เครื่องกำเนิดสัญญาณคลื่นพาหะด้วยวงจรสังเคราะห์ความถี่ 15-20 MHz

GENERATOR SYNTHESIZER FOR CARRIER FREQUENCY 15-20 MHz

โดย

นายธีรเดช จุลอดุง

นายสิงหา บำรุงจิตร

น.ส.อัจฉราภรณ์ ดียิ่ง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

หัวข้อปริญญานิพนธ์	**เครื่องกำเนิดสัญญาณคลื่นพาหะด้วยวงจรสังเคราะห์ความถี่ 15-20 MHz** GENERATOR SYNTHESIZER FOR CARRIER FREQUENCY 15-20 MHz
คณะผู้จัดทำ	นายธีรเดช จุลอดุง นายสิงหา บำรุงจิตร น.ส.อัจฉราภรณ์ ดียิ่ง
อาจารย์ที่ปรึกษา	อาจารย์ ดลชัย สุขเจริญผล
สาขา	เทคโนโลยีอิเล็กทรอนิกส์
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2539

ศณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการการสอบปริญญานิพนธ์

---------- ประธานกรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )

เครื่องกำเนิดสัญญาณคลื่นพาหะด้วยวงจรสังเคราะห์ความถี่ 15-20MHz

โดย	นายธีรเดช	จุลอดุง	รหัส 38012009
	นายสิงหา	บำรุงจิตร	รหัส 38012031
	น.ส.อัจฉราภรณ์	ดียิ่ง	รหัส 38012042

อาจารย์ที่ปรึกษา อาจารย์ดลชัย สุขเจริญผล

ปีการศึกษา 2539

บทคัดย่อ

โครงงานฉบับนี้กล่าวถึง การกำเนิดสัญญาณคลื่นพาหะที่สร้างขึ้นโดยวงจรฟรีเควนซี่ซินทิไซเซอร์ แบบโปรแกรมเลือกความถี่ได้ โดยรับโปรแกรมที่จะทำให้ความถี่คลื่นพาหะเปลี่ยน รับจากคอมพิวเตอร์เป็นตัวควบคุมความถี่ ความถี่ที่สร้างขึ้นจะออกแบบให้อยู่ในช่วงความถี่ระหว่าง 15-20 เมกกะเฮิรตซ์ และในโครงงานนี้ยังนำไปประยุกต์ใช้งานเป็นวงจรฟรีเควนซีคอนเวอร์เตอร์ ซึ่งทำหน้าที่ในการแปลงความถี่คลื่นพาหะของสัญญาณให้สูงหรือต่ำลงได้

GENERATOR SYNTHESIZER FOR CARRIER FREQUENCY 15-20 MHz

BY	MR.THEETRADECH	JUN-ADUNG	NO. 38012009
	MR.SINGHA	BAMRUNGJIT	NO. 38012031
	MISS.ATCHARAPORN	DEEYING	NO. 38012042

ADVISER MR.DOLACHAI SOOKCHAROENPHOL

YEAR 1996

ABSTRACT

This project presents carrier generate by frequency Synthsizer. Carrier programable frequency is selected by program control from the computer to control carrier frequency. Frequency range is between 15 to 20 MHz. And this project is used to be frequency convertor by convertor carrier frequency for upper or lower.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดีเพราะได้รับความร่วมมือจากเพื่อนๆ ในกลุ่ม ที่ได้ร่วมแรงร่วมใจช่วยกัน ทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จออกมา และต้องขอขอบพระคุณอาจารย์คลชัย สุขเจริญผล ที่ได้ให้คำปรึกษา คำแนะนำและข้อคิดเห็นต่างๆ รวมทั้งเครื่องมือ อุปกรณ์และสถานที่ในการทำโครงงานครั้งนี้

ในด้านปัจจัยทางทุนทรัพย์ ต้องกราบขอบพระคุณ คุณพ่อ-คุณแม่ ที่ได้ให้ความอุปการะและสนับสนุนทางด้านการเงินและคอยให้กำลังใจ รวมทั้งต้องขอบคุณเพื่อนๆ ที่ให้คำแนะนำ คอยช่วยเหลือ และคอยให้กำลังใจเสมอมา

คณะผู้จัดทำ

นายธีรเดช จุลอดุง

นายสิงหา บำรุงจิตร

น.ส.อัจฉราภรณ์ ดียิ่ง

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

วงจรสังเคราะห์ความถี่หรือฟรีเควนซี่ซินทิไซเซอร์ (Frequency Synthesizer) เป็นวงจรที่สามารถผลิตความถี่ได้หลายๆ ค่าที่เอาท์พุทโดยใช้วงจรผลิตความถี่ที่มีสัญญาณอ้างอิงภายในวงจรเพียงสัญญาณเดียว ปัจจุบันที่นิยมใช้กันมากในเครื่องมือสื่อสารสมัยใหม่ เพราะวงจรสังเคราะห์ความถี่เป็นความถี่ที่มีเสถียรภาพสูงทั้งทางด้านความถี่และขนาดรวมทั้งสามารถควบคุมการทำงานได้สะดวก การสังเคราะห์ความถี่สามารถทำได้ทั้งทางตรงและทางอ้อมแต่ในปัจจุบันนิยมใช้การสังเคราะห์ความถี่โดยทางอ้อม (Indirect Coherent Synthesizer) ซึ่งอาศัยการทำงานของวงจรเฟสล็อกลูป (Phase Locked Loop) ซึ่งสามารถกระทำได้ทั้งแบบอนาล็อกและดิจิตอลซึ่งจะใช้คอมพิวเตอร์เป็นตัวควบคุมการทำงาน

วงจรสังเคราะห์ความถี่โดยตรงแบบดิจิตอล สามารถสัญญาณได้หลายรูปแบบขึ้นอยู่กับโปรแกรมแต่ก็มีข้อจำกัดในเรื่องความถี่สูงสุดจะขึ้นอยู่กับความเร็วของคอมพิวเตอร์ สำหรับการสังเคราะห์แบบโดยตรงในปัจจุบันไม่นิยมกัน เพราะสิ้นเปลืองอุปกรณ์อย่างมาก

สำหรับโครงงานนี้จะแสดงการสังเคราะห์โดยวิธีทางอ้อมโดยการใช้เฟสล็อกลูปซึ่งในบางครั้งเราอาจเรียกได้ว่า การสังเคราะห์ความถี่แบบเฟสล็อกลูป (Phase Lock Loop Frequency Synthesizer)

และในโครงงานนี้ยังสร้างสัญญาณ AM แบบ Balance Modulation (สัญญาณข้อมูลความถี่ 256 kHz สัญญาณ Carrier ความถี่ 1 MHz) ซึ่งใช้เป็นสัญญาณทดสอบจะเห็นได้ว่ามีความถี่ตายตัว เมื่อต้องการเปลี่ยนสัญญาณทดสอบนี้ส่งไปตามความถี่ที่ต้องการก็สามารถทำได้โดยนำสัญญาณทดสอบ AM นี้ ไปผสมความถี่ (Mixer หรือ Convertor) กับ Carrier ที่สร้างจาก Synthesizer แล้วจะได้ความถี่ผลบวก (Up Convertor) และความถี่ผลต่าง (Down Convertor) ซึ่งทำให้สามารถส่งความถี่ที่แตกต่างไปได้ และตรงกับความถี่ที่ต้องการ

บทที่ 2
หลักการโดยทั่วไปของการสังเคราะห์ความถี่

2.1 ระบบสังเคราะห์ความถี่

เครื่องรับส่งวิทยุในปัจจุบัน ส่วนใหญ่นิยมใช้วิธีสังเคราะห์ความถี่แทบทั้งนั้นวงจรที่ทำหน้าที่สังเคราะห์ความถี่เรียกว่า ซินทิไซเซอร์ซึ่งแปลว่าสังเคราะห์ ด้วยวิธีสังเคราะห์ความถี่นี้ ทำให้เครื่องรับและเครื่องส่งวิทยุมีการพัฒนาขีดความสามารถขึ้นสามารถโปรแกรมความถี่ใช้งานได้ทำให้เกิดความคล่องตัวในวงจรสื่อสารเป็นอย่างมาก

2.1.1 วิธีการสังเคราะห์ความถี่

วงจรสังเคราะห์ความถี่ คือ วงจรที่ทำหน้าที่ผลิตสัญญาณความถี่ขนาดพอเหมาะและมีขนาดความถี่ตามต้องการ การโปรแกรมสามารถทำได้โดยการตั้งสวิทซ์หรือปุ่ม แต่ในปัจจุบันนิยมใช้วิธีสั่งงานด้วย Computer

ช่วงความถี่ใช้งานของวงจรสังเคราะห์ความถี่จะจำกัดอยู่ในช่วงที่แน่นอนแล้วแต่การใช้งาน และความละเอียดของความถี่ที่เปลี่ยนไปเรียกว่า รีโซลูชั่น

วิธีการสังเคราะห์ความถี่แบ่งออกได้เป็น 2 วิธี คือ

1. วิธีสังเคราะห์โดยตรง (Direct Synthesis)

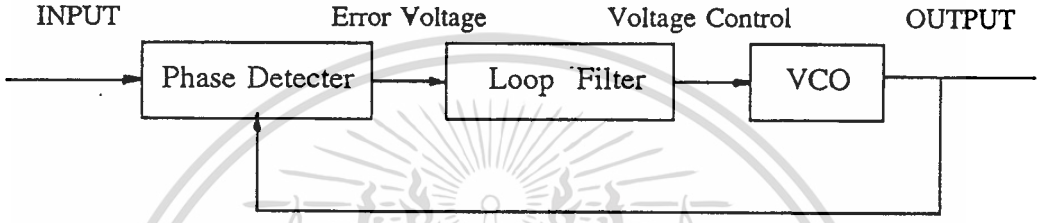
ซึ่งต้องใช้ความถี่หลายๆ ค่าผสมกันเพื่อให้ได้ความถี่ที่ต้องการ โดยปกติใช้ Xtal หลายชุด

2. วิธีสังเคราะห์โดยทางอ้อม (Indirect Synthesis)

วิธีการนี้อาศัยหลักการของเฟสล็อกลูป (Phase Lock Loop) โดยการกำหนดสัญญาณจากวงจรออสซิเลทเตอร์ ซึ่งควบคุมความถี่โดยปรับแรงดันที่เรียกว่า VCO สัญญาณจาก VCO จะถูกป้อนกลับมาเปรียบเทียบกับความถี่อ้างอิง จากนั้นนำความถี่ที่คาดเคลื่อนแปลงเป็นแรงดัน ไปทำการควบคุมการผลิตความถี่ของ VCO อีกครั้งหนึ่ง

2.2 เฟสล็อกลูป (Phase Lock Loop)

เฟสล็อกลูปเป็นระบบป้อนกลับที่บังคับให้วงจรออสซิลเลเตอร์มีความถี่หรือเฟสเปลี่ยนแปลงไปตามความถี่หรือเฟสของสัญญาณอ้างอิงภายนอก เฟสล็อกลูปประกอบด้วยภาคสำคัญ 3 ภาค คือ เฟสดีเทคเตอร์ (Phase Detector) ลูปฟิลเตอร์ (Loop Filter) และภาค VCO ดังรูป



รูปที่ 1 แสดงแผนผังเบื้องต้นของเฟสล็อกลูป

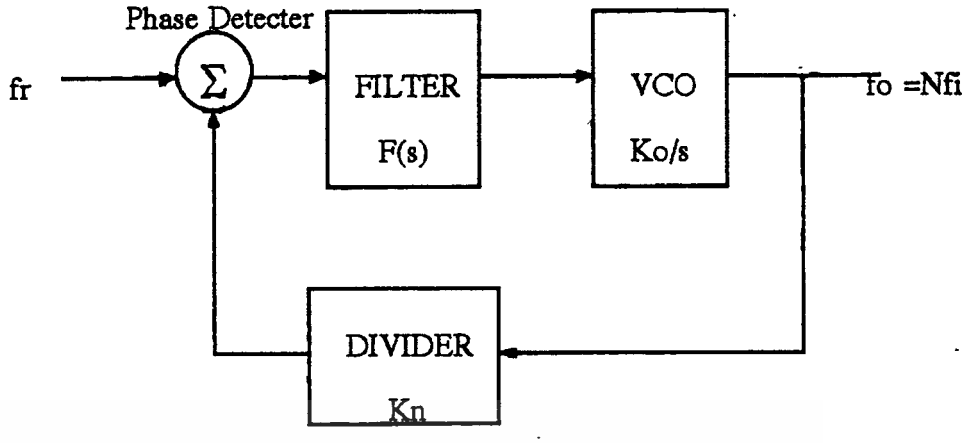
สมมุติมีสัญญาณความถี่อ้างอิงภายนอกเป็นสัญญาณรายคาบ (Periodic) เข้ามาที่อินพุท ภาคเทียบเฟสทำหน้าที่เปรียบเทียบเฟสระหว่างสัญญาณอ้างอิง

<u>การใช้เฟสล็อกลูปในการสังเคราะห์ความถี่</u>

การสังเคราะห์ความถี่มีอยู่หลายแบบ ตัวอย่างที่กล่าวดังต่อไปนี้เป็นการสังเคราะห์ความถี่ซึ่งมีการกำหนดความถี่แต่ละขั้นมีค่าเท่ากับความถี่อ้างอิง (fr)

<u>เฟสล็อกลูปโดยตรง</u>

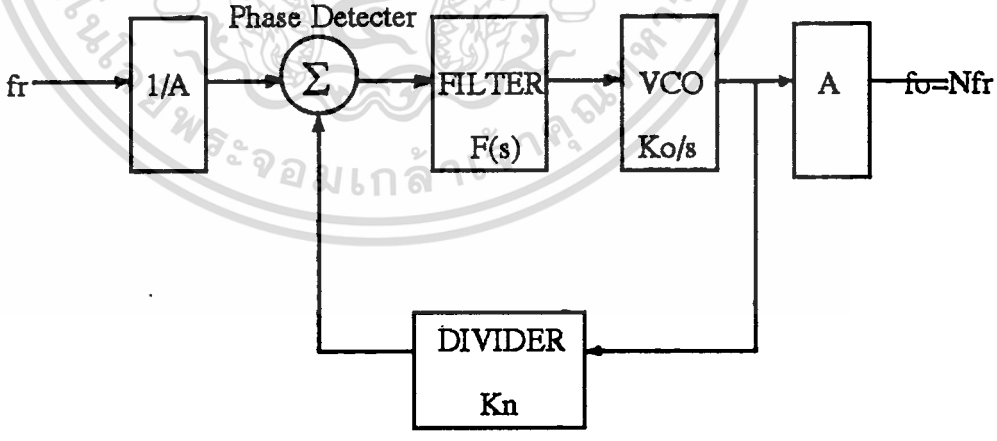
วิธีการสังเคราะห์แบบนี้ใช้ เฟสล็อกลูปแบบโดยตรงนับว่าเป็นวิธีการที่ง่าย โดยความถี่เอาท์พุทมีค่าเป็น N เท่าของความถี่อ้างอิง ดังรูป ในที่นี้ VCO ต้องสามารถทำงานได้ตลอดย่านความถี่เอาท์พุท ซึ่งความถี่อาจขึ้นไปถึง 200 MHz อย่างไรก็ดีวงจรนับที่ทำหน้าที่โปรแกรมเป็นตัวหาร N นั้นมีราคาแพง เราจึงจำเป็น ต้องปรับปรุงวิธีการสังเคราะห์ความถี่เป็นแบบอื่น



รูปที่ 2 แสดงการสังเคราะห์ความถี่เฟสล็อกลูปแบบโดยตรง

เฟสล็อกลูปแบบคูณความถี่

รูปที่ 3 เราจะหารความถี่อ้างอิงลง 9 เท่าก่อนที่จะป้อนให้แก่วงจรเฟสดีเทคเตอร์ และเอาท์พุทจาก VCO ก็คูณความถี่ขึ้นไป 9 เท่า วิธีนี้ช่วยลดความถี่การทำงานของวงจรหาร N ลง แต่ก็ทำให้การตอบสนองต่อการเปลี่ยนความถี่ของเฟสล็อกลูป ช้าลงเนื่องจากที่ใช้ในการเทียบเฟสต่ำลง

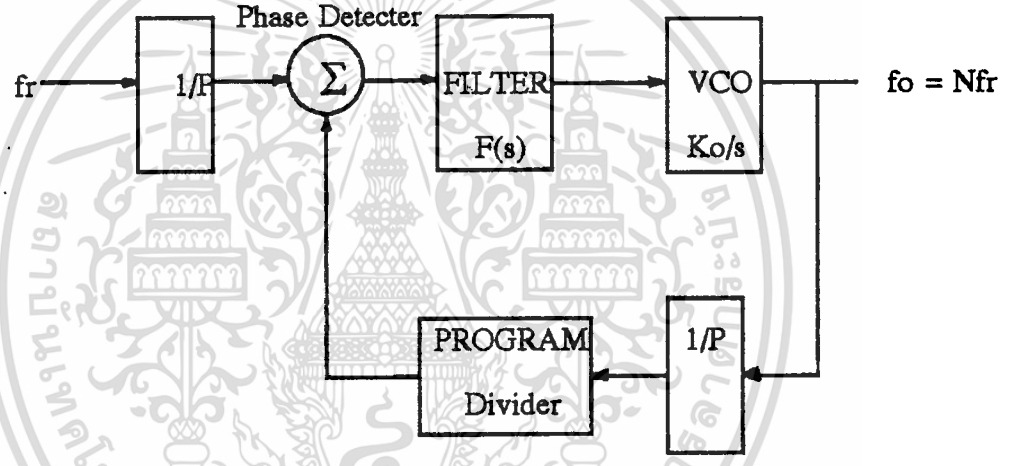


รูปที่ 3 แสดงเฟสล็อกลูปแบบคูณความถี่

เฟสล็อกลูปแบบพรีสเกลเลอร์

เฟสล็อกลูปในรูปที่ 4 ใช้วิธีการหารความถี่อ้างอิงลง P เท่าก่อนที่จะป้อนแก่วงจรดีเทคเตอร์และใช้วิธีคูณความถี่ขึ้นไป P เท่า ภายในลูปแทนที่จะคูณความถี่ภายนอกลูปดังเช่น เฟสล็อกลูปแบบคูณความถี่วงจร VCO ในกรณีนี้ต้องทำงานขึ้นไปถึงความถี่ใช้งาน โดยไม่ต้องมีวงจรทวีคูณ (Multipler)

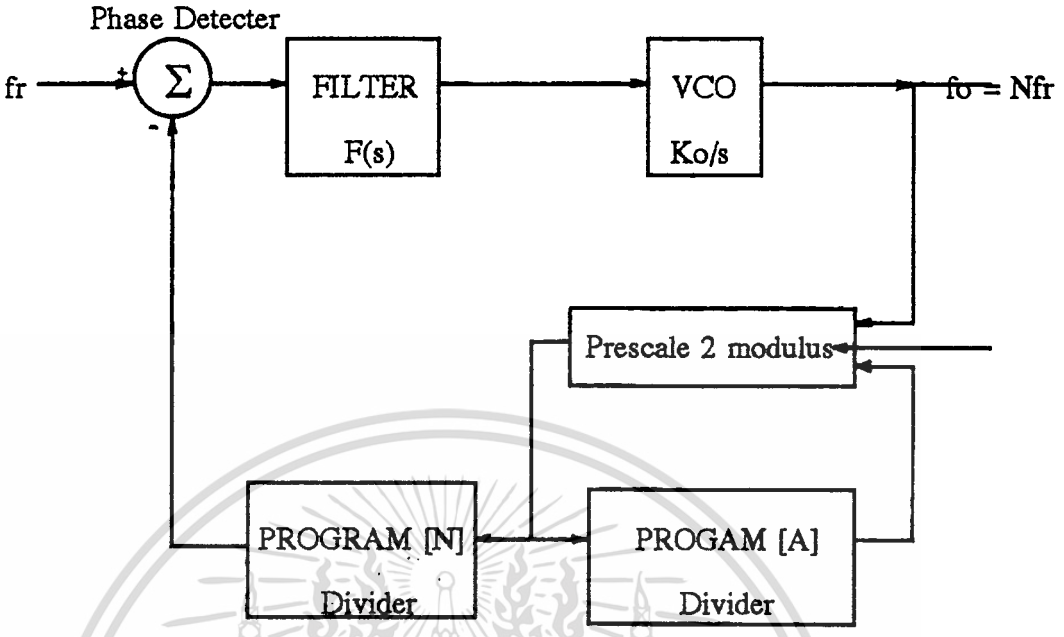
วงจรนับหาร P เป็นชุดวงจรฟลิบฟลอบธรรมดาซึ่งตัวหารกำหนดไว้ตายตัวและสามารถทำงานที่ความถี่สูงได้ เราเรียกวงจรนี้ว่า วงจรพรีสเกลเลอร์ ส่วนวงจรนับหาร N ซึ่งโปรแกรมตัวหารได้นั้นทำงานที่ความถี่ต่ำลงเช่นเดียวกับเฟสล็อกลูปในรูปที่ 3



รูปที่ 4 แสดงเฟสล็อกลูปแบบพรีสเกลเลอร์

เฟสล็อกลูปแบบพรีสเกลเลอร์สองโมดูลัส (Doual Modulus Prescaler)

เฟสล็อกลูปในรูปที่ 5 ใช้พรีสเกลเลอร์เช่นเดียวกันกับในรูปที่ 4 เว้นแต่วงจรพรีสเกลลอร์นี้ไม่ใช่เป็นวงจรนับหารค่าตายตัว P แต่เป็นวงจรนับหารเปลี่ยนค่าได้ระหว่าง P กับP+1 เราเรียกพรีสเกลเลอร์สองโมดูลัส(เลือกตัวหาร P ก็ได้ หรือเลือก P+1 ก็ได้) วงจรนับหาร N ซึ่งโปรแกรมตัวหารได้นั้นทำงานที่ความถี่ต่ำลง



รูปที่ 5 แสดงพรีสเกลเลอร์แบบสองโมดูลัส

2.3 เฟสดีเทคเตอร์

เฟสดีเทคเตอร์ (Phase Detector) เป็นส่วนหนึ่งของระบบเฟสล็อกลูป ซึ่งเป็นตัวทำหน้าที่ให้เกิดแรงดันอนาล็อกหรือดิจิตอลที่เอาท์พุทของตัวมัน ซึ่งค่าแรงดันที่ปรากฏออกมานี้จะเป็นอัตราที่แปรผันตามความต่างเฟสของสัญญาณอินพุท 2 สัญญาณที่เข้ามาในเฟสดีเทคเตอร์

ลักษณะของวงจรเฟสดีเทคเตอร์สามารถแบ่งการทำงานออกเป็น 2 แบบ คือ

1. อนาล็อกเฟสดีเทคเตอร์ (Analog Phase Detector)

ได้แก่ การมิกเซอร์ที่เอาท์พุทจะแปรตามขนาดของสัญญาณอินพุท วงจรที่ทำหน้าที่ได้แก่ Balance Mixer และ Sampling Detector เป็นการกำหนดโดยให้สัญญาณอ้างอิง fr เป็นสัญญาณพัลซ์ซึ่งมีคาบเวลาคงที่ ไปทำการสุ่มสัญญาณ อินพุทด้วยช่วงเวลาสั้นๆ โดยที่เอาท์พุทจะเป็นอัตราส่วนโดยตรงกัน

2. ดิจิตอลเฟสดีเทคเตอร์ (Digital Phase Detector)

มีหลายชนิดได้แก่

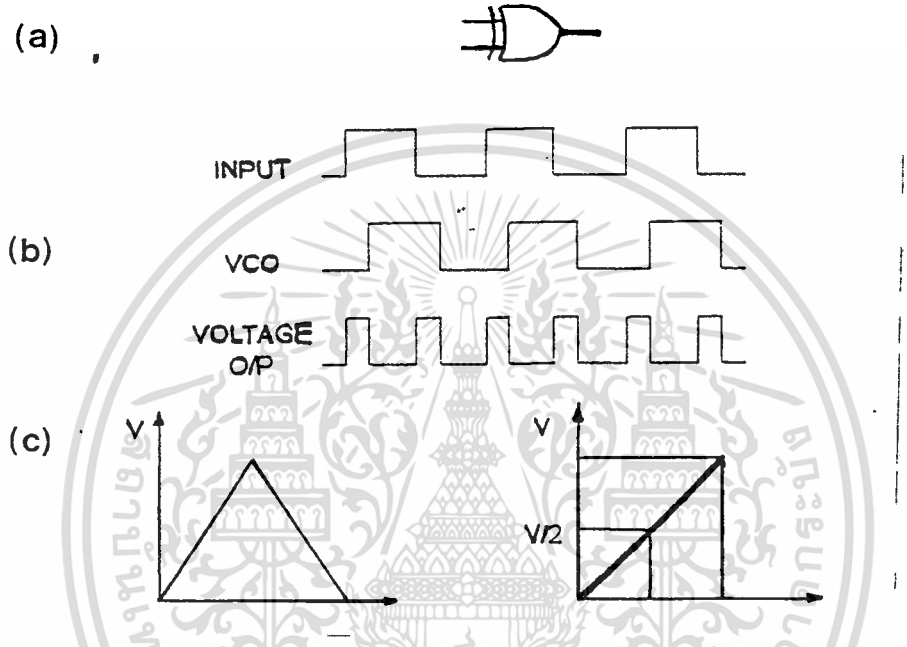
2.1 เอคคูซีฟออร์เฟสดีเทคเตอร์ (Exclusive OR Phase Detector)

เราสามารถนำเอาเอคคูซีฟออร์เกทมาทำเป็นเฟสดีเทคเตอร์ได้ โดยที่เอาท์พุทเป็น 1 ก็ต่อเมื่อสัญญาณอินพุทมีเฟสต่างกันและลอจิกเป็น 0 เมื่ออินพุททั้งสองมีเฟสเหมือนกัน แรงดันเฉลี่ยเอาท์พุทของเฟสดีเทคเตอร์จะเป็นตามสมการ

$$VOdc = Vp + D$$

โดยที่ Vp = แรงเคลื่อนสูงสุดของลอจิก 1

D = duty cycle



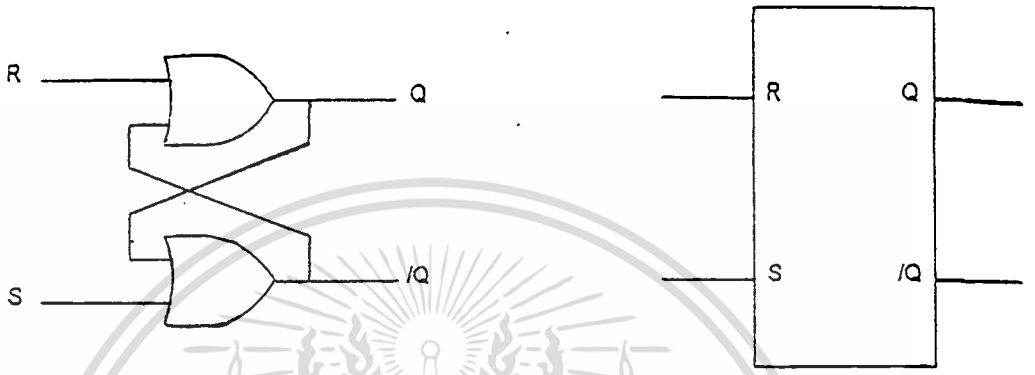
รูปที่ 6 a แสดงสัญญลักษณ์ของแอคคูซีฟออร์เฟสดีเทคเตอร์
b แสดงแรงดันเอาท์พุทที่สัมพันธ์กันระหว่างอิพุททั้งสองที่เข้ามา
c แสดงคุณสมบัติอินพุทเอาท์พุทของเฟสดีเทคเตอร์

จากคุณสมบัติอินพุทเอาท์พุทของเฟสดีเทคเตอร์ชนิดนี้ จะเห็นว่าสามารถใช้ได้ในช่วงต่างเฟสระหว่าง 0 - π โดยที่สัญญาณอินจำเป็นต้องมีค่าดิวตี้ไซเคิลเท่ากับ 50 เปอร์เซ็นต์ และเอาท์พุทที่ได้จะมีค่าความถี่เป็น 2 เท่าของความถี่อินพุท ส่วนค่าของคอนเวอร์ชันเกนเท่ากับ VDD/2π

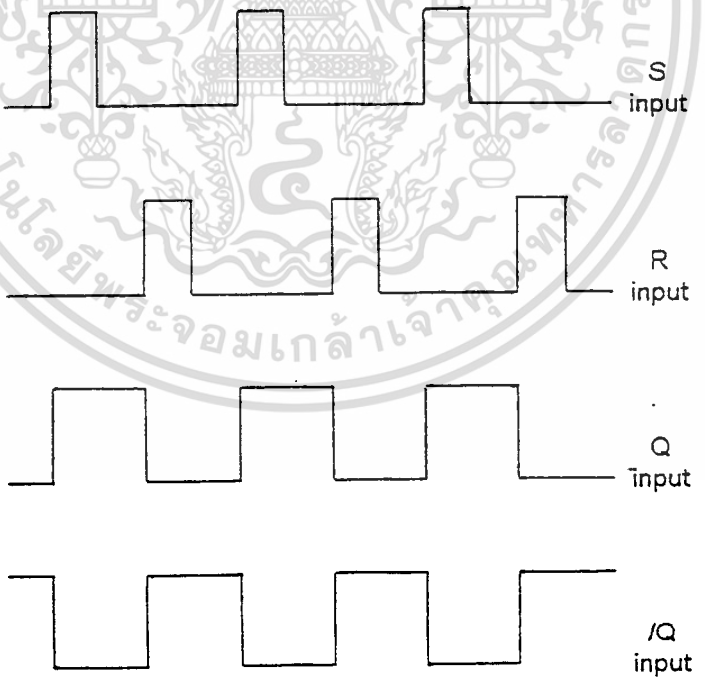
2.2 เอด-ทริกเกอร์ เฟสดีเทคเตอร์หรือฟลิปฟลอบเฟสดีเทคเตอร์

(Edge-Teiggered Phase Detector)

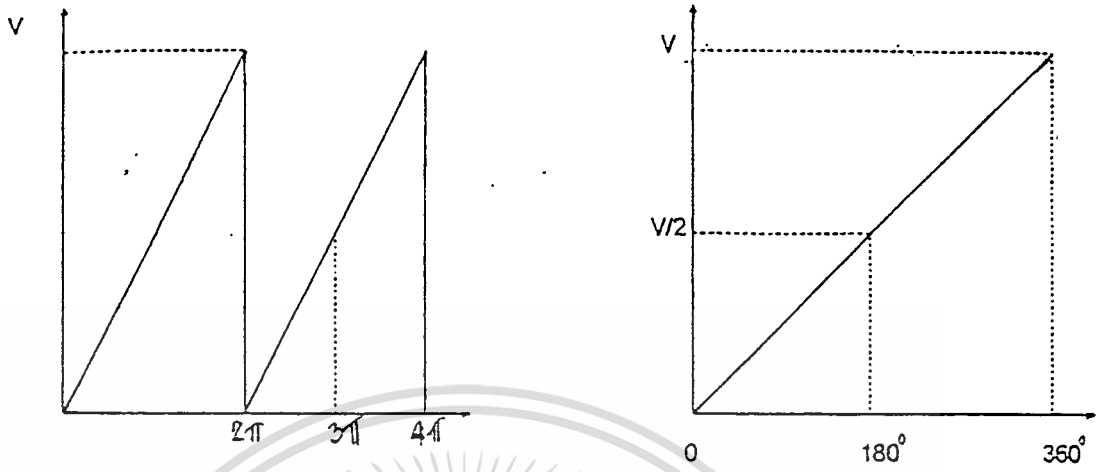
เป็นเฟสดีเทคเตอร์อีกชนิดหนึ่งที่ใช้ฟลิบฟลอบเป็นตัวทำให้เกิดแรงดันเอาท์พุทที่มีอัตราแปรผันกับสัญญาณอินพุททั้ง 2 ที่เข้ามา ดังแสดงในรูป



รูปที่ 7 แสดง RS F/F เฟสดีเทคเตอร์

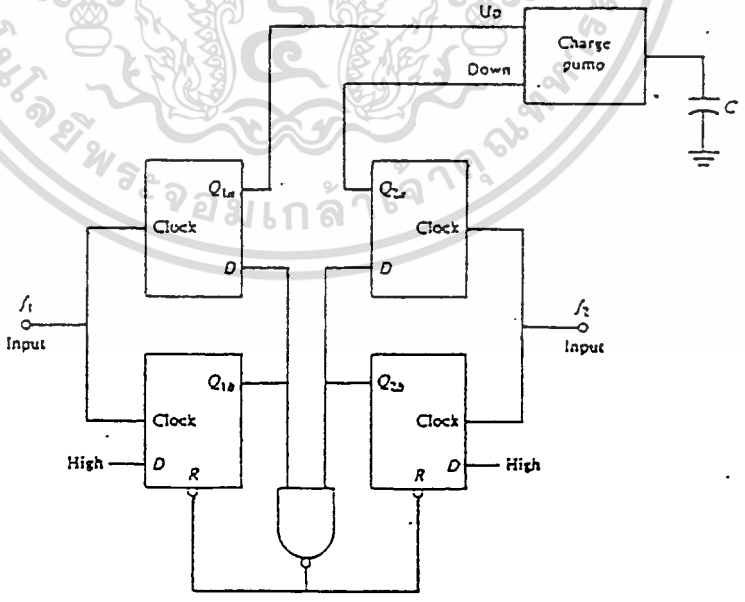


รูปที่ 8 แสดงสัญญาณอินพุทเอาท์พุทของ RS F/F เฟสดีเทคเตอร์



รูปที่ 9 แสดงคุณสมบัติอินพุทของ F/F เฟสดีเทคเตอร์

จากรูปจะเห็นได้ว่า วงจรเฟสดีเทคเตอร์ที่ใช้ทริกด้วยขอบสัญญาณพัลซ์ สามารถใช้ความต่างเฟสได้ตั้งแต่ 0-2π (ปีน 2 เท่าของแอคคลูซีฟออร์) และความถี่ของสัญญาณเอาท์พุท จะเท่ากับ 50 เปอร์เซนต์ ส่วนคอนเวอร์ชันแกนมีค่าเท่ากับ Vdd / 2π



รูปที่ 10 เฟสดีเทคเตอร์ที่สร้างจาก D-Flip Flop

3. <u>เฟส-ฟรีเควนซีดีเทคเตอร์</u> (Phase-Frequency Detertor)

จะเห็นได้ว่าเฟสดีเทคเตอร์แบบแอคคลูซีฟออร์และแบบฟลิบฟลอบจะเป็นวงจรที่ทำหน้าที่เป็นเฟสดีเทคเตอร์ได้ แต่มีข้อจำกัดอยู่ในตัวเองคือจำเป็นต้องมีการฟิลเตอร์สัญญาณเอาท์พุทที่ดี เพื่อแยกค่าเฉลี่ยของแรงดันดีซีที่ต้องการและเมื่อนำไปใช้ในวงจรเฟสล็อกลูปแล้ว จะให้ผลตอบสนองช้าเมื่ออินพุทมีความถี่ที่ต่างกันมาก จึงได้มีการพัฒนาเป็นวงจรเฟส-ฟรีเคว่นซีดีเทคเตอร์เพื่อแก้ข้อบกพร่องดังกล่าว

วงจรเฟส-ฟรีเควนซีดีเทคเตอร์จะนิยมใช้กับระบบเฟสล็กลูปที่ต้องการ การตอบสนองในย่านกว้าง เช่นในวงจรสังเคราะความถี่(Frequency Synthesizer) หรือวงจรควบคุมความเร็ว ของมอเตอร์ เป็นต้น

2.4 ลูปฟิลเตอร์(LOOP FILTER)

ลูปฟิลเตอร์เป็นส่วนสำคัญอีกส่วนหนึ่งในระบบเฟสล็อกลูป หน้าที่ของวงจรนี้ได้แก่การควบคุมการล็อก,แคปเจอร์,แบนด์วิด และการตอบสนองค่าทรานเชี้ยนของลูป สำหรับลูปฟิลเตอร์ในที่นี้ก็ คือ วงจรชนิดโลพาสธรรมดา ทำหน้าที่กรองเอาเฉพาะสัญญาณความถี่ต่ำมาควบคุมความถี่ของ VOC ลูปฟิลเตอร์ เป็นตัวกำหนดคุณสมบัติการเปลี่ยนแปลงก่อนเข้าสู่สภาวะล็อกที่เรียกว่า คุณสมบัติชั่วครู่ (Transient) ถ้าเลือกอัตราการขยายลูปจะไม่(Loop Gain) และค่าคงตัวของลูป(Loop Time Constant) ไม่เหมาะสม ความถี่ของเฟสล็อกลูปจะไม่ล็อกและจะเปลี่ยนแปลงอยู่ตลอดเวลา

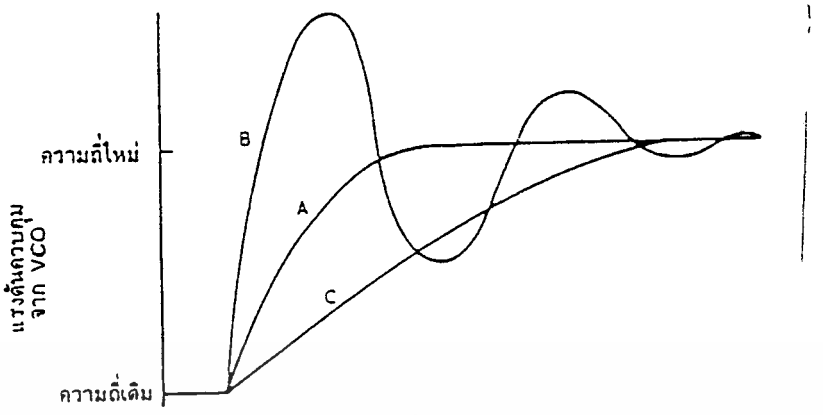
ดังนั้น ค่าคงตัวเวลาของลูปพิลเตอร์จะต้องไม่มากเกินไป เพื่อว่าทุกครั้งที่เปลี่ยน ความถี่เฟสล็อกลูปจะล็อกได้เร็ว โดยไม่มีการสะบัด (Over Shoot) หรือใช้เวลาเปลี่ยน ความถี่อย่างรวดเร็วแต่ค่าคงตัวเวลาก็ไม่ควรจะน้อยเกินไปจนกระทั่งความถี่สั่นหรือไม่นิ่ง ดูรูปที่12 ซึ่งแสดงการเปลี่ยนความถี่ของ VCO จะเห็นว่าเส้นทางเปลี่ยนแปลงแรงดันมี 3 เส้นทาง

เส้นทาง A เป็นเส้นทางคริติคอลแดมป์ (Critical Damp) ใช้เวลาในการเปลี่ยนเข้าสู่ความถี่ใหม่น้อยที่สุด

เส้นทาง B เรียกว่า เส้นทางอันเดอร์แดมป์ (Under Damp) มีการสะบัดเนื่องจากโอเวอร์ชู๊ต

เส้นทาง C เป็นเส้นทางโอเวอร์แดมป์ (Over Damp) ไม่มีโอเวอร์ชู๊ตแต่เวลาที่ใช้ในการเข้าสู่ความถี่ใหม่จะช้า

ดังนั้นจะเห็นว่า เส้นทาง A เป็นเส้นทางที่ดีที่สุดในการออกแบบ ค่าคงตัวเวลาวงจรลูปฟิลเตอร์เพราะใช้เวลาเปลี่ยนความถี่เร็วและไม่มีโอเวอร์ชู๊ต



รูปที่ 11 คุณสมบัติในการเปลี่ยนความถี่ของเฟสล็อกลูป

หน้าที่ของ Low Pass Filter ในเฟสล็อกลูป มีหน้าที่ใหญ่ๆ อยู่ 2 ประการ คือ

1. ลดค่าความคลาดเคลื่อนที่เป็นความถี่สูงที่ออกจากวงจรเปรียบเทียบเฟส (Phase Compatator) โดยการใช้คุณสมบัติการกำจัดสัญญาณรบกวนและเป็นตัวทำให้เกิดค่าแรงดันเฉลี่ย (Average DC Voltage) เพื่อนำไปควบคุมวงจร VCO

2. ทำหน้าที่ควบคุมการทำงานของลูป ซึ่งขึ้นอยู่กับเงื่อนไขต่างๆ ดังนี้

2.1 แคปเจอร์และล็อกเรนจ์

2.2 แบนด์วิด

2.3 การตอบสนองต่อทรานเชี้ยน

เนื่องจาก Low Pass Filter ลดค่าแรงดันคลาดเคลื่อนของความถี่ระหว่างลูปแล้ว ยังเป็นตัวควบคุมการแคปเจอร์โดยตรงและคุณสมบัติต่อผลตอบสนองชั่วขณะของเฟสล็อกลูป

การลดช่วงกว้างของฟิลเตอร์ จะส่งผลไปยังการทำงานของระบบ คือ

1. ขบวนการแคปเจอร์จะช้าลงและฟูลอินไทม์ (Full in Time) เพิ่มขึ้น

2. ช่วงแคปเจอร์จะลดลง

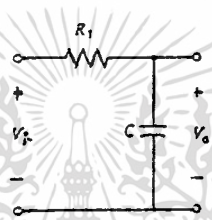
3.คุณสมบัติทางอินเตอร์พีเรนซ์(Interference Rejection)ของเฟสล็อกลูปจะดีขึ้นเพราะค่าแรงดันคลาดเคลื่อนเนื่องจากความถี่ของสัญญาณรบกวนจะถูกลดลงไป

4.ผลตอบสนองชั่วขณะของเฟสล็อกลูปต่อการเปลี่ยนทันทีของสัญญาณเข้าสู่ช่วงความถี่แคปเจอร์จะอยู่ในลักษณะภายใต้การแดมป์

<u>วงจร Low Pass Filter (LPF)</u>

ในระบบเฟสล็อกลูปจะมี Low Pass Filter เป็นส่วนประกอบอยู่เสมอ เราจะกล่าวถึง Low Pass Filter ที่นิยมใช้กันอยู่ 3 แบบดังนี้

1. <u>วงจรกรองความถี่ต่ำอันดับ 1 แบบ RC</u>



รูปที่ 12 แสดง Low Pass Filter อันดับที่ 1 โดยใช้ RC

รูปที่ 13 โดยทั่วไปจะต่ออยู่ระหว่างเฟสดีเทคเตอร์กับ VCO ค่าของความถี่คัทออฟ (Cutoff Frequency, W_{LPF}) สามารถหาได้จากสมการ

$$W_{LPF} = 1/RC$$

ค่าของความถี่ธรรมชาติของลูป (Loop Natural Frequncy, Wn) สามารถหาได้จากความถี่คัทออฟของวงจรกรองความถี่ โดยสมการ

$$Wn = \sqrt{(Kd.Kv.W_{LPF})}$$

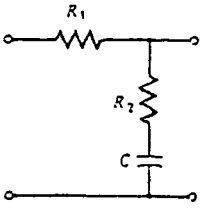
เมื่อ ; Kd คอนเวอร์ชันเกน (Convertion Gain) ของเฟสดีเทคเตอร์ หน่วย(Volt/Sec)

Kv คอนเวอร์ชันเกนของ VCO หน่วย (Rad/Sec/Volt)

เราสามารถหาค่าแดมปิ้งแฟคเตอร์จากสมการ

$$\zeta = N.Wn/(2Kd.Kv)$$

2. <u>วงจรกรองความถี่ต่ำแบบแลก-ลีด</u> (Lag-Lead Circuit)



รูปที่ 13 วงจร แลก-ลีด อันดับหนึ่ง

ต่าความถี่คัทออฟสำหรับวงจรกรองความถี่ชนิดนี้หาได้จากสมการ

$$W_{LPF} = 1/(R1+R2)C$$

และความถี่ธรรมชาติหาได้จากสมการ

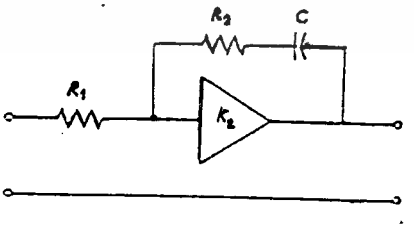
$$Wn = \sqrt{(Kd.Kv/NC(R1+R2))}$$

และแดมปิ้งแฟคเตอร์หาได้จากสมการ

$$\zeta = 0.5Wn(R2C + N/Kd.Kv)$$

3. <u>วงจรพาสซีฟแบบแลก-ลีด</u>

เราสามารถนำมาสร้างเป็นวงจรแอคทีฟฟิลเตอร์



รูปที่ 14 วงจรแอคทีฟฟิลเตอร์

ความถี่คัทออฟหาได้จากสมการ

$$W_{LPF} = 1/R1.C$$

ค่าของลูปความถี่ธรรมชาติ

$$Wn = \sqrt{(Kd.Kv/N.C.R1)}$$

แดมปิ้งแฟคเตอร์ หาจากสมการ

$$\zeta = (Wn.R2.C)/2$$

<u>วงจร Band Pass Filter (BPF)</u>

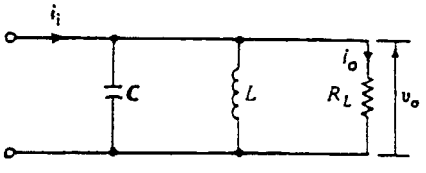
ตัวอย่างง่ายๆ ของตัวกรองผ่านความถี่เป็นช่วง (BPF) คือวงจรเรโซแนนซ์ขนานตามรูป 15 (a) ซึ่งเราต้องใช้ทรานสเฟอร์ฟังก์ชั่นแบบกระแสเนื่องจากโวลเตจของสัญญาณเข้าและสัญญาณออกเท่ากันจะได้

$$H(\omega) = Io/I1$$

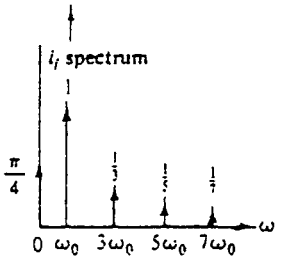
สำหรับรูปคลื่นสี่เหลี่ยมจัตุรัสตามรูปที่ 15 (b) จะได้สเปรคตรัมสัญญาณออกเมื่อวงจรรีโซแนนซ์ที่ฮาร์โมนิคที่สามตามรูป 15 (e) กระแสตรงก็จะถูกตัดออกโดย L ซึ่งทำหน้าที่ลัดวงจรไม่ออกไปที่ตัวภาระ (Load) R และเฟสก์มีค่าเป็นศูนย์ที่ความถี่รีโซแนนซ์ กระแสและโวลเตจออกจะเป็นคลื่นไซน์ที่ความถี่ 3fo

กำหนดให้ $fo = 1/T$

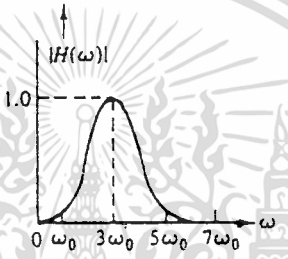
หลักการนี้ใช้ในวงจรเพิ่มความถี่ (Frequency multiplier) ซึ่งความถี่จะเป็นสามเท่าของความถี่เดิม



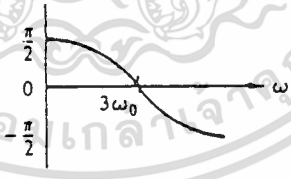
(a) รูปวงจร



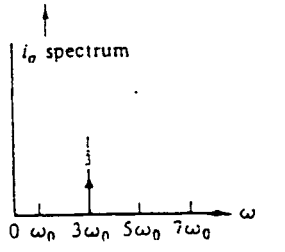
(b) รูปสเปคตรัมความถี่สัญญาณเข้า



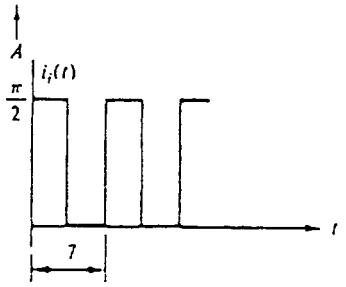
(c) ทรานสเฟอร์ฟังชั่นความสูง



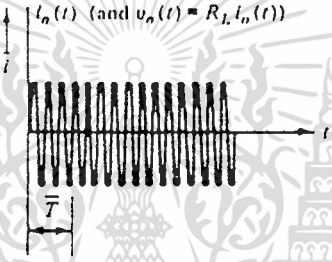
(d) ทรานสเฟอร์ฟังชั่นเฟส



(e) สเปคตรัมความถี่สัญญาณออก



(f) รูปคลื่นสัญญาณเข้า



(g) รูปคลื่นสัญญาณออก

รูปที่ 15 BPF (Band Pass Filter)

Band Pass Filter (BPF) เป็นวงจรกรองผ่านความถี่ที่ยอมให้ความถี่อยู่ในช่วง Low frequency cutoff กับ High frequncy cutoff ผ่านไปได้เท่านั้น โดยส่วนความถี่ที่นอกเหนือจากนี้ไปจะไม่สามารถผ่านได้

ในโครงงานนี้จะใช้วงจรกรองแบบอแกทีฟ (Active Filter) ซึ่งจัดวงจรแบบบัตเตอร์เวอธ (Butterworth filter) สำหรับการออกแบบเราจะกำหนดความถี่ที่ต้องการ คือ ความถี่คัทออฟทั้งทางด้านสูงและด้านต่ำ โดยมีขั้นตอนการออกแบบดังนี้

1. เลือกค่าความถี่ด้านต่ำเท่ากับ 400 KHz ความถี่ด้านสูงเท่ากับ 520 KHz เพราะโครงงานนี้ใช้ความถี่เท่ากับ 450 KHz
2. เลือกค่าตัวเก็บประจุ
   - ทางด้าน Low pass filter ใช้ค่า 60 pF
   - ทางด้าน High pass filter ใช้ค่า 20 pF

3. แทนค่าในสูตร

- ทางด้าน Low pass filter

จากสูตร $R = 1/2\pi f_H C$

$= 1/2\pi*(400x10^{3})*(60x10^{-12})$

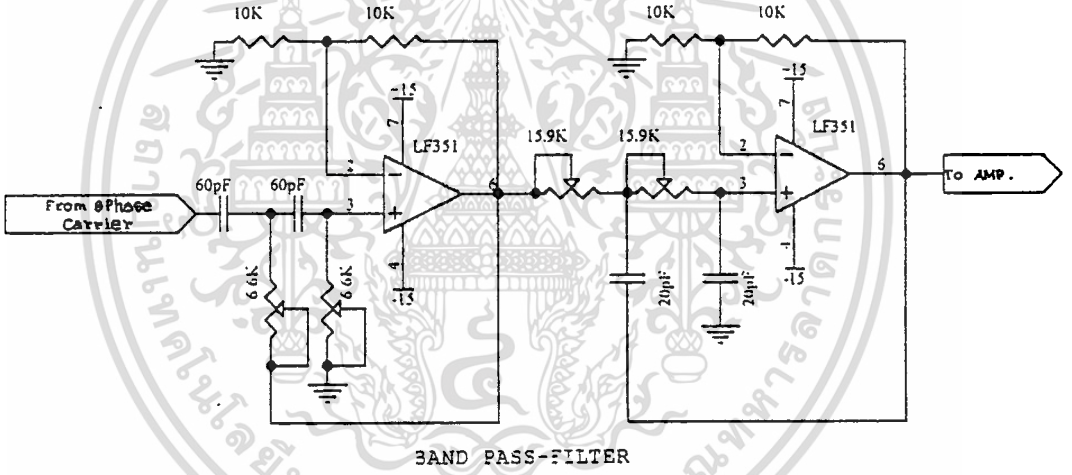
จะได้ $R = 6.63\ k\Omega$

- ทางด้าน High pass filter 2π

จากสูตร $R = 1/2\pi f_H C$

$= 1/2\pi*(500x10^{3})*(20x10^{-12})$

จะได้ $R = 15.91\ k\Omega$



รูปที่ 16 แสดงวงจรกรองผ่านความถี่ (BPF) ที่ใช้งาน

<u>ข้อพิจารณาในการออกแบบลูปฟิลเตอร์</u>

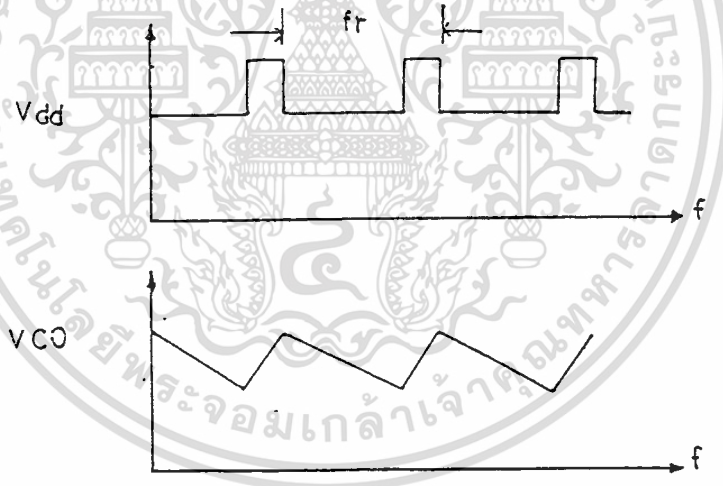
1. เนื่องจากตัวฟิลเตอร์และอินทิเกรเตอร์ที่ใช้ออปแอมป์ มีฟังก์ชันเป็นอินเวอร์ติ้ง ดังนั้นจำเป็นต้องดัดแปลงแก้ไขการกลับเฟสนี้ก่อน เพื่อให้ค่าแรงดันคลาดเคลื่อน(Error Voltage) จากออสซิลเลเตอร์สามารถควบคุม VCO ได้ถูกทิศทางกับความผิดพลาดที่เกิดขึ้น ซึ่งทำได้ง่ายที่สุดโดยการสลับอินพุท Fr และ Fv ที่เฟสดีเทคเตอร์

2. กรณีเฟสดี เทคเตอร์มีเอาท์พุทเป็นดับเบิ้ลเอนด์ Kd = Vdd/2

3. วงจรแอคทีฟฟิลเตอร์อาจจะเกิดการอิ่มตัว ถ้าลูปเกิดการผิดพลาดเชิงเฟสที่เฟสดีเทคเตอร์มีขนาดใหญ่พร้อมๆ กับเกิดทรานเซี้ยนโอเวอร์ชูตขึ้นในลูป กรณีนี้จะเกิด ขึ้นเฉพาะกับลูปที่ใช้เฟสดีเทคเตอร์เป็นชนิดดิจิตอล เนื่องจากเอาท์พุทของเฟสดีเทคเตอร์เปลี่ยนแปลงเป็น 0 หรือ 1 ในทันทีทันใด นอกจากนี้ความถี่อินพุทของฟิลเตอร์มักจะมีค่ามาก

ดังนั้นถ้าอัตราส่วนของ R1/R2 มากกว่า 10 ความถี่นี้จะถูกขยายด้วยอัตราส่วนของ R1/R2 ถ้าสามารถทำได้ควรให้อัตราส่วนนี้มีค่าน้อยที่สุด

4. แรงดันที่ใช้ควบคุม VCO ควรมีส่วนประกอบที่ไม่ใช่ AC น้อยที่สุด ส่วนประกอบที่ไม่ใช่ DC จะทำให้ความถี่เอาท์พุทของ VCO เกิดเอาท์พุทที่ไม่ต้องการ (Spurious Output) แนนไซด์แบนด์ ของความถี่อ้างอิงควรถูกกำจัดไปให้มากที่สุด

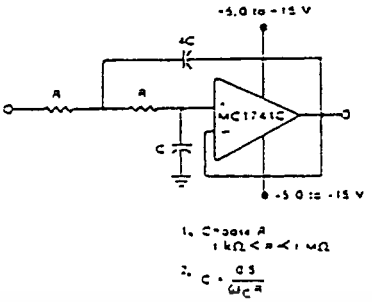


รูปที่ 18 แสดงรูปคลื่นเอาท์พุทของเฟสดีเทคเตอร์และอินทิเกรเตอร์

รูปคลื่นจากรูปที่ 18 ทำให้เกิดไซด์แบนที่สัมพันธ์กับ Carrier ของ VCO ที่สามารถคาดคะเนโดยประมาณได้จาก

$$(\text{Side Band/Carrier}) = V.Kv/2W_R$$

เมื่อ; V คือ ค่าแรงดันยอด (Peak Value) ของความถี่อ้างอิงที่อินพุทของ VCO

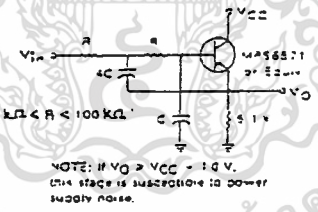


รูปที่ 18 วงจร Low Pass Filter อันดับ 2 โดยใช้ออปแอมป์

Wc = 0.636/RC

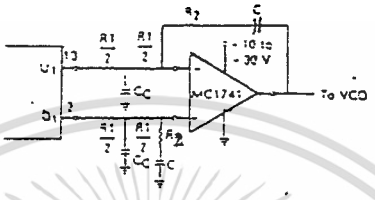
ให้เลือกค่า R ; 1K < R < 1M

ให้เลือกค่า C; 0.5/Wc.R



รูปที่ 19 วงจร LPF อันดับ 2

โดยใช้ทรานซิสเตอร์ต่อแบบ (Emitter Follower) ถ้าเอาท์พุทมีค่าน้อยกว่า Vcc อยู่ 0.1 V วงจรจะมีความไวต่อสัญญาณรบกวนจากแหล่งจ่าย Vcc



รูปที่ 20 วงจรฟิลเตอร์และซัมมิ่งเนทเวอร์ค

2.5 วงจรผลิตความถี่ควบคุมด้วยแรงดัน (Voltage Control Oscillator-VCO)

คุณสมบัติหลักของ VCO ที่ใช้ในเฟสล็อกลูป เราพิจารณาได้ดังนี้

1. <u>การเบี่ยงเบนของความถี่</u> (Frequency Deviation)

   จุดสูงสุดของเคปเจอร์เรนจ์จะเท่ากับการขยายของลูปเปิด (Open Loop Gain)

2. <u>เสถียรภาพทางความถี่</u> (Frequency Stability)

   การมีเสถียรภาพทางความถี่มีความจำเป็นอย่างยิ่งสำหรับวงจรสังเคราะห์ความถี่ ความไวของการมอดดูเลท (Modulation Sensitivity) ควรจะมีค่าสูง

3. <u>การตอบสนอง</u> (Response)

   VCO ควรมีการตอบสนองสัญญาณได้ดีและไม่ควรให้มีผลต่อคุณสมบัติทางด้านเสถียรภาพของลูป

4. <u>คุณสมบัติของความถี่และแรงดัน</u> (Frequency Voltage Charactoristic)

   VCO จะต้องมีอัตราส่วนของความถี่ต่อแรงดัน (F/V) ที่มีความเป็นเชิงเส้น (Linear)

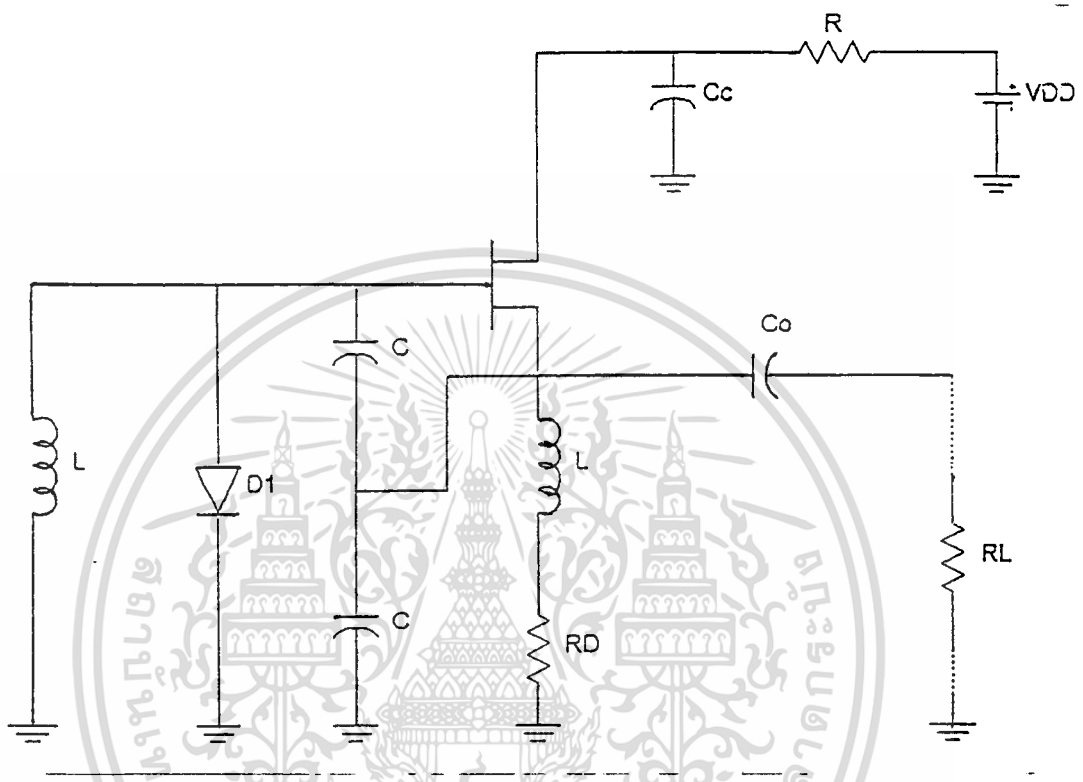
5. Spectral Purity

ในการประยุกต์ใช้งาน เช่น การสังเคราะห์ความถี่แบบอนาล็อก วงจรผลิตความถี่ควบคุมโดยแรงดัน ควรจะมีสัญญาณเอาท์พุทที่บริสุทธิ์คือถ้าเป็นคลื่นรูปซายน์ควรจะเป็นคลื่นที่คงที่สม่ำเสมอ

ในการออกแบบเฟสล็อกลูป ออสซิลเลเตอร์ที่ควบคุมแรงดันมักจะเป็นส่วนที่จะต้องพิจารณามากที่สุด เพราะว่ามีลักษณะพิเศษของระบบ อย่างเช่น เสถียรภาพของระบบและเสถียรภาพของความถี่ รวมทั้งการดีมอดดูเลทคลื่นเอฟเอ็มตามปกติแล้วจะขึ้นอยู่กับ VCO เพื่อให้เกิดความคล่องตัวมากที่สุด VCO จะต้องมีคุณสมบัติดังนี้

6. ลักษณะการเปลี่ยนแรงดันเป็นความถี่เชิงเส้น
7. เสถียรภาพของความถี่ที่ดี
8. สามารถใช้กับความถี่สูงได้
9. อัตราการขยายสูง
10. พิสัยการติดตามกว้าง
11. การติดตั้งความถี่กระทำได้ง่าย

การวิเคราะห์ที่กล่าวมาจะแสดงให้เห็นเฉพาะวงจรเชิง AC ในการใช้งานจริงก็ต้องมีการไบอัส DC เพื่อให้วงจรทำงานได้เอาท์พุทของวงจรสามารถคัปปลิ้งออกได้ทั้งจุด 1,2 และ 3 ดังรูป 22 ซึ่งโหมดจะมีผลต่อวงจรเช่นกัน การวิเคราะห์ได้แสดงถึงความถี่ของสัญญาณเท่านั้น การออกแบบจะกำหนดให้ลูปเกนของวงจรที่วิเคราะห์แบบเชิงเส้นมีมากเป็น 3-4 เท่า เมื่อวงจรเกิดการออสซิลเลทแล้ว gm ของอุปกรณ์แอคทีฟจะลดลงจนถึงสภาวะเสถียร ซึ่งในขณะนี้ขนาดของสัญญาณจะคงที่ โดยปกติกำลังของสัญญาณที่ได้มักจะไม่เกิน 25% ของกำลังงาน DC ที่วงจรดึงจากแหล่งจ่าย อุปกรณ์แอคทีฟที่ใช้สามารถนำ FET หรือ MOS-FET มาใช้งานก็จะทำงานได้ผลดีเช่นเดียวกัน



รูปที่ 21 แสดงวงจรโคลพิทท์ออสซิลเลเตอร์ซึ่งใช้ FET

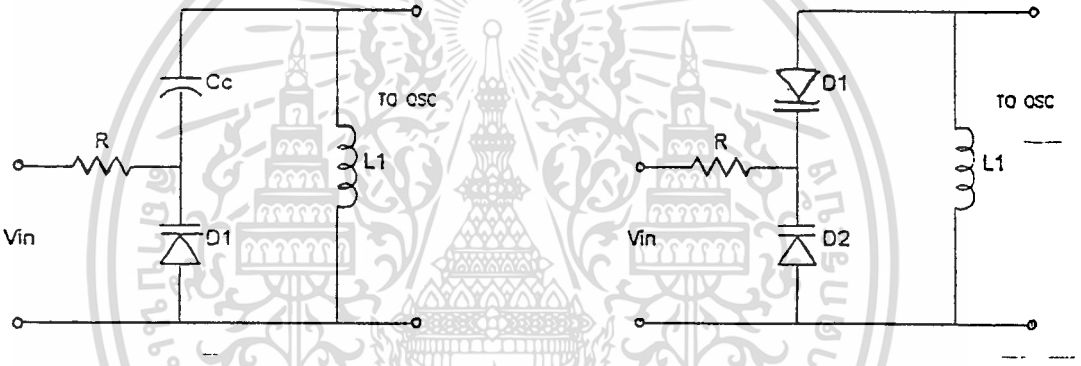
วงจรที่ใช้ FET แสดงในรูป 22

เมื่อ ; Rd ทำหน้าที่สร้างแรงดันไบอัสให้กับ FET และ R.F Choke ทำให้ Rd มีค่าอิมพีแดนซ์สูงมากที่ความถี่ใช้งาน ซึ่งจะทำให้ Rd ไม่มีผลต่อวงจร

D1 ทำหน้าที่จำกัดขนาดแรงดัน AC ที่เกิดจากการออสซิลเลทไม่ให้มีขนาดใหญ่จนเกินไปจนทำให้รอยต่อระหว่างขาเกทและซอร์สของ FET นำกระแสได้ ซึ่งไม่จำเป็นต้องมีในกรณีที่แรงดัน AC มีขนาดเล็กหรือในกรณีที่เป็น MOS-FET วงจรสามารถรับ

โหลดที่เป็นค่าความต้านทานสูงๆ ได้เท่านั้น สำหรับโหลดที่มีค่าความต้านทานต่ำๆ ค่าของ Co จะเล็กลง เพื่อให้ rco ที่ความถี่ใช้งานมีค่าสูงจนทำให้โหลด ไม่มีผลต่อวงจรและ R กับ C ทำหน้าที่ บายพาสแรงดัน AC

วงจร VCO เป็นส่วนประกอบที่สำคัญของระบบเฟสล็อกลูป ความถี่เอาท์พุทของ VCO จะแปรตามแรงดันอินพุทที่ควบคุมซึ่งจะใช้วิธีเปลี่ยนแรงดันไบอัสให้กับวาริแคป



รูปที่ 22 แสดงการใช้วาริแคปกับวงจรออสซิลเลเตอร์

รูปที่ 22 A คาปาซิเตอร์ จะมีค่าอิมพีแดนซ์ต่ำที่ความถี่ใช้งาน ทำหน้าที่แยกแรงดัน DC ออกจากวงจรรีโซแนนซ์

วงจรในรูปที่ 22 มีข้อจำกัดที่แรงดัน AC คร่อม D1 ต้องมีค่าน้อย (น้อยกว่า 600 mV rms) ไม่เช่นนั้น D1 อาจจะทำการเรคติไฟร์แรงดันคร่อม L1 ซึ่งจะทำให้ความเพี้ยน เกิดขึ้นกับความถี่ที่ใช้งาน

วิธีแก้ไขทำได้โดยรูปที่ 22 B ไดโอด D1,D2 ต่อกลับกัน ทำให้แก้ปัญหาการเรคติไฟร์ แต่ค่าความจุ (Capacitance) รวมของ D1,D2 จะลดลงครึ่งหนึ่งและ D1,D2 ต้องมีคุณสมบัติใกล้เคียงกันมากที่สุดหรืออาจจะรวมกันอยู่ในตัวถังเดียวกัน

เมื่อใช้วาริแคปร่วมกับวงจรกับวงจรออสซิลเลเตอร์ ในรูปที่ 22 ค่าความจุของวาริแคปต้องนำไปรวมกับ C1,C2 .และในทางปฏิบัติกรณีที่ FET ก็รวมค่าของ CISS (COMMON SOURCE INPUT CAPACITANCE) เข้ากับ C1 และ COSS (COMMON SOURCE OUTPUT CAPACITANCE) เข้ากับ C2 โดยกำหนดว่าค่าจริงของ Y11 และ Y22 ไม่มีผลกับวงจร

2.6 วงจรหารแบบสองโมดูลัส (PRESCALE 2 MODULUS)

ส่วนสำคัญของวงจรหารแบบนี้คือ ไอซีตระกูล ECL ซึ่งมีความสามารถในการทำงานที่ความถี่สูง การทำการหารล่วงหน้าหรือ PRESCALE ก่อน หมายถึง มีการทำงาน ในลักษณะที่หารได้ 2 ครั้งด้วยค่า 2 ค่าสลับกันในตัวไอซีเดียว เรานิยมเรียกไอซีตระกูล ECL ว่า พรีสเกลเลอร์ชนิดสองโมดูลัส (Dual Modulus Prescaler)

พรีสเกลเลอร์นี้สามารถหารความถี่ด้วยตัวเลข 2 ตัว ซึ่งต่างกันอยู่ 1 เช่น หาร 10 หรือ 11 เรียกว่า 10/11 พรีสเกลเลอร์

ในโครงงานนี้ใช้ MC 12017 ซึ่งเป็น 64/65 พรีสเกลเลอร์ เอาท์พุทของพรีสเกลเลอร์จะป้อนให้แก่วงจรหาร 2 ตัว โดยตัวหนึ่งเป็นตัวนับหลัก (Main Counter) ส่วนอีกตัวเป็นตัวเสริม (Auxiliary Counter)

ตัวหารเสริมจะเป็นตัวบังคับให้พรีสเกลเลอร์หารด้วยตัวหาร (Modulus) ตัวใดคือหารด้วย 64 หรือ 65 เช่น ป้อนข้อมูล (ความถี่) หรือพรีเซตัวเลขให้ตัวนับ (หาร) เสริม และในขณะนี้ ECL พรีสเกลเลอร์ใช้ 65 เป็นตัวหาร เมื่อเคาน์เตอร์เสริมหยุดนับจึงส่งคำสั่งไปบังคับให้พรีสเกลเลอร์เปลี่ยนตัวหารเป็น 64

ตัวนับหารก็เช่นกัน จะค่อยๆ นับถอยหลังไปเรื่อยๆ จนเป็นศูนย์ เมื่อตัวนับหลักและตัวนับเสริมนับถึงศูนย์เมื่อใดทั้งคู่จะถูกพรีเซตด้วยตัวเลขข้อมูล (ความถี่) เนื่องจากตัวนับเสริมจะต้องนับถึงศูนย์ก่อน

ดังนั้น ตัวเลขที่พรีเซตให้เคาน์เตอร์เสริมจะต้องน้อยกว่าตัวเลขที่พรีเซตให้เคาน์เตอร์หลัก

สมมุติว่าตัวเลขที่พรีเซตเป็น M ให้ตัวนับหลัก และ A ให้แก่ตัวนับเสริม เริ่มแรก ให้พรีสเกลเลอร์อยู่ในสภาวะหาร 65 ซึ่งยังคงหารตัวหาร 65 ไปจนกว่าตัวนับเสริมจะนับเป็นศูนย์นั่นคือ เวลาที่ใช้ในการนับของตัวนับเสริมเป็นศูนย์คิดเป็นจำนวนไซเคิล(ของ VCO) ที่ผ่านไปเท่ากับ 65 คูณด้วย A ไซเคิล

หลังจากนั้นพรีสเกลเลอร์จะถูกบังคับให้เปลี่ยนตัวหารเป็น 64 (โดยตัวนับเสริม) ในขณะนี้ตัวนับหลักนับผ่านไป A แล้ว (พร้อมตัวนับเสริม) เช่นกัน ยังเหลืออยู่อีก (M-A) ไซเกิลก่อนที่นับเป็นศูนย์ นั่นคือ จะต้องใช้เวลาในการนับตัวนับศูนย์ต่อไปอีกคิดเป็นจำนวนไซเกิล (ของ VCO) ที่ผ่านไปเท่ากับ 64 คูณด้วย (M-A)

ฉะนั้นรวมเวลาที่ใช้จึงเป็นผลรวมของเวลาทั้งสองข้างต้น

$$\text{VCO ไซเกิล} = 65A + 64(M-A)$$

$$= 64A + A$$

ความถี่ของ VCO จะเท่ากับ (64M + A) เท่า ของความถี่อ้างอิง หรือ

$$Fsynth = Fref\ (64M + A)$$

ผลของตัวเลข M มีผลต่อความถี่ Fsynth มากกว่าตัวเลข A อยู่ 64 เท่า นอกจากนี้ตัวหาร 64(M+A) ก็ไม่สามารถหาได้ครบทุกค่า เนื่องจากมีข้อจำกัดตรงที่ M จะต้องมากกว่า (หรือเท่ากับ) A

กรณีที่พรีสเกลเลอร์มีค่าเป็น P และ N ตัวหารจะเป็นดังนี้

ตัวหารของระบบสังเคราะห์ความถี่ = PM + A

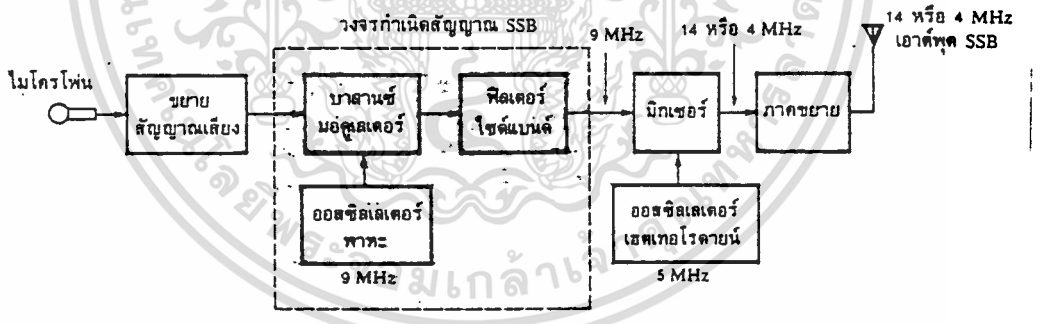
ตัวหารต่ำสุด = P(P-1)

ตัวหารสูงสุด = Pmmax + Amax

2.7 การแปลงความถี่ (Frequency conversion)

ไม่ว่าสัญญาณ SSB ที่กำเนิดขึ้นมาโดยวิธีใดก็ตาม ความถี่ของสัญญาณจะมีค่าตายตัวซึ่งไม่ตรงตามความถี่ที่เราจะใช้งาน ดังนั้นเราจึงจำเป็นต้องแปลงความถี่ (convert) ของสัญญาณ SSB ให้เป็นความถี่ใช้งานที่ต้องการ รูปที่ 24 แสดงวิธีการแปลงความถี่ เทคนิคการแปลงความถี่นี้บางทีเรียกว่า "การผสมคลื่น" (mixing) หรือ "เฮตเทอโรดายน์" (heterodyne)

จะเห็นว่าหลังจากสัญญาณเสียงมอดูเลต (บนพาหะ) เป็นสัญญาณ SSB แล้ว ความถี่กลางจะได้ประมาณ 9 เมกกะเฮิรตซ์ ซึ่งเมื่อป้อนให้แก่วงจรมิกเซอร์เพื่อผสมกับสัญญาณออสซิลเลเตอร์ 5 เมกะเฮิรตซ์จะได้เป็นสัญญาณที่มีความถี่ประมาณ 14 หรือ 4 เมกะเฮิรตซ์ (เรามักเลือกใช้ความถี่ใดความถี่หนึ่งเท่านั้น) ถ้าออสซิลเลเตอร์ 5 เมกะเฮิรตซ์มีค่าความถี่ปรับค่าได้ ความถี่ใช้งานก็จะสามารถเลือกได้ตามต้องการ



รูปที่ 23 แผนผังของเครื่องส่ง SSB

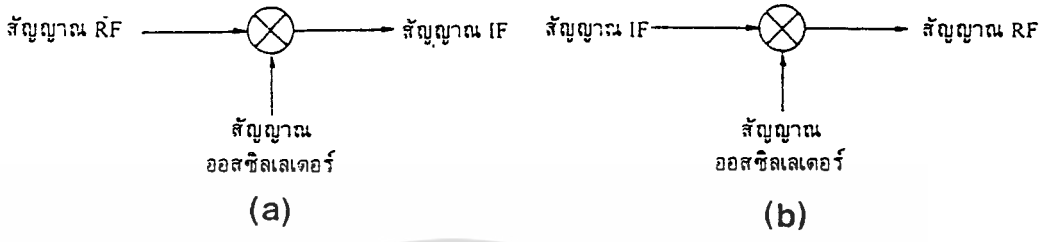
ลองพิจารณารูปที่ 23 อีกครั้ง วงจรออสซิลเลเตอร์พาหะ (carrier oscillator) กำเนิดสัญญาณ 9 เมกกะเฮิรตซ์ ซึ่งอยู่ในย่านความถี่ผ่านของไซด์แบนด์ฟิลเตอร์ สัญญาณไซด์แบนด์ข้างใดข้างหนึ่ง (USB หรือ LSB) จะผ่านไปสู่วงจรมิกเซอร์ ความถี่ของไซด์แบนด์จะอยู่ประมาณใกล้เคียงกับ 9 เมกกะเฮิรตซ์ วงจรมิกเซอร์นี้เป็นวงจรที่ทำงานแบบนอนลิเนียร์ (อาจจะใช้วงจรบาลานซ์มอดูเลเตอร์ก็ได้)

ออสซิลเลเตอร์เฮตเทอโรดายน์ในที่นี้กำเนิดความถี่ 5 เมกกะเฮิรตซ์ อาจจะมีความถี่ปรับได้หรือมีสวิทซ์เลือกความถี่เพื่อปรับให้ได้ตามความถี่ที่ต้องการ ในตัวอย่างนี้สัญญาณ 9 เมกกะเฮิรตซ์ กับ 5 เมกกะเฮิรตซ์จะผสมกันในวงจรมิกเซอร์ เกิดผลลัพธ์ขึ้นเป็นสัญญาณความถี่ผลรวม 14 เมกกะเฮิรตซ์ กับสัญญาณความถี่ผลต่าง 4 เมกกะเฮิรตซ์ จากนั้นสัญญาณทั้งคู่ที่ต้องการใช้งานขยายออกไปสู่สายอากาศ วงจรขยายกำลังจะต้องจูนไว้ที่ความถี่เดียวกับความถี่ของแทงค์ กำลังของสัญญาณ SSB จากวงจรมิกเซอร์มีค่าน้อย เมื่อขยายที่วงจรขยายกำลังก็จะได้กำลังส่งตามที่ต้องการ สรุปแล้วสัญญาณนิยมกำเนิดที่ระดับสัญญาณต่างๆ และมีความถี่ตายตัว จากนั้นจึงแปลงความถี่ให้ได้เป็นความถี่ที่ต้องการโดยกรรมวิธีเฮตเทอโรดายน์หรือมิกซ์ (หรือผสมคลื่น)

2.7.1 วงจรมิกเซอร์

วงจรมิกเซอร์ แบ่งออกได้เป็น 2 ประเภทคือ ประเภทแอคทีฟ (active) ใช้ทรานซิสเตอร์หรือไอซีรวมทั้งอุปกรณ์อื่นๆ ที่ให้อัตราการขยาย (ในการผสมคลื่น) และประเภทพาสซีฟ (passive) ใช้ไดโอดซึ่งไม่มีการขยายสัญญาณ

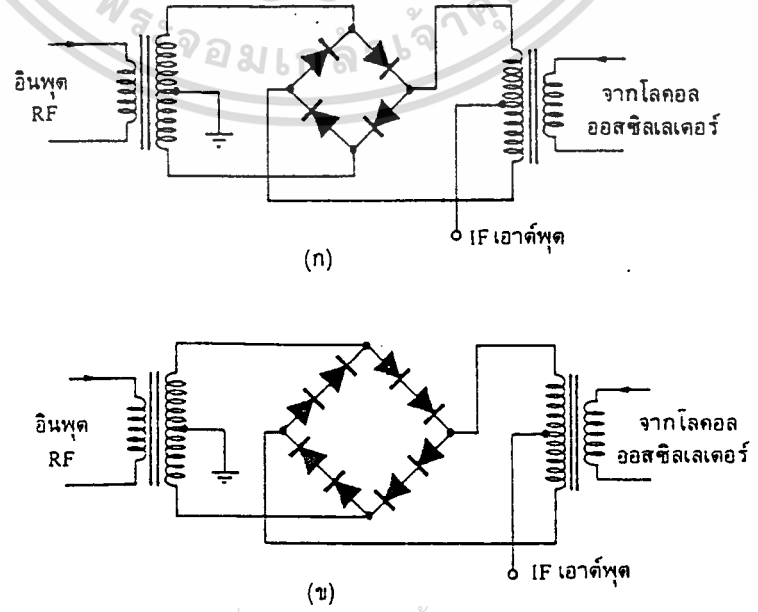
นอกจากนี้ เราอาจแบ่งวงจรมิกเซอร์ได้เป็น 2 ประเภทคือ แบบสมดุลหรือบาลานซ์กับแบบไม่สมดุลหรืออันบาลานซ์ วงจรมิกเซอร์แบบสมดุลนี้ เราต่อวงจรให้ขั้วอินพุตของวงจรมิกเซอร์ไม่เกิดปฏิกิริยาซึ่งกันและกัน (สัญญาณไม่เล็ดลอดระหว่างขั้ว) คุณสมบัตินี้เราเรียกว่า "การแยกระหว่างขั้ว" หรือไอโซเลชั่น (isolation) คงจำได้ว่าขั้วอินพุทของวงจรมิกเซอร์มี 2 ขั้ว คือ สัญญาณ RF(หรือ IF) กับสัญญาณออสซิลเลเตอร์ และมีขั้วเอาท์พุต 1 ขั้ว คือสัญญาณ IF (หรือ RF) ดูรูปที่ 24 ลองพิจารณาในกรณีของภาคเครื่องรับจะเห็นว่าการแยกระหว่างขั้ว RF และขั้วออสซิลเลเตอร์จะช่วยมิให้สัญญาณออสซิลเลเตอร์ย้อนกลับออกสู่สายอากาศแผ่กระจายคลื่นออกไปได้ และการแยกระหว่างขั้ว RF กับขั้ว IF จะช่วยมิให้สัญญาณที่มีความถี่พอดีตรงกับความถี่ IF เล็ดลอดเข้าไปสู่วงจรขยาย IF ในกรณีของภาคส่งก็พิจารณาทำนองเดียวกัน



รูปที่ 24 (a) ตัวอย่างมิกเซอร์ของภาคเครื่องรับ, (b) มิกเซอร์ของภาคเครื่องส่ง

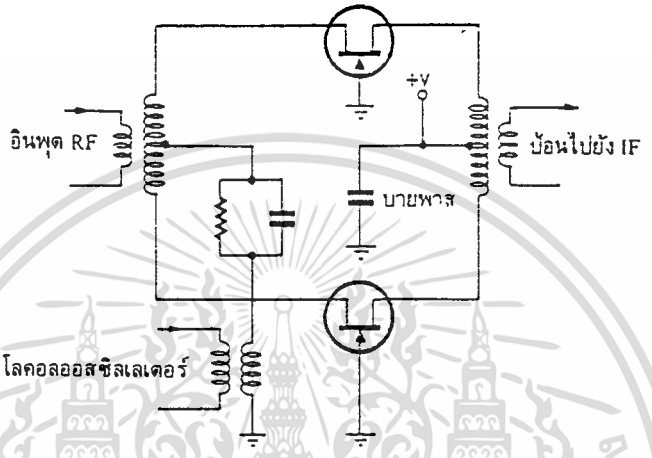
สำหรับวงจรมิกเซอร์แบบไม่สมดุลนั้น มีคุณสมบัติการแยกระหว่างขั้วไม่ดีเหมือนกับแบบสมดุลถ้าต้องการให้มีการแยกสัญญาณดีต้องใช้ฟิลเตอร์ช่วยในการกรองสัญญาณต่างหากอีก

ดูตัวอย่างวงจรบาลานซ์มิกเซอร์ รูปที่ 24 ซึ่งใช้ในภาคเครื่องรับ ไดโอดที่ใช้ในภาคเครื่องรับ ไดโอดที่ใช้ต้องมีคุณสมบัติเหมือนกัน และหม้อแปลงก็ต้องสมมาตรกับจุดกลางวงจรในรูปที่ 24 (b) จะแตกต่างจาก รูปที่ 24 (a) ตรงที่ใช้จำนวนไดโอดเพิ่มอีก 4 ตัว เพื่อให้เหมาะกับการผสมสัญญาณที่มีความแรงมากกว่า (เช่นมิกเซอร์ในภาคเครื่องส่ง)

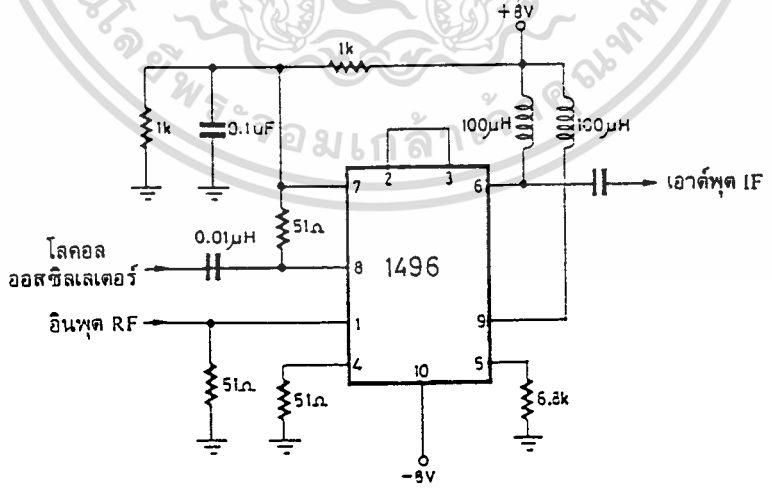


รูปที่ 25 วงจรบาลานซ์มิกเซอร์ชนิดพาสซีฟ

ในรูปที่ 25 แสดงวงจรบาลานซ์มิกเซอร์แบบแอกตีฟ ซึ่งให้อัตราขยายในการผสมคลื่น (แทนที่จะให้อัตราการสูญเสียในการผสมคลื่นเหมือนกับแบบพาสซีฟ) และรูปที่24 เป็นวงจรบาลานซ์มิกเซอร์อีกแบบหนึ่งที่ใช้ไอซี

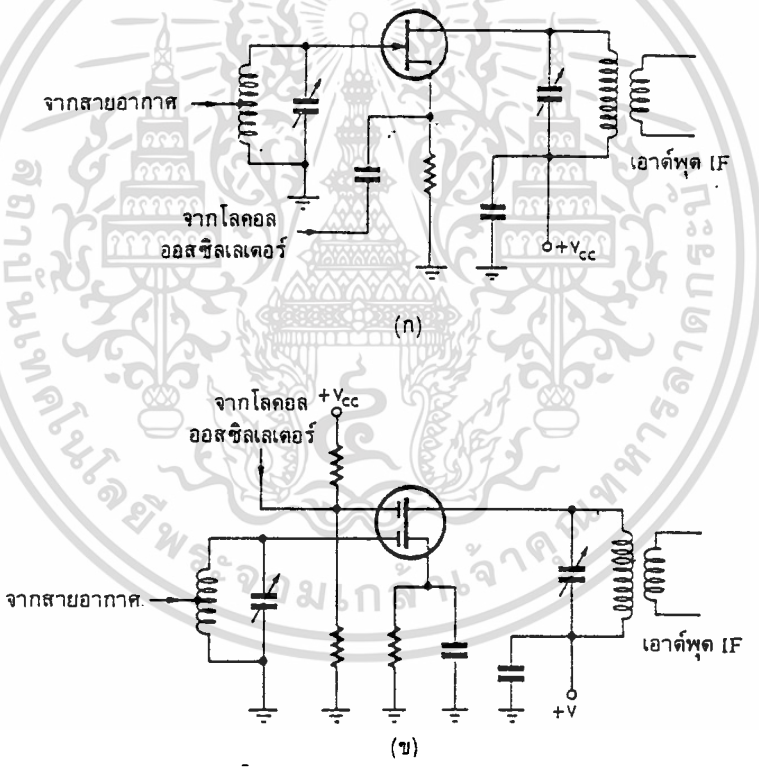


รูปที่ 26 วงจรบาลานซ์มิกเซอร์ ชนิดแอ็กตีฟ แบบใช้ FET



รูปที่ 27 วงจรบาลานซ์มิกเซอร์ ชนิดแอกตีฟ แบบใช้ IC

วงจรมิกเซอร์แบบไม่สมดุลแสดงไว้ในรูปที่ 28 (ข) ซึ่งใช้ MOSFET คุณสมบัติของวงจร คือ มีการแยกระหว่างขั้วออสซิลเลเตอร์กับสายอากาศค่อนข้างดี แต่ระหว่างขั้ว RF และขั้ว IF ไม่ค่อยดี เราจำเป็นต้องใช้ฟิลเตอร์ช่วยกรองความถี่เพื่อกำจัดสัญญาณ RF มิให้เล็ดลอดเข้าสู่ขั้ว IF ได้ ในรูปที่ 28 (ก) เราใช้ JFET โดยป้อนสัญญาณออสซิลเลเตอร์เข้าทางซอส และสัญญาณ RF เข้าทางเกตซึ่งคุณสมบัติการแยกระหว่างขั้ว RF กับขั้วออสซิลเลเตอร์จะไม่ค่อยดี



รูปที่ 28 วงจรมิกเซอร์แบบไม่สมดุล

บทที่ 3

การออกโปรแกรมและทฤษฎีการใช้งาน 8255

3.1 ขั้นตอนการออกแบบโปรแกรม

1. คิดรูปแบบการติดต่อการใช้งานระหว่าง sofe ware และ hard ware ภาษาที่ใช้

2. เขียน Flow chart การทำงานของโปรแกรมคร่าวๆ

3. เขียน Flow chart ที่ละเอียดขึ้น พร้อมทั้งเขียนโปรแกรมไปด้วย

4.ทดสอบโปรแกรมการติดต่อต่างๆให้ได้ พร้อมทั้งเพิ่มเติมรายละเอียดตามต้องการ

ใน Project ชิ้นนี้ต้องการติดต่อระหว่าง software กับ hardware ผ่านทาง computer โดยการมีเมนูให้ติดต่อระหว่าง sofe ware กับ hard ware ผ่านทางจอมอนิเตอร์ ที่เลือกใช้ computer ในการติดต่อ เพราะปัจจุบันมีการใช้ computer อย่างกว้างขวางในทุกๆวงการ ดังนั้นจึง ทำการพัฒนาโปรแกรมลงบน computer เพื่อที่จะให้ง่ายและสะดวกในการใช้งาน ซึ่งจะเห็นได้ว่าชีวิตประจำวันในปัจจุบันนี้จะมี computer เข้ามาเกี่ยวข้องอยู่เสมอ และในตอนนี้ก็มีการพัฒนาโปรแกรมใน computer ให้ใช้งานได้หลายอย่างขึ้นไม่ว่าจะเป็น การแต่งเพลง ร้อง-ฟังเพลง เป็น SW control อุปกรณ์ต่างๆ และ internet เป็นต้น

เพื่อจะให้ง่ายต่อการเขียนเมนูทางจอภาพและสามารถติดต่อ port ต่างๆได้สะดวก ใน project นี้เลือกใช้ ภาษา C เป็น compiler ในการเขียนโปรแกรม เพราะเป็นภาษาที่มีการใช้งานอย่างกว้างขวาง มีข้อมูลให้ศึกษาการทำงานได้มาก ซึ่งภาษามีคุณลักษณะพิเศษและมีข้อจำกัดดังนี้

---- คุณลักษณะพิเศษของภาษาซี

1. C เป็นภาษาที่มีการโปรแกรมแบบ procedure ที่มีประสิทธิภาพมากที่สุด เพราะโปรแกรมสั่งงานได้เร็วกว่าทุกภาษา (ยกเว้นภาษาแอสเซมบลี่) ด้วยเหตุนี้โปรแกรมสมัยปัจจุบันมักจะเขียนด้วนภาษาซี และถ้าส่วนไหนต้องการความเร็วเพิ่มขึ้น จะใช้แอสเซมบลี่เขียนในส่วนนั้นๆ ของโปรแกรม

2. C เป็นภาษาที่ใช้ติต่อกับส่วนต่างๆ ของ computer ได้ดีกว่าภาษาอื่น

3. ถึงแม้ว่าตัวแปลงภาษาซี ให้เป็นภาษาเครื่องจะมีอยู่มาก แต่ทั้งหมดก็ยึดเอามาตรฐานของ ANSI เป็นหลัก จะมีต่างกันก็เรื่องของ Graphics เท่านั้น (ANSI ย่อมาจาก American National Standards Institute)

4. C เป็นภาษาที่มี Protabillity สูง ดังนั้นโปรแกรมที่เขียนด้วย C จึงง่ายต่อการที่จะนำไปใช้งานกับเครื่องชนิดต่างชนิกกัน เช่นจากPersoonal Computer ไปสู่ Mainframe หรือจาก MS-DOS ไปสู่ UNIX เป็นต้น

5. C เป็นภาษาที่สั่งงาน computer ได้แทบทุกอย่าง เท่าที่ภาษาจะสั่งงานทำได้ โดยอาจใช้ C ในการเขียน Operating System, เขียนตัวแปลงภาษา (Compiler) ของภาษาอื่นๆ, เขียนโปรแกรมเกี่ยวกับ (Moder) หรือเกี่ยวกับปัญญาประดิษฐ์ (Artificial Intelligence)

6. C เป็นภาษาที่มีโปรแกรมสนับสนุนมากและราคาต่ำ

7. C เป็นภาษาที่กำหนดรูปแบบของข้อมูลได้อย่างกว้างขวาง

ข้อจำกัดของภาษาซี

1. เมื่อเปรียบเทียบกับภาษาอื่น C จะเป็นภาษาที่อยู่ในระดับต่ำกว่า ดังนั้นโปรแกรมภาษาซี จึงยาวกว่าภาษาอื่นๆ โดยปรกติเวลาจะเขียนโปรแกรมสำเร็จรูปจริงๆอาจต้องซื้อโปรแกรมสนับสนุนมาช่วยเพื่อทำให้โปรแกรมสมบูรณ์ยิ่งขึ้น นั่นคือจะต้องเสียค่าใช้จ่ายมากขึ้น

2. C เป็นภาษาที่มีการสั่งงานเป็นแบบ procedure นั้นคือก่อนที่จะจะทำการแก้ปัญหาใดๆจะต้องรู้ขั้นตอนการแก้ปัญญานั้นก่อน

3. C เป็นโปรแกรมเป็นภาษาชั้นสูงที่ตัวโปรแกรมอ่านเข้าใจได้ยากกว่าภาษาอื่น

4. ถึงแม้ว่า C จะถูกทำให้อยู่ในมาตราฐานเดียวกัน แต่ก็ยังมีสิงที่แตกต่างกันอยู่มาก

ส่วนประกอบในโปรแกรมของภาษาซี

โปรแกรมภาษาซีประกอบด้วยส่วนสำคัญ 3 ส่วน คือ (ตย.ด้านล่าง)

1. Comment lines คือ ส่วนของข้อความที่สามารถเขียนแทรกเข้าไว้ในส่วนต่างๆของโปรแกรมเพื่อประโยชน์ในการช่วยเตือนความจำหรืออธิบายว่าส่วนนั้นๆ ของโปรแกรมทำอะไร โดยเริ่มต้นของ comment line ต้องเป็นเครื่องหมาย /* และปิดท้ายด้วย

เครื่องหมาย */ ที่อาจเป็นบรรทัดเดียวหรือหลายบรรทัดก็ได้ ส่วนของ comment line นี้อาจมีแทรกอยู่ส่วนไหนของ โปรแกรมก็ได้

2. Compiler Directive คือ ส่วนที่ใช้บอก Compiler ถึงการรวมไฟล์ต่างๆเข้ามาร่วมในการคอมไพล์ด้วย เช่น #include <sidio.h> จะเป็นการบอก Compiler ว่าคอมไพล์โปรแกรมนี้ให้นำหน้าข้อมูลในไฟล์ stdio.h เข้ามารวมกับโปรแกรมนี้ก่อน ที่เราต้องทำเช่นนี้เพราะฟังก์ชั่น scanf และ printf ได้ถูกกำหนดในนิยาม (prototype) ไว้ใน stdio.h

3. ส่วนของ Body จะมีว่า main( ) เพื่อบอกจุดเริ่มต้นของโปรแกรมโดยมีเครื่องหมายปีกกาเปิด (open brace) " { " และเครื่องหมายปีกกาปิด (close brace) " } " เป็นตัวหัวท้ายของโปรแกรม นอกจากนั้นภายในตัวโปรแกรมยังประกอบด้วย

ก. Declaration

คือ ส่วนที่บอกคอมพิวเตอร์ว่า ตัวแปรที่ใช้ในโปรแกรมนี้จะเป็นตัวแปรใดบ้าง ชนิดไหน เช่นในโปรแกรมตัวอย่างมรการบอกว่า จะมีการใช้ตัวแปรx, y และ z โดยตัวแปรทั้งสาม เป็นตัวแปรที่ใช้เก็บเลขทศนิยม (floating point number)

ข. Input

คือ ส่วนที่ทำการอ่านค่าของตัวแปรเข้ามา ซึ่งอาจอ่านเข้ามาทางคีย์บอร์ดหรืออ่านเข้ามาทางไฟล์ข้อมูลของแผ่นดีสเกตท์ ก็ได้

ค. Computation หรือ Assignment

คือ ส่วนที่ใช้ในการคำนวณรวมไปถึงการกำหนดค่าให้กับตัวแปรด้วย

ง. Output

คือ การแสดงผลซึ่งอาจแสดงออกทางจอภาพหรือ นำผลเก็บไว้ในไฟล์ข้อมูลก็ได้

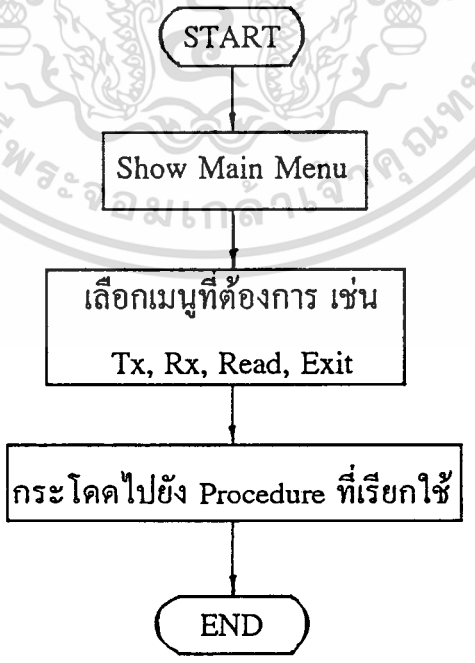
ตัวอย่าง โปรแกรมภาษาซี

```
/* My first Program */              -Comment
#include (stdio.h)                   -Compiler Directive
main()                              -
{                                   -
    float x,y,z;                    - Declarations
    scanf ("%f %f",&x, &y);         - Inputs
    z = x+y;                        - Computations
    printf("The sum is %f\n",z);    - Output
}                                   -
```

ข้อสังเกต แต่ละคำสั่งต้องลงท้ายด้วยเครื่องหมาย sami-colon (;) เสมอ

ซึ่งในที่นี้จะไม่กล่าวถึงวิธีการเขียนโปรแกรมโดยละเอียด ถ้าสนใจจะศึกษาให้หาข้อมูลได้ตามหนังสืออ้างอิงด้านหลัง

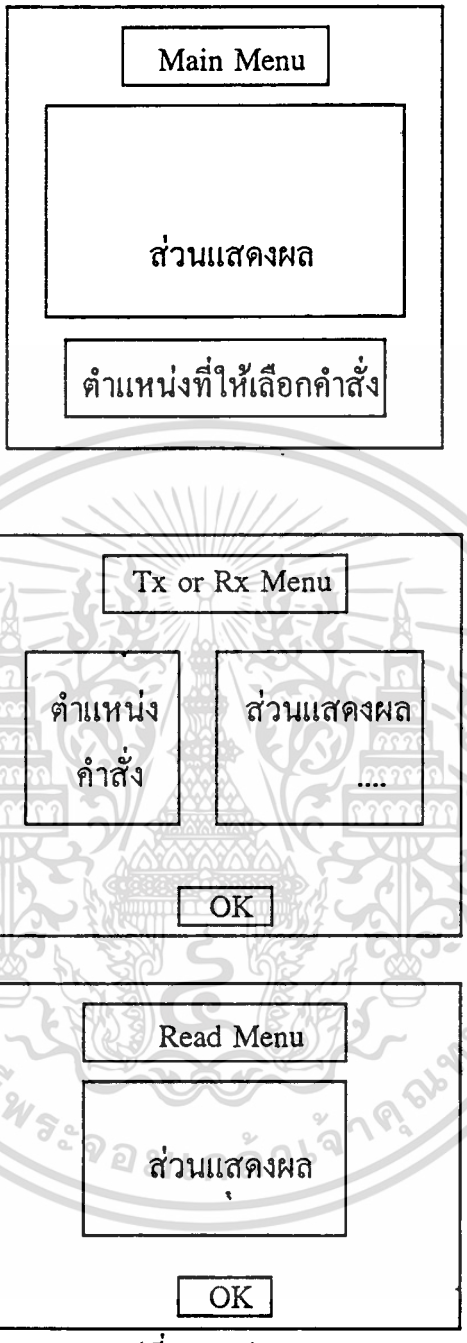
ทำการเขียน **Flow chart** การทำงานคร่าวๆของโปรแกรม และรูปแบบเมนูต่างๆว่าจะให้อยู่ในลักษณะใด



รูปที่ 29 Flow Chart Main Menu

รูปที่ 30 Flow Chart Tx, Rx, Read Menu

รูปแบบเมนู



รูปที่ 31 รูปแบบเมนู

จาก Flow Chart นี้ทำการเขียนโปรแกรมเพื่อให้ได้ผลตามต้องการ ซึ่งอาจมีการเพิ่มเติมรายละเอียดต่างๆลงไปอีกได้เช่น Help Menu, รายละเอียดโปรแกรม เป็นต้น

ในส่วนการแสดงผลภาษาซีสามารถแสดงได้ 2 โหมด คือ Garphics Mode และ Text Mode ใน project นี้ใช้ Graphics Mode เพราะมีรูปแบบในการแสดงผลที่สวยงามกว่า แต่การเขียนโปรแกรมจะยุ่งยากกว่า Text Mode อยู่มาก

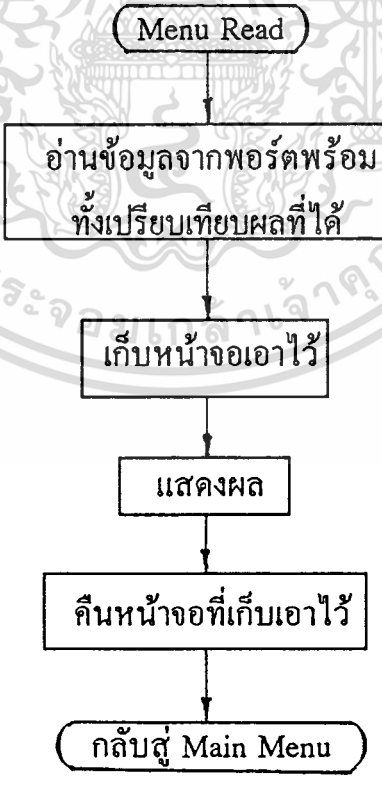
ส่วนการติดต่อระหว่างผู้ใช้กับโปรแกรม ก็มีอยู่ 2 วิธี เช่นเดียวกัน คือใช้Keyborad หรือ Mouse อย่างเดียว หรือทั้งสองอย่างคู่กัน ซึ่งใน project นี้ใช้ทั้ง Keyborad และ Mouse เพื่อความสะดวกของผู้งานกับโปรแกรมที่เขียนขึ้น

เพื่อให้การติดต่อระหว่างผู้ใช้กับโปรแกรม ทำงานแล้วไม่ขาดตอน จึงเขียนโปรแกรมให้แสดงผลแบบ Pop up windows จะหยุดการทำงานของโปรแกรมไว้ชั่วคราว ซึ่งทำให้ผู้ใช้เกิดความรู้สึกที่ดีกว่าการแสดงผลแบบธรรมดา (Pop up windows เมื่อเรียกใช้งาน จะเขียนหน้าจอใหม่ทับหน้าจอที่ใช้งานปัจจุบันที่ใช้งานอยู่เดิม ซึ่งหลักการจริงๆ ก็คือจะมีการนำข้อความเดิมที่ถูกทับไว้นำไปเก็บไว้ใน Memory แล้วแสดงผลที่ต้องการเมื่อออกจาก Pop up windows จะนำข้อมูลที่อยู่ใน Memory กับมาไว้ยังตำแหน่งเดิม)

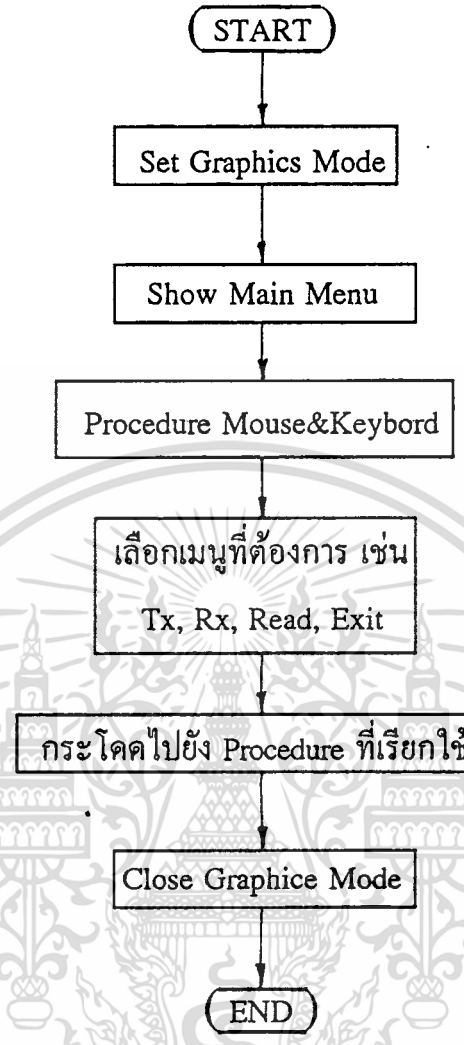
ในการเขียนโปรแกรมเราจะเขียนเป็น Procedure ย่อยๆ หลาย Procedure เพื่อจะง่ายต่อการออกแบบโปรแกรม และสามารถดัดแปลงแก้ไขได้ง่าย

จากที่กล่าวมาทั้งหมดเราสามารถเขียนเป็น Flow Chart การทำงานของโปรแกรมที่ละเอียดกว่าเดิมได้ดังนี้

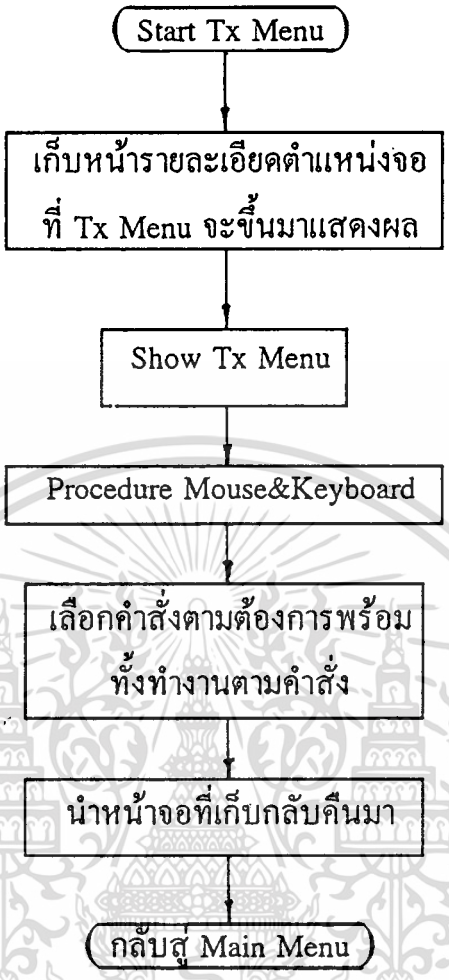
แสดง Flow Chart ใหม่ ที่ละเอียดขึ้น



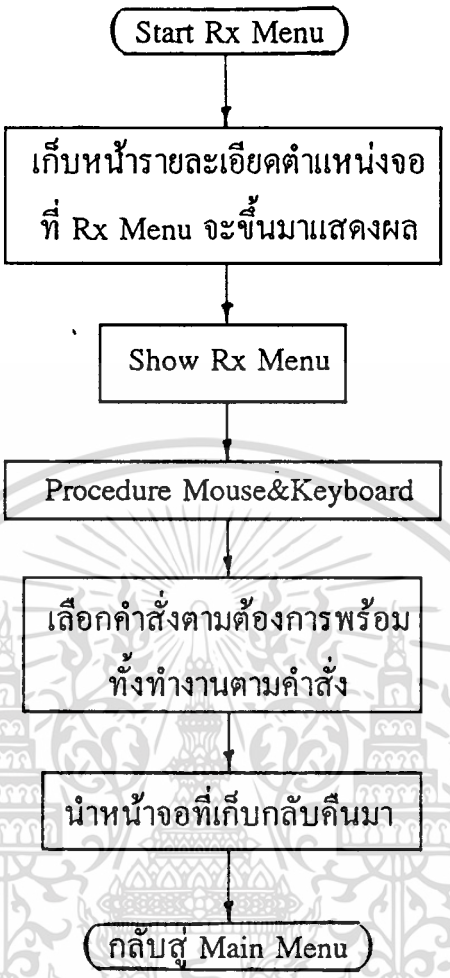
รูปที่ 32 Flow Chart Read Menu ใหม่



รูปที่ 33 Flow Chart Main Menu ใหม่



รูปที่ 34 Flow Chart Tx Menu ใหม่



รูปที่ 35 Flow Chart Rx Menu ใหม่

ทำการทดสอบโปรแกรมในการติอต่อกับ Hardware โดยสร้างชุดทดลองขึ้นมาโดยมี Buffer กั้นข้อมูลไว้ ซึ่งจะทำหน้าที่ขับ LED เพื่อแสดงผลข้อมูลที่รับส่งมาให้เห็นจริง และ ป้องการอุปกรณ์ในการติดต่อ (ในที่นี้ใช้ IC เบอร์ 8255 เป็นไอซีควบคุมพอร์ตขนาน 8 bit จำนาน 3 พอร์ต A.B,C up และ C down ทั้ง 3 สามารถเป็นได้ทั้งพอร์ตอินพุท และพอร์ตเอาท์พุต ซึ่งขึ้นกับ Control port ที่ส่งไปโปนแกรมการทำงาน ของ 8255 เอง) ไม่ให้เสียหาย ถ้าเกิดการ short ที่ output โดยชุดทดลองที่สร้างขึ้นมาจะต่อกับ พอร์ตหมายเลขที่ใช้ในการรับหรือส่งข้อมูล ทำการทดลองจนสมบูรณ์

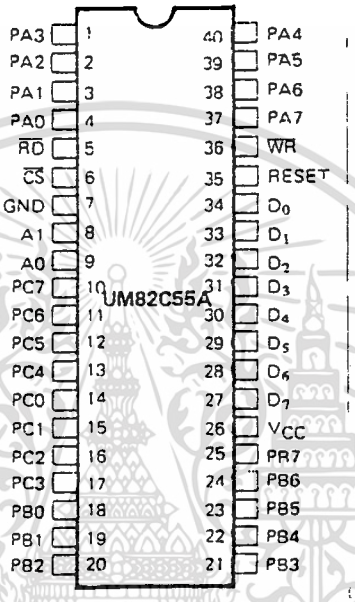
เปลี่ยนชุดทดลองเป็นวงจรใช้งานจริง ทำการทดลองและแก้ไขส่วนที่ผิดพลาดจนสมบูรณ์

3.2 ทฤษฎีและการใช้งาน 8255

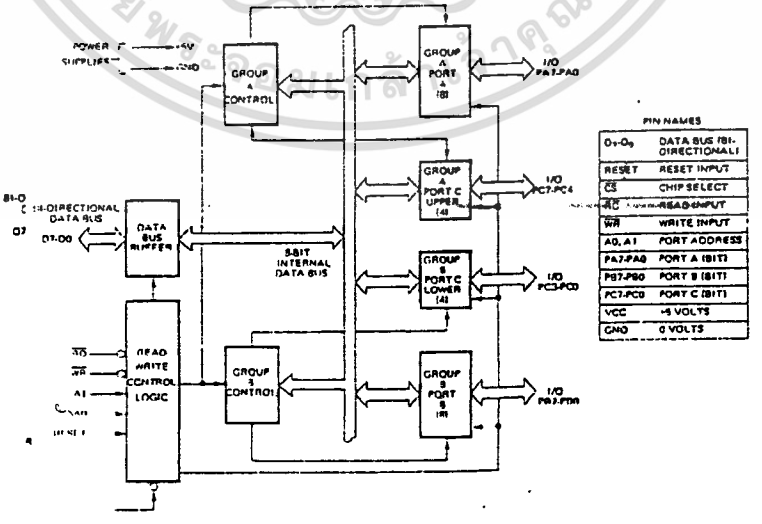
ไมโครโปรเซสเซอร์นั้นนอกจากติดต่อกับหน่วยความจำโดยกานำข้อมูลไปเก็บไว้หรืออ่านข้อมูลใดๆ ออกจากหน่วยความจำแล้วตัว CPU เองอาจจะต้องติดต่อกับส่วนประกอบภายนอกอื่นๆนั้น CPU ต้องติดต่อ (รับหรือส่งข้อมูล) โดยผ่านทางอินพุทหรือเอาท์พุทพอร์ท ซึ่งอาจสามารถใช้ไอซี TTL บางเบอร์มาใช้เป็นพอร์ทสำหรับ CPU ได้ แต่ทั้งนี้การใช้ไอซี TTL มีข้อจำกัดหลายอย่าง เช่น ในกรณีที่มีความจำเป็นจะต้องใช้พอร์ทหลายๆพอร์ท เพราะต้องติดต่อกับอุปกรณ์ภายนอกหลายจุด จึงจำเป็นต้องใช้ไอซีเหล่านี้จำนวนหลายตัวและอาจทำให้ยากในการออกแบบวงจร อีกทั้งไม่สารถจะเปลี่ยนแปลงลักษณะการทำงานให้แตกต่างไปจากเดิมที่ได้ออกแบบไว้แล้ว ดังนั้นผู้ผลิต CPU ในตระกูลต่างๆจึงมักจะผลิตไอซีประเภท LSI ที่ทำหน้าที่เป็นพอร์ทมาเพื่อใช้งานร่วมกับ CPU เบอร์นั้นๆ ได้สะดวกซึ่งจะทำให้การรับส่งข้อมูลมีความเชื่อถือได้สูงและยังสามารถเปลี่ยนแปลงชนิดของพอร์ท (จากอินพุทเป็นเอาท์พุทหรือจากเอาท์พุทเป็นอินพุท) ได้ง่ายโดยการควบคุมของ CPU เอง ในบทนี้จะกล่าวถึงไอซีที่ทำหน้าที่เป็นอินพุทและเอาท์พุทพอร์ท ซึ่งนิยมในการนำไปใช้งานมากที่สุดอีกทั้งยังมีราคาถูกหาซื้อได้ง่ายคือไอซีเบอร์ 8255 ของบริษัทอินเทลที่จริงแล้วไอซีเบอร์ 8255 นี้ได้ถูกออกแบบและผลิตขึ้นมาเพื่อใช้งานร่วมกับ CPU เบอร์ 8080 แต่ก็สามารถนำมาใช้กับ Z-80 หรือ CPU เบอร์อื่นๆได้ โดยในบทนี้จะกล่าวถึงคุณสมบัติการทำงานรวมทั้งการนำ 8255 ไปใช้งาน

8255 นั้นเป็นไอซี LSI ขนาด 40 ขา จากรูปที่ 36 แสดงตำแหน่งของขาต่างๆทั้ง 40 ขา ส่วนรูป 37 แสดงแผนผังภายในของ 8255 ซึ่ง 8255 นี้มีพอร์ทสำหรับรับส่งข้อมูลอยู่ด้วยกัน 3 พอร์ทมีชื่อดังนี้คือ พอร์ท A,B และ C โดยพอร์ท C นี้จะแบ่งออกเป็น 2 ส่วนตือ พอร์ท C ล่าง (CLO) กับพอร์ท C บน (CHI) นอกจากนี้แล้วยังมีพอร์ทหนึ่ง ซึ่งทำหน้าที่ควบคุมการทำงานของพอร์ท A B และ C โดยการรับคำสั่งมาจาก CPU พอร์ทนี้เรียกว่า พอร์ทควบคุม (Control port) พอร์ทนี้ใช้งานก็ต่อเมื่อ CPU ต้องการกำหนดลักษณะการทำงานของพอร์ท A B และ C หรือต้องการเปลี่ยนแปลงค่าจากที่กำหนดไว้เดิม CPU จะส่งรหัสควบคุมมาทางดาต้าบัส (Data Bus) ให้แก่พอร์ทควบคุมนี้

การกำหนดรหัสที่ใช้ในการควบคุมพอร์ทต่างๆ นี้จะกล่าวในตอนต่อไปในทางปฏิบัติผู้ออกแบบระบบต้องนำหรัสควมคุมที่ได้มาตามกำหนดของ 8255 นี้ไปใส่ในโปรแกรมเพื่อให้ CPU ทำการส่งรหัสควมคุมนี้มายังพอร์ทควมคุมเมื่อระบบนั้นเริ่มต้นทำงานหน้าที่ของขาต่างๆก่อนที่จะกล่าวถึงการนำ 8255 ไปใช้งานควรทราบหน้าที่ของขาต่างๆของ 8255 ทั้ง 40 ขาเสียก่อน จะทำให้เข้าใจ ถึงวิธีการใช้งานได้ดียิ่งขึ้นขาต่างๆของ 8255 สามารถแบ่งออกได้ดังนี้



รูปที่ 36 ตำแหน่งขาต่าง ๆ ของ 8255



รูปที่ 37 แผนผังภายในของ 8255

CS (Chip Select) ขานี้ใช้สำหรับรับสัญญาณจากภายนอกเพื่อใช้ในการเลือกว่าจะใช้ 8255 ตัวนี้ทำงานหรือไม่โดยถ้าขานี้ได้รับลอจิก "0" จะทำให้ 8255 เชื่อมต่อเข้ากับระบบบัสต่าง ๆ ของ CPU (โดยการเป็น Hi-Z)

RD (Read Enable) เป็นขาอินพุทที่จะรับสัญญาณจาก CPU ถ้าขานี้ได้รับลอจิก "0" และขณะนั้นขา CS ต้องเป็นลอจิก "0" ด้วย 8255 จะทำการส่งข้อมูลจากพอร์ทที่ CPU ต้องการติดต่อด้วยนั้นให้แก่ CPU ทางดาต้าบัส

WR (Write Enable) มีหน้าที่การทำงานตรงข้ามกับขา RD คือ ถ้าขา WR นี้ได้รับ ลอจิก "0" (CS ต้องเป็น "0" ด้วยเช่นกัน) 8255 จะรับข้อมูลจากดาต้าบัสของ CPU ส่งออกไปยังพอร์ทที่ CPU กำหนดไว้

RESET คือ ขาที่ทำหน้าที่ Reset 8255 เมื่อใดที่ 8255 ได้รับสัญญาณ Reset มันจะกลับเข้าสู่โหมดอินพุทคือทุก ๆ พอร์ทจะเป็นอินพุทพอร์ท ขา RESET นี้ใช้เมื่อต้องการเคลียร์สถานะต่าง ๆ ของ 8255

D0-D7 คือ ขาข้อมูลที่ใช้ในการติดต่อรับส่งข้อมูลกับ CPU โดยขา D0-D7 นี้จะต่อเข้ากับดาต้าบัสของ CPU เพื่อให้ CPU ส่งข้อมูลออกไปยังพอร์ทหรือรับข้อมูลจากพอร์ทส่งให้แก่ CPU ผ่านทาง D0-D7 นี้

A0-A1 คือ ขาแอดเดรสที่ใช้ในการเลือกพอร์ทที่ CPU ต้องการจะติดต่อด้วย ซึ่งมีความเป็นไปได้ทั้งหมด 4 ค่า ดังนี้ คือ

00 = พอร์ท A
01 = พอร์ท B
10 = พอร์ท C
11 = พอร์ทควบคุม

PA0-PA7 เป็นขาสัญญาณของพอ์ท A ใน 8255 ซึ่งจะถูกเลือกโดยค่าของ A0-A1 และเมื่อพอร์ทนี้ถูกเลือกใช้ ข้อมูลต่าง ๆ ก็จะถูกส่งผ่าน PA0-PA7 นี้ไปยัง D0-D7 (กรณีที่ให้พอร์ท A นี้เป็นอินพุทพอร์ท) หรือจาก D0-D7 (กรณีที่เป็นเอาท์พุทพอร์ท)

PB0-PB7 เป็นขาสัญญาณของพอร์ท B ซึ่งจะถูกเลือกโดยลอจิกที่ A0-A1 เช่นกันกับพอร์ท A และ พอร์ท B นี้มีข้อจำกัดในการรับส่งข้อมูลที่ต่างจากพอร์ท A ในบางกรณี

PC0-PC7 ซึ่งเป็นสายสัญญาณของพอร์ท C ซึ่งจะแบ่งออกเป็น 2 กลุ่ม คือ PC0-PC3 และ PC4-PC7 โดยแต่ละกลุ่มสามารถแยกกันทำงานได้โดยอิสระ คือ กลุ่มหนึ่งอาจเป็นอินพุทพอร์ทในขณะที่อีกกลุ่มหนึ่งเป็นเอาท์พุทพอร์ทได้ แต่จะทำงานพร้อม ๆ กัน โดยการเลือกด้วยลอจิกที่ A0-A1 การใช้งาน 8255 นั้น แบ่งลักษณะการทำงานออกเป็น 3 โหมด (Mode) ด้วยกันคือ

- โหมด 0 เป็นโหมดอินพุทหรือเอาท์พุทพอร์ทอย่างใดอย่างหนึ่งซึ่งทั้ง 3 พอร์ท คือ A, B และ C สามารถทำงานในโหมดนี้ได้
- โหมด 1 เป็นโหมดอินพุท หรือเอาท์พุทพอร์ทอย่างใดอย่างหนึ่งเช่นกัน แต่จะมีการทำงานเป็นลักษณะของ Handshaking ซึ่งจะกล่าวรายละเอียดในภายหลัง ในโหมดนี้ทำงานได้เฉพาะพอร์ท A และ B
- โหมด 2 เป็นโหมด Bidirectional คือ เป็นได้ทั้งอินพุท และเอาท์พุทพอร์ทในเวลาเดียวกัน และทำงานแบบ Handshaking เช่นเดียวกับโหมด 1 ในโหมดนี้ใช้ได้เฉพาะในพอร์ท A เท่านั้น

การกำหนดโหมดการทำงานของ 8255 นั้นทำได้โดย CPU ทำการส่งรหัสควบคุมผ่านทางดาต้าบัสมายังพอร์ทควบคุม (Control port) ของ 8255 รหัส ควบคุมนี้จะมีขนาด 1 ไบท์ เรียกว่า Control Byte และในแต่ละบิทของ Control Byte (1 Byte = 8 bit) นั้นจะมีความหมายของตัวเองดังแสดงในรูปที่ 2 ซึ่งจะอธิบายได้ดังนี้

บิท D7 เป็นบิทที่แสดงว่า Byte นี้เป็นรหัสควบคุม (Control Byte) ซึ่งจะมีผลต่อการกหนดโหมดการทำงานของ 8255

บิท D6 และ D5 มีความหมายในการเลือกโหมดของพอร์ท A ซึ่งสามารถทำงานได้ทั้ง 3 โหมด โดยลอจิกที่ D6 และ D5 จะมีความหมายดังนี้

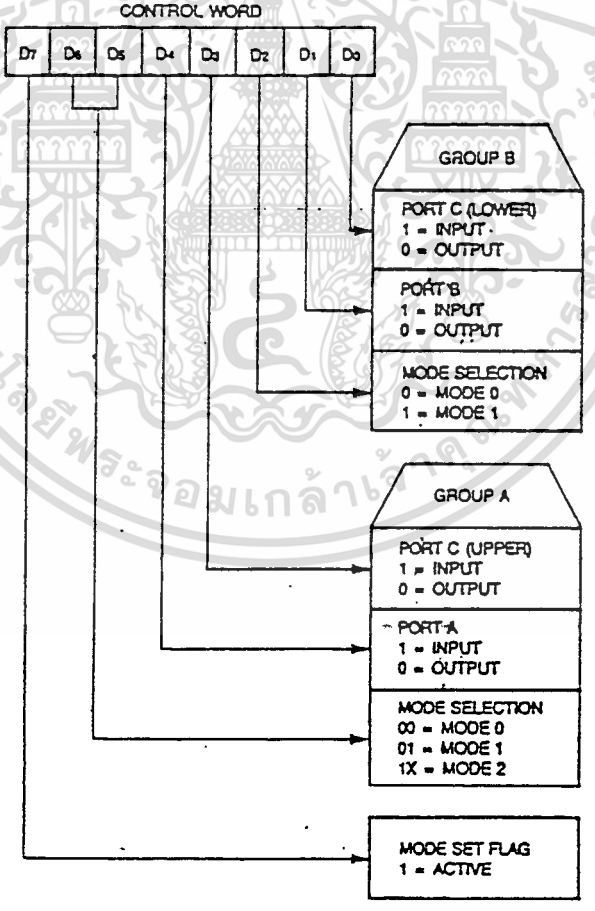
00	=	โหมด 0
01	=	โหมด 1
10	=	โหมด 2
11	=	โหมด 2

บิท D4 ถ้าเป็นลอจิก “0” หมายถึงสั่งให้พอร์ท A ทำหน้าที่เป็นเอาท์พุทพอร์ท แต่ถ้าเป็นลอจิก “1” พอร์ท A จะเป็นอินพุทพอร์ทบิทนี้จะมีความหมายเมื่อเราให้ 8255 ทำงานในโหมด 0 หรือ โหมด 1 เท่านั้นเพราะในโหมดที่ 2 พอร์ท A จะเป็นทั้งอินพุท

และเอาท์พุทพอร์ท ในเวลาเดียวกันบิท D3 เป็นบิทที่กำหนดการทำงานของพอร์ท C บน (PC4-PC7) ถ้าบิทนี้เป็นลอจิก "0" พอร์ท C บนนี้จะเป็นเอาท์พุทพอร์ท ถ้าเป็น "1" จะเป็นอินพุทพอร์ท

บิท D2 เป็นบิทที่ใช้สำหรับกำหนดโหมดการทำงานของพอร์ท B ถ้าเป็นลอจิก "0" หมายถึงให้พอร์ท B ทำงานในโหมด 0 ถ้าเป็นลอจิก "1" จะทำงานในโหมด 1 บิท D1 เป็นการกำหนดให้พอร์ท B เป็นอินพุทหรือเอาท์พุทพอร์ท ถ้า D1 เป็นลอจิก "0" จะเป็นเอาท์พุทพอร์ทแต่ถ้าเป็นลอจิก "1" จะเป็นอินพุทพอร์ท

บิท D0 เป็นบิทที่ใช้กำหนดการเป็นอินพุทหรือเอาท์พุทของพอร์ท C ล่าง (PC0-PC3) ถ้าบิทนี้เป็น "0" จะเป็นเอาท์พุท ถ้าเป็น "1" จะเป็นอินพุท ตัวอย่างการกำหนดรหัสควบคุมที่จะส่งให้แก่พอร์ทควบคุมของ 8255 หรือที่เรียกว่า การโปรแกรม 8255 เช่น ถ้าเราต้องการให้พอร์ท A และพอร์ท C บนเป็นอินพุทพอร์ท ส่วนพอร์ท B กับ



รูปที่ 38 แสดงความหมายของแต่ละบิทในรหัสควบคุม

พอร์ท C ล่างเป็นเอาท์พุทพอร์ท และทุกพอร์ททำงานในโหมด 0 จะสามารถกำหนดรหัสควบคุมได้ดังรูปที่ 38 จากรูปเราจะได้รหัสควบคุมหรือคอนโทรลไบท์นี้เป็น 10011000 หรือ 98h ในระบบเลขฐานสิบหก และนำมาเขียนเป็นคำสั่งสำหรับ Z-80 ได้ดังนี้

```
LD  A,98h                ; กำหนดรหัสควบคุม
OUT (Control Port ) , A  ; ส่งรหัสควบคุมให้แก่พอร์ทควบคุม
```

3.4 การทำงานโหมดต่างๆ ของ 8255

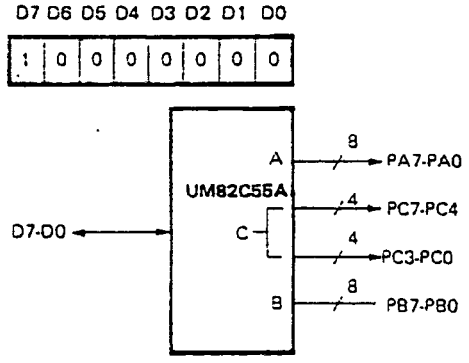
- โหมด 0

ในโหมด 0 นี้ เป็นการกำหนดให้พอร์ททุกพอร์ทของ 8255 นี้ เป็นอินพุทเอาท์พุทพอร์ทแบบพื้นฐานหรือที่เรียกว่า Simple I/O Port ซึ่งเป็นที่นิยมใช้กันมากเมื่อนำมาเป็นพอร์ทของ Z-80 มีรูปแบบความเป็นไปได้ในการโปรแกรมให้พอร์ท A , B และ C เป็นอินพุทหรือเอาท์พุทได้ทั้งหมด 16 รูปแบบ ดังรูปที่ 39

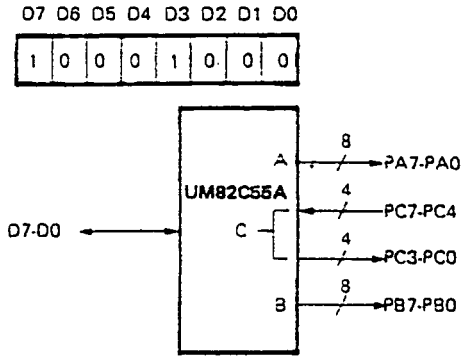
จากรูปในแต่ละช่องจะแสดงรูปบบดการทำงานในโหมด 0 ของ 8255 หนึ่งรูปแบบ โดยจะแสดงถึงสถานะการทำงานของพอร์ท A , B และ C ด้วยลูกศรถ้าลูกศรชี้ออกจากตัว 8255 หมายถึง เป็นอินพุทพอร์ท ส่วนพอร์ท C นั้นจะแบ่งเป็น 2 ส่วน คือ พอร์ท C บน และพอร์ท C ล่าง พอร์ทละ 4 เส้นส่วนด้านบนของแต่ละช่องแสดงต่าของรหัสควบคุม (Control Word) ที่ต้องส่งให้แก่พอร์ทควบคุมในรูปเลขฐาน 16 เพื่อโปรแกรมให้พอร์ทต่างๆทำงานเป็นอินพุท เอาท์พุทตามที่แสดงไว้ในช่องนั้นเช่นถ้าเราต้องการให้พอร์ท A และพอร์ท C บน เป็นอินพุทพอร์ท ส่วนพอร์ท B และพอร์ท C ล่าง เป็นเอาท์พุทพอร์ทดังตัวอย่างที่เคยกล่าวมาแล้วถ้าดูจากรูปที่ 5 ก็จะตรงกับช่องของ Control Word # 14 มีลอจิกที่ D_0-D_7 ดังนี้คือ 10011010 หรือ 98H ในฐาน 16 เช่นเดียวกับที่เคยกล่าวมาแล้วนั่นเอง

Mode 0 Configurations

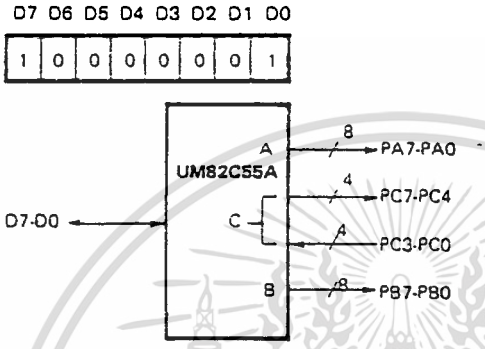
CONTROL WORD #0



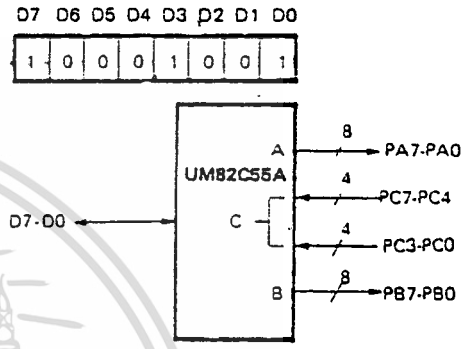
CONTROL WORD #4



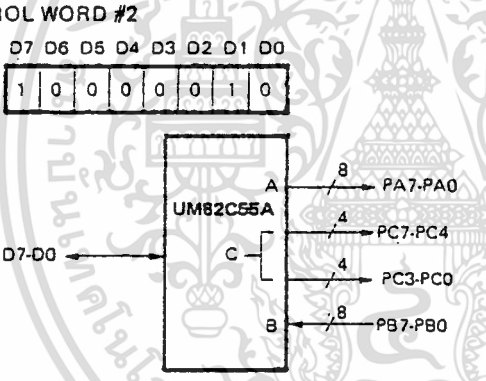
CONTROL WORD #1



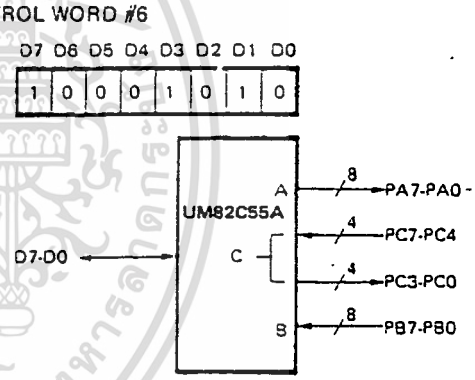
CONTROL WORD #5



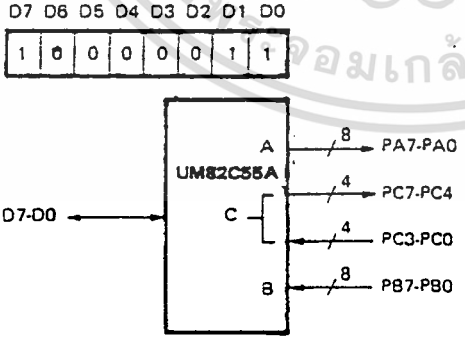
CONTROL WORD #2



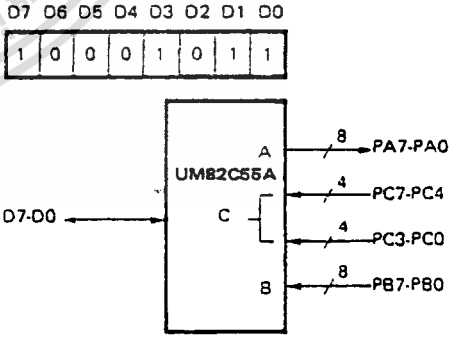
CONTROL WORD #6



CONTROL WORD #3

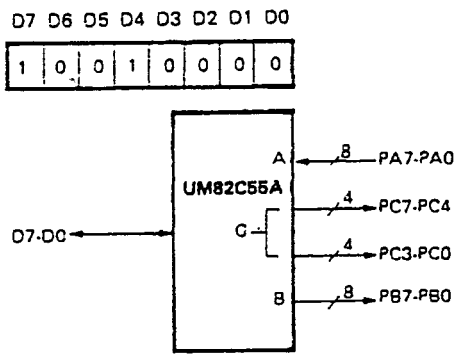


CONTROL WORD #7

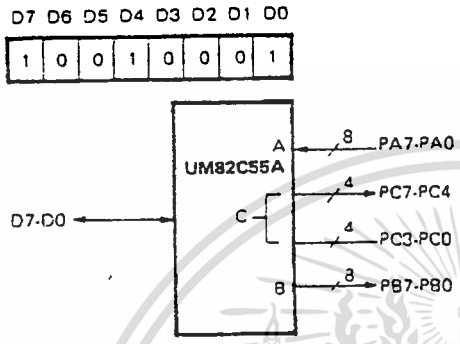


รูปที่ 39 รหัสควบคุมของการทำงานในโหมด 0

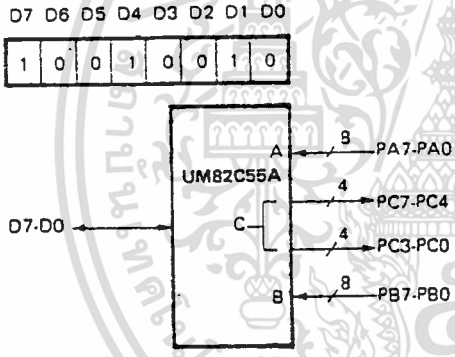
CONTROL WORD #8



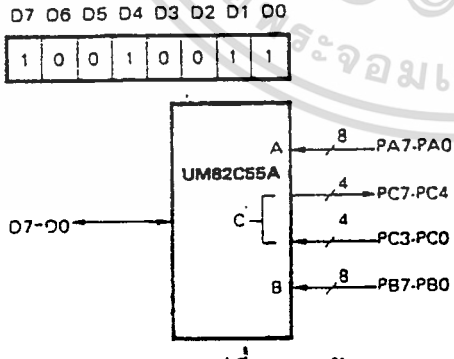
CONTROL WORD #9



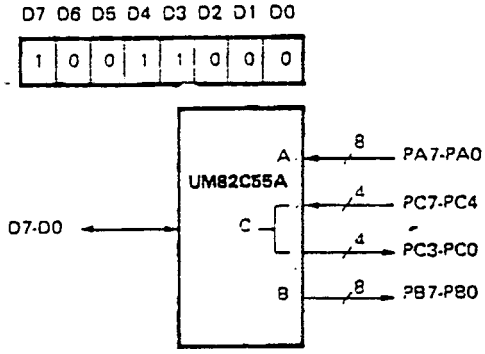
CONTROL WORD #10



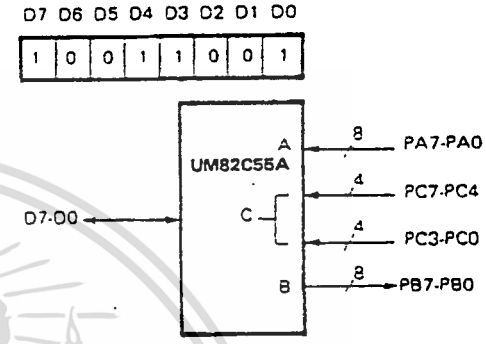
CONTROL WORD #11



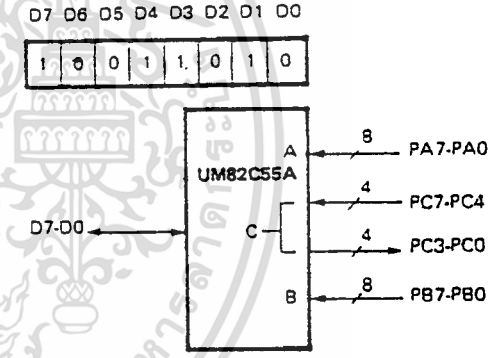
CONTROL WORD #12



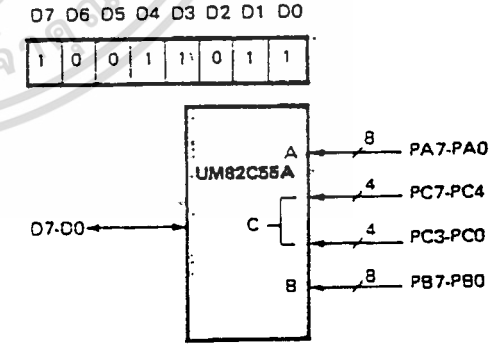
CONTROL WORD #13



CONTROL WORD #14



CONTROL WORD #15



รูปที่ 39 รหัสควบคุมของการทำงานในโหมด 0 (ต่อ)

ตัวอย่าง จงเขียนโปรแกรมควบคุม 8255 ซึ่งมีหมายเลขพอร์ทต่างๆดังกล่าวข้างต้นให้ทำงานในโหมด 0 โดยให้พอร์ท A ,B และ พอร์ม C ล่างเป็นเอาท์พุทส่วนพอร์ท C บน เป็นอินพุทพอร์ทหลังจากนั้นให้ส่งข้อมูลค่า A7H ออกไปยังพอร์ท A , B และ C ตามลำดับ

Sol^{n} จากโจทย์ดังกล่าวเราสามารถหาค่าของ Control Word ได้ จากรูปที่ 39 ซึ่งจะตรงกับ Control Word # 4 มีลอจิก D7-D0 เป็น 10001000 เท่ากับ 88H ในเลขฐาน 16 สามารถนำไปเขียนเป็นโปรแกรมได้ดังนี้

```
LD  A,88H          ; กำหนดรหัสควบคุม
OUT (03H) , A      ; ส่งไปยังพอร์ทควบคุม
LD  A,A7H          ; เตรียมค่าที่กำหนด (จากโจทย์)
OUT (00H) , A      ; ส่งไปยังพอร์ท A
OUT (01H) , A      ; ส่งไปยังพอร์ท B
OUT (02H) , A      ; ส่งไปยังพอร์ท C
```

หลังจากทำโปรแกรมนี้เสร็จแล้วที่พอร์ท A และ B จะมีค่าเป็น A7H และ ค่าคงที่นี้ไว้ (Latch) จนกว่า CPU จะส่งค่าใหม่มาแทน ส่วนพอร์ท C นั้นเนื่องจากเป็นเอาท์พุทพอร์ทเฉพาะพอร์ท C ล่าง(4 bit) ดังนั้นค่าที่ส่งมายังพอร์ท C ล่างก็คือค่าจอง 4 bit ล่างของดาต้าบัส (D_0-D_7) ได้แก่ D_0-D_3 เท่านั้น ทำให้ได้ค่าจากพอร์ท C เป็น 7H (ถ้าเขียนให้เต็ม 8 bit จะได้เป็น 07H) และในการแบ่งพอร์ท C ออกเป็น 2 ส่วน เช่นนี้สิ่งที่ต้องคำนึงถึงคือ ถ้า CPU ต้องการรับหรืออ่านข้อมูลจากพอร์ท C ส่วนที่เป็นอินพุทพอร์ทเพียง 4 bit ที่เป็นเอาท์พุทพอร์ทมาด้วย เช่นถ้าเราเขียนโปรแกรมต่อจากโปรแกรมข้างบนด้วยคำสั่งถ่ายข้อมูลจากพอร์ท C เข้ามาไว้ที่รีจิสเตอร์ A ดังนี้

```
IN  A, (02H)       ; อ่านข้อมูลจากพอร์ท C บนมาไว้ที่ A
```

สมมติให้ข้อมูลที่พอร์ท C บน เป็น 0110 ข้อมูลที่อ่านได้จากพอร์ท C และนำมาไว้ที่ A จะเป็น67H ค่า "6" คือ 0110 ซึ่งจะเป็นของ 4 ถึงบิท 7 ของพอร์ท C (PC4-PC7) ส่วนค่า "7" คือค่าของพอร์ท C ล่าง (PC0-PC3) ที่ได้ถูกส่งออกมาก่อนที่จะมีการอ่านค่าจากพอร์ท C แต่ถ้าเราต้องการเฉพาะข้อมูลของพอร์ท C บน ที่อ่านมาได้จะต้องนำไปตัดส่วนของ 4bit ล่าง ออกด้วยคำสั่ง AND หรืออื่นๆดังนี้

```
IN  A, (02H)          ; อ่านข้อมูลจากพอร์ท C บน มาไว้ที่ A
AND F0H               ; ตัดส่วนของ 4 bit ล่างออก
```

จะได้ข้อมูลที่ A เป็น 60H คือ ข้อมูลที่อ่านมาได้เพื่อนำไปใช้งานต่อไปตามจุดประสงค์ของผู้เขียนโปรแกรม จากตัวอย่างที่ผ่านมาจะเห็นได้ว่าการใช้งาน 8255 ในโหมด 0 นี้ง่ายและสะดวกมากจึงนิยมใช้กันในไมโครคอมพิวเตอร์แผ่นพิมพ์เดี่ยว (Single Board) ในส่วนของระบบแสดงผล (LED 7-LEGMENT) การรับคีย์บอร์ด , การกำเนิดเสียง ฯลฯ

- โหมด 1

จากที่กล่าวมาแล้วในตอนต้นว่าสำหรับโหมด 1 นี้ เป็นการรับส่งข้อมูลในแบบ Handshaking ความหมายของ Handshaking ก็คือ ระหว่าง CPU , พอร์ท และอุปกรณ์ภายนอกนั้นขณะที่รับ-ส่ง ข้อมูลกันนั้น นอกจากการรับ-ส่ง ข้อมูลกันแล้ว ยังต้องมีสัญญาณในการตอบรับในแต่ละครั้งของการรับ-ส่ง ข้อมูล โดยผู้รับกับผู้ส่งนั้น จะต้องทำงานสัมพันธ์กันตลอดเวลาซึ่งเป็นประโยชน์ในกรณีที่อุปกรณ์ภายนอกนั้นมีการทำงานที่ช้ากว่า CPU ทำให้ไม่สามารถทำงาน (รับหรือส่ง) ข้อมูลได้ทัน CPU จึงต้องใช้วิธีรส่งข้อมูลแบบ Handshaking โดยอุปกรณ์ภายนอกจะเป็นตัวกำหนดจังหวะในการรับ-ส่งข้อมูลเอง เช่น การส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ซึ่งเครื่องพิมพ์นั้นทำงานได้ช้ากว่าคอมพิวเตอร์มาก เมื่อคอมพิวเตอร์ส่งข้อมูลตัวอักษรตัวแรกให้แก่เครื่องพิมพ์ทำการพิมพ์ เครื่องพิมพ์ก็จะทำการประมวลผลต่าง ๆ และเมื่อพร้อมที่จะพิมพ์ตัวอักษรนั้นแล้วก็จะส่งสัญญาณบอกคอมพิวเตอร์ให้ทำการส่งตัวอักษรตัวต่อมาได้ การติดต่อระหว่างเครื่องพิมพ์กับคอมพิวเตอร์จึงเป็นไปโดยไม่ผิดพลาด ส่วนสัญญาณที่ใช้ในการควบคุมการรับส่งข้อมูลนี้จะได้มาจากพอร์ท C เพราะในโหมดนี้พอร์ทที่ใช้ในการรับส่งข้อมูลได้ คือ พอร์ท A และ B เท่านั้น ส่วนพอร์ท C จะเป็นตัวส่งและรับสัญญาณควบคุมกับอุปกรณ์ภายนอก และสัญญาณควบคุมในแต่ละบิทของพอร์ท C จะเป็นตามตารางที่ 2 ลักษณะของการรับส่งข้อมูลแบบ Handshaking นั้นแสดงในรูปที่ 40 ในกรณีเป็นอินพุทพอร์ท(A หรือ B ก็ตาม) เมื่ออุปกรณ์ภายนอกต้องการส่งข้อมูลให้แก่ CPU มันก็จะส่งข้อมูลเข้ามายังพอร์ท พร้อมกับส่งสัญญาณ STB (Strobe) มาบอกแก่ 8255 เมื่อ 8255 ได้รับสัญญาณ STB ก็จะรับข้อมูลนั้นไปเก็บ

ไว้ในรีจิสเตอร์ภายในก่อน แล้ว 8255 ก็จะส่งสัญญาณ IBF (Input Buffer Full) เป็นลอจิก "1" ไปบอกแก่อุปกรณ์ภายนอกว่า พอร์ทได้รับข้อมูลมาเก็บไว้ในบัฟเฟอร์(รีจิสเตอร์) แล้ว และอย่าเพิ่งส่งมาอีกจนกว่า CPU จะรับข้อมูลนั้นไปจากพอร์ทแล้ว (โดยดูจากการที่ CPU ส่งสัญญาณ RD มายัง 8255) สัญญาณ IBF ก็จะกลับเป็น "0" เพื่อบอกให้อุปกรณ์ภายนอกส่งข้อมูลต่อไปได้ ส่วนในกรณีที่เป็นเอาท์พุทพอร์ทนั้น เมื่อ CPU ส่งข้อมูลออกมายังพอร์ทของ 8255 ตัว 8255 ก็จะรับข้อมูลนั้นไปเก็บไว้ในรีจิสเตอร์ภายใน แล้วส่งสัญญาณ OBF(Output Buffer Full) บอกไปยังอุปกรณ์ภายนอกว่ามีข้อมูลมารอที่บัฟเฟอร์(รีจิสเตอร์) ของพอร์ทแล้วให้มารับไปได้ อุปกรณ์ภายนอกก็จะส่งสัญญาณตอบรับ ACK(Acknowledge) มายังพอร์ทเพื่อขอรับข้อมูลไป หลังจากนั้นสัญญาณ OBF ก็จะกลับเป็น "1" เพื่อรอให้ CPU ส่งข้อมูลใหม่ต่อไป ในทางปฏิบัติแล้วการที่ CPU จะทราบได้อย่างไรว่าเมื่อไหร่ที่อุปกรณ์ภายนอกได้ส่งข้อมูลเข้ามารอที่บัฟเฟอร์ของพอร์ทแล้ว ในกรณีที่เป็นอินพุทพอร์ท หรือ

ขา	กรณีอินพุท	กรณีเอาท์พุท
PC_0	$INTR_B$	$INTR_B$
PC_1	IBF_B	OBF_B
PC_2	STB_B	ACK_B
PC_3	$INTR_A$	$INTR_A$
PC_4	STB_A	I/O
PC_5	IBF_A	I/O
PC_6	I/O	ACK_A
PC_7	I/O	OBF_A

ตารางที่ 1

เมื่อไรที่อุปกรณ์ภายนอกได้รับข้อมูลจากพอร์ทไปแล้ว พร้อมที่จะให้ CPU ส่งข้อมูลต่อไปได้ ในกรณีที่เป็นอินพุทพอร์ทนั้น มีวิธีที่จะให้ CPU ทราบ 2 ลักษณะ คือ

- การเขียนโปรแกรมให้ CPU ทำการตรวจสอบสัญญาณควบคุมจาก 8255 เช่น ในกรณีเอาท์พุทพอร์ทก็จะให้ CPU คอยตรวจสอบบิทที่ 7 ของพอร์ท C (OBF) หลังจากที่ส่งข้อมูลให้แก่พอร์ทไปแล้ว ถ้าบิท 7 นี้ยังเป็นลอจิก "0" อยู่แสดงว่า อุปกรณ์ภายนอกยังไม่ได้รับข้อมูลไปจากพอร์ท แต่ถ้าบิทที่ 7 เป็น "1" แสดงว่าอุปกรณ์ภายนอกได้รับข้อมูลไปแล้ว จึงให้ CPU ทำการส่งข้อมูลชุดใหม่ต่อไปเช่นเดียวกัน ในกรณีของอินพุทพอร์ทก็ให้ CPU ตรวจสอบที่บิทที่ 1 ของพอร์ท C ได้ในทำนองเดียวกัน แวิธีนี้ CPU จะต้องเสียเวลาตรวจสอบลอจิกของบิทดังกล่าวอยู่ตลอดเวลา ซึ่งจะทำให้เสียเวลาของ CPU โดยเปล่าประโยชน์ จึงควรเลือกใช้วิธีนี้ต่อเมื่อพิจาราณาแล้วว่าการเสียเวลาของ CPU นั้นไม่มีผลเสียต่อระบบ อีกวิธีหนึ่งนั้นคือการให้อุปกรณ์ภายนอกนั้นทำการของอินเทอรัพท์จากที่แสดงในตารางที่ 1 นั้นจะเห็นว่าที่บิท 0 และบิท 3 ของพอร์ท C นั้นจะเป็นสัญญาณอินเทอรัพท์(INTR) ของพอร์ท B และ A ตามลำดับซึ่งสามารถใช้สัญญาณอินเทอรัพท์นี้ให้เป็นประโยชน์ได้ ในการต่อใช้งานสัญญาณ INTR ไปที่ขา INT ของ Z-80 นี้จะต้องผ่าน Not gate(Inverter) เสียก่อนเนื่องจากการขออินเทอรัพท์ของ Z-80 นั้น เป็นแอคทีพ "0" การรับส่งข้อมูลในแบบนี้แสดงในรูปที่ 40 ในรูปที่ 40 กรณีที่เป็นอินพุทพอร์ท เมื่ออุปกรณ์ภายนอกส่งข้อมูลให้แก่พอร์ท พร้อมทั้งส่งสัญญาณ STB มาให้พอร์ทรับข้อมูลนี้ไปเก็บในรีจิสเตอร์ เมื่อพอร์ทรับข้อมูลนี้ไปเก็บแล้วก็จะส่งสัญญาณ IBF ตอบรับไปยังอุปกรณ์ภายนอกเพื่อให้หยุดการส่งข้อมูลซ้อน แล้วจึงทำการขออินเทอรัพท์โดยการขอส่งสัญญาณ INTR ผ่าน Inverter ไปที่ขา INT ของ CPU เพื่อขอให้ CPU มารับข้อมูลนี้ไป หลังจาก CPU รับข้อมูลนี้ไปแล้ว 8255 ได้รับสัญญาณ RD พอร์ทก็จะขอถอนอินเตอรัพท์ ทำให้สัญญาณ IBF กลับเป็น "0" เป็นการบอกให้อุปกรณ์ภายนอกได้ทราบว่าพร้อมที่จะรับข้อมูลไบท์ต่อไปได้แล้ว ส่วนในรูปที่ 40 เป็นกรณีของเอาท์พุทพอร์ทเมื่อ CPU ต้องการส่งข้อมูลให้อุปกรณ์ภายนอก โดยการส่งสัญญาณ WR แก่ 8255 ให้รับข้อมูลไปไว้ที่รีจิสเตอร์บัฟเฟอร์ โดยพอร์ทจะรับข้อมูลไว้ แล้วส่งสัญญาณ OBF ไปบอกอุปกรณ์ภายนอกให้มารับข้อมูลไป และหลังจากที่อุปกรณ์ภายนอกรับข้อมูลไปแล้วก็จะตอบรับโดยการส่งสัญญาณ ACK แก่ 8255 ทำให้สัญญาณ OBF กลับเป็น "1" พร้อมกับทำให้สัญญาณ INTR เป็น "1" ด้วย จึงจะเป็นการขออินเทอรัพท์ให้ CPU ส่งข้อมูลใหม่ต่อไปในกรณีที่ต้องการส่งข้อมูลต่อเนื่องตามกรรมวิธีเดิม แต่ถ้า CPU ยังไม่ต้องการส่งข้อมูลต่อหรือยังไม่มีข้อมูล

ก็สามารถสั่งให้พอร์ทขอถอนอินเทอรัพท์ได้ (ทำให้ INTR เป็น “0”)ด้วยการส่งแอดเดรส ไบท์ไปยังพอร์ทควบคุมเป็นการกำหนดให้บิทใด ๆ ของพอร์ท C เป็นลอจิก “0” หรือ “1” ตามต้องการ มีวิธีการกำหนดดังนี้

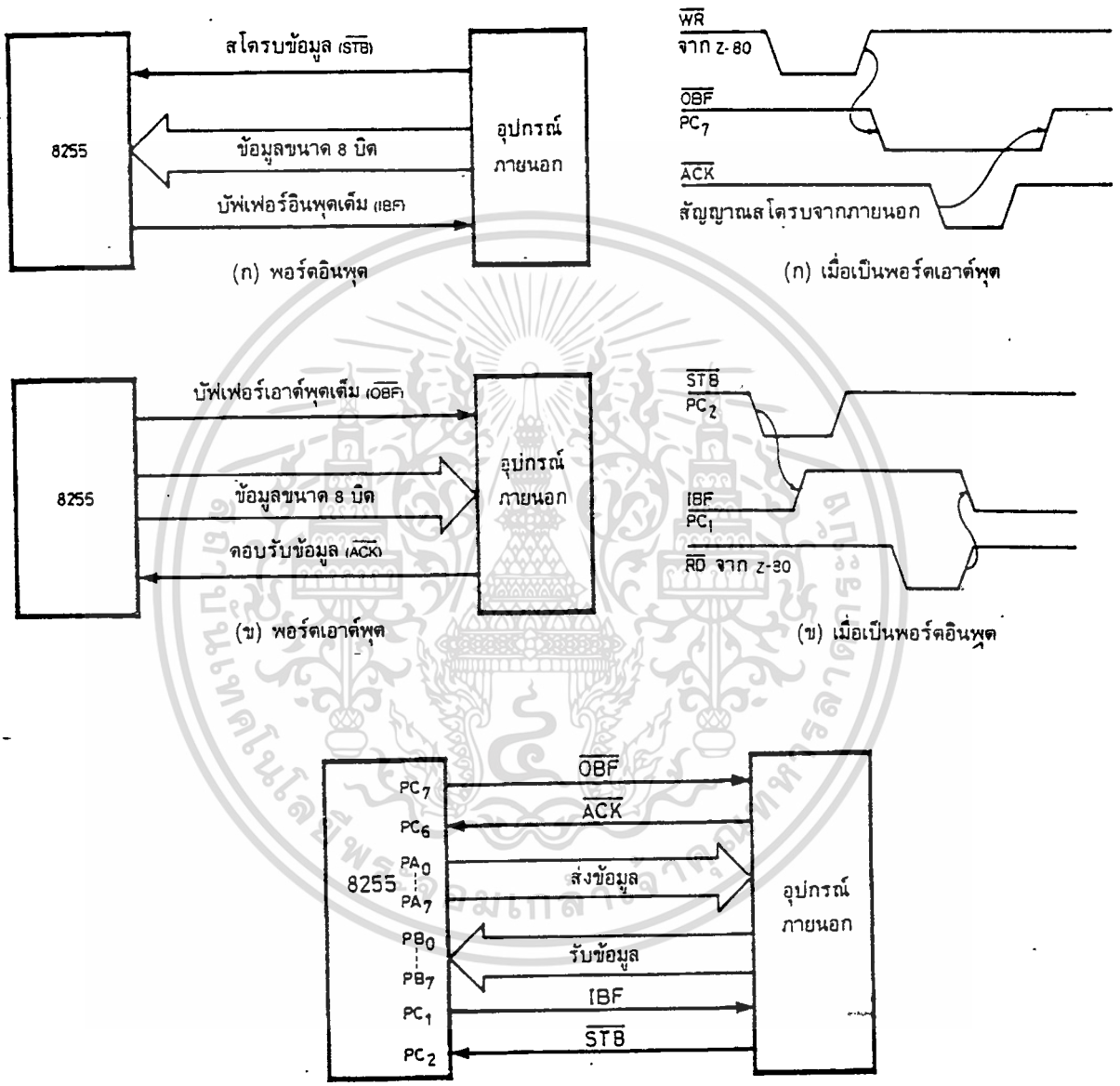
บิท 7 เป็น “0” เพื่อแสดงว่าเป็นแอสเดรสไบท์

บิท 6 ถึงบิท 4 ไม่ใช้จะให้เป็นลอจิกใดก็ได้ ปกติจะให้เป็น “0”

บิท 3 ถึงบิท 1 เป็นการบอกตำแหน่งบิทของพอร์ท C ที่ต้องการอ้างถึงดังนี้

000	=	PC0 (บิท 0 ของ Port C)
001	=	PC1
010	=	PC2
011	=	PC3
100	=	PC4
101	=	PC5
110	=	PC6
111	=	PC7

บิท 0ให้บิทใด ๆ ของพอร์ท C ที่กำหนดไว้ (ด้วยบิท 3-บิท1) นั้นมีลอจิกเป็น “0” หรือ “1” เช่นถ้าต้องการให้พอร์ท B ขอถอนอินเทอรัพท์ ก็คือทำให้ขา PC0 ซึ่งเป็นขา INTR ของพอร์ท B นั้นกลับเป็นลอจิก “0” จะได้แอดเดรสไบท์เป็น 00000000 หรือ 00H ในฐาน 16 การรับส่งข้อมูลแบบ Handshaking โดยกรรมวิธีขออินเทอรัพท์นี้ ถ้าในกรณีที่มี 8255 ทำงานร่วมกันหลายตัวจะทำให้ วิธีการขออินเทอรัพท์นี้ยุ่งยากมาก เนื่องจากต้องมีกรรมวิธีในการจัดลำดับความสำคัญเมื่อเกิดมีอุปกรณ์ที่ร้องขออินเทอรัพท์พร้อมกันหลาย ๆ ตัว ซึ่งมีเทคนิคหลายแบบทั้งนี้ผู้ที่ใช้ 8255 ทำงานในโหมด 1 นี้ (หรือโหมด 2 ก็ตาม) จะต้องมีความเข้าใจการทำงาน รวมทั้งมีความรู้ในระบบไมโครคอมพิวเตอร์มากพอสมควร



รูปที่ 40 การจัดสัญญาณการ HANDSHAKING

- โหมด 2

ในโหมดที่ 2 นี้ เป็นการรับ-ส่งข้อมูลแบบ Handshaking เช่นเดียวกับโหมด 1 แต่พอร์ทที่จะใช้งานในโหมดนี้ได้นั้นมีเพียงเฉพาะพอร์ท A พอร์ทเดียว แต่พอร์ท A ในโหมดนี้สามารถเป็นได้ทั้งอินพุทและเอาท์พุทพอร์ท หรือที่เรียกว่าเป็นพอร์ทแบบสองทิสทาง (Bidirectional) ส่วนสัญญาณที่ใช้ในการทำ Handshaking คือสัญญาณจากพอร์ท C ซึ่งมีลักษณะตามตารางที่ 3 ขณะที่โปรแกรมให้พอร์ท A ทำงานในโหมด 2 นี้ เรายังสามารถโปรแกรมให้พอร์ท B ไปใช้งานอื่น ๆ ได้อีกโดยที่เป็นอิสระจากกัน

พอร์ท C	ความหมาย
PC_0	I/O
PC_1	I/O
PC_2	I/O
PC_3	$INTR_A$
PC_4	STB_A
PC_5	IBF_A
PC_6	ACK_A
PC_7	OBF_A

ตารางที่ 2

3.4 การนำ 8255 ใช้งาน

ในที่นี้เราจะไม่ออกแบบการ์ด 8255 ของวงจรการใช้งานเอง แต่จะใช้วงจรสำเร็จรูป ET-PC8255 ซึ่ง ET-PC8255 จะเป็นการ์ดต่อขยายระบบเครื่อง PC ให้มีส่วนของ Input, Output Port ใช้งานมากขึ้น โดยจะมี Port ใช้งานเป็น Input หรือ Output จำนวน 9 Port หรือ 72 Bit I/O(TTL 0-5 V)

ซึ่งใน Project นี้โปรแกรมให้ 8255 ทำงานในโหมด 0 เพราะต้องใช้งานให้ทั้ง 3 พอร์ต เป็นทั้งอินพุทและเอาท์พุท รวมทั้งควบคุมการทำงานได้ง่ายกว่าโหมดอื่น

การติดตั้ง ET-PC8255 กับเครื่อง PC

1. ปิด SW.Power ของเครื่องคอมพิวเตอร์ PC นั้นก่อน
2. เปิดฝาเครื่องคอมพิวเตอร์
3. Set DIP SW. ตำแหน่ง Port ของการ์ด ET-PC8255 ไม่ให้ตรงกับตำแหน่ง Port ของการ์ดอื่นๆ
4. นำการ์ด ET-PC8255 ใส่เข้าไปยังเครื่องคอมพิวเตอร์ PC ทาง SOLT PC 62 PIN ดูการใส่การ์ดให้เรียบร้อยสนิทดีด้วยกับ SLOT PC
5. ถ้ามีการต่อสายแพร์จาก 34 PIN ไปใช้งานก็ให้ต่อก่อนให้เรียบร้อยแล้วเสร็จก่อน
6. เปิด SW Power เครื่องคอมพิวเตอร์ PC

การทำงานของ ET-PC8255

การ์ด ET-PC8255 จะประกอบไปด้วย 2 ส่วนใหญ่ๆ ก้คือ ส่วน IC 8255 ซึ่งเป็น IC ทำหน้าที่เป็น Input, Output PORT และส่วนของวงจร IC Decode (เลือกตำแหน่งของ Port 8255) คือ IC 74LS688, 74LS139 และ DIP SW.

PC HARDWARE I/O MAP

8088 Class Systems

Address	Function
ØØØ–ØØF	DMA Controller (8237A)
Ø2Ø–Ø21	Interrupt controller (8259A)
Ø4Ø–Ø43	Timer (8253)
Ø6Ø–Ø63	PPI (8255A)
Ø8Ø–Ø83	DMA page register (74LS612)
ØAØ–ØAF	NMI – Non Maskable Interrupt
2ØØ–2ØF	Game Port Joystick controller
21Ø–217	Expansion Unit
2E8–2EF	COM4: Serial Port
2F8–2FF	COM2: Serial Port
3ØØ–31F	Prototype Card
32Ø–32F	Hard Disk
378–37F	Parallel Printer Port 1
38Ø–38F	SDLC
3BØ–3BF	MDA – Monochrome Adapter and printer
3DØ–3D7	CGA – Color Graphics Adapter
3E8–3EF	COM3: Serial Port
3FØ–3F7	Floppy Diskette Controller
3F8–3FF	COM1: Serial Port

80286 /386/486 Class Systems

Address	Function
ØØØ–Ø1F	DMA Controller #1 (8237A–5)
Ø2Ø–Ø3F	Interrupt controller #1 (8259A)
Ø4Ø–Ø5F	Timer (8254)
Ø6Ø–Ø6F	Keyboard (8Ø42)
Ø7Ø–Ø7F	NMI – Non Maskable Interrupt & CMOS RAM
Ø8Ø–Ø9F	DMA page register (74LS612)
ØAØ–ØBF	Interrupt controller #2 (8259A)
ØCØ–ØDF	DMA Controller #2 (8237A)
ØFØ–ØFF	8Ø287 Math Coprocessor
1FØ–1F8	Hard Disk
2ØØ–2ØF	Game Port Joystick controller
258–25F	Intel Above Board
278–27F	Parallel Printer Port 2
2E8–2EF	COM4: Serial Port
2F8–2FF	COM2: Serial Port
3ØØ–31F	Prototype Card
378–37F	Parallel Printer Port 1
38Ø–38F	SDLC or Bisynchronous Comm Port 2
3AØ–3AF	Bisynchronous Comm Port 1
3BØ–3BF	MDA – Monochrome Adapter
3BC–3BE	Parallel Printer on Monochrome Adapter
3CØ–3CF	EGA – Reserved
3DØ–3D7	CGA – Color Graphics Adapter
3E8–3EF	COM3: Serial Port
3FØ–3F7	Floppy Diskette Controller
3F8–3FF	COM1: Serial Port

PC Hardware

ตารางที่ 3 PC HARDWARE I/O MAP

Port ของ 8255 ที่เหลือนั้นทำหน้าที่อะไร เป็น Input หรือ Output Port เราจะต้องเป็นผู้กำหนด Control Code Port ควบคุม

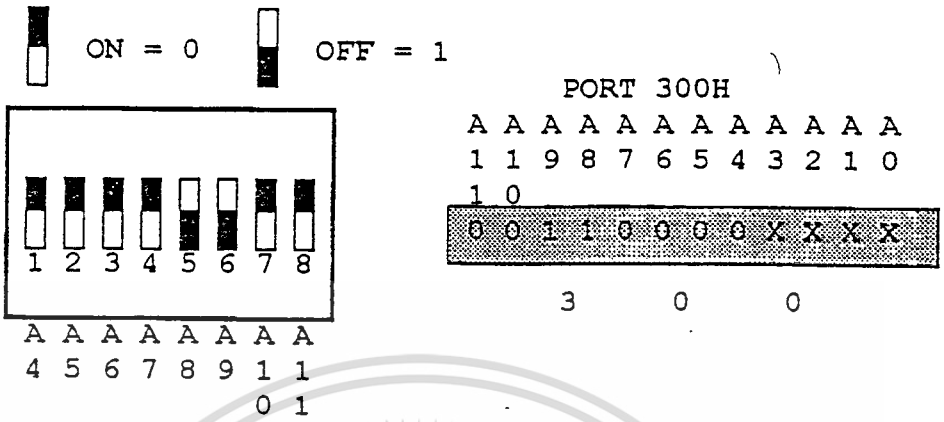
การ Decord Port

Decode Port 8255 บนการ์ดเราจะใช้ IC TTL 74LS688, IC TTL 74LS139 และ DIP SW. 8 PIN เป็นวงจร Decode เพื่อให้สามารถปรับ Set DIP SW. ตั้งตำแหน่งเบอร์ Port ของการ์ด ได้ โดยในการปรับ DIP SW. นั้นจะต้องไม่ไปตรงกับตำแหน่ง Port ของเครื่องคอมพิวเตอร์ PC ด้วย ดังรูป

โดยการ์ด ET-PC8255 จะใช้ตำแหน่ง Port 12 Port ต่อการ์ด Decode Port

XX0H	PORT A (#1)
XX1H	PORT B (#1)
XX2H	PORT C (#1)
XX3H	CONTROL PORT (#1)
XX4H	PORT A (#2)
XX5H	PORT B (#2)
XX6H	PORT C (#2)
XX7H	CONTROL PORT (#2)
XX8H	PORT A (#3)
XX9H	PORT B (#3)
XXAH	PORT C (#3)
XXBH	CONTROL PORT (#3)

เราตั้งเบอร์ Decoed Port ได้โดยการปรับ DIP SW. ซึ่งก็มีค่าเท่ากับค่า Adderss นั้นๆ เช่น เราตั้งตำแหน่ง 300H จะ Set DIP SW. ดังนี้



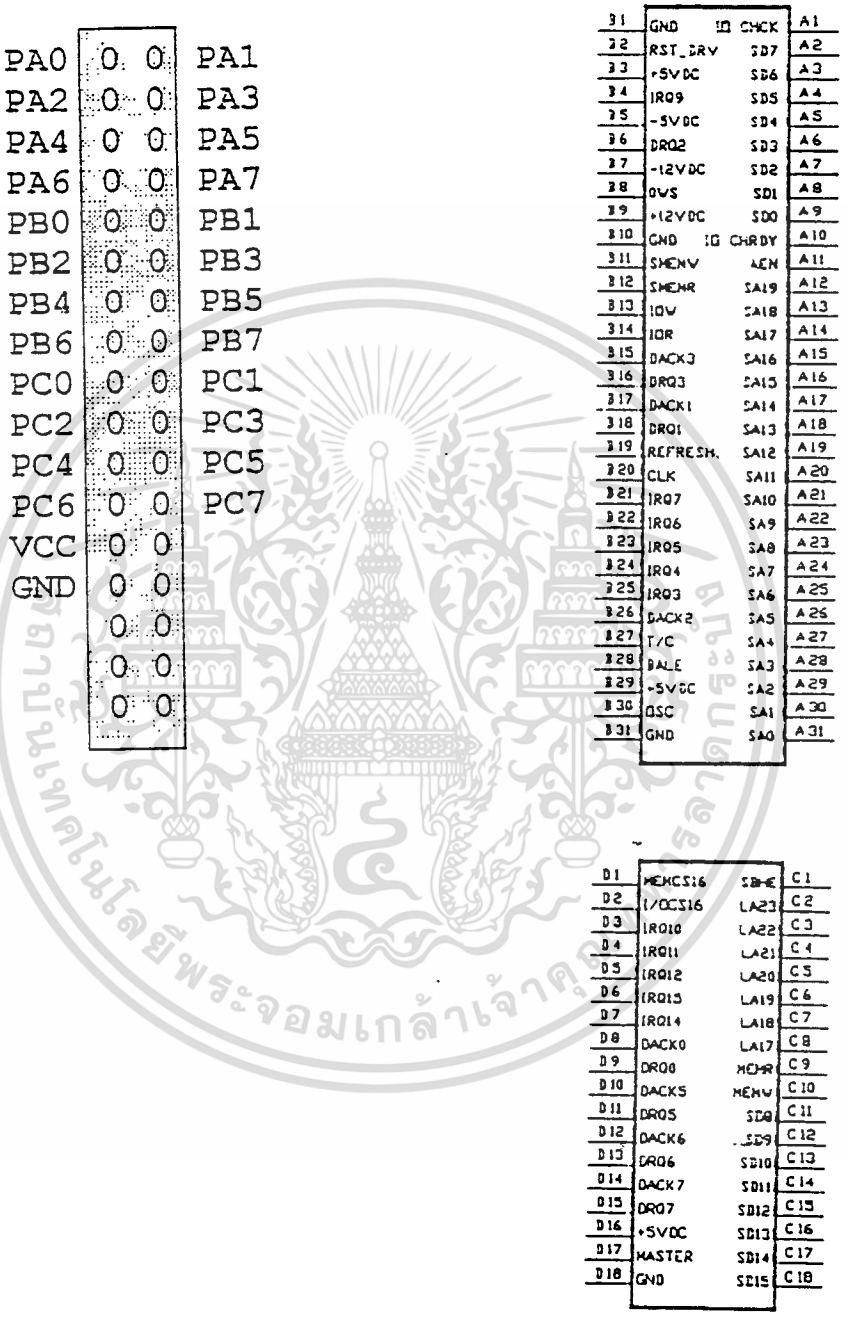
รูปที่ 41 การ Set Dip Sw.

<u>ใน Project นี้ โปรแกรมจะเริ่มต้น Port ที่หมายเลข 300H</u>

ตัวอย่างโปรแกรม

```
# include (dos.h)
# include (conio.h)
# include (stdio.h)
main()
        {       char ch = 0;
                outportp(0x303,0x82);/*set control port*/
                outportp(0x300,0xFF);/*out*/
                ch = inportp(0x301);/* in */
                clrscr();
                printf("sent port A = %.4x\n",0xFF);
                printf("read port B = %.4x\n",ch);
                getch();
        }
```

34 PIN I/O BUS



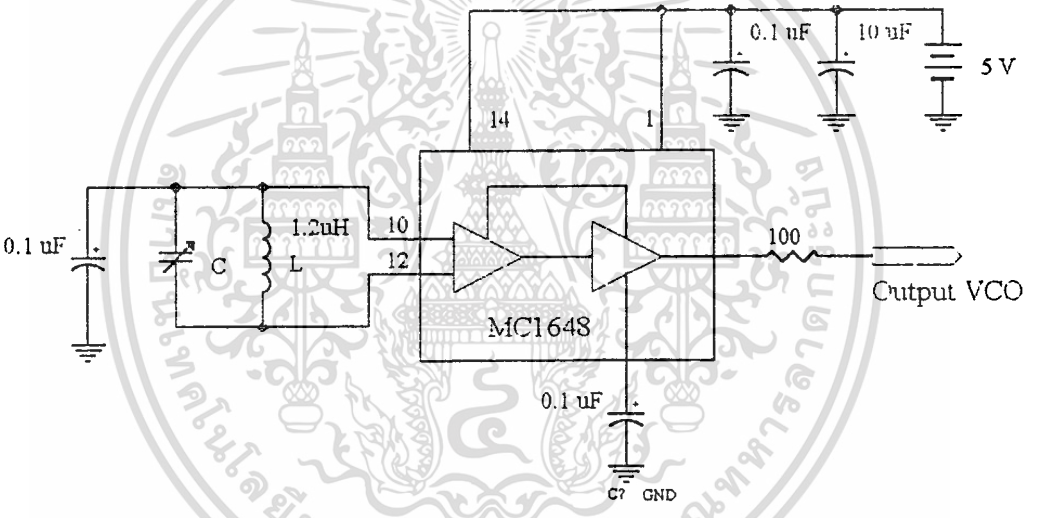
รูปที่ 42 PIN I/O BUS

บทที่ 4

ผลการทดลอง

การทดลองในส่วนของ VCO

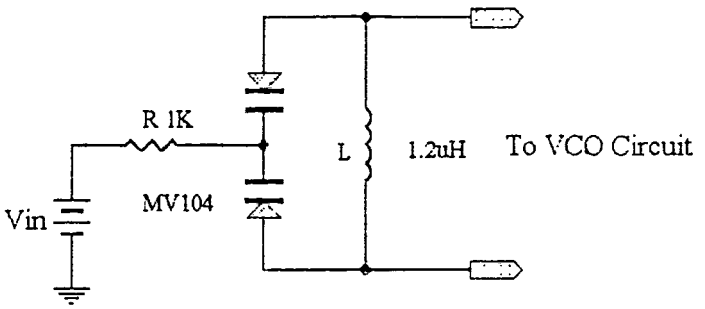
ในส่วนของ Voltage Control Oscillator (VCO) เราจะนำไอซีเบอร์ MC1648 ของบริษัทโมโตโรล่า มาทำการประยุกต์ใช้ทำเป็น VCO โดยที่เราทำการต่อวงจรแทงค์แบบขนานเข้าที่อินพุทระหว่างขา 10 และขา 12 ของ MC1648 และทำการจัดไบอัสให้กับวงจรตามรูป



รูปที่ 43 วงจร Voltage Control Oscillator (VCO)

ในส่วนของวงจรแทงค์จูนเราใช้ Varicap Diode เป็นตัวเก็บประจุปรับค่า ได้ตามแรงไฟ DC ซึ่งในที่นี้เราใช้เบอร์ MV104 เป็น Silicon Epicap Diode สามารถรับแรงไฟไบอัสกลับ (Reverse Voltage ; Vr) ได้ถึง 32 V จากรูปข้างล่าง ถ้าเราออกแบบให้ Reverse Voltage อยู่ในช่วง 0 - 15 V เราจะได้ค่าความจุ ของ Varicap Diode ในช่วง 20 pF - 80 pF ดังรูป

จากนั้นจะนำเอา varicap มาต่อขนานกับอินดัคเตอร์ L เป็นวงจรแทงค์จูนแบบขนานดังรูป



รูปที่ 44 การออกแบบ วงจร L,C tank OSC

ในการทดลองในส่วนของ VCO ในส่วนของภาค Voltage Control Oscillator (VCO) เราจะนำไอซีเบอร์ MC1408 ของบริษัทโมโตโรล่า มาทำการประยุกต์ใช้ทำเป็น VCO โดยที่เราทำการต่อวงจรแทงค์แบบขนานเข้าที่อินพุทระหว่างขา 10 และ 12 ของไอซี MC1408 และทำการจัดไบอัสให้กับวงจร

ในส่วนของวงจรแทงค์จูนใช้ Varicap Diode เป็นตัวเก็บประจุปรับค่าได้ตามแรงไฟDC ซึ่งในที่นี้เราใช้เบอร์ MV104 เป็น Silicon Epicap Diode ซึ่งสามารถรับแรงไฟไบอัส (Reverse Voltage: Vr) ได้ถึง 32 V ถ้าเราออกแบบให้ Reverse Voltage อยู่ในช่วง 0-15 V เราจะได้ค่าความจุของ Varicap Diode ในช่วง 20pF-80pF

<u>การออกแบบวงจร VCO</u>

เราต้องการความถี่เอาท์พุท VCO ในช่วง 15-20 MHz โดยใช้ไอซีเบอร์ MC1648 เป็น วงจรผลิตความถี่ออสซิเลเตอร์ ความถี่ของวงจรจะหาได้จากสมการ

$$f_0 = 1/\{2\pi\sqrt{L(Cd + Cs)}\}$$

เมื่อ ; Cd คือ ค่าความจุของวาแรกเตอร์

Cs คือ ค่าความจุภายใน MC1648 มีค่าประมาณ 6 pF

จะได้

ที่ Vin = 0 V

$$f_{0\,min} = 1/\{2\pi\sqrt{L_{max}\,(Cd_{max} + Cs)}\}$$

$f_{0\,min} = 15$ MHz

$Cs = 6$ pF

$Cd_{max} = 80$ pF

ได้ $L = 1.2\ \mu H$

ที่ Vin = 15 V

$Cs = 6$ pF

$Cd_{min} = 20$ pF

ได้ $L = 1.02$ μH

ดังนั้น $f_{0\ man} = 30$ MHz

ออกแบบ LPF

ขั้นตอนการออกแบบ

1. เลือกความถี่อ้างอิง (Fref)
2. คำนวณย่านของตัวหาร โดย Ntot = fmax/fref
3. เลือกค่า ζ ในที่กำหนดให้ $\zeta = 0.707$
4. หาค่า Wn

   โดยเลือกค่า Wn.t จากกราฟการตอบสนองของ PLL TYPE2

   กำหนดค่า $Wn.t = 4.5$; $t = 1$ mS

   $Wn = 4500$ rad/S
5. หาค่า Kd (Phase Detector Gain) = $VDD/2\pi = 0.796$ V/rad
6. หาค่า Kv (VCO Gain) = $2\pi\Delta Fvco/\Delta Vvco$
7. ออกแบบ LOOP FILTER

   การใช้สูตรนั้น[(KdKv)/(NmaxC(Wn)^2)]ต้องพิจารณาจากวงจรว่าเป็นได้ในที่นี้เป็นแบบ Active Filter

   - กำหนดค่า
   - หาค่า R1 จากสูตร $R1 = [(KdKv)/(NmaxC(Wn)^2)]$
   - หาค่า R2 จากสูตร $R2 = 2\zeta/(Wn.C)$
   - หาค่า Cc จากสูตร $Cc = 0.8/(R1*Wn)$

การออกแบบที่ใช้ในวงจร

1. กำหนดค่า fref = 20 KHz
2. ที่ f = 15-30 MHz

จะได้ Ntot = fmax/fref

= 30 MHz/20 KHz

= 1500

3. กำหนด C = 33nF
4. $R1 = [(KdKv)/(NmaxC(Wn)^2)]$

$= (0.796*3.552*10^6)/(1000*33\eta*(4500^2))$

$= 4.31\ k\Omega$

5. $R2 = 2\zeta/(Wn.C)$

$= 2*0.707/(4500*33\eta)$

$= 9.5\ k\Omega$

6. $Cc = 0.8/(R1*Wn)$

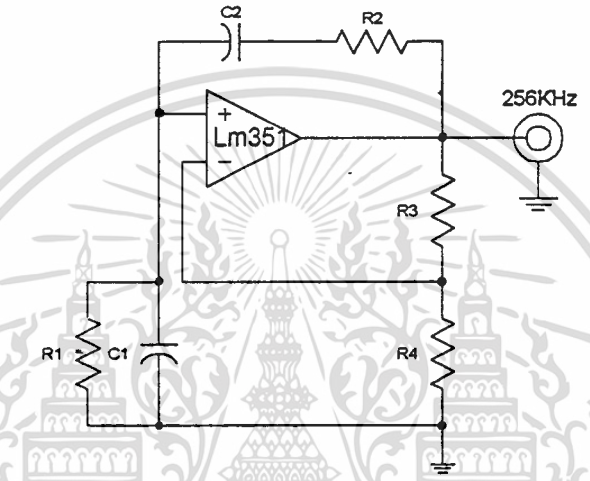
$= 0.8/(4.31k*4500)$

$= 41.3\ nF$

<u>การสร้างสัญญาณทดสอบ AM แบบ Balance Modulation</u>

การสร้างสัญญาณข้อมูล 256 kHz

ในวงจรนี้จะใช้การสร้างสัญญาณให้ได้ความถี่ 256 kHz โดยใช้ Op-Amp และ Resistor, Capacitor เป็นตัวกำเนิดความถี่



รูปที่ 45 ผลิตความถี่ 256 kHz

<u>การคำนวณ</u>

กำหนดให้ R1 = R2 = R และ C1 = C2 = C

$C = 0.001\ \mu F$

สูตร $f = 1/\{2\pi RC\}$

$R = 1/\{2\pi * 256kHz * 0.001\ \mu F\}$

$= 621\ \Omega$

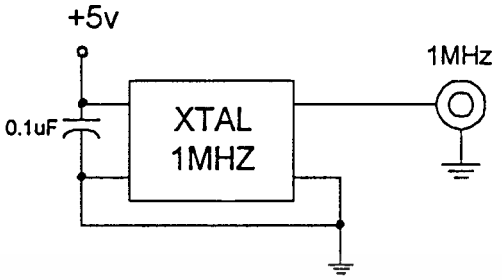
เลือกอัตราการขยาย = 2

$= Rf/Rin$

$Rf = R3 = 30\ k\Omega$

$Rin = R4 = 15\ k\Omega$

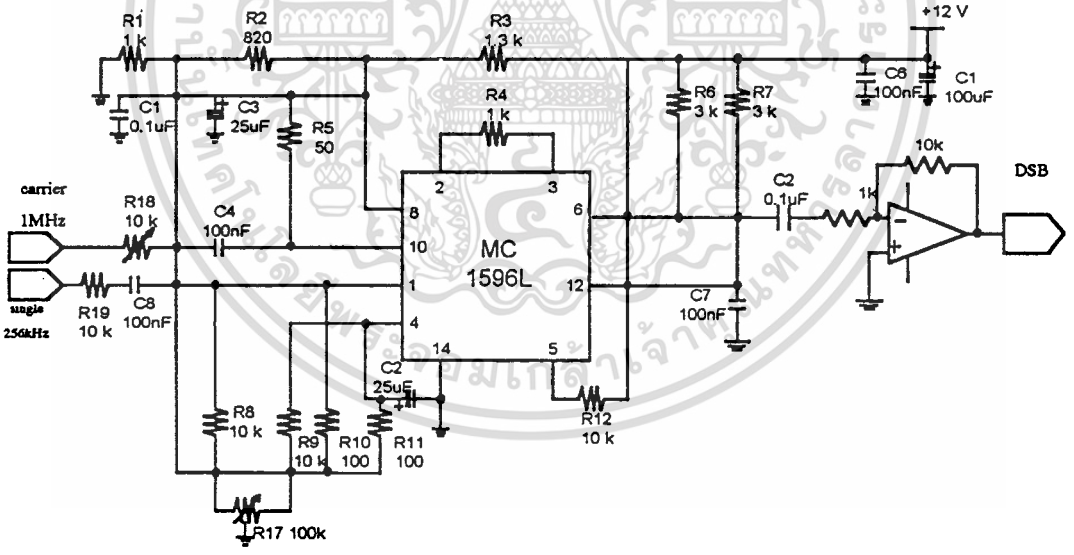
<u>การสร้างสัญญาณ 1MHz ใช้ XTAL 4 ขา</u>



รูปที่ 46 วงจรผลิตความถี่ 1 MHz

<u>วงจรบาลานซ์มอดดูเลเตอร์</u>

รายละเอียดของวงจรบาลานซ์มอดดูเลเตอร์ ดังแสดงไว้ในรูป



รูปที่ 47 วงจรบาลานซ์มอดดูเลเตอร์

ในการออกแบบ Balance Modulation จะเป็นแบบ suppress carrierr โดย การปรับความต้านทาน 100K ระหว่างขา 1 กับ ขา 4 ของ MC1596 สัญญาณออกทางขา 6

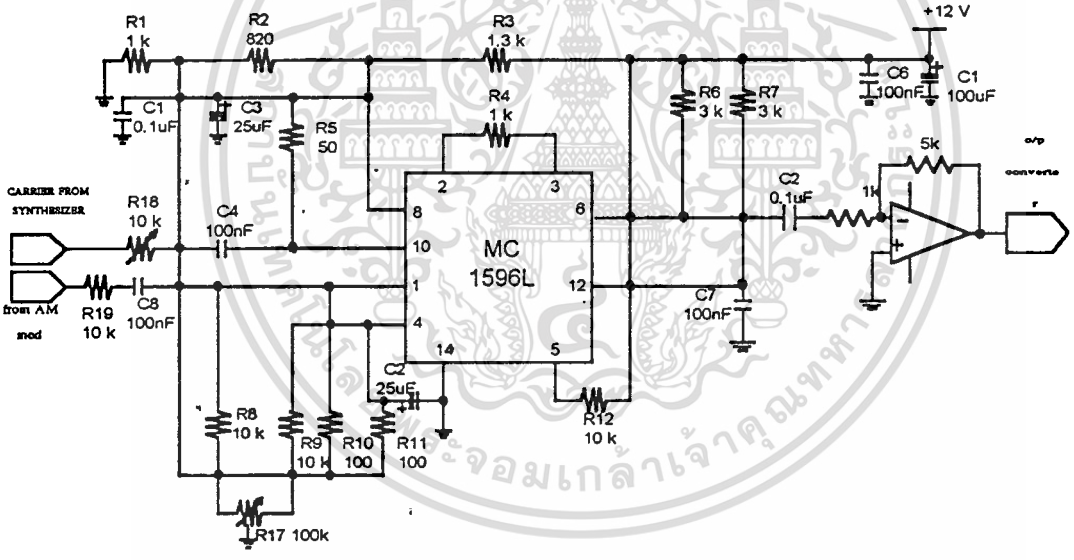
ข้อควรคำนึงในวงจรส่วนนี้คือ ข้อกำหนดจากData Sheet ของ IC MC1596 ที่ได้ระบุไว้ว่าขนาดของสัญญาณที่จะนำมามอดดูเลทควรมีค่าประมาณ 400 mV_{p-p} และขนาด

ของสัญญาณพาหะควรมีค่าประมาณ 100mVrms ดังนั้นสัญญาณทั้งสองก่อนที่จะเข้าไปคูณกันต้องผ่านตัวต้านทานปรับค่าได้เสียก่อน เพื่อจะได้ควบคุมระดับสัญญาณทั้งสองให้เป็นไปตามข้อกำหนดดังกล่าว

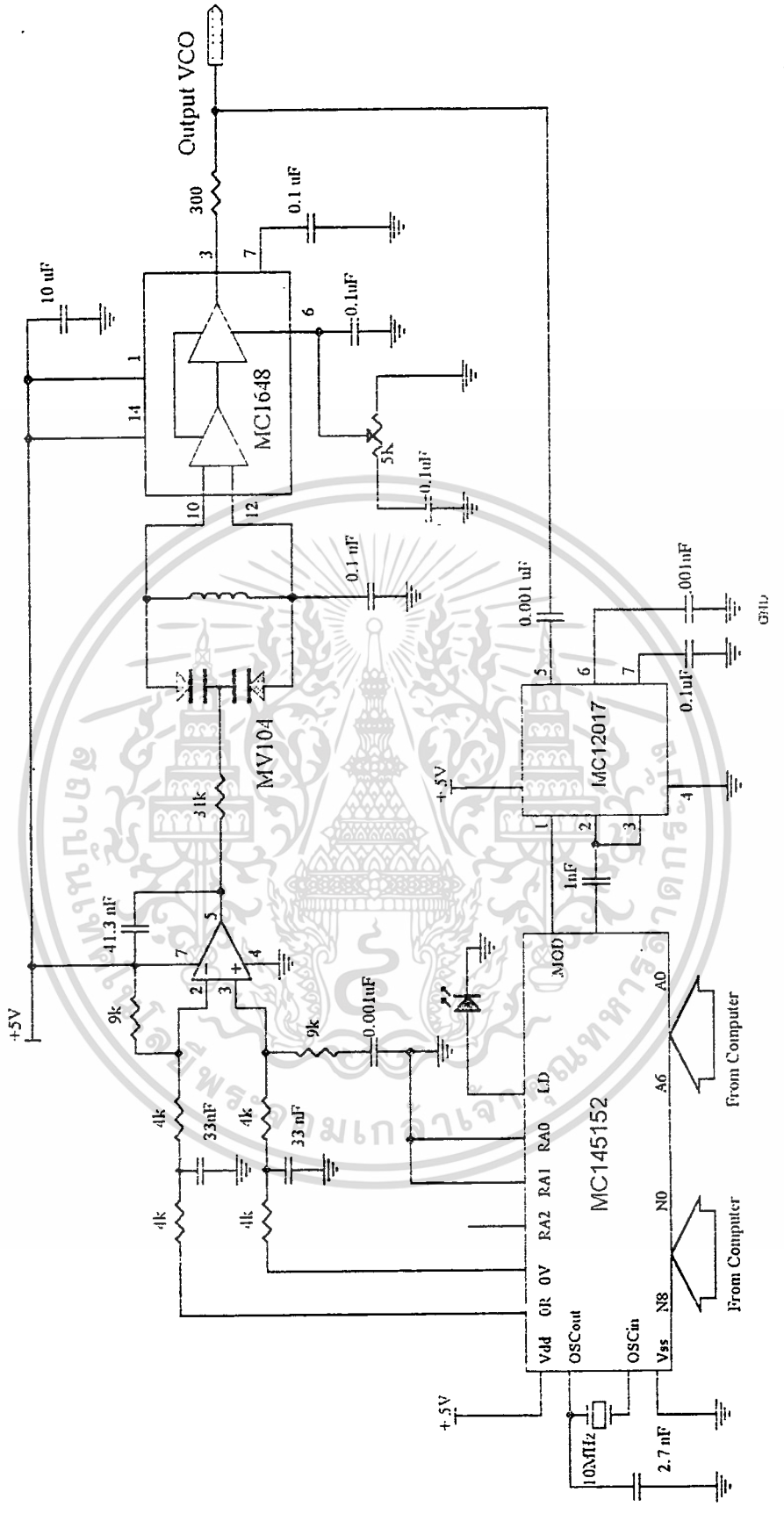
สัญญาณที่ได้จากวงจรบาลานซ์มอดดูเลเตอร์นั้นมีขนาดเล็กมาก จึงต้องผ่าน วงจรขยายอีกครั้งเพื่อจะได้สัญญาณที่มีขนาดสูงขึ้น

<u>วงจรแปลงความถี่ (Mixer OR Convertor)</u>

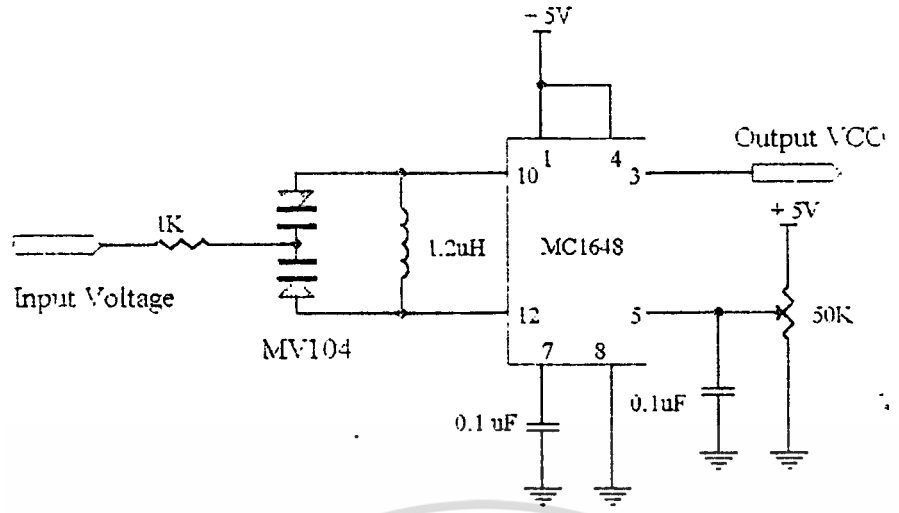
ในการออกแบบในโครงงานนี้จะใช้ IC MC1596 ดูจาก Data Sheet แล้วสามารถนำมาทำวงจรแปลงความถี่ (Mixer) ได้ เพราะว่าที่เอาท์พุทของวงจรความถี่ที่ได้ออกมามีทั้งความถี่ผลบวก (Up Convertor) และความถี่ผลต่าง (Down Convertor) ซึ่งตรงกับหลักการของการแปลงความถี่ (Mixer)



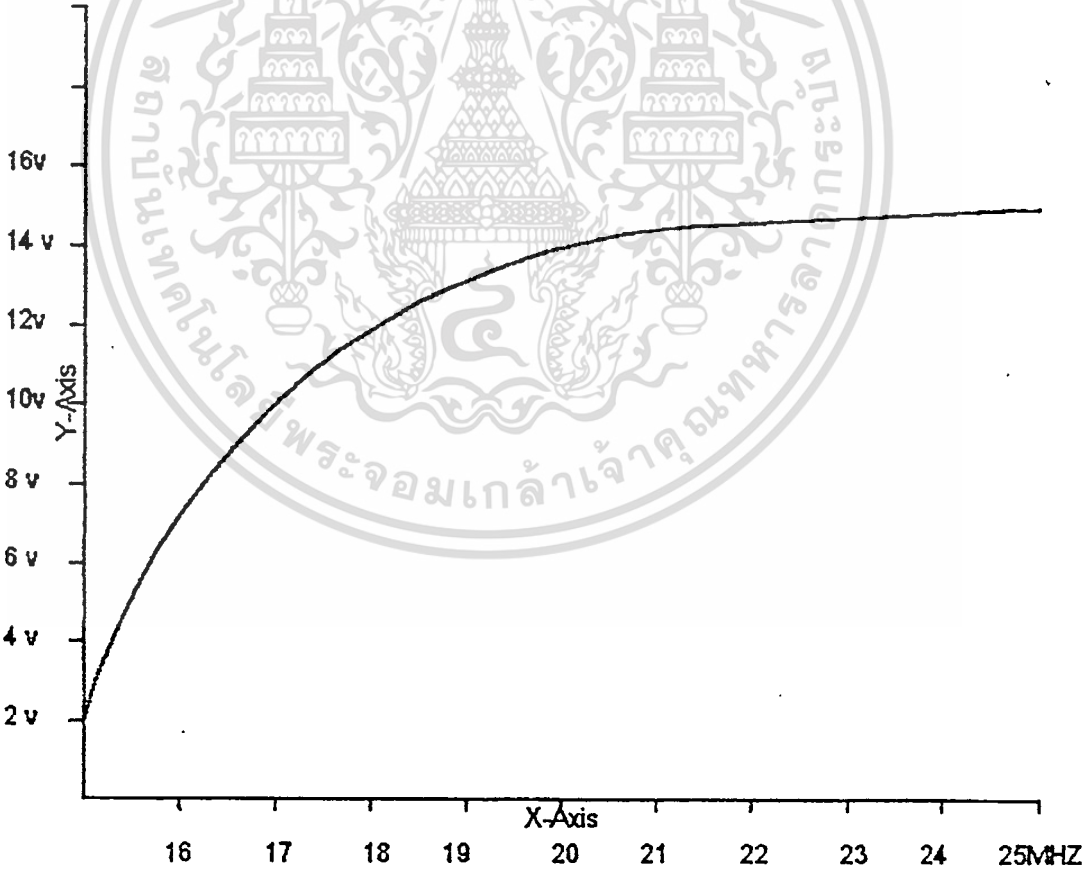
รูปที่ 48 วงจรแปลงความถี่ (Mixer or Convertor)



รูปที่ 49 วงจร Phase Lock Loop (PLL)



รูปที่ 50 วงจร VCO



รูปที่ 51 กราฟแสดงความสัมพันธ์ระหว่างแรงดันของ VCO และความถี่

บทที่ 5

สรุปผลการทดลอง

ผลการทดลอง

วงจรสามารถผลิตความถี่ได้ในช่วง 15-20 MHz แต่ในช่วงความถี่สูงระดับของสัญญาณออกมามีขนาดน้อยและสัญญาณก็ไม่ค่อยนิ่ง เนื่องจากที่ความถี่สูงจะมีสัญญาณรบกวนมาก

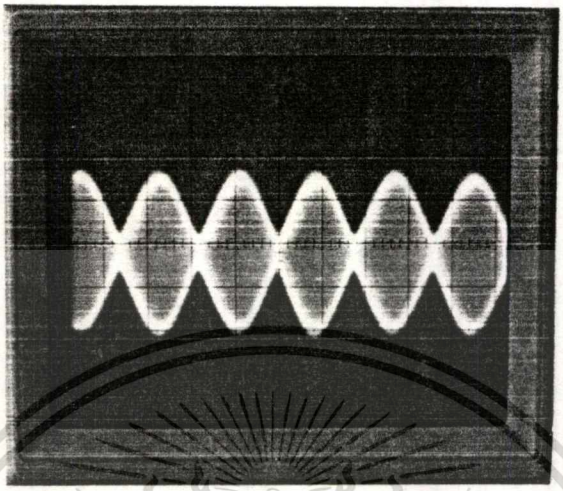
ปัญหาที่เกิดขึ้นในการทดลอง

1. ขาดแคลนเครื่องมือในการทำงาน ต้องแบ่งการใช้งานกับอีกหลายๆ กลุ่ม สเปคตรัม
2. เสียเวลาศึกษาโปรแกรมภาษา C
3. ปัญหาที่พบมากในการทดลองก็คือ เรื่องรูปร่างของสัญญาณในบางช่วงความถี่รูปของสัญญาณจะมีรูปร่างที่ผิดไปและมีขนาดสัญญาณต่ำที่ความถี่สูง
4. ปัญหาของความถี่ไม่ตรงกับที่โปรแกรมไว้ และบางที่ก็ไม่เปลี่ยนความถี่เมื่อโปรแกรมให้ความถี่เปลี่ยนไปตามที่กำหนด
5. ปัญหาในการรับสัญญาณรบกวนจากภายนอก คือ เมื่อกลุ่มโปรเจกอื่นทดลองส่งความถี่ในย่านเดียวกันหรือมากกว่า จะทำให้รูปสัญญาณเอาท์พุทหายไปหรือมีสัญญาณจากภายนอกมาปรากฏด้วยจนไมาสามารถวัดสัญญาณได้เลย
6. IC ที่ทำวงจรแปลงความถี่ (Convertor) ที่ความถี่สูงจะเริ่มเพี้ยน

แนวทางการพัฒนา

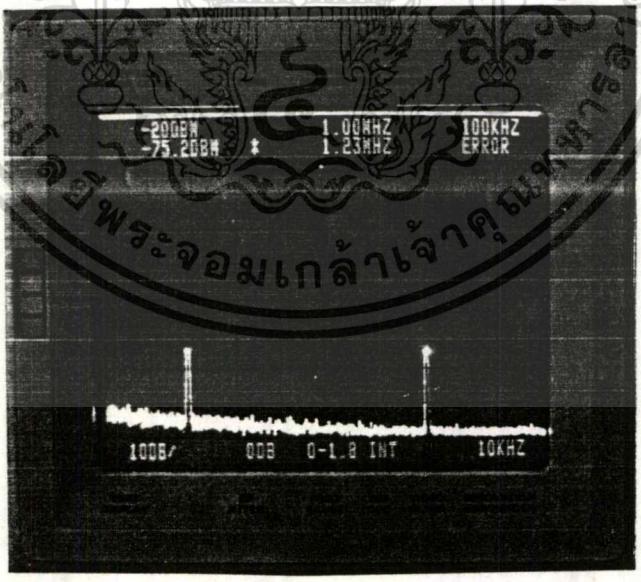
1. ให้มีช่องสัญญาณให้มากกว่านี้
2. เมื่อปิดเครื่องคอมพิวเตอร์แล้วข้อมูลไม่เปลี่ยนแปลง
3. ใช้ได้กับทุกจอภาพ
4. พัฒนาโปรแกรมให้สามารถ RUN ใน Windows ได้

รูปของสัญญาณเอาท์พุทที่ความถี่ต่างๆ

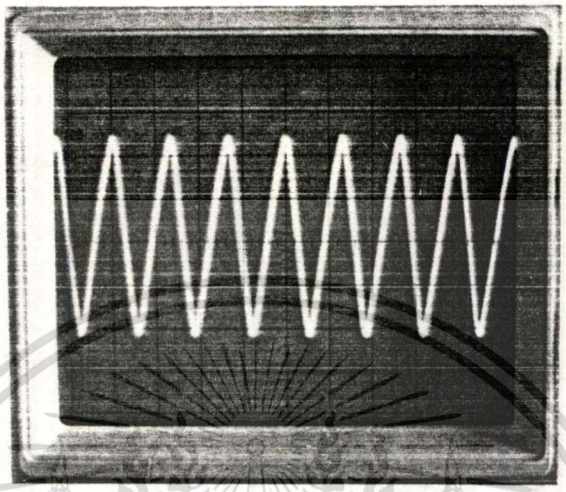


รูปที่ 52 รูปของสัญญาณทดสอบ AM DBS Signal = 256 kHz, fc = 1 MHz

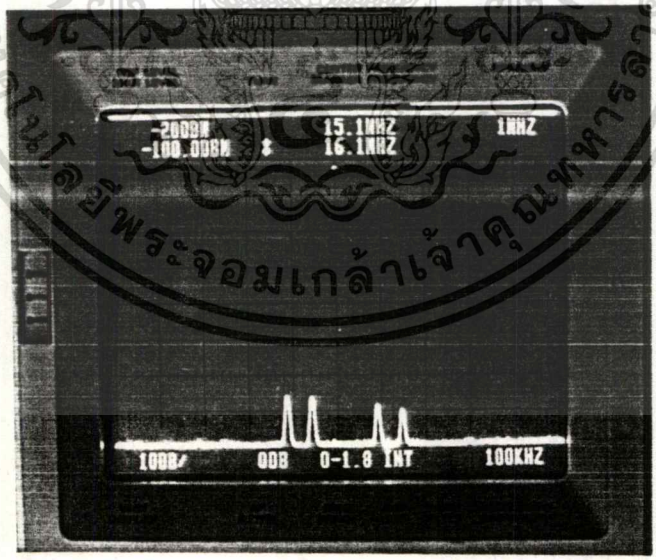
(1 μs/DIV, 50 mV/DIV)



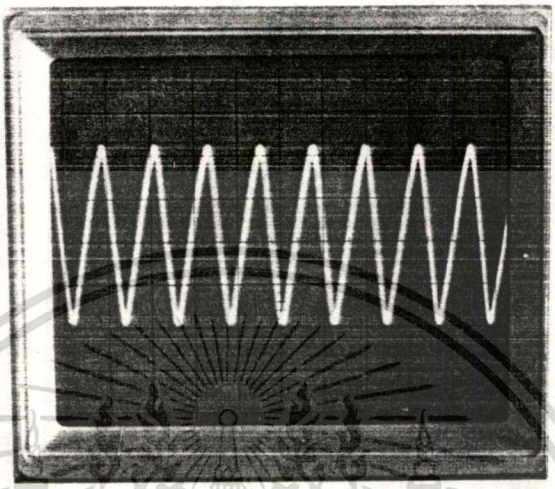
รูปที่ 53 รูปของสัญญาณ SIDEBAND AM DBS



รูปที่ 54 รูปของสัญญาณ fc = 15.1 MHz (0.5 μs/DIV, 50 mV/DIV)

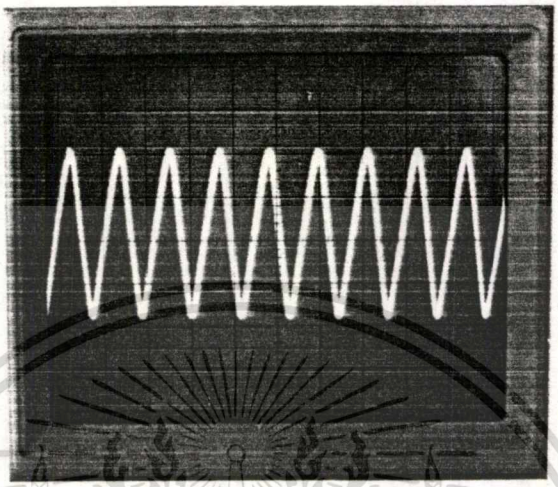


รูปที่ 55 รูปของสัญญาณแปลงความถี่ (Convertor) fc = 15.1 MHz กับ AM DBS

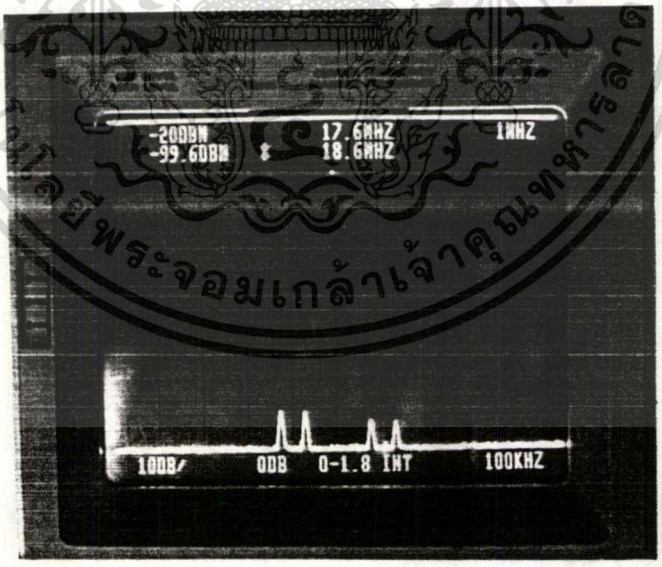


รูปที่ 56 รูปของสัญญาณ fc = 16.3 MHz (0.5 μs/DIV, 50 mV/DIV)

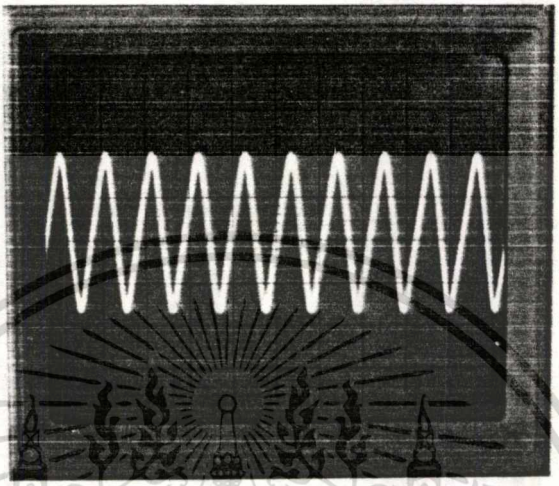
รูปที่ 57 รูปของสัญญาณแปลงความถี่ (Convertor) fc = 16.3 MHz กับ AM DBS



รูปที่ 58 รูปของสัญญาณ fc = 17.6 MHz (0.5 μs/DIV, 50 mV/DIV)



รูปที่ 59 รูปของสัญญาณแปลงความถี่ (Convertor) fc = 17.6 MHz กับ AM DBS



รูปที่ 60 รูปของสัญญาณ fc = 18.9 MHz (0.5 μs/DIV, 50 mV/DIV)

รูปที่ 61 รูปของสัญญาณแปลงความถี่ (Convertor) fc = 18.9 MHz กับ AM DBS

เอกสารอ้างอิง

ธันวา ศรีประโมง.การเขียนโปรแกรม ภาษาซี สำหรับวิศวกรรม,โครงการตำราวิชาการมหานคร, กรุงเทพ, 2537.

ประกิจ ตังติสานนท์.วิ ศวกรรมการสื่อสารไฟฟ้าอิเล็กทรอนิกส์, กรุงเทพ, 2527.

มนตรี พจนารถลาวัณย์.การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี,ซีเอ็ดยูเคชั่น, กรุงเทพ, 2537.

ยืน ภู่วรวรรณ. ทฤษฎีและการประยุกต์ไมโครโปรเซสเซอร์ Z-80, ซีเอ็ดยูเคชั่น, กรุงเทพ, 2536.

วิทยา เรืองพรวิสุทธิ์.คู่มือโปรแกรมภาษา C สำหรับผู้เริ่มต้น,ซีเอ็ดยูเคชั่น, กรุงเทพ, 2537.

ศิวัฒน์ ศิวะบวร, พรชัย จักรดำรงค์,จิรศักดิ์ ชัยวิริยะกุล.การประยุกต์ใช้งานภาษาซี, ซีเอ็ดยูเคชั่น, กรุงเทพ, 2536.

สุชาติ กังวารจิตต์. เครื่องรับส่งวิทยุและระบบสื่อสาร, ซีเอ็ดยูเคชั่น, กรุงเทพ, 2532.

สมพันธุ์ ชาญศิลป์. ภาษาซี (C-Language). หน่วยสารบรรณ งานบริการและธุรการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น, ขอนแก่น, 2537.

ETT. ETT-PC8255, ETT, 2538

ภาคผนวก

โปรแกรมใช้งาน

```
/*************************************************************************
        Menu program    TX   RX
        Run in graphphic mode  VGA 640*480 monitor
Programer  ROOM N(AGODE) DEPARTMENT OF INDUSTRIAL  TECHNOLOGY
        FACULTY OF ENGINEERING
        King Mongkut Institute of Technology Ladkrabang
 *************************************************************************/

#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <process.h>
#include <alloc.h>

#define ESC     0x1b          /* Define the escape key      */
#define ENTER   0x0d          /* Defin the enter key        */
#define SY      150           /* Define the value start y   */
#define EY      350           /* Define the value end y     */
#define Po1     0x303     /* Control port 8255 1 */
#define Pa1     0x300     /* Port A 8255 1 */
#define Pb1     0x301     /* Port B 8255 1 */
#define Pc1     0x302     /* Port C 8255 1 */
#define Po2     0x307     /* Control port 8255 2 */
#define Pa2     0x304     /* Port A 8255 2 */
```

```
#define Pb2     0x305     /* Port B 8255 2 */
#define Pc2     0x306     /* Port C 8255 2 */
struct PTS   {  int x, y;  }; /* Structure to hold vertex points   */

int    GraphDriver;          /* The Graphics device driver          */
int    GraphMode;            /* The Graphics mode value             */
double AspectRatio;          /* Aspect ratio of a pixel on the screen*/
int    MaxX, MaxY;           /* The maximum resolution of the screen */
int    MaxColors;            /* The maximum # of colors available    */
int    ErrorCode;            /* Reports any graphics errors          */
int year,semonth,z,*image,xx,yy,m,d,menuy,ans;
int p0,p2,a[10];
int xa1,xa2,xa3,xa4,ya1,ya2,ya3,ya4,*image1;
int far *image2;

char ch1[][10] = {{""},{""},{""},{""},{""},
                  {"    1"},    {"    2"},    {"    3"},
                  {"    4"},    {"    5"}     };

char ch2[][20] =
      { {" f = 18.5 MHz  "}, {" f = 19.5 MHz  "},
        {" f = 20.5 MHz  "}, {" f = 21.5 MHz  "},
        {" f = 22.5 MHz  "},
        {"  Tx  1 "}, {"  Tx  2 "},
        {"  Tx  3 "}, {"  Tx  4 "},
        {"  Tx  5 "}
        };
```

```
char ch[][30] =
        { {" f = 18 MHz  "},    {" f = 19 MHz  "},
          {" f = 20 MHz  "},    {" f = 21 MHz  "},
          {" f = 22 MHz  "},
          {"chang1"},   {"chang2"},
          {"chang3"},   {"chang4"},
          {"chang5"}
          };

/*                                                      */
/*      Function prototypes                             */
/*                                                      */

void Initialize(void);
void BodyMenu(void);
void changetextstyle(int font, int direction, int charsize);
void setregion(int x1,int y1,int x2,int y2);
void Esc_ok(int x1,int y1,int x2,int y2,int numb);
void Choice_r(int x1,int y1,int x2,int y2);
void Choice_t(int x1,int y1,int x2,int y2);
void setpi(int x1,int y1,int color);
void Choicemain(int x1,int y1,int x2,int y2);
void mal(int x1,int y1,int x2,int y2);
void put(void);
void mal2(int x1,int y1,int x2,int y2);
void put1(void);
void changelinestyle(int linestyle,int upattern,int thickness);
int getmouse1(void);
struct palettetype palette;
```

```
/*      INITIALIZE: Initializes the graphics system and reports        */
/*      any errors which occured.                                      */
/*                                                                     */

void Initialize(void)
{
  int xasp, yasp;                  /* Used to read the aspect ratio*/

  GraphDriver = DETECT;            /* Request auto-detection       */
  initgraph( &GraphDriver, &GraphMode, "" );
  ErrorCode = graphresult();       /* Read result of initialization*/
  if( ErrorCode != grOk ){         /* Error occured during init    */
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }

  getpalette( &palette );          /* Read the palette from board  */
  MaxColors = getmaxcolor() + 1;   /* Read maximum number of colors*/

  MaxX = getmaxx();
  MaxY = getmaxy();                /* Read size of screen          */

  getaspectratio( &xasp, &yasp );  /* read the hardware aspect     */
  AspectRatio = (double)xasp / (double)yasp; /* Get correction factor   */

}
```

```
/*      CHANGETEXTSTYLE: similar to settextstyle, but checks for        */
/*      errors that might occur whil loading the font file.             */
/*                                                                      */

void changetextstyle(int font, int direction, int charsize)
{
  int ErrorCode;

  graphresult();                    /* clear error code              */
  settextstyle(font, direction, charsize);
  ErrorCode = graphresult();        /* check result                  */
  if( ErrorCode != grOk ){          /* if error occured              */
    closegraph();
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }
}

/*                                                                      */

void changelinestyle(int linestyle,int upattern,int thickness)
{
  int ErrorCode;

  graphresult();                    /* clear error code              */
  setlinestyle(linestyle,upattern,thickness);
  ErrorCode = graphresult();        /* check result                  */
  if( ErrorCode != grOk ){          /* if error occured              */
```

```
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }
}


/*      BODYMENU : this is show menu body                */
/*                                                       */

void BodyMenu(void)
{

  cleardevice();
  setviewport( 0, 0, MaxX, MaxY, 1 );   /* Open port to full screen      */
  mouseoff();
  setfillstyle(1,CYAN);
  bar(0,0,MaxX,MaxY);
  chan_set_text(MAGENTA);
  outtextxy(MaxX/2,460/*MaxY/2-40*/,
     " KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG ");
  outtextxy(MaxX/2,10," PROGRAM PLL SYNTHESIZER (TX & RX) ");

  windows3(310,210,310,210,BLUE,GREEN,WHITE,LIGHTGRAY,"MAIN MENU");
  mal(150,150,150,150);
  windows1(150,150,150,150,1,LIGHTGRAY,WHITE,CYAN,CYAN,"PROGRAM
VERSION 1.0");
  mouseoff();
  about1();
```

```
  mouseon();
  Esc_ok(150,150,150,150,1);
  put();
  Choicemain(310,210,310,210);
}

void mal(int x1,int y1,int x2,int y2)  /*     mall  memory    */
{
int x3,x4,y3,y4/*,*image*/;

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;

mouseoff();
image=malloc(imagesize(x3-1,y3-1,x4+1,y4+1));
getimage(x3-1,y3-1,x4+1,y4+1,image);
image1=image;
xa1=x3;
xa2=x4;
ya1=y3;
ya2=y4;
mouseon();
}
```

```
/*          mal2   far  memory  */

void mal2(int x1,int y1,int x2,int y2)
{
int x3,x4,y3,y4/*,*image*/;

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;

mouseoff();
image2=farmalloc(imagesize(x3-1,y3-1,x4+1,y4+1));
getimage(x3-1,y3-1,x4+1,y4+1,image2);
/*image1=image;*/
xa3=x3;
xa4=x4;
ya3=y3;
ya4=y4;
mouseon();
}

void put(void)    /* put memory   */
{
int x3,x4,y3,y4/* *image*/;

mouseoff();
x3=xa1;
```

```
x4=xa2;
y3=ya1;
y4=ya2;
image=image1;
putimage(x3-1,y3-1,image,0);
setregion(0,0,MaxX,MaxY);
free(image);
mousegoto(MaxX/2,MaxY/2);
mouseon();
}

void put1(void)   /* put far memory */
{
int x3,x4,y3,y4/* *image*/;

mouseoff();
x3=xa3;
x4=xa4;
y3=ya3;
y4=ya4;
putimage(x3-1,y3-1,image2,0);
setregion(0,0,MaxX,MaxY);
farfree(image2);
mousegoto(MaxX/2,MaxY/2);
mouseon();
}
```

```
/*  windows  normal                    */

int windows1(int x1,int y1,int x2,int y2,int numb,int color1,
             int color2,int color3,int color4,char *text1)
{
int x3,y3,x4,y4,x5,x6,y5,y6,x7,x8,y7,y8,x9,y9,x10,y10;
char dif1;

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;

setregion(x3-1,y3-1,x4+1,y4+1);
mousegoto(MaxX/2,MaxY/2);
mouseoff();

BoxColor(x3,y3,x4,y4,color1);

x5=x3+10;
y5=y3+10;
x6=x4-10;
y6=y3+35;
BoxColor(x5,y5,x6,y6,color3);
chan_set_text(YELLOW);
outtextxy((MaxX/2),y6-15,text1);
```

```
if (numb == 2 )
{
x7=x3+10;
y7=y4-35;
x8=x3+62;
y8=y4-10;
BoxColor(x7,y7,x8,y8,color4);
chan_set_text(YELLOW);

outtextxy((x7+x8)/2,(y7+y8)/2,"Cancel");
setcolor(BLUE);
outtextxy((x7+x8)/2,(y7+y8)/2,"C      ");



x9=x4-62;
y9=y4-35;
x10=x4-10;
y10=y4-10;
BoxColor(x9,y9,x10,y10,color4);
chan_set_text(YELLOW);

outtextxy((x9+x10)/2,(y9+y10)/2," K");
setcolor(BLUE);
outtextxy((x9+x10)/2,(y9+y10)/2,"O ");

.}
```

```
else
 {
        x7=(MaxX/2)-20;
        y7=y4-35;
        x8=(MaxX/2)+20;
        y8=y4-10;
        BoxColor(x7,y7,x8,y8,color4);
       chan_set_text(YELLOW);

        outtextxy((x7+x8)/2,(y7+y8)/2,"Ok");
        setcolor(BLUE);
        outtextxy((x7+x8)/2,(y7+y8)/2,"O ");
 }

klick(x5/*x3+10*/,y5/*y3+10*/,x6/*x4-10*/,y6/*y3+35*/,BLACK,WHITE);

getcolor();
settextstyle(0,0,0);
settextjustify(0,2);
mouseon();
mousegoto(MaxX/2,MaxY/2);

 }
```

```
    /*  windows 2        Rx      */

int windows2_r(int x1,int y1,int x2,int y2,int color1,
            int color2,int color3,int color4,char *text1)
{
int x3,y3,x4,y4,e1;
int x11[20],y11[20];

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;

setregion(x3-1,y3-1,x4+1,y4+1);
mousegoto(MaxX/2,MaxY/2);
mouseoff();
BoxColor(x3,y3,x4,y4,color1);

x11[2]=x3+5;
y11[2]=y3+5;
x11[3]=x4-5;
y11[3]=y3+27;
BoxColor(x11[2],y11[2],x11[3],y11[3],color2);
chan_set_text(YELLOW);
outtextxy((MaxX/2),(y11[2]+y11[3])/2-2,text1);

/*x11[4]=x3+10;y11[4]=y4-35;
  x11[5]=x3+62;  y11[5]=y4-10;
```

```
BoxColor(x11[4],y11[4],x11[5],y11[5],color3);
chan_set_text(YELLOW);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"Cancel");
setcolor(BLUE);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"C      ");*/

x11[6]=x4-62;
y11[6]=y3+196;
x11[7]=x4-10;
y11[7]=y3+216;
BoxColor(x11[6],y11[6],x11[7],y11[7],color4);
chan_set_text(GREEN);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2," K");
setcolor(BLUE);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2,"O ");

x11[8]=x3+10+20;
y11[8]=y3+50;
x11[9]=x3+62+20;
y11[9]=y3+70;
BoxColor(x11[8],y11[8],x11[9],y11[9],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[8]+y11[9])/2,ch[0]);
outtextxy((x11[8]+x11[9])/2,(y11[8]+y11[9])/2,ch[5]);
setcolor(RED);
outtextxy((x11[8]+x11[9])/2,(y11[8]+y11[9])/2,ch1[5]);

x11[10]=x3+10+20;
y11[10]=y3+85;
```

```
x11[11]=x3+62+20;
y11[11]=y3+105;
BoxColor(x11[10],y11[10],x11[11],y11[11],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[10]+y11[11])/2,ch[1]);
outtextxy((x11[10]+x11[11])/2,(y11[10]+y11[11])/2,ch[6]);
setcolor(RED);
outtextxy((x11[10]+x11[11])/2,(y11[10]+y11[11])/2,ch1[0,6]);

x11[12]=x3+10+20;
y11[12]=y3+120;
x11[13]=x3+62+20;
y11[13]=y3+140;
BoxColor(x11[12],y11[12],x11[13],y11[13],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[12]+y11[13])/2,ch[0,2]);
outtextxy((x11[12]+x11[13])/2,(y11[12]+y11[13])/2,ch[0,7]);
setcolor(RED);
outtextxy((x11[12]+x11[13])/2,(y11[12]+y11[13])/2,ch1[0,7]);

x11[14]=x3+10+20;
y11[14]=y3+155;
x11[15]=x3+62+20;
y11[15]=y3+175;
BoxColor(x11[14],y11[14],x11[15],y11[15],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[14]+y11[15])/2,ch[0,3]);
outtextxy((x11[14]+x11[15])/2,(y11[14]+y11[15])/2,ch[0,8]);
setcolor(RED);
```

```
outtextxy((x11[14]+x11[15])/2,(y11[14]+y11[15])/2,ch1[0,8]);

setcolor(WHITE);
line(x3+5,y11[15]+11,x4-5,y11[15]+11);
setcolor(BLACK);
line(x3+5,y11[15]+10,x4-5,y11[15]+10);

klick(x11[2],y11[2],x11[3],y11[3],BLACK,WHITE);

getcolor();
settextstyle(0,0,0);
settextjustify(0,2);
mouseon();
mousegoto(MaxX/2,MaxY/2);
}


  /*  windows 2_1        Tx      */

int windows2_t(int x1,int y1,int x2,int y2,int color1,
            int color2,int color3,int color4,char *text1)
{
int x3,y3,x4,y4,e1;
int x11[20],y11[20];

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;
```

```
setregion(x3-1,y3-1,x4+1,y4+1);
mousegoto(MaxX/2,MaxY/2);
mouseoff();
BoxColor(x3,y3,x4,y4,color1);

x11[2]=x3+5;
y11[2]=y3+5;
x11[3]=x4-5;
y11[3]=y3+27;
BoxColor(x11[2],y11[2],x11[3],y11[3],color2);
chan_set_text(YELLOW);
outtextxy((MaxX/2),(y11[2]+y11[3])/2-2,text1);

/*x11[4]=x3+10;y11[4]=y4-35;
x11[5]=x3+62;  y11[5]=y4-10;
BoxColor(x11[4],y11[4],x11[5],y11[5],color3);
chan_set_text(YELLOW);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"Cancel");
setcolor(BLUE);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"C      ");*/

x11[6]=x4-62;
y11[6]=y3+196;
x11[7]=x4-10;
y11[7]=y3+216;
BoxColor(x11[6],y11[6],x11[7],y11[7],color4);
chan_set_text(GREEN);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2," K");
```

```
setcolor(BLUE);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2,"O ");

x11[8]=x3+10+20;
y11[8]=y3+50;
x11[9]=x3+62+20;
y11[9]=y3+70;
BoxColor(x11[8],y11[8],x11[9],y11[9],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[8]+y11[9])/2,ch2[0]);
outtextxy((x11[8]+x11[9])/2,(y11[8]+y11[9])/2,ch2[5]);
setcolor(RED);
outtextxy((x11[8]+x11[9])/2,(y11[8]+y11[9])/2,ch1[5]);

x11[10]=x3+10+20;
y11[10]=y3+85;
x11[11]=x3+62+20;
y11[11]=y3+105;
BoxColor(x11[10],y11[10],x11[11],y11[11],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[10]+y11[11])/2,ch2[1]);
outtextxy((x11[10]+x11[11])/2,(y11[10]+y11[11])/2,ch2[6]);
setcolor(RED);
outtextxy((x11[10]+x11[11])/2,(y11[10]+y11[11])/2,ch1[0,6]);

x11[12]=x3+10+20;
y11[12]=y3+120;
x11[13]=x3+62+20;
y11[13]=y3+140;
```

```
BoxColor(x11[12],y11[12],x11[13],y11[13],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[12]+y11[13])/2,ch2[0,2]);
outtextxy((x11[12]+x11[13])/2,(y11[12]+y11[13])/2,ch2[0,7]);
setcolor(RED);
outtextxy((x11[12]+x11[13])/2,(y11[12]+y11[13])/2,ch1[0,7]);

x11[14]=x3+10+20;
y11[14]=y3+155;
x11[15]=x3+62+20;
y11[15]=y3+175;
BoxColor(x11[14],y11[14],x11[15],y11[15],CYAN);
setcolor(BLACK);
outtextxy(MaxX/2,(y11[14]+y11[15])/2,ch2[0,3]);
outtextxy((x11[14]+x11[15])/2,(y11[14]+y11[15])/2,ch2[0,8]);
setcolor(RED);
outtextxy((x11[14]+x11[15])/2,(y11[14]+y11[15])/2,ch1[0,8]);

setcolor(WHITE);
line(x3+5,y11[15]+11,x4-5,y11[15]+11);
setcolor(BLACK);
line(x3+5,y11[15]+10,x4-5,y11[15]+10);

klick(x11[2],y11[2],x11[3],y11[3],BLACK,WHITE);
getcolor();
settextstyle(0,0,0);
settextjustify(0,2);
mouseon();
mousegoto(MaxX/2,MaxY/2);
```

```
}


/*  windows main menu            */
/*                               */

int windows3(int x1,int y1,int x2,int y2,int color1,
             int color2,int color3,int color4,char *text1)
{
int x3,y3,x4,y4;
int x11[25],y11[25];

x3=(MaxX/2)-x1;
y3=(MaxY/2)-y1;
x4=(MaxX/2)+x2;
y4=(MaxY/2)+y2;
setregion(x3-1,y3-1,x4+1,y4+1);
mousegoto(MaxX/2,MaxY/2);
mouseoff();
BoxColor(x3,y3,x4,y4,color1);

x11[2]=x3+5;
y11[2]=y3+5;
x11[3]=x4-5;
y11[3]=y3+27;
BoxColor(x11[2],y11[2],x11[3],y11[3],color2);
chan_set_text(YELLOW);
```

```
outtextxy((MaxX/2),(y11[2]+y11[3])/2-2,text1);

x11[4]=x3+10;
y11[4]=y4-35;
x11[5]=x3+62;
y11[5]=y4-10;
BoxColor(x11[4],y11[4],x11[5],y11[5],color3);

setcolor(BLACK);
line(x11[2]+3,y11[4]-10,x11[3]-3,y11[4]-10);
setcolor(WHITE);
line(x11[2]+3,y11[4]-9,x11[3]-3,y11[4]-9);

setcolor(BLACK);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"Exit");
setcolor(BLUE);
outtextxy((x11[4]+x11[5])/2,(y11[4]+y11[5])/2,"E    ");

x11[6]=x4-62;
y11[6]=y4-35;
x11[7]=x4-10;
y11[7]=y4-10;
BoxColor(x11[6],y11[6],x11[7],y11[7],color4);

setcolor(BLACK);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2,"Read");
setcolor(BLUE);
outtextxy((x11[6]+x11[7])/2,(y11[6]+y11[7])/2,"    d");
```

```
x11[16]=x11[6]-72;
y11[6]=y4-35;
x11[17]=x11[6]-20;
y11[7]=y4-10;
BoxColor(x11[16],y11[6],x11[17],y11[7],color4);

setcolor(BLACK);
outtextxy((x11[16]+x11[17])/2,(y11[6]+y11[7])/2,"HELP");
setcolor(BLUE);
outtextxy((x11[16]+x11[17])/2,(y11[6]+y11[7])/2,"H    ");

x11[18]=x11[16]-72;
y11[6]=y4-35;
x11[19]=x11[16]-20;
y11[7]=y4-10;
BoxColor(x11[18],y11[6],x11[19],y11[7],color4);

setcolor(BLACK);
outtextxy((x11[18]+x11[19])/2,(y11[6]+y11[7])/2,"RX");
setcolor(BLUE);
outtextxy((x11[18]+x11[19])/2,(y11[6]+y11[7])/2,"R  ");

x11[20]=x11[18]-72;
y11[6]=y4-35;
x11[21]=x11[18]-20;
y11[7]=y4-10;
BoxColor(x11[20],y11[6],x11[21],y11[7],color4);
```

```
setcolor(BLACK);
outtextxy((x11[20]+x11[21])/2,(y11[6]+y11[7])/2,"TX");
setcolor(BLUE);
outtextxy((x11[20]+x11[21])/2,(y11[6]+y11[7])/2,"T ");

x11[22]=x11[20]-72;
y11[6]=y4-35;
x11[23]=x11[20]-20;
y11[7]=y4-10;
BoxColor(x11[22],y11[6],x11[23],y11[7],color4);
setcolor(BLACK);
outtextxy((x11[22]+x11[23])/2,(y11[6]+y11[7])/2,"Clear");
setcolor(BLUE);
outtextxy((x11[22]+x11[23])/2,(y11[6]+y11[7])/2," l   ");

BoxColor(x11[22]-72,y11[6],x11[22]-20,y11[7],color4);
setcolor(BLACK);
outtextxy(((x11[22]-72)+(x11[22])-20)/2,(y11[6]+y11[7])/2,"About");
setcolor(BLUE);
outtextxy(((x11[22]-72)+(x11[22])-20)/2,(y11[6]+y11[7])/2,"A    ");

klick(x11[2],y11[2],x11[3],y11[3],BLACK,WHITE);

getcolor();
settextstyle(0,0,0);
settextjustify(0,2);
mouseon();
mousegoto(MaxX/2,MaxY/2);
```

```
}

/* Esc  & OK  */
/*          */

void Esc_ok(int x1,int y1,int x2,int y2,int numb)
{
int xx1,yy1,xx2,yy2,xx3,xx4;
char c;

xx1=/*x3*/((MaxX/2)-x1)+10;
yy1=/*y4*/((MaxY/2)+y2)-35;
xx2=/*x3*/((MaxX/2)-x1)+62;
yy2=/*y4*/((MaxY/2)+y2)-10;
xx3= ((MaxX/2)+x2)-62;
xx4= ((MaxX/2)+x2)-10;

mouseon();

if (numb == 2)
{
while(ans != 2)
{
m = getmouse1();

if((a[1]/*yy*/ > yy1 && a[1]/*yy*/ < yy2) && m == 1)
  {
  if((a[0]/*xx*/ > xx1 && a[0]/*xx*/ < xx2) && m == 1) /*cannel*/
    {   goto ok_c1;
```

```
    }
   else  if((a[0]/*xx*/ > xx3 && a[0]/*xx*/ < xx4) && m==1) /*ok*/
    {    goto ok_ok;
    }
  }
 else if(kbhit())
  {
  c = getch();
  if(c == 'C' || c == 'c'|| c == ESC)
    {
    ok_c1:
    klick(xx1,yy1,xx2,yy2,BLACK,WHITE);
    delay(550);
    ans = 2;  c = 'c';
    }
   else if(c == 'o'|| c == 'O'|| c == ENTER)
    {   ok_ok:
        ans=1;
        klick(xx3,yy1,xx4,yy2,BLACK,WHITE);
        delay(550);
        closegraph();
        clrscr();
        exit(1);
      }
    }
 }


 }
 else
```

```
{
xx1 = (MaxX/2)-20;
xx2 = (MaxX/2)+20;
while(ans != 2)
{
m = getmouse1();
if((a[1]/*yy*/ > yy1 && a[1]/*yy*/ < yy2) && m == 1)
  {
  if((a[0]/*xx*/ > xx1 && a[0]/*xx*/ < xx2) && m == 1)   /*  ok1   */
    {  goto ok_ok1;
    }
  }
else if(kbhit())
  {
  c = getch();
  if(c == 'o' || c == 'O'|| c == ENTER)
    {      ok_ok1:
    /*    ans = 1;*/
    klick(xx1,yy1,xx2,yy2,BLACK,WHITE);
    delay(550);
    /*  exit(1);  */
    ans = 2;       c = 'o';
    }
  }
}
}

ans = 0;mouseoff();
}
```

```
/*        choice              rx      */

void Choice_r(int x1,int y1,int x2,int y2)
{
 int x3,x4,y3,y4,xx[20],yy[20],p1,po1,va[5];
 char c,c1;

 x3=(MaxX/2)-x1;
 y3=(MaxY/2)-y1;
 x4=(MaxX/2)+x2;
 y4=(MaxY/2)+y2;

 xx[4]= x3+10+20;
 xx[5]= x3+62+20;
 xx[6]= x4-62;
 xx[7]= x4-10;

 yy[4]= /*y4-35;*/y3+196;
 yy[5]= /*y4-10;*/y3+216;
 yy[8]= y3+50;
 yy[9]= y3+70;
 yy[10]=y3+85;
 yy[11]=y3+105;
 yy[12]= y3+120;
 yy[13]= y3+140;
 yy[14]= y3+155;
 yy[15]= y3+175;
```

```
p1=0; c1=0;

mouseon();

c1 = inportb(Pb1);
printf(" Pb1 %.4X ",c1);
if (c1==0xffa1)       goto w2_k1;
if (c1==0xffa2)       goto w2_k2;
if (c1==0xffa3)       goto w2_k3;
if (c1==0xffa4)       goto w2_k4;

while(ans !=2)
{
m = getmouse1(); /*mousetimes()*/

if((a[1]/*y*/ > yy[4] && a[1] < yy[5]) && m == 1)
  {                                    /*  Cannel  */
    if((a[0] > xx[6] && a[0] < xx[7]) && m==1)   /*    ok      OK  */
          goto w2_ok;
  }
else
 if((a[0]/*x*/ > xx[4] && a[0] < xx[5]) && m == 1)
  {
  if((a[1]/*y*/ > yy[8] && a[1] < yy[9]) && m == 1) /*   c1   C1 */
          goto w2_k1;
  if((a[1] > yy[10] && a[1] < yy[11]) && m==1) /*  C2    c2  */
         goto w2_k2;
  if((a[1] > yy[12] && a[1] < yy[13]) && m==1) /*  c3    C3  */
          goto w2_k3;
```

```
  if((a[1] > yy[14] && a[1] < yy[15]) && m==1) /*  c4    C4  */
          goto w2_k4;
  }
else if(kbhit())
  {
  c = getch();
  if(c == '1')
    {
    w2_k1:
    klick(xx[4],yy[8],xx[5],yy[9],BLACK,WHITE);
    delay(550);
    klick(xx[4],yy[8],xx[5],yy[9],WHITE,BLACK);

    setpi(x3+15,yy[10]+10,LIGHTGRAY);
    setpi(x3+15,yy[12]+10,LIGHTGRAY);
    setpi(x3+15,yy[14]+10,LIGHTGRAY);
    setpi(x3+15,yy[8]+10,MAGENTA);

    p1= p0 = 1;
    c = '1';
    va[0] = 0x5d;
    va[1] = 0xa1;
    }
   if(c == '2')
    {
    w2_k2:
    klick(xx[4],yy[10],xx[5],yy[11],BLACK,WHITE);
    delay(550);
```

```
klick(xx[4],yy[10],xx[5],yy[11],WHITE,BLACK);

setpi(x3+15,yy[12]+10,LIGHTGRAY);
setpi(x3+15,yy[14]+10,LIGHTGRAY);
setpi(x3+15,yy[8]+10,LIGHTGRAY);
setpi(x3+15,yy[10]+10,BROWN);

p1=p0=2;
c = '2';
va[0] = 0x11;
va[1] = 0xa2;
}
if(c == '3')
{
w2_k3:
klick(xx[4],yy[12],xx[5],yy[13],BLACK,WHITE);
delay(550);
klick(xx[4],yy[12],xx[5],yy[13],WHITE,BLACK);

setpi(x3+15,yy[10]+10,LIGHTGRAY);
setpi(x3+15,yy[14]+10,LIGHTGRAY);
setpi(x3+15,yy[8]+10,LIGHTGRAY);
setpi(x3+15,yy[12]+10,WHITE);

p1=p0=3;
c = '3';
va[0] = 0x22;
va[1] = 0xa3;
}
```

```
if(c == '4')
  {
  w2_k4:
  klick(xx[4],yy[14],xx[5],yy[15],BLACK,WHITE);
  delay(550);
  klick(xx[4],yy[14],xx[5],yy[15],WHITE,BLACK);

  setpi(x3+15,yy[10]+10,LIGHTGRAY);
  setpi(x3+15,yy[12]+10,LIGHTGRAY);
  setpi(x3+15,yy[8]+10,LIGHTGRAY);
  setpi(x3+15,yy[14]+10,LIGHTBLUE);

/*  BoxColor((MaxX/2)-25,yy[4],(MaxX/2)+25,yy[5],LIGHTGREEN);
  chan_set_text(BLACK);
  outtextxy(MaxX/2,(yy[4]+yy[5])/2,ch[0,8]);*/
  p1=p0=4;
  c = '4';
  va[0] = 0xaf;
  va[1] = 0xa4;
  }
 if(c == 'o'|| c == 'O'|| c == ENTER)
  {
  w2_ok:
  /*  ans=1;*/
      klick(xx[6],yy[4],xx[7],yy[5],BLACK,WHITE);
      delay(550);
  if(p1==1||p1==2||p1==3||p1==4)
```

```
        port1(va[0],va[1]);
        ans = 2;
        }
      else
        {
         mal2(150,70,150,70);
         windows1(150,70,150,70,1,LIGHTGRAY,YELLOW,CYAN,CYAN,"Error");
         mouseoff();
         chan_set_text(YELLOW);
         outtextxy(MaxX/2,MaxY/2,"You not chioce !");
         mouseon();
         Esc_ok(150,70,150,70,1);
         put1();
        setregion(x3-1,y3-1,x4+1,y4+1);
         mousegoto(MaxX/2,MaxY/2);
         mouseon();
         ans = 0;
        }
      klick(xx[6],yy[4],xx[7],yy[5],WHITE,BLACK);
      c = 'o';
       }
    }
}

ans = 0;mouseoff();delay(50);
}
```

```
/*                choice_t    */

void Choice_t(int x1,int y1,int x2,int y2)
{
 int x3,x4,y3,y4,xx[20],yy[20],p1,po1,va[5];
 char c,c2;

 x3=(MaxX/2)-x1;
 y3=(MaxY/2)-y1;
 x4=(MaxX/2)+x2;
 y4=(MaxY/2)+y2;

 xx[4]= x3+10+20;
 xx[5]= x3+62+20;
 xx[6]= x4-62;
 xx[7]= x4-10;

 yy[4]= /*y4-35;*/y3+196;
 yy[5]= /*y4-10;*/y3+216;
 yy[8]= y3+50;
 yy[9]= y3+70;
 yy[10]=y3+85;
 yy[11]=y3+105;
 yy[12]= y3+120;
 yy[13]= y3+140;
 yy[14]= y3+155;
 yy[15]= y3+175;
```

```
p1=0; c2=0;

mouseon();

c2 = inportb(Pb2);
printf(" Pb2 %X ",c2);
if (c2==0xfff1)      goto w2_tk1;
if (c2==0xfff2)      goto w2_tk2;
if (c2==0xfff3)      goto w2_tk3;
if (c2==0xfff4)      goto w2_tk4;
printf(" Pb2 %.4X ",c2);
while(ans !=2)
{
m = getmouse1(); /*mousetimes()*/

if((a[1]/*y*/ > yy[4] && a[1] < yy[5]) && m == 1)
  {                                    /*  Cannel  */
    if((a[0] > xx[6] && a[0] < xx[7]) && m==1)   /*    ok      OK  */
       goto w2_tok;
  }
else
 if((a[0]/*x*/ > xx[4] && a[0] < xx[5]) && m == 1)
  {
  if((a[1]/*y*/ > yy[8] && a[1] < yy[9]) && m == 1) /*   c1   C1 */
           goto w2_tk1;
  if((a[1] > yy[10] && a[1] < yy[11]) && m==1) /*  C2    c2  */
          goto w2_tk2;
  if((a[1] > yy[12] && a[1] < yy[13]) && m==1) /*  c3    C3  */
          goto w2_tk3;
```

```
  if((a[1] > yy[14] && a[1] < yy[15]) && m==1) /*  c4    C4  */
          goto w2_tk4;
   }
else if(kbhit())
  {
  c = getch();
  if(c == '1')
    {
    w2_tk1:
    klick(xx[4],yy[8],xx[5],yy[9],BLACK,WHITE);
    delay(550);
    klick(xx[4],yy[8],xx[5],yy[9],WHITE,BLACK);

    setpi(x3+15,yy[10]+10,LIGHTGRAY);
    setpi(x3+15,yy[12]+10,LIGHTGRAY);
    setpi(x3+15,yy[14]+10,LIGHTGRAY);
    setpi(x3+15,yy[8]+10,MAGENTA);

/*    BoxColor((MaxX/2)-25,yy[4],(MaxX/2)+25,yy[5],MAGENTA);
    chan_set_text(BLACK);
    outtextxy(MaxX/2,(yy[4]+yy[5])/2,ch[0,5]);*/
    p1 = p2 = 1;
    c = '1';
    va[0] = 0x5d;
    va[1] = 0xf1;
    }
  if(c == '2')
    {
```

```
w2_tk2:
klick(xx[4],yy[10],xx[5],yy[11],BLACK,WHITE);
delay(550);
klick(xx[4],yy[10],xx[5],yy[11],WHITE,BLACK);

setpi(x3+15,yy[12]+10,LIGHTGRAY);
setpi(x3+15,yy[14]+10,LIGHTGRAY);
setpi(x3+15,yy[8]+10,LIGHTGRAY);
setpi(x3+15,yy[10]+10,BROWN);
p1 = p2=2;
c = '2';
va[0] = 0x11;
va[1] = 0xf2;
}
if(c == '3')
{
w2_tk3:
klick(xx[4],yy[12],xx[5],yy[13],BLACK,WHITE);
delay(550);
klick(xx[4],yy[12],xx[5],yy[13],WHITE,BLACK);

setpi(x3+15,yy[10]+10,LIGHTGRAY);
setpi(x3+15,yy[14]+10,LIGHTGRAY);
setpi(x3+15,yy[8]+10,LIGHTGRAY);
setpi(x3+15,yy[12]+10,WHITE);
p1= p2=3;
c = '3';
va[0] = 0x22;
va[1] = 0xf3;
```

```
  }
if(c == '4')
  {
  w2_tk4:
  klick(xx[4],yy[14],xx[5],yy[15],BLACK,WHITE);
  delay(550);
  klick(xx[4],yy[14],xx[5],yy[15],WHITE,BLACK);

  setpi(x3+15,yy[10]+10,LIGHTGRAY);
  setpi(x3+15,yy[12]+10,LIGHTGRAY);
  setpi(x3+15,yy[8]+10,LIGHTGRAY);
  setpi(x3+15,yy[14]+10,LIGHTBLUE);
  p1 = p2 = 4;
  c = '4';
  va[0] = 0xaf;
  va[1] = 0xf4;
  }
if(c == 'o'|| c == 'O'|| c == ENTER)
  {
  w2_tok:
  /*  ans=1;*/
      klick(xx[6],yy[4],xx[7],yy[5],BLACK,WHITE);
      delay(550);
      printf(" p1 %d ",p1);
  if(p1==1||p1==2||p1==3||p1==4)
    {
    port2(va[0],va[1]);
    ans = 2;
    printf(" p2 = %d ",p2);
```

```
        }
      else
        {
          mal2(150,70,150,70);
          windows1(150,70,150,70,1,LIGHTGRAY,YELLOW,CYAN,CYAN,"Error");
          chan_set_text(YELLOW);
          outtextxy(MaxX/2,MaxY/2,"You not chioce !");
          mouseon();
          Esc_ok(150,70,150,70,1);
          put1();
         setregion(x3-1,y3-1,x4+1,y4+1);
          mousegoto(MaxX/2,MaxY/2);
          mouseon();
          ans = 0;
        }
      klick(xx[6],yy[4],xx[7],yy[5],WHITE,BLACK);
      c = 'o';
       }
    }
}
ans = 0;mouseoff();delay(50);
}
```

```
/* Choicemain      */
/*                 */

void Choicemain(int x1,int y1,int x2,int y2)
{
 int x3,x4,y3,y4,xx[20],yy[20],i,po1,po2;
 char c,c1,c2;

 x3=(MaxX/2)-x1; y3=(MaxY/2)-y1;
 x4=(MaxX/2)+x2; y4=(MaxY/2)+y2;

 xx[4]= x3+10;
 xx[5]= x3+62;
 xx[6]= x4-62;
 xx[7]= x4-10;
 xx[16]=xx[6]-72;
 xx[17]=xx[6]-20;
 xx[18]=xx[16]-72;
 xx[19]=xx[16]-20;
 xx[20]=xx[18]-72;
 xx[21]=xx[18]-20;
 xx[22]=xx[20]-72;
 xx[23]=xx[20]-20;

 yy[4]= y4-35;
 yy[5]= y4-10;
 yy[8]= y3+50;
 yy[9]= y3+70;
```

```
yy[10]=y3+85;
yy[11]=y3+107;
yy[12]= y3+120;
yy[13]= y3+145;
yy[14]= y3+155;
yy[15]= y3+180;
i = 0;
p2=16;    p0=17;

mouseon();
while(ans !=2)
{
m = getmouse1(); /*mousetimes()*/

if((a[1]/*y*/ > yy[4] && a[1] < yy[5]) && m == 1)
  {
  if((a[0]/*x*/ > xx[4] && a[0] < xx[5]) && m == 1)   /* Esc  */
    goto esc_1;

  if((a[0] > xx[6] && a[0] < xx[7]) && m==1)          /*  ok  */
    goto ok_1;

  if((a[0]/*x*/ > xx[16] && a[0] < xx[17]) && m == 1) /* help  */
   goto help_1;

  if((a[0]/*x*/ > xx[18] && a[0] < xx[19]) && m == 1) /*  'rx' */
    goto t1;

  if((a[0]/*x*/ > xx[20] && a[0] < xx[21]) && m == 1) /*   'tx'  */
```

```
    goto t2;

  if((a[0]/*x*/ > xx[22] && a[0] < xx[23]) && m == 1) /* clear */
    goto clear_1;

  if((a[0]/*x*/ > xx[22]-72 && a[0] < xx[22]-20) && m == 1) /* about */
    goto abou1;
  }

else if(kbhit())
  {
  c = getch();
  if (c == 'l' || c == 'L')   /*    CLEAR     clear     */
  {
    clear_1:
    p0= 6;       p2=7;
    i = 0;
    klick(xx[22],yy[4],xx[23],yy[5],BLACK,WHITE);
    delay(550);
    klick(xx[22],yy[4],xx[23],yy[5],WHITE,BLACK);
    port1(0x00,0x00);
    port2(0x00,0x00);

    mouseon_on(x3,y3,x4,y4);
    setfillstyle(1,BLUE);
    bar(xx[4],yy[8]-20,xx[7],yy[13]+50);
    ans = 0;
    c = 'l';
  }
```

```
if (c == 'a' || c == 'A')   /*     about      About   */
{
  abou1:
  p0= 6;       p2=7;
  i = 0;

  klick(xx[22]-72,yy[4],xx[22]-20,yy[5],BLACK,WHITE);
  delay(550);
  klick(xx[22]-72,yy[4],xx[22]-20,yy[5],WHITE,BLACK);

  mal(150,150,150,150);
  windows1(150,150,150,150,1,LIGHTGRAY,WHITE,CYAN,CYAN,
  "PROGRAM VERSION 1.0");
  mouseoff();
  about1();
  mouseon();
  Esc_ok(150,150,150,150,1);
  put();
  mouseon_on(x3,y3,x4,y4);
  ans = 0;
  c = 'a';
  }
 if(c == 'E' || c == 'e'|| c == ESC)          /*   EXIT      exit     */
  {
  esc_1:
  klick(xx[4],yy[4],xx[5],yy[5],BLACK,WHITE);
  delay(550);
  klick(xx[4],yy[4],xx[5],yy[5],WHITE,BLACK);
```

```
mal(150,90,150,90);
windows1(150,90,150,90,2,LIGHTGRAY,WHITE,RED,CYAN,"Exit");
mouseoff();
chan_set_text(RED);
outtextxy(MaxX/2,MaxY/2,"Are you sure you want to exit ?");
outtextxy(MaxX/2,MaxY/2+20,"plese botton");
mouseon();
Esc_ok(150,90,150,90,2);
put();
mouseon_on(x3,y3,x4,y4);
i = 0;
ans = 0;
c = 'e';
}
if (c == 'h' || c == 'H')    /* HELP      help    */
{
help_1:
klick(xx[16],yy[4],xx[17],yy[5],BLACK,WHITE);
delay(550);
klick(xx[16],yy[4],xx[17],yy[5],WHITE,BLACK);
mal(100,190,100,190);
windows1(100,190,100,190,1,LIGHTGRAY,WHITE,LIGHTMAGENTA,CYAN,
"HELP");
mouseoff();
help1();
mouseon();
Esc_ok(100,190,100,190,1);
put();
mouseon_on(x3,y3,x4,y4);
```

```
/*BoxColor((MaxX/2)-25,yy[4],(MaxX/2)+25,yy[5],BLUE);
  klick((MaxX/2)-25,yy[8],(MaxX/2)+25,yy[9],LIGHTGRAY,LIGHTGRAY);*/
  i = 0;
  ans = 0;
  c = 'h';
  }
if(c == 'r' || c == 'R')    /*   RX        rx   */
 {
  t1:
  klick(xx[18],yy[4],xx[19],yy[5],BLACK,WHITE);
  delay(550);
  klick(xx[18],yy[4],xx[19],yy[5],WHITE,BLACK);

  mal(200,120,200,105);
  windows2_r(200,120,200,105,LIGHTGRAY,GREEN,CYAN,RED," Rx MENU ");
  Choice_r(200,120,200,105);
  put();
    if(p0==1)
     {
  setfillstyle(1,MAGENTA);
  bar(xx[4],yy[8],xx[7],yy[9]);
  chan_set_text(BLACK);
  outtextxy(MaxX/2-100,(yy[8]+yy[9])/2," Rx Sent ");
  outtextxy(MaxX/2,(yy[8]+yy[9])/2,ch[0]);
     }
    if(p0==2)
     {
  setfillstyle(1,BROWN);
```

```
bar(xx[4],yy[8],xx[7],yy[9]);
chan_set_text(BLACK);
outtextxy(MaxX/2-100,(yy[8]+yy[9])/2," Rx Sent ");
outtextxy(MaxX/2,(yy[8]+yy[9])/2,ch[1]);
   }
  if(p0==3)
   {
setfillstyle(1,WHITE);
bar(xx[4],yy[8],xx[7],yy[9]);
chan_set_text(BLACK);
outtextxy(MaxX/2-100,(yy[8]+yy[9])/2," Rx Sent ");
outtextxy(MaxX/2,(yy[8]+yy[9])/2,ch[2]);
   }
  if(p0==4)
   {
setfillstyle(1,LIGHTBLUE);
bar(xx[4],yy[8],xx[7],yy[9]);
chan_set_text(BLACK);
outtextxy(MaxX/2-100,(yy[8]+yy[9])/2," Rx Sent ");
outtextxy(MaxX/2,(yy[8]+yy[9])/2,ch[3]);
   }
mouseon_on(x3,y3,x4,y4);

i = 0;
ans =0;
c= 'r';
}
```

```
if (c == 'T' || c == 't')       /*    TX      tx     */
    {
    t2:
    klick(xx[20],yy[4],xx[21],yy[5],BLACK,WHITE);
    delay(550);
    klick(xx[20],yy[4],xx[21],yy[5],WHITE,BLACK);
    mal(200,120,200,105);
    windows2_t(200,120,200,105,LIGHTGRAY,GREEN,CYAN,YELLOW,
    " Tx MENU ");
    Choice_t(200,120,200,105);
    put();
    printf(" p2 %d ",p2);
      if(p2==1)
       {
    setfillstyle(1,MAGENTA);
    bar(xx[4],yy[10],xx[7],yy[11]);
    chan_set_text(BLACK);
    outtextxy(MaxX/2-100,(yy[10]+yy[11])/2," Tx Sent ");
    outtextxy(MaxX/2,(yy[10]+yy[11])/2,ch[5]);
       }
      if(p2==2)
       {
    setfillstyle(1,BROWN);
    bar(xx[4],yy[10],xx[7],yy[11]);
    chan_set_text(BLACK);
    outtextxy(MaxX/2-100,(yy[10]+yy[11])/2," Tx Sent ");
    outtextxy(MaxX/2,(yy[10]+yy[11])/2,ch[6]);
       }
```

```
  if(p2==3)
   {
 setfillstyle(1,WHITE);
 bar(xx[4],yy[10],xx[7],yy[11]);
 chan_set_text(BLACK);
 outtextxy(MaxX/2-100,(yy[10]+yy[11])/2," Tx  Sent ");
 outtextxy(MaxX/2,(yy[10]+yy[11])/2,ch[7]);
   }
  if(p2==4)
   {
 setfillstyle(1,LIGHTBLUE);
 bar(xx[4],yy[10],xx[7],yy[11]);
 chan_set_text(BLACK);
 outtextxy(MaxX/2-100,(yy[10]+yy[11])/2," Tx Sent ");
 outtextxy(MaxX/2,(yy[10]+yy[11])/2,ch[8]);
   }
 mouseon_on(x3,y3,x4,y4);

 i = 0;
 ans =0;
 c= 't';
 }
if(c == 'd'|| c == 'D')      /*  OK      ok  */
 { ok_1:
 /*  ans=1;*/
      klick(xx[6],yy[4],xx[7],yy[5],BLACK,WHITE);
       delay(550);
      klick(xx[6],yy[4],xx[7],yy[5],WHITE,BLACK);
```

```
c1 = inportb(Pb1);
printf(" 0x301 = %c ",c1);
printf(" 0x301 = %.4X ",inp(Pb1));
c2 = inportb(Pb2);
printf(" 0x305 = %c ",c1);
printf(" 0x305 = %.4X ",inp(Pb2));

mal2(150,80,150,80);
windows1(150,80,150,80,1,LIGHTGRAY,MAGENTA,CYAN,CYAN,
"Read Tx & Rx");
mouseoff();
chan_set_text(BROWN);
if (c1==0xffa1)  outtextxy(MaxX/2,MaxY/2-20,"RX sent  Cannal 1");
if (c1==0xffa2)  outtextxy(MaxX/2,MaxY/2-20,"RX sent  Cannal 2");
if (c1==0xffa3)  outtextxy(MaxX/2,MaxY/2-20,"Rx sent  Cannal 3");
if (c1==0xffa4)  outtextxy(MaxX/2,MaxY/2-20,"Rx sent  Cannal 4");
if (c1!=0xffa4&&&c1!=0xffa1&&&c1!=0xffa2&&&c1!=0xffa3)
                 outtextxy(MaxX/2,MaxY/2-20,"Rx sent  Clear");
if (c2==0xfff1)  outtextxy(MaxX/2,MaxY/2,"TX sent  Cannal 1");
if (c2==0xfff2)  outtextxy(MaxX/2,MaxY/2,"TX sent  Cannal 2");
if (c2==0xfff3)  outtextxy(MaxX/2,MaxY/2,"Tx sent  Cannal 3");
if (c2==0xfff4)  outtextxy(MaxX/2,MaxY/2,"Tx sent  Cannal 4");
if (c2!=0xfff4&&&c2!=0xfff1&&&c2!=0xfff2&&&c2!=0xfff3)
                 outtextxy(MaxX/2,MaxY/2,"Tx sent  Clear");
outtextxy(MaxX/2,MaxY/2+20,"plese botton");
mouseon();
Esc_ok(150,80,150,80,1);
put1();
```

```
        setregion(x3-1,y3-1,x4+1,y4+1);
        mousegoto(MaxX/2,MaxY/2);
        mouseon();
      ans = 0;    c ='o';    /*    printf(" o = %c",c);*/
     }
   }
}

ans = 0;mouseoff();
}

chan_set_text(int color1)
{
    changetextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    settextjustify(CENTER_TEXT,TOP_TEXT);
    setcolor(color1);
}

help1()
{
    chan_set_text(RED);
    outtextxy(MaxX/2,MaxY/2-150,"Main Menu");
    outtextxy(MaxX/2,MaxY/2+5,"Tx & Rx Menu");
    outtextxy(MaxX/2,MaxY/2+139,"Key <Enter> = Ok       ");
    outtextxy(MaxX/2,MaxY/2+124,"Key <Esc> = Cannel,exit");
    chan_set_text(BLACK);
    outtextxy(MaxX/2,MaxY/2-130,"Key <E,e> = Exit       ");
    outtextxy(MaxX/2,MaxY/2-110,"Key <A,a> = About Menu");
    outtextxy(MaxX/2,MaxY/2-90, "Key <L,l> = Clear Port");
```

```
    outtextxy(MaxX/2,MaxY/2-70, "Key <T,t> = Tx Menu    ");
    outtextxy(MaxX/2,MaxY/2-50, "Key <R,t> = Rx Menu    ");
    outtextxy(MaxX/2,MaxY/2-30, "Key <H,h> = Help Menu ");
    outtextxy(MaxX/2,MaxY/2-10, "Key <D,d> = Read Menu ");
    outtextxy(MaxX/2,MaxY/2+25, "Key <1>   = Channel 1 ");
    outtextxy(MaxX/2,MaxY/2+45, "Key <2>   = Channel 2 ");
    outtextxy(MaxX/2,MaxY/2+65, "Key <3>   = Channel 3 ");
    outtextxy(MaxX/2,MaxY/2+85, "Key <4>   = Channel 4 ");
    outtextxy(MaxX/2,MaxY/2+105,"Key <O,o> = Ok         ");
}

about1()
{
    chan_set_text(BLACK);
    outtextxy(MaxX/2,MaxY/2-80,"*| PROGRAMER BY |*");
    outtextxy(MaxX/2,MaxY/2-40,"THEERADECH JUN-ADUNG");
    outtextxy(MaxX/2,MaxY/2-20,"SINGHA BAMRUNGJIT");
    outtextxy(MaxX/2,MaxY/2,   "ATCHARAPORN DEEYING");
    outtextxy(MaxX/2,MaxY/2+20,"********");
    outtextxy(MaxX/2,MaxY/2+40,"ROOM N");
    outtextxy(MaxX/2,MaxY/2+80,"DEPARTMENT OF
     INDUSTRIAL TECHNOLOGY");
    outtextxy(MaxX/2,MaxY/2+100,"FACULTY OF ENGINEERING");
}
```

```
/*  Klick  */
/*         */
int klick(int x3,int y3,int x4,int y4,int color1,int color2,int thick)
{
mouseoff();
setcolor(color1);
setlinestyle(0,0,thick);
line(x3,y3,x3,y4);
line(x3,y3,x4,y3);
setcolor(color2);
setlinestyle(0,0,thick);
line(x4,y3,x4,y4);
line(x3,y4,x4,y4);
mouseon();
}

/*      BOXCOLOR : show color bar                    */
/*                                                   */
int BoxColor(int x1,int y1,int x2,int y2,int color)
{
   setfillstyle(1,color);
   bar(x1,y1,x2,y2);
   klick(x1,y1,x2,y2,WHITE,BLACK,1);
   getcolor();
}
```

```
int port1(int val1,int val2) /* out 8255 number 1 (303) mode 0*/
{
  outportb(Po1,0x82);
  outportb(Pa1,val1);
  printf(" out %.2X ",val1);
  outportb(Pa1,val2);
  printf(" out %.2X ",val2);
}

int port2(int val1,int val2) /* out 8255 number 2 (307) mode 0*/
{
  outportb(Po2,0x82);
  outportb(Pa2,val1);
  printf(" out %.2x ",val1);
  outportb(Pa2,val2);
  printf(" out %.2X ",val2);
}

/*      MOUSE_INITIALIZE : initializes mouse system and return value    */
/*      initialize                                                      */

mouse_initialize()
{
    union REGS regs;
    regs.x.ax=0;
    int86(0x33,&regs,&regs);
    return regs.x.ax;
}
```

```
/*      MOUSEON : show mouse cursor                    */
/*                                                     */
mouseon()
{    union REGS regs;
     regs.x.ax=1;
     int86(0x33,&regs,&regs);
}

mouseon_on(int x3,int y3,int x4,int y4)
{
setregion(x3-1,y3-1,x4+1,y4+1);
mousegoto(MaxX/2,MaxY/2);
mouseon();
}

/*      MOUSEOFF : not show mouse cursor               */
/*                                                     */
mouseoff()
{
     union REGS regs;
     regs.x.ax=2;
     int86(0x33,&regs,&regs);
 }
```

```
/*      MOUSEGOTO : move mouse cursor to row,column */
/*                                                  */
mousegoto(int row,int column)
{
    union REGS regs;
    regs.x.ax=4;
    regs.x.cx=row;
    regs.x.dx=column;
    int86(0x33,&regs,&regs);
}

/*      GETMOUSE : get address mouse giving address xx,yy */
/*                                                        */
int  getmouse1(void)
{
    union  REGS  r;
    int click,x,y;
    r.x.ax = 3;
    int86(0x33,&r,&r);
    click = r.x.bx;
    x = r.x.cx;
    y = r.x.dx;
    a[0] = x;
    a[1] = y;
    return(click);
}
```

```
/*      SETREGION : get address show mouse in viewer    */
/*                                                      */
void setregion(int x1,int y1,int x2,int y2)
{
    _CX=x1;
    _DX=x2;
    _AX=0x07;
    geninterrupt(0x33);
    _CX=y1;
    _DX=y2;
    _AX=0x08;
    geninterrupt(0x33);
}

/* circle chioce tx,rx */
void setpi(int x1,int y1,int color)
{
   setfillstyle(1,color);
   setcolor(color);
   pieslice(x1,y1,0,360,6);
}
```

```
/*  MAIN  */

int
main(void)
{
  int x,y;

  Initialize();                 /* Set system into Graphics mode  */
  mouse_initialize();
  menuy=SY;
  mousegoto(MaxX/2,MaxY/2);
  setregion(0,0,MaxX,MaxY);
  mouseon();
  BodyMenu();
  mouseoff();
  closegraph();
}
```

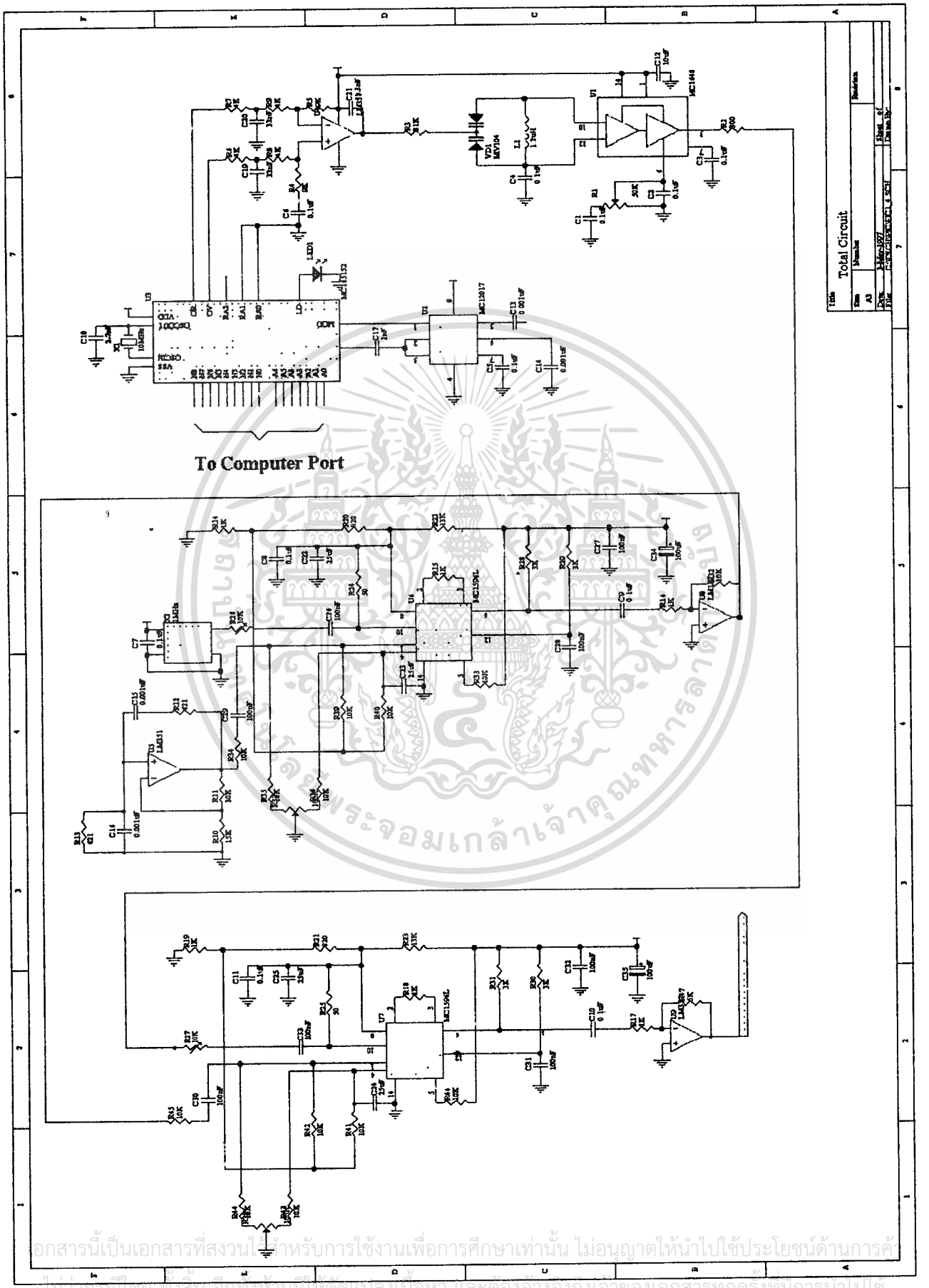
การใช้งานโปรแกรม

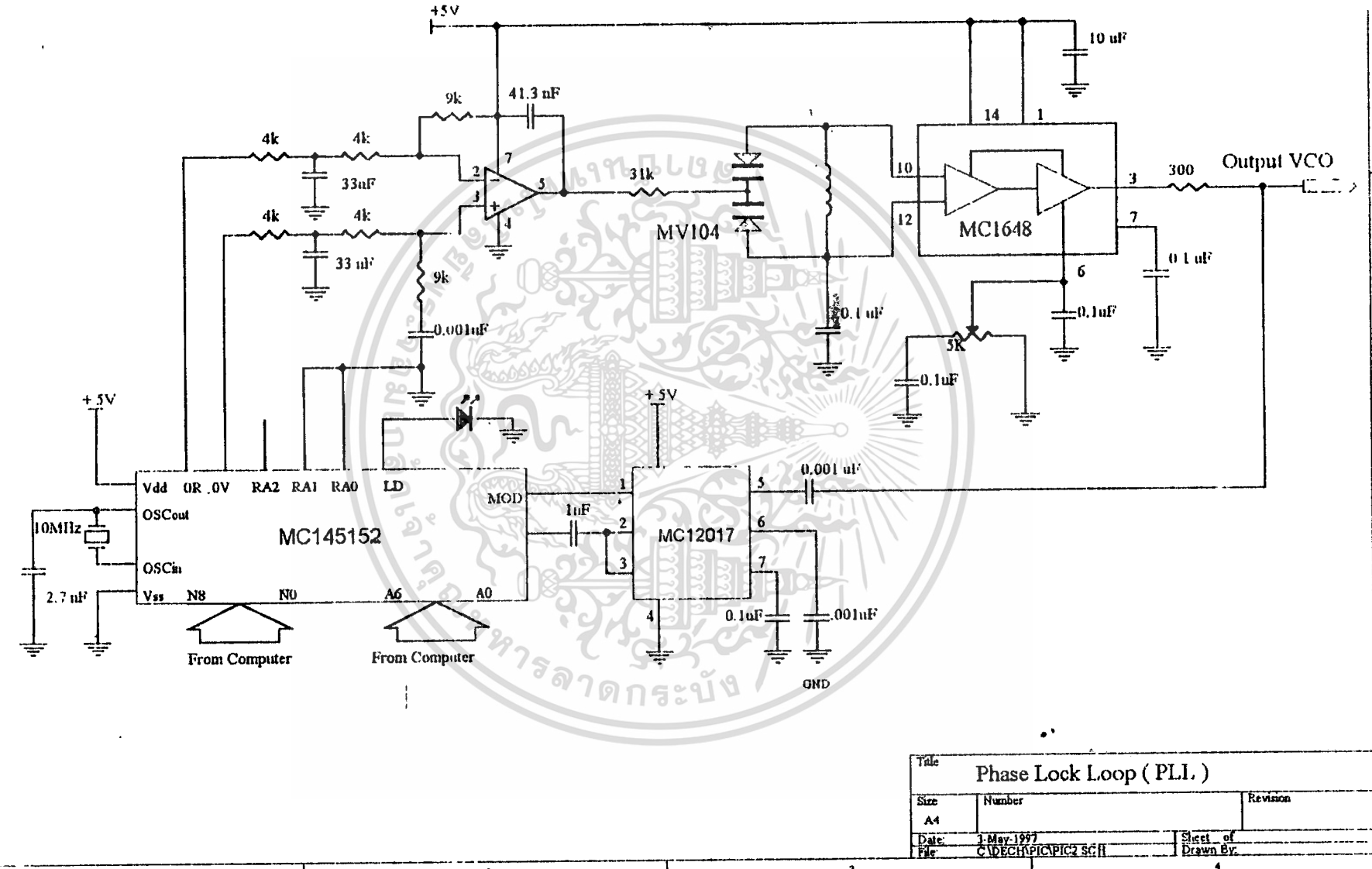
ต่ออุปกรณ์ Hardware ภายนอกให้เรียบร้อยพร้อมจะใช้งาน โดยต่อการ์ดอินเตอร์เฟส 8255 กับคอมพิวเตอร์ และต่อคอนเนคเตอร์ไปยังเครื่องสังเคราะห์ความถี่ เช็คดูจุดต่อให้เรียบร้อย แล้วเปิดเครื่องสังเคราะห์ความถี่และเครื่องคอมพิวเตอร์เพื่อ RUN โปรแกรม เพื่อเลือกช่องความถี่ซึ่งจะมี Menu ให้เลือกเป็นปุ่มคำสั่งต่างๆ โดยจะใช้งานได้ทั้ง Keyborad และ Mouse ในการเลือกคำสั่งการทำงานของโปรแกรม ในครั้งแรกที่เปิดเครื่องต้องทำการเลือกช่องความถี่ที่จะใช้งานทางด้านส่งซึ่งมีไว้ up ความถี่ที่จะใช้งานเสียก่อน โดยเลื่อนเมาส์ไปกดที่ปุ่ม Tx หรือกดปุ่ม T,t ก็ได้ จะมีเมนูให้เลือกช่องใช้งานซึ่งในที่นี้ได้ออกแบบไว้เพียง 4 ช่องเท่านั้น ซึ่งโปรแกรมจะอ่านข้อมูลจาก Hardware มาก่อนว่าใช้งานช่องไหนอยู่ ถ้าพบว่าข้อมูลที่อ่านเข้ามาตรงกับช่องใดช่องหนึ่งก็จะแสดงผลให้ดูว่าใช้งานช่องไหนอยู่ ถ้าไม่ตรงเลยก็จะต้องเลือกช่องใดช่องหนึ่งก่อนออกจากเมนูนี้ ไม่เช่นนั้นจะกลับสู่เมนูหลักไม่ได้ ซึ่งในการเลือกหรือเปลี่ยนช่องก็ทำได้โดยเลื่อนเมาส์ไปกดยังช่องที่ต้องการ หรือกดปุ่ม 1,2,3 หรือ 4 ในการเลือกช่องก็ได้ แล้วกด O,o หรือเลื่อนเมาส์ไปกดปุ่ม OK เพื่อส่งข้อมูลที่ได้เลือกไว้และกลับสู่เมนูหลักเพื่อจะทำคำสั่งอื่นต่อไป และผลจากการเลือกช่องจะแสดงผลให้ดูที่เมนูหลักด้วย โดยข้อมูลที่ได้เลือกไว้แล้วจะยังคงที่ตลอดเวลาแม้จะออกจากโปรแกรมนี้ไปใช้งานโปรแกรมอื่นแล้วก็ตามข้อมูลที่เลือกจะยังคงเดิมอยู่ถ้าโปรแกรมที่ใช้ไม่ไปยุ่งเกี่ยวกับการ์ดอินเตอร์เฟส 8255 เพราะ 8255 จะยังคง Latch ข้อมูลเดิมที่ส่งไปล่าสุดจนกระทั่งปิดเครื่องคอมพิวเตอร์ หรือส่งคอนโทรลพอร์ตข้อมูลไปใหม่หรือส่งข้อมูลไปใหม่ทับข้อมูลเดิม

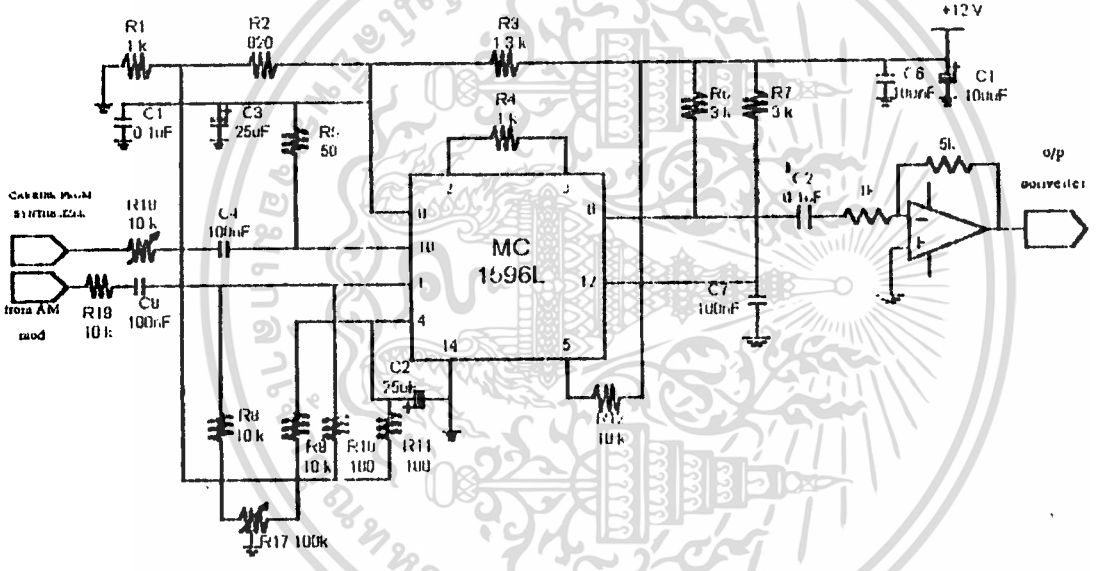
ในเมนูหลักยังจะมี Rx เมนูที่ออกแบบเผื่อไว้สำหรับภาครับสำหรับ down ความถี่กลับมาคงเดิม ซึ่ง Rx เมนูมีไว้เลื่อนช่องความถี่เช่นเดียวกลับ Tx แต่เป็นคนละความถี่กับ Tx ซึ่งในการใช้งานและหลักการทำงานก็ใช้หลักการเดียวกับ Tx เมนู ทุกอย่าง โดยจะเข้าสู่ Rx เมนูได้ก็ต้องเลื่อนเมาส์ไปยังปุ่ม Rx หรือกดปุ่ม R,r ก็ได้ ข้อมูลที่เลือกไว้จะส่งไปอีกหมายเลขพอร์ตที่ไม่ซ้ำกับหมายเลขเดิมของ Tx

Read เมนู เลือกได้โดยเลื่อนเมาส์ไปกดยังปุ่ม Read หรือกดปุ่ม R,r ก็ได้ การทำงานจะทำการอ่านข้อมูลจากพอร์ตทั้งด้านส่งและด้านรับเข้ามาแล้วเปรียบเทียบกับข้อมูลอ้างอิงเพื่อเช็คว่าขณะนี้ข้อมูลทางด้านส่งและรับถูกใช้งานที่ช่องใด ถ้าตรวจสอบแล้วตรงกับช่องใดช่องนั้นก็จะแสดงผลให้เห็นผ่านทาง Read เมนู ถ้าไม่ตรงกับช่องใดเลยจะบอกว่า data error ถ้าจะไม่ให้ error ก็เข้าไปสู่ Tx และ Rx เพื่อเลือกช่องที่ต้องการ

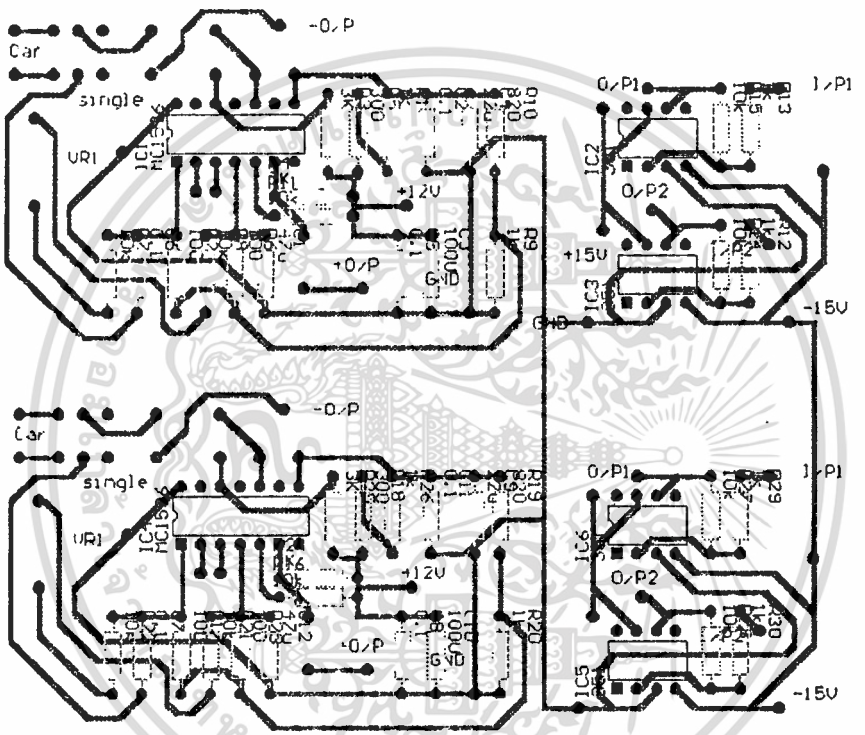
ในเมนูหลักยังมีปุ่ม Clear, About, Help และ Exit ซึ่ง การจะเลือกใช้งานก็ต้องเลื่อนเมาส์ไปกดยังปุ่มที่ต้องการ หรือกดปุ่มตามต้องการ ซึ่งปุ่มใช้งานต่างๆจะแสดงอยู่ที่ Help เมนู ส่วนปุ่ม About จะแสดงถึงรายชื่อผู้เขียนโปรแกรม ส่วนปุ่ม Clear จะ clear หน้าจอส่วนที่แสดงผลว่าขณะนี้ Tx และ Rx ทำงานที่ช่องใดบ้าง ส่วนปุ่ม Exit ไว้สำหรับออกจากโปรแกรมโดยจะมีเมนูให้ตัดสินใจให้เลือกอีกครั้งว่าพร้อมที่จะออกจากโปรแกรมหรือยัง ถ้าไม่พร้อมก็จะเลือกกลับมาเมนูหลักอีกครั้ง ถ้าจะออกก็เลื่อนเมาส์ไปที่ปุ่ม OK หรือ กด O,o ก็ได้ เป็นการสิ้นสุดการทำงานของโปรแกรม



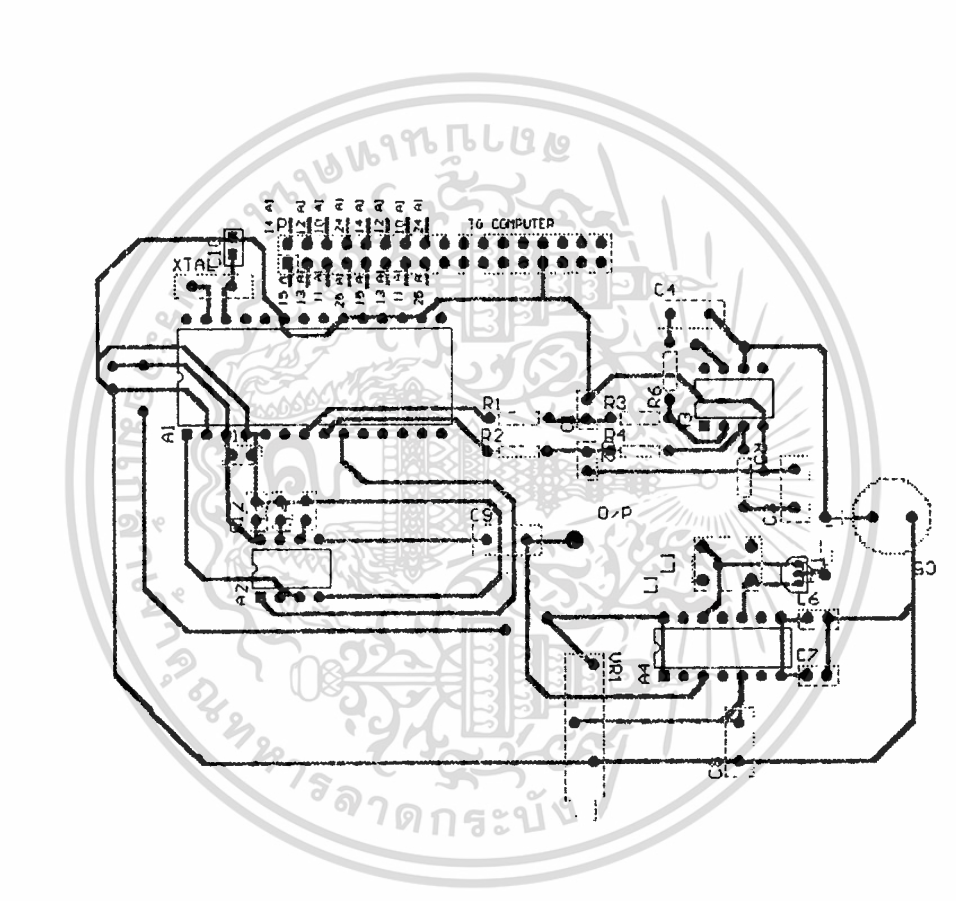

+5V
10 uF
9k
41.3 nF
4k
4k
33nF
4k
4k
33 nF
9k
0.001uF
31k
MV104
14
1
10
12
MC1648
3
300
Output VCO
7
0.1 uF
6
0.1uF
0.1 uF
5K
0.1uF
+ 5V
+ 5V
Vdd
0R
.0V
RA2
RA1
RA0
LD
OSCout
10MHz
OSCin
2.7 nF
Vss
N8
N0
A6
A0
MC145152
From Computer
From Computer
MOD
1uF
MC12017
0.001 uF
0.1uF
.001uF
GND
Title
Phase Lock Loop (PLL)
Size
A4
Number
Revision
Date:
3-May-1997
Sheet of
File:
C:\DECHKPIC\PIC2.SCH
Drawn By:



รูปวงจรแปลงความถี่ (Mixer)



ลายทองแดงวงจรแปลงความถี่ (Mixer)



ลายทองแดงวงจร Synthesizer

MC145152-2

Parallel-Input PLL Frequency Synthesizer

Interfaces with Dual-Modulus Prescalers

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

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

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

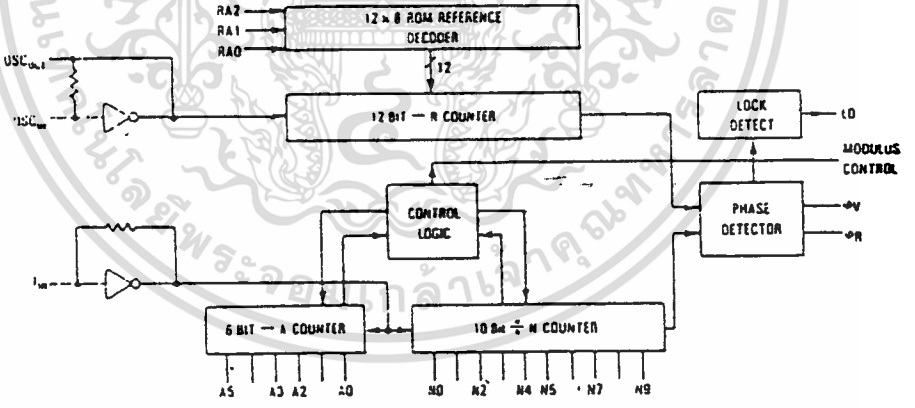
P SUFFIX
PLASTIC
CASE 710

FN SUFFIX
PLCC
CASE 776

ORDERING INFORMATION

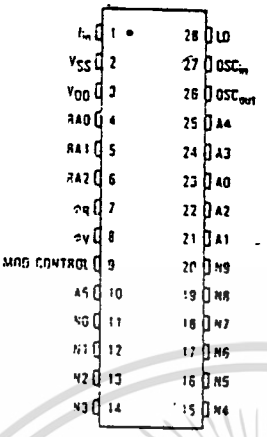
MC145152P2	Plastic DIP
MC145152FN2	PLCC Package

BLOCK DIAGRAM

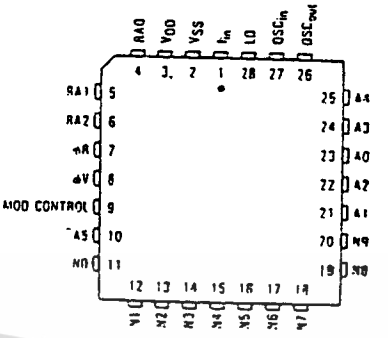


NOTE: N0 through N9, A0 through A5, and RA0 through RA2 have pullup resistors not shown.

PIN ASSIGNMENTS



PLASTIC DIP



PLCC PACKAGE

PIN DESCRIPTIONS

INPUTS

f_{in} — Frequency Input

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

RA0, RA1, RA2 — Reference Address Inputs

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

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

N Inputs — N Counter Programming Inputs

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

A Inputs — A Counter Programming Inputs

The A inputs define the number of clock cycles of f_{in} that require a logic zero on the modulus control output. (See Dual-Modulus Prescaling section.) The A inputs all have internal pullup resistors that ensure that inputs left open will remain at a logic one.

OSC_{in}, OSC_{out} — Reference Oscillator Input/Output

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

OUTPUTS

φR, φV — Phase Detector Outputs

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

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

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

If the frequency of fV = f_R and both are in phase, then both φV and φR remain high except for a small minimum time period when both pulse low in phase.

Modulus Control — Dual-Modulus Prescale Control Output

Signal generated by the on chip control logic circuitry for controlling an external dual-modulus prescaler. The modulus control level will be low at the beginning of a count cycle and will remain low until the ÷ A counter has counted down from its programmed value. At this time, modulus control goes high

and remains high until the ÷ N counter has counted the rest of the way down from its programmed value (N – A additional counts since both ÷ N and ÷ A are counting down during the first portion of the cycle). Modulus control is then set back low, the counters preset to their respective programmed values, and the above sequence repeated. This provides for a total programmable divide value (NT) = N·P + A where P and P + 1 represent the dual-modulus prescaler divide values respectively for high and low modulus control levels. N the number programmed into the ÷ N counter, and A the number programmed into the ÷ A counter.

LD—Lock Detector Output

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

POWER SUPPLY

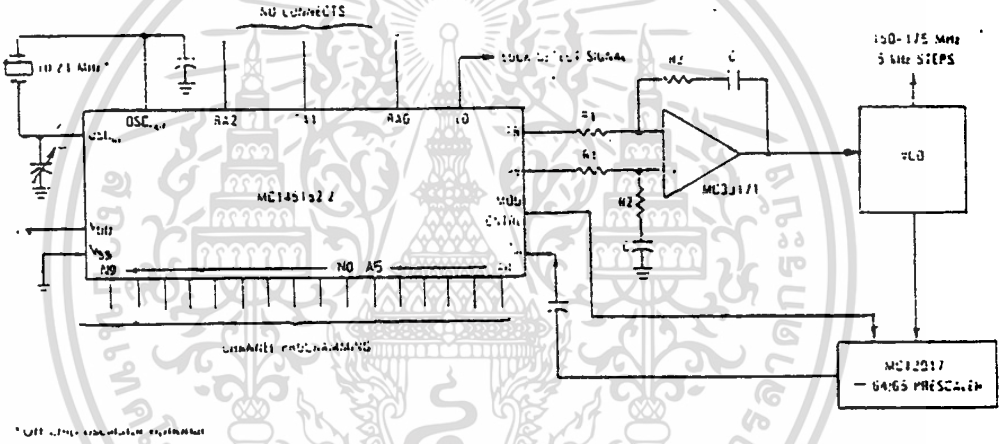
V_{DD}

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

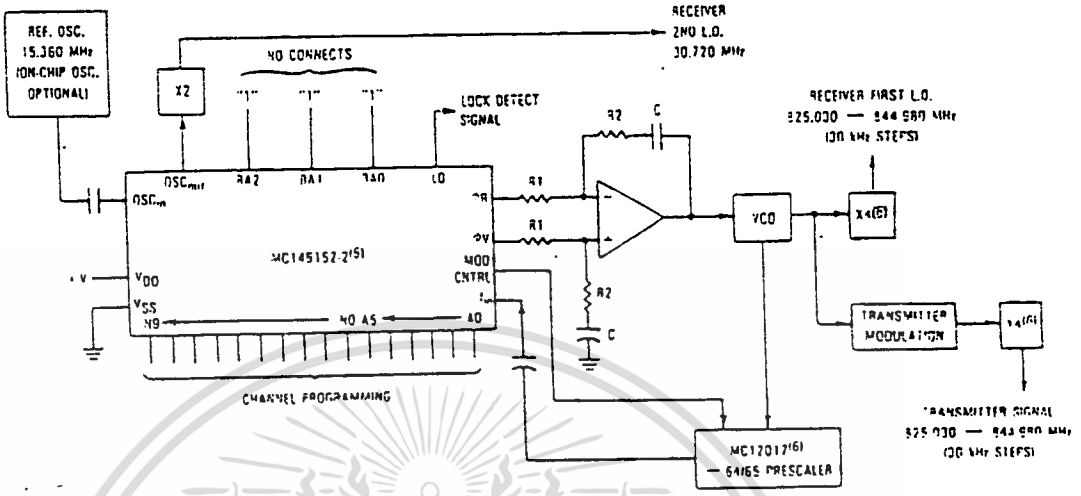
V_{SS}

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

TYPICAL APPLICATIONS



Synthesizer for Land Mobile Radio VHF Bands



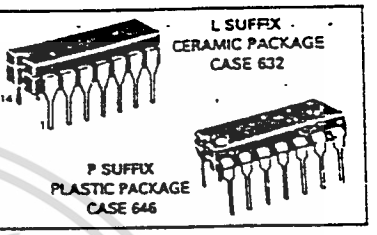
NOTES:
1. Receiver 1st. I.F. = 45 MHz, low side injection. Receiver 2nd. I.F. = 11.7 MHz, low side injection.
2. Duplex operation with 45 MHz receiver/transmit separation.
3. f_R = 7.5 kHz, — R = 2048.
4. N_{total} = N · 64 + A = 27501 to 28166; N = 429 to 440; A = 0 to 63.
5. MC145158-2 may be used where serial data entry is desired.
6. High frequency prescalers — e.g., MC12018 (520 MHz) and MC12022 (1 GHz) — may be used for higher frequency VCO and [illegible]

666-Channel, Computer-Controlled, Mobile Radiotelephone Synthesizer for
800 MHz Cellular Radio Systems

MOTOROLA

MC1648

VOLTAGE-CONTROLLED OSCILLATOR



VOLTAGE-CONTROLLED OSCILLATOR

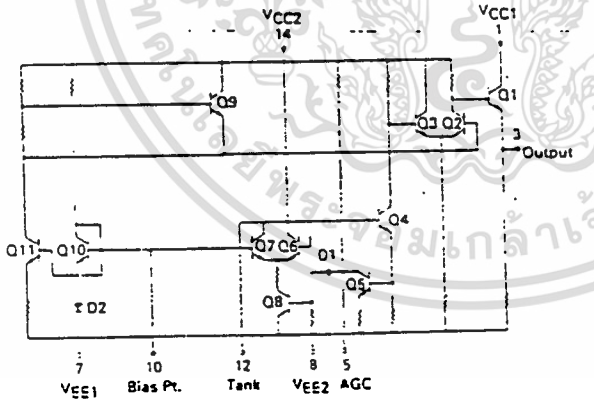
The MC1648 requires an external parallel tank circuit consisting of the inductor (L) and capacitor (C).

A varactor diode may be incorporated into the tank circuit to provide a voltage variable input for the oscillator (VCO). The MC1648 was designed for use in the Motorola Phase-Locked Loop shown in Figure 9. This device may also be used in many other applications requiring a fixed or variable frequency clock source of high spectral purity. (See Figure 2.)

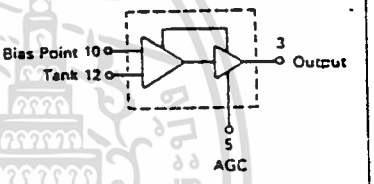
The MC1648 may be operated from a +5.0 Vdc supply or a −5.2 Vdc supply, depending upon system requirements.

Supply Voltage	Gnd Pins	Supply Pins
+5.0 Vdc	7, 8	1, 14
−5.2 Vdc	1, 14	7, 8

FIGURE 1 — CIRCUIT SCHEMATIC



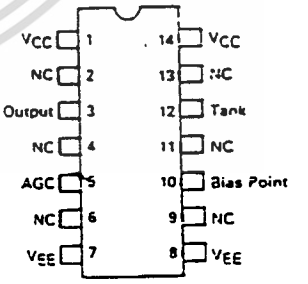
LOGIC DIAGRAM



Input Capacitance = 6.0 pF typ
Maximum Series Resistance for L (External Inductance) = 50 Ω typ
Power Dissipation = 150 mW typ/pkg (−5.0 Vdc Supply)
Maximum Output Frequency = 225 MHz typ

V_{CC1} = Pin 1
V_{CC2} = Pin 14
V_{EE} = Pin 7

PIN ASSIGNMENT



@ Test Temperature	TEST VOLTAGE/CURRENT VALUES (Volts) V_{IHmax}	V_{ILmin}	V_{CC}	mAdc I_L
	MC1648			
-30°C	-2.0	-1.5	5.0	-5.0
+25°C	-1.85	-1.35	5.0	-5.0
+85°C	-1.7	-1.2	5.0	-5.0

ELECTRICAL CHARACTERISTICS

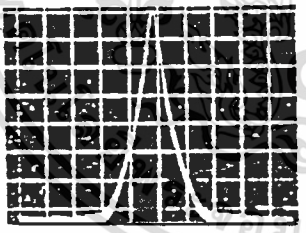
Supply Voltage = +5.0 Volts

Characteristic	Symbol	-30°C Min	-30°C Max	+25°C Min	+25°C Max	+85°C Min	+85°C Max	Unit	Conditions
Power Supply Drain Current	I_E	—	—	—	41	—	—	mAdc	Inputs and outputs open
Logic "1" Output Voltage	V_{OH}	3.955	4.185	4.04	4.25	4.11	4.36	Vdc	V_{ILmin} to Pin 12, I_L @ Pin 3.
Logic "0" Output Voltage	V_{OL}	3.16	3.4	3.2	3.43	3.22	3.475	Vdc	V_{IHmax} to Pin 12, I_L @ Pin 3.
Bias Voltage	V_{Bias}*	1.6	1.9	1.45	1.75	1.3	1.6	Vdc	V_{ILmin} to Pin 12.

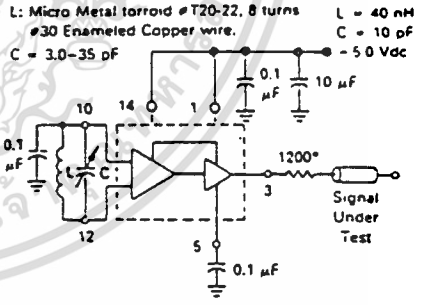
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Peak-to-Peak Tank Voltage	V_{p-p}	—	—	—	—	400	—	—	—	—	mV	See Figure 3.
Output Duty Cycle	Vdc	—	—	—	—	50	—	—	—	—	%	
Oscillation Frequency	f_{max}**	—	225	—	200	225	—	—	225	—	MHz	

*This measurement guarantees the dc potential at the bias point for purposes of incorporating a varactor tuning diode at this point.
**Frequency variation over temperature is a direct function of the ΔC/Δ Temperature and ΔL/Δ Temperature.

FIGURE 2 — SPECTRAL PURITY OF SIGNAL OUTPUT FOR 200 MHz TESTING



B.W. = 10 kHz
Center Frequency = 100 MHz
Scan Width = 50 kHz/div
Vertical Scale = 10 dB/div



*The 1200 ohm resistor and the scope termination impedance constitute a 25:1 attenuator probe. Coax shall be CT-070-50 or equivalent.

@ Test Temperature	TEST VOLTAGE/CURRENT VALUES (Volts) V_{IHmax}	V_{ILmin}	V_{CC}	mAdc I_L
MC1648				
-30°C	-3.2	-3.7	-5.2	-5.0
-25°C	-3.35	-3.85	-5.2	-5.0
-85°C	-3.5	-4.0	-5.2	-5.0

ELECTRICAL CHARACTERISTICS

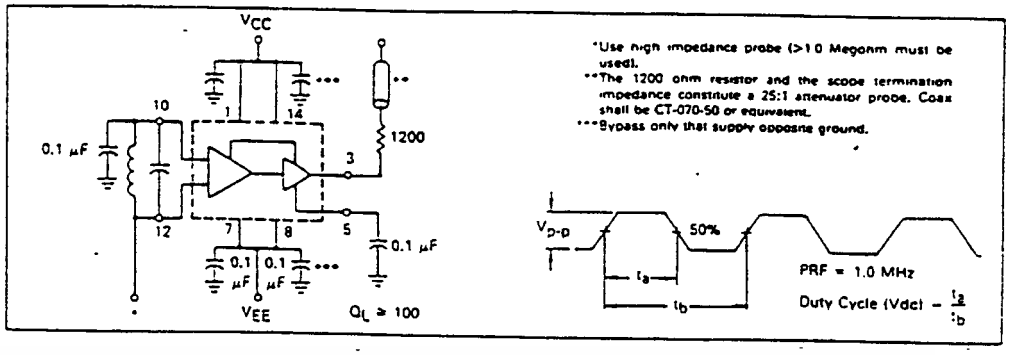
Supply Voltage = -5.0 Volts

Characteristic	Symbol	-30°C Min	-30°C Max	+25°C Min	+25°C Max	+85°C Min	+85°C Max	Unit	Conditions
Power Supply Drain Current	I_E	—	—	—	41	—	—	mAdc	Inputs and outputs open.
Logic "1" Output Voltage	V_{OH}	-1.045	-0.815	-0.96	-0.75	-0.89	-0.64	Vdc	V_{ILmin} to Pin 12, I_L @ Pin 3.
Logic "0" Output Voltage	V_{OL}	-1.89	-1.65	-1.85	-1.62	-1.83	-1.575	Vdc	V_{IHmax} to Pin 12, I_L @ Pin 3.
Bias Voltage	V_{Bias}*	-3.6	-3.3	-3.75	-3.45	-3.9	-3.6	Vdc	V_{ILmin} to Pin 12.
		Min	Typ Max	Min	Typ Max	Min	Typ Max		
Peak-to-Peak Tank Voltage	V_{P-P}	—	— —	—	400 —	—	— —	mV	See Figure 3.
Output Duty Cycle	Vdc	—	— —	—	50 —	—	— —	%	
Oscillation Frequency	f_{max}**	—	225 —	200	225 —	—	225 —	MHz	

*This measurement guarantees the dc potential at the bias point for purposes of incorporating a varactor tuning diode at this point.

**Frequency variation over temperature is a direct function of the ΔC/Δ Temperature and ΔL/Δ Temperature.

FIGURE 3 — TEST CIRCUIT AND WAVEFORMS



OPERATING CHARACTERISTICS

Figure 1 illustrates the circuit schematic for the MC1648. The oscillator incorporates positive feedback by coupling the base of transistor Q6 to the collector of Q7. An automatic gain control (AGC) is incorporated to limit the current through the emitter-coupled pair of transistors (Q7 and Q6) and allow optimum frequency response of the oscillator.

In order to maintain the high Q of the oscillator, and provide high spectral purity at the output, transistor Q4 is used to translate the oscillator signal to the output differential pair Q2 and Q3. Q2 and Q3, in conjunction with output transistor Q1, provides a highly buffered output which produces a square wave. Transistors Q9 and Q11 provide the bias drive for the oscillator and output buffer. Figure 2 indicates the high spectral purity of the oscillator output (pin 3).

When operating the oscillator in the voltage controlled mode (Figure 4), it should be noted that the cathode of the varactor diode (D) should be biased at least "2" V_{BE} above V_{EE} (≈1.4 V for positive supply operation).

When the MC1648 is used with a constant dc voltage to the varactor diode, the output frequency will vary slightly because of internal noise. This variation is plotted versus operating frequency in Figure 5.

FIGURE 4 — THE MC1648 OPERATING IN THE VOLTAGE CONTROLLED MODE

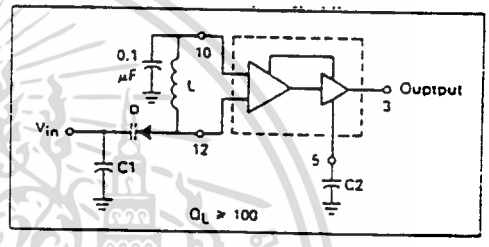
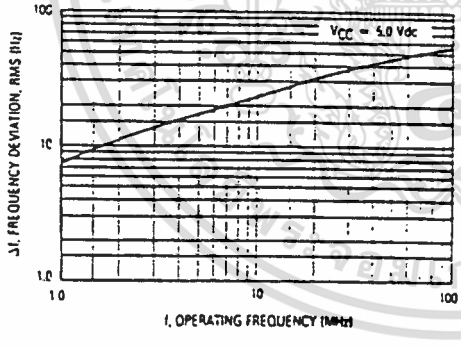
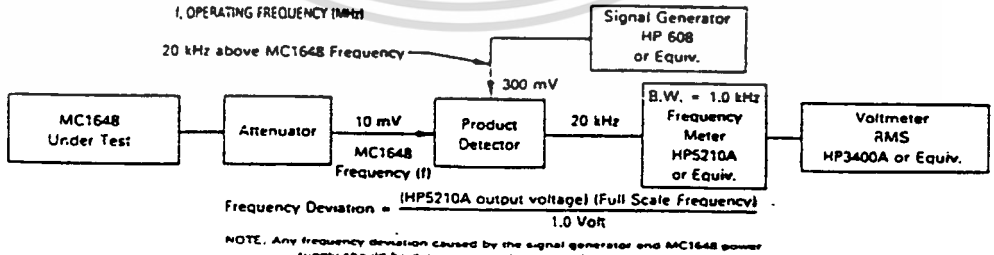


FIGURE 5 — NOISE DEVIATION TEST CIRCUIT AND WAVEFORM



Oscillator Tank Components
(Circuit of Figure 4)

f MHz	D	L μH
1.0–10	MV2115	100
10–60	MV2116	2.3
60–100	MV2106	0.15



$$\text{Frequency Deviation} = \frac{\text{(HP5210A output voltage) (Full Scale Frequency)}}{\text{1.0 Volt}}$$

NOTE: Any frequency deviation caused by the signal generator and MC1648 power supply should be determined and minimized prior to testing.

TRANSFER CHARACTERISTICS IN THE VOLTAGE CONTROLLED MODE USING EXTERNAL VARACTOR DIODE AND COIL. T_A = 25°C

FIGURE 6

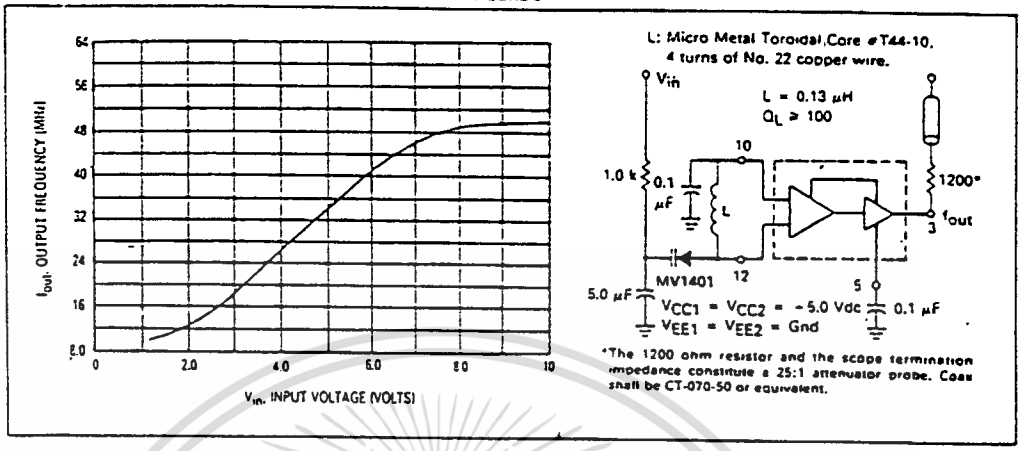


FIGURE 7

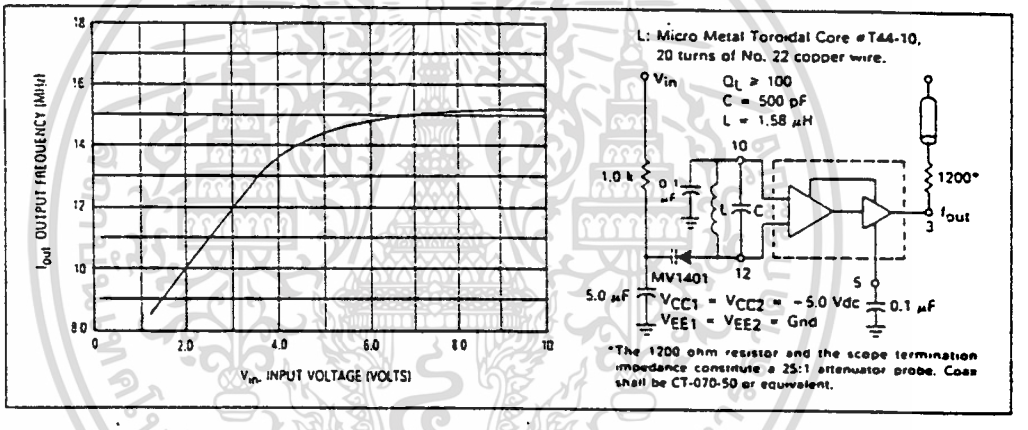
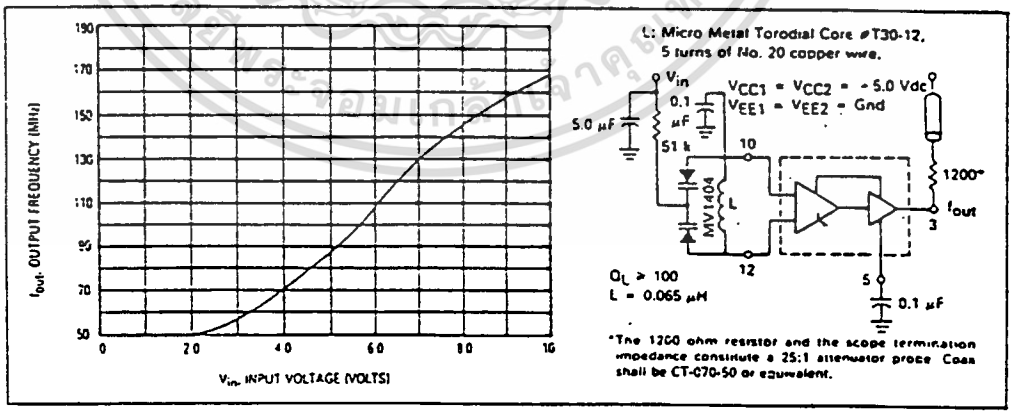


FIGURE 8



Typical transfer characteristics for the oscillator in the voltage controlled mode are shown in Figures 6, 7, and 8. Figures 6 and 8 show transfer characteristics employing only the capacitance of the varactor diode (plus the input capacitance of the oscillator, 6.0 pF typical). Figure 7 illustrates the oscillator operating in a voltage controlled mode with the output frequency range limited. This is achieved by adding a capacitor in parallel with the tank circuit as shown. The 1.0 kΩ resistor in Figures 6 and 7 is used to protect the varactor diode during testing. It is not necessary as long as the dc input voltage does not cause the diode to become forward biased. The larger-valued resistor (51 kΩ) in Figure 8 is required to provide isolation for the high-impedance junctions of the two varactor diodes.

The tuning range of the oscillator in the voltage controlled mode may be calculated as:

$$\frac{f_{max}}{f_{min}} = \frac{\sqrt{C_{D(max)} + C_S}}{\sqrt{C_{D(min)} + C_S}}$$

where $f_{min} = \frac{1}{2\pi\sqrt{LI(C_{D(max)} + C_S)}}$

C_S = shunt capacitance (input plus external capacitance).

C_D = varactor capacitance as a function of bias voltage.

Good RF and low-frequency bypassing is necessary on the power supply pins. (See Figure 2.)

Capacitors (C1 and C2 of Figure 4) should be used to bypass the AGC point and the VCO input (varactor diode), guaranteeing only dc levels at these points.

For output frequency operation between 1.0 MHz and 50 MHz a 0.1 μF capacitor is sufficient for C1 and C2. At higher frequencies, smaller values of capacitance should be used; at lower frequencies, larger values of capacitance. At high frequencies the value of bypass capacitors depends directly upon the physical layout of the system. All bypassing should be as close to the package pins as possible to minimize unwanted lead inductance.

The peak-to-peak swing of the tank circuit is set internally by the AGC circuitry. Since voltage swing of the tank circuit provides the drive for the output buffer, the AGC potential directly affects the output waveform. If it is desired to have a sine wave at the output of the MC1648, a series resistor is tied from the AGC point to the most negative power potential (ground if − 5.0 volt supply is used, − 5.2 volts if a negative supply is used) as shown in Figure 10.

At frequencies above 100 MHz typ, it may be desirable to increase the tank circuit peak-to-peak voltage in order to shape the signal at the output of the MC1648. This is accomplished by tying a series resistor (1.0 kΩ minimum) from the AGC to the most positive power potential (+ 5.0 volts if a + 5.0 volt supply is used, ground if a − 5.2 volt supply is used). Figure 11 illustrates this principle.

APPLICATIONS INFORMATION

The phase locked loop shown in Figure 9 illustrates the use of the MC1648 as a voltage controlled oscillator. The figure illustrates a frequency synthesizer useful in tuners for FM broadcast, general aviation, maritime and land-mobile communications, amateur and CB receivers. The system operates from a single +5.0 Vdc supply, and requires no internal translations, since all components are compatible.

Frequency generation of this type offers the advantages of single crystal operation, simple channel selection, and elimination of special circuitry to prevent harmonic lockup. Additional features include dc digital switching (preferable over RF switching with a multiple crystal system), and a broad range of tuning (up to 150 MHz, the range being set by the varactor diode).

The output frequency of the synthesizer loop is determined by the reference frequency and the number programmed at the programmable counter; $f_{out} = Nf_{ref}$. The channel spacing is equal to frequency (f_{ref}).

For additional information on applications and designs for phase locked-loops and digital frequency synthesizers, see Motorola Brochure BR504/D, Electronic Tuning Address Systems, (ETAS).

Figure 10 shows the MC1648 in the variable frequency mode operating from a + 5.0 Vdc supply. To obtain a sine wave at the output, a resistor is added from the AGC circuit (pin 5) to V_{EE}.

Figure 11 shows the MC1648 in the variable frequency mode operating from a + 5.0 Vdc supply. To extend the useful range of the device (maintain a square wave output

MOTOROLA

MC12015
MC12016
MC12017

MECL PLL COMPONENTS

LOW-POWER
TWO—MODULUS
PRESCALER

P SUFFIX
PLASTIC PACKAGE
CASE 626

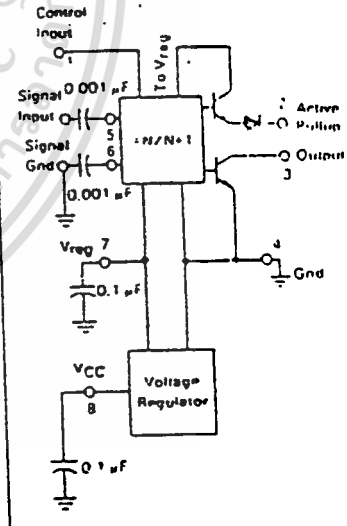
D SUFFIX
PLASTIC SOIC PACKAGE
CASE 751

LOW-POWER TWO-MODULUS PRESCALER

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

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

PRESCALER BLOCK DIAGRAM



1. Vreg @ pin 7 is not guaranteed to be between 4.5 and 5.5 V when VCC is being applied to pin 8.
2. Pin 7 is not to be used as a source of regulated output voltage.

MAXIMUM RATINGS

Characteristic	Symbol	Range	Unit
Regulated Voltage, Pin 7	V_{reg}	8.0	Vdc
Power Supply Voltage, Pin 8	V_{CC}	10.0	Vdc
Operating Temperature Range	T_A	-40 to +85	°C
Storage Temperature Range	T_{stg}	-65 to +175	°C

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.5 to 9.5, V_{reg} = 4.5 to 5.5 V, T_A = -40°C to +85°C)

Characteristic	Symbol	Min	Typ	Max	Unit
Toggle Frequency (Sine wave input)	f_{max}	225	—	—	MHz
	f_{min}	—	—	35	MHz
Supply Current	I_{CC}	—	6.0	7.8	mA
Control Input High (÷32, 40 or 64)		2.0	—	—	V
Control Input Low (÷33, 41 or 65)		—	—	0.8	V
Output Voltage High* (I_{source} = 50 µA)	V_{OH}	2.5	—	—	V
Output Voltage Low* (I_{sink} = 2 mA)	V_{OL}	—	—	0.5	V
Input Voltage Sensitivity 35 MHz	V_{in}	400	—	300	mVPP
50-225 MHz		200	—	800	
PLL Response Time (Notes 1 and 2)	t_{PLL}	—	—	t_{out} -70	ns

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

*Pin 2 connected to Pin 3

ORDERING INFORMATION

Device	Temperature Range	Package
MC1496B	0°C to +70°C	SO-14
MC1496G		Metal Can
MC1496L		Ceramic DIP
MC1496P		Plastic DIP
MC1596G	−55°C to +125°C	Metal Can
MC1596L		Ceramic DIP

MC1496
MC1596

Specifications and Applications Information

BALANCED MODULATOR/DEMODULATOR

SILICON MONOLITHIC INTEGRATED CIRCUIT

BALANCED MODULATOR/ DEMODULATOR

. . . designed for use where the output voltage is a product of an input voltage (signal) and a switching function (carrier). Typical applications include suppressed carrier and amplitude modulation, synchronous detection, FM detection, phase detection, and chopper applications. See Motorola Application Note AN-531 for additional design information.

- Excellent Carrier Suppression — 65 dB typ @ 0.5 MHz
  — 50 dB typ @ 10 MHz
- Adjustable Gain and Signal Handling
- Balanced Inputs and Outputs
- High Common Mode Rejection — 85 dB typ

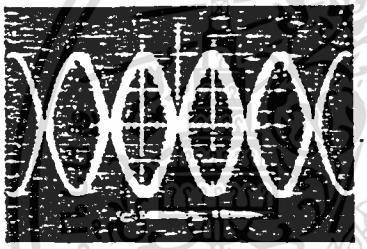


FIGURE 1 — SUPPRESSED-CARRIER OUTPUT WAVEFORM

FIGURE 2 — SUPPRESSED-CARRIER SPECTRUM

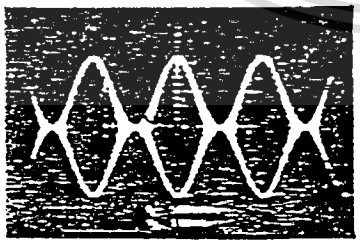


FIGURE 3 — AMPLITUDE MODULATION OUTPUT WAVEFORM

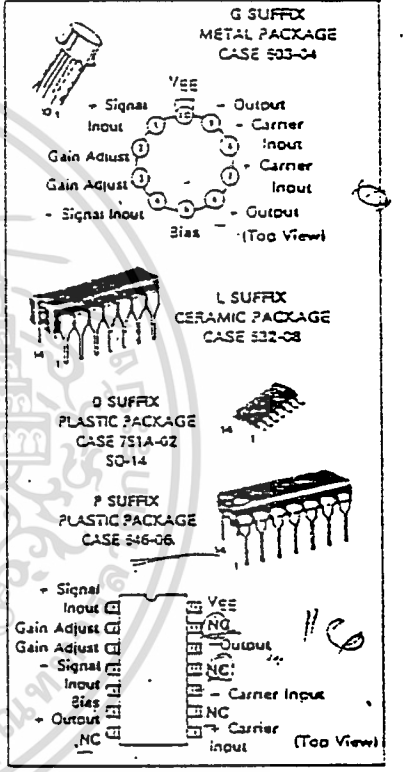
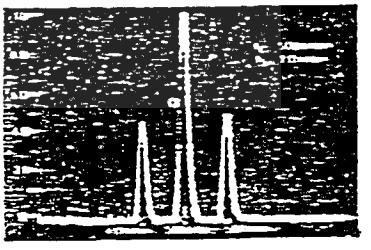


FIGURE 4 — AMPLITUDE-MODULATION SPECTRUM



MAXIMUM RATINGS* (T_A = +25°C unless otherwise noted)

Rating	Symbol	Value	Unit
Applied Voltage ($V_5 - V_7$, $V_9 - V_1$, $V_9 - V_7$, $V_9 - V_8$, $V_7 - V_4$, $V_7 - V_1$, $V_8 - V_4$, $V_6 - V_8$, $V_2 - V_5$, $V_7 - V_5$)	ΔV	30	Vdc
Differential Input Signal	$V_7 - V_8$ $V_4 - V_1$	±5.0 ±(5 + I_5R_e)	Vdc
Maximum Bias Current	I_5	10	mA
Thermal Resistance, Junction to Air Ceramic Dual In-Line Package Plastic Dual In-Line Package Metal Package	$R_{\theta JA}$	 100 100 160	°C/W
Operating Temperature Range MC1496 MC1596	T_A	 0 to +70 −55 to +125	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C

ELECTRICAL CHARACTERISTICS* (V_{CC} = +12 Vdc, V_{EE} = −8.0 Vdc, I_5 = 1.0 mAdc, R_L = 3.9 kΩ, R_e = 1.0 kΩ, T_A = +25°C unless otherwise noted) (All input and output characteristics are single-ended unless otherwise noted.)

Characteristic		Fig.	Note	Symbol	MC1596 Min	MC1596 Typ	MC1596 Max	MC1496 Min	MC1496 Typ	MC1496 Max	Unit
Carrier Feedthrough V_C = 60 mV(rms) sine wave and offset adjusted to zero	f_C = 1.0 kHz	5	1	V_{CFT}	—	40	—	—	40	—	μV(rms)
	f_C = 10 MHz				—	140	—	—	140	—	
V_C = 300 mV(p-p square wave): offset adjusted to zero	f_C = 1.0 kHz				—	0.04	0.2	—	0.04	0.4	mV(rms)
offset not adjusted	f_C = 1.0 kHz				—	20	100	—	20	200	
Carrier Suppression f_S = 10 kHz, 300 mV(rms) f_C = 500 kHz, 60 mV(rms) sine wave		5	2	V_{CS}	50	55	—	40	55	—	dB
f_C = 10 MHz, 60 mV(rms) sine wave					—	50	—	—	50	—	k
Transadmittance Bandwidth (Magnitude) (R_L = 50 ohms) Carrier Input Port, V_C = 60 mV(rms) sine wave f_S = 1.0 kHz, 300 mV(rms) sine wave		8	3	BW_{3dB}	—	300	—	—	300	—	MHz
Signal Input Port, V_S = 300 mV(rms) sine wave $\|V_{CI}\|$ = 0.5 Vdc					—	80	—	—	80	—	
Signal Gain V_S = 100 mV(rms), f = 1.0 kHz; $\|V_{CI}\|$ = 0.5 Vdc		10	3	A_{VS}	2.5	3.5	—	2.5	3.5	—	V/V
Single-Ended Input Impedance, Signal Port, f = 5.0 MHz Parallel Input Resistance		6	—	r_{io}	—	200	—	—	200	—	kΩ
Parallel Input Capacitance				c_{io}	—	2.0	—	—	2.0	—	pF
Single-Ended Output Impedance, f = 10 MHz Parallel Output Resistance		5	—	r_{oo}	—	40	—	—	40	—	kΩ
Parallel Output Capacitance				c_{oo}	—	5.0	—	—	5.0	—	pF
Input Bias Current $I_{bS} = \frac{I_1 + I_4}{2}$; $I_{bC} = \frac{I_7 + I_8}{2}$		7	—	I_{bS}	—	12	25	—	12	30	μA
				I_{bC}	—	12	25	—	12	30	
Input Offset Current $I_{ioS} = \|I_1 - I_4\|$; $I_{ioC} = \|I_7 - I_8\|$		7	—	$\|I_{ioS}\|$	—	0.7	5.0	—	0.7	7.0	μA
				$\|I_{ioC}\|$	—	0.7	5.0	—	0.7	7.0	
Average Temperature Coefficient of Input Offset Current (T_A = −55°C to +125°C)		7	—	$\|TC_{Iio}\|$	—	2.0	—	—	2.0	—	nA/°C
Output Offset Current ($I_6 - I_9$)		7	—	$\|I_{oo}\|$	—	14	50	—	14	80	μA
Average Temperature Coefficient of Output Offset Current (T_A = −55°C to +125°C)		7	—	$\|TC_{Ioo}\|$	—	90	—	—	90	—	nA/°C
Common-Mode Input Swing, Signal Port, f_S = 1.0 kHz		9	4	CMV	—	5.0	—	—	5.0	—	V_{p-p}
Common-Mode Gain, Signal Port, f_S = 1.0 kHz, $\|V_{CI}\|$ = 0.5 Vdc		9	—	ACM	—	−85	—	—	−85	—	dB
Common-Mode Quiescent Output Voltage (Pin 6 or Pin 9)		10	—	V_{out}	—	8.0	—	—	8.0	—	V_{dc}
Differential Output Voltage Swing Capability		10	—	V_{out}	—	8.0	—	—	8.0	—	V_{p-p}
Power Supply Current $I_6 + I_9$		7	6	I_{CC}	—	2.0	3.0	—	2.0	4.0	mAdc
I_{10}				I_{EE}	—	3.0	4.0	—	3.0	5.0	
DC Power Dissipation		7	5	P_D	—	33	—	—	33	—	mW

* Pin number references pertain to this device when packaged in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic packaged devices refer to the first page of this specification sheet.

GENERAL OPERATING INFORMATION *

Note 1 — Carrier Feedthrough

Carrier feedthrough is defined as the output voltage at carrier frequency with only the carrier applied (signal voltage = 0).

Carrier null is achieved by balancing the currents in the differential amplifier by means of a bias trim potentiometer (R_1 of Figure 5).

Note 2 — Carrier Suppression

Carrier suppression is defined as the ratio of each sideband output to carrier output for the carrier and signal voltage levels specified.

Carrier suppression is very dependent on carrier input level, as shown in Figure 22. A low value of the carrier does not fully switch the upper switching devices, and results in lower signal gain, hence lower carrier suppression. A higher than optimum carrier level results in unnecessary device and circuit carrier feedthrough, which again degenerates the suppression figure. The MC1596 has been characterized with a 60 mV(rms) sinewave carrier input signal. This level provides optimum carrier suppression at carrier frequencies in the vicinity of 500 kHz, and is generally recommended for balanced modulator applications.

Carrier feedthrough is independent of signal level, V_S. Thus carrier suppression can be maximized by operating with large signal levels. However, a linear operating mode must be maintained in the signal-input transistor pair — or harmonics of the modulating signal will be generated and appear in the device output as spurious sidebands of the suppressed carrier. This requirement places an upper limit on input-signal amplitude (see Note 3 and Figure 20). Note also that an optimum carrier level is recommended in Figure 22 for most carrier suppression and minimum spurious sideband generation.

At higher frequencies circuit layout is very important in order to minimize carrier feedthrough. Shielding may be necessary in order to prevent capacitive coupling between the carrier input leads and the output leads.

Note 3 — Signal Gain and Maximum Input Level

Signal gain (single-ended) at low frequencies is defined as the voltage gain

$$A_{VS} = \frac{V_o}{V_S} = \frac{R_L}{R_e + 2r_e} \quad \text{where } r_e = \frac{25\ mV}{I_5\ (mA)}$$

A constant dc potential is applied to the carrier input terminals to fully switch two of the upper transistors "on" and two transistors "off" (V_C = 0.5 Vdc). This in effect forms a cascode differential amplifier.

Linear operation requires that the signal input be below a critical value determined by R_E and the bias current I_5

$$V_S \leq I_5 R_E \text{ (Volts peak)}$$

Note that in the test circuit of Figure 10, V_S corresponds to a maximum value of 1 volt peak.

Note 4 — Common-Mode Swing

The common-mode swing is the voltage which may be applied to both bases of the signal differential amplifier, without saturating the current sources or without saturating the differential amplifier itself by swinging it into the upper switching devices. This swing is variable depending on the particular circuit and biasing conditions chosen (see Note 6).

Note 5 — Power Dissipation

Power dissipation, P_D, within the integrated circuit package should be calculated as the summation of the voltage-current products at each port (i.e., assuming $V_8 - V_5$, $I_5 \approx I_8$ and ignoring base current, $P_D = 2\,I_5\,(V_5 - V_{10}) + I_5\,(V_5 - V_{10})$ where subscripts refer to pin numbers).

Note 6 — Design Equations

The following is a partial list of design equations needed to operate the circuit with other supply voltages and input conditions. See Note 3 for R_E equation.

A. Operating Current

The internal bias currents are set by the resistors at pin 5. Assume:

$$I_5 = I_6 = I_8$$

$$I_8 \ll I_C \text{ for all transistors}$$

then

$$R_5 = \frac{V^- - 0.7}{I_5} - 500\ \Omega \quad \text{where } R_5 \text{ is the resistor between pin 5 and ground}$$

0.7 V at $T_A = +25^\circ C$

The MC1596 has been characterized for the condition I_5 = 1.0 mA and is the generally recommended value.

B. Common Mode Quiescent Output Voltage

$$V_5 = V_8 = V^+ - I_5 R_L$$

Note 7 — Biasing

The MC1596 requires three dc bias voltage levels which must be set externally. Guidelines for setting up these three levels include maintaining at least 2 volts collector-base bias on all transistors while not exceeding the voltages given in the absolute maximum rating table.

$$20\ \text{Vdc} \geq |(V_6, V_8) - (V_7, V_8)| \geq 2\ \text{Vdc}$$

$$20\ \text{Vdc} \geq |(V_7, V_8) - (V_1, V_4)| \geq 2.7\ \text{Vdc}$$

$$20\ \text{Vdc} \geq |(V_1, V_4) - (V_5)| \geq 2.7\ \text{Vdc}$$

The foregoing conditions are based on the following approximations:

$$V_5 = V_8, \quad V_7 = V_8, \quad V_1 = V_4$$

Bias currents flowing into pins 1, 4, 7, and 8 are transistor base currents and can normally be neglected if external bias dividers are designed to carry 1.0 mA or more.

Note 8 — Transadmittance Bandwidth

Carrier transadmittance bandwidth is the 3-dB bandwidth of the device forward transadmittance as defined by:

$$Y_{21C} = \left| \frac{i_o\ (\text{each sideband})}{v_i\ (\text{signal})} \right|_{V_o = 0}$$

Signal transadmittance bandwidth is the 3-dB bandwidth of the device forward transadmittance as defined by:

$$Y_{21S} = \left| \frac{i_o\ (\text{signal})}{v_i\ (\text{signal})} \right|_{V_C = 0.5\ \text{Vdc},\ V_o = 0}$$

*Pin numbers references pertain to this device when packaged in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic packaged devices refer to the first page of this specification sheet.

Note 9 – Coupling and Bypass Capacitors C_1 and C_2

Capacitors C_1 and C_2 (Figure 5) should be selected for a reactance of less than 5.0 ohms at the carrier frequency.

Note 10 – Output Signal, V_o

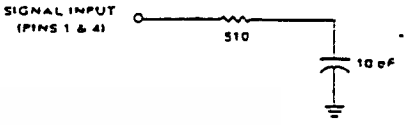
The output signal is taken from pins 6 and 9, either balanced or single-ended. Figure 12 shows the output levels of each of the two output sidebands resulting from variations in both the carrier and modulating signal inputs with a single-ended output connection.

Note 11 – Negative Supply, V_{EE}

V_{EE} should be dc only. The insertion of an RF choke in series with V_{EE} can enhance the stability of the internal current sources.

Note 12 – Signal Port Stability

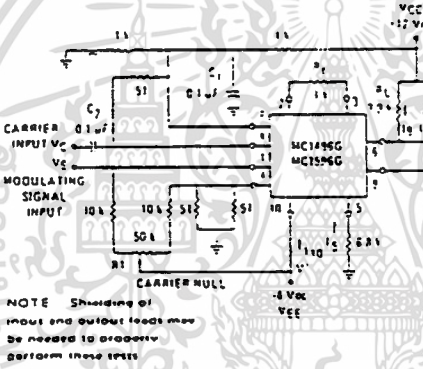
Under certain values of driving source impedance, oscillation may occur. In this event, an RC suppression network should be connected directly to each input using short leads. This will reduce the Q of the source-tuned circuits that cause the oscillation.



An alternate method for low-frequency applications is to insert a 1 kohm resistor in series with the inputs, pins 1 and 4. In this case input current drift may cause serious degradation of carrier suppression.

TEST CIRCUITS

FIGURE 5 – CARRIER REJECTION AND SUPPRESSION



NOTE Shielding of input and output leads may be needed to properly perform these tests.

FIGURE 6 – INPUT-OUTPUT IMPEDANCE

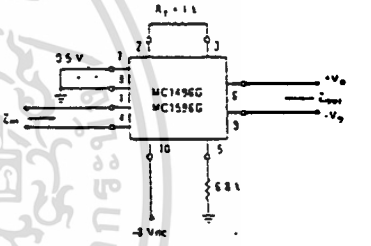


FIGURE 7 – BIAS AND OFFSET CURRENTS

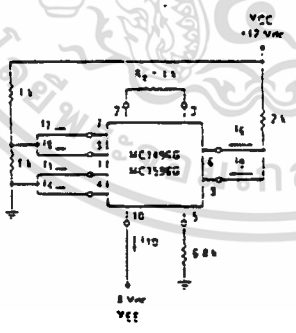
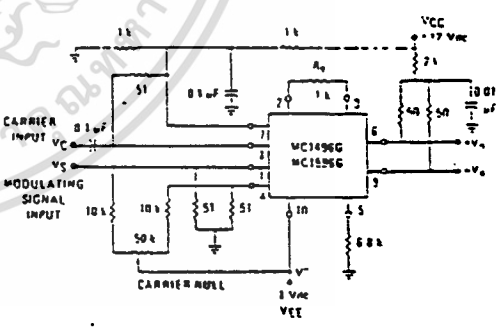


FIGURE 8 – TRANSCONDUCTANCE BANDWIDTH



NOTE Pin number references pertain to this device when packaged in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic packaged devices refer to the first page of this specification sheet.

TEST CIRCUITS (continued)

FIGURE 9 – COMMON-MODE GAIN

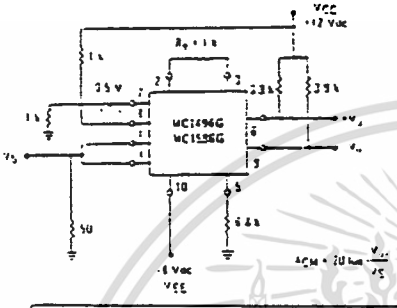
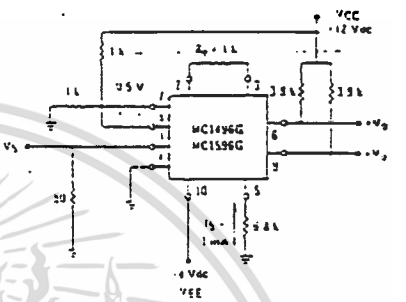


FIGURE 10 – SIGNAL GAIN AND OUTPUT SWING



*NOTE: Pin numbers referenced apply to this device when used in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic dual-in-line packages, refer to the test circuit of the specific device sheet.

TYPICAL CHARACTERISTICS (continued)

Typical characteristics were obtained using circuit shown in Figure 9 (f_C = 500 kHz (sine wave), V_{C} = 60 mV(rms), f_S = 1 kHz, V_S = 300 mV(rms), T_A = +25°C unless otherwise noted.

FIGURE 11 – SIDEBAND OUTPUT versus CARRIER LEVELS

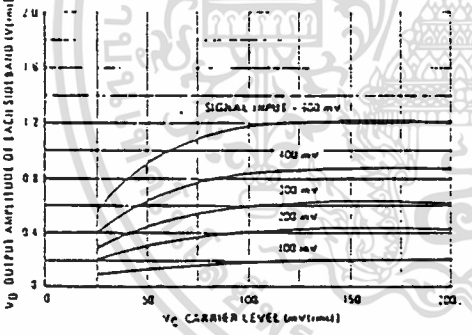


FIGURE 12 – SIGNAL-PORT PARALLEL EQUIVALENT INPUT RESISTANCE versus FREQUENCY

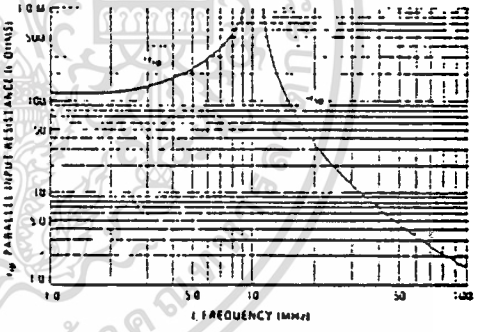


FIGURE 13 – SIGNAL-PORT PARALLEL-EQUIVALENT INPUT CAPACITANCE versus FREQUENCY

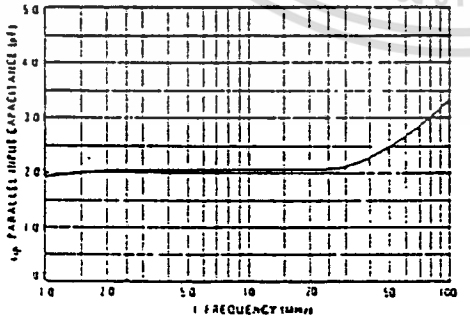
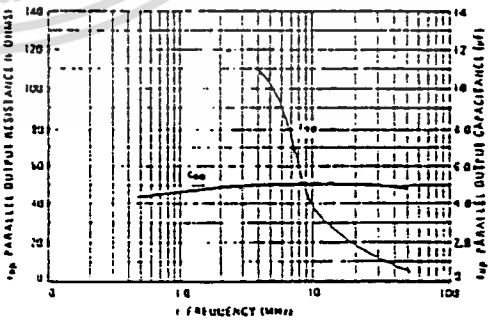


FIGURE 14 – SINGLE-ENDED OUTPUT IMPEDANCE versus FREQUENCY



TYPICAL CHARACTERISTICS (continued)

Typical characteristics were obtained with circuit shown in Figure 5, f_C = 500 kHz (sine wave), V_C = 60 mV(rms), f_S = 1 kHz, V_S = 300 mV(rms), T_A = +25°C unless otherwise noted.

FIGURE 15 – SIDEBAND AND SIGNAL PORT TRANSADMITTANCES versus FREQUENCY

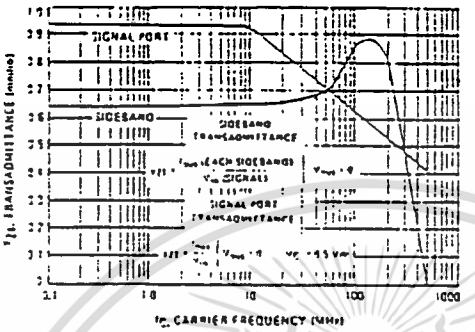


FIGURE 16 – CARRIER SUPPRESSION versus TEMPERATURE

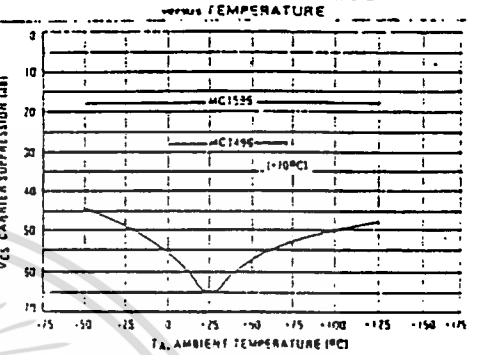


FIGURE 17 – SIGNAL PORT FREQUENCY RESPONSE

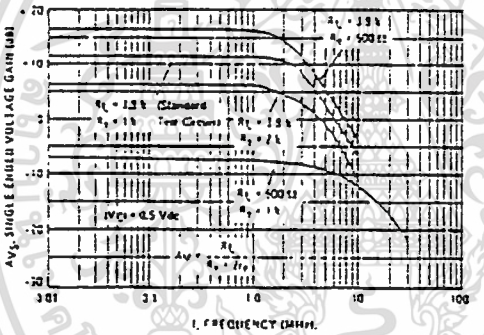


FIGURE 18 – CARRIER SUPPRESSION versus FREQUENCY

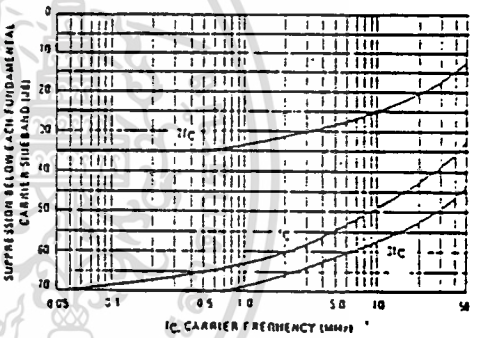


FIGURE 19 – CARRIER FEEDTHROUGH versus FREQUENCY

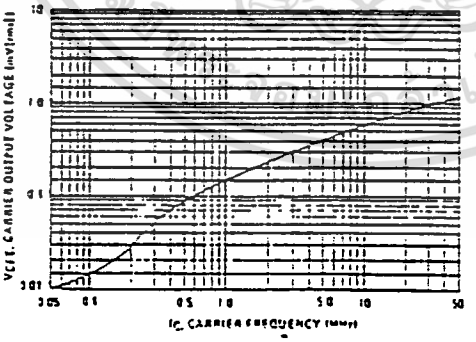
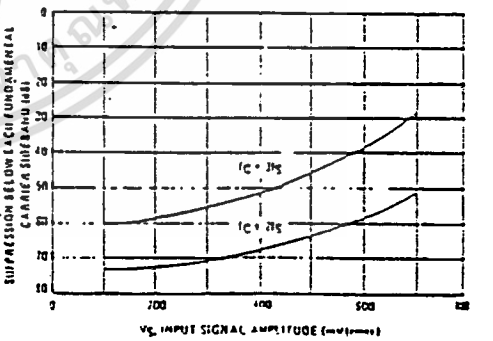


FIGURE 20 – SIDEBAND HARMONIC SUPPRESSION versus INPUT SIGNAL LEVEL



TYPICAL CHARACTERISTICS (continued)

FIGURE 21 – SUPPRESSION OF CARRIER HARMONIC SIDEBANDS versus CARRIER FREQUENCY

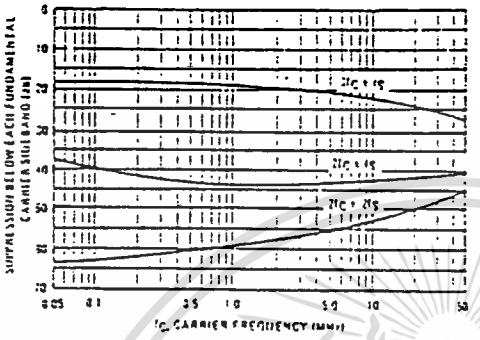
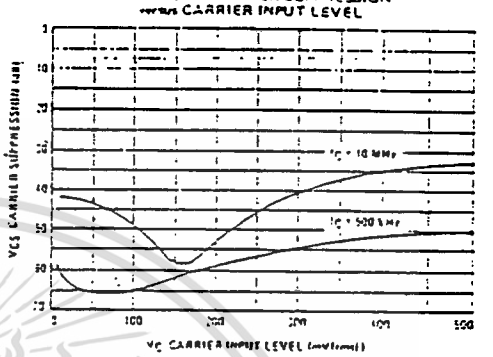


FIGURE 22 – CARRIER SUPPRESSION versus CARRIER INPUT LEVEL



OPERATIONS INFORMATION

The MC1595/MC1496, a monolithic balanced modulator circuit, is shown in Figure 23.

The circuit consists of an upper quad differential amplifier driven by a standard differential amplifier with dual current sources. The output collectors are cross-coupled so that full-wave balanced multiplication of the two input voltages occurs. That is, the output signal is a constant times the product of the two input signals.

Mathematical analysis of linear ac signal multiplication indicates that the output spectrum will consist of only the sum and difference of the two input frequencies. Thus, the device may be used as a balanced modulator, double balanced mixer, product detector, frequency doubler, and other applications requiring these particular output signal characteristics.

The lower differential amplifier has its emitters connected to the package pins so that an external emitter resistance may be used. Also, external load resistors are employed at the device outputs.

Signal Levels

The upper quad differential amplifier may be operated either in a linear or a saturated mode. The lower differential amplifier is operated in a linear mode for most applications.

For low-level operation at both input ports, the output signal will contain sum and difference frequency components and have an amplitude which is a function of the product of the input signal amplitudes.

For high-level operation at the carrier input port and linear operation at the modulating signal port, the output signal will contain sum and difference frequency components of the modulating signal frequency and the fundamental and odd harmonics of the carrier frequency. The output amplitude will be a constant times the modulating signal amplitude. Any amplitude variations in the carrier signal will not appear in the output.

FIGURE 23 – CIRCUIT SCHEMATIC

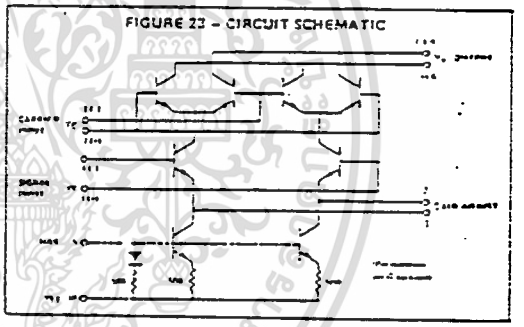
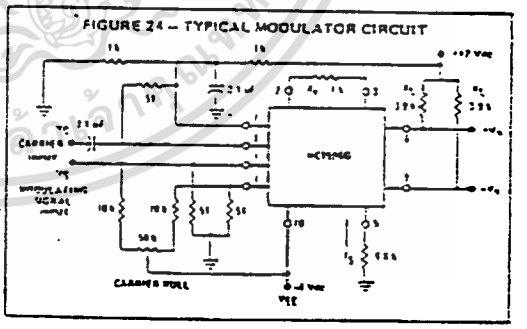


FIGURE 24 – TYPICAL MODULATOR CIRCUIT



NOTE: Pin numbers references pertain to the device when packaged in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic packaged devices refer to the first page of this specification sheet.

OPERATIONS INFORMATION (continued)

The linear signal handling capabilities of a differential amplifier are well defined. With no emitter degeneration the maximum input voltage for linear operation is approximately 25 mV peak. Since the upper differential amplifier has its emitters internally connected, this voltage applies to the carrier input port for all conditions.

Since the lower differential amplifier has provisions for an external emitter resistance, its linear signal handling range may be adjusted by the user. The maximum input voltage for linear operation may be approximated from the following expression:

$$V = \left(\frac{I_5}{2}\right)(R_E) \text{ volts peak.}$$

This expression may be used to compute the minimum value of R_E for a given input voltage amplitude.

FIGURE 25 – TABLE 1
VOLTAGE GAIN AND OUTPUT FREQUENCIES

Carrier Input Signal (V_C)	Approximate Voltage Gain	Output Signal Frequency(s)
Low-level dc	$\frac{R_L V_C}{2[R_E + 2r_e]\left(\frac{KT}{q}\right)}$	f_M
High-level dc	$\frac{R_L}{R_E + 2r_e}$	f_M
Low-level ac	$\frac{R_L V_{C(rms)}}{2\sqrt{2}\left(\frac{KT}{q}\right)[R_E + 2r_e]}$	$f_C \pm f_M$
High-level ac	$\frac{0.637\,R_L}{R_E + 2r_e}$	$f_C \pm f_M$, $3f_C \pm f_M$, $5f_C \pm f_M$, . . .

The gain from the modulating signal input port to the output is the MC1595/MC1496 gain parameter which is most often of interest to the designer. This gain has significance only when the lower differential amplifier is operated in a linear mode, but this includes most applications of the device.

As previously mentioned, the upper quad differential amplifier may be operated either in a linear or a saturated mode. Approximate gain expressions have been developed for the MC1596/MC1496 for a low-level modulating signal input and the following carrier input conditions:

1) Low-level dc
2) High-level dc
3) Low-level ac
4) High-level ac

These gains are summarized in Table 1, along with the frequency components contained in the output signal.

NOTES:

1. Low-level Modulating Signal, V_M, assumed in all cases. V_C = Carrier Input Voltage.
2. When the output signal contains multiple frequencies, the gain expression given is for the output amplitude of each of the two desired outputs. $f_C = f_M$ and $f_C = f_M$.
3. All gain expressions are for a single-ended output. For a differential output connection, multiply each expression by two.
4. R_L = Load resistance.
5. R_E = Emitter resistance between pins 2 and 3.
6. r_e = Transistor dynamic emitter resistance, at +25°C:

$$r_e \approx \frac{26 \text{ mV}}{I_5 \text{ (mA)}}$$

7. K = Boltzmann's Constant, T = temperature in degrees Kelvin, q = the charge on an electron.

$$\frac{KT}{q} \approx 26 \text{ mV at room temperature}$$

APPLICATIONS INFORMATION

Double sideband suppressed carrier modulation is the basic application of the MC1596/MC1496. The suggested circuit for this application is shown on the front page of this data sheet.

In some applications, it may be necessary to operate the MC1596/MC1496 with a single dc supply voltage instead of dual supplies. Figure 26 shows a balanced modulator designed for operation with a single +12 Vdc supply. Performance of this circuit is similar to that of the dual supply modulator.

AM Modulator

The circuit shown in Figure 27 may be used as an amplitude modulator with a minor modification.

All that is required to shift from suppressed carrier to AM operation is to adjust the carrier null potentiometer for the proper amount of carrier insertion in the output signal.

However, the suppressed carrier null circuitry as shown in Figure 27 does not have sufficient adjustment range. Therefore, the modulator may be modified for AM operation by changing two resistor values in the null circuit as shown in Figure 28.

Product Detector

The MC1596/MC1496 makes an excellent SSB product detector (see Figure 29).

This product detector has a sensitivity of 3.0 microvolts and a dynamic range of 90 dB when operating at an intermediate frequency of 9 MHz.

The detector is broadband for the entire high frequency range. For operation at very low intermediate frequencies down to 50 kHz the 0.1 µF capacitors on pins 1 and 8 should be increased to 1.0 µF. Also, the output filter at pin 9 can be tailored to a specific intermediate frequency and audio amplifier input impedance.

As in all applications of the MC1596/MC1496, the emitter resistance between pins 2 and 3 may be increased or decreased to adjust circuit gain, sensitivity, and dynamic range.

This circuit may also be used as an AM detector by introducing carrier signal at the carrier input and an AM signal at the SSB input.

The carrier signal may be derived from the intermediate frequency signal or generated locally. The carrier signal may be introduced with or without modulation, provided its level is sufficiently high to saturate the upper quad differential amplifier. If the carrier signal is modulated, a 300 mV(rms) input level is recommended.

APPLICATIONS INFORMATION (continued)

Doubly Balanced Mixer

The MC1596/MC1496 may be used as a doubly balanced mixer with either broadband or tuned narrow band input and output networks.

The local oscillator signal is introduced at the carrier input port with a recommended amplitude of 100 mV(rms).

Figure 30 shows a mixer with a broadband input and a tuned output.

Frequency Doubler

The MC1596/MC1496 will operate as a frequency doubler by introducing the same frequency at both input ports.

Figures 31 and 32 show a broadband frequency doubler and a tuned output very high frequency (VHF) doubler, respectively.

Phase Detection and FM Detection

The MC1596/MC1496 may be used as a phase detector when equal level input signals are introduced at both inputs. When both inputs are at the same frequency the MC1596/MC1496 will deliver an output which is a function of the phase difference between the two input signals.

An FM detector may be constructed by using the phase detector principle. A tuned circuit is added at one of the inputs to cause the two input signals to vary in phase as a function of frequency. The MC1596/MC1496 will then provide an output which is a function of the input signal frequency.

NOTE: Pin number references pertain to the circuit when packaged in a metal can. To ascertain the corresponding pin numbers for plastic or ceramic packages refer to the last page of this specification sheet.

TYPICAL APPLICATIONS

FIGURE 26 – BALANCED MODULATOR (+12 Vdc SINGLE SUPPLY)

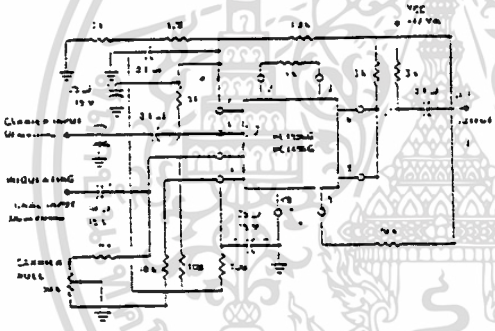


FIGURE 27 – BALANCED MODULATOR-DEMODULATOR

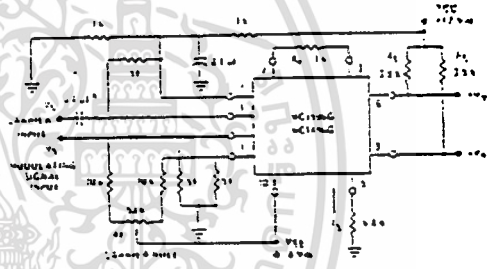


FIGURE 28 – AM MODULATOR CIRCUIT

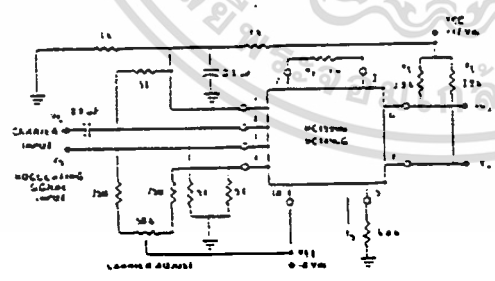


FIGURE 29 – PRODUCT DETECTOR (+12 Vdc SINGLE SUPPLY)