวลา 0.001 วินาที ดังรูปที่ 2.8

ดังนั้น จะเห็นว่าเวลาที่ใช้อย่างน้อยที่สุดต้องมีค่าเท่ากับ 0.001 วินาที ต่อสัญญาณดิจิตอลที่มีสภาวะเป็น "0" ในการส่งสัญญาณดิจิตอลที่มีสภาวะลอจิก "0" และ "1" ความถี่ f_2 ที่ใช้ในการส่งสัญญาณ เอฟ.เอส.เค. จะมีค่าเป็น 2 เท่าของ f_1 เพราะว่า จะมีความสะดวกในการออกแบบวงจรกรองแถบความถี่ผ่านของภาคดีมอดูเลเตอร์ ดังนั้น ในการส่งสัญญาณดิจิตอลที่มีสภาวะลอจิก "0" และ "1" ถือว่าเป็นตัวกำหนดอัตราบิตของการส่งต่ำสุด ซึ่งสามารถคำนวณค่าอัตราบิตของการส่งได้

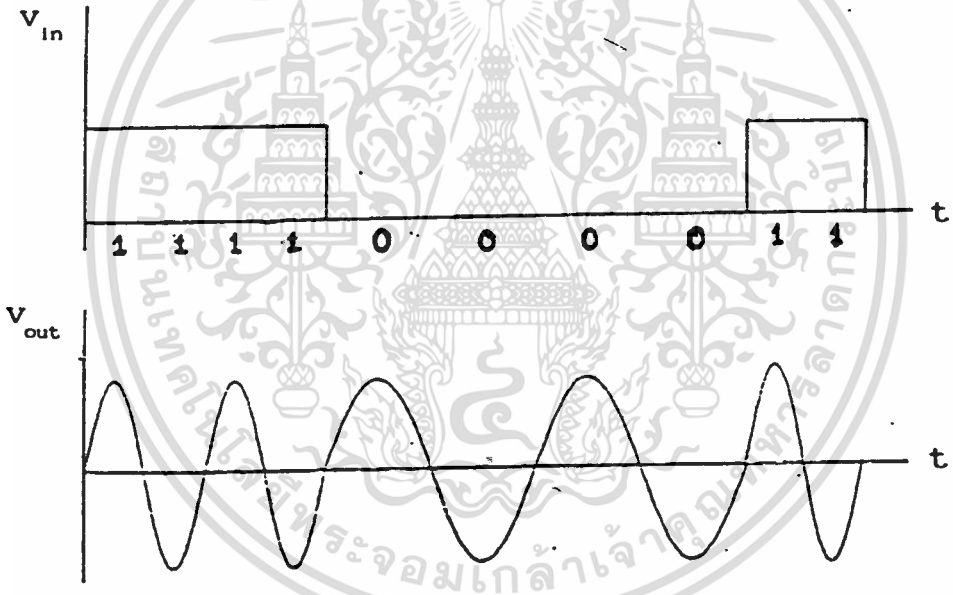
เท่ากับ 1000 บิตต่อวินาที สำหรับการส่งความถี่ไปหนึ่งคาบเวลาต่อข้อมูล 1 บิต แต่ในการส่งจริง ๆ แล้วเพื่อให้การอ่านข้อมูลแน่นอน การส่งข้อมูล 1 บิตจะใช้เวลามากกว่า1 คาบเวลา



รูปที่ 2.8 รูปคลื่น 2 ความถี่ 1 kHz และ 2 kHz

จากรูปที่ 2.8 จะเห็นว่าถ้าเวลาในการส่งข้อมูล 1 บิตเท่ากันระหว่างในการส่งข้อมูลไบนารีสภาวะลอจิก "0" และ "1" จะต้องทำการส่งความถี่สูงไป 2 ไซเคิล ทำให้เสียเวลาในการส่งไปโดยเปล่าประโยชน์ 0.0005 วินาที และถ้าเพิ่มความถี่ต่ำให้ใกล้เคียงกับความถี่สูงเป็น 1.9 กิโลเฮิรตซ์ ซึ่งมีคาบเวลาเท่ากับ 0.000526 วินาที ก็ยังเกิดเวลาสูญเปล่าเท่ากับ 0.000526 - 0.0005 วินาที (0.000026 วินาที) แต่สามารถเพิ่มอัตราบิตได้เป็น 1900 บิตต่อวินาที จากการเพิ่มความถี่ต่ำใกล้เคียงกับความถี่สูงมากเท่าไรยิ่งทำให้เสียเวลาสูญเปล่าน้อยลงมากเท่านั้น แต่เนื่องจากความถี่ใกล้เคียงกันมาก วงจรกรองแถบความถี่ผ่านในภาคดีมอดูเลเตอร์จะต้องมีค่า Q สูงมากด้วยเพื่อตัดสัญญาณได้เด็ดขาด และเมื่อใช้วงจรที่มีค่า Q สูง ๆ โอกาสเกิดการผิดพลาดเนื่องจากการเลื่อนความถี่เพียงเล็กน้อยก็มี

มาก ดังนั้น แนวความคิดในการเพิ่มอัตราบิตของการส่งให้สูงขึ้นใน ระบบ เอฟ. เอส. เค. ตามโครงการนี้ จะกำหนดให้ตัวกำเนิดสัญญาณ เอฟ. เอส. เค. กำเนิดสัญญาณออกมาเพียงครึ่งคาบเวลาของแต่ละความถี่ต่อสัญญาณข้อมูล 1 บิต เพื่อให้เกิดการสูญเปล่าในการส่งน้อยที่สุด นั่นคือ ถ้าให้ f_1 และ f_2 มีความถี่ 1 และ 2 กิโลเฮิรตซ์ ตามลำดับ ถ้าข้อมูลเป็น "0" ทั้งหมด อัตราบิตของระบบมีค่า 2000 บิตต่อวินาที และมีค่า 4000 บิตต่อวินาที เมื่อข้อมูลเป็น "1" ทั้งหมด จากรูปแบบของการมอดูเลตแบบ เอฟ. เอส. เค. โดยใช้สัญญาณครึ่งคาบเวลาต่อข้อมูล 1 บิต ตามที่ได้กล่าวมาแล้วจะมีรูปสัญญาณดังรูป 2.9



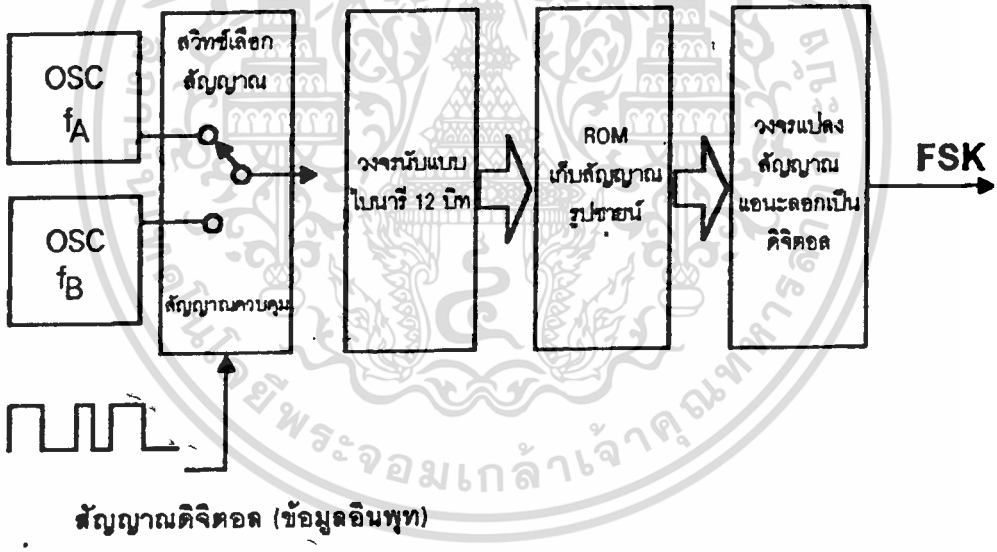
รูป 2.9 รูปแบบสัญญาณ เอฟ. เอส. เค. แบบครึ่งคาบเวลาต่อ 1 บิต

จากรูป 2.9 จะเห็นว่ารูปแบบของสัญญาณ เอฟ. เอส. เค. ก็จะเป็นสัญญาณแบบต่อเนื่องและไม่มีการสูญเสียเวลาโดยเปล่าประโยชน์เหมือนระบบ เอฟ. เอส. เค. แบบเดิม และทางภาคดีมอดูเลเตอร์จะต้องสามารถทำการดีมอดูเลตสัญญาณกลับมา

ได้ถูกต้อง

2.5 แนวทางการออกแบบภาคมอดูเลเตอร์

จากแนวความคิดถ้าใช้วงจรในลักษณะเดิมจะเป็นการยากที่จะควบคุมสัญญาณความถี่จากตัวออสซิลเลเตอร์โดยตรงให้มีสัญญาณออกมาเพียงครึ่งคาบเวลา และให้สัญญาณที่ได้เป็นแบบต่อเนื่อง ดังนั้นทางออกคือแทนที่จะใช้สัญญาณจากออสซิลเลเตอร์หลักมาทำเป็นตัวกำเนิดสัญญาณ เอฟ.เอส.เค. โดยตรงก็เปลี่ยนมาให้สัญญาณจากออสซิลเลเตอร์ไปทำการควบคุมตัวกำเนิดสัญญาณ เอฟ.เอส.เค.[5] อีกต่อหนึ่งมีรูปแบบการทำงานดังรูปที่ 2.10



รูป 2.10 แผนภาพภาคมอดูเลเตอร์ที่ออกแบบ

จากแผนภาพการทำงานของวงจร สามารถอธิบายการทำงานได้ดังนี้ f_A และ f_B เป็นความถี่ที่ได้จากออสซิลเลเตอร์ เมื่อสัญญาณข้อมูลดิจิตอลแบบไบนารีเข้า

มาควบคุมสวิทช์เลือกสัญญาณความถี่โดยจะทำตามสภาวะของลอจิกที่อินพุท ทำให้สัญญาณที่เอาท์พุทของสวิทช์เลือกมีการเปลี่ยนความถี่ตามสภาวะลอจิกที่เข้ามา

สัญญาณ f_A หรือ f_B ที่ออกจากวงจรสวิทช์เลือกสัญญาณจะเข้าไปยังวงจรนับแบบไบนารี (Binary Counter) ดังนั้นที่ขาสัญญาณนาฬิกา(Clock)' ของวงจรนับแบบไบนารี จะได้รับสัญญาณนาฬิกาตามสภาวะของลอจิกที่เข้ามาควบคุม ถ้าให้ความถี่ f_A เมื่อสภาวะลอจิก "0" เข้ามา f_B เมื่อสภาวะลอจิก "1" เข้ามา และเมื่อลอจิก "0" เข้ามาจะสวิทช์เลือกสัญญาณความถี่ f_A เข้ามาเป็นสัญญาณนาฬิกาให้กับวงจรนับแบบไบนารีขนาด 12 บิตและในทำนองเดียวกันถ้าสภาวะลอจิก "1" เข้ามาความถี่ f_B ก็จะเป็นสัญญาณนาฬิกาให้กับวงจรนับแบบไบนารี ซึ่งวงจรนับแบบไบนารีนี้จะต่อกับหน่วยความจำแบบรอม(ROM) ซึ่งเราจะเก็บสัญญาณรูปไซน์ (Sine Wave) ไว้ วงจรนับแบบไบนารีเป็นตัวสร้างค่าตำแหน่งของรอม เพื่อให้อ่านสัญญาณรูปไซน์ออกมาทางขาข้อมูลของรอม ดังนั้นในกรณีที่ขานาฬิกาของวงจรนับแบบไบนารีเป็น f_B ความเร็วในการอ่านข้อมูลออกจากรอมจะเร็วกว่าที่ได้รับความถี่ f_A ซึ่งหมายความว่าสัญญาณรูปไซน์ที่ได้ออกมากรึ่งคาบเวลา จะช้าหรือเร็วขึ้นอยู่กับสถานะลอจิกทางด้านอินพุท

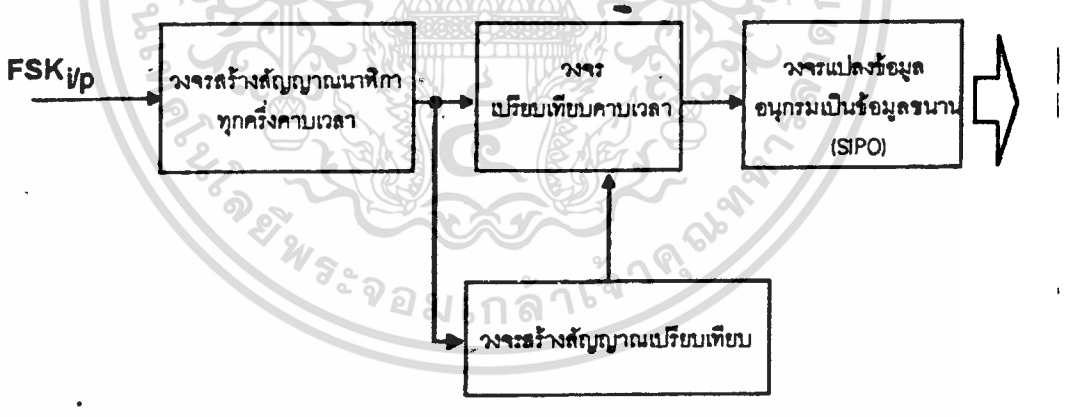
จากการที่ใช้วงจรนับแบบไบนารีขนาด 12 บิต เป็นตัวอ่านสัญญาณรูปไซน์ออกจากหน่วยความจำแบบรอมนั้น ทำให้เราสามารถกำหนดค่าตำแหน่งสุดท้ายและกึ่งกลางของรูปไซน์ได้ ถ้าขนาดรูปไซน์มีความละเอียด 1024 จุด จะได้ตำแหน่งที่ครึ่งคาบเวลาและสุดท้ายของสัญญาณรูปซายน์ที่เก็บไว้ในรอมก็จะเป็นตำแหน่งที่ 512 และ 1024 เช่นกัน หมายความว่าเมื่อใดก็ตามที่วงจรนับแบบไบนารีนับมาถึงตำแหน่งที่ 512 เราจะได้สัญญาณรูปไซน์ครึ่งคาบเวลา อย่างแน่นอน และใช้สัญญาณของวงจรนับแบบไบนารีตำแหน่งที่ 512 ป้อนกลับไปควบคุมวงจรส่วนหน้าให้ทราบเพื่อทำการส่งข้อมูลไบนารีบิตต่อไปเข้ามา เพื่อให้วงจรทำการนับต่อไปให้ถึงตำแหน่งที่

1024 ก็จะได้รูปสัญญาณไซน์อีกครึ่งคาบเวลาที่อยู่ในหน่วยความจำรอม และเอาสัญญาณจากวงจรนับ 1024 ไปควบคุมวงจรส่วนหน้าให้ทำการส่งข้อมูลไบนารีต่อไป

ดังนั้น สัญญาณ เอฟ.เอส.เค. ที่ได้เป็นแบบครึ่งคาบเวลาต่อข้อมูลไบนารี 1 บิตและเป็นสัญญาณอย่างต่อเนื่อง

2.6 แนวทางการออกแบบภาคดีมอดูเลเตอร์

จากวงจร เอฟ.เอส.เค.ดีมอดูเลเตอร์ของวิธีการที่กล่าวมาแล้วนั้น มีข้อเสียหลายอย่างในการใช้วงจรกรองแถบความถี่ผ่าน และที่สำคัญคือมีขีดจำกัดในการที่ทำให้ไม่สามารถเพิ่มอัตราบิตของการส่งสัญญาณด้วยวิธี เอฟ.เอส.เค. ในวิธีการเดิมได้ ดังนั้น ในโครงงานนี้จะนำเสนอวงจรภาคดีมอดูเลตสัญญาณเพื่อรับสัญญาณ เอฟ.เอส.เค. ที่ส่งอัตราบิตสูงกว่าวิธีการเดิมมีแผนภาพการทำงานดังรูปที่ 2.11

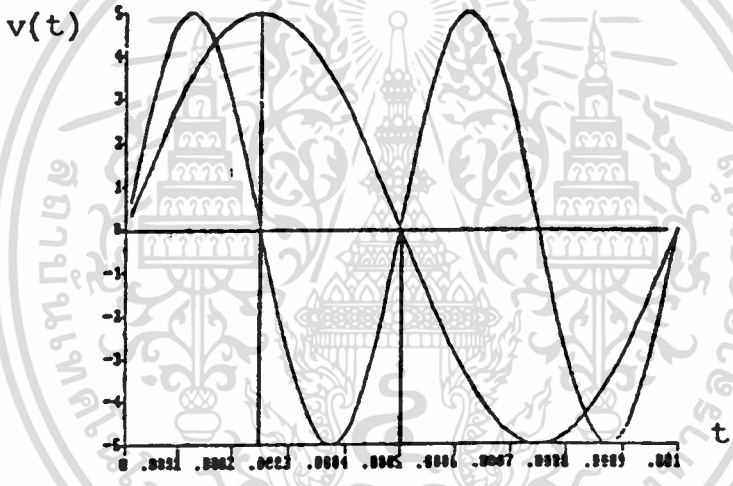


รูปที่ 2.11 แผนภาพภาคดีมอดูเลเตอร์

จากการส่งสัญญาณด้วยวิธีเอฟ.เอส.เค. ออกมาเพียงครึ่งไซเคิลต่อ 1 บิต สามารถทำการตรวจสอบคาบเวลาของสัญญาณที่เข้ามา เปรียบเทียบกับสัญญาณเวลา

ที่มีค่าคงที่ค่าหนึ่ง โดยเริ่มทำการตรวจสอบทุก ๆ ครึ่งไซเคิล กล่าวคือ ถ้าสมมุติว่าสัญญาณเอฟ.เอส.เค. เป็นการเปลี่ยนแปลงของสัญญาณความถี่ 1.0 กิโลเฮิรตซ์และ 2.0 กิโลเฮิรตซ์ แล้วจะพบว่าในครึ่งไซเคิลของความถี่มีค่าต่างกัน 2 เท่า

จากรูปที่ 2.12 จะเห็นว่าที่ครึ่งคาบเวลาของความถี่ 1 กิโลเฮิรตซ์มีค่า 0.0005 วินาทีและที่ความถี่ 2 กิโลเฮิรตซ์มีค่า 0.00025 วินาที จะเห็นว่าค่าเวลาครึ่งคาบเวลาของความถี่ 2 กิโลเฮิรตซ์ มีค่าเพียง.1/4 คาบเวลาของความถี่ 1 กิโลเฮิรตซ์



รูป 2.12 แสดงคาบเวลาของ 2 ความถี่ 1 kHz และ 2 kHz

ดังนั้น ถ้านำสัญญาณ เอฟ.เอส.เค. ที่ส่งด้านความถี่สูงมีค่าเป็น 2 เท่าของความถี่ต่ำที่รับเข้ามาไปเปรียบเทียบกับสัญญาณพัลส์ที่มีค่าคาบเวลา เท่ากับ 1/4 ของความถี่ต่ำหรือมากกว่าจะเห็นว่าถ้าครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. ที่เข้ามามีช่วงเวลาเท่ากับสัญญาณพัลส์ที่นำมาเปรียบเทียบ จะได้ค่าสัญญาณลอจิกที่มีค่าเป็น 1 ออกมา และถ้าครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. มีเวลามากกว่า

สัญญาณพัลส์เปรียบเทียบก็จะได้สัญญาณลอจิกที่มีสภาวะเป็น 0 และสัญญาณลอจิกที่ได้ออกมาในส่วนนี้ เป็นข้อมูลดิจิตอลแบบไบนารีที่มีค่าเวลาของลอจิกสภาวะ 0 และ 1 ที่ไม่เท่ากัน จึงไม่สามารถที่จะเอาข้อมูลในส่วนนี้ออกมาอ่านโดยตรงได้ เมื่อต้องการที่จะอ่านข้อมูลนี้ออกมาให้ถูกต้อง จะต้องนำสัญญาณข้อมูลดิจิตอลไปผ่านยังวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน(Serial In Parallel Out)เพื่อให้ได้ข้อมูลออกมาถูกต้องตรงกับสัญญาณข้อมูลที่ส่งมาจากภาคมอดูเลเตอร์

บทที่ 3
การออกแบบภาคมอดูเลต
สัญญาณ เอฟ. เอส. เค.

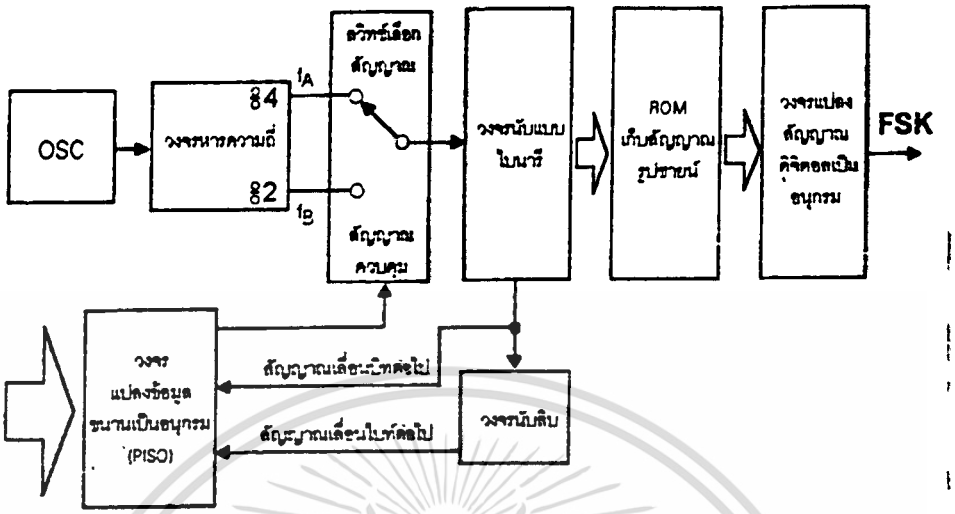
3.1 กล่าวนำ

จากบทที่ 2 ได้กล่าวถึงแนวทางในการออกแบบภาคมอดูเลเตอร์มาแล้วนั้น จะเห็นว่ามีการควบคุมการเกิดสัญญาณ เอฟ. เอส. เค. ที่มีรูปลักษณะแบบครึ่งคาบเวลาต่อข้อมูล 1 บิต ตามที่ได้อธิบายในหลักการกว้าง ๆ ในการควบคุมการทำงานของภาคออสซิลเลเตอร์มาแล้ว ในบทที่ 2 ซึ่งเป็นส่วนสำคัญมากในส่วนของภาคมอดูเลเตอร์ หรือชุดส่งสัญญาณ เอฟ. เอส. เค. บทนี้ จะกล่าวถึงการออกแบบและวงจรที่ใช้งานจริง ๆ ที่ได้ทดลองสร้างขึ้น

3.2 การทำงานของภาคมอดูเลตสัญญาณ เอฟ. เอส. เค

การทำงานของภาคมอดูเลเตอร์ที่จะทำการแปลงข้อมูลดิจิตอล 2 ระดับ ให้เป็นสัญญาณครึ่งไซเคิลของความถี่ f_1 และ f_2 สามารถนำมาเขียนเป็นแผนภาพได้ดังรูปที่ 3.1

จากแผนภาพการทำงานภาคมอดูเลเตอร์ สามารถจะแยกการทำงานของวงจรเป็นส่วนใหญ่ ๆ ได้เป็น 2 ส่วน[5] ในส่วนที่ 1 มีวงจรประกอบด้วย วงจรกำเนิดความถี่(Oscillator) วงจรเปลี่ยนสัญญาณอินพุทจากข้อมูลดิจิตอลแบบขนานเป็นแบบอนุกรม และวงจรควบคุมในการเลือกความถี่ ปล่อยบิตและไบท์ ต่อไปของสัญญาณอินพุท และส่วนที่ 2 มีวงจรประกอบด้วย วงจรนับแบบไบนารี หน่วยความจำแบบรอม(ROM) ที่เก็บข้อมูลรูปสัญญาณไซน์ และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนาลอก



รูป 3.1 แผนภาพแสดงการทำงานของภาคมอดูเลเตอร์

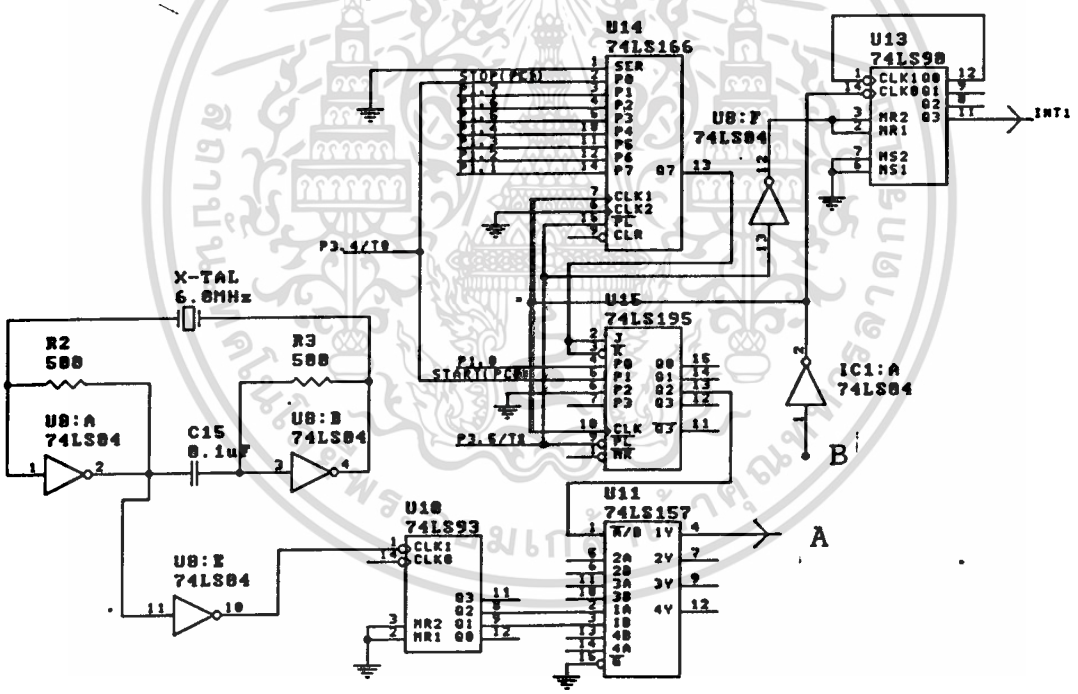
จากส่วนที่ 1 วงจรภาคกำเนิดสัญญาณความถี่หลักทางภาคมอดูเลเตอร์ และวงจรหารความถี่เป็นการหาร 4 และหาร 2 ของความถี่หลัก เพื่อให้ได้ความถี่ออกมาต่างกันเป็นความถี่ f_A และ f_B ผ่านเข้าไปยังวงจรสวิทช์เลือก สัญญาณควบคุมในการเลือกสัญญาณความถี่ f_A และ f_B ตามสถานะลอจิกของข้อมูลไบนารีเข้ามาทางอินพุทของภาคมอดูเลเตอร์ ผ่านวงจรแปลงข้อมูลจากขนานเป็นอนุกรม โดยจะทำการเลื่อนข้อมูลแบบอนุกรมออกไปทีละบิตตามสัญญาณควบคุม เมื่อทำการเลื่อนข้อมูลไบนารีครบ 10 บิตแล้ว จะมีสัญญาณอินเตอร์รัพท์เพื่อไปควบคุมให้ส่วนอินพุทรับข้อมูลอินพุตไบท์ต่อไปค้างไว้ (Latch) เพื่อรอทำการส่งต่อไป

จากส่วนที่ 2 วงจรนับแบบไบนารี จะทำการนับสัญญาณความถี่ความถี่ f_A และ f_B ที่เข้ามาเพื่อสร้างค่าตำแหน่ง (Address) ให้กับหน่วยความจำแบบรอมเพื่ออ่านข้อมูลออกมาแล้ว นำข้อมูลที่อ่านได้ไปเข้ายังวงจรเปลี่ยนสัญญาณดิจิตอลเป็นแอนาลอก(Digital to Analog Converter, DAC) ทำหน้าที่แปลงค่าสัญญาณที่ออกมาจากวงจรหน่วยความจำแบบรอม ให้เป็นสัญญาณรูปไซน์ที่เปลี่ยนค่าความถี่ทุกครึ่งคาบเวลาตามความถี่ที่ป้อนให้กับวงจรนับแบบไบนารี

สำหรับสัญญาณอีกส่วนหนึ่งที่มาใช้เป็นสัญญาณควบคุม เพื่อให้มีการเลื่อนไบท์ต่อไป ได้มาจากวงจรควบคุมซึ่งในวงจรควบคุมส่วนนี้ใช้ ซีพียู 8031 ซึ่งจะส่งสัญญาณออกมาเมื่อมีการส่งข้อมูลไปครบ 10 บิตแล้ว และเป็นวงจรที่เปลี่ยนข้อมูลไบนารีแบบอนุกรมจากเครื่องคอมพิวเตอร์ให้เป็นข้อมูลไบนารีแบบขนานเพื่อป้อนเป็นสัญญาณอินพุทของภาคมอดูเลเตอร์

3.2.1 การทำงานของวงจรส่วนที่ 1

จากการทำงานของวงจรส่วนที่ 1 ในการทำการทดลอง มีรายละเอียดของวงจรในการทำงานดังรูปที่ 3.2



รูปที่ 3.2 วงจรทดลองส่วนที่ 1 ของภาคมอดูเลเตอร์

วงจรกำเนิดสัญญาณความถี่หลักแบบควบคุมความถี่ด้วยคริสตอล(X-TAL) ความถี่ 6 เมกกะเฮิรตซ์ ซึ่งเป็นความถี่หลักของภาคมอดูเลเตอร์ เหตุที่ใช้คริสตอล

ควบคุมวงจรกำเนิดสัญญาณความถี่หลักเนื่องจากสัญญาณที่ได้มีความถี่เที่ยงตรงสูง จากวงจรกำเนิดความถี่หลักสามารถเลือกได้ว่า ให้ทำการหาร 2, 4 หรือ 8 ของความถี่หลัก โดยความถี่หลักผ่านเข้ายังวงจรนับแบบไบนารีขนาด 4 บิต จากการทำงานของวงจรเลือกใช้ความถี่หาร 4 และ 2 จะได้สัญญาณความถี่ 1.5 เมกกะเฮิรตซ์ และ 3.0 เมกกะเฮิรตซ์ ให้เป็นความถี่ f_A และ f_B ตามลำดับ

จากสัญญาณความถี่ f_A และ f_B ผ่านเข้าสู่วงจรสวิตช์เลือกสัญญาณ ในวงจรทดลองใช้ ไอซีเบอร์ 74LS157 เป็นวงจรสวิตช์เลือกข้อมูลจาก 2 อินพุทออก 1 (Data Selectors 2 line to 1 line) ความถี่ที่เอาท์พุท จะมีค่าเป็น f_A หรือ f_B จะถูกควบคุมในการเลือกโดยสภาวะลอจิกที่ขาซีเล็ค(Select) ตามสัญญาณข้อมูลอินพุท เมื่อมีสภาวะลอจิกเป็น 0 วงจรจะเลือกเอาความถี่ f_A ออกไป และถ้ามีสภาวะลอจิกเป็น 1 จะเลือกความถี่ f_B ออกไป

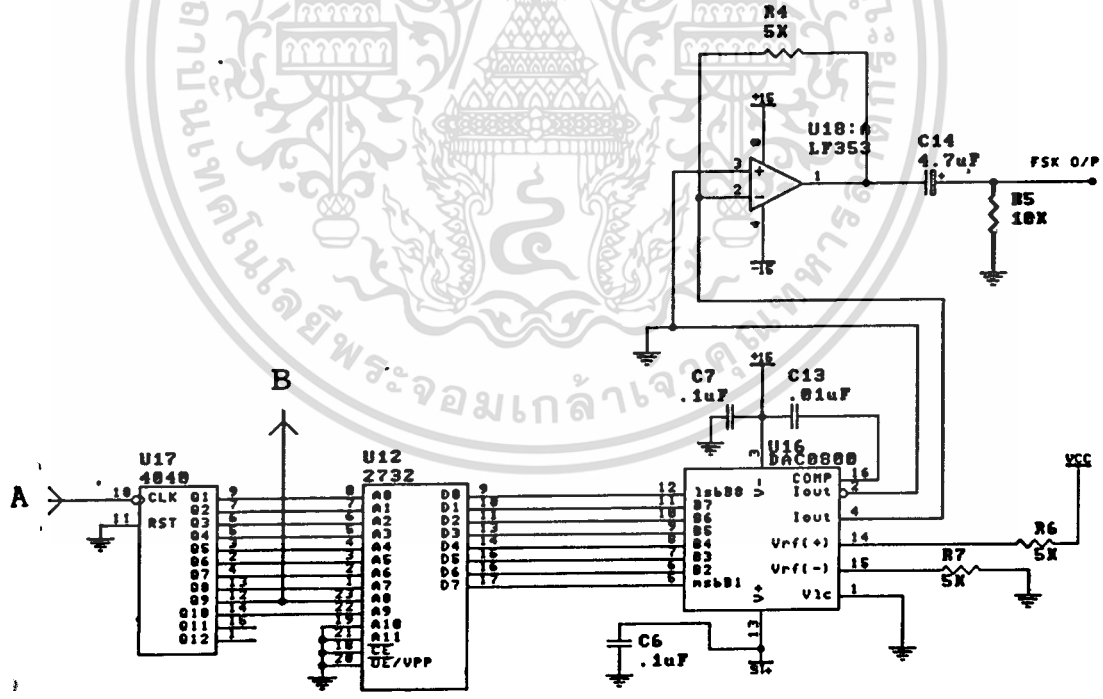
วงจรเปลี่ยนรูปสัญญาณอินพุทที่เป็นข้อมูลดิจิตอลแบบขนาน ให้เป็นข้อมูลแบบอนุกรมใช้ ไอซีเบอร์ 74LS166 และ 74LS195 เป็นไอซีแบบ 8 Bit และ 4 Bit Shift Register ตามลำดับ เพราะในการทำการทดลองโครงการนี้ได้ทำการส่งข้อมูลสัญญาณดิจิตอลขนาด 10 บิต (เป็นสัญญาณข้อมูล 8 บิตและมี Start Bit กับ Stop Bit อีก 2 บิต) และในการควบคุมให้เลื่อนข้อมูลออกไปทีละบิตได้ตามสัญญาณที่มาจากส่วนที่ 2 เป็นสัญญาณนาฬิกาที่เข้าไปควบคุม Shift Register ให้เลื่อนบิตต่อไป และเมื่อมีสัญญาณจากวงจรส่วนที่ 2 ออกมาเมื่อมีการส่งสัญญาณครบทุกครึ่งคาบเวลาของความถี่ที่จะใช้ส่งสัญญาณ เอฟ.เอส.เค. จะทำให้เกิดสัญญาณนาฬิกาไปป้อนให้กับวงจร Shift Register และวงจรนับสิบ (Decade counter) ซึ่งใช้ ไอซีเบอร์ 74LS90 เพื่อควบคุมให้เลื่อนบิตต่อไปทุกครึ่งคาบเวลา และเวลาในการเลื่อนบิตจะไม่เท่ากันตลอด เพราะขึ้นอยู่กับข้อมูลที่เข้ามา ส่วนสัญญาณที่เข้าวงจรนับสิบครบ 10 พัลส์ ก็แสดงว่าทำการส่งข้อมูลครบ 10 บิตแล้ว

จะมีสัญญาณอินเตอร์รัพท์ไปยังวงจรควบคุม ให้ทำการส่งสัญญาณชิฟโหลด เพื่อให้ส่วนอินพุททำการรับข้อมูลไบท์ต่อไปเข้ามา ซึ่งควบคุมโดยสัญญาณจากวงจรควบคุมทาง Port 3.4 /T1 เพื่อให้วงจร Shift Register ทำการรับข้อมูลไบท์ต่อไปเข้ามาค้างค่า (Latch)ไว้

จะเห็นว่ามีการส่งข้อมูลไปทีละบิต เป็นข้อมูลดิจิตอลแบบอนุกรมที่มีค่าเวลาของสัญญาณลอจิกจากเอาท์พุทของไอซีเบอร์ 74LS195 มีค่าไม่เท่ากัน และสัญญาณนี้จะไปควบคุมวงจรสวิสช์เลือกความถี่ซึ่งใช้ ไอซีเบอร์ 74LS157 ต่อไป

3.2.2 การทำงานของวงจรส่วนที่ 2

วงจรในส่วนที่ 2 ในการทำการทดลอง มีรายละเอียดของวงจรการทำงานดังรูปที่ 3.3



รูปที่ 3.3 วงจรส่วนที่ 2 ของภาคมอดูเลเตอร์

การทำงานในส่วนนี้ เริ่มจากวงจรนับแบบไบนารีใช้ ไอซีเบอร์ 4040 ซึ่งเป็นวงจรนับแบบไบนารีขนาด 12 บิต นำค่าที่ได้มาเป็น แอดเดรส สร้างค่าตำแหน่งให้กับวงจรหน่วยความจำแบบรอม ซึ่งเก็บตารางค่าของสัญญาณรูปไซน์ที่แบ่งออกเป็น 256 ค่า (0 - 255) เพื่อให้วงจรหน่วยความจำแบบรอมส่งข้อมูลของสัญญาณรูปไซน์ออกมาแล้วไปเข้ายังวงจรเปลี่ยนสัญญาณดิจิตอลเป็นแอนาลอก เพื่อให้ได้สัญญาณ เอฟ.เอส.เค. เมื่อพิจารณาดูวงจรจะเห็นว่าสัญญาณนาฬิกาของวงจรนับแบบ 12 บิต จะเป็นความถี่ f_B ซึ่งมีความถี่เท่ากับ 3 เมกกะเฮิรตซ์ เมื่อสถานะข้อมูลของลอจิกที่อินพุทของวงจรส่วนที่ 1 มีสถานะลอจิกเป็น 1 ข้อมูลของสัญญาณรูปไซน์ในหน่วยความจำแบบรอม จะถูกอ่านออกไปครึ่งคาบเวลาของความถี่ f_2 โดยใช้เวลาประมาณ 0.00017 วินาที หรือเมื่อเทียบเป็นความถี่ จะได้ว่า f_2 มีค่าเท่ากับ 2929.6875 เฮิรตซ์ หรือประมาณ 2930 เฮิรตซ์

(การ Samping 1 จุด ใช้เวลาอ่าน = 0.00000033 วินาที
" 1,024 จุด ใช้เวลาอ่าน = 0.00034133 วินาที
ครึ่งคาบเวลา 512 จุด ใช้เวลาอ่าน = 0.00017067 วินาที
และ ความถี่ f_2 = 1/0.00034133 = 2930 เฮิรตซ์)

ถ้าความถี่ f_A เท่ากับ 1.5 เมกกะเฮิรตซ์ และถ้าอินพุทของวงจรส่วนที่ 1 มีสถานะลอจิก เป็น 0 ดังนั้น ครึ่งคาบเวลาของความถี่ f_1 มีค่าเท่ากับ 0.00034133 วินาที เมื่อเทียบเป็นความถี่ f_1 มีค่าเท่ากับ 1464.84375 เฮิรตซ์ หรือประมาณ 1465 เฮิรตซ์ และจากการที่ข้อมูลของสัญญาอินพุทเข้ามา 1 บิต เมื่ออ่านค่าสัญญาณรูปไซน์ที่เก็บไว้ในหน่วยความจำแบบรอม ออกมาครบครึ่งคาบเวลาแล้วก็ จะมีสัญญาณกลับมาควบคุมให้วงจรเปลี่ยนข้อมูลแบบขนานเป็นอนุกรม(PISO) ทำ

การส่งข้อมูลบิตต่อไป และหน่วยความจำแบบรอมก็จะอ่านข้อมูลอีกครึ่งคาบเวลา ตามความถี่สัญญาณนาฬิกาที่เข้ามาที่อินพุทของวงจรนับแบบไบนารีขนาด 12 บิต ดังนั้น เมื่อข้อมูลเข้ามา 2 บิต จึงจะได้ค่าสัญญาณรูปไซน์ออกมาครบ 1,024 จุด และข้อมูลที่ออกมาจากหน่วยความจำรอม ก็ส่งไปเข้ายังวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนาลอก ไอซีเบอร์ DAC 0800 ซึ่งจะเห็นว่าสัญญาณรูปไซน์ที่ได้ออกมาจากวงจรแปลงสัญญาณดิจิตอลเป็นแอนาลอก ในครึ่งคาบเวลานั้นจะมีค่าเปลี่ยนแปลงทุกครึ่งคาบเวลา ถ้าข้อมูลที่เข้ายังวงจรอินพุทของวงจรส่วนที่ 1 มีค่าลอจิกเปลี่ยนสถานะไป และจะไม่เปลี่ยนเมื่อลอจิกที่เข้ามามีค่าสถานะเดียวกัน

ดังนั้น จะเห็นว่าในการส่งข้อมูลอย่างต่อเนื่อง 10 บิต เข้าที่อินพุทส่วนที่ 1 จะต้องใช้สัญญาณรูปไซน์ในการส่งเพียง 5 ไซเคิล ซึ่งคาบเวลาของสัญญาณขึ้นอยู่กับข้อมูลที่ส่งเข้ามา ดังเช่น ถ้าส่งข้อมูล ลอจิก 1 ทั้ง 10 บิต จะใช้วลาในการส่ง 0.001707 วินาที และถ้าลอจิกทั้ง 10 บิตเป็นลอจิก 0 จะใช้เวลาในการส่ง 0.003413 วินาที

จากข้อมูลที่ได้มาจะเห็นว่ารูปสัญญาณ เอฟ.เอส.เค. ที่ได้จากวงจรเปลี่ยนสัญญาณดิจิตอลเป็นแอนาลอก จะมีค่าสูงสุดที่อัตราประมาณ 5.86 กิโลบิตต่อวินาที และต่ำสุดประมาณ 2.93 กิโลบิตต่อวินาที

ในวงจรส่วนที่ 2 นั้น ส่วนที่สำคัญที่จะกล่าวถึง ก็คือ หน่วยความจำแบบรอม เพราะว่าเป็นตัวกำหนด ค่าสัญญาณรูปไซน์ ซึ่งโครงการนี้เลือกใช้หน่วยความจำแบบรอมเป็น EPROM(Erasable PROM) ไอซีเบอร์ 2732 ซึ่งเป็นหน่วยความจำแบบรอมที่มีขนาด 8x4 กิโลบิต ในการเกิดข้อมูลสัญญาณรูปไซน์มีความละเอียด 1,024 จุด โดยข้อมูลทั้งหมดมีค่า 1,024ค่า นำมาสร้างตารางข้อมูลได้[6] ดังนี้

```
DFB  127,128,129,130,130,131,132,133,134,134
DFB  135,136,137,138,138,139,140,141,142,142
DFB  143,144,145,146,146,147,148,149,150,150
DFB  151,152,153,153,154,155,156,157,157,158
DFB  159,160,160,161,162,163,163,164,165,166
DFB  167,167,168,169,170,170,171,172,173,173
DFB  174,175,175,176,177,178,178,179,180,181
DFB  181,182,183,183,184,185,186,186,187,188
DFB  188,189,190,190,191,192,193,193,194,195
DFB  195,196,197,197,198,199,199,200,201,201
DFB  202,202,203,204,204,205,206,206,207,207
DFB  208,209,209,210,210,211,212,212,213,213
DFB  214,215,215,216,216,217,217,218,219,219
DFB  220,220,221,221,222,222,223,223,224,224
DFB  225,225,226,226,227,227,228,228,229,229
DFB  230,230,231,231,232,232,232,233,233,234
DFB  234,235,235,235,236,236,237,237,237,238
DFB  238,239,239,239,240,240,240,241,241,241
DFB  242,242,242,243,243,243,244,244,244,245
DFB  245,245,245,246,246,246,247,247,247,247
DFB  248,248,248,248,248,249,249,249,249,250
DFB  250,250,250,250,250,251,251,251,251,251
DFB  251,252,252,252,252,252,252,252,252,252
```

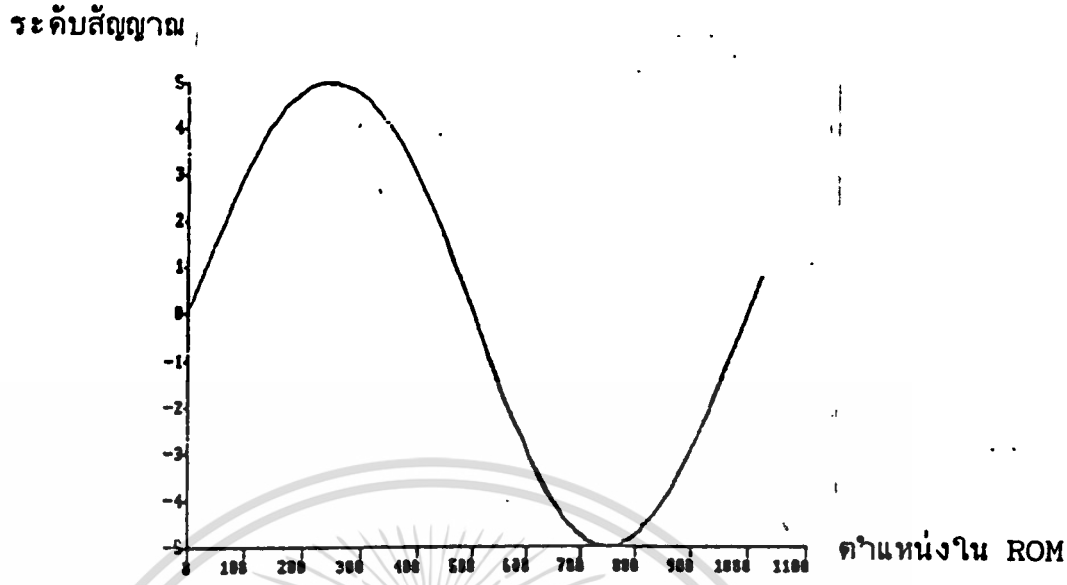
```
DFB  253,253,253,253,253,253,253,253,253,254
DFB  254,254,254,254,254,254,254,254,255,255
DFB  255,255,255,255,255,255,255,255,255,255
DFB  255,255,255,254,254,254,254,254,254,254
DFB  254,254,253,253,253,253,253,253,253,253
DFB  253,252,252,252,252,252,252,252,252,252
DFB  251,251,251,251,251,251,250,250,250,250
DFB  250,250,249,249,249,249,248,248,248,248
DFB  248,247,247,247,247,246,246,246,245,245
DFB  245,245,244,244,244,243,243,243,242,242
DFB  242,241,241,241,240,240,240,239,239,239
DFB  238,238,238,237,237,237,236,236,235,235
DFB  235,234,234,233,233,232,232,232,231,231
DFB  230,230,229,229,228,228,227,227,226,226
DFB  225,225,224,224,223,223,222,221,221,221
DFB  220,220,219,219,218,217,217,216,216,215
DFB  215,214,213,213,212,212,211,210,210,209
DFB  209,208,207,207,206,206,205,204,204,203
DFB  202,202,201,201,200,199,199,198,197,197
DFB  196,195,195,194,193,193,192,191,190,190
DFB  189,188,188,187,186,186,185,184,183,183
DFB  182,181,181,180,179,178,178,177,176,175
DFB  175,174,173,173,172,171,170,170,169,168
```

```
DFB  167,167,166,165,164,163,163,162,161,160
DFB  160,159,158,157,157,156,155,154,153,153
DFB  152,151,150,150,149,148,147,146,146,145
DFB  144,143,142,142,141,140,139,138,138,137
DFB  136,135,134,134,133,132,131,130,130,129
DFB  128,127,127,126,125,124,123,123,122,121
DFB  120,119,119,118,117,116,115,115,114,113
DFB  112,111,111,110,109,108,107,107,106,105
DFB  105,104,103,103,102,101,100,100,99,98
DFB  98,97,96,96,95,94,93,93,92,91
DFB  90,90,89,88,88,87,86,86,85,84
DFB  83,83,82,81,80,80,79,78,78,77
DFB  76,76,75,74,74,73,72,72,71,70
DFB  70,69,68,67,67,66,65,65,64,63
DFB  63,62,61,60,60,59,58,58,57,57
DFB  56,55,54,54,53,52,52,51,51,50
DFB  49,49,48,47,47,46,46,45,45,44
DFB  44,43,43,42,41,41,40,40,39,38
DFB  38,37,37,36,36,35,35,34,34,33
DFB  33,32,32,31,31,30,30,29,29,28
DFB  28,27,27,26,26,25,25,24,24,23
DFB  23,22,22,21,21,21,20,20,19,19
DFB  18,18,18,17,17,16,16,16,15,15
```

```
DFB  14,14,14,13,13,13,12,12,12,11
DFB  11,11,10,10,10,10,9,9,9,8
DFB  8,8,8,7,7,7,7,6,6,6
DFB  6,6,5,5,5,5,5,5,4,4
DFB  4,4,4,4,3,3,3,3,3,3
DFB  3,3,3,2,2,2,2,2,2,2
DFB  2,2,1,1,1,1,1,1,1,1
DFB  1,0,0,0,0,0,0,0,0,0
DFB  0,0,0,0,0,0,0,0,0,0
DFB  0,1,1,1,1,1,1,1,1,2
DFB  2,2,2,2,2,2,2,2,3,3
DFB  3,3,3,3,3,3,3,4,4,4
DFB  4,4,4,5,5,5,5,5,5,6
DFB  6,6,6,6,7,7,7,7,8,8
DFB  8,8,9,9,9,10,10,10,10,11
DFB  11,11,12,12,12,13,13,13,14,14
DFB  14,15,15,16,16,16,17,17,18,18
DFB  18,19,19,20,20,21,21,21,22,22
DFB  22,23,24,24,25,25,26,26,27,27
DFB  28,28,29,29,30,30,31,31,32,32
DFB  33,33,34,34,35,36,36,37,37,38
DFB  38,39,40,40,41,41,42,43,43,44
DFB  44,45,46,46,47,47,48,49,49,50
```

```
DFB  51,51,52,52,53,54,54,55,56,56
DFB  57,57,58,58,59,60,60,61,62,63
DFB  63,64,65,65,66,67,67,68,69,70
DFB  70,71,72,72,73,74,75,75,76,77
DFB  78,78,79,80,80,81,82,83,83,84
DFB  85,86,86,87,88,89,90,90,91,92
DFB  93,93,94,95,96,96,97,98,99,100
DFB  100,101,102,103,103,104,105,106,107,107
DFB  108,109,110,111,111,112,113,114,115,115
DFB  116,117,118,119,119,120,121,122,123,123
DFB  124,125,126,127
```

จากลักษณะข้อมูลที่กำหนดเป็นข้อมูลตัวเลขฐานสิบ โดยมีค่าจากยอดถึงยอด (Peak to Peak) ของสัญญาณรูปไซน์ 256 ค่า (0-255) จากนั้นนำข้อมูลที่ได้กำหนดไว้ทั้งหมดไปทำให้เป็นข้อมูลเลขฐาน 2 แล้วนำไปเก็บไว้ในหน่วยความจำแบบรอม โดยให้กำหนดตำแหน่งเริ่มต้น (Start Address) ที่ตำแหน่ง 0000 และมีตำแหน่งสุดท้าย (End Address) ที่ 03FF ซึ่งมีค่าเท่ากับตำแหน่งที่ 1024 ลักษณะของรูปไซน์ที่ได้จากการกำหนดค่า แสดงได้ดังรูป 3.4 จากลักษณะรูปไซน์ที่เก็บอยู่ในหน่วยความจำแบบรอมมีความละเอียด 1024 จุด และตำแหน่งสุดท้ายในการเก็บข้อมูลคือตำแหน่งที่ 1024 ดังนั้นที่ครึ่งคาบเวลาของสัญญาณรูปไซน์ที่เก็บค่าไว้ เป็นตำแหน่งที่ 512

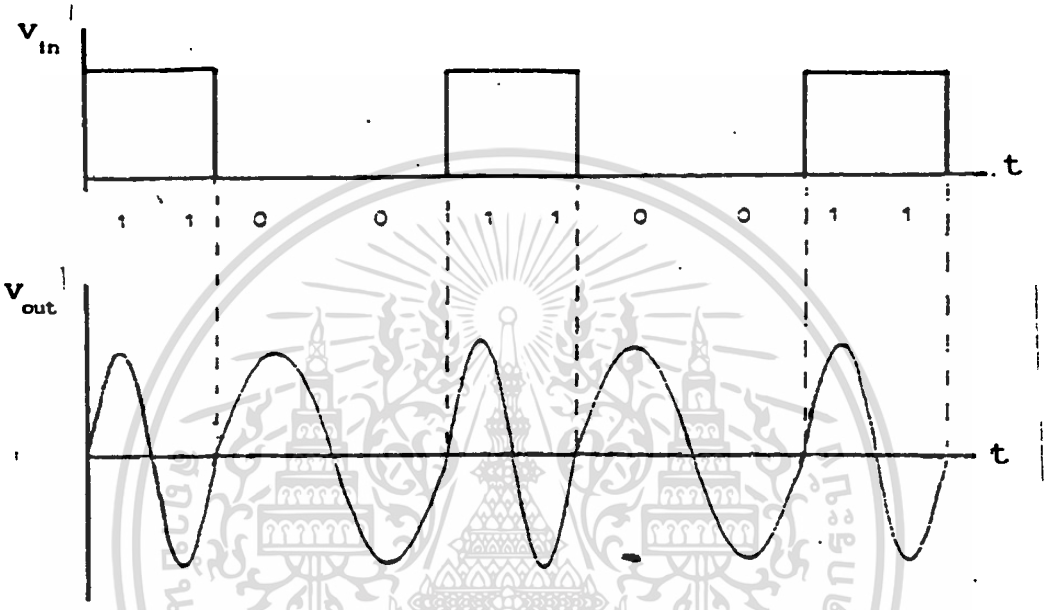


รูปที่ 3.4 รูปคลื่นสัญญาณไซน์ในวงจรรอม

จากสัญญาณรูปไซน์ที่ได้จากวงจรรอมหมายความว่า เมื่อมีการเปลี่ยนแปลงของข้อมูลไบนารี 1 บิต ซึ่งมีค่าสัญญาณไซน์ครึ่งคาบเวลา แล้วใช้สัญญาณครึ่งคาบเวลาที่ได้จากวงจรนับแบบไบนารีขนาด 12 บิต ที่ตำแหน่งจุดที่ 512 ไปป้อนกลับให้กับไอซี 74LS166 และ 74LS195 เพื่อทำการเลื่อนข้อมูลบิตต่อไปเข้ามาโดยการควบคุมนี้เป็นการควบคุมให้สัญญาณ เอฟ.เอส.เค. ใช้เพียงครึ่งคาบเวลาต่อการส่งข้อมูลไบนารี 1 บิตตลอดเวลา สัญญาณ เอฟ.เอส.เค. แบบครึ่งคาบเวลาที่ได้จากภาคมอดูเลเตอร์แสดงได้ดังรูปที่ 3.5

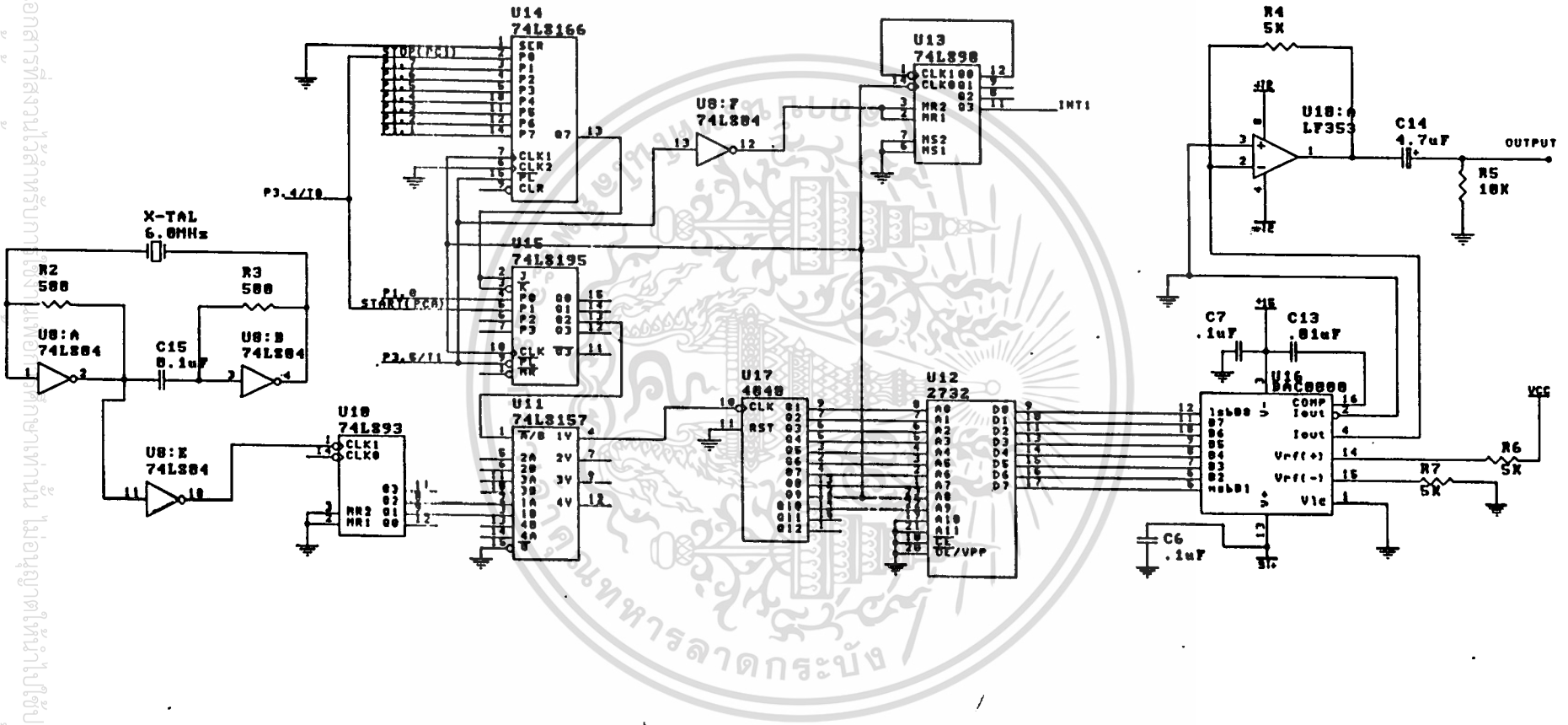
วงจรอีกส่วนหนึ่งที่นอกเหนือจากวงจรในภาคมอดูเลเตอร์ได้แก่วงจรควบคุม ซึ่งในการทดลองได้นำเอา ซีพียู(CPU) เบอร์ 8031 มาใช้ควบคุม วงจรควบคุมในส่วนนี้ จะทำการแปลงสัญญาณข้อมูลแบบอนุกรมที่ได้ถูกส่งมาจากเครื่องคอมพิวเตอร์ทางพอร์ท RS 232 ให้เป็นข้อมูลแบบขนานเพื่อป้อนให้กับส่วนอินพุทของภาคดีมอดูเลต สัญญาณ เอฟ.เอส.เค. และจะเป็นตัวส่งสัญญาณข้อมูล Start bit และStop

bit ออกมารวมกับสัญญาณจากเครื่องคอมพิวเตอร์ และจะส่งสัญญาณออกทางพอร์ท เพื่อควบคุมให้มีการส่งข้อมูลไบท์ต่อไปเข้ามาค้างค่าไว้ ในวงจรแปลงข้อมูลแบบขนานเป็นแบบอนุกรม เมื่อได้รับสัญญาณอินเตอร์รัพท์ และสัญญาณอินเตอร์รัพท์จะเกิดขึ้นเมื่อมีการส่งข้อมูลออกไปครบ 10 บิตแล้ว

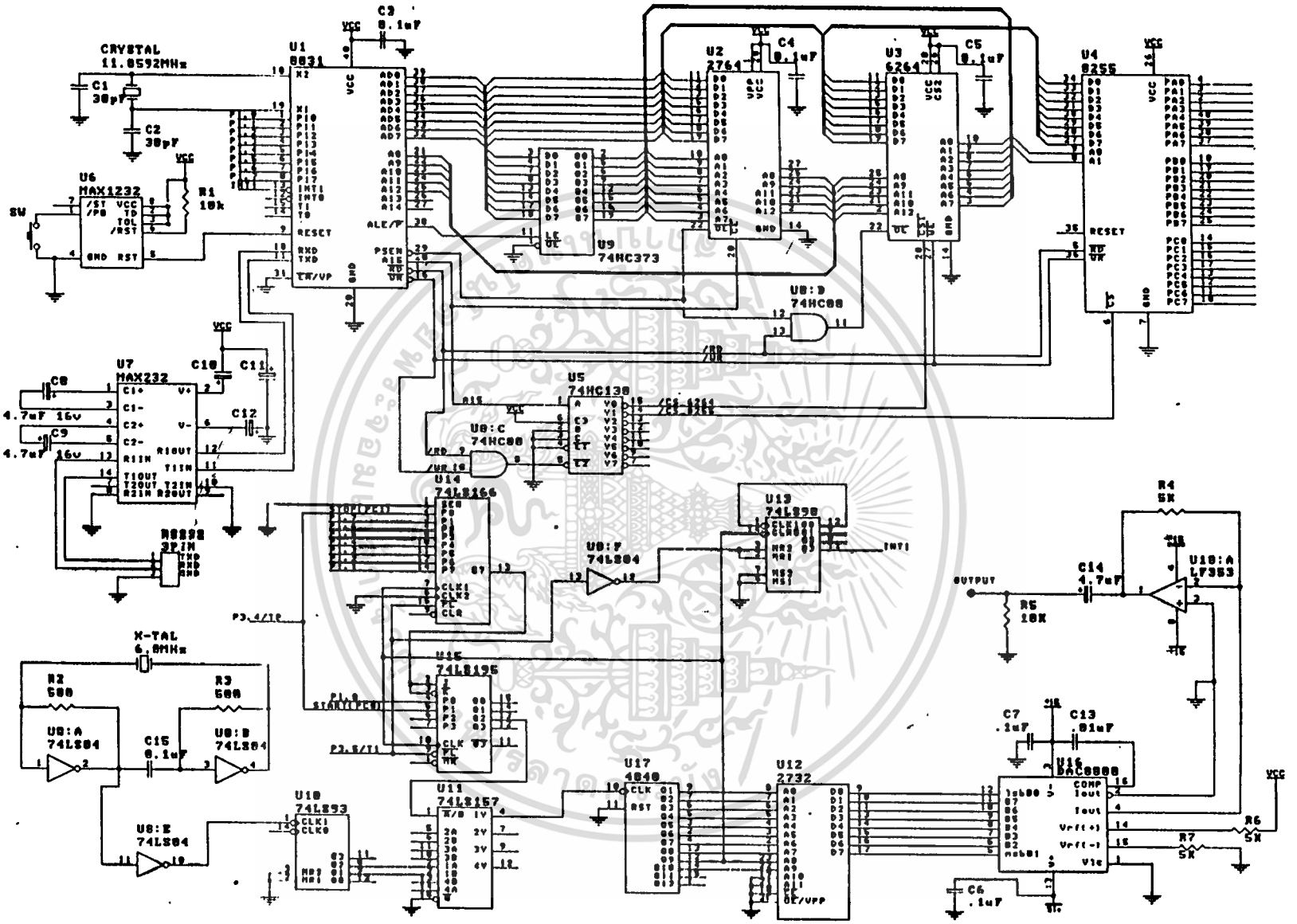


รูปที่ 3.5 สัญญาณ เอฟ.เอส.เค. แบบครึ่งคาบเวลา

ในการควบคุมให้วงจรควบคุมนี้ทำงานตามขั้นตอนต่าง ๆ นั้น จะอาศัยโปรแกรมมาใช้ควบคุมซึ่งมีรายละเอียดของโปรแกรมอยู่ในภาคผนวก ส่วนรายละเอียดของวงจรทั้งภาคมอดูเลเตอร์และวงจรควบคุมแสดงไว้ได้ดังรูปที่ 3.6 และ 3.7 ตามลำดับ



รูปที่ 3.6 วงจรภาคมอดูเลเตอร์



รูปที่ 3.7 วงจรภาคมอดูเลเตอร์และวงจรควบคุม

บทที่ 4
การออกแบบภาคดีมอดูเลต
สัญญาณ เอฟ. เอส. เค.

4.1 กล่าวนำ

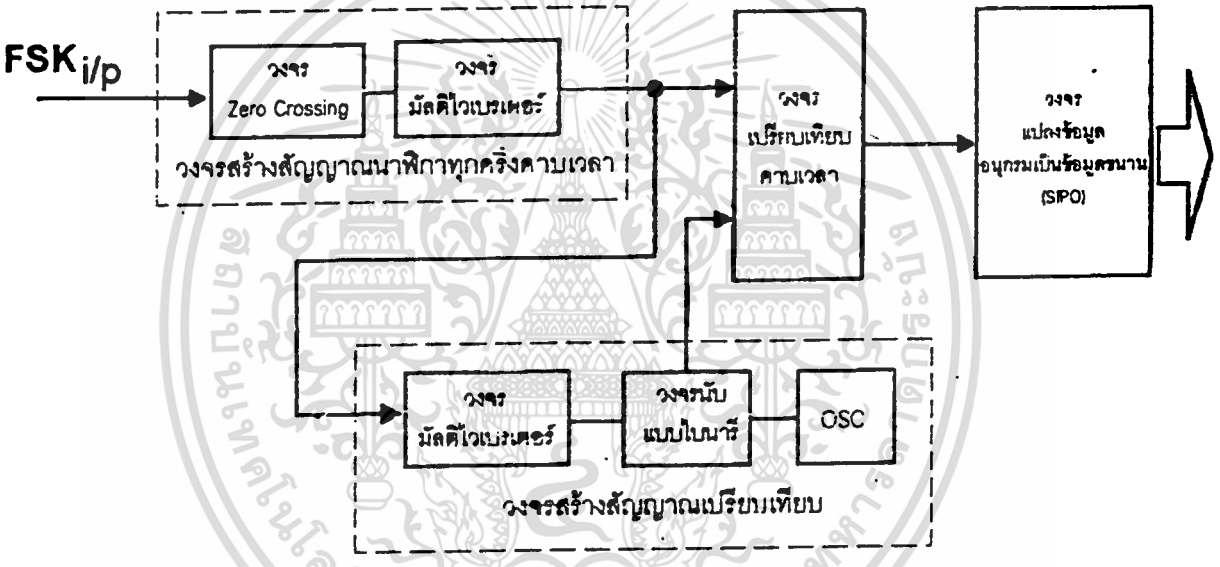
จากสัญญาณ เอฟ.เอส.เค. ที่ได้จากวงจรภาคมอดูเลเตอร์เป็นแบบสัญญาณครึ่งคาบเวลาต่อการส่งข้อมูลไบนารี 1 บิต ซึ่งช่วงเวลาในครึ่งคาบเวลาของความถี่จะมีค่าไม่เท่ากันเมื่อข้อมูลไบนารีมีสภาวะต่างกัน ขั้นตอนในการดีมอดูเลตชันเพื่อให้ได้ข้อมูลดั้งเดิมที่ปลายทางภาครับนั้นเป็นขั้นตอนที่มีความสำคัญมาก และจากแนวทางที่ได้กล่าวไว้ใน บทที่ 2 ได้กล่าวถึง การออกแบบวงจรดีมอดูเลตชัน วิธีที่ใช้อยู่ในปัจจุบันเป็นการเปรียบเทียบขนาดของสัญญาณที่เวลาคงที่ แต่ในบทนี้กล่าวถึงวิธีการเปรียบเทียบค่าเวลาในครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. ที่ได้ กับสัญญาณเปรียบเทียบที่มีค่าคงที่ค่าหนึ่งซึ่งเป็นส่วนสำคัญที่จะทำให้การแยกข้อมูลในการรับว่าเป็นลอจิกสถาวะ 0 หรือ 1 ได้ถูกต้อง

4.2 ภาค เอฟ.เอส.เค. ดีมอดูเลเตอร์

วงจรในภาคนี้เป็นการเปลี่ยนสัญญาณ เอฟ.เอส.เค. แบบครึ่งคาบเวลาที่มีการเปลี่ยนแปลงความกว้างของครึ่งคาบเวลาของความถี่ที่เปลี่ยนไป ตามข้อมูลของไบนารี เมื่อสภาวะเปลี่ยนไปให้กลับคืนเป็นข้อมูลไบนารีดั้งเดิม ซึ่งการทำงานของภาคดีมอดูเลเตอร์ แสดงได้ดังรูปที่ 4.1

จากบล็อกไดอะแกรมการทำงานของภาคดีมอดูเลเตอร์ในรูปที่ 4.1 จะเห็นว่าเมื่อสัญญาณ เอฟ.เอส.เค. เข้ามาที่อินพุตของวงจรสร้างสัญญาณนาฬิกา จะได้สัญญาณนาฬิกาออกมาทุกครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. เสมือนเกิดพัลส์ขึ้นทุกครั้งที่ส่งข้อมูลออกไป 1 บิต ซึ่งสัญญาณ ที่ได้ก็จะถูกแยกไปเข้ายังวงจรเปรียบ

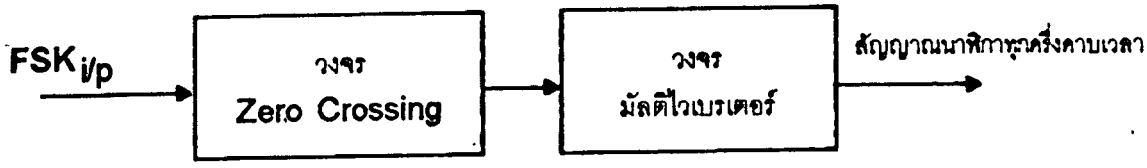
เทียบคาบเวลาเพื่อค้างค่า (Latch) และวงจรสร้างสัญญาณเปรียบเทียบ โดยสัญญาณเอาท์พุตของวงจรสร้างสัญญาณเปรียบเทียบ จะไปเข้ายังวงจรเปรียบเทียบคาบเวลาเพื่อค้างค่าและได้สัญญาณที่ออกมาเป็น ข้อมูลไบนารีแบบอนุกรมที่มีค่าเวลาของลอจิก 0 และ 1 ไม่เท่ากัน ข้อมูลส่วนนี้ไม่สามารถนำไปอ่านได้โดยตรง จึงนำสัญญาณนี้ไปเข้ายังวงจรแปลงข้อมูลอนุกรมเป็นขนาน(Serial In Parallel Out) เพื่อให้ได้ข้อมูลดิจิตอลที่ถูกต้อง ตรงตามที่ส่งมาจากภาคมอดูเลเตอร์



รูปที่ 4.1 แผนภาพของภาคดีมอดูเลเตอร์

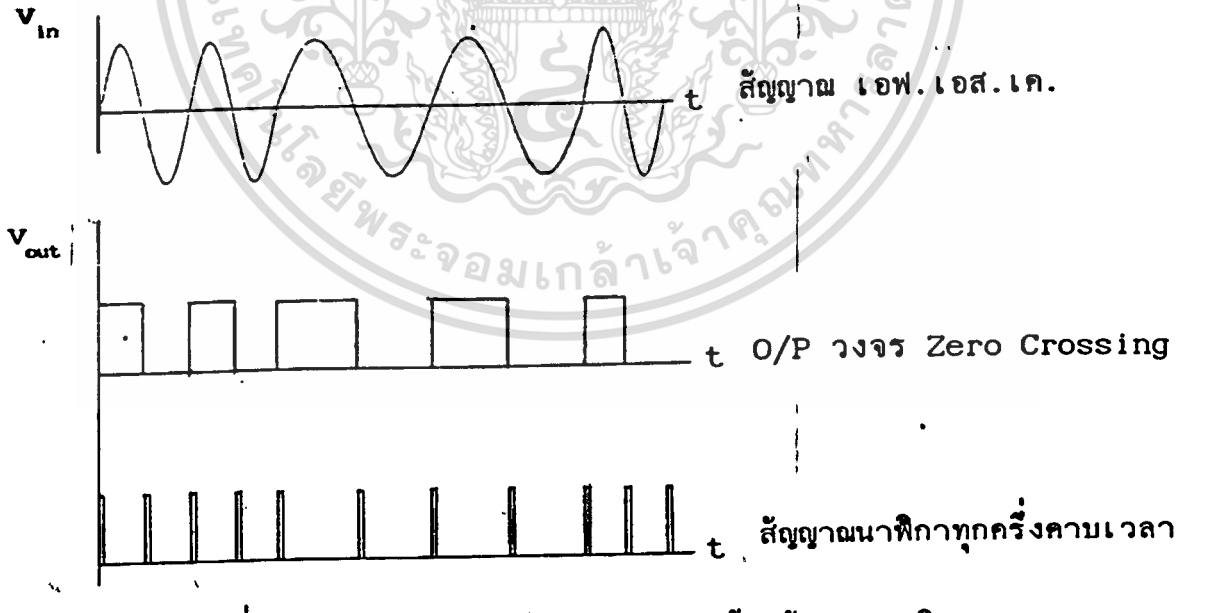
4.2.1 หลักการทำงานของวงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลา

จากวงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลาในรูปที่ 4.1 หรือเป็นวงจรสร้างสัญญาณนาฬิกาให้กับวงจรค้างค่าและวงจรสร้างสัญญาณเปรียบเทียบจะเห็นว่าวงจรย่อย ๆ ประกอบอยู่ดังแสดงในรูปที่ 4.2 ได้แก่วงจร Zero Crossing และวงจรมัลติไวเบรเตอร์



รูปที่ 4.2 แผนภาพวงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลา

เมื่อสัญญาณ เอฟ.เอส.เค. เข้ามายังวงจร Zero Crossing ทำให้เกิดสัญญาณรูปคลื่นสี่เหลี่ยมตามสัญญาณในซีกบวก แล้วผ่านไปยังวงจรมัลติไวเบรเตอร์ เพื่อให้เกิดสัญญาณพัลส์ขึ้นทุก ๆ จุดขาขึ้นและขาลงของพัลส์สี่เหลี่ยม ที่ได้มาจากวงจร Zero Crossing เป็นสัญญาณนาฬิกาให้กับวงจรเปรียบเทียบคาบเวลาซึ่งใช้วงจร D Flip-Flop เพื่อค้างค่าสัญญาณอินพุท และวงจรสร้างสัญญาณเปรียบเทียบ จากสัญญาณ เอฟ.เอส.เค. ที่ครึ่งคาบเวลาของความถี่ต่างกัน เวลาที่เกิดพัลส์ของสัญญาณนาฬิกาที่จุดต่าง ๆ มีค่าไม่เท่ากันตลอด ดังแสดงไว้ในรูปที่ 4.3



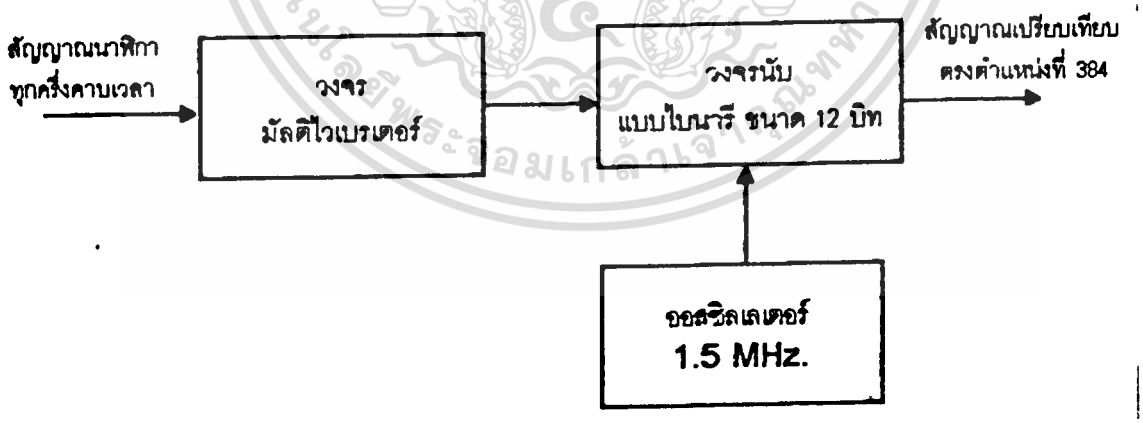
รูปที่ 4.3 รูปสัญญาณในส่วนของวงจรสร้างสัญญาณนาฬิกา

ดังนั้น สมมติถ้านำสัญญาณความถี่รูปไซน์ที่ความถี่ 1.5 กิโลเฮิรตซ์ และ 3 กิโลเฮิรตซ์เวลาในครึ่งคาบเวลาของความถี่จะมีค่าเวลาเท่ากับ 0.000333วินาที และ 0.000167 วินาทีตามลำดับ ซึ่งเห็นว่าสัญญาณพัลส์ที่เกิดขึ้นเมื่อส่งความถี่ 1.5 กิโลเฮิรตซ์และ 3 กิโลเฮิรตซ์ เข้ามาจะห่างกันเท่ากับ 0.000333 วินาที และ 0.000167 วินาทีตามลำดับ

สัญญาณเอาท์พุตที่ได้จาก วงจรสร้างสัญญาณนาฬิกานี้จะเข้าไปยังวงจรเปรียบเทียบคาบเวลาและวงจรสร้างเปรียบเทียบ และเป็นสัญญาณนาฬิกาที่มีสัญญาณเกิดขึ้นตามสัญญาณ เอฟ.เอส.เค. ที่รับเข้ามา

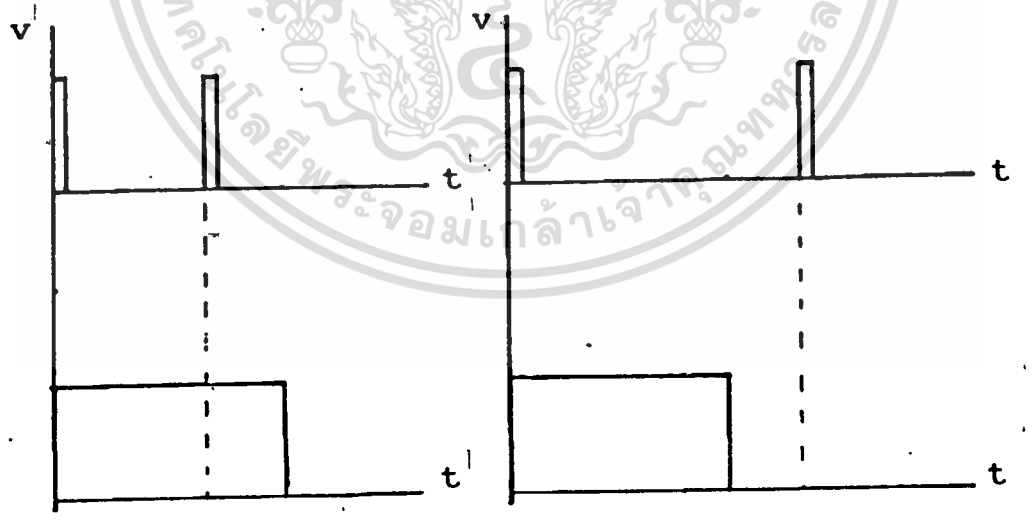
4.2.2 หลักการทำงานของวงจรสร้างสัญญาณเปรียบเทียบ

วงจรสร้างสัญญาณเปรียบเทียบนี้มีการทำงานดังรูปที่ 4.4 ซึ่งประกอบด้วยวงจรย่อย ๆ คือวงจร มัลติไวเบรเตอร์ วงจรนับแบบไบนารีขนาด 12 บิตและวงจรออสซิลเลเตอร์



รูปที่ 4.4 แผนภาพของวงจรสร้างสัญญาณเปรียบเทียบ

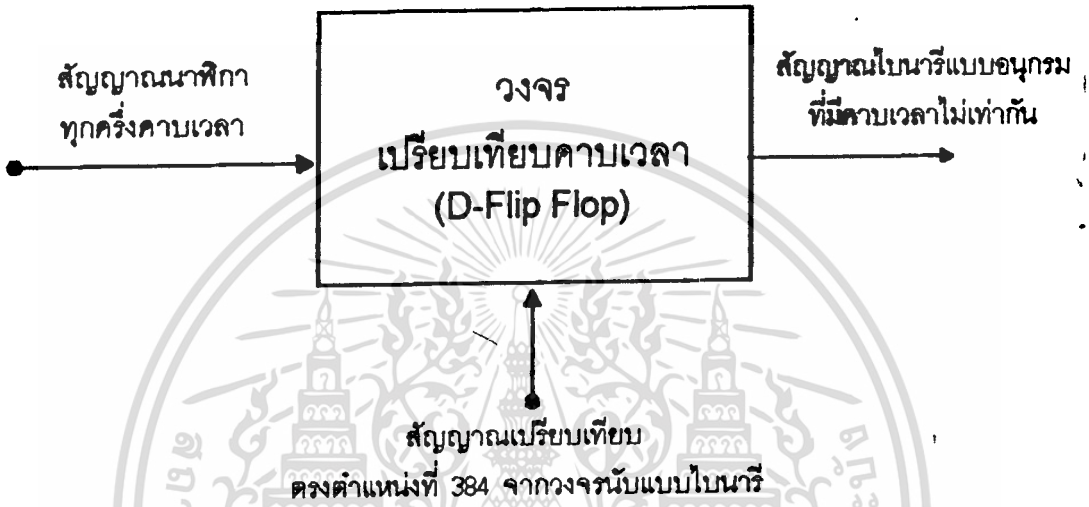
จากบล็อกไดอะแกรมจะเห็นว่าเมื่อสัญญาณพัลส์ ที่ได้จากวงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลาเข้ามายังวงจรมัลติไวเบรเตอร์ เพื่อต้องการสร้างสัญญาณพัลส์ที่จะใช้เป็นสัญญาณรีเซท(Reset) ให้กับวงจรนับแบบไบนารีขนาด 12 บิต โดยให้ทำการรีเซทหลังจากเมื่อสัญญาณนาฬิกาเกิดขึ้นแล้ว เพื่อป้องกันการอ่านข้อมูลผิดพลาดในวงจรค้างค่า ดังนั้น ถ้าให้สัญญาณรูปไซน์ใน 1 คาบเวลา แบ่งออกได้เป็น 1024 จุด ที่กล่าวมาแล้วในภาคมอดูเลเตอร์ สัญญาณที่ได้จากวงจรนับแบบไบนารีในภาคดีมอดูเลเตอร์นี้ให้นับจุดตำแหน่งที่ 384 ของความถี่ 1.5 กิโลเฮิรตซ์ ซึ่งสัญญาณนี้จะส่งไปป้อนให้กับวงจรค้างค่าต่อไป สาเหตุที่ให้วงจรนับแบบไบนารีทำการนับตำแหน่งนี้ เพราะว่าครึ่งคาบเวลาของความถี่ 3.0 กิโลเฮิรตซ์ มีช่วงเวลาเท่ากับ 1/4 เท่าของความถี่ 1.5 กิโลเฮิตซ์หรือตรงจุดตำแหน่งที่ 256 ของความถี่ 1.5 กิโลเฮิรตซ์นั่นเอง ดังนั้น เมื่อกำหนดให้วงจรนับแบบไบนารีนับที่ตำแหน่ง 384 เพื่อป้องกันไม่ให้วงจรค้างค่าอ่านข้อมูลออกมาผิดพลาด สัญญาณเอาท์พุตของวงจรนับแบบไบนารีขนาด 12 บิตที่ไปเข้ายังวงจรค้างค่ามีรูปร่างดังที่แสดงไว้ในรูปที่ 4.5



รูปที่ 4.5 รูปของสัญญาณเอาท์พุทของวงจรนับแบบไบนารีเทียบกับสัญญาณรีเซท

4.2.3 หลักการทำงานของวงจรเปรียบเทียบคาบเวลา

วงจรเปรียบเทียบคาบเวลา มีลักษณะดังรูปที่ 4.6 ซึ่งใช้วงจร D-Flip Flop เข้ามาทำการค้างค่า(Latch) ข้อมูลของสัญญาณที่รับเข้ามา

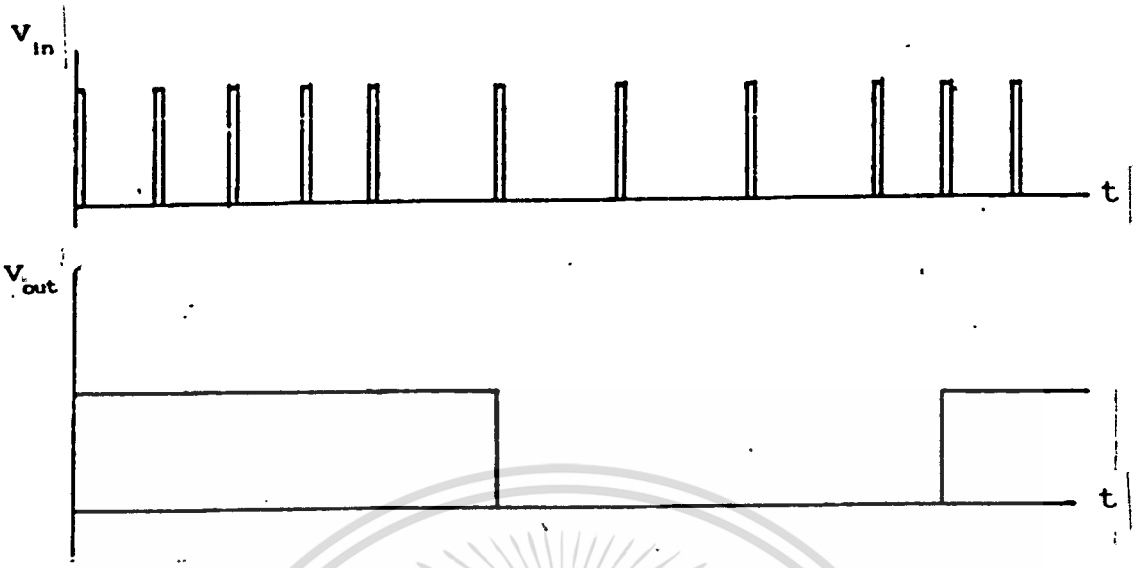


รูปที่ 4.6 แผนภาพของวงจรเปรียบเทียบคาบเวลา

เมื่อสัญญาณนาฬิกาที่มีคาบเวลาไม่เท่ากัน ที่ได้จากวงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลา เข้ามาเป็นสัญญาณนาฬิกาให้กับวงจรเปรียบเทียบคาบเวลา เพื่อค้างข้อมูลของสัญญาณที่ขาอินพุท ซึ่งได้มาจากเอาท์พุทของวงจรแบบไบนารีขนาด 12 บิต ตรงตำแหน่งสัญญาณทริกนั่นเอง

ดังนั้น สัญญาณเอาท์พุทของวงจรค้างค่านี้ จะมีรูปสัญญาณออกมาเป็นสัญญาณไบนารีที่มีคาบเวลาของบิต 0 และ 1 ไม่เท่ากันแสดงได้ดังรูปที่ 4.7

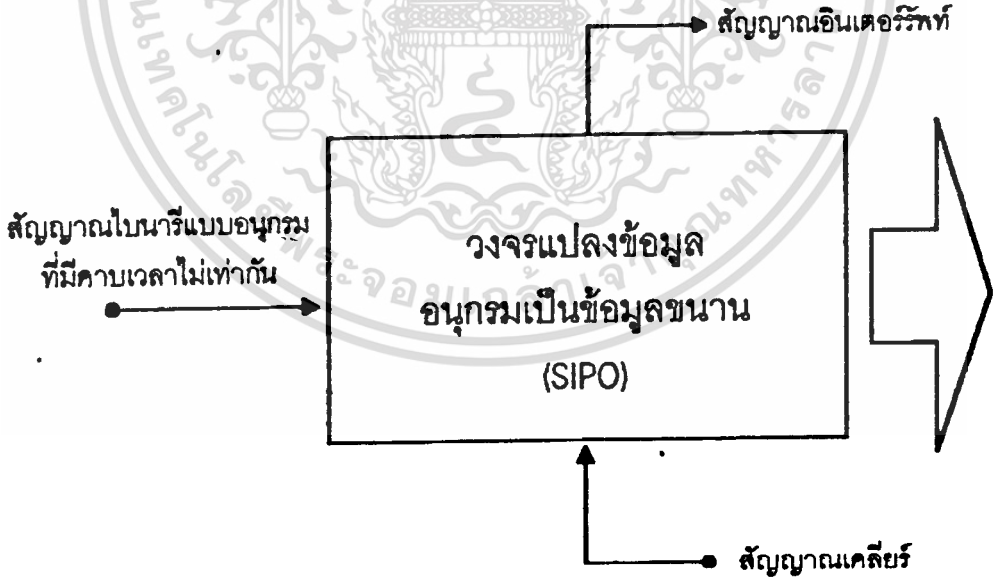
สัญญาณเอาท์พุตที่ได้จากวงจรค้างค่าถูกส่งต่อไปยังวงจรแปลงข้อมูลแบบอนุกรมเป็นข้อมูลขนานต่อไปเพื่อจัดข้อมูลที่ได้ให้ถูกต้องออกมา



รูปที่ 4.7 รูปของสัญญาณในส่วนของวงจรเปรียบเทียบคาบเวลา

4.2.4 หลักการทำงานของวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน(SIPO)

วงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน มีลักษณะดังรูป 4.8



รูปที่ 4.8 วงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน

จากวงจรจะเห็นว่าสัญญาณที่เข้ามาที่อินพุทของวงจร จะเป็นข้อมูลแบบไบนารีที่มีค่าเวลาของบิตไม่เท่ากัน ดังนั้น เมื่อต้องการข้อมูลที่ถูกต้องโดยไม่คำนึงถึงค่าเวลาจะต้องนำข้อมูลไบนารีนั้น ไปแปลงเป็นข้อมูลขนาน จากวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนานเมื่อมีการส่งสัญญาณข้อมูลครบ 1 ไบท์หรือมีการส่งบิต Start และบิต Stop แล้วก็จะมีสัญญาณอินเตอร์รัพท์ออกมา 1 พัลส์ เพื่อส่งต่อไปยังชุดวงจรควบคุม และวงจรควบคุมจะทำการส่งสัญญาณพัลส์ กลับมาทำการเคลียให้กับวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนานเพื่อให้ส่งข้อมูลขนานไบท์ต่อไป

4.3 วงจรทดลองภาคดีมอดูเลเตอร์

วงจรภาคดีมอดูเลเตอร์ ประกอบด้วยวงจรหลัก ๆ 4 วงจร คือ วงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลา วงจรสร้างสัญญาณเปรียบเทียบ วงจรเปรียบเทียบคาบเวลา และวงจรแปลงข้อมูลแบบอนุกรมเป็นข้อมูลแบบขนาน (SIPO) ซึ่งในการทดลองโครงการนี้มีวงจรประกอบดังรูปที่ 4.9

จากรูป 4.9 ที่จะขอกล่าวถึงรายละเอียดก่อนอื่นคือ วงจรสร้างสัญญาณนาฬิกาทุกครึ่งคาบเวลา ซึ่งเป็นวงจรที่จะสร้างสัญญาณนาฬิกาให้เกิดขึ้นทุกครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. ที่รับเข้ามาให้กับวงจรเปรียบเทียบคาบเวลา และเป็นสัญญาณพัลส์ในการรีเซทให้กับวงจรสร้างสัญญาณเปรียบเทียบ จะเห็นว่าเมื่อสัญญาณ เอฟ.เอส.เค. ที่รับเข้ามา เป็นสัญญาณรูปไซน์ที่ทุกครึ่งคาบเวลาจะมีค่าเปลี่ยนแปลงตลอด เมื่อมีการส่งข้อมูลดิจิตอลแบบไบนารีที่มีสภาวะของลอจิกเปลี่ยนไปและจะมีค่าเท่ากันเมื่อข้อมูลของสัญญาณไบนารีที่มีลอจิกสภาวะเดียวกัน จากภาคมอดูเลเตอร์เพราะว่าในการส่งสัญญาณ เอฟ.เอส.เค. จากภาคมอดูเลเตอร์จะใช้ความถี่สูงเมื่อสภาวะลอจิกเป็น 1 และใช้ความถี่ต่ำเมื่อสภาวะลอจิกเป็น 0 สัญญาณ เอฟ.เอส.เค. ที่รับเข้ามาที่ภาคดีมอดูเลเตอร์จะเข้าไปยังวงจร Zero Crossing

ซึ่งใช้ไอซีเบอร์ LM 311 ที่เป็นไอซีออปแอมป์(Op-amp) ทำหน้าที่ตั้งค่าระดับแรงดันเปรียบเทียบกับกราวด์หรือ 0 โวลท์ ดังนั้น สัญญาณ เอฟ.เอส.เค. รูปไซน์ที่เข้ามาส่วนที่มีระดับแรงดันสูงกว่า 0 โวลท์จะทำให้เอาท์พุทของวงจรมีค่าเป็นสัญญาณลอจิกที่มีสภาวะเป็น 1 และส่วนที่สัญญาณรูปไซน์ต่ำกว่า 0 โวลท์ ก็จะได้ลอจิกที่มีสภาวะเป็น 0 ดังรูปที่ 4.3 เอาท์พุตที่เป็นสัญญาณลอจิกจาก วงจร Zero Crossing ถูกส่งไปยังวงจรมัลติไวเบรเตอร์ซึ่งใช้ ไอซีเบอร์ 74LS123 สัญญาณเอาท์พุทของวงจรมัลติไวเบรเตอร์เป็นรูปสัญญาณพัลส์ที่เกิดขึ้นทุก ๆ ขอบขาขึ้นและขาขอบลงของสัญญาณที่ได้จากวงจร Zero Crossing หรือพูดได้ว่าจะเกิดสัญญาณพัลส์ขึ้นทุก ๆ ครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. ที่เข้ามานั่นเอง

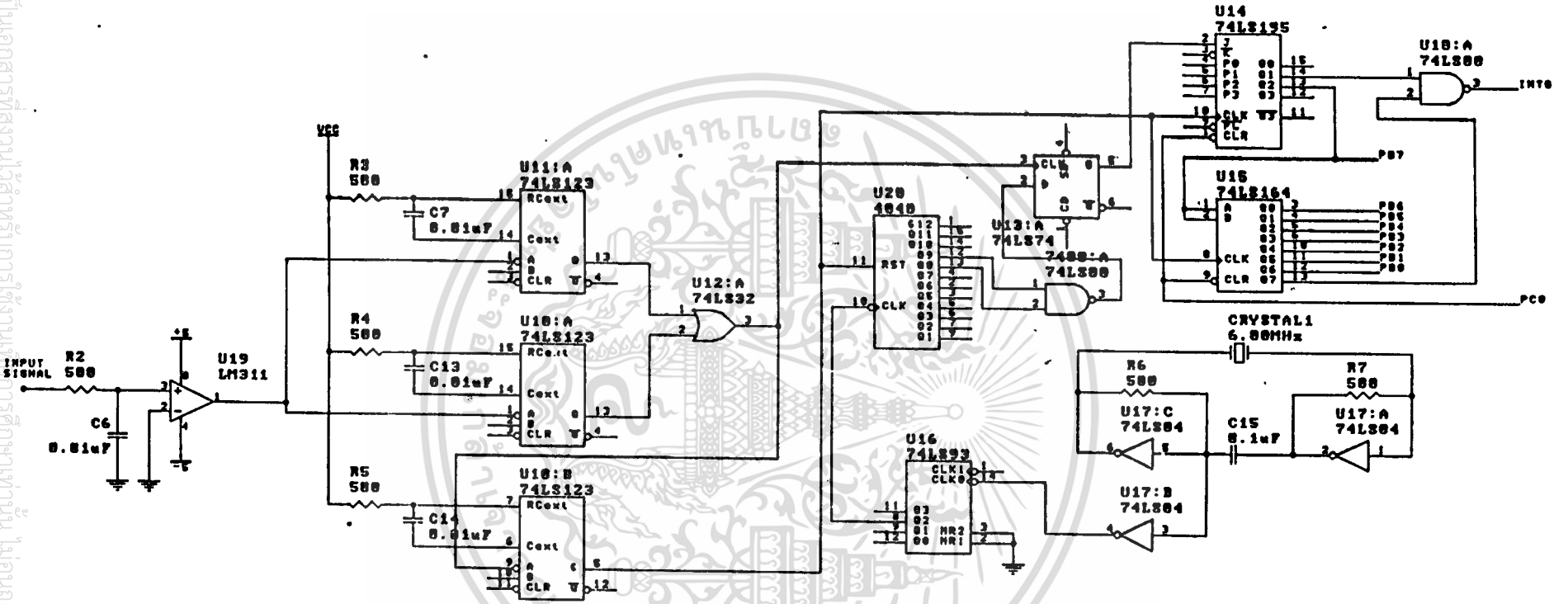
สัญญาณที่ได้จากวงจรมัลติไวเบรเตอร์แยกเป็น 2 ทาง คือ จะเข้าไปยังวงจรเปรียบเทียบคาบเวลาเพื่อค้างค่าซึ่งในการทดลองใช้ ไอซีเบอร์ 74LS74 เป็นไอซีประเภท D-Flip Flop โดยเป็นสัญญาณนาฬิกาให้กับ 74LS74 และไปเข้ายังวงจรมัลติไวเบรเตอร์ในส่วนของวงจรสร้างสัญญาณเปรียบเทียบ ใช้ไอซีเบอร์ 74LS123 เช่นกัน เพื่อใช้สร้างสัญญาณรีเซทให้กับวงจรนับแบบไบนารีขนาด 12 บิต ใช้ไอซีเบอร์ 4040 โดยมีสัญญาณนาฬิกาจากวงจรออสซิลเลเตอร์ 6 เมกกะเฮิรตซ์ มาผ่านวงจรหาร 4 ดังนั้น สัญญาณนาฬิกาจะมีความถี่ 1.5 เมกกะเฮิรตซ์ วงจรนับแบบไบนารีจะทำการรีเซททุกครั้งเมื่อสัญญาณเปรียบเทียบ ที่ได้จากวงจรสัญญาณเปรียบเทียบสิ้นสุดไปแล้วทุก ๆ ครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค.

วงจรนับแบบไบนารีขนาด 12 บิต ถูกจัดให้นับที่ตำแหน่ง 384 ของสัญญาณรูปไซน์ที่มีความถี่เท่ากับ 1500000/1024 เฮิรตซ์ หรือประมาณ 1465 เฮิรตซ์ ซึ่งเท่ากับทางภาคมอดูเลเตอร์ เพื่อป้องกันความผิดพลาดในการอ่านข้อมูล ในวงจรเปรียบเทียบคาบเวลา สัญญาณเอาท์พุทที่ได้จากวงจรไบนารีนี้ไปเข้ายังวงจรเปรียบเทียบคาบเวลาเพื่อค้างค่าที่ขาสัญญาณอินพุทซึ่งเป็นสัญญาณที่ออกมาจากวงจรนับ

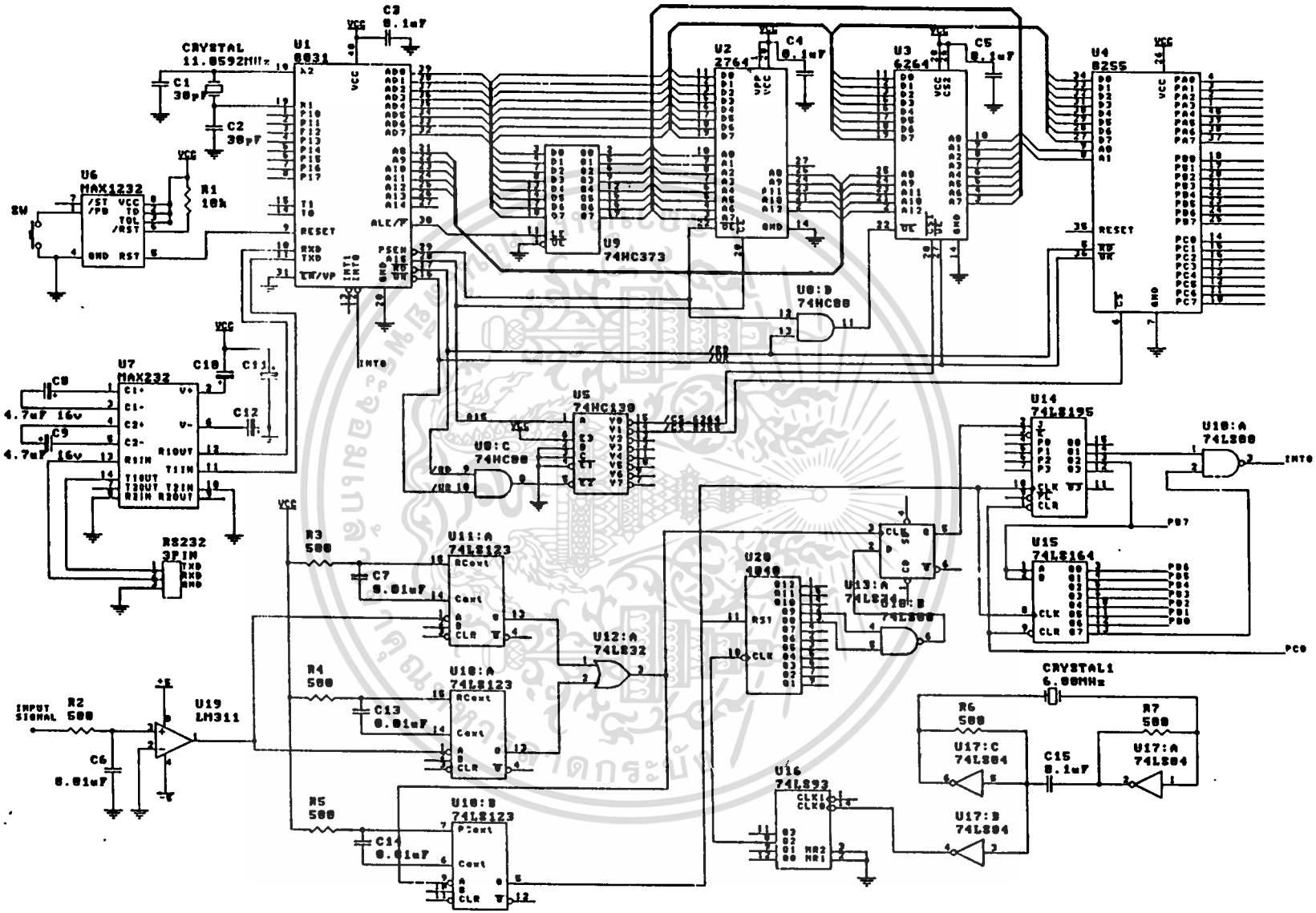
แบบไบนารีตรงที่นำพิกาเข้ามา สัญญาณข้อมูลที่ได้จากวงจรเปรียบเทียบคาบเวลานี้เป็นรูปของข้อมูลดิจิตลแบบไบนารีที่มีค่าเวลาของลอจิก 0 และ 1 ไม่เท่ากัน จึงไม่สามารถที่จะนำข้อมูลนี้ไปอ่านโดยตรง ดังนั้นข้อมูลดิจิตอลแบบไบนารีที่มีค่าเวลาของลอจิก 0 และ 1 ไม่เท่ากันจะถูกส่งต่อไปยัง วงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน(SIPO) ต่อไป ซึ่งใช้ไอซีเบอร์ 74LS164 และ 74LS195 เพื่อให้ได้ข้อมูลของสัญญาณไบนารีที่ถูกต้องโดยเป็นข้อมูลขนาน สัญญาณที่ได้ออกมาส่วนนี้จะมีข้อมูลบิต Start และบิต Stop ออกมาและนำเอาสัญญาณทั้งสองมาทำการ NAND กันแล้วเป็นสัญญาณอินเตอร์รัพท์กับชุดวงจรควบคุม ให้ทำการส่งสัญญาณมาเคลียให้กับวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน ให้ทำการรับ-ส่งข้อมูลไบท์ต่อไป

ส่วนของวงจรควบคุม ที่นำมาใช้เชื่อมโยงระหว่างชุดดีมอดูเลเตอร์ กับคอมพิวเตอร์นั้นได้ใช้ไมโครโปรเซสเซอร์ CPU 8031 เป็นตัวหลักในการมาทำการควบคุมเปลี่ยนข้อมูลขนานที่ได้จากภาคดีมอดูเลเตอร์ เป็นข้อมูลดิจิตอลแบบอนุกรมที่สามารถติดต่อกับคอมพิวเตอร์ โดยส่งผ่านทาง พอร์ท RS232 และทำงานร่วมกับโปรแกรมที่ใช้ในการควบคุมเพื่อให้แสดงผลข้อมูลที่รับเข้ามาบนจอเครื่องคอมพิวเตอร์ ซึ่งการทำงานจะกลับกับการทำงานในส่วนของภาคมอดูเลเตอร์

ดังนั้น วงจรควบคุมที่ใช้ในการทดลอง เมื่อนำมาประกอบร่วมกับภาคดีมอดูเลเตอร์สามารถแสดงได้ดังรูปที่ 4.10



รูปที่ 4.9 วงจรภาคดีมอดูเลเตอร์



รูปที่ 4.10 วงจรภาคดีมอดูเลเตอร์และวงจรควบคุม

บทที่ 5
ผลการทดลอง

5.1 กล่าวนำ

จากการทดลองการทำงานของวงจรที่ได้ประกอบซึ่งเป็นการส่งข้อมูลด้วยวิธี เอฟ.เอส.เค. แบบครึ่งคาบเวลา ซึ่งเป็นการส่งสัญญาณความถี่ครึ่งคาบเวลาต่อข้อมูลไบนารี 1 บิต การทดลองนี้ได้ทำการต่อคอมพิวเตอร์เข้ากับวงจรควบคุมที่ใช้ ซีพียู 8031 ทำการรับข้อมูลไบนารีแบบอนุกรมจากเครื่องคอมพิวเตอร์ แล้วแปลงสัญญาณให้เป็นข้อมูลไบนารีแบบขนาน ส่งต่อไปยังวงจรอินพุทของภาคมอดูเลตสัญญาณ เอฟ.เอส.เค. แบบครึ่งคาบเวลา แล้วส่งข้อมูล เอฟ.เอส.เค. ไปตามสายเข้าไปยังภาคดีมอดูเลเตอร์ เมื่อรับสัญญาณเข้ามาแล้วก็ทำการดีมอดูเลตสัญญาณให้ได้สัญญาณเอาท์พุตเป็นข้อมูลไบนารีแบบขนาน และมีข้อมูลตรงตามข้อมูลที่ส่งมาจากภาคมอดูเลเตอร์สัญญาณจากภาคดีมอดูเลเตอร์นี้ถูกส่งไปยังวงจรควบคุมที่ใช้ ซีพียู 8031 ที่มีวงจริเหมือนกับทางภาคมอดูเลเตอร์ แต่ทำหน้าที่เปลี่ยนข้อมูลไบนารีแบบขนานเป็นข้อมูลไบนารีแบบอนุกรม แล้วส่งผลไปแสดงบนจอเครื่องคอมพิวเตอร์เพื่อแสดงว่ารับข้อมูลได้ถูกต้อง

5.2 ผลการทดลองภาคมอดูเลเตอร์

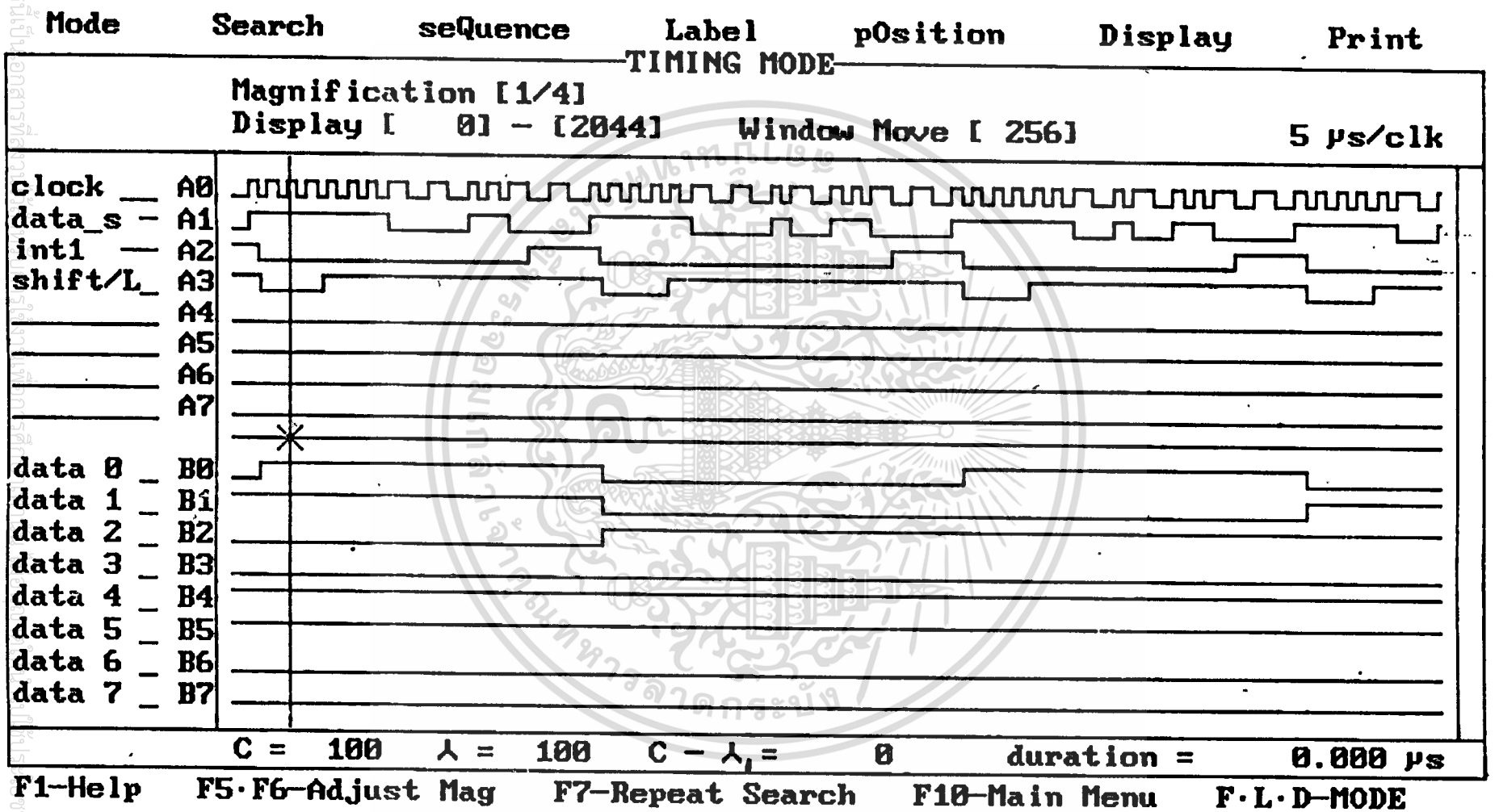
จากการทดลองส่งข้อมูลจากเครื่องคอมพิวเตอร์ ซึ่งเป็นสัญญาณข้อมูลไบนารีแบบอนุกรมให้กับวงจรควบคุมที่ใช้ ซีพียู เบอร์ 8031 และสัญญาณข้อมูลแบบขนานที่เอาท์พุตของวงจรควบคุมนี้จะเข้าไปยังภาคมอดูเลเตอร์ เพื่อทำการมอดูเลตสัญญาณให้เป็นสัญญาณ เอฟ.เอส.เค แบบครึ่งคาบเวลาต่อการส่งสัญญาณ 1 บิต

ผลของการทำการทดลองที่ได้ นำสัญญาณเข้าเครื่องลอจิกแอนาไลเซอร์ (Logic Analyzer) เพื่อทำการวัดรูปคลื่นสัญญาณข้อมูลจุดต่าง ๆ ซึ่งมีสัญญาณข้อมูล

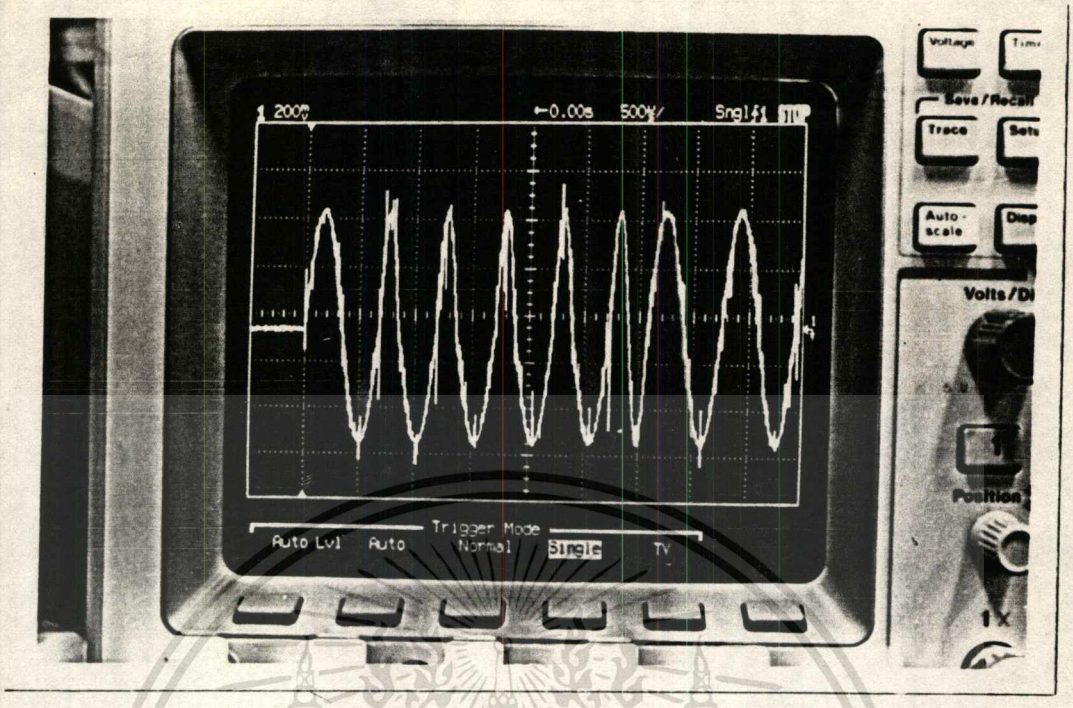
Data 0 ถึง Data 7 เป็นสัญญาณข้อมูลไบนารีแบบขนานจำนวน 8 บิต ที่มาจากวงจรควบคุมแล้วเข้าไปยังวงจรแปลงข้อมูลแบบขนานเป็นอนุกรมของภาคมอดูเลเตอร์ที่ใช้ไอซี 74LS166 และ 74LS195 สัญญาณ Clock เป็นสัญญาณนาฬิกาที่ได้จากวงจรในส่วนที่ 2 ซึ่งเป็นสัญญาณในการเลื่อนบิตต่อไปและไปเข้ายังวงจรนับสิบ เพื่อให้เกิดสัญญาณอินเตอร์รัพท์ขึ้นเมื่อมีการส่งข้อมูลครบ 10 บิต (ซึ่งเป็นสัญญาณข้อมูล 8 บิต และเป็นสัญญาณบิต Start แบบบิต Stop อีก 2 บิต)

จะเห็นว่า เป็นลักษณะสัญญาณนาฬิกาที่มีลักษณะกว้างที่ตำแหน่งลอจิกเป็น 0 และลักษณะแคบที่ตำแหน่งลอจิกเป็น 1 สัญญาณ Data S เป็นสัญญาณข้อมูลอินพุทที่แปลงเป็นสัญญาณข้อมูลอนุกรมแล้ว เห็นว่าช่วงเวลาของข้อมูลลอจิก 1 และลอจิก 0 จะมีค่าเวลาไม่เท่ากัน สัญญาณ Int 1 เป็นสัญญาณอินเตอร์ชิพที่เกิดขึ้นเมื่อมีการส่งข้อมูลครบ 10 บิตแล้ว ส่วนสัญญาณ Shift/L เป็นสัญญาณชิพโหลดที่จะเกิดขึ้นเมื่อมีสัญญาณอินเตอร์ชิพเกิดขึ้น ซึ่งจะเกิดสัญญาณขึ้นเมื่อข้อมูลเข้ามาครบ 10 บิต เพื่อไปควบคุมให้วงจรส่วนหน้ารับข้อมูลไบท์ต่อไปเข้ามา รูปร่างของคลื่นสัญญาณที่จับได้แสดงไว้ดังรูปที่ 5.1

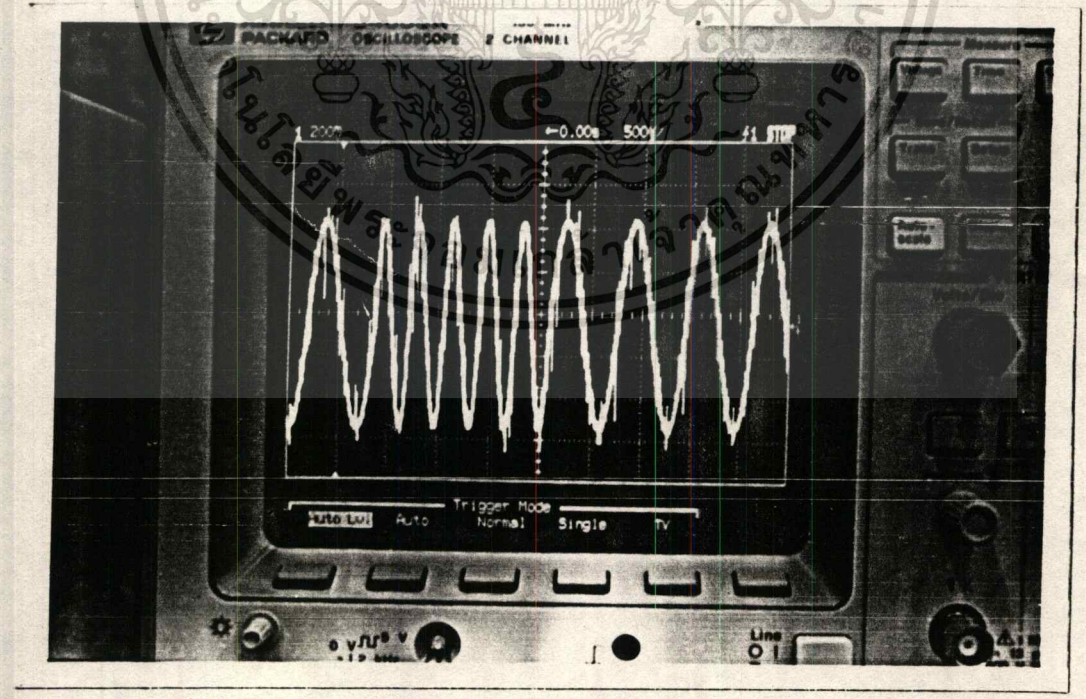
ส่วนรูปคลื่นสัญญาณ เอฟ.เอส.เค.แบบครึ่งคาบเวลาต่อการส่งข้อมูล 1 บิต นั้น ในการทดลองได้ทำการส่งข้อมูลไปที่ละไบท์แล้วใช้สโคปทำการจับรูปคลื่นสัญญาณ เอฟ.เอส.เค. ที่ได้จากภาคมอดูเลเตอร์ รูปคลื่นสัญญาณแสดงได้ดังรูปที่ 5.2



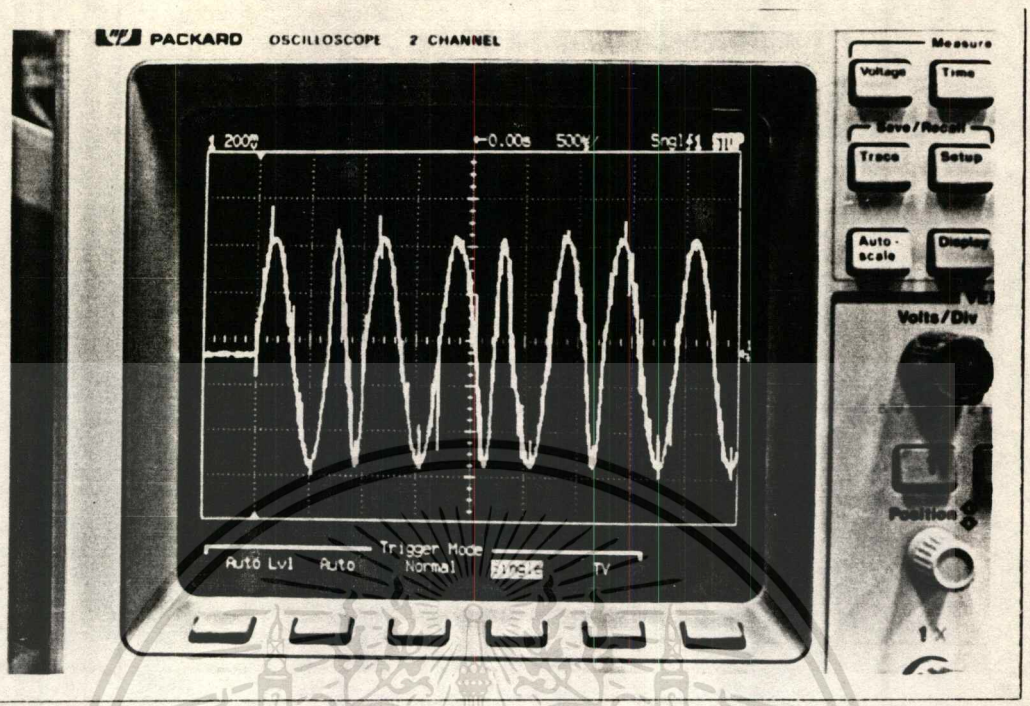
รูปที่ 5.1 สัญญาณในภาคมอดูเลเตอร์



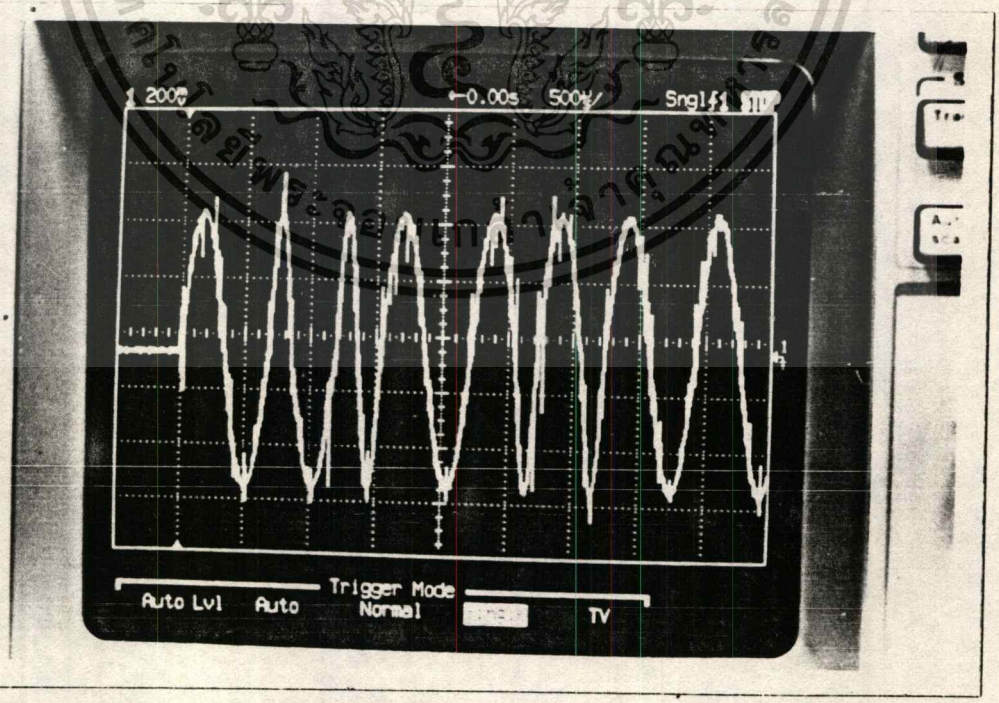
รูปที่ 5.2 a) สัญญาณข้อมูล AA



รูปที่ 5.2 b) สัญญาณข้อมูล FF



รูปที่ 5.2 c) สัญญาณข้อมูล 31

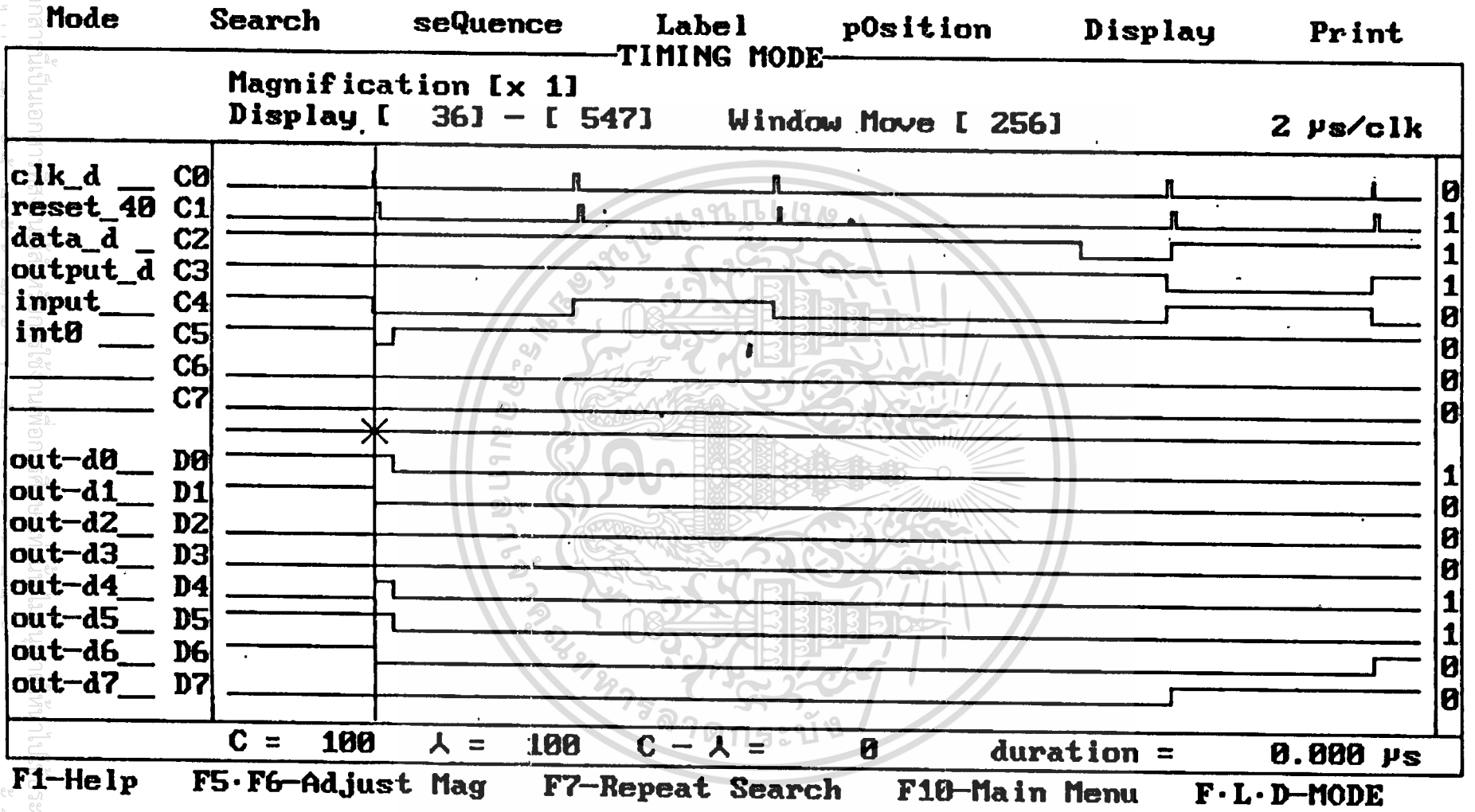


รูปที่ 5.2 d) สัญญาณข้อมูล 46

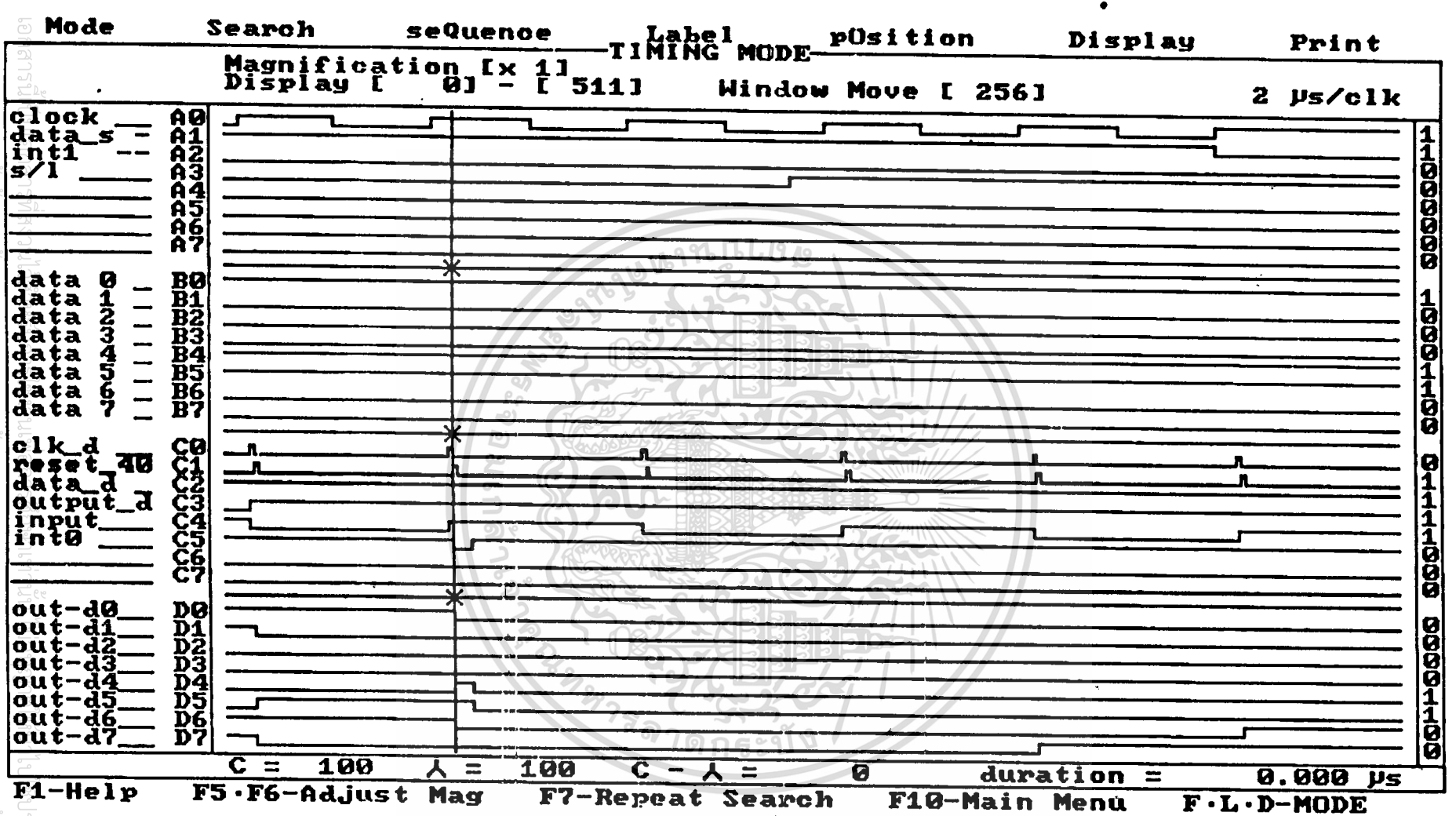
5.3 ผลการทดลองภาคดีมอดูเลเตอร์

จากการทดลองส่งข้อมูล เอฟ.เอส.เค. แบบครึ่งคาบเวลาต่อข้อมูล 1 บิต จากภาคมอดูเลเตอร์ไปตามสายแล้วใช้ภาคดีมอดูเลเตอร์รับที่ปลายทาง รูปแบบขนาดคลื่นสัญญาณต่าง ๆ ที่ได้จากเครื่องลอจิกแอนาไลเซอร์ มีดังนี้ clk d. เป็นสัญญาณนาฬิกาที่ได้จากการแปลงสัญญาณ เอฟ.เอส.เค. รูปไซน์มาเป็นรูปสี่เหลี่ยม และจะเกิดพัลส์ทุก ๆ ขอบขาขึ้นและขาลงของสัญญาณ หรือทุก ๆ ครึ่งคาบเวลาของสัญญาณ เอฟ.เอส.เค. ที่รับเข้ามา เป็นสัญญาณที่ได้จาก วงจรสร้างสัญญาณนาฬิกาในภาคดีมอดูเลเตอร์ สัญญาณ Reset 40 เป็นสัญญาณ Reset ให้กับวงจรนับแบบไบนารีขนาด 12 บิต ซึ่งจะเกิดขึ้นหลังจากเกิดพัลส์สัญญาณนาฬิกาทุก ๆ ลูกเพื่อป้องกันการอ่านข้อมูลในวงจรเปรียบเทียบคาบเวลาผิดพลาดไป สัญญาณ Data d เป็นสัญญาณที่ได้จากวงจรนับแบบไบนารีขนาด 12 บิต ซึ่งจะเห็นว่าครึ่งคาบเวลาของความถี่ต่ำในสัญญาณ เอฟ.เอส.เค. สัญญาณนี้ตรงกับสัญญาณนาฬิกาต่อไปจะมีค่าเป็น 0 ส่วนครึ่งคาบเวลาของความถี่สูงสัญญาณยังมีค่าเป็น 1 อยู่ สัญญาณ Output d เป็นสัญญาณข้อมูลไบนารีแบบอนุกรมที่มีค่าคาบเวลาของลอจิก 0 และ 1 ไม่เท่ากันตลอด สัญญาณ Input เป็นสัญญาณ เอฟ.เอส.เค. ที่ผ่านวงจร Zero Crossing ออกมา สัญญาณ Int0 เป็นสัญญาณอินเตอร์รัพท์ให้กับวงจรควบคุมเพื่อให้ทราบว่าข้อมูลที่เข้ามาครบ 10 บิตแล้ว และให้ส่งสัญญาณมาเคลียวงจรเปลี่ยนข้อมูลอนุกรมเป็นข้อมูลขนาน ส่วนสัญญาณ Out d0 ถึง Out d7 เป็นสัญญาณข้อมูลไบนารีแบบขนานที่ได้จากวงจรแปลงข้อมูลอนุกรมเป็นข้อมูลขนาน รูปสัญญาณต่าง ๆ แสดงได้ดังรูปที่ 5.3

จะเห็นว่า เมื่อจับรูปคลื่นสัญญาณทั้งทางภาคมอดูเลเตอร์และดีมอดูเลเตอร์เปรียบเทียบกันแล้ว ดังรูปที่ 5.4 แสดงให้เห็นว่าการส่ง-รับข้อมูล เอฟ.เอส.เค. นี้เป็นไปอย่างถูกต้อง

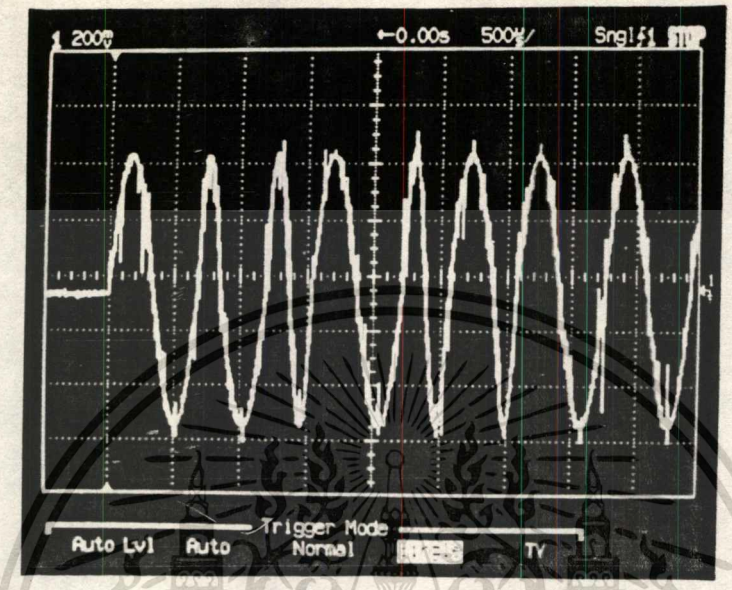


รูปที่ 5.3 สัญญาณในภาคดีมอดูเลเตอร์

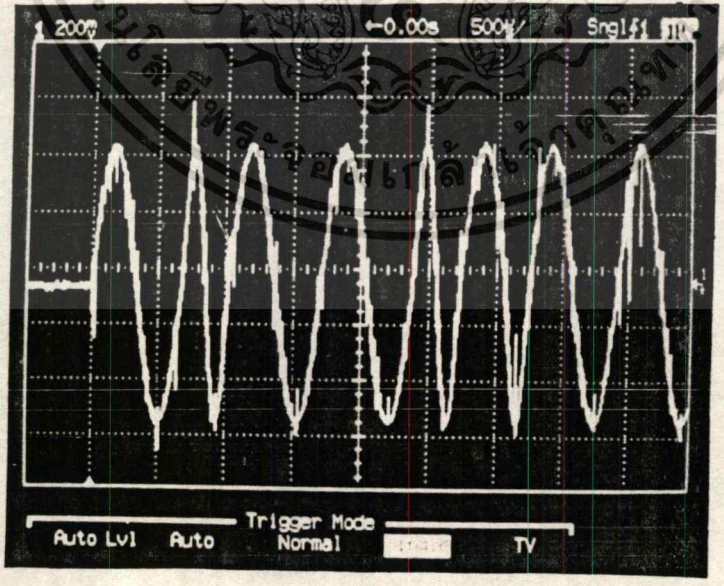


รูปที่ 5.4 สัญญาณเปรียบเทียบระหว่างภาคมอดูเลเตอร์และภาคดีมอดูเลเตอร์

หลังจากที่ทำการมอดูเลตสัญญาณผ่านทางสายแล้วโดยส่งสัญญาณ เอฟ.เอส.เค แบบครึ่งคาบเวลาต่อข้อมูล 1 บิต นี้ ไปเข้าเครื่องส่ง เอฟ.เอ็ม. แบบไร้สาย (FM Wireless) ส่วนในด้านภาคดีมอดูเลเตอร์ ได้ทำการต่อเข้าเครื่องรับ เอฟ.เอ็ม. แล้วนำสัญญาณที่ได้จากภาครับ เอฟ.เอ็ม. ไปเข้ายังภาคดีมอดูเลเตอร์ที่ทำขึ้น ทดลองทำการส่งสัญญาณข้อมูล เอฟ.เอส.เค ปรากฏว่า การส่ง-รับ ข้อมูลนี้ถูกต้องเป็นที่น่าพอใจในระยะทางไม่ไกลมาก ในการทดลองได้ทำการวัดสัญญาณที่เอาท์พุทของเครื่องรับ เอฟ.เอ็ม. ดังแสดงในรูปที่ 5.5 สามารถส่งข้อความได้ถูกต้อง ซึ่งแสดงให้เห็นว่าถ้านำวิธีการนี้ไปใช้กับสถานีวิทยุกระจายเสียงในการส่งข้อมูล มีแนวทางที่สามารถเป็นไปได้ และสามารถนำไปใช้ประโยชน์ด้านต่าง ๆ ในการทดลองได้ทำการทดลองส่งข้อมูลที่เป็นแบบไบท์และข้อมูลที่เป็นแบบไฟล์ ซึ่งในการทดลองส่งข้อมูลนั้นสามารถดีมอดูเลตสัญญาณได้ถูกต้อง ขั้นตอนในการส่ง-รับข้อมูลจะมีการแสดงขั้นตอนต่างๆ ที่จอเครื่องคอมพิวเตอร์ให้ทราบ แสดงได้ดังรูปที่ 5.6 ถึงรูปที่ 5.9



รูปที่ 5.5 a) สัญญาณข้อมูล 66



รูปที่ 5.5 b) สัญญาณข้อมูล 61

รูปที่ 5.5 สัญญาณผ่านเครื่อง FM Wireless

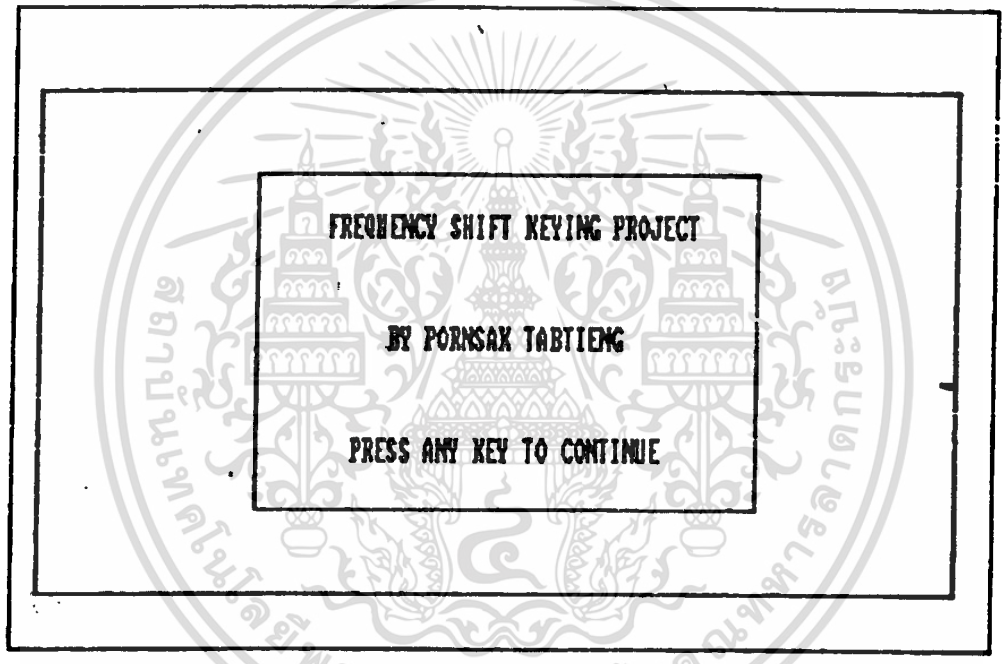
LINKING...

รูปที่ 5.6 a) ขั้นตอนที่ 1 โหลดโปรแกรม

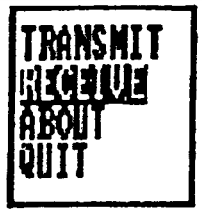
COMMUNICATION COMPLETE

PRESS ANY KEY TO CONTINUE

รูปที่ 5.6 b) ขั้นตอนที่ 2



รูปที่ 5.6 c) ขั้นตอนที่ 3



รูปที่ 5.6 d) ขั้นตอนที่ 4 เลือกการรับหรือส่งข้อมูล

SELECT TRANSMISSION TYPE

SELECT TRANSMISSION TYPE

รูปที่ 5.6 e) ขั้นตอนที่ 5 เลือกรับ-ส่งไบท์และไฟล์

รูปที่ 5.6 แสดงผลหน้าจอขั้นตอนการรับ-ส่งข้อมูล


```
A:\>dir demo.txt

 Volume in drive A is FSK_PROJECT
 Volume Serial Number is 0E5A-1DD3
 Directory of A:\

DEMO     TXT      4756 07-27-92   8:16
        1 file(s)       4756 bytes
                      869888 bytes free
```

```
TRANSMIT FILE MODE
PLEASE ENTER FILENAME TO TRANSMIT : demo.txt
PLEASE SELECT BLOCK SIZE TO TRANSMIT
```

รูปที่ 5.7 a) เลือกขนาดบล็อกของการส่งไฟล์

รูปที่ 5.7 b) ไฟล์ข้อมูลที่ใช้ทดลองส่ง

รูปที่ 5.7 แสดงผลหน้าจอในการส่งข้อมูลและไฟล์ข้อมูลที่ใช้ทดลองส่ง

```
        TRANSMIT FILE MODE
PLEASE ENTER FILENAME TO TRANSMIT : demo.txt
PLEASE SELECT BLOCK SIZE TO TRANSMIT
FILE LENGTH = 4756 BYTE
BLOCK SIZE  = 1 KBYTE (1024BYTE)
NUMBER OF BLOCK = 4 BLOCK
REMAINDER = 660 BYTE
  ♦♦ ö@demo     txt
TRANSMISSION BLOCK  4
TRANSMISSION REMAINDER
TRANSMISSION FINISH
DO YOU WANT TO TRANSMIT OTHER FILES (Y/N) :
```

รูปที่ 5.8 a) ด้านเครื่องส่ง

```
          RECEIVE FILE MODE
RECEIVING HEADER
 ♦♦ ö@demo     txt
RECEIVE HEADER FINISH
FILENAME = demo.txt
BLOCK SIZE = 1 KBYTE
NUMBER OF BLOCK = 4 BLOCK
REMAINDER BYTE  = 660 BYTE
FILE LENGTH = 4756 BYTE
```

รูปที่ 5.8 b) ด้านเครื่องรับ

รูปที่ 5.8 แสดงผลหน้าจอการรับ-ส่งข้อมูลแบบไบท์

```
        TRANSMIT BYTE MODE

TRANSFER ASCII CODE FOR KEY PRESS (Esc = EXIT)

adssedffyghgvjyijloilk;kHBNK;KMYTDFRTE13345656779809809809-97642ED vtgvextvbyuhvCR
YIYBYvtu6c5iyuoiu6563vhxcmkjnh87yb6t67dh7i7gtihjt3x4t5   685f3cethi8 8yyg9o8hv g
qqqqsssssss  t6rdf3wyuyu  t7ce5478yoh  ygiydcg K K  M K K K
```

รูปที่ 5.9 a) ด้านเครื่องส่ง

```
        RECEIVE BYTE MODE

adssedffyghgvjyijloilk;kHBNK;KMYTDFRTE13345656779809809809-97642ED vtgvextvbyuhvCR
YIYBYvtu6c5iyuoiu6563vhxcmkjnh87yb6t67dh7i7gtihjt3x4t5   685f3cethi8 8yyg9o8hv g
qqqqsssssss  t6rdf3wyuyu  t7ce5478yoh  ygiydcg K K  M K K K
```

รูปที่ 5.9 b) ด้านเครื่องรับ

รูปที่ 5.9 แสดงผลหน้าจอการรับ-ส่งข้อมูลแบบไฟล์

บทที่ 6
สรุป

จากการทดลองส่งและรับข้อมูลโดยการมอดูเลตและดีมอดูเลตสัญญาณ เอฟ. เอส. เค. แบบสัญญาณครึ่งคาบเวลาต่อข้อมูล 1 บิต เมื่อเทียบกับวิธีการเดิม จะเห็นว่าสามารถส่งข้อมูลได้อัตราบิตสูงขึ้นกว่าวิธีเดิม[1] ถึงแม้จะมีการใช้ความถี่เท่ากัน โดยมีอัตราบิตสูงสุดในการส่งถึง 5860 บิตต่อวินาทีและอัตราบิตต่ำสุด 2930 บิตต่อวินาทีเพราะว่าข้อมูลที่ได้จากการทดลองมีสัญญาณแบบต่อเนื่อง และไม่มีการสูญเสียเวลาโดยเปล่าประโยชน์เหมือนในแบบเดิมทั้งมีวงจรที่ไม่ยุ่งยาก

จากการทดลองโดยการใช้สายเป็นตัวกลาง นำสัญญาณข้อมูลจากเครื่องคอมพิวเตอร์ไปยังเครื่องคอมพิวเตอร์อีกตัวหนึ่ง โดยผ่านวงจรควบคุมที่ใช้ ซีพียู 8031 ปรากฏผลเป็นที่น่าพอใจ ถึงแม้ว่าการส่งข้อมูลนี้จะมีความกว้างของสัญญาณข้อมูลบิต"0"และ"1"ไม่เท่ากันและไม่สามารถใช้ร่วมกับระบบที่ใช้อยู่ในปัจจุบัน จากการทำงานของภาคดีมอดูเลเตอร์เป็นแบบการเปรียบเทียบเวลาของสัญญาณ ดังนั้นในการส่งระยะทางไกล ๆ จะทำให้ขนาดของสัญญาณลดลง เนื่องจากการสูญเสียของสัญญาณในตัวกลาง ซึ่งจะเห็นว่าเป็นสาเหตุทำเกิดการผิดพลาดในการดีมอดูเลตสัญญาณในระบบปัจจุบัน

จากการส่งข้อมูล เอฟ. เอส. เค. จะเห็นว่า ไม่สามารถนำไปใช้งานร่วมกับอุปกรณ์อื่นที่มีมาตรฐานอยู่ในปัจจุบันได้ เช่น มาตรฐานการส่งข้อมูลแบบอนุกรมและไม่ชิงโครไนซ์[1] ซึ่งจะมีความกว้างของสัญญาณข้อมูลบิต "0" และ "1" เท่ากัน แต่เนื่องจากวงจรมีการทำงานง่าย ไม่ยุ่งยากและราคาถูก เมื่อเทียบกับแบบที่เคยใช้มาซึ่งจะเหมาะกับการนำไปใช้ในการสั่งการข้อมูลดิจิตอลบางอย่างที่ไม่ต้องการต่อกับอุปกรณ์ที่ใช้ตามมาตรฐานในปัจจุบัน และข้อดีของ เอฟ. เอส. เค. แบบครึ่งคาบเวลาต่อข้อมูล 1 บิตตามโครงการนี้ เป็นการใช้การเปรียบเทียบเวลาของสัญญาณ

เอฟ.เอส.เค. ที่รับเข้ามากับสัญญาณที่มีเวลาคงที่ ดังนั้น แม้ว่าสัญญาณจะมีขนาดลดต่ำลงไปเนื่องจากการลดทอนในสายส่ง ก็จะไม่ทำให้ภาคดีมอดูเลเตอร์ทำงานผิดพลาดทำให้การรับ-ส่งข้อมูลเป็นไปอย่างถูกต้อง

จากการส่งข้อมูล เอฟ.เอส.เค แบบครึ่งคาบเวลานี้เห็นว่า ขีดจำกัดในการส่งข้อมูลก็คือแบนด์วิดท์ของตัวกลาง หรือสายส่งสัญญาณระหว่างเครื่องรับ-ส่ง[3] เช่นเดียวกับวิธีเดิมที่เคยใช้มา และอีกสิ่งคือความถี่ของสัญญาณไซน์ของข้อมูลบิต 0 และ 1 จะต้องเลือกความถี่ในการใช้งานให้เหมาะสมคือ ในความถี่สูงสุดมีค่าใกล้เคียงกับตัวกลางทางสายส่งที่จะสามารถส่งไปได้ และความถี่ต่ำที่ใช้ก็ควรให้มีค่าใกล้เคียงกับความถี่สูงแต่อย่าให้ใกล้เคียงกันมาก เพราะจะเกิดการยุ่งยากในการดีมอดูเลตสัญญาณทำให้การรับ-ส่งข้อมูลมีโอกาสผิดพลาดได้มากยิ่งขึ้น ดังนั้น ต้องทำการเลือกความถี่ที่ใช้งานที่เหมาะสม เพื่อจะให้เกิดการผิดพลาดในการส่งข้อมูลน้อยที่สุด

การส่งสัญญาณ เอฟ.เอส.เค. ด้วยวิธีนี้สามารถที่จะนำไปใช้งานเฉพาะในส่วนที่มีวิธีการเหมือนกันเท่านั้น ดังนั้น จึงสามารถนำไปใช้ประโยชน์ในด้านเฉพาะงานตามแต่หน่วยงานผู้ใช้

ภาคผนวก ก.

โปรแกรมควบคุมที่ใช้กับวงจรควบคุม CPU 8031

```
          CPU       "8051.TBL"
          HOF       "INT8"

;MCS-51 INTERNAL REGISTERS

B:        EQU       0F0H      ;B REGISTER
A:        EQU       0E0H      ;A REGISTER SAME AS ACC
ACC:      EQU       0E0H      ;ACCUMULATOR
PSW:      EQU       0D0H      ;PROGRAM STATUS WORD
IPC:      EQU       0B8H      ;INTERRUPT PRIORITY
P3:       EQU       0B0H      ;PORT 3
IEC:      EQU       0A8H      ;INTERRUPT ENABLE
P2:       EQU       0A0H      ;PORT 2
SBUF:     EQU       99H       ;SEND BUFFER
SCON:     EQU       98H       ;SERIAL CONTROL
P1:       EQU       90H       ;PORT 1
TH1:      EQU       8DH       ;TIMER 1 HIGH
TH0:      EQU       8CH       ;TIMER 0 HIGH
TL1:      EQU       8BH       ;TIMER 1 LOW
TL0:      EQU       8AH       ;TIMER 0 LOW
TMOD:     EQU       89H       ;TIMER MODE
TCON:     EQU       88H       ;TIMER CONTROL
PCON:     EQU       87H       ;POWER CONTROL REGISTER
DPH:      EQU       83H       ;DATA POINTER HIGH
DPL:      EQU       82H       ;DATA POINTER LOW
SP:       EQU       81H       ;STACK POINTER
P0:       EQU       80H       ;PORT 0

;MCS-51 INTERNAL BIT ADDRESSES

CY:       EQU       0D7H      ;CARRY FLAG
AC:       EQU       0D6H      ;AUXILIARY-CARRY FLAG
F0:       EQU       0D5H      ;USER FLAG 0
RS1:      EQU       0D4H      ;REGISTER SELECT MSB
RS0:      EQU       0D3H      ;REGISTER SELECT LSB
OV:       EQU       0D2H      ;OVERFLOW FLAG
P:        EQU       0D0H      ;PARITY FLAG
PS:       EQU       0BCH      ;PRIORITY SERIAL PORT
PT1:      EQU       0BBH      ;PRIORITY TIMER 1
PX1:      EQU       0BAH      ;PRIORITY EXTERNAL 1
PT0:      EQU       0B9H      ;PRIORITY TIMER 0
PX0:      EQU       0B8H      ;PRIORITY EXTERNAL 0
EA:       EQU       0AFH      ;ENABLE ALL INTERRUPT
ES:       EQU       0ACH      ;ENABLE SERIAL INTERRUPT
ET1:      EQU       0ABH      ;ENABLE TIMER 1 INTERRUPT
EX1:      EQU       0AAH      ;ENABLE EXTERNAL 1 INTERR
ET0:      EQU       0A9H      ;ENABLE TIMER 0 INTERRUPT
EX0:      EQU       0A8H      ;ENABLE EXTERNAL 0 INTERR
SM0:      EQU       09FH      ;SERIAL MODE 0
SM1:      EQU       09EH      ;SERIAL MODE 1
SM2:      EQU       09DH      ;SERIAL MODE 2
REN:      EQU       09CH      ;SERIAL RECEPTION ENABLE
TB8:      EQU       09BH      ;TRANSMITT BIT 8
RB8:      EQU       09AH      ;RECEIVE BIT 8
TI:       EQU       099H      ;TRANSMIT INTERRUPT FLAG
RI:       EQU       098H      ;RECEIVE INTERRUPT FLAG
TF1:      EQU       08FH      ;TIMER 1 OVERFLOW FLAG
TR1:      EQU       08EH      ;TIMER 1 RUN CONTROL BIT
```

```
TF0:        EQU         08DH        ;TIMER 0 OVERFLOW FLAG
TR0:        EQU         08CH        ;TIMER 0 RUN CONTROL BIT
IE1:        EQU         08BH        ;EXT INTERR. 1 EDGE FLAG
IT1:        EQU         08AH        ;EXT INTERR. 1 TYPE FLAG
IE0:        EQU         089H        ;EXT INTERR. 0 EDGE FLAG
IT0:        EQU         088H        ;EXT INTERR. 0 TYPE FLAG

ESC:    EQU             1BH

ERRCD:          EQU             11H
CRRCD:          EQU             22H
STCMD:          EQU             33H
PC_TER:         EQU             55H
TER_PC:         EQU             0AAH

BYTEMD:         EQU             0FH
TRANMD:         EQU             0FH

FILEMD:         EQU             0F0H
RECEMD:         EQU             0F0H

BUFF_COMMD:     EQU 30H
CHKSUM:         EQU             42H
STACK:          EQU             45H

            ORG         0000H
            LJMP        INITIAL

INITIAL:
          MOV           R0,#00
          MOV           R1,#00
DLSTART1:
            DJNZ        R0,$                ;DELAY FOR START PROGRAM
          DJNZ          R1,DLSTART1

          LCALL           CLRBUFF           ;CLEAR BUFFER COMMAND
          MOV           P1,#00H
          MOV           SP,#STACK

          MOV           DPTR,#8003H       ;PORT A & PORT C OUTPUT
          MOV           A,#82H          ;PORT B INPUT
          MOVX          @DPTR,A


;INITIAL SERIAL PORT
;BUAD RATE = 9600 BPS

          MOV           SCON,#01010000B             ;SET SERIAL PORT
MODE1
          MOV           TMOD,#00100000B             ;BUAD RATE = 9600
          MOV           TH1,#253D
          SETB          TR1

        MOV             A,#TER_PC
        MOV             SBUF,A
;SEND DATA TO LINK WITH PC
          TX_PC                 ;RETURN STATUS LINK COMPLETE TO PC

;CHECK COMMUNICATION LINK
CHKCOMLK:
          RX_PC
          CJNE          A,#PC_TER,CHKCOMLK
```



```
;SEND COMMAND IF DATA IS  0FH - TRANSMITTER
```

```
;                   F0H - RECEIVER

CHKCOMMD1:
        MOV          DPTR,#8002H
        MOV          A,#10H ;SET RECEIVER
        MOVX         @DPTR,A

        RX_PC
        CJNE         A,#0F0H,CHKESCMAIN
        MOV          DPTR,#8002H
        MOV          A,#10H
        MOVX         @DPTR,A                ;PREPARE LINE FOR
RECEIVE
        LCALL        RECEIVER               ;SELECT FOR RE-
CEIVER
        SJMP         NEXTCOMMD
CHKESCMAIN:
        CJNE         A,#ESC,BTRANS
        LJMP         INITIAL
BTRANS:
        MOV          DPTR,#8002H
        MOV          A,#20H
        MOVX         @DPTR,A
        LCALL        TRANSMITTER
NEXTCOMMD:
        LCALL        CLRBUFF
        CLR            IT0
        CLR            IT1
        SJMP         CHKCOMMD1

;SEND COMMAND IF DATA IS  0FH - COMMUNICATION BYTE
;                        F0H - COMMUNICATION FILE
TRANSMITTER:
        RX_PC
        CJNE         A,#0F0H,TCHKESC
        LCALL        TRANFILE
        SJMP         TRANSMITTER
TCHKESC:
        CJNE         A,#ESC,BTRANBYTE
        SJMP         RETURNT
BTRANBYTE:
        LCALL        TRANBYTE
        SJMP         TRANSMITTER
RETURNT:
        RET

;COMMUNICATION BYTE OF DATA IS THAT SENDING OUT FOR ONE
CHARACTER
;UNTIL DATA IS '1B' (ESCAPE) IS ESC

TRANBYTE:
        PUSH           A
LPTRANBYTE:
        RX_PC
        LCALL          MODDATB
        CJNE         A,#ESC,LPTRANBYTE ;CHECK ESCAPE KEY
        POP            A
        RET


;ROUTINE FOR MODULATE DATA IN  FSK PATTERN
;DATA THAT SENDING IN ACCUMULATOR
MODDATB:
        PUSH         DPH
```

```
        PUSH        DPL
        PUSH        A

        SETB        IT1
        MOV         R3,A

        MOV         R2,#180D
MODDAT1B:
        JNB         IE1,$
        CLR         IE1
        MOV         A,R3
        MOV         P1,A                ;PUT DATA TO MOD FSK

        SETB        P3 OR 4
        CLR         P3 OR 5
        DJNZ        R2,$                ;DELAY FOR LOAD
        SETB        P3 OR 5
        CLR         IT1

        POP         A
        POP         DPL
        POP         DPH
        RET

;COMMUNICATION FILE BY SEND BLOCK OF DATA
(1K,2K,4K,8K,32K)
TRANFILE:
        PUSH        A
RECECMDPC:
        MOV         R0,#BUFF_COMMD
        RX_PC
        CJNE        A,#ESC,RECENEXT
        LCALL       MODDAT
        LJMP        QUITTF
RECECMDPC1:
        RX_PC
RECENEXT:
        MOV         @R0,A  ;SAVE COMMAND 18 BYTES
        INC         R0
        CJNE        R0,#BUFF_COMMD+18,RECECMDPC1      ;RE-
CEIVE COMMAND

        MOV         R0,#BUFF_COMMD
SECMDFSK:
        MOV         A,@R0
        LCALL       MODDAT
        INC         R0
        CJNE        R0,#BUFF_COMMD+18,SECMDFSK

;CHECK NUMBER OF BLOCK DATA IS 00 ?

        MOV         A,34H
        CJNE        A,#00,TRANNORM
        MOV         A,33H
        CJNE        A,#00,TRANNORM
        MOV         A,#STCMD
        TX_PC
        SJMP        QUITBLK
TRANNORM:
;START PROCESSING OF TRANSMISSION FILE
        MOV       A,#STCMD
        TX_PC
```

```
        MOV         DPTR,#0000

LPSEBLKFSK:
        LCALL         RE1BLKPC          ;RECEIVE ONE BLOCK OF
DATA FROM PC
LPREPSE:
        LCALL       SE1BLKFSK           ;SEND ONE BLOCK

        MOV           R6,#00
        MOV         R7,#00
DLTRANS:
        DJNZ        R6,$

        LCALL         SECRPC
        INC         DPTR
        MOV         A,DPH
        CJNE        A,34H,LPSEBLKFSK    ;CHECK NUMBER OF
BLOCK
        MOV         A,DPL
        CJNE        A,33H,LPSEBLKFSK

QUITBLK:

;CHECK NUMBER OF REMAINDER DATA IS 00 ?

        MOV         A,36H
        CJNE        A,#00,LPSERMD
        MOV         A,35H
        CJNE        A,#00,LPSERMD
        SJMP        NEXT_1
LPSERMD:
        LCALL         RECERMDPC         ;RECEIVE REMAINDER
FROM PC
        MOV         R5,#50D
        MOV         R6,#00H
        MOV         R7,#00H
DLBREMD:
        DJNZ          R6,$
        DJNZ          R7,DLBREMD
        DJNZ        R5,DLBREMD

LPREPRMD:
        LCALL       SERMDFSK            ;SEND REMAINDER

NEXT_1:
        LCALL       SECRPC
        LJMP        RECECMDPC
QUITTF:
        POP         A
        RET

;ROUTINE FOR SEND ONE BLOCK OF DATA TO RF CIRCUIT
SE1BLKFSK:
        PUSH        DPH
        PUSH          DPL
        PUSH          A

        MOV         DPTR,#0000

;SEND BLOCK OF DATA
LPSE1BLK:
        MOVX        A,@DPTR             ;READ DATA FROM EXTER-
NAL RAM
        LCALL       MODDAT
```

```
        INC         DPTR
        MOV         A,DPH
        CJNE        A,32H,LPSE1BLK      ;COMPARE HIGH ORDER
BYTE
        MOV         A,DPL
        CJNE        A,31H,LPSE1BLK      ;COMPARE LOW ORDER
BYTE

        POP         A
        POP         DPL
        POP         DPH
        RET

;ROUTINE FOR MODULATE DATA IN  FSK PATTERN
;DATA THAT SENDING IN ACCUMULATOR
MODDAT:
        PUSH        DPH
        PUSH        DPL
        PUSH        A

        MOV         R7,#00H
        MOV         R6,#10H
DLMODDAT:
        DJNZ        R7,$
        DJNZ        R6,DLMODDAT

        SETB        IT1
        MOV         R3,A

        MOV         R2,#180D
MODDAT1:
        JNB         IE1,$
        CLR         IE1
        MOV         A,R3
        MOV         P1,A                ;PUT DATA TO MOD FSK
        SETB        P3 OR 4
        CLR         P3 OR 5
        DJNZ        R2,$                ;DELAY FOR LOAD
        SETB        P3 OR 5
        CLR         IT1

        POP         A
        POP         DPL
        POP         DPH
        RET

;ROUTINE FOR RECEIVE BLOCK OF DATA FROM PC (RECEIVE
BLOCK)
RE1BLKPC:
        PUSH        DPH
        PUSH        DPL
        PUSH        A

        MOV         R6,#00
        MOV         DPTR,#0000
LPREBLKPC:
        RX_PC
        MOVX        @DPTR,A             ;STORE DATA IN EXTER-
NAL RAM

        INC         DPTR
        MOV         A,DPH
        CJNE        A,32H,LPREBLKPC     ;CHECK HIGH BYTE
```

```
        MOV         A,DPL
        CJNE        A,31H,LPREBLKPC     ;CHECK LOW BYTE

        POP           A
        POP           DPL
        POP           DPH
        RET

;ROUTINE FOR SEND REMAINDER OF DATA TO RF CIRCUIT
SERMDFSK:
        PUSH        DPH
        PUSH          DPL
        PUSH          A

        MOV         DPTR,#0000
LPSERE:
        MOVX        A,@DPTR
        LCALL       MODDAT
        INC           DPTR
        MOV         A,DPH
        CJNE        A,36H,LPSERE      ;COMPARE HIGH ORDER
BYTE
        MOV         A,DPL
        CJNE        A,35H,LPSERE        ;COMPARE LOW ORDER
BYTE

        POP           A
        POP           DPL
        POP         DPH
        RET

;ROUTINE FOR RECEIVE REMAINDER OF DATA FROM PC
RECERMDPC:
        PUSH          DPH
        PUSH          DPL
        PUSH          A
        MOV         DPTR,#0000
LPRERMD:
        RX_PC
        MOVX        @DPTR,A           ;STORE DATA IN EXTER-
NAL RAM
        INC         DPTR
        MOV         A,DPH
        CJNE        A,36H,LPRERMD       ;CHECK REMAINDER
        MOV         A,DPL
        CJNE        A,35H,LPRERMD

        POP           A
        POP           DPL
        POP           DPH
        RET

;ROUTINE FOR CLEAR BUFFER COMMAND
CLRBUFF:
        PUSH          A
        MOV         R0,#BUFF_COMMD
        MOV         A,#00
LPCLRBF:
        MOV         @R0,A
        INC         R0
        CJNE        R0,#BUFF_COMMD+18,LPCLRBF
        POP           A
        RET
```

```
;ROUTINE AS RECEIVER
;RECEIVE COMMAND IF DATA IS  0FH - COMMUNICATION BYTE
;                            F0H - COMMUNICATION FILE
RECEIVER:
        RX_PC

        CJNE        A,#0F0H,RCHKESC
        LCALL       REFILE
        SJMP        RECEIVER
RCHKESC:
        CJNE        A,#ESC,BREBYTE
        SJMP        RETURNR
BREBYTE:
        LCALL       REBYTE
        SJMP        RECEIVER
RETURNR:
        RET

;ROUTINE FOR RECEIVE DATA IN BYTE FORMAT
REBYTE:
        PUSH          A
LPGETBYTE:
        LCALL         GET1BYTE
        TX_PC
        CJNE        A,#ESC,LPGETBYTE      ;CHECK ESCAPE KEY
        POP         A
        RET

;COMMUNICATION FILE BY RECEIVE BLOCK OF DATA
(2K,4K,8K,32K)
REFILE:
        PUSH          A
LPGETCMD:
        MOV         R0,#BUFF_COMMD
LPGETCMD1:
        LCALL         GET1BYTE
        CJNE        A,#ESC,SAVECMD
        TX_PC
        LJMP        QUITRF
GETCMDNEXT:
        LCALL         GET1BYTE
SAVECMD:
        MOV         @R0,A
        INC         R0
        CJNE        R0,#BUFF_COMMD+18,GETCMDNEXT      ;RE-
CEIVE COMMAND
        MOV         R0,#BUFF_COMMD

;TIME THE WASTE FOR SEND COMMAND TO PC ABOUT 0.5 SEC
SECMDPC:
        MOV         A,@R0
        TX_PC
        INC         R0
        CJNE        R0,#BUFF_COMMD+18,SECMDPC      ;SEND
COMMAND TO PC

;CHECK NUMBER OF BLOCK DATA IS 00 ?

        MOV         A,34H
        CJNE        A,#00,RECENORM
        MOV         A,33H
        CJNE        A,#00,RECENORM
        SJMP        CHKREM1
RECENORM:
```

```
;RECEIVE FILE
        MOV         DPTR,#0000
LPGETBLK:
        LCALL       GET1BLK             ;RECEIVE ONE BLOCK
        LCALL       SEBLKPC             ;SEND BLOCK OF DATA TO
PC
        INC         DPTR
        MOV         A,DPH
        CJNE        A,34H,LPGETBLK
        MOV         A,DPL
        CJNE        A,33H,LPGETBLK

;CHECK NUMBER OF REMAINDER DATA IS 00 ?
CHKREM1:
        MOV         A,36H
        CJNE        A,#00,LPGETRMD0
        MOV         A,35H
        CJNE        A,#00,LPGETRMD0
        SJMP        RETURNM
LPGETRMD0:
        LCALL       GETRMDRF                ;REMAINDER
        LCALL         SERMDPC

RETURNM:
        LJMP        LPGETCMD
QUITRF:
        POP         A
        RET

;ROUTINE FOR RECEIVE ONE BLOCK OF DATA FROM RF CIRCUIT
GET1BLK:
        PUSH        DPH
        PUSH          DPL
        PUSH          A
        MOV         DPTR,#0000
        MOV         R6,#00

;GET BLOCK OF DATA
LPGET1BLK:
        LCALL       GET1BYTE
        MOVX        @DPTR,A             ;STORE DATA IN EXTER-
NAL RAM
        INC         DPTR
        MOV         A,DPH
        CJNE        A,32H,LPGET1BLK     ;CHECK BLOCK
        MOV         A,DPL
        CJNE        A,31H,LPGET1BLK
        POP         A
        POP         DPL
        POP         DPH
        RET

;ROUTINE FOR RECEIVE ONE BYTE OF DATA
;RETURN DATA IN ACCUMULATOR
GET1BYTE:
        PUSH          DPH
        PUSH          DPL
        SETB        IT0
        MOV           DPTR,#8002H           ;PREPARE FOR RE-
CEIVE DATA
        MOV         A,#10H
        MOVX          @DPTR,A             ;CLEAR SHIFT REGISTER
        SETB        A OR 0
```

```
        MOVX    @DPTR,A             ;ENABLE SHIFT REGIS-
TER

        JNB     IE0,$
        CLR       IE0
        MOV     DPTR,#8001H
        MOVX    A,@DPTR
        MOV     R4,A

        MOV     DPTR,#8002H
        MOV     A,#10H
        MOVX    @DPTR,A
        MOV     A,R4
        CLR       IT0

        POP     DPL
        POP     DPH
        RET

;ROUTINE FOR SEND BLOCK OF DATA TO PC
SEBLKPC:
        PUSH      DPH
        PUSH      DPL
        PUSH      A
        MOV     DPTR,#0000
LPSEBLKPC:
        MOVX    A,@DPTR             ;READ DATA FROM EXTER-
NAL RAM
        TX_PC
        INC       DPTR
        MOV     A,DPH
        CJNE    A,32H,LPSEBLKPC     ;COMPARE LOW ORDER
BYTE
        MOV     A,DPL
        CJNE    A,31H,LPSEBLKPC     ;COMPARE HIGH OR-
DER BYTE
        POP     A
        POP       DPL
        POP     DPH
        RET

;ROUTINE FOR RECEIVE REMAINDER OF DATA FROM RF CIRCUIT
GETRMDRF:
        PUSH    DPH
        PUSH      DPL
        PUSH      A
        MOV     DPTR,#0000
        MOV     R6,#00
LPGETRMDRF:
        LCALL   GET1BYTE
        MOVX    @DPTR,A             ;WRIE TO EXTERNAL RAM
        INC     DPTR
        MOV     A,DPH
        CJNE    A,36H,LPGETRMDRF    ;CHECK REMAINDER
        MOV     A,DPL
        CJNE    A,35H,LPGETRMDRF
        POP     A
        POP       DPL
        POP     DPH
        RET

;ROUTINE FOR SEND REMAINDER OF DATA TO PC
SERMDPC:
        PUSH      DPH
```

```
        PUSH        DPL
        PUSH        A
        MOV         DPTR,#0000
LPSERMDPC:
        MOVX        A,@DPTR             ;RESTORE DATA FROM
EXTERNAL RAM
        TX_PC
        INC         DPTR
        MOV         A,DPH
        CJNE        A,36H,LPSERMDPC   ;COMPARE HIGH ORDER
BYTE
        MOV         A,DPL
        CJNE        A,35H,LPSERMDPC     ;COMPARE LOW ORDER
BYTE
        POP         A
        POP         DPL
        POP         DPH
        RET


;SEND CORRECT CODE TO PC
SECRPC:
        PUSH        A
        MOV         A,#CRRCD
        TX_PC
        POP         A
        RET


;TRANSMISSION TO PC
TX_PC:      MACRO
        JNB         TI,$
        CLR         TI
        MOV         SBUF,A
        ENDM

;RECEIVING FROM PC
RX_PC:      MACRO
        JNB         RI,$
        CLR         RI
        MOV         A,SBUF
        ENDM

END
```

ภาคผนวก ข

โปรแกรมสำหรับควบคุมและแสดงการรับ-ส่งข้อมูลบนเครื่องคอมพิวเตอร์

```
#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <alloc.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>
#include <io.h>

#define  BORDER   1
#define  ESC      27
#define  REV_VID  0x70
#define  NORM_VID 7
#define  MAX_BUF 8192+200

int   popup(char *menu[],char *, int, int, int, int);
void  save_video(int,int,int,int,unsigned int *);
void  restore_video(int,int,int,int,unsigned char *);
void  goto_xy(int,int);
void  cls(void);
void  write_video(int,int,char *,int);
void  display_menu(char *menu[],int,int,int);
void  draw_border(int,int,int,int);
void  transmit(void);
void  receive(void);
void  transmit_byte(void);
void  transmit_file(void);
void  receive_byte(void);
void  receive_file(int);
void  title(void);
int   get_resp(int,int,int,char *menu[],char *);
int   is_in(char *, char);
void  initial(void);
void  handshake(unsigned char,unsigned char);
void  senddata(unsigned char);
unsigned char recedata(void);

unsigned int  COM1 = 0x3F8;   /* serial port address */
unsigned int  TXBUFF1,RXBUFF1;
unsigned int  DIV_LSB1,DIV_MSB1,IER1,IIR1,LCR1,MCR1,LSR1,MSR1;
unsigned char BAUDL1,BAUDH1;

unsigned int  COM2 = 0x2F8;   /* serial port address */
unsigned int  TXBUFF2,RXBUFF2;
unsigned int  DIV_LSB2,DIV_MSB2,IER2,IIR2,LCR2,MCR2,LSR2,MSR2;
unsigned char BAUDL2,BAUDH2;

unsigned long length;
//unsigned char Buf[MAX_BUF];
unsigned char filename[30],buf[MAX_BUF];
FILE *loadfile,*savefile;

char *main_menu[] = {
   "TRANSMIT",
   "RECEIVE",
   "ABOUT",
   "QUIT"
```

```
};

char *sup_menu[] = {
   "BYTE",
   "FILE"
};

char *size_menu[] = {
   " 1 KBYTE",
   " 2 KBYTE",
   " 4 KBYTE",
   " 8 KBYTE",
   "32 KBYTE"
};

void main()
{
   int i,select;
   cls();
   initial();
   goto_xy(10,0);
   printf("                                LINKING...");
   handshake(0x55,0xaa);
   goto_xy(9,0);
   printf("                              COMMUNICATION COMPLETE\n");
   printf("                                                    \n");
   printf("                             PRESS ANY KEY TO CONTINUE");
   getch();
   title();
   do{
     cls();
     select = popup(main_menu,"traq",4,1,3,BORDER);
     switch(select){
      case -1 : cls();
                break;
      case  0 : transmit();
                break;
      case  1 : receive();
                break;
      case  2 : title();
                break;
      case  3 : inportb(RXBUFF1);
                exit(0);
     }
   }while(1);
}

void initial(void)
{

/*INITIAL 8250 */
TXBUFF1 = COM1; RXBUFF1  = COM1;
DIV_LSB1 = COM1;     DIV_MSB1 = COM1+1;
IER1     = COM1+1;   IIR1     = COM1+2; LCR1     = COM1+3;
MCR1     = COM1+4;   LSR1     = COM1+5; MSR1     = COM1+6;
BAUDL1   = 12;     BAUDH1   = 0;

/*SET DLAB = 1 buad rate = 2400 bps */
outportb(LCR1,128);  outportb(DIV_LSB1,BAUDL1);
outportb(DIV_MSB1,BAUDH1);
/*SET DLAB = 0 buad rate = 2400 bps */
```

```
outportb(LCR1,3);    outportb(IIR1,1); outportb(IER1,0); outportb(MCR1,3);

TXBUFF2 = COM2; RXBUFF2  = COM2;
DIV_LSB2 = COM2;     DIV_MSB2 = COM2+1;
IER2     = COM2+1;   IIR2     = COM2+2; LCR2     = COM2+3;
MCR2     = COM2+4;   LSR2     = COM2+5; MSR2     = COM2+6;
BAUDL2   = 6;     BAUDH2   = 0;

/*SET DLAB = 1 buad rate = 9600 bps */
outportb(LCR2,128);  outportb(DIV_LSB2,BAUDL2); outportb(DIV_MSB2,BAUDH2);

/*SET DLAB = 0 buad rate = 9600 bps */
outportb(LCR2,3);    outportb(IIR2,1); outportb(IER2,0); outportb(MCR2,3);

}


void handshake(unsigned char s_data,unsigned char r_data)
{
  while(recedata() != r_data);
  senddata(s_data);
}


void senddata(unsigned char data)
{
unsigned char status_uart;
status_uart = inportb(LSR1) & 32 ;
 while (status_uart == 0)
    {
    status_uart = inportb(LSR1) & 32;  /* wait for tx buffer empty */
    }
    outportb(TXBUFF1,data);
}


unsigned char recedata(void)
{
unsigned char status_uart;
status_uart = inportb(LSR1) & 1;
 while (status_uart == 0)
 {
 status_uart = inportb(LSR1) & 1;
  }  /* wait for rx buffer received data */
   return inportb(RXBUFF1);
}


void transmit()
{
  int select;
  cls();
  senddata(0x0f);
  delay(10);
  goto_xy(12,10);
  printf("SELECT TRANSMISSION TYPE");
  select = popup(sup_menu,"bf",2,1,10,BORDER);
  switch(select){
      case -1 : cls();
            senddata(0x1b);
            break;
      case  0 : transmit_byte();
            break;
```

```
        case  1 : transmit_file();
              break;
    }
}

void receive()
{
  int select,shake;
  cls();
  senddata(0xf0);
  delay(10);
  goto_xy(12,10);
  printf("SELECT RECEIVE TYPE");
  select = popup(sup_menu,"bf",2,1,10,BORDER);
  switch(select){
      case -1 : cls();
            senddata(0x1b);
            break;
      case  0 : receive_byte();
            break;
      case  1 : do{
                receive_file(shake);
            }while(shake==0);
            break;
    }
}


void transmit_byte()
{
 unsigned char key;
 cls();
 senddata(0x0f);
 delay(10);
 goto_xy(12,10);
 printf("TRANSMIT BYTE MODE \n");
 printf("\nTRANSFER ASCII CODE FOR KEY PRESS (Esc = EXIT)\n");
 do{
   key = getche();
   putchar(0x7);
   senddata(key);
   delay(1);
 }while(key != 0x1b); /* Esc */
}


void transmit_file(void)
{
  unsigned char *size;
  int  i, j, k, nread;
  unsigned long  block, block_size, num_block, remainder;
  int  select;
  senddata(0xf0);
  delay(10);
  do{
    cls();
    goto_xy(12,10);
    printf("TRANSMIT FILE MODE\n");
    printf("PLEASE ENTER FILENAME TO TRANSMIT : ");
    gets(filename);
    loadfile = fopen(filename,"rb");
```

```
if(loadfile == NULL){
  printf("NO FILE %s\n",filename);
  printf("PRESS ANY KEY TO BACK TO MAIN MENU");
  getch();
}
else{
  printf("PLEASE SELECT BLOCK SIZE TO TRANSMIT");
  select = popup(size_menu,"12483",5,1,18,BORDER);
  switch(select){
  case -1 : cls();
            break;
  case  0 : block = 1;
            break;
  case  1 : block = 2;
            break;
  case  2 : block = 4;
            break;
  case  3 : block = 8;
            break;
  case  4 : block = 32;
  }
  if(select != -1){
  length = filelength(fileno(loadfile));
  block_size = block*1024;
  num_block = length / block_size;
  remainder = length % block_size;
  goto_xy(15,0);
  printf("FILE LENGTH = %lu BYTE\n",length);
  printf("BLOCK SIZE  = %lu KBYTE (%luBYTE)\n",block,block_size);
  printf("NUMBER OF BLOCK = %lu BLOCK\n",num_block);
  printf("REMAINDER = %lu BYTE\n",remainder);
  senddata(0xff);
  putchar(0xff);
  size = (char *)&block_size;
  senddata(*size);                  /* SENT LOW  BYTE OF BLOCK 'S SIZE */
  putchar(*size);
  senddata(*(size+1));      /* SENT HIGH BYTE OF BLOCK 'S SIZE */
  putchar(*(size+1));
  size = (char *)&num_block;
  senddata(*size);                  /* SENT LOW  BYTE OF NUMBER OF BLOCK
*/
  putchar(*size);
  senddata(*(size+1));              /* SENT HIGH BYTE OF NUMBER OF BLOCK
*/
  putchar(*(size+1));
  size = (char *)&remainder;
  senddata(*size);                /* SENT LOW  BYTE OF REMAINDER */
  putchar(*size);
  senddata(*(size+1));            /* SENT HIGH BYTE OF REMAINDER */
  putchar(*(size+1));
  i = 0;
  while((i<8) && (filename[i] != '.')){
    senddata(filename[i]);
    putchar(filename[i]);
    i++;
  }
  j = i+1;
  for(;i<8;i++){
    senddata(0);
    putchar(0);
  }
  for(i=j;i<(j+3);i++){
    senddata(filename[i]);
    putchar(filename[i]);
```

```
        }
        putchar('\n');
        while(recedata() != 0x33)
          printf("RECEIVING CORRECT COMMAND\r");
        printf("                              \r");
        for(i=0;i < num_block ;i++){
          printf("TRANSMISSION BLOCK %2d\r",i+1);
          for(j=0;j < (block*4);j++){
            nread = fread(buf,1,256,loadfile);
            for(k=0;k < nread;k=k+1){
              senddata(buf[k]);
            }
          }
          while(recedata() != 0x22);
        }
        printf("\nTRANSMISSION REMAINDER\n");
        while(!feof(loadfile)){
          nread = fread(buf,1,256,loadfile);
          for(i=0;i<nread;i=i+1){
            senddata(buf[i]);
          }
        }
        while(recedata() != 0x22);
        printf("TRANSMISSION FINISH\n");
        printf("PRESS ANY KEY TO BACK TO MAIN MENU\n");
        getch();
        }
        fclose(loadfile);
      }
    }while(select == -1);
}


void receive_byte()
{
 unsigned char key;
 cls();
 senddata(0x0f);
 delay(10);
 goto_xy(12,10);
 recedata();
 printf("RECEIVE BYTE MODE\n");
 while(key != 0x1b){
   key = recedata();
   printf("%c",key);
 }
}


void receive_file(int shake)
{
//  char far *fptr;
  int i, j;
  unsigned int *s;
  unsigned long  S, N, R, L;
  unsigned long  block_size, num_block, remainder;
  cls();
  senddata(0xf0);
  delay(10);
  goto_xy(12,10);
  recedata();
  printf("RECEIVE FILE MODE\n");
  printf("RECEIVING HEADER\n");
```

```
    if((buf[0] = recedata()) == 0x1b)
      shake = 0;
    else{
      shake = 1;
    }
    for(i=1;i<18;i++){
      buf[i] = recedata();
      putchar(buf[i]);
    }
    printf("\nRECEIVE HEADER FINISH\n");
    s = (unsigned int *)&buf[1];
    block_size = *s;
    s = (unsigned int *)&buf[3];
    num_block  = *s;
    s = (unsigned int *)&buf[5];
    remainder  = *s;
    for(i=0,j=0;i<8;i++){
      if(buf[i+7] != 0){
        filename[i] = buf[i+7];
        j++;
      }
    }
    filename[j] = '.';
    sscanf(&buf[15],"%3s",&filename[j+1]);
    length = (block_size*num_block)+remainder;
    printf("FILENAME = %s\n",filename);
    printf("BLOCK SIZE = %lu KBYTE\n",block_size/1024);
    printf("NUMBER OF BLOCK = %lu BLOCK\n",num_block);
    printf("REMAINDER BYTE  = %lu BYTE\n",remainder);
    printf("FILE LENGTH = %lu BYTE\n",length);
    savefile = fopen(filename,"wb");
    if(savefile == NULL){
      printf("NO FILE %s\n",filename);
      printf("PRESS ANY KEY TO BACK TO MAIN MENU");
      shake = 0;
      getch();
    }
    else{
//    fptr = (char far *) farmalloc(block_size);
      L = 0;
      for(N=0;N<num_block;N++){
        for(S=0;S<block_size;S++){
        buf[L] = recedata();
//      printf("BYTE RECEIVED =  %7lu\r",S+1);
//      fputc(key,savefile);
        L++;
        }


    }
//  printf("\nRECEIVING REMAINDER (%lu)\n",remainder);
    for(R=0;R<remainder;R++){
      buf[L] = recedata();
      L++;
    }
    printf("BLOCK RECEIVED = %2lu LENGTH = %7lu\n",N,L);
//    farfree(fptr);
    fwrite(buf,L,1,savefile);
    fclose(savefile);
    printf("BYTE RECEIVED = %7lu\r",R);
    printf("\nTOTAL BYTE RECEIVE %lu\n",L);
    printf("RECEIVE FINISH\n");
    printf("PRESS ANY KEY TO BACK TO MAIN MENU");
```

```
    getch();
  }
}


void title()
{
 cls();
 draw_border(1,1,24,79);
 draw_border(3,3,21,76);
 draw_border(6,20,18,60);
 goto_xy(8,26);
 printf("FREQUENCY SHIFT KEYING PROJECT");
 goto_xy(16,26);
 printf("   PRESS ANY KEY TO CONTINUE");
 getch();
 putchar(0x7);
}


int  popup(menu,keys,count,x,y,border)
  char *menu[];
  char *keys;
  int count;
  int x, y;
  int border;
{
   register int i,len;
   int endx, endy, choice;
//   char far *fptr;
   unsigned int *p;
   if((x>24) || (x<0) || (y>79) || (y<0)) {
     printf("range error");
     return -2;
   }
   len = 0;
   for(i = 0; i<count; i++)
     if(strlen(menu[i]) > len) len = strlen(menu[i]);
   endy = len + 2 + y;
   endx = count + 1 + x;
   if((endx+1>24) || (endy+1>79)) {
     printf("menu won't fit");
     return -2;
   }
   p = (unsigned int *)malloc((endx-x+1) * (endy-y+1));
   if(!p) exit(1);

   save_video(x, endx+1, y, endy+1, p);

   if(border) draw_border(x,y,endx,endy);

   display_menu(menu, x+1, y+1, count);

   choice = get_resp(x+1, y, count, menu, keys);

   restore_video(x, endx+1, y, endy+1, (char *)p);
   free(p);
   return choice;
}
```

```
void display_menu(menu, x, y, count)
  char  *menu[];
  int   x, y, count;
{
   register  int i;

   for(i = 0; i<count; i++,x++) {
     goto_xy(x, y);
     printf(menu[i]);
   }
}


void draw_border(startx,starty, endx, endy)
  int  startx, starty, endx, endy;
{
   register int i;

   for(i=startx+1; i<endx; i++) {
      goto_xy(i, starty);
      putchar(179);
      goto_xy(i, endy);
      putchar(179);
   }
   for(i=starty+1; i<endy; i++) {
      goto_xy(startx, i);
      putchar(196);
      goto_xy(endx, i);
      putchar(196);
   }
   goto_xy(startx, starty);putchar(218);
   goto_xy(startx, endy);putchar(191);
   goto_xy(endx, starty);putchar(192);
   goto_xy(endx, endy);putchar(217);
}


int get_resp(x, y, count, menu, keys)
  int x, y, count;
  char *menu[];
  char *keys;
{
   union inkey {
     char ch[2];
     int i;
   } c;
   int arrow_choice=0, key_choice;

   y++;

   goto_xy(x, y);
   write_video(x, y, menu[0], REV_VID);

   for(;;) {
     while(!bioskey(1)) ;
     c.i = bioskey(0);

     goto_xy(x+arrow_choice, y);
     write_video(x+arrow_choice, y, menu[arrow_choice], NORM_VID);

     if(c.ch[0]) {
       key_choice = is_in(keys, tolower(c.ch[0]));
       if(key_choice) return key_choice-1;
```

```
        switch(c.ch[0]){
        case '\r' : return arrow_choice;
        case ' '  : arrow_choice++;
                    break;
        case ESC  : return -1;
        }
      }
      else {
        switch(c.ch[1]) {
        case 72 : arrow_choice--;
                  break;
        case 80 : arrow_choice++;
                  break;
        }
      }
      if(arrow_choice==count) arrow_choice=0;
      if(arrow_choice<0) arrow_choice=count-1;

      goto_xy(x+arrow_choice, y);
      write_video(x+arrow_choice, y, menu[arrow_choice], REV_VID);
    }
  }


  void write_video(x, y, p, attrib)
    int x, y;
    char *p;
    int attrib;
  {
     union REGS r;
     register int i, j;

     for(i=y; *p; i++) {
       goto_xy(x, i);
       r.h.ah = 9;
       r.h.bh = 0;
       r.x.cx = 1;
       r.h.al = *p++;
       r.h.bl = attrib;
       int86(0x10, &r, &r);
     }
  }


  void save_video(startx, endx, starty, endy, buf_ptr)
    int startx, endx, starty, endy;
    unsigned int *buf_ptr;
  {
     union REGS r;
     register int i, j;
     for(i=starty ; i<endy; i++)
       for(j=startx; j<endx; j++) {
         goto_xy(j, i);
         r.h.ah = 8;
         r.h.bh = 0;
         *buf_ptr++ = int86(0x10, &r, &r);
         putchar(' ');
       }
  }


  void restore_video(startx, endx, starty, endy, buf_ptr)
    int startx, endx, starty, endy;
```

```
  unsigned char *buf_ptr;
{
   union REGS r;
   register int i, j;
   for(i=starty ; i<endy; i++)
     for(j=startx; j<endx; j++) {
       goto_xy(j, i);
       r.h.ah = 9;
       r.h.bh = 0;
       r.x.cx = 1;
       r.h.al = *buf_ptr++;
       r.h.bl = *buf_ptr++;
       int86(0x10, &r, &r);
     }
}

void cls()
{
   union REGS r;

   r.h.ah = 6;
   r.h.al = 0;
   r.h.ch = 0;
   r.h.cl = 0;
   r.h.dh = 24;
   r.h.dl = 79;
   r.h.bh = 7;
   int86(0x10, &r, &r);
}

void goto_xy(x, y)
  int x,y;
{
   union REGS r;

   r.h.ah = 2;
   r.h.dl = y;
   r.h.dh = x;
   r.h.bh = 0;
   int86(0x10, &r, &r);
}

int is_in(s, c)
  char *s, c;
{
   register int i;

   for(i=0; *s; i++) if(*s++==c) return i+1;
   return 0;
}
```

เอกสารอ้างอิง(REFFERENCE BOOK)

[1] Ferrel G. Stremler," INTRODUCTION TO COMMUNICATION SYSTEM ",Addison-Wesley publishing Company,1990.

[2] Frederick F. Driscoll,"DATA COMMUNICATIONS", Saunder College Publishing,1992.

[3] D.Petersen, "AUDIO, VIDEO AND TELECOMMUNICATIONS", McGraw-Hill Book Company,1992 .

[4] Leon W. Couch2, "DIGITAL AND ANALOG CONMMUNICATION SYSTEM", Macmillan Publishing Company.

[5] ดร.สุเจตน์ จันทรังษ์, " การเพิ่มอัตราการส่งข้อมูลด้วยวิธี FSK ", เอกสารประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 16, CP 019, วันที่ 25 - 26 พฤศจิกายน 2536

[6] ธันยวัต พึงแสงโชติช่วง, " เอฟ.เอส.เค. ", ปริญญานิพนธ์ คณะวิศวกรรมศาสตร์ วิทยาลัยมหานคร, ประจำปีการศึกษา 2535

[7] Martin S. Roden , "DIGITAL COMMUNICATION SYSTEMS DESIGN", Prentice-Hall International Inc,1989.