โปรแกรมวิเคราะห์สัญญาณทางระบบสื่อสาร

Software for communication system analysis

นายวีระ เนียมโภคะ

นายวีรศักดิ์ นามวง

นายศราวุธ แก้วกันเนตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ใบรับรองปริญญานิพนธ์

ภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

โปรแกรมวิเคราะห์สัญญาณทางระบบสื่อสาร

(Software for communication system analysis)

โดย

นายวีระ เนียมโภคะ รหัส 34131171

นายวีรศักดิ์ นามวง รหัส 34131172

นายศราวุธ แก้วกันเนตร รหัส 34131173

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

_____ ประธานกรรมการ

(_____)

_____ กรรมการ

(_____)

_____ กรรมการ

(_____)

ชื่อปริญญานิพนธ์	:	โปรแกรมวิเคราะห์สัญญาณทางระบบสื่อสาร
ชื่อผู้เขียน	:	นายวีระ เนียมโภคะ
		นายวีระศักดิ์ นามวง
		นายศราวุธ แก้วกันเนตร
อาจารย์ที่ปรึกษา	:	อาจารย์กฤดากร กล่อมการ
ปริญญานิพนธ์	:	อุตสาหกรรมศาสตร์บัณฑิต (อิเล็กทรอนิกส์)

บทคัดย่อ

ปริญญานิพนธ์แบบนี้ ได้กล่าวถึงในรายละเอียดในเรื่องเกี่ยวกับทฤษฎีเบื้องต้นของระบบสื่อสาร โดยศึกษาสัญญาณการโมดูเลททั้งแบบ AM และ FM เพื่อให้เข้าใจสมการทางคณิตศาสตร์ เพื่อจะได้นำมาใช้ในการเขียนโปรแกรมวิเคราะห์ระบบสื่อสาร

โปรแกรมวิเคราะห์สัญญาณทางระบบสื่อสาร จะใช้การออกแบบโดยใช้ภาษาซีในการเขียนโปรแกรมทั้งนี้เพราะ มีการทำงานที่รวดเร็วเนื่องจากมีโครงร่างเป็นภาษาระดับสูง และภาษาซีมีวิธีการเขียนโปรแกรมเป็นโมดูลจึงสะดวกและคล่องตัวในการใช้โปรแกรม

ในส่วนของโปรแกรม จะเป็นการวิเคราะห์การมอดูเลทสัญญาณ 2 แบบคือ แบบ AM และ FM ซึ่งจะวิเคราะห์สัญญาณทางเวลา และสเปคตรัมของสัญญาณ นอกจากนี้แล้วยังมีส่วนติวเตอร์ ซึ่งเป็นส่วนที่ให้ผู้ใช้โปรแกรมได้เข้าใจถึงรูปสัญญาณการมอดูเลท สามารถเข้าใจการทำงานของโปรแกรมได้ง่ายขึ้น

Project Name	Software for communication system analysis
Student Name	Mr. Weera Neamphoka
	Mr. Weerasak Namvong
	Mr. Sarawoot Kaewkannetra
Advisor	Mr. Kitdakorn Klomkarn
Bachelor Degree	in Industrial Technology (Electronics)
Year	1992

ABSTRACT

This thesis is described about communications system theory , especienly Amplitude Modulation and Frequency Modulation. Which applied to Communication System Analysis Program.

Software for Communications System Analysis is designed by C language. Because of it is easily, flexible and fastly in operating. C language is write in modul form,

This software consisted of 2 part, Amplitude Modulation and Frequency Modulation. It analysised in time domain and frequency domain and the other it has the tutor for communication system understanding of user.

สารบัญ

บทที่ 1. บทนำ

ความเป็นมา

ในปัจจุบัน เทคโนโลยีทางการสื่อสารได้พัฒนาก้าวหน้าไปมาก และยังคงก้าวต่อไป การเรียนวิชาทางวิศวกรรมโทรคมนาคมจึงต้องให้ทันกับเทคโนโลยีใหม่ ๆ ทั้งนี้ผู้เรียนต้องเข้าใจในเรื่องเกี่ยวกับการมอดูเลทชั่น ซึ่งเป็นพื้นฐานของระบบสื่อสารทั่วไป ให้ดีเสียก่อน เพื่อที่จะช่วยให้การศึกษาถึงพื้นฐานนี้เข้าใจได้รวดเร็ว และมองเห็นภาพพจน์ในการบรรยายได้ชัดเจน ทางผู้จัดทำจึงได้มีแนวความคิดที่จะเขียนโปรแกรมในเชิงช่วยสอนในระบบสื่อสาร โดยจะเน้นในด้านการวิเคราะห์การมอดูเลทชั่น

วัตถุประสงค์ของโครงงาน

โปรแกรมวิเคราะห์ระบบสื่อสารนี้เขียนขึ้นมาเพื่อ วิเคราะห์สัญญาณการมอดูเลท ในเชิงสัญญาณทางเวลา และสเปคตรัมของสัญญาณ โดยจะเน้นวิเคราะห์การมอดูเลท 2 แบบ คือ แบบ AM, และแบบ FM นอกจากจะมีส่วนวิเคราะห์แล้ว โปรแกรมนี้ยังได้เพิ่มส่วนติวเตอร์ เพื่อให้ผู้ใช้โปรแกรมสามารถนำโปรแกรมนี้ไปใช้บรรยายในชั้นเรียน วิชาวิศวกรรมโทรคมนาคม จะช่วยให้ผู้เรียนเข้าใจถึงรูปสัญญาณการมอดูเลท

ขั้นตอนและวิธีการดำเนินงาน

การดำเนินงานได้แบ่งขั้นตอนการทำงานเป็น 3 ขั้นตอน ดังนี้ ขั้นแรก เริ่มศึกษาทฤษฎีพื้นฐานระบบสื่อสาร โดยศึกษาถึงสัญญาณการมอดูเลท ทั้งแบบ AM และแบบ FM เพื่อให้ทราบถึงหลักการและวิธีการ และให้เข้าใจระบบในทางสมการคณิตศาสตร์ จะได้นำมาใช้เป็นแนวทางออกแบบโปรแกรมในขั้นถัดไป ขั้นต่อมา ได้ศึกษาถึงหลักการเขียนโปรแกรมโดยใช้ภาษาซี และอังกอลิธึมในการแปลงสัญญาณ จากสัญญาณเชิงเวลาไปเป็นสัญญาณเชิงความถี่ ในขั้นตอนนี้จะใช้เวลาการศึกษามาก ได้มีการทดลองโปรแกรมการแปลงสัญญาณหลายโปรแกรมโดยใช้อังกอลิธึมที่ต่างกัน เพื่อให้ได้โปรแกรมการวิเคราะห์สัญญาณที่เหมาะสม และโปรแกรมที่เขียนขึ้นนี้ได้เลือกใช้อังกอลิธึม ฟาสฟูเรียร์ ในการแปลงสัญญาณ ซึ่งจะมีความเร็วในการโปรเซสสัญญาณกว่า วิธีดีสครีทฟูเรียร์ทรานซ์ฟอร์ม และเหตุผลที่

เลือกใช้ภาษาซีในการเขียนโปรแกรม เพราะเป็นภาษาที่มีความคล่องตัวที่จะประยุกต์เข้ากับงานต่างๆ ได้เป็นอย่างดี และมีความเร็วในการทำงานเร็วกว่าเนื่องมาจากภาษาซีมีโครงร่างเป็นภาษาระดับสูง แต่จะมีการทำงานใกล้เคียงกับภาษาแอสเซมบลีซึ่งมีข้อดีด้านความเร็ว และภาษาซีมีวิธีการเขียนโปรแกรมเป็นโมดูล ที่เรียกว่า โปรแกรมโครงสร้าง จึงสะดวกและคล่องตัวในการเรียกใช้โปรแกรม

ขั้นตอนสุดท้าย เป็นขั้นตอนการลงมือเขียนโปรแกรมและการทดสอบโปรแกรม รวมทั้งการเขียนคู่มือการใช้โปรแกรม ซึ่งได้อธิบายไว้ในบทที่ 4

บทที่ 2 ทฤษฎีเบื้องต้นระบบสื่อสาร

แอมปลิจูดโมดูเลชั่น

แอมปลิจูดโมดูเลชั่นที่ใช้กันอยู่ส่วนมากใช้ในระบบวิทยุกระจายเสียง(broadcasting) การแพร่ภาพของทีวี การรับส่งสัญญาณผ่านคลื่นสั้น(short wave)หรือคลื่นความถี่สูง (high frequency) และในระบบเรดาร์ (radar) แบบง่ายๆ เป็นต้น

ระบบเอเอ็มโดยทั่วไปสามารถแบ่งประเภทตามลักษณะคลื่นได้ดังนี้

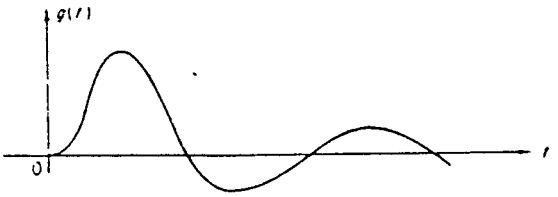
1. ไซด์แบนคู่ (Double Sideband (DSB))
2. เอเอ็มทั่วไป (Conventional AM)
3. ไซด์แบนเดี่ยว Single Sideband (SSB))

โมดูเลชั่นแบบไซด์แบนคู่

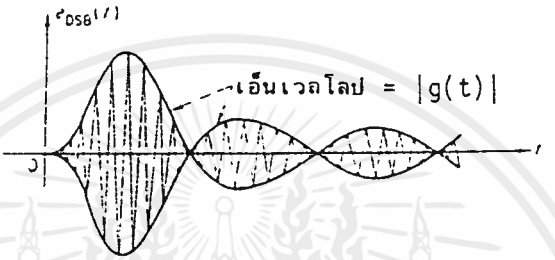
โมดูเลชั่นแบบไซด์แบนคู่ เป็นวิธีการโมดูเลชั่นอย่างตรงไปตรงมา โดยการคูณสัญญาณโมดูเลติ่งกับคลื่นตัวพา ซึ่งเรียกว่า เครื่องโมดูเลทแบบบาลานซ์ (Balance Modulator) โดยเป็นวงจรที่มีสมการสมมาตรกันที่จะลบล้างไม่ให้สัญญาณตัวพาปรากฏในแบนเดียวกับผลลัพธ์

ถ้าให้สัญญาณโมดูเลติ่ง g(t) มีความถี่ต่ำกว่า f_m ($0 < f < f_m$) สัญญาณนี้เรียกว่าสัญญาณเบสแบนด์ (Baseband Signal)เพราะมีความถี่ต่ำถูกจำกัดแถบความถี่อยู่ (Band Limited) สัญญาณโมดูเลติ่งที่ผ่านเครื่องโมดูเลทแบบบาลานซ์ ซึ่งในอุปกรณ์ไม่เชิงเส้น (nonlinear device)จะลบล้างไม่ให้ส่วนของตัวพา Carrier Component) นี้จะมีแถบความถี่อยู่สองข้างของความถี่ของตัวพา จึงเรียกว่า (Double Sideband Suppressed Carrier DSB-SC))

สมมุติให้สัญญาณโมดูเลติ่งเป็น g(t) และตัวพาเป็นไซน์ยูซอยดัลสัญญาณขาออกของเครื่องโมดูเลทแบบบาลานซ์จะได้



(ก) คลื่นโมดูเลติ้ง



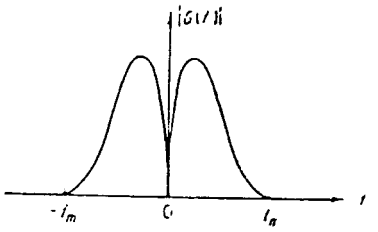
(ข) คลื่นตัวพาที่ถูกโมดูเลทแล้ว

รูปที่ 1 สัญญาณแบสแบนด์และรูปคลื่นแบบไซด์แบนด์คู่

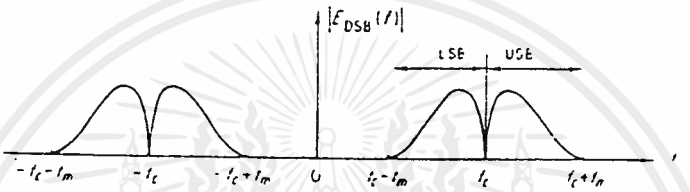
$$e_{DSB}(t) = g(t)\,C(t) = g(t)\cos 2\pi f_c t, \quad f_c >> f_m$$

จะเห็นได้ว่าโมดูเลชั่นไซด์แบนด์คู่ก็คือ การคูณกันระหว่างตัวพากับสัญญาณโมดูเลติ้ง จากรูปแสดงให้เห็นถึงผลคูณของสัญญาณเบสแบนด์กับคลื่นตัวพาแบบไซด์นูซอยดอลตามรูปสัญญาณที่ถูกโมดูเลทจะมีขนาดแอมปลิจูด (Amplitude) เปลี่ยนตามขนาดสัญญาณโมดูเลติ้ง โดยมีความถี่เท่ากับความถี่ของตัวพา จะเห็นว่าโมดูเลชั่นไซด์แบนด์คู่จะเกิดการทับกัน (cross over) ในกรณีโมดูเลชั่นมากเกินไป (over modulation) เมื่อสัญญาณเบสแบนด์ (baseband) g(t) มีส่วนที่เป็นลบ ส่วนที่เกิดการทับกันนี้จะทำให้เกิดการแยกสัญญาณกลับเหมือนเดิมได้ยาก เมื่อเวลาดีโมดูเลททำให้เกิดความเพี้ยนขึ้นได้

ถ้าให้สัญญาณเบสแบนด์ $g(t)$ มีสเปคตรัมความถี่ (frequency spectrum) ดังรูปข้างล่าง



(ก) สเปคตรัมเบสแบนด์



(ข) สเปคตรัมไซด์แบนด์คู่

รูปที่ 2 สเปคตรัมของ double-sideband

ก่อนที่จะมีการโมดูเลชัน สเปคตรัมของสัญญาณเบสแบนด์ จะถูกจำกัดแถบความถี่อยู่ภายใน $|f_m|$ เท่านั้น เมื่อโมดูเลทกับตัวพาแล้วจะถูกย้ายความถี่ไปที่ความถี่ตัวพาโดยขนาดของสเปคตรัมจะลดลงครึ่งหนึ่ง และแถบความถี่ที่อยู่เหนือความถี่ตัวพาเรียกว่า ไซด์แบนด์คู่ (Upper Sideband (USB)) และที่อยู่ต่ำกว่าความถี่ตัวพาเรียกว่า ไซด์แบนด์ต่ำ (Lower Sideband (LSB)) ฉะนั้นจึงเรียกชื่อโมดูเลชันนี้ว่าโมดูเลชั่นไซด์แบนด์คู่ (Double Sideband Modulation)

เนื่องจากสัญญาณ DSB มีส่วนที่ทับกันในดีโมดูเลชันจึงไม่อาจใช้เครื่องตรวจจับเอ็นเวลโลบ(envelope detector) ที่จะตรวจจับเฉพาะเอ็นเวลโลบที่เป็นบวกหรือเป็นลบของสัญญาณเท่านั้น ในกระบวนการดีโมดูเลชันนี้ย้อนวิธีการโมดูเลชันทุกประการ ฉะนั้นถ้าใช้เครื่องตรวจจับผลคูณ (product detector) หรือเครื่องโมดูเลทแบบบาลานซ์ (balanced modulator) จะได้สัญญาณออกมาสองส่วนคือ ส่วนที่เป็นสองเท่าของความถี่ตัวพา และส่วนที่เป็นสัญญาณเบสแบนด์ ถ้าทั้งสองส่วนผ่านวงจรกรองความถี่ต่ำ (low pass filter) ก็จะได้ส่วนที่เป็นสัญญาณเบสแบนด์ตามต้องการ

สมมุติให้สัญญาณที่ได้รับเป็น DSB โดยไม่ทราบเฟส (phase)

$$e_r(t) = g(t)\cos(2\pi f_c t+\phi_c)$$

และให้ local carrier ที่จะนำไป beat เข้ากับสัญญาณที่ได้รับ เกิดความคลาดเคลื่อนจากสัญญาณที่ได้รับทั้งความถี่และเฟสดังนี้

$$c_o(t) = \cos(2\pi f_c t+\phi_o)$$

สัญญาณขาออกของเครื่องตรวจจับผลคูณ จะได้

$$\begin{aligned} c_o(t)e_r(t) &= g(t)\cos(2\pi f_o t+\phi_o)\cos(2\pi f_c t+\phi_c) \\ &= 1/2\ g(t)\{\cos[2\pi(f_c-f_o)t+\phi_c-\phi_o] \\ &\quad +\cos[2\pi(f_c+f_o)t+\phi_c+\phi_o]\} \end{aligned}$$

เมื่อไม่มีความผิดพลาดทางความถี่ หรือเฟสเกิดขึ้นระหว่างตัวพาทั้งสองนั่นคือ $f_c = f_o$ และ $\phi_c = \phi_o$ สัญญาณขาออกของเครื่องตรวจจับผลคูณจะได้

$$c_o(t)e_r(t) = 1/2\ g(t) + 1/2\ g(t)\cos[4\pi f_c t+2\phi_c]$$

สัญญาณขาออกถ้าผ่านเครื่องกรองความถี่ผ่านต่ำ ความถี่ต่ำ $g(t)$ จะผ่านได้ แต่เทอมหลังมีความถี่เป็นสองเท่าของความถี่ตัวพาจะผ่านไปไม่ได้ และถูกกรองออกไป ผลที่ได้ก็จะเหลือแต่ $1/2\ g(t)$

ในกรณีที่ตัวพาทั้งสองเกิดความคลาดเคลื่อนทางเฟสอย่างเดียว นั่นคือ (f_c เท่ากับ f_o และ ϕ_c ไม่เท่ากับ ϕ_o) สัญญาณขาออกขั้นสุดท้ายของ coherent detector จะเท่ากับ

$$1/2\ g(t)\cos(\phi_c - \phi_o)$$

ในกรณีนี้จะได้ข่าวสารเหมือนเดิมทุกประการ แต่ขนาดข่าวสารจะถูกลดลงขึ้นอยู่กับค่า cos ($\phi_c - \phi_o$) ถ้า $\phi_c - \phi_o = 0$ หรือกรณีที่ไม่มีความคลาดเคลื่อนเกิดขึ้นทั้งทางความถี่และเฟสจะได้ข่าวสารเหมือนเดิมโดยไม่ลดขนาด แต่ถ้า $\phi_c - \phi_o = \pi/2$ จะทำให้ไม่สามารถรับสัญญาณ $g(t)$ ได้เลย ฉะนั้นใน coherent detector สิ่งที่สำคัญคือ จะต้องมีวงจรมี automatic frequency/phase controlled loop (AF/PC) ที่ล็อกความถี่พาห์ขาเข้าแล้วสร้าง local carrier ขึ้นมา

แอมปลิจูดโมดูเลชั่นที่ใช้กันทั่วไป

ในระบบเอเอ็ม เอ็นเวลโลปของเอเอ็มจะเปลี่ยนตาม $g(t)$ สัญญาณเบสแบนด์ทุกขณะ และมีรูปร่างเหมือนกับ $g(t)$ คลื่นเอเอ็มนั้นกำเนิดได้จากการบวกสัญญาณ DSB เข้ากับคลื่นพาห์ ผลที่ได้จะเท่ากับ

$$e_{AM}(t) = [1+g(t)] \cos 2\pi f_c t$$

ถ้าเพิ่ม m_a ซึ่งเป็นดัชนีโมดูเลชั่น โดยให้

$$[g(t_{max})] \leq 1, \quad 0 < m_a < 1$$

จะได้ $$e_{AM}(t) = [1 + m_a g(t)] \cos 2\pi f_c t \qquad (1)$$

โดย $g(t)$ จะมีความถี่ $|f| < f_m$ และ $f_m << f_c$ m_a จะเป็นตัววัดเปอร์เซนต์โมดูเลชั่น ซึ่งเป็นเป็นตัวแสดงดีกรีของแอมปลิจูดโมดูเลชั่น

จากรูปที่ 3 แสดงรูปคลื่นของสัญญาณเบสแบนด์ซ้อนอยู่บนตัวพา เอ็นเวลโลปของสัญญาณเอเอ็มนี้มีค่าแตกต่างของสัญญาณ DSB ฉะนั้นจึงต้องการกำลังสูงสุด (peak power) มากกว่าสัญญาณ DSB

สเปคตรัมความถี่ของคลื่นเอเอ็มได้จากฟูเรียร์ทรานฟอร์ม (fourier transform) สมการ (1) ถ้าให้ $G(f)$ เป็นสเปคตรัมความถี่ ของสัญญาณเบสแบนด์ $g(t)$ สเปคตรัมเอเอ็มหาได้ดังนี้

$$e_{AM}(t) <=> E_{AM}(f) = [\ (f)+m_a G(f)].\ 1/2[\ (f-f_c)+\ (f+f_c)\ ]$$

$$= 1/2\ [\ (f-f_c)+\ (f+f_c)\ ]+\ m_a/2[G(f-f_c)+G(f+f_c)\ ]$$

ตัวอย่างของการคำนวณกำลัง (power) ที่ส่วนของตัวพาและไวด์แบนด์ของเอเอ็ม โดยให้สัญญาณเบสแบนด์เป็นไซน์ยูซอยดัล $g(t) = \cos 2\pi f_c t$ สัญญาณเอเอ็มจะได้

$$e_{AM}(t) = A_c(1+m_a \cos 2\pi f_m t)\ \cos 2\pi f_c t$$

$$= A_c \cos 2\pi f_c t+\ m_a A_c/2[\cos 2\pi(f_c+f_m)t+\cos 2\pi(f_c-f_m)t]$$

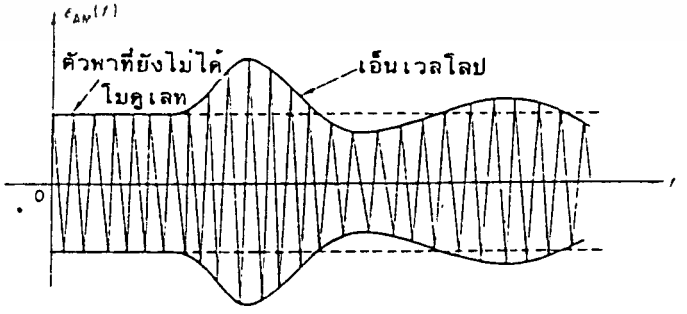
สเปคตรัมความถี่ของสัญญาณเอเอ็มจะเท่ากับ

$$E_{AM}(f) = a_c/2\ [\ (f-f_c)+\ (f+f_c)\ ]$$

$$+m_a A_c/4[\ (f-f_c-f_m)+\ (f+f_c+f_m)$$

$$+\ (f-f_c+f_m)\ +\ \ (f+f_c+f_m)\ ]$$

จะเห็นว่าในระบบเอเอ็มมีส่วนของตัวพา (carrier component) อยู่และมีผลบวกและผลต่างของความถี่ของตัวพาและของสัญญาณโมดูเลติ่งอยู่ที่เรียกว่า USB และ LSB ตามลำดับ



ตัวพาที่ถูกโมดูเลทแล้ว

รูปที่ 3 รูปคลื่นเบสแบนด์และเอเอ็ม

เอเอ็มที่ใช้กันทั่วไป

ในระบบเอเอ็มที่มีสัญญาณเบสแบนด์เป็นคลื่นไซน์ยูซอยดัลความถี่เดียวสัญญาณที่ได้รับจะเป็น

$$e_r(t) = A[1+m_a \cos 2\pi f_m t]\cos 2\pi f_c t + n(t)$$

โดยเสียงรบกวนจะแสดงในรูปของเอ็นเวลโลปและเฟส

$$n(t) = \quad (t) \cos[2\pi f_c t + \theta(t)]$$

ค่า SNR คือค่าเฉลี่ยของสัญญาณที่ได้รับกำลังสอง หรือที่เรียกว่า mean-square value

$$\text{av } e_r^2(t) = 1/T \int_0^T A_c \left[1+m_a \cos 2\pi f_m t\right]^2 \cos^2 2\pi f_c t\,dt + n^2(t)$$

$$= A_c^2/2\ (1 + m_a^2/2) + N$$

ไซด์แบนเดี่ยว (SSB)

ในการดีโมดูเลท SSB สัญญาณและเสียงรบกวนถูกย้ายความถี่จากความถี่วิทยุไปยังความถี่ต่ำ โดยไม่มีการเปลี่ยนค่าใดๆ ค่า SNR จึงไม่เปลี่ยนแปลง เนื่องจาก SSB ใช้แถบความถี่เพียงครึ่งหนึ่งของ DSB และเอเอ็มเท่านั้น เสียงรบกวนในระบบ SSB ก็เท่ากับครึ่งหนึ่งของ DSB และเอเอ็มเท่านั้น แม้ว่า SNR ของ SSB จะน้อยกว่า DSB 2 เท่าก็ตาม ผลลัพธ์ของ SNR ของ SSB จะเท่ากับ SNR ของ DSB และเอเอ็ม

$$(S/N)_{o(SSB)} = (S/N)_{i(SSB)}$$

$$(S/N)_{o(SSB)} = (S/N)o(\text{DSB หรือ AM})$$

สัญญาณเอฟเอ็มและพีเอ็ม

สัญญาณเอฟเอ็ม คือ สัญญาณที่เกิดจากโมดูเลชั่นที่เปลี่ยนความถี่ของตัวพาตามขนาดของสัญญาณโมดูเลติ่ง m(t) โดยมีขนาดของตัวพาคงที่เสมอ ดังนั้น

$$\phi(t) = k_f \int_{-\infty}^{t} m(t)\, dt \qquad \text{FM}$$

ในที่นี้ k_f คือค่าคงที่ของระบบ

ส่วนสัญญาณพีเอ็ม คือสัญญาณที่เกิดจากโมดูเลชั่นที่เปลี่ยนเฟสของตัวพาตามขนาดของสัญญาณโมดูเลติ่ง m(t) โดยมีขนาดของตัวพาคงที่เสมอ ดังนั้น

$$\phi(t) = k_p\, m(t) \qquad \text{PM}$$

ในที่นี้ kp คือค่าคงที่ของการเบี่ยงเบนของเฟส (phase deviation)(หน่วย radian/volt)

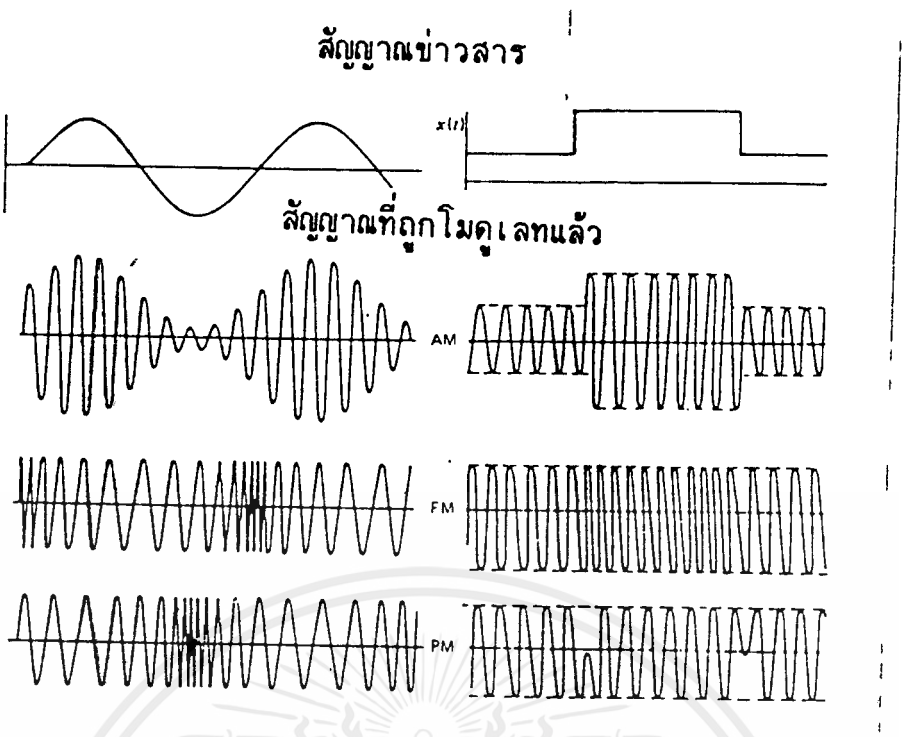
เฟสขณะใดขณะหนึ่งของสัญญาณที่ถูกโมดูเลท e(t) คือ

$$\theta_i(t) = w_c t + \quad (t)$$

และความถี่ขณะใดขณะหนึ่งของสัญญาณที่ถูกโมดูเลทคือ

$$w_i(t) = d\theta_i(t)/dt = w_c + d\ (t)/dt$$

ฟังก์ชั่น $\phi(t)$ และ $d\phi(t)/dt$ คือการเบี่ยงเบนมากที่สุดของเฟสและความถี่ตามลำดับ



รูปที่ 4 รูปคลื่นของเอเอ็ม เอฟเอ็ม และ พีเอ็ม

จากรูปแสดงรูปคลื่นของเอเอ็มและพีเอ็ม จะเห็นได้ว่าขนาดของสัญญาณเอฟเอ็มและพีเอ็มมีค่าคงที่เสมอ ส่วนความถี่และเฟสจะเปลี่ยนตามสัญญาณโมดูเลติ่ง ซึ่งดูคล้ายกับว่า ข้ามไปข้ามมาตามแนวกึ่งกลางของสัญญาณ การเปลี่ยนแปลงความถี่และเฟสของสัญญาณเอฟเอ็มและพีเอ็ม ขึ้นอยู่กับอัตราซีโร่ครอสซิ่ง (zero crossing) สรุปได้ว่าข่าวสารอยู่ที่ซีโรครอสซิ่งของสัญญาณแองเกิลโมดูเลชั่นเมื่อความถี่ของตัวพามีค่าสูง

สเปคตรัมของสัญญาณเอฟเอ็ม

สัญญาณเอฟเอ็มมีรูปคลื่นดังนี้

$$e_{FM}(t) = A_c \cos[2\pi f_c t + k_f \int_{-\infty}^{t} m(t)dt] \qquad (2)$$

ในกรณีของการโมดูเลทด้วยเสียง (tone modulation) สัญญาณโมดูเลติ่ง m(t) จะมีรูปคลื่นดังนี้

$$m(t) = A_m \cos 2\pi f_m t \qquad (3)$$

แทนสมการ (3) ใน สมการ (4) จะได้สัญญาณเอฟเอ็ม

$$e_{FM}(t) = A_c \cos[2\pi f_c t + \beta \sin 2\pi f_m t]$$

บทที่ 3 ฟาสท์ฟูเรียร์ทรานสฟอร์ม Fast Fourier Transform (FFT)

ถ้าหากเรานิยายให้ x(t) มีค่าเท่ากับฟังก์ชั่นบนช่วง $(-t_o, t_o)$ และนิยายให้มันมีคาบ $2t_o$ ณ จุดอื่นๆ จากนั้นเขียนแทนฟังก์ชั่นนี้ด้วยอนุกรมฟูเรียร์แล้วทำการพิจารณาให้ t_o เข้าใกล้อนันต์

$$x(w) = \int x(t) \exp(-jwt)\, dt$$

สมการข้างบนเรียกว่า ฟูเรียร์อินทิกรัล หรือการแปลงฟูเรียร์
สำหรับการแปลงฟาสฟูเรียร์แบบเต็มหน่วย (Discrete Fourier transform) ที่ลำดับ x(m) ที่ยาว N จุด สามารถนิยายได้ดังนี้

$$X(k) = \Sigma\, x(m).W^{mk}$$

สมการข้างบนเขียนให้อยู่ในรูปสมการเมตริกซ์ได้ดังนี้

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

หลักสำคัญของการแปลงฟาสฟูเรียร์ ที่ลดจำนวนครั้งในการคูณจำนวนเชิงซ้อนได้โดยอาศัย คุณสมบัติของ ภาวะเป็นคาบ (periodicity) ของจำนวนเชิงซ้อน W คือ

$$W^{mk} = W^{[k \bmod (n)]}$$

ซึ่ง [mk mod(N)] หมายถึง ส่วนที่เหลือหลังจากการหารพจน์ mk ด้วย N โดยอาศัยคุณสมบัติภาวะเป็นคาบนี้ทำให้สมการเมตริกซ์ ได้เป็น

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

ขั้นตอนวิธี หรือ ลำดับการ ในการคำนวณ DFT ให้เร็วที่มีชื่อเรียกว่า การแปลงฟาสท์ฟูเรียร์ ได้มีการพัฒนาวิธีการต่างๆ กันอย่างมากนั้น เป็นผลมาจากผลงานของ J.W. Cooley กับ J.W Tukey ในปี ค.ศ. 1965 ทำให้เกิดการพัฒนาวิธีการขึ้นมาอีกหลายวิธี FFT จะทำให้การคำนวณ DFT ใช้การคูณจำนวนเชิงซ้อนเพียง N log N ครั้งเท่านั้น หรือ จำนวณครั้งในการคูณตัวเลขลดลงถึง $N/(\log_2 N)$ เท่า ถึงแม้ FFT จะมีชื่อว่าการแปลงฟาสท์ฟูเรียร์ แต่ตัว FFT เองนั้นไม่ใช่การแปลงฟูเรียร์ แท้จริงแล้วเป็นเพียง วิธีการ หรือ ลำดับการในการคำนวณที่ช่วยให้การคำนวณ DFT ซึ่งเป็นการแปลงฟูเรียร์ได้เร็วขึ้น

หลักการเบื้องต้นของ FFT

การแปลงฟูเรียร์แบบเต็มหน่วยสำหรับ ลำดับ x(m) ที่ยาว N จุด สามารถนิยามดังนี้

$$x(k) = \sum_{m=1}^{N-1} [\, x(m).W^{mk} \,] \qquad (1)$$

สมการ (1) สามารถเขียนให้อยู่ในรูปของสมการเมตริกซ์ได้คือ

$$x(k) = \{A\}.\{x\} \qquad (2a)$$

ตัวอย่างเช่น พิจารณากรณี N = 4 สามารถเขียนแยกออกได้เป็น

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad (2b)$$

จากคุณสมบัติภาวะเป็นคาบนี้เองทำให้เราสามารถแยกตัวประกอบของเมตริกซ์ {A} ออกเป็นเมตริกซ์ย่อยหลายเมตริกซ์คูณกัน และ สมาชิกภายในเมตริกซ์ย่อยให้มีค่าเป็นศูนย์มากที่สุด วิธีการแยกตัวประกอบนี้จะไม่กระทำโดยตรงจาก {A} แต่จะมีการสลับตำแหน่ง หรือจัดกลุ่มของเมตริกซ์ (อาจเป็นการสลับสมาชิกตามแนวตั้ง

หรือแนวนอน อย่างใดอย่างหนึ่ง และการสลับทั้งสองแบบจะให้ผลต่างกัน) ด้วยวิธีการของ การผันกลับบิต (bit reversed) และ เมตริกซ์หลังจัดการสลับแถวแล้วจะนำมาแยกตัวประกอบอีกครั้งหนึ่ง

ในที่นี้จะขออธิบายวิธีการแยกตัวประกอบด้วยวิธีการเขียนแทน ดรรชนี k และ m จะมีค่าได้เพียง 0,1,2 และ 3 เท่านั้น เพราะฉะนั้นสามารถแทน ตัวเลข 4 ฐานสิบ ได้ด้วย ตัวเลขฐานสอง สองหลัก คือ

$$k = (k_1,k_0) \quad , \quad m = (m_1,m_0) \tag{3}$$

โดยที่ k_1, k_0, m_1 และ m_0 เป็นเลขโดที่มีค่าได้แค่ 0 หรือ 1 ดรรชนีในสมการที่ (3) สามารถเขียนใหม่เป็น

$$k = 2k_1 + k_0 \quad , \quad m = 2m_1 + m_0 \tag{4}$$

เมื่อนำค่าดรรชนี k และ m นี้ไปแทนลงในสมการที่ (1) ทำให้ได้

$$X(k_1,k_0) = \sum_{m=1}^{N-1} [\, x(m_1,m_0).W^{(2k_1,k_0)(2m_1,m_0)}] \tag{5}$$

จากคุณสมบัติภาวะเป็นคาบของ W ทำให้ได้ความสัมพันธ์

$$\begin{aligned} W^{(2m_1+m_0)\cdot(2k_1+k_0)} &= W^{(2k_1+k_0)2m_1} \cdot W^{(2k_1+k_0)m_0} \\ &= W^{4m_1k_1} \cdot W^{2m_1k_0} \cdot W^{(2k_1+k_0)m_0} \\ &= W^{2m_1k_0} \cdot W^{(2k_1+k_0)m_0} \end{aligned}$$

โดยที่ $W^{4m_1k_1} = 1$ เพราะฉะนั้นเขียนสมการ (5) ได้ใหม่เป็น

$$X(k_1,k_0) = \sum_{m=1}^{N-1} [\, x(m_1,m_0).W^{(2m_1k_0)} \cdot W^{(2k_1+k_0)m_0} \,] \tag{6}$$

โดยการสมมุติให้ ตัวแปร $x_1(k_0,m_0)$ เป็นการคำนวณระหว่างกลาง ผลจาก (6) อาจจะเขียนเป็นสมการเมตริกซ์ได้คือ

$$\begin{matrix}(k_1,k_0)\\ \\ \end{matrix} \qquad\qquad \begin{matrix}(m_1,m_0)\\ \\ \end{matrix}$$

$$\begin{bmatrix} X(0,0)\\ X(0,1)\\ X(1,0)\\ X(1,1)\end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0\\ 1 & W^2 & 0 & 0\\ 0 & 0 & 1 & W^1\\ 0 & 0 & 1 & W^3\end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0\\ 0 & 1 & 0 & W^0\\ 1 & 0 & W^2 & 0\\ 0 & 1 & 0 & W^2\end{bmatrix} \begin{bmatrix} x(0,0)\\ x(0,1)\\ x(1,0)\\ x(1,1)\end{bmatrix} \qquad (7)$$

โดยที่ผลการคำนวณระหว่างกลาง และ ผลลัพธ์สามารถหาได้ตามลำดับ

$$\begin{matrix}(k_1,m_0)\\ \\ \end{matrix} \qquad\qquad \begin{matrix}(m_0,m_0)\\ \\ \end{matrix}$$

$$\begin{bmatrix} x_1(0,0)\\ x_1(0,1)\\ x_1(1,0)\\ x_1(1,1)\end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0\\ 0 & 1 & 0 & W^0\\ 1 & 0 & W^2 & 0\\ 0 & 1 & 0 & W^2\end{bmatrix} \begin{bmatrix} x(0,0)\\ x(0,1)\\ x(1,0)\\ x(1,1)\end{bmatrix} \qquad (8)$$

และค่า DFT ของลำดับสัญญาณเป็น

$$\begin{matrix}(k_1,k_0)\\ \\ \end{matrix} \qquad\qquad \begin{matrix}(k_0,m_0)\\ \\ \end{matrix}$$

$$\begin{bmatrix} x(0,0)\\ x(0,1)\\ x(1,0)\\ x(1,1)\end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0\\ 0 & W^2 & 0 & 0\\ 0 & 0 & 1 & W^1\\ 0 & 0 & 1 & W^3\end{bmatrix} \begin{bmatrix} x_1(0,0)\\ x_1(0,1)\\ x_1(1,0)\\ x_1(1,1)\end{bmatrix} \qquad (9)$$

ผลที่ได้จากสมการที่ (7) ซึ่งเป็นสมการแยกตัวประกอบของ {A} นี้จะเห็นว่าสมาชิกตามแนวนอนของเมตริกซ์ที่ได้จากการแยกตัวประกอบนั้น มีเพียง 2 ตัว เท่านั้นที่ไม่เป็นศูนย์ และในสองตัวนี้มีสมาชิกตัวหนึ่งมีค่าเป็นหนึ่งเสมอ ส่วนอีกตัวหนึ่งนั้นก็จะเป็นจำนวณเชิงซ้อน แต่ถ้าพิจารณาการคูณเมตริกซ์ย่อย แต่ละเมตริกซ์ของ (8) จะเห็นว่าต้องการการคูณเชิงซ้อนเพียง N = 4 ครั้ง โดยที่ N อยู่ในรูป

ของสองยกกำลังใด ๆ ($N = 2^r$) และถ้าลองเขียนสมการ (6) สำหรับกรณีนี้ ดูเห็นว่าสามารถเขียนแยกเป็นสมการย่อยได้ $\log_2 N = \log_2 4 = 2$ สมการ คือ (7) และ (8) เพราะฉะนั้นด้วยวิธีการนี้จำนวนครั้งของการคูณจำนวนเชิงซ้อนจึงมีเท่ากับ $N \log_2 N = 8$ ครั้ง และบวกจำนวนเชิงซ้อน 8 ครั้ง ซึ่งจำนวนนี้น้อยกว่ากรณีคำนวณโดยตรงซึ่งใช้การคูณจำนวนเชิงซ้อน 16 ครั้ง และบวกจำนวนเชิงซ้อน 12 ครั้ง มาถึงตอนนี้เห็นได้ว่า ในกรณีทั่วไป ถ้าหากเรามีกรรมวิธีในการแยกการคำนวณ DFT ให้เป็นเมตริกซ์ย่อยดังเช่นสมการ (7) ได้ ก็จะทำให้จำนวนครั้งในการคูณจำนวนเชิงซ้อนน้อยลง

ความเป็นจริงแล้วจำนวนครั้งในการคูณจำนวนเชิงซ้อน อาจลดทอนลงไปได้อีก มาดูการคำนวณสัญญาณระหว่างกลาง $x_1(0,0)$ และ $x_1(0,1)$ ซึ่ง

$$x_1(0,0) = x(0,0) + W^0.x(1,0) = x(0,0) + W^0.x(1,0)$$

และ $$x_1(1,0) = x(0,0) + W^2.x(1,0) = x(0,0) - W^0.x(1,0) \quad (10)$$

ผลจากคุณสมบัติของจำนวนเชิงซ้อน $W^2 = -W^0$ ทำให้การคำนวณ $x_1(0,0)$ และ $x_1(0,1)$ ต้องการการคูณจำนวนเชิงซ้อนเพียงครั้งเดียวเท่านั้น ซึ่งทำได้โดยการคำนวณพจน์ $W^0.x(1,0)$ ก่อนแล้วนำไปบวกและลบกับพจน์ $x(0,0)$ เพื่อให้ได้ลำดับ $x_1(0,0)$ และ $x_1(1,0)$ ตามลำดับ อย่างไรก็ตามสำหรับกรณีทั่ว ๆ ไปอาจกล่าวได้ว่า จำนวนครั้งในกรณีการคำนวณ DFT ขนาด N จุด โดยใช้ FFT ใช้การคูณจำนวนเชิงซ้อนเพียง $N \log_2 N$ ครั้ง

ลำดับการคำนวณของ FFT ตามสมการ (6),(7) เป็นผลงานที่เสนอโดย Cooley และ Tukey ต่อมาได้มีผู้เสนอการคำนวณ FFT แบบอื่นขึ้นมา ผลงานเหล่านี้หลักการใหญ่เหมือนกัน ต่างกันเพียงวิธีการในรายละเอียดย่อยเท่านั้น รูปข้างล่าง เป็น กราฟการไหลสัญญาณ (signal flow graph) ที่เขียนวิธีการคำนวณของ FFT ตามสมการ (8),(9) โดยที่หัวลูกศรชี้ทิศทางการคำนวณ ส่วนตัวอักษรกำกับเป็นตัวคูณค่าของสัญญาณที่ต้นทางของลูกศรนั้น และที่ ปม (Node) เป็นการรวมหรือบวกกันของสัญญาณ ส่วน $x_1(k_0,m_0)$ แทนลำดับการคำนวณระหว่างกลาง และ $X(k_1,k_0)$ เป็นค่า DFT ของลำดับสัญญาณ

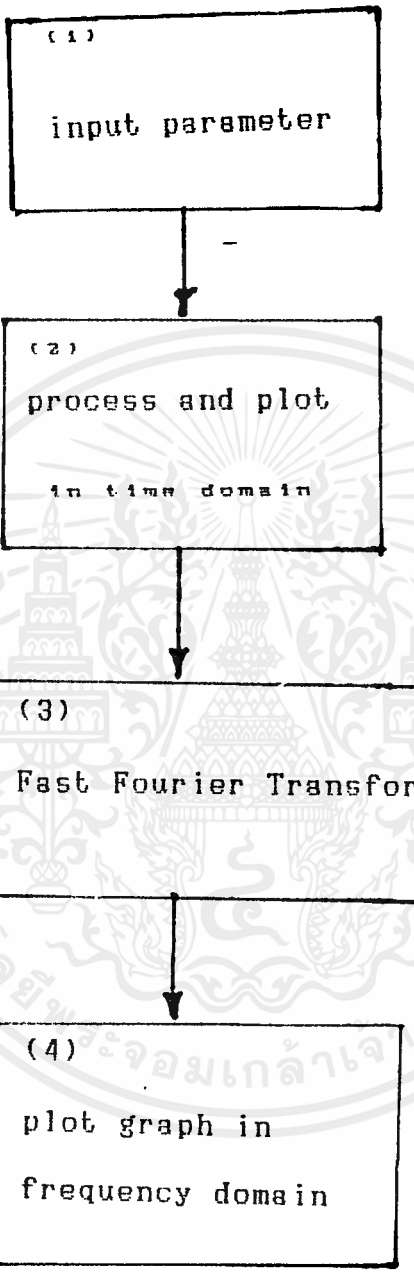
บทที่ 4. การใช้โปรแกรม

เมื่อเรียกใช้โปรแกรมครั้งแรก จะมีเมนูให้เลือก TUTOR, ANALYSIS, หรือ EXIT ในส่วนของ TUTOR จะแสดงตารางที่น่าสมใจทางด้านระบบสื่อสารต่าง ๆ สำหรับส่วนของ ANALYSIS เป็นการวิเคราะห์สัญญาณที่ได้จากวิธีการต่าง ๆ ทางระบบ เช่น สัญญาณจากการมอดูเลทแบบ AM และแบบ FM สำหรับในส่วนของ ANALYSIS นี้จะประกอบไปด้วยเมนูย่อยอีก ดังนี้

F1 :	< make >	สร้างสัญญาณซายนูซอยดอล ที่กำหนดความถี่เองได้
F2 :	< mik >	ทำการมิกซิ่งสัญญาณสองสัญญาณ
F3 :	< mod >	ทำการมอดูเลทสัญญาณเบสแบนด์กับแคเรียร์
F4 :	< plot >	ทำการพล็อตสัญญาณแสดงออกหน้าจอ
F5 :	< AM >	แสดงการมอดูเลทแบบ AM
F6 :	< FM >	แสดงการมอดูเลทแบบ FM
F7 :	<return>	กลับไปยังเมนูหลัก

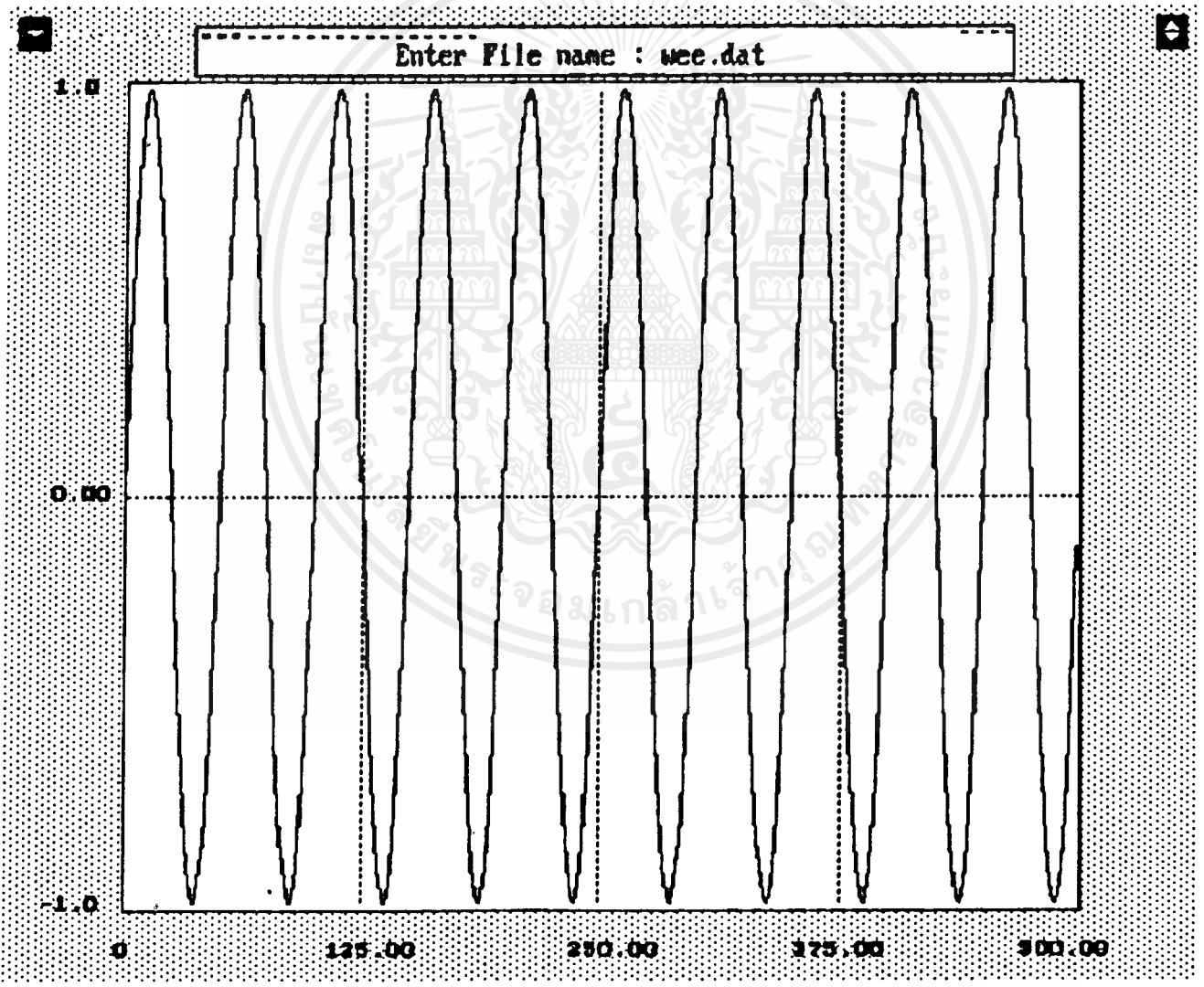
การเลือกใช้เมนูในส่วน ANALYSIS โดยการกดคีย์ F1 - F7 รายละเอียดการใช้งานของแต่ละฟังก์ชั่นจะได้อธิบายในหน้าต่อไป

FLOW CHART แสดงการทำงานของโปรแกรม ในส่วนการวิเคราะห์สัญญาณ

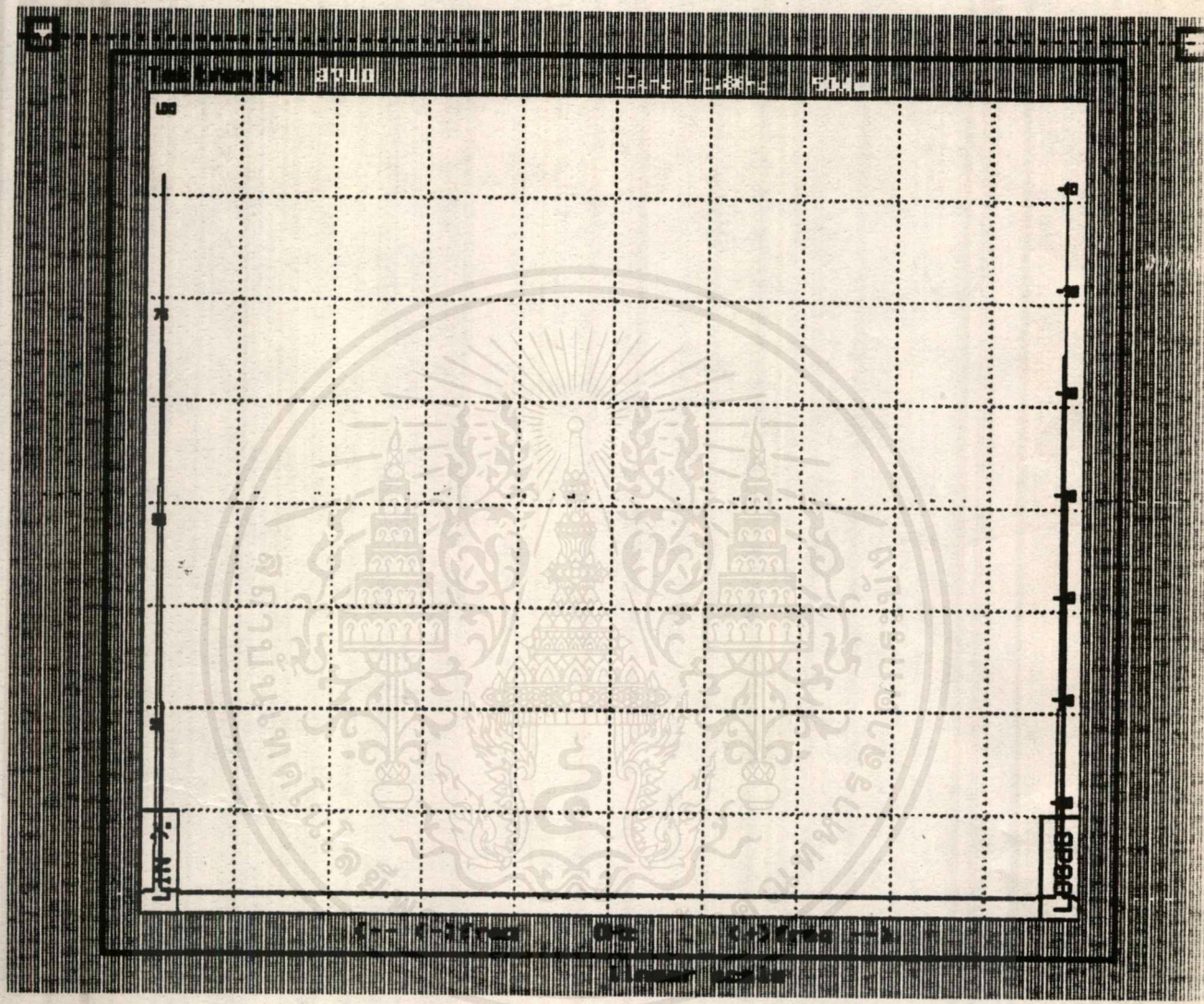


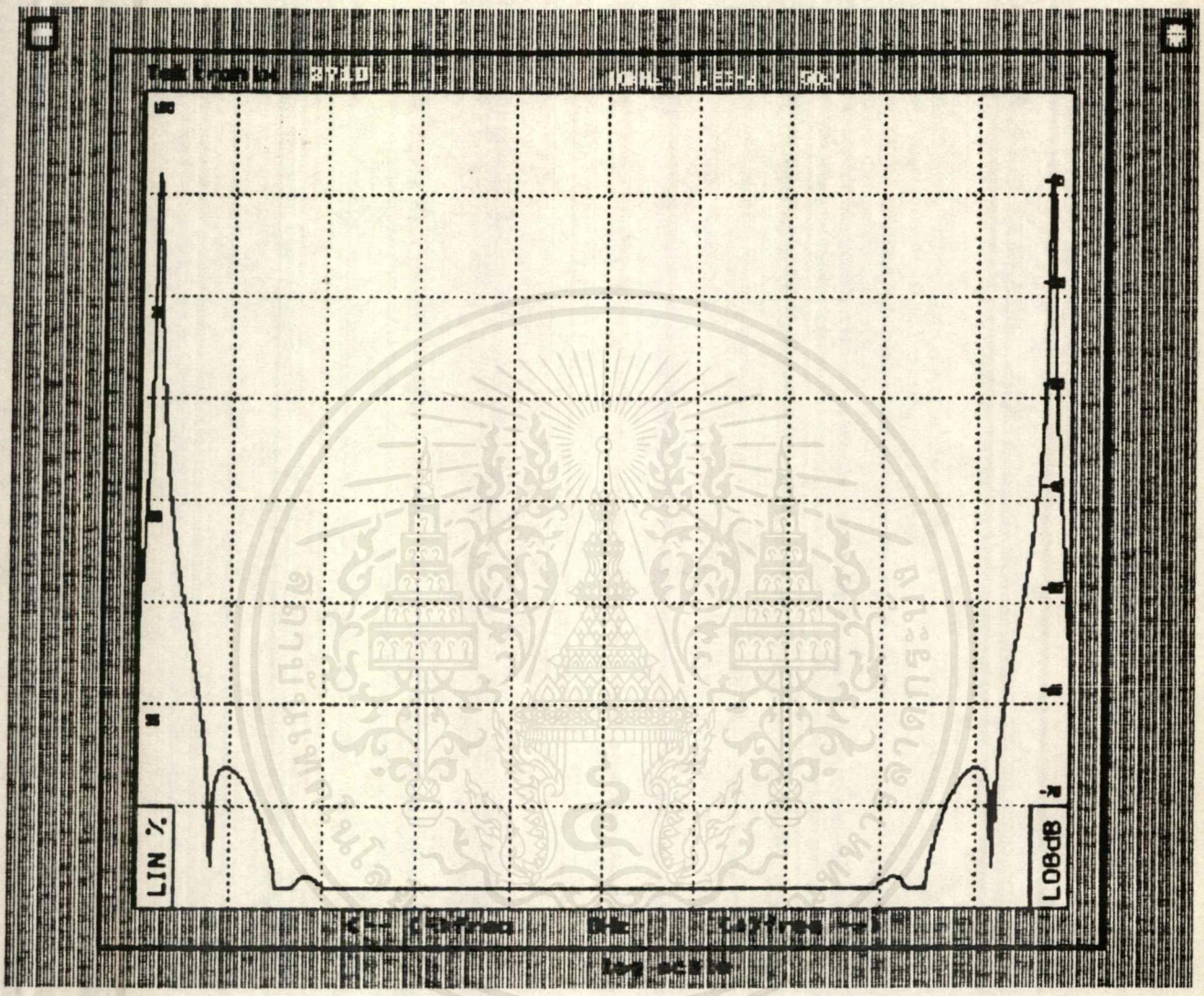
F1 - MAKE

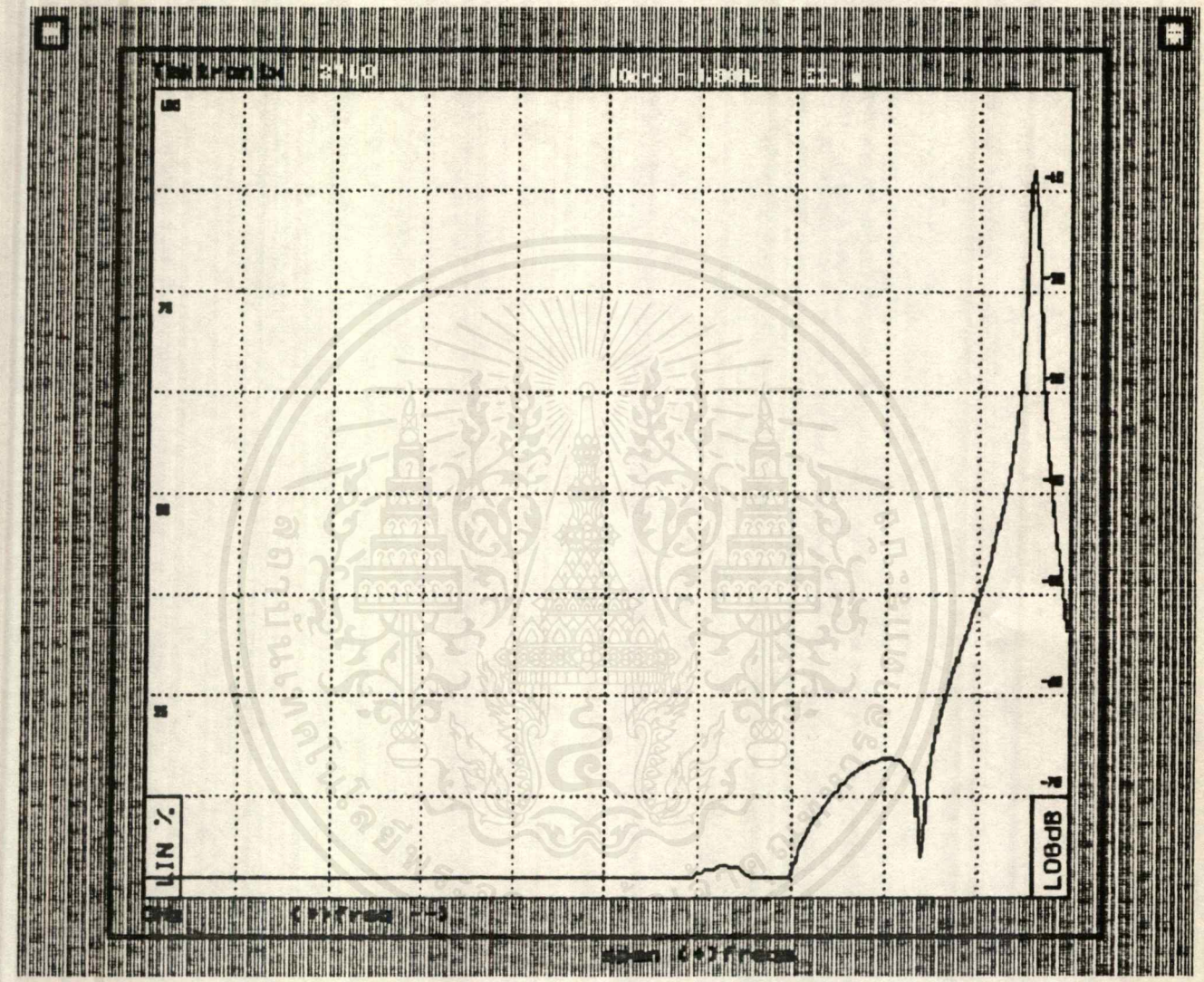
```
Enter number of samples to generate  [ 2...10000 ]    500
Enter number of frequency in sum     [ 1...10 ]       1
Enter frequency #0   [ 0...0.5 ]                      0.02

Enter file name :                                     WEE.DAT
```



เมื่อต้องการดู spectrum ของสัญญาณ โดยการกด **ENTER**





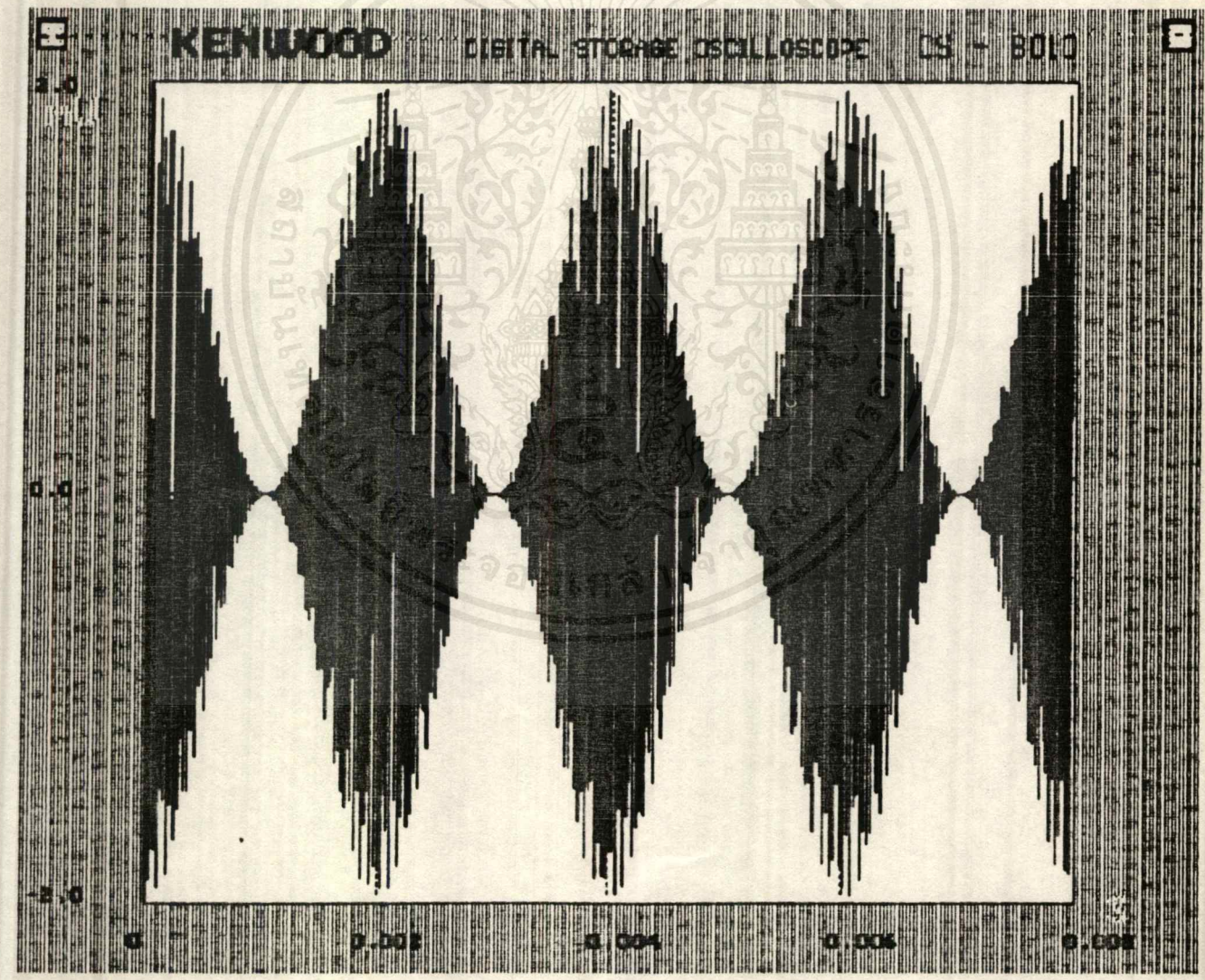


F4 - AM

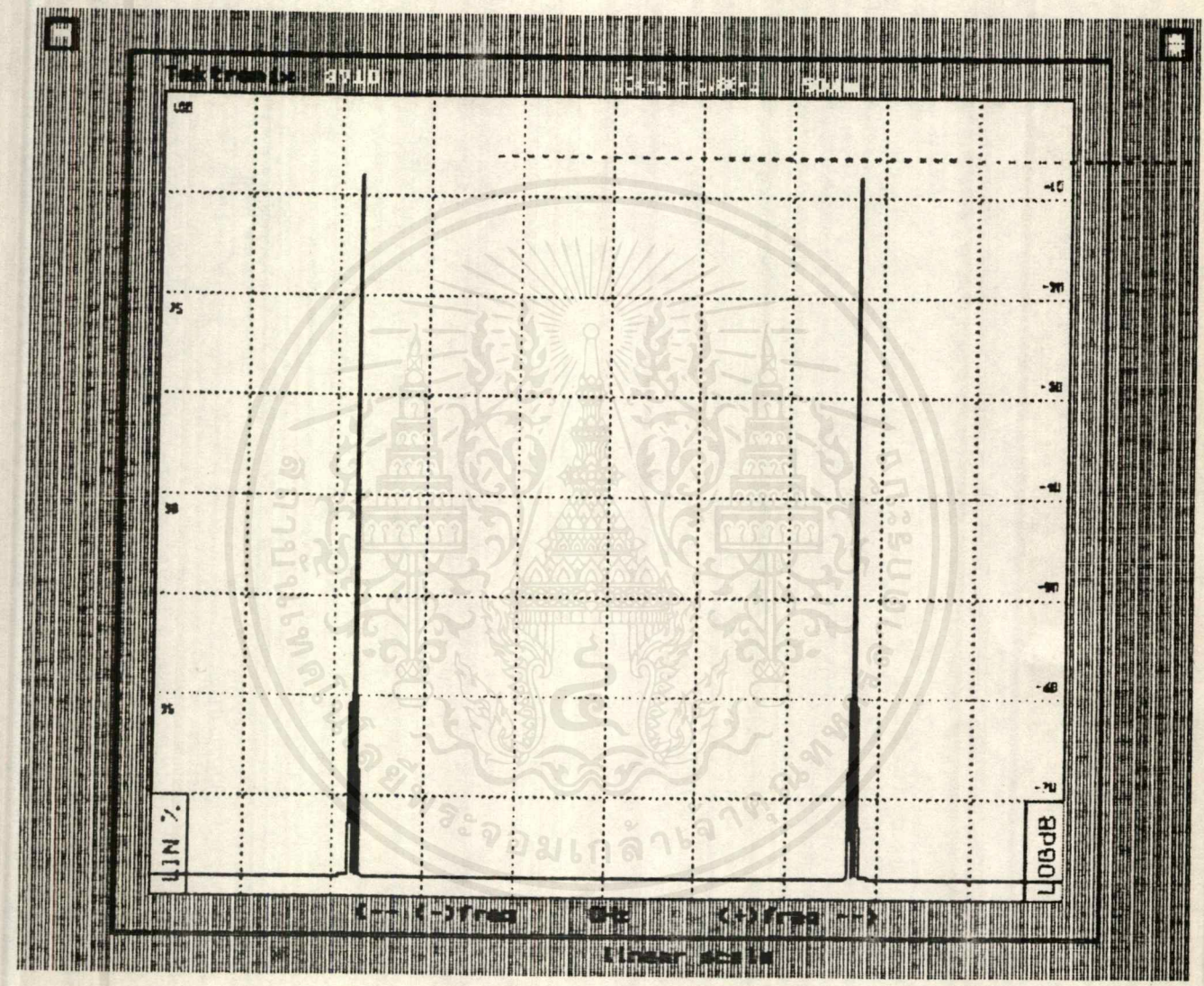
การมอดูเลทสัญญาณแบบ AM แสดงเป็นสมการคณิตศาสตร์ได้ข้างล่าง

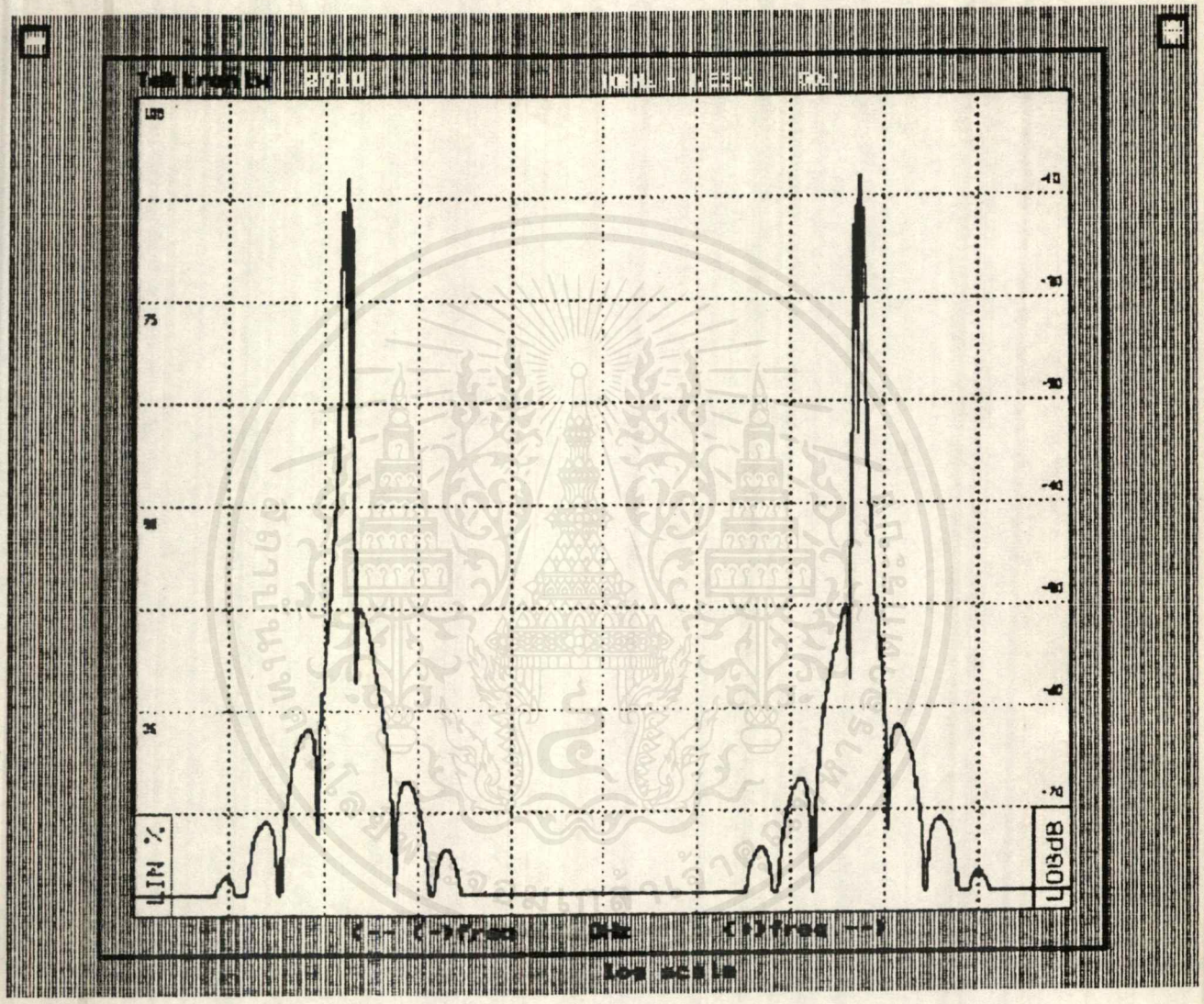
$$Re\{AM\} = A\,(1+m\cos(w_{\_}))\cos(w_{c})$$

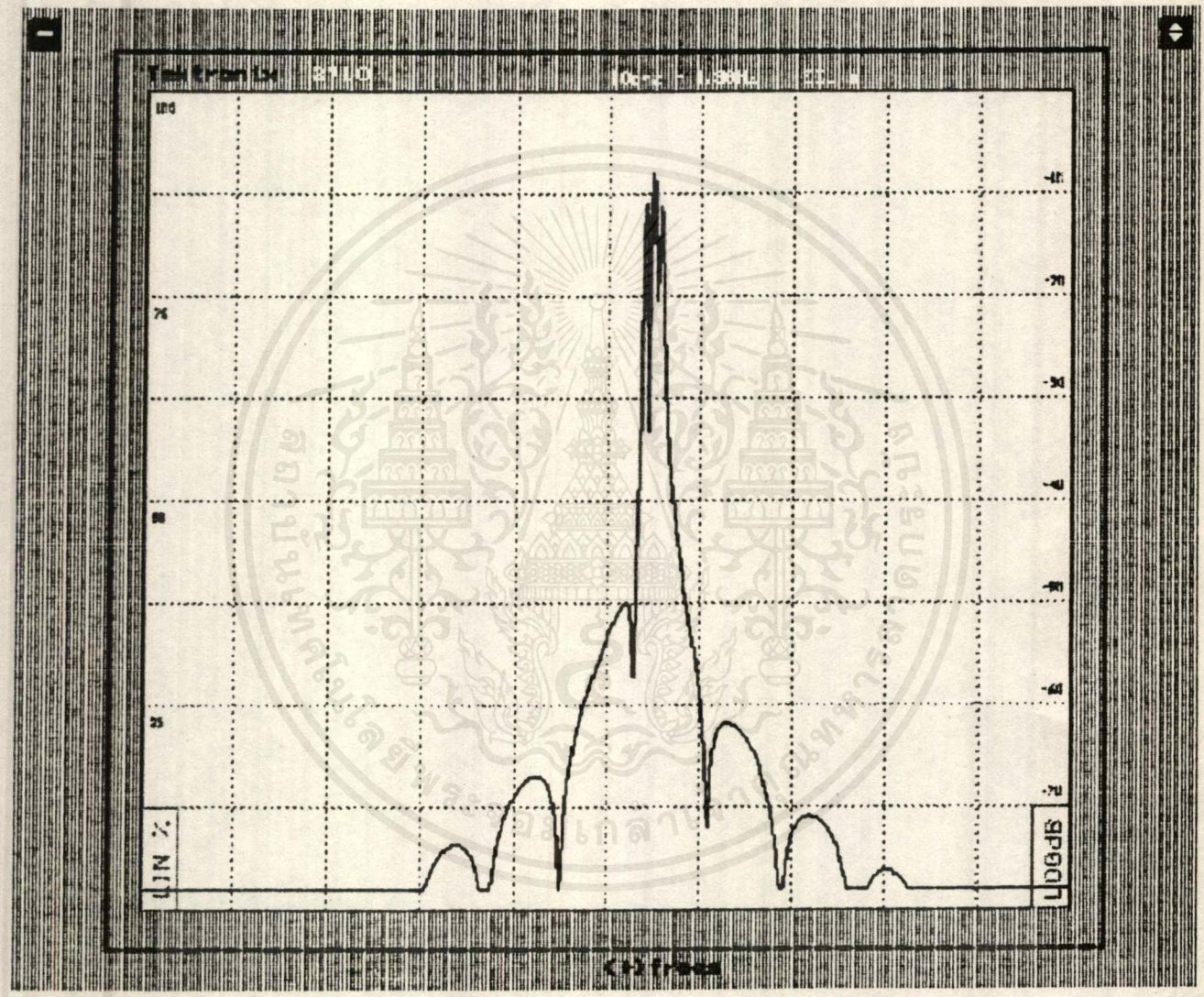
Enter number of amplitude of signal	[1...100] mV	2
Enter PERCENTAGE of modulation	[1...150] %	100
Enter input frequency baseband	[1...15000] Hz	500
Enter input frequency carrier	[540...1600] kHz	600



เมื่อต้องการดูspectrumของสัญาณโดยการกด **ENTER** โปรแกรมจะบอกใช้ผู้ใช้คอยประมาณ 15 วินาที ด้วยข้อความ Please wait! 15 second







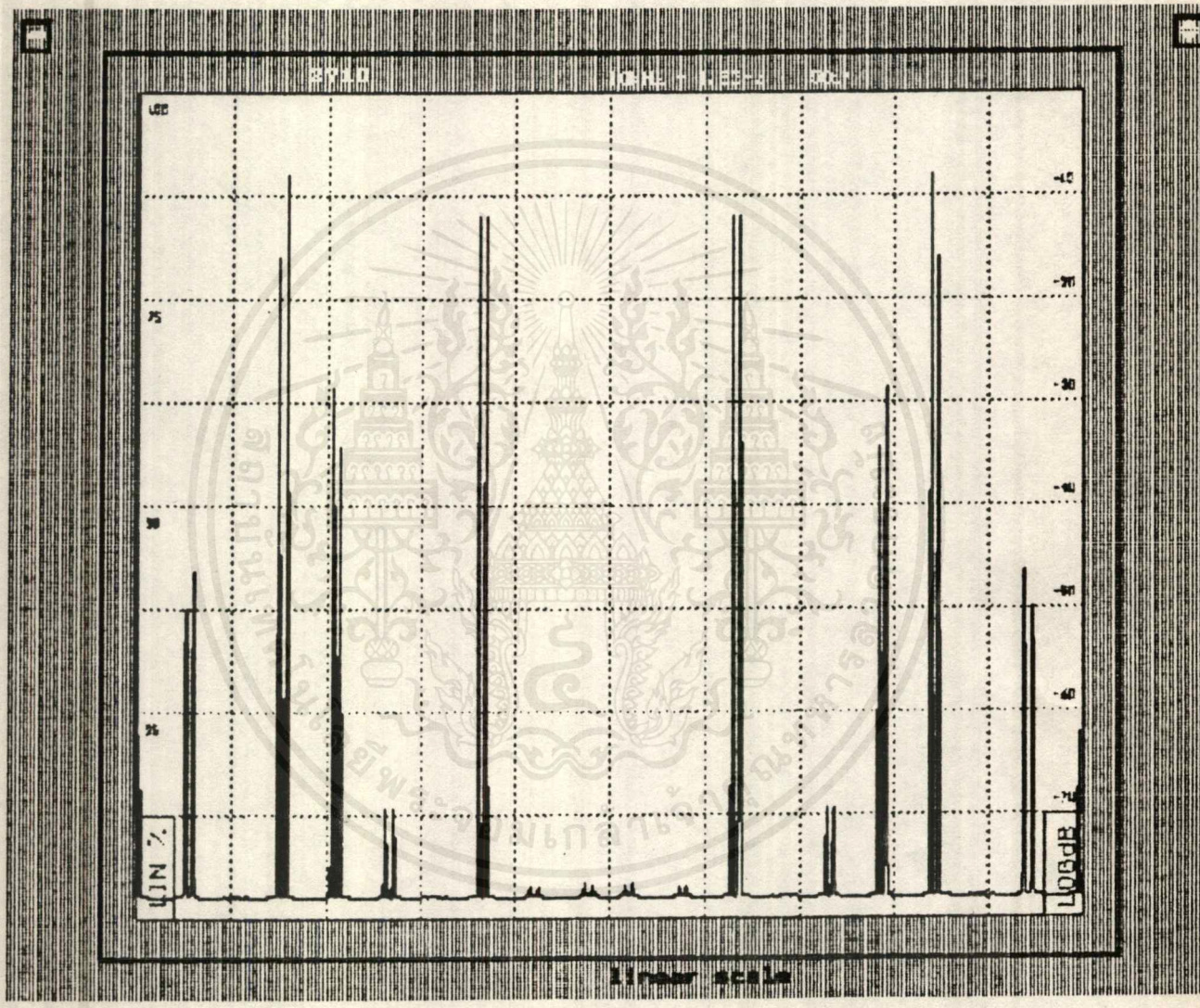
F6 - FM

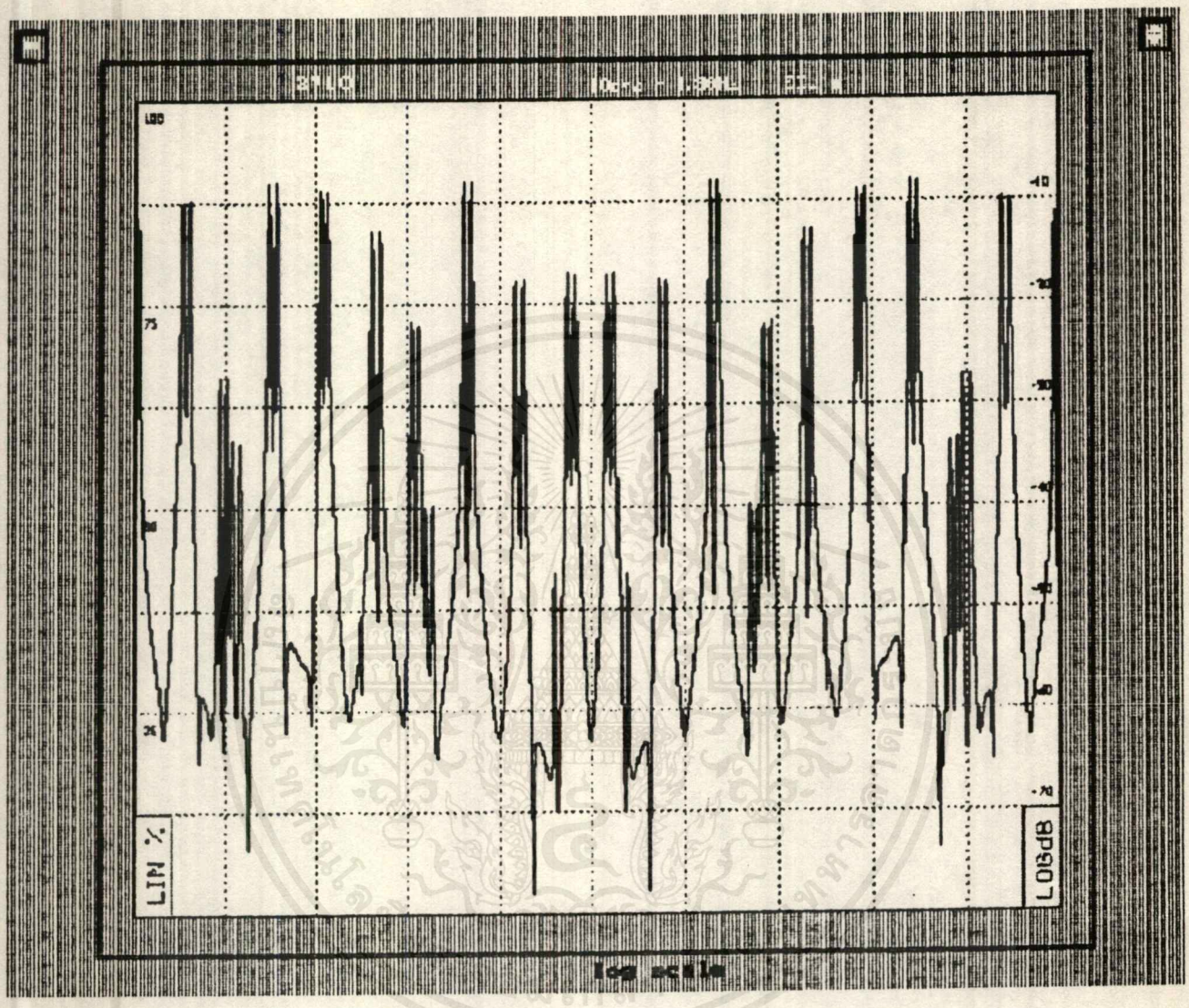
โปรแกรมนี้สามารถทำการมอดูเลทแบบ FM โดย baseband จะเป็นสัญญาณซายน์เวฟ ซึ่งด้วยสมการการมอดูเลทได้ดังข้างล่าง

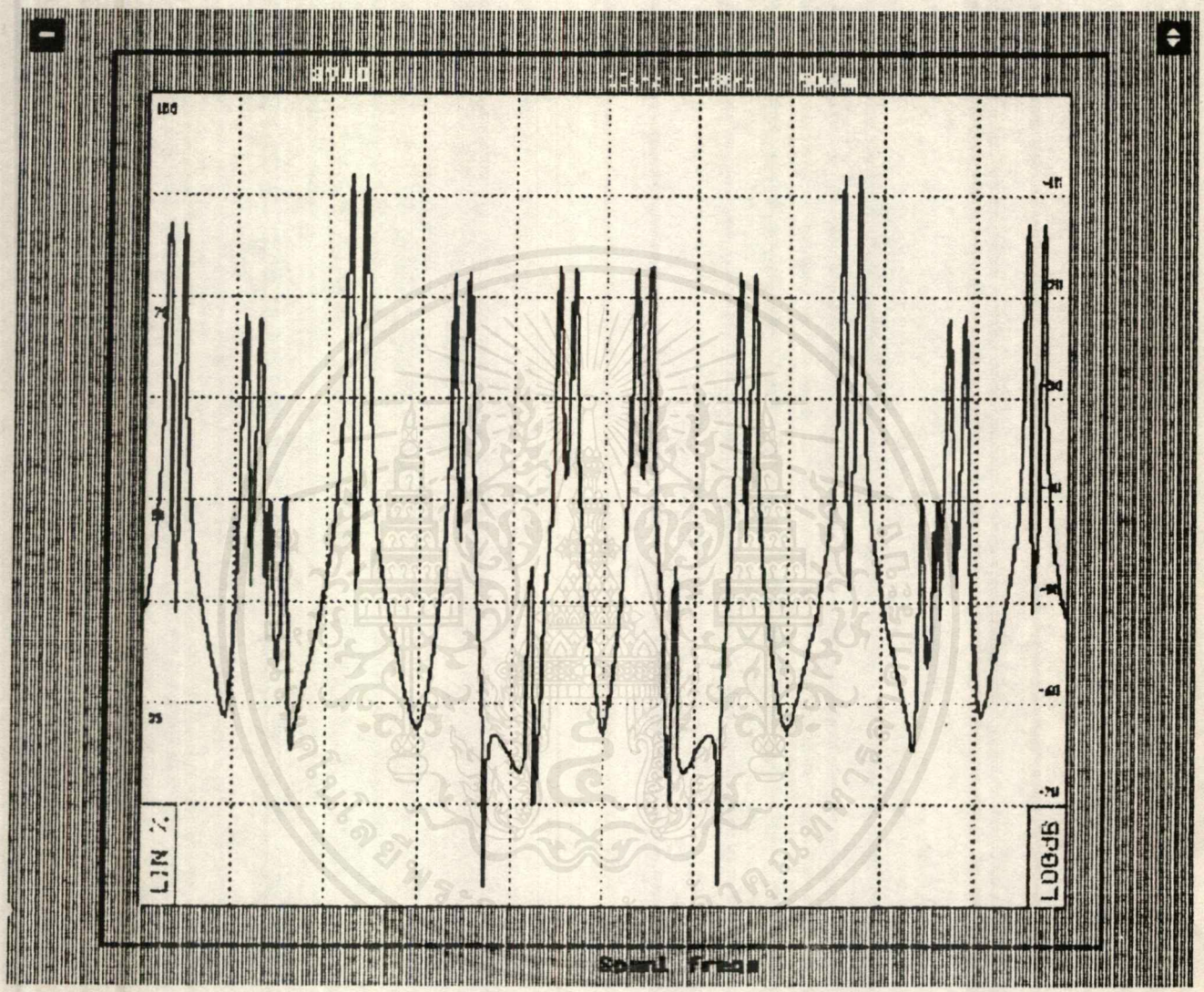
$$Re\{FM\} = A*\cos(W_c + B*\sin(W_m))$$

Enter number of Amplitde of signal	[1...100] mV	**100**
Enter modulation index	[β= 0...12.1]	**5**
Enter carrier frequency	[88...108] MHz	**100**
Enter baseband frequency	[1...15000] Hz	**10000**

เมื่อต้องการดูspectrumของสัญญาณโดยการกด **ENTER** โปรแกรมจะบอกใช้ผู้ใช้คอยประมาณ 15 วินาที ด้วยข้อความ Please wait! 15 second







บทที่ 5. สรุปผลการดำเนินงาน

การวิเคราะห์สัญญาณการมอดูเลททั้งแบบ AM และ FM จะได้เฉพาะสัญญาณซายน์เวฟ

โปรแกรมนี้สามารถแสดงสัญญาณได้ทั้งเชิงเวลา และเชิงความถี่ เนื่องจากโปรแกรมนี้เกี่ยวข้องกับการจัดเก็บและเรียกใช้ไฟล์ข้อมูลสัญญาณ ผู้ใช้จะต้องรู้มาก่อนว่าไฟล์สัญญาณที่จะนำมาวิเคราะห์นั้นมีชื่อว่าอะไร และเก็บไว้ที่ Drive ใด ถ้าหากผู้ใช้เรียกไฟล์ผิดหรือไม่มีไฟล์นั้นอยู่ โปรแกรมจะเกิดข้อผิดพลาด โดยมีข้อความแสดงว่า Enter opening (null) in open_read หลังจากนั้นโปรแกรมจะไม่ทำงานอื่นอีกเลย จึงเป็นการไม่สะดวกเมื่อผู้ใช้เรียกไฟล์ผิด และการเรียกไฟล์นั้นไม่สามารถเรียกในลักษณะ *.* ได้ จึงไม่สามารถเลือกไฟล์ที่มีอยู่ได้

ปัญหาข้างบนนี้เนื่องมาจาก ช่วงเวลาในการเขียนโปรแกรมนั้นมีจำกัด การเขียนโปรแกรม จะใช้เวลาส่วนมากในการทดสอบและแก้ไขอังกอลิธึมของการวิเคราะห์สัญญาณ ซึ่งเป็นส่วนสำคัญที่สุดตามจุดประสงค์ของโครงงาน ดังนั้นโปรแกรมจึงขาดความสะดวกในการใช้ไปบ้างในบางส่วน ดังที่ได้กล่าวข้างต้น

ผู้จัดทำได้รวมโปรแกรมไว้ที่ภาคผนวก ก. เพื่อที่ว่าเมื่อมีผู้สมใจในโครงงานนี้จะได้นำโปรแกรมไปแก้ไขต่อ
ทางผู้จัดทำจึงยังไม่สามารถเขียน

ผลการรันโปรแกรมในส่วนของการวิเคราะห์สัญญาณการมอดูเลทแบบ AM และ FM

1. การวิเคราะห์สัญญาณการมอดูเลทแบบ AM

เราจะได้ศึกษาถึงผลที่ได้จากการมอดูเลทแบบ AM ทั้งเชิงความถี่และสเป็คตรัมของสัญญาณ รูปแบบการมอดูเลทสัญญาณซายน์เขียนเป็นสมการคณิตศาสตร์

$$c(t) = A*(1 + m*\cos(w_m))*\cos(w_c)$$

โดยที่ A : เป็นแอมปริจูดของสัญญาณเบสแบนด์

m : เป็นค่าเปอร์เซนต์การมอดูเลท

w_m : เป็นความถี่ของเบสแบนด์

w_c : เป็นความถี่ของแคเรียร์

เมื่อกำหนด Amplitude of signal = 10mv, Percent of modulate = 100, Frequency baseband = 100 Hz, Frequency carrier = 1000 kHz จะได้สัญญาณเชิงเวลาดังรูปที่ 5.3 และสเปคตรัมสัญญาณแสดงดังรูป 5.4 ถึง 5.7โดยมีการขยายสเกลความถี่ออกไปเพื่อดูรายละเอียดทางความถี่ จะเห็นถึงไซน์โลบที่ชัดเจน

หลังจากนั้นลองเปลี่ยนค่าเปอร์เซนต์การมอดูเลท จาก 100 เป็น 125 และ 75 % ตามลำดับ จะดูสัญญาณทางเวลาและสเปคตรัมของสัญญาณได้ดังรูป 5.12, 5.13, 5.15, 5.16 ตามลำดับจะสังเกตุเห็นว่าเมื่อเปอร์เซนต์การมอดูเลทเกิน 100% แล้งจะทำให้สัญญาณ AM เพี้ยนไปดังรูปที่ 5.12 ดังนั้นการมอดูเลทที่ได้สัญญาณแรงที่สุดโดยไม่เกิดการผิดเพี้ยนคือ 100 % สำหรับสเป็ค

2. การวิเคราะห์สัญยาณการมอดูเลทแบบ FM

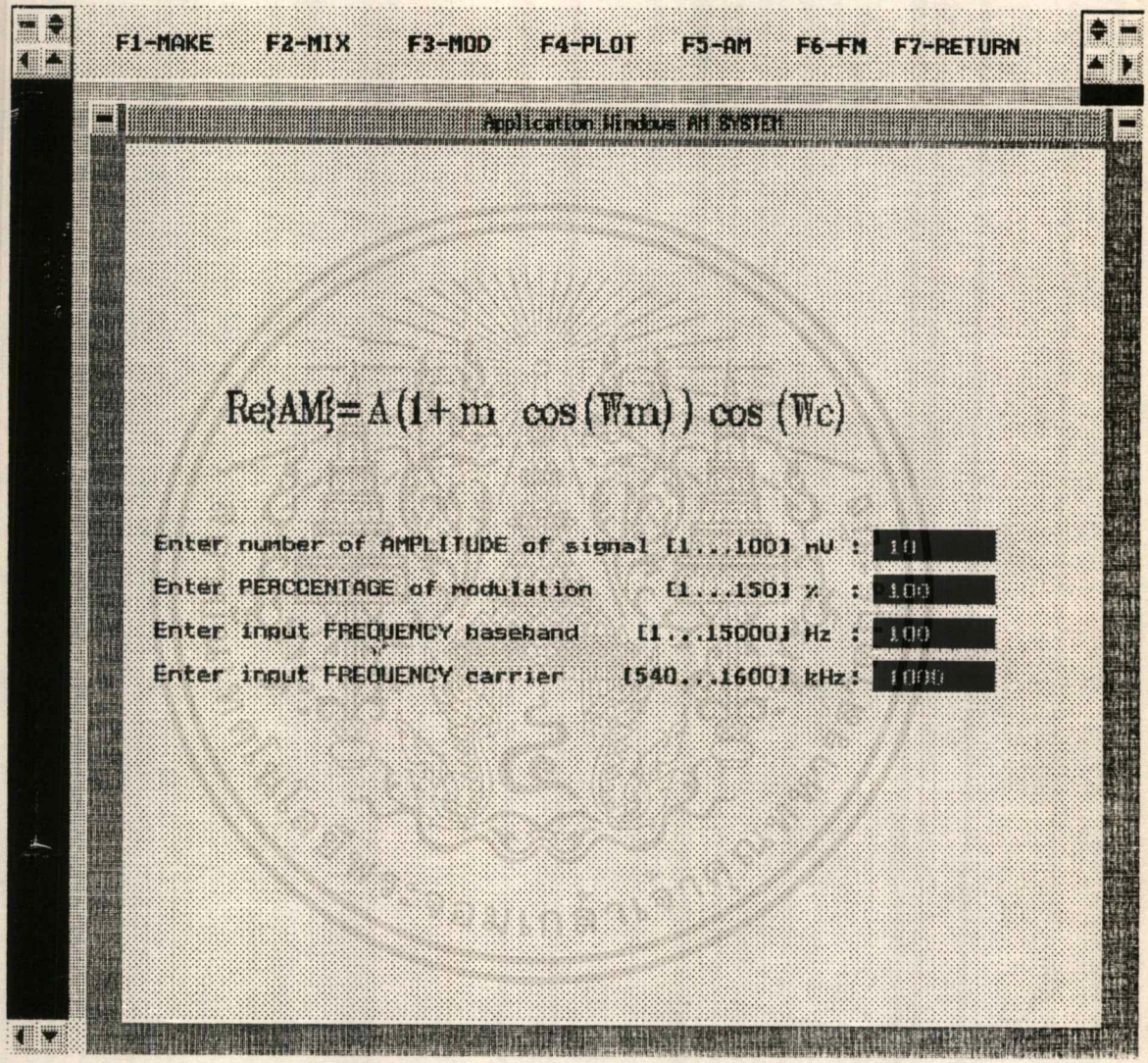
การมอดูเลทสัญญาณซายน์แบบ FM แสดงเป็นสมการการมอดูเลทได้ คือ

$$c(t) = A*\cos((w_c) + \beta*\sin(w_m))$$

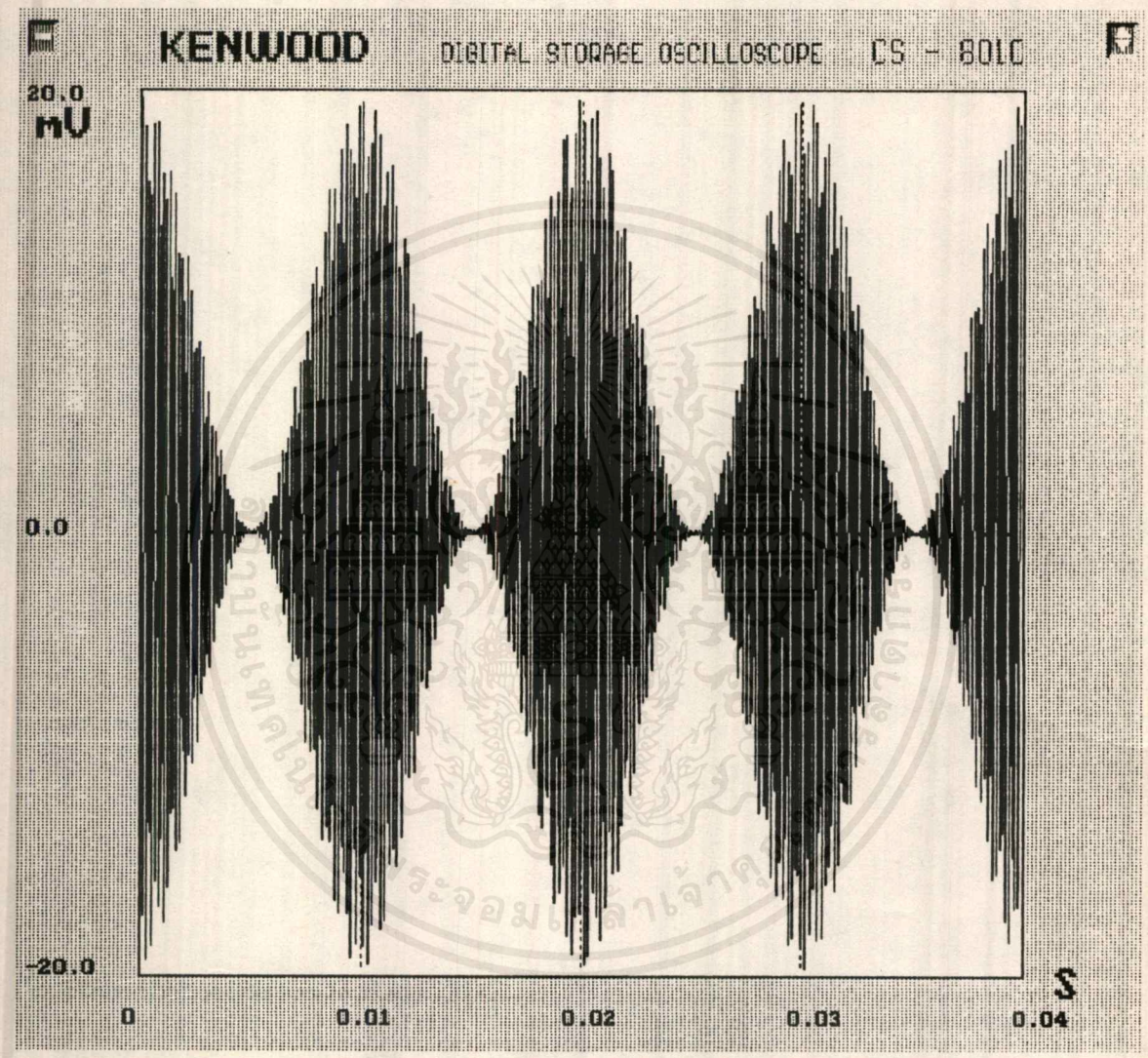
เมื่อกำหนด Amplitude of signal = 10, modulation index = 5, carrier frequency = 88, baseband frequency = 100

จะได้สัญญาณทางเวลาดังรูปที่ 5.18 เมื่อเราเปลี่ยนค่า มอดูเลชั่นอินเดคเป็น 1, 5, 10, 15 แล้วเราจะมาวิเคราะห์สเป็คตรัมดู สังเกตุเห็นว่าเมื่อค่ามอดูเลชั่น อินเด็ค เพิ่มขึ้นสเป็คตรัมของสัญญาณจะเพิ่มตาม เปรียบเทียบจากรูป 5.20, 5.22, 5.24, 5.26 ตามลำดับ

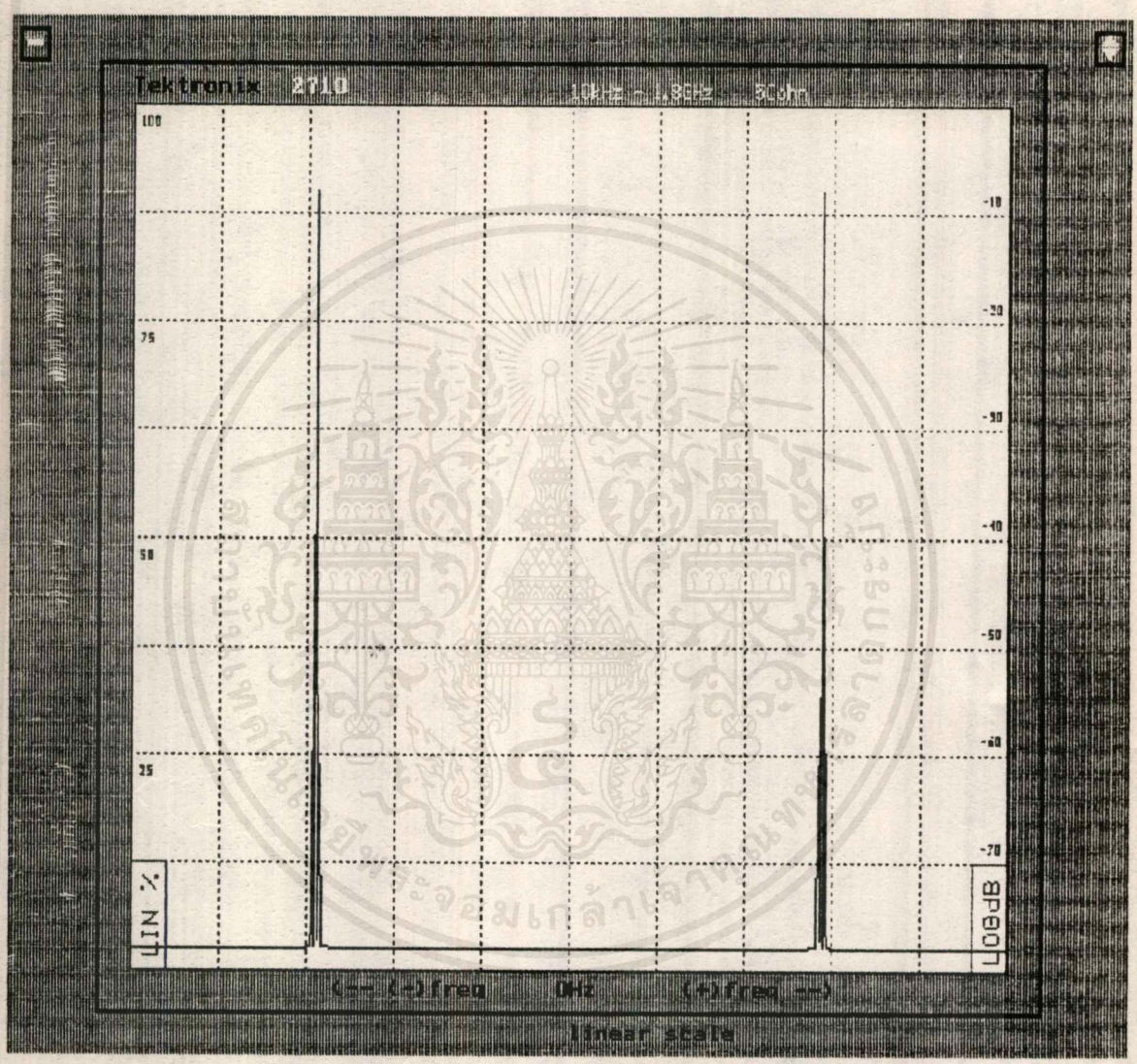
รูปที่ 5.1



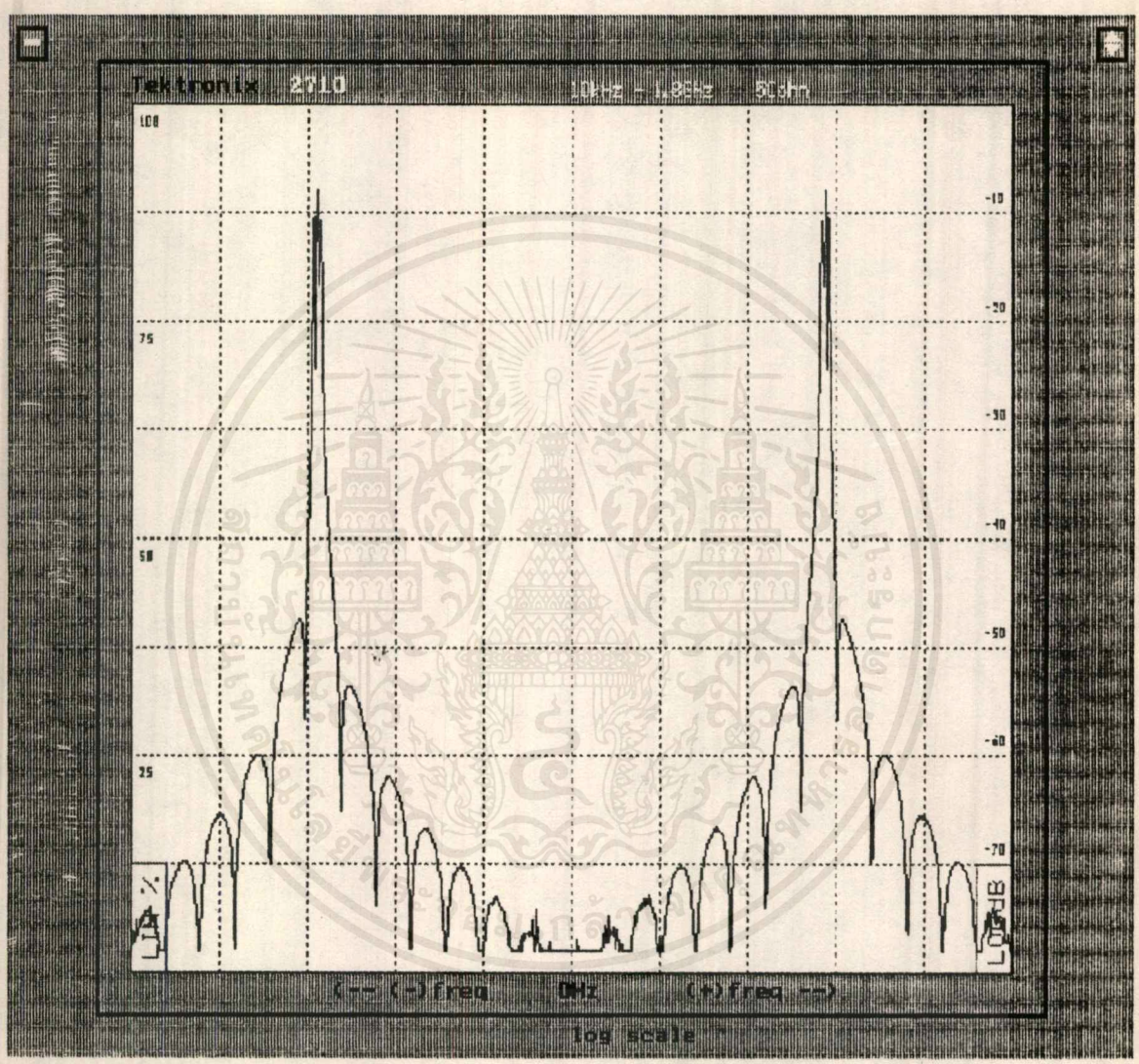
รูปที่ 5.2



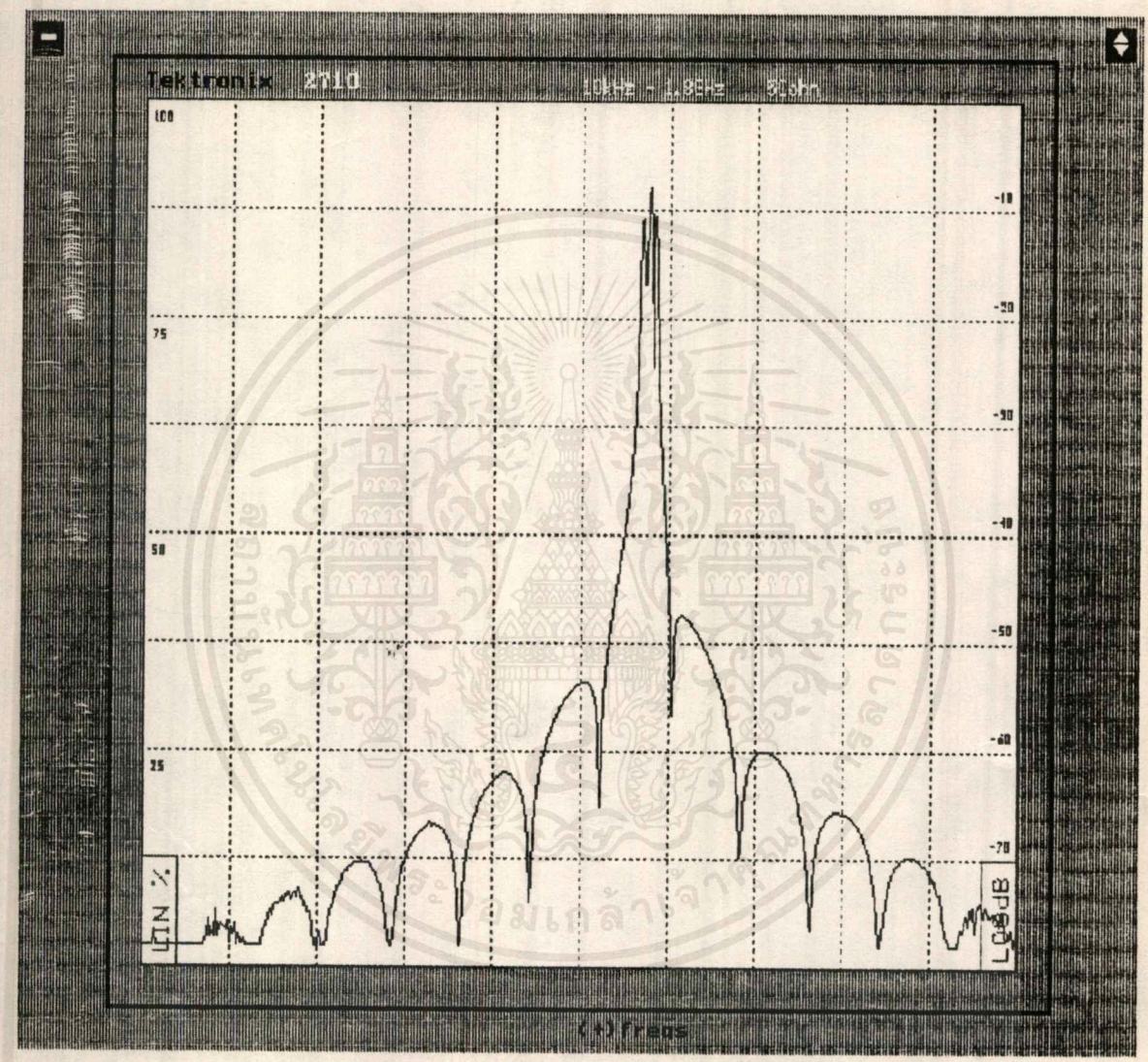
รูปที่ 5.3



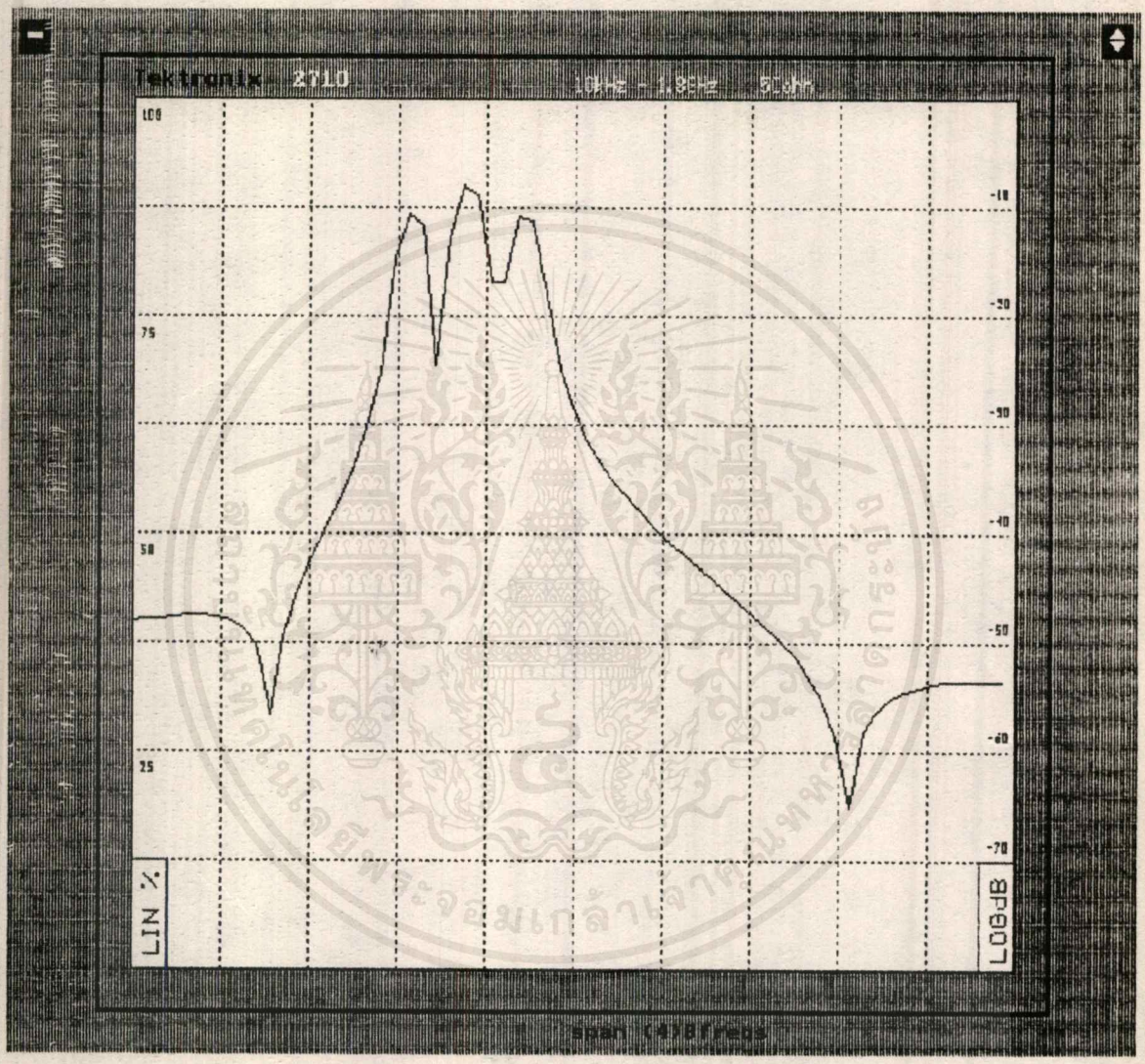
รูปที่ 5.4



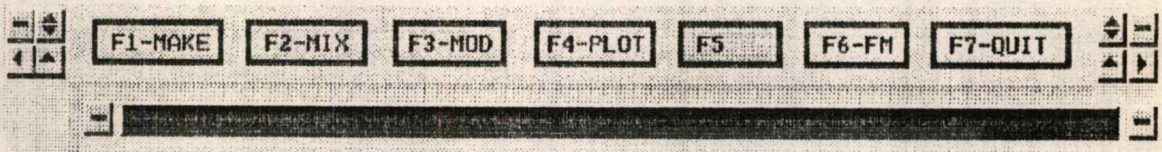
รูปที่ 5.5



รูปที่ 5.6



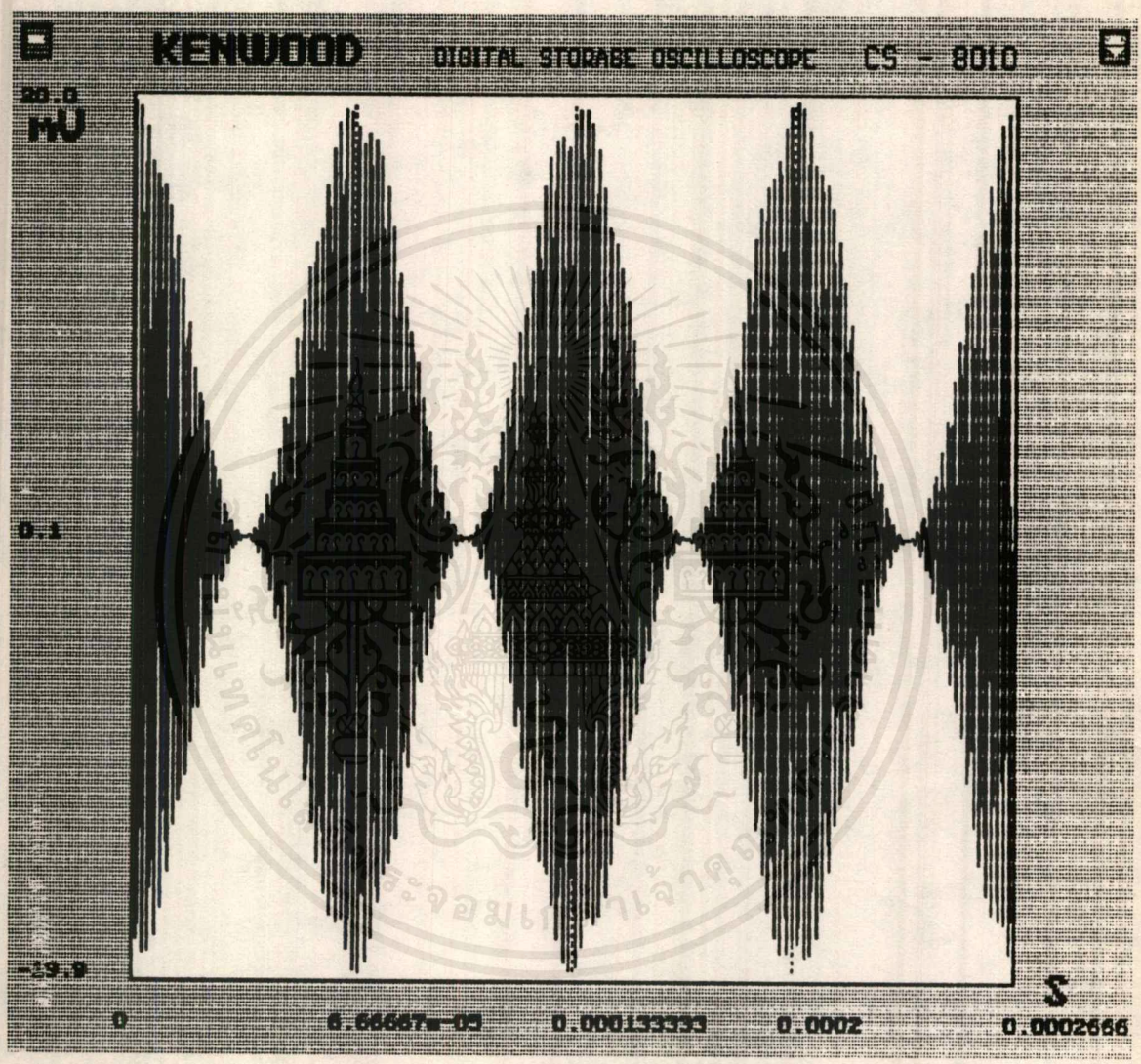
รูปที่ 5.7



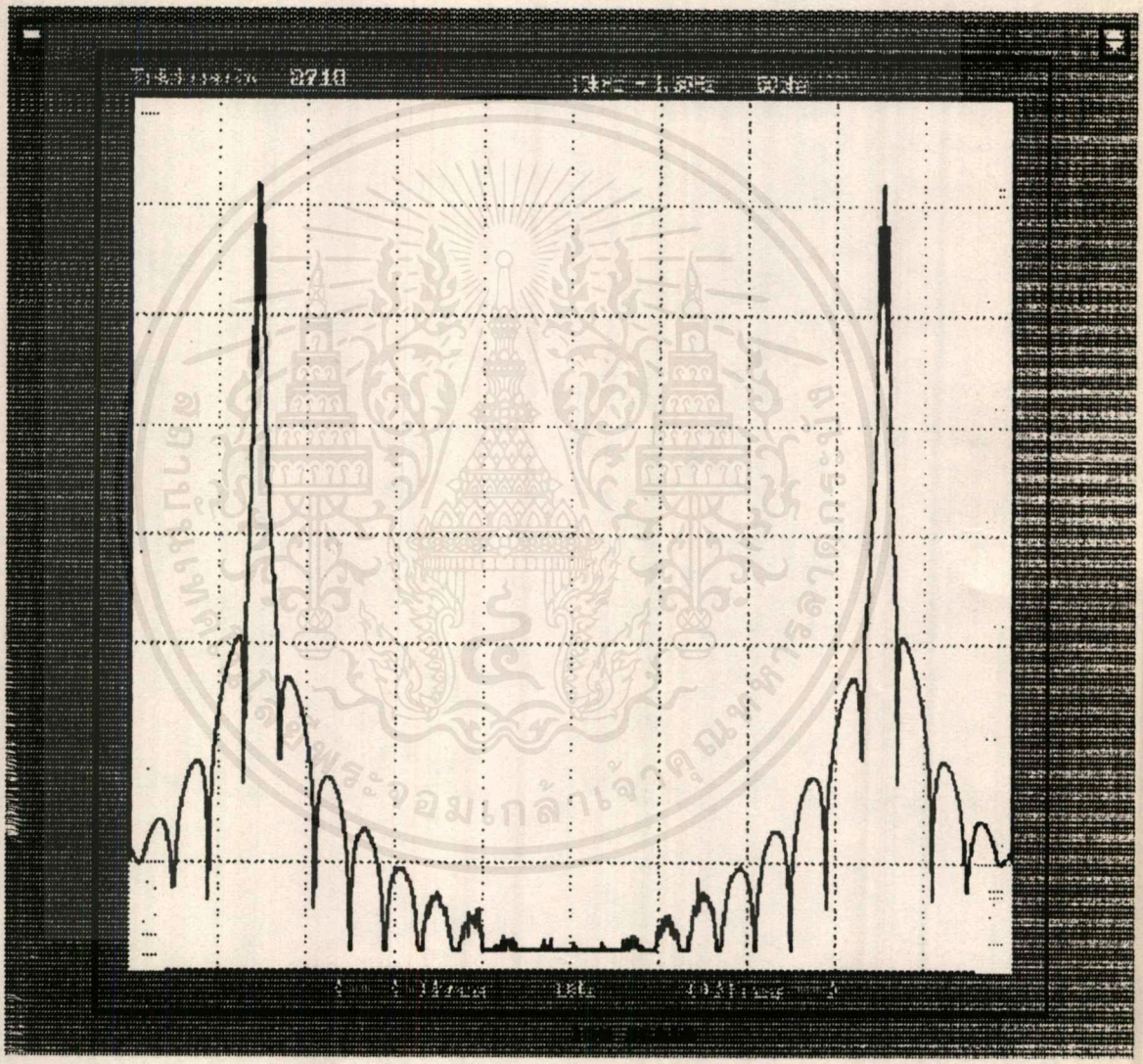
$$Re\{AM\} = A(1 + m\ \cos(\mathbb{W}m))\ \cos(\mathbb{W}c)$$

Enter number of AMPLITUDE of signal [1...100] mV : 10
Enter PERCCENTAGE of modulation [1...150] % : 100
Enter input FREQUENCY baseband [1...25000] Hz : 15000
Enter input FREQUENCY carrier [540...1600] kHz: 550

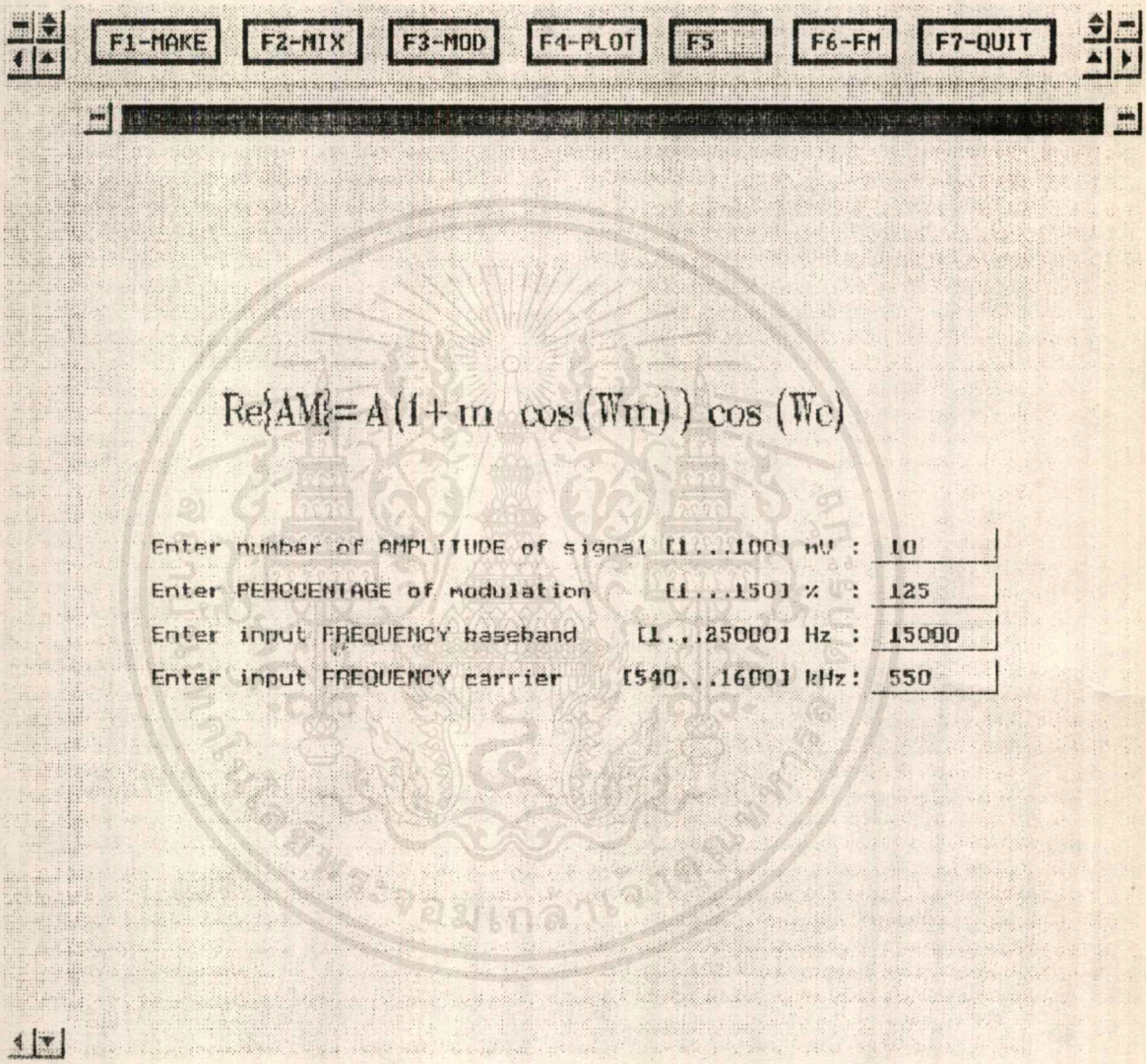
รูปที่ 5.8



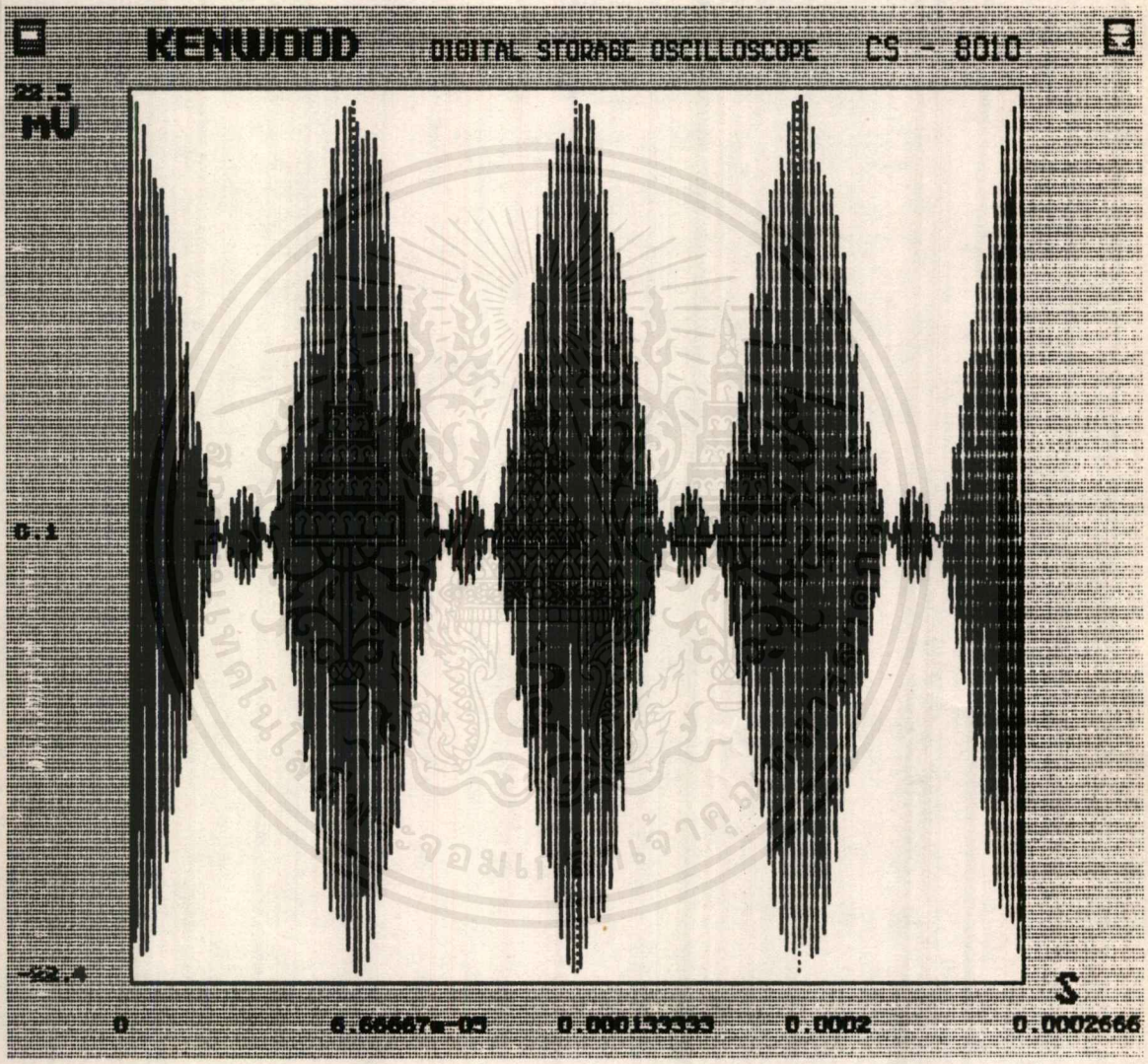
รูปที่ 5.9



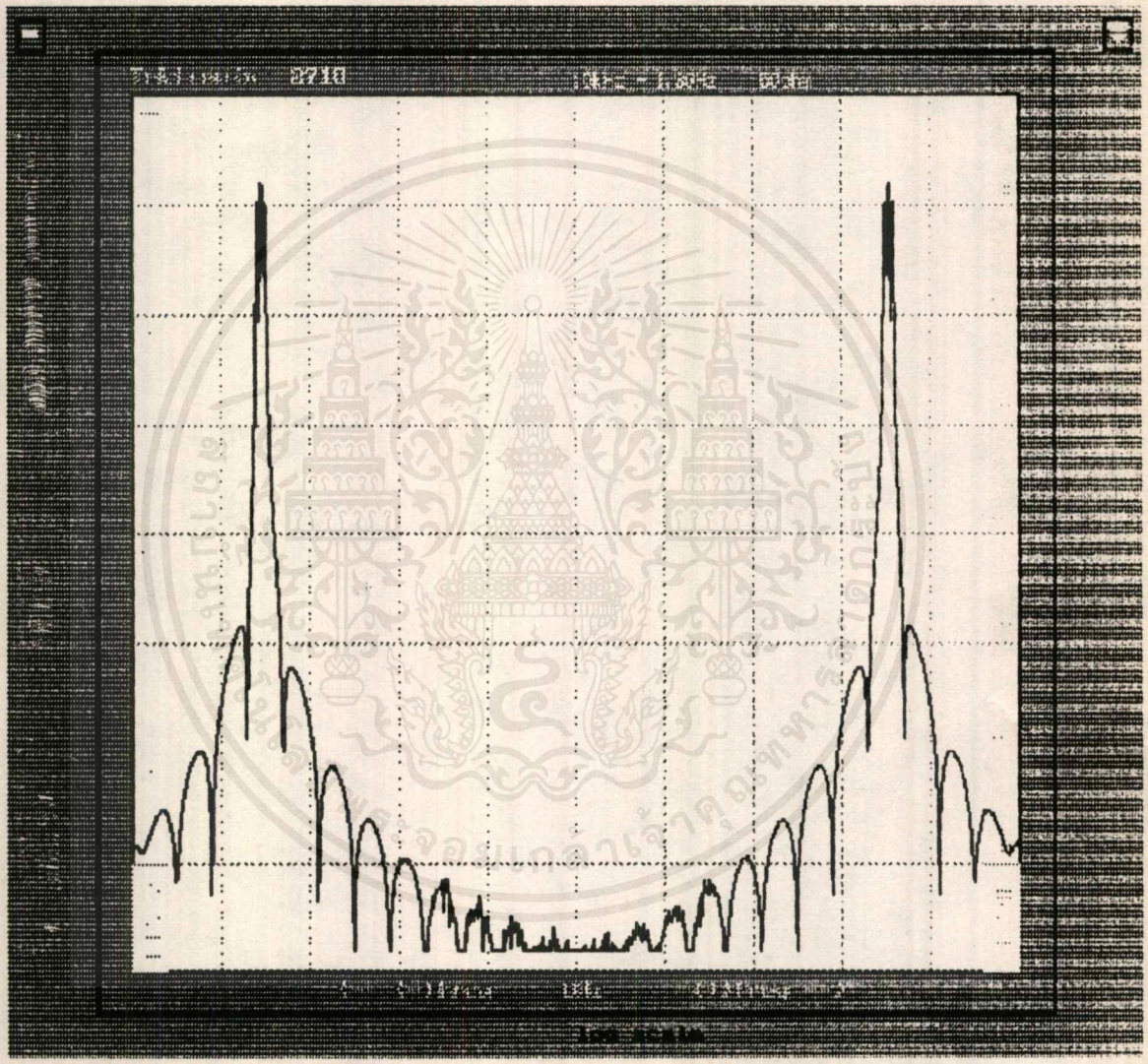
รูปที่ 5.10



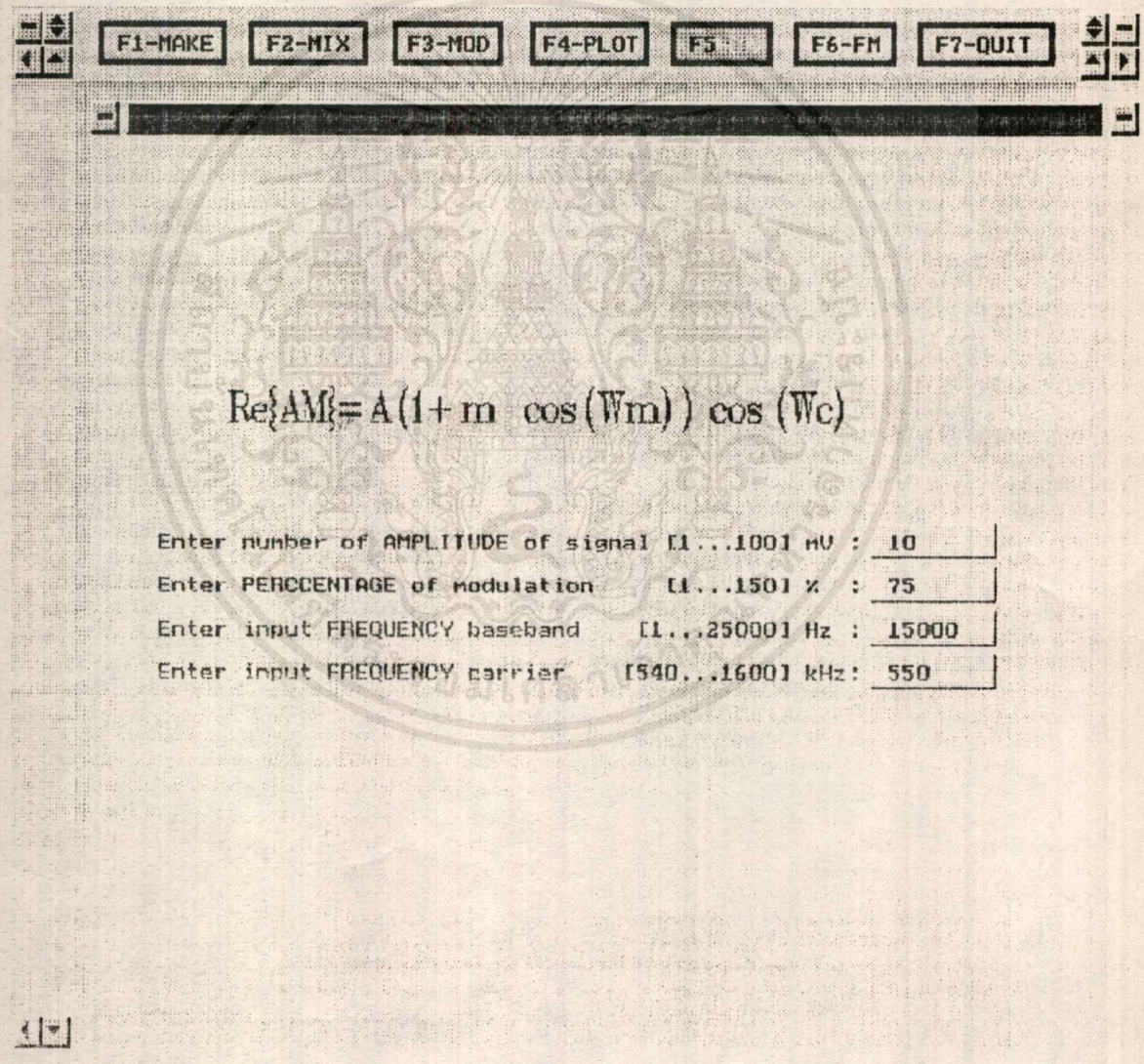
รูปที่ 5.11



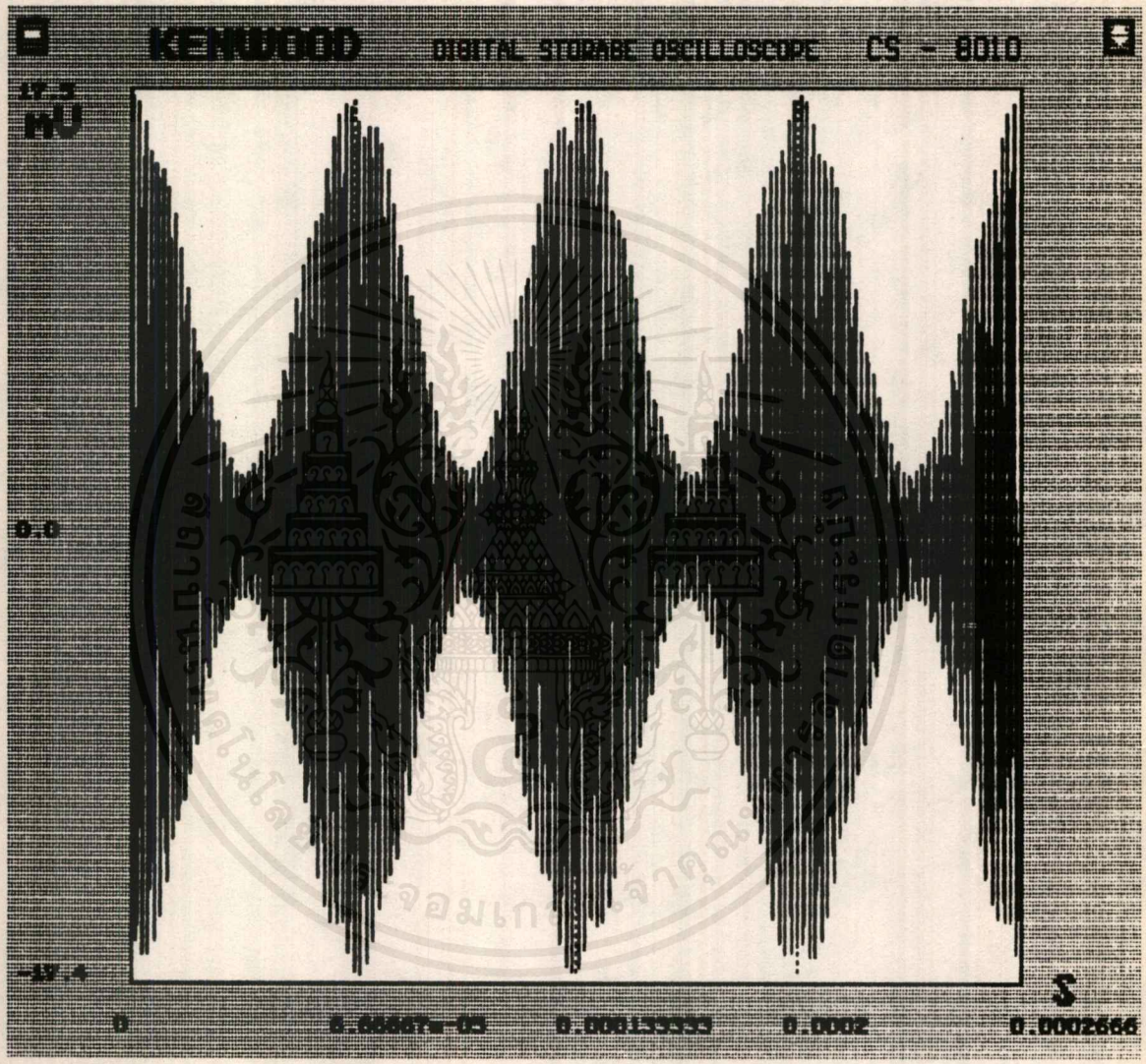
รูปที่ 5.12



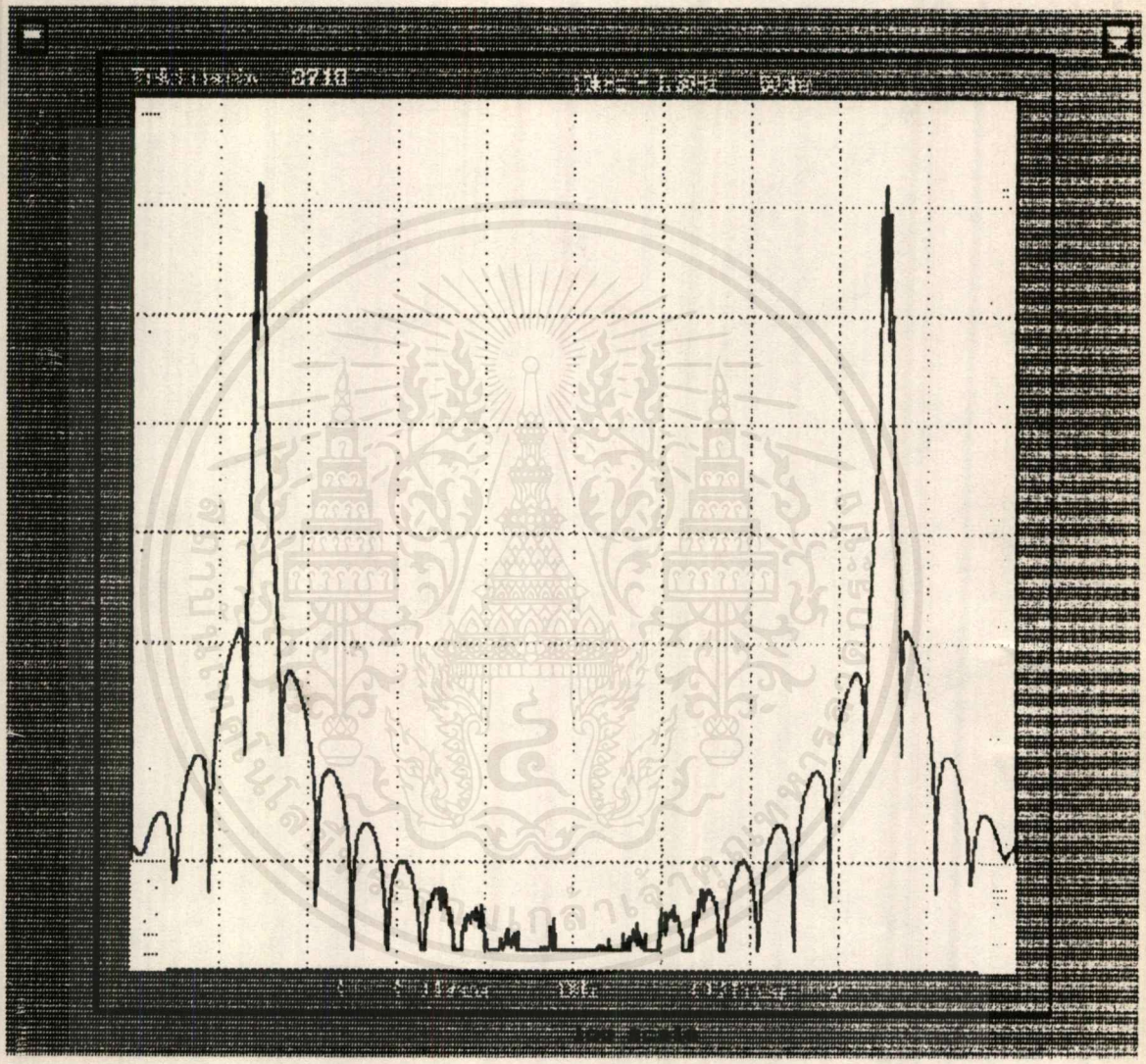
รูปที่ 5.13



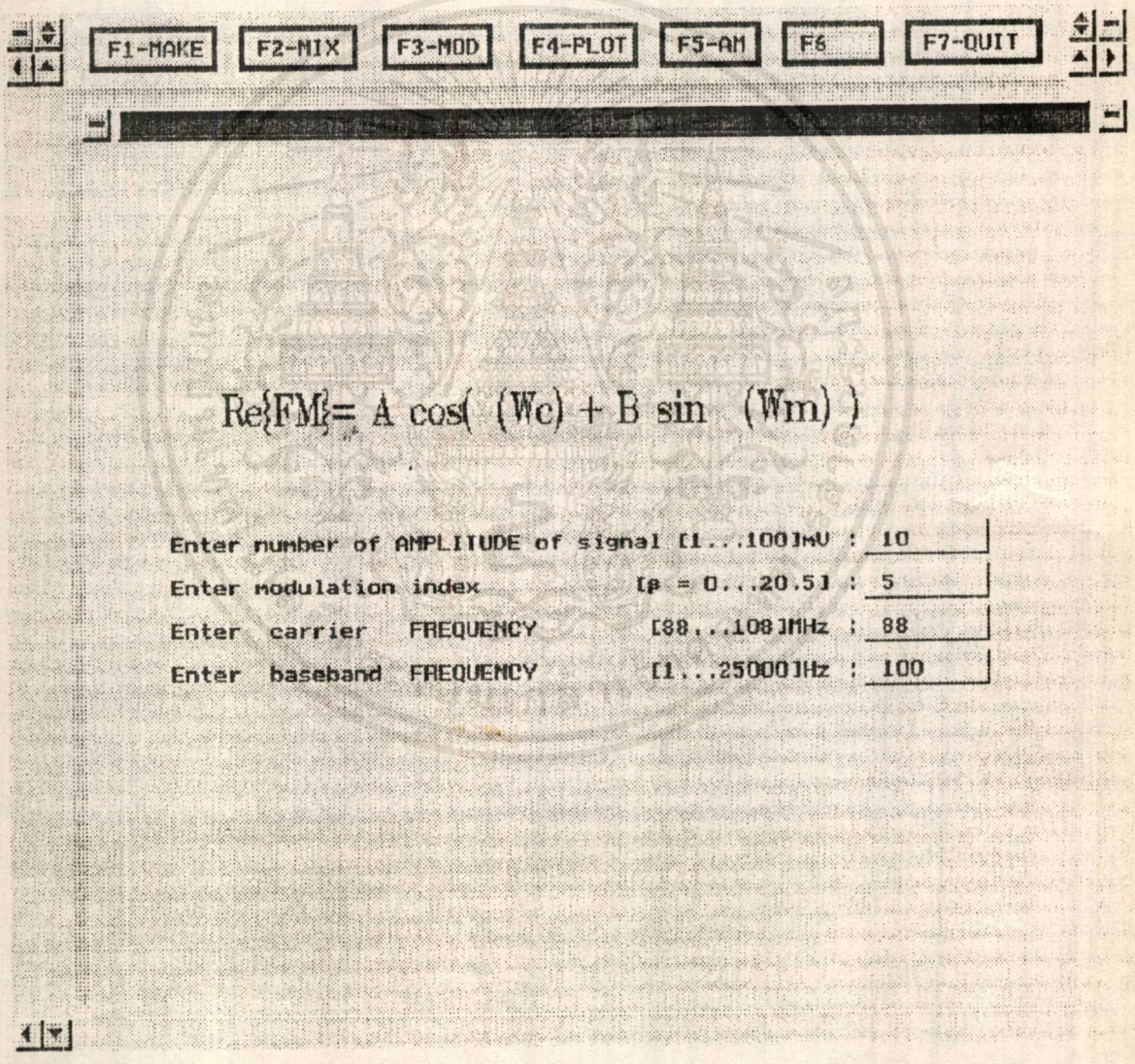
รูปที่ 5.14



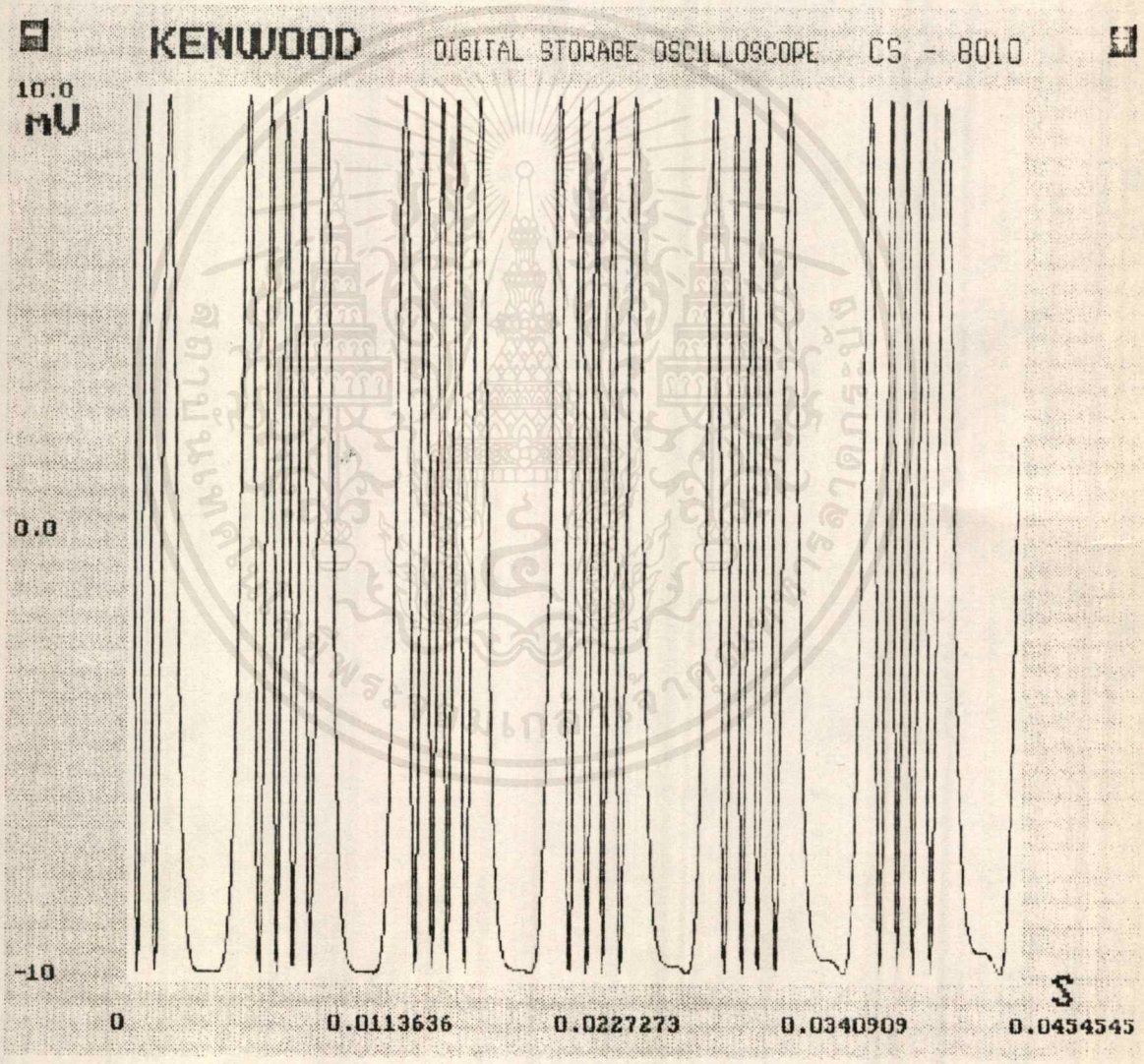
รูปที่ 5.15



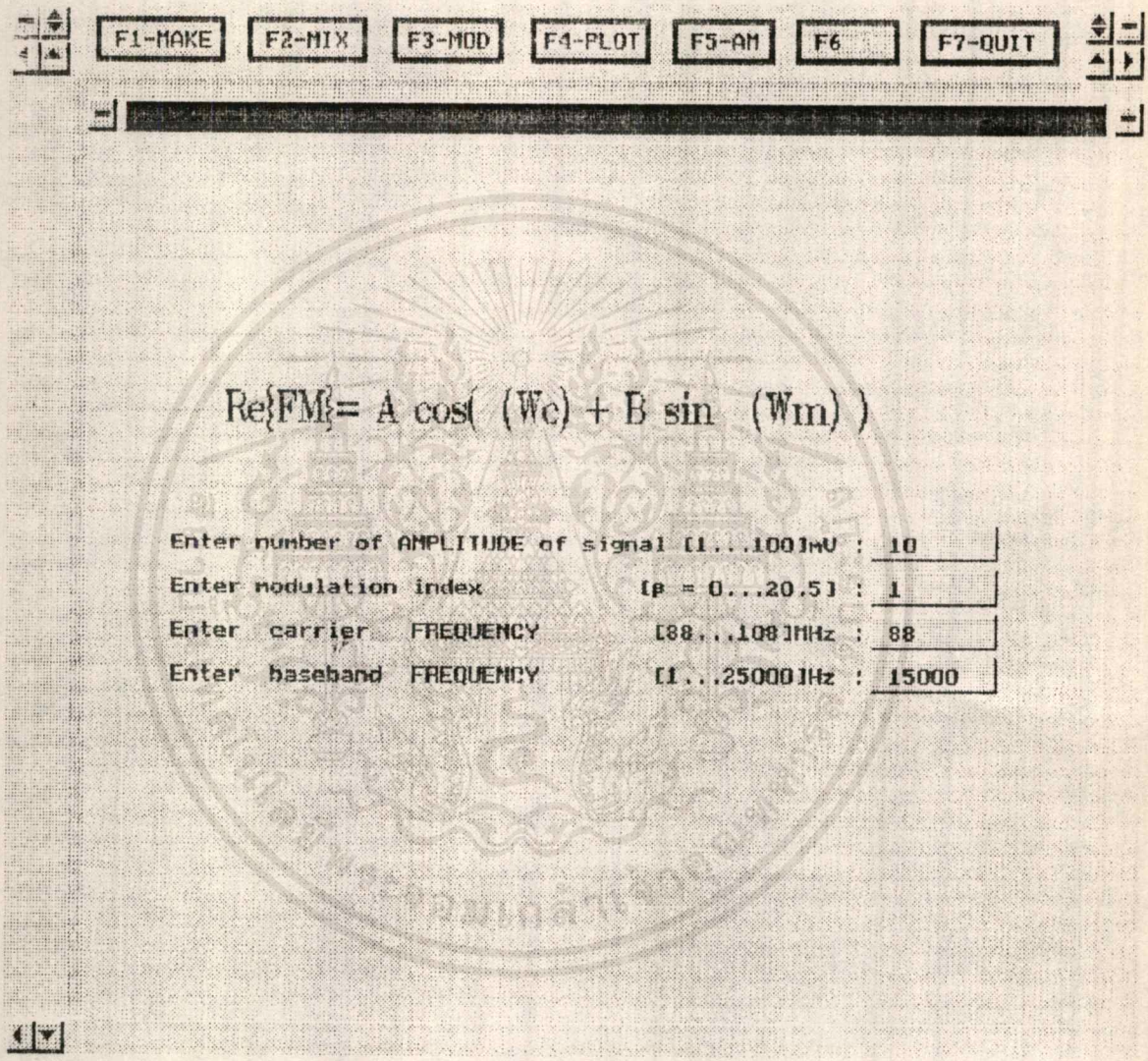
รูปที่ 5.16



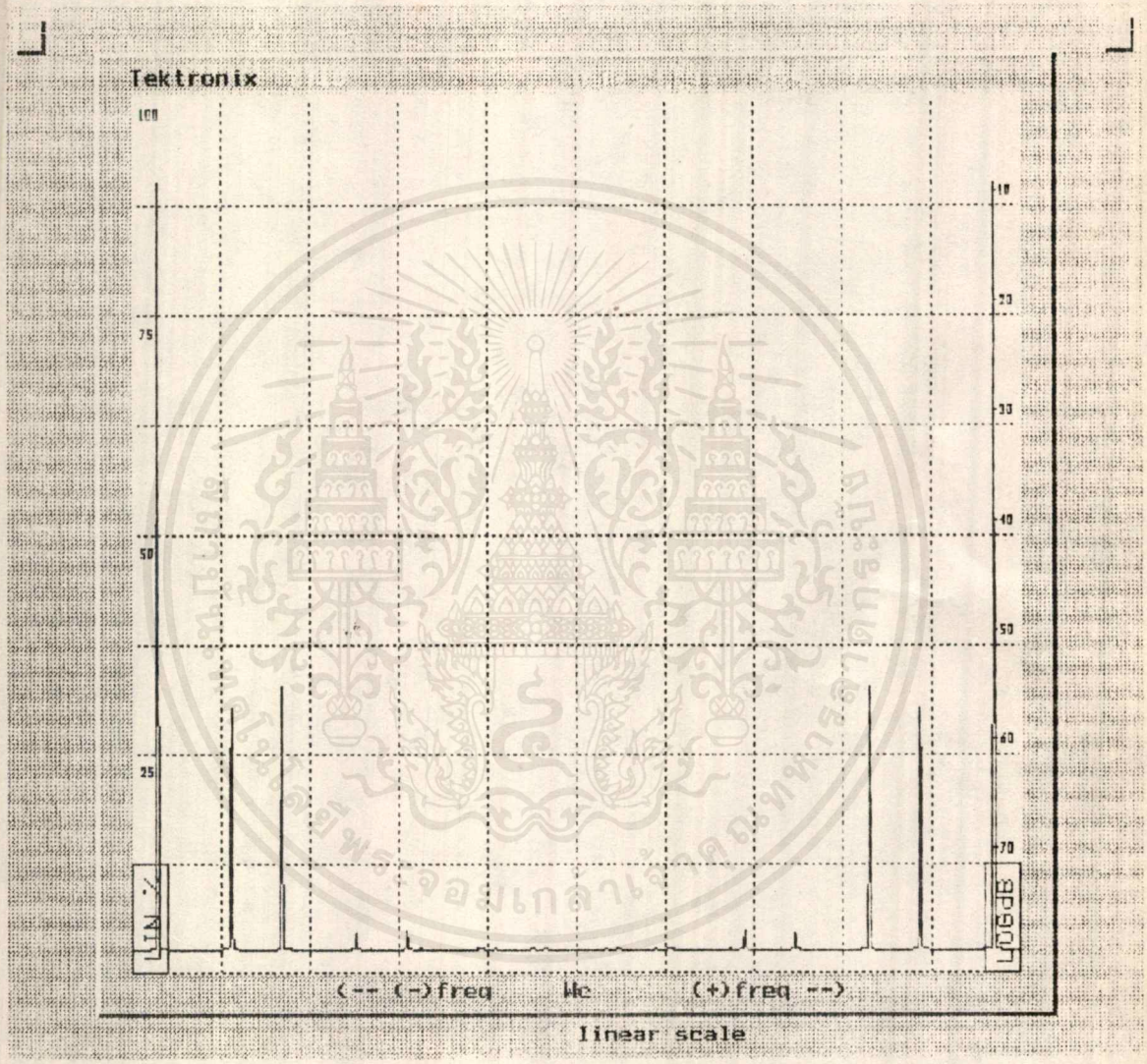
รูปที่ 5.17



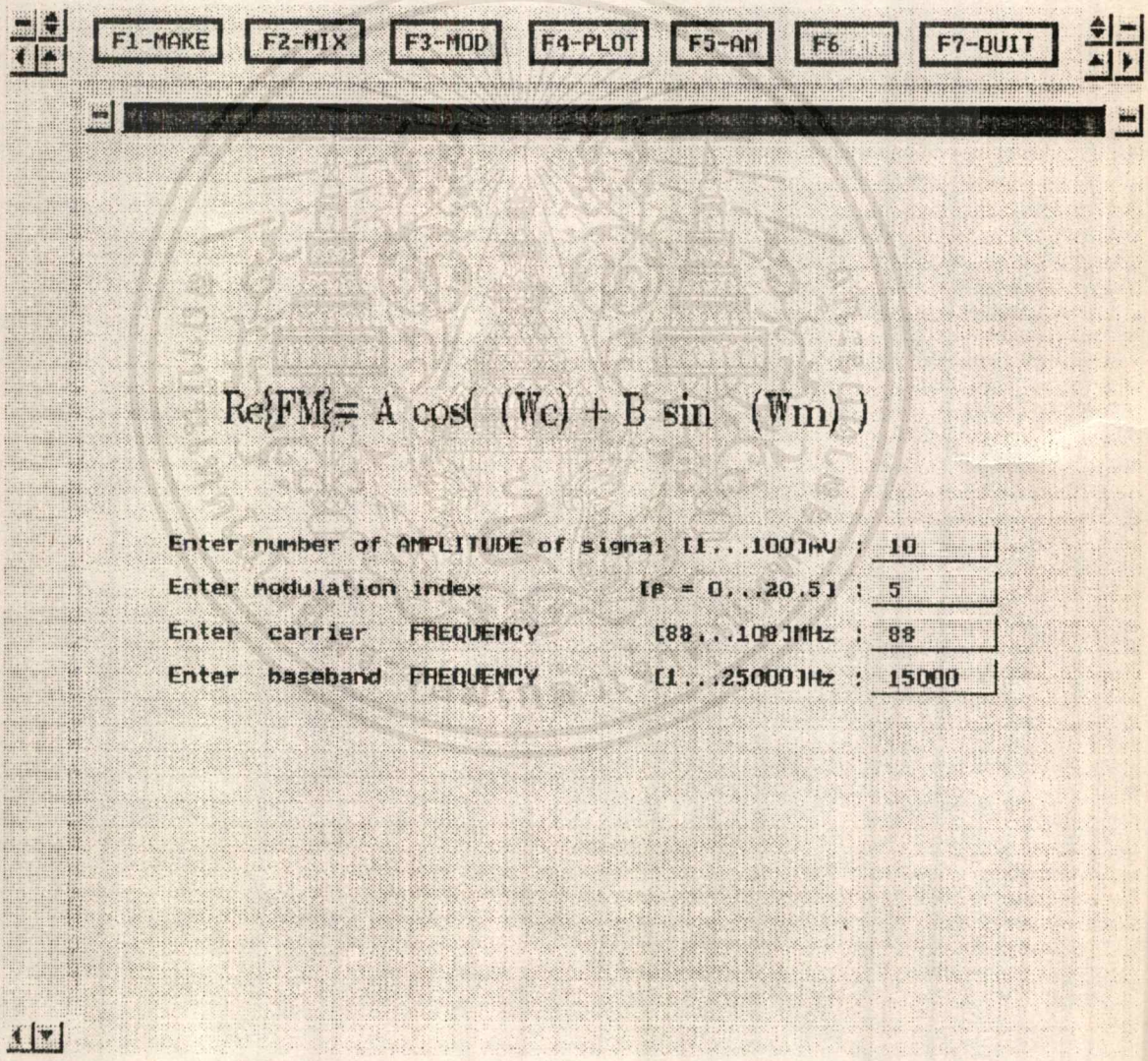
รูปที่ 5.18



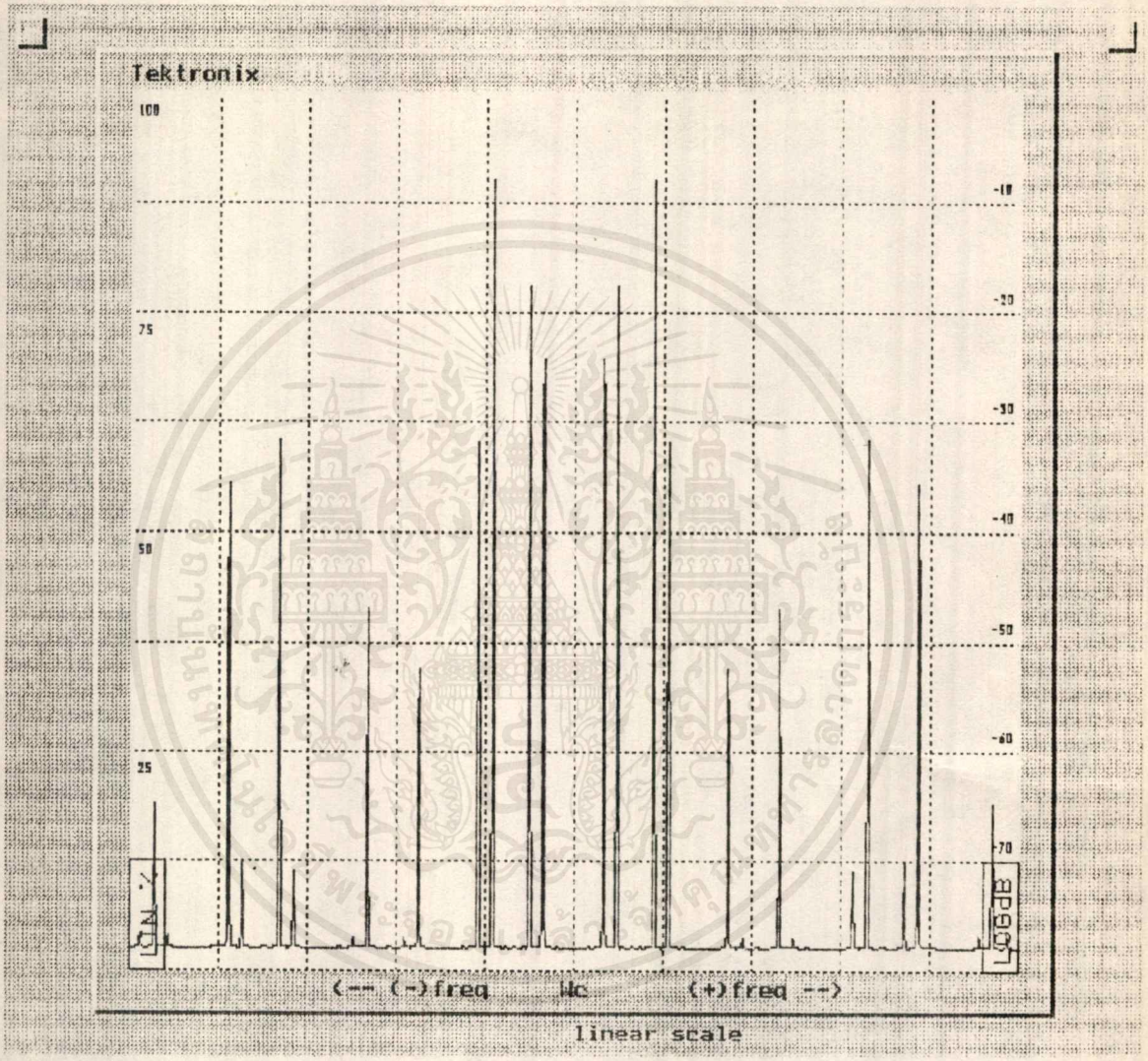
รูปที่ 5.19



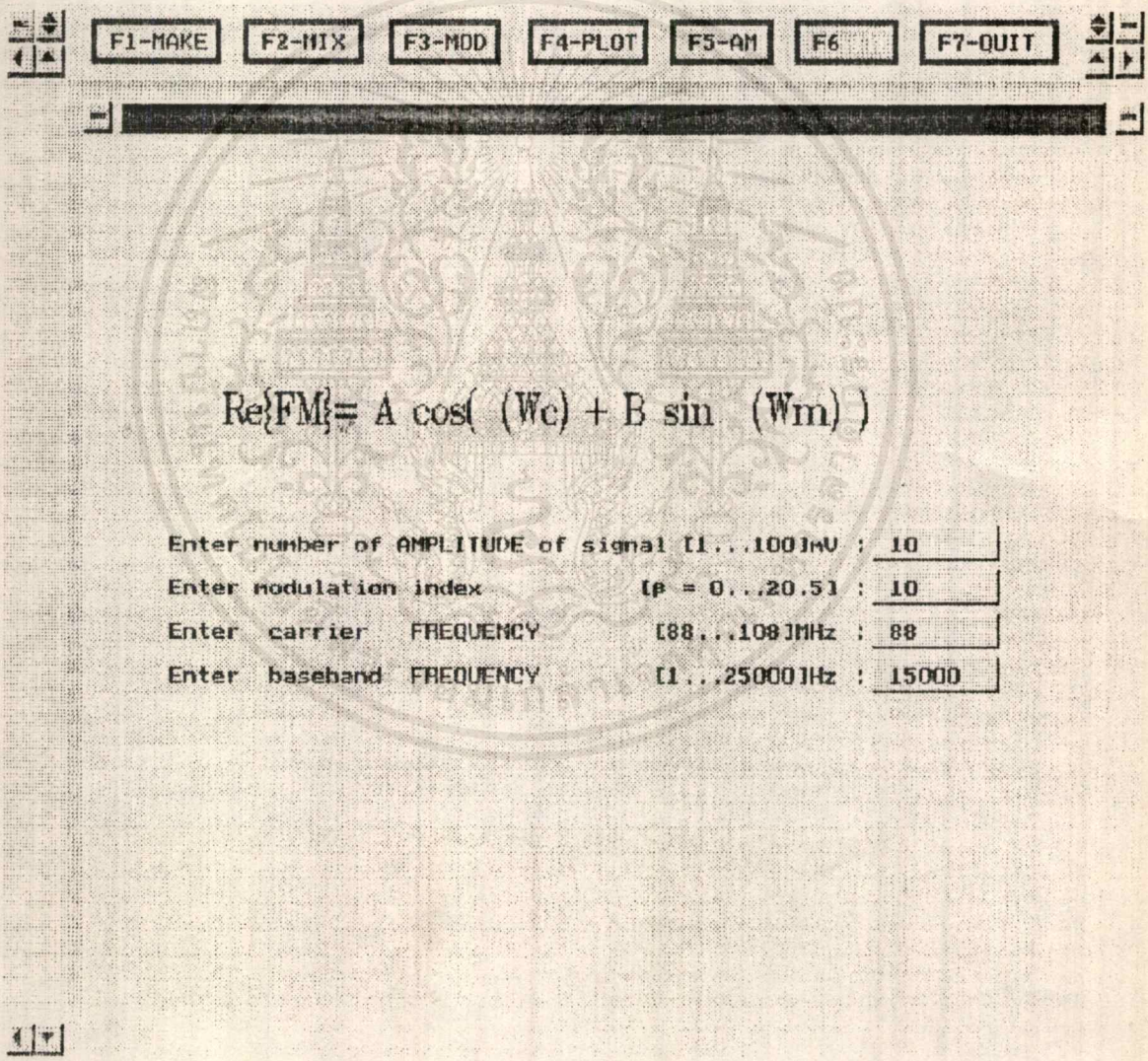
รูปที่ 5.20



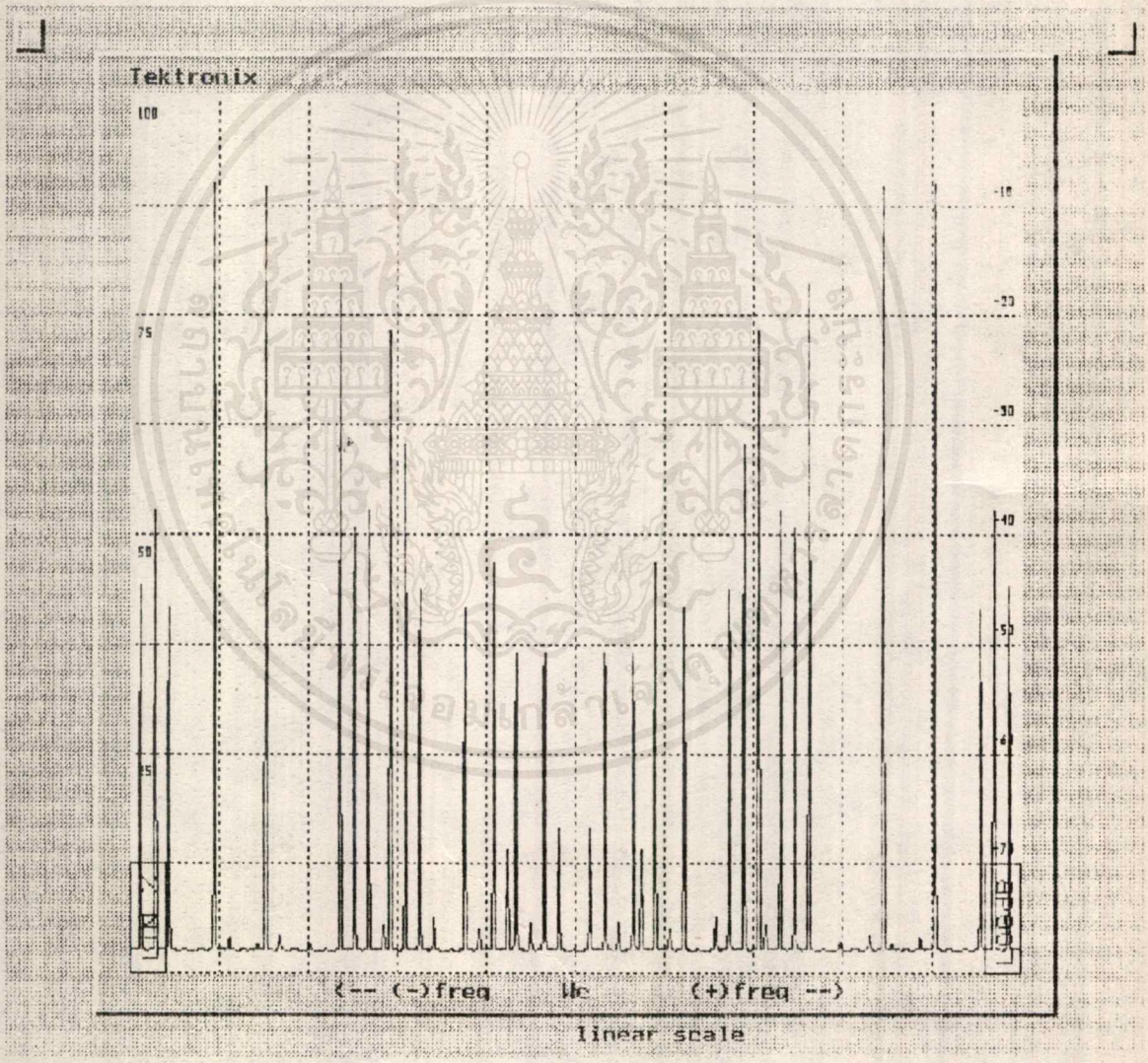
รูปที่ 5.21



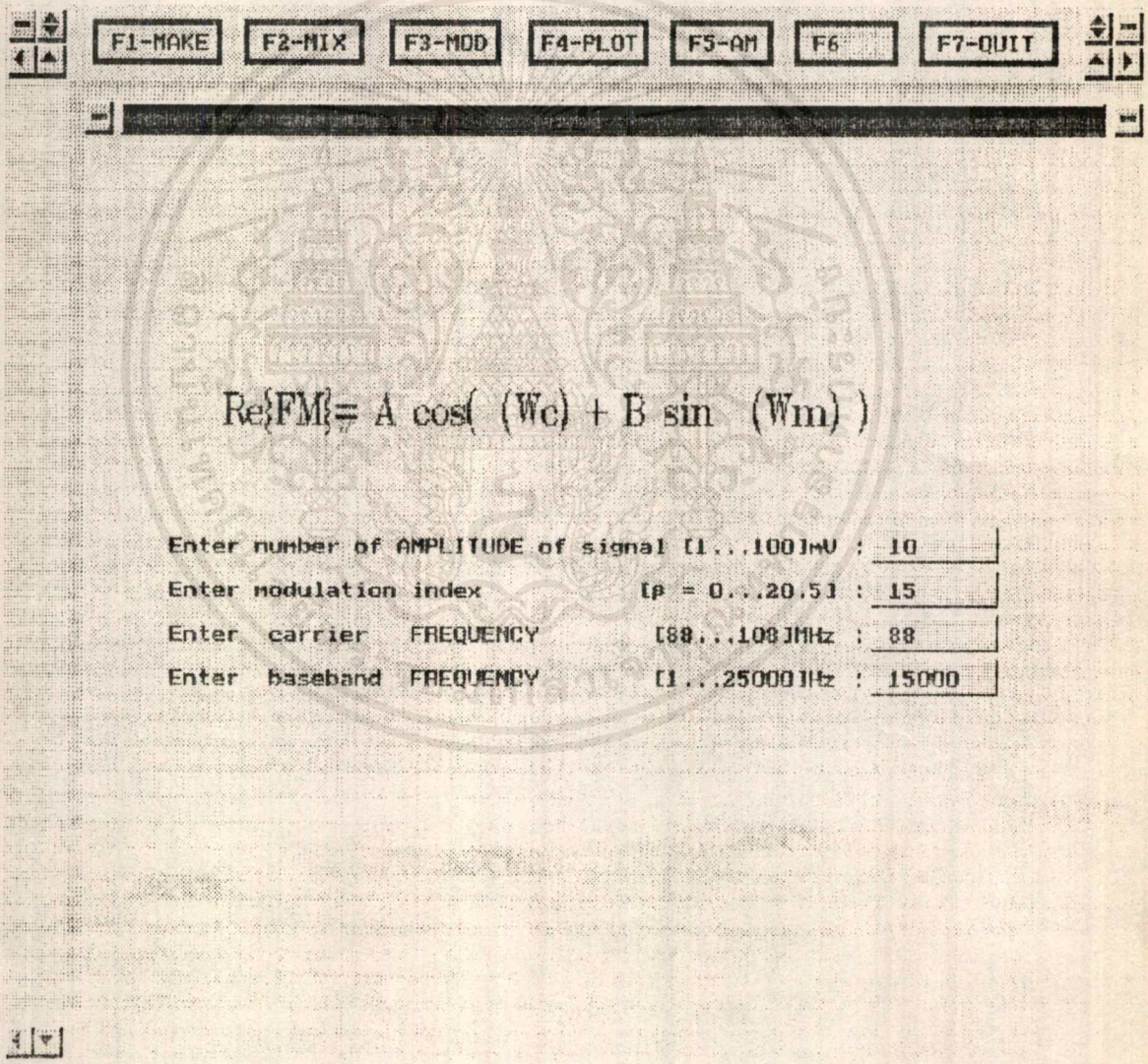
รูปที่ 5.22



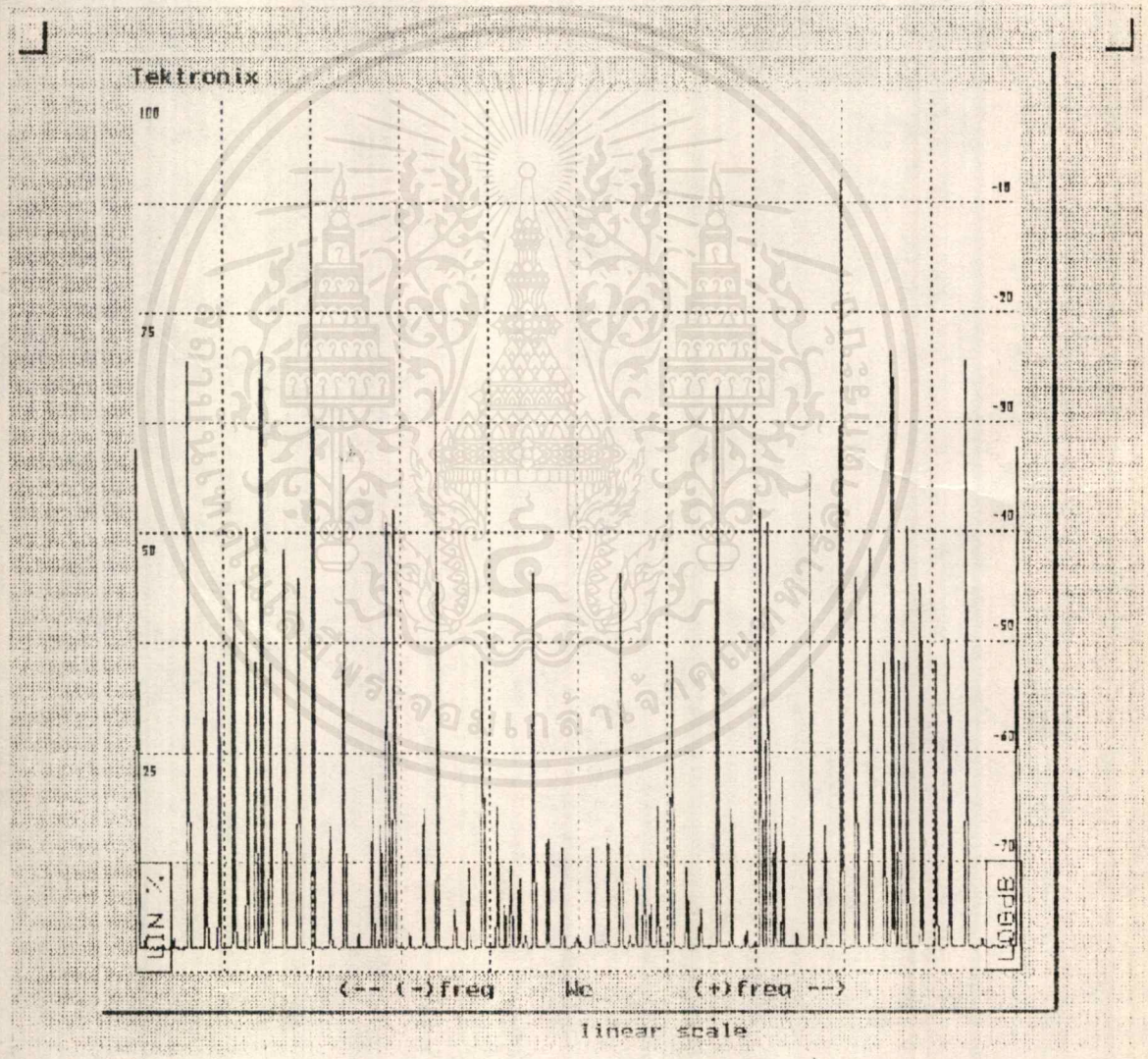
รูปที่ 5.23



รูปที่ 5.24



รูปที่ 5.25



รูปที่ 5.26

กิตติกรรมประกาศ

โครงงานปริญญานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี ส่วนหนึ่งได้รับความอนุเคราะห์จากอาจารย์ที่ปรึกษา คืออาจารย์กฤดากร กล่อมการ ซึ่งให้คำปรึกษา และให้ยืมตำราเพื่อใช้ค้นคว้าหาข้อมูล อีกทั้งยังอนุญาตให้ใช้ห้องและเครื่องคอมพิวเตอร์ในการเขียนโปรแกรม ยังผลให้การดำเนินงานสำเร็จลุลวงด้วยดี คณะผู้จัดทำใคร่ขอขอบพระคุณอาจารย์เป็นอย่างสูงมา ณ. โอกาสนี้

นายวีระ เนียมโภคะ รหัส 34131171
นายวีรศักดิ์ นามวง รหัส 34131172
นายศราวุธ แก้วกันเนตร รหัส 34131173

เอกสารอ้างอิง

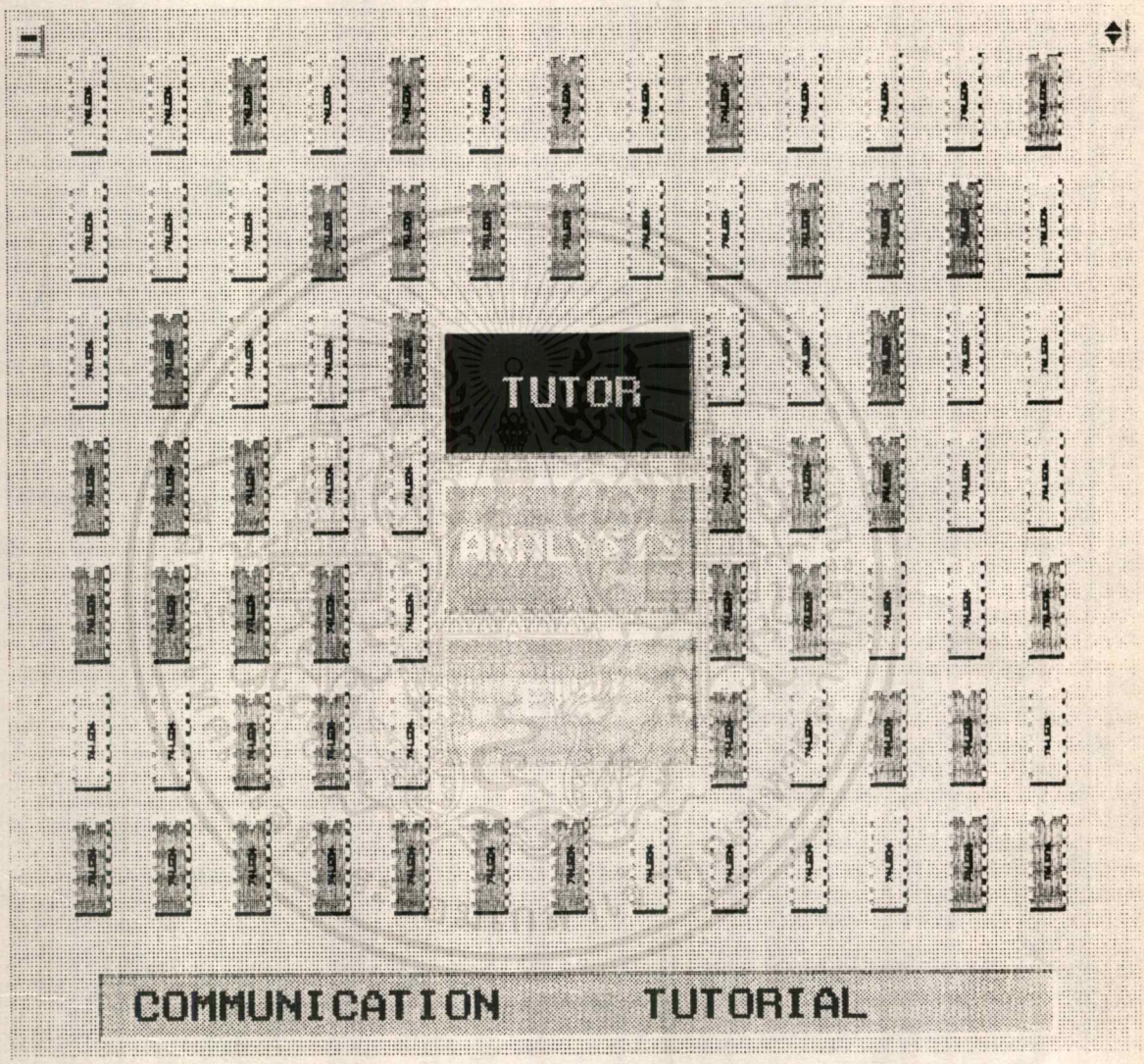
ดร.ประสิทธิ์ ประพิณมงคลการ หลักการระบบสื่อสาร: บริษัทซีเอ็ดยูเคชั่น จำกัด

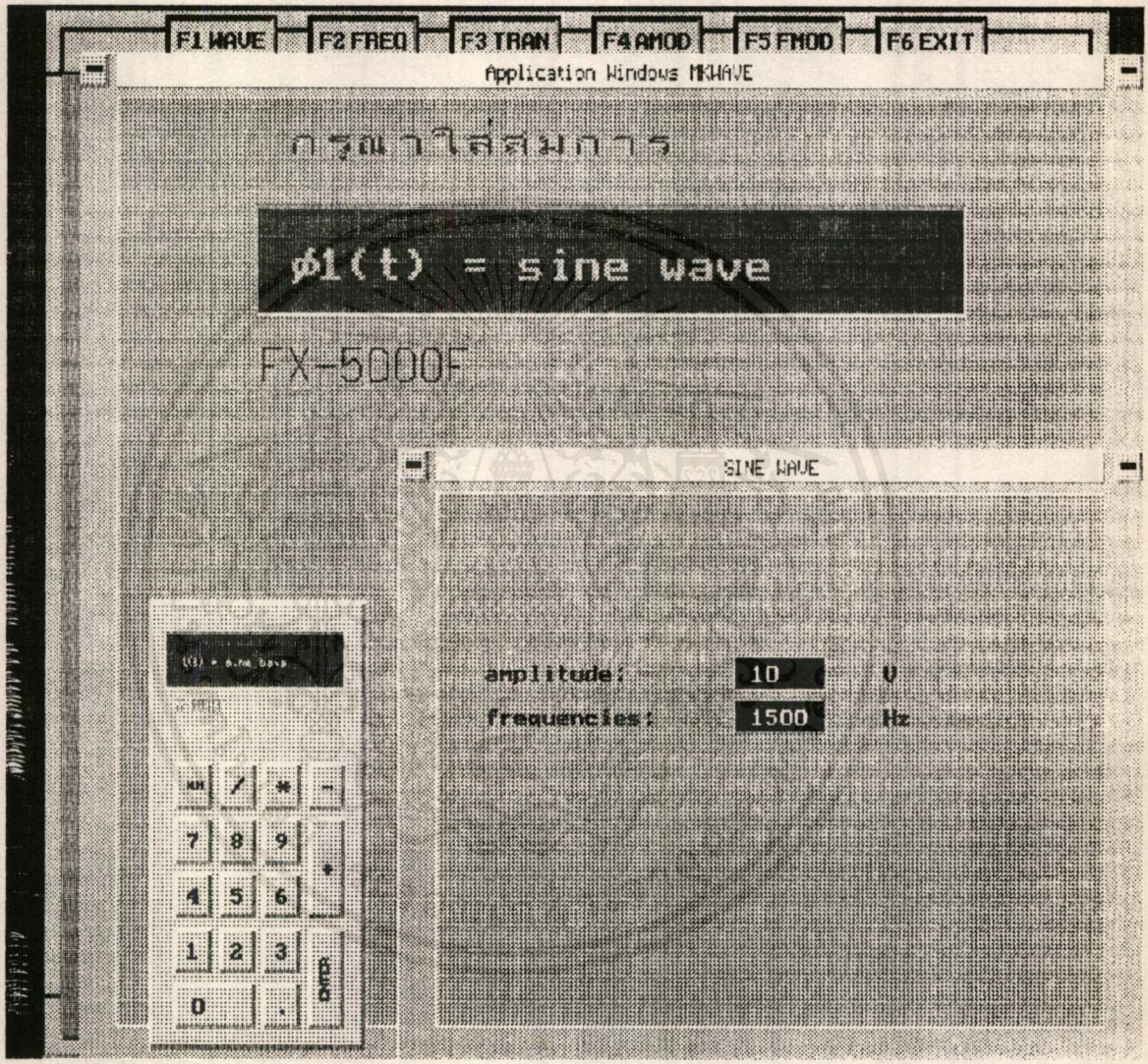
มนตรี พจนารถลาวัณย์ การเขียนโปรแกรมคอมพิวเตอร์ด้วย เทอร์โบซี: บริษัท ซีเอ็ดยูเคชั่น จำกัด

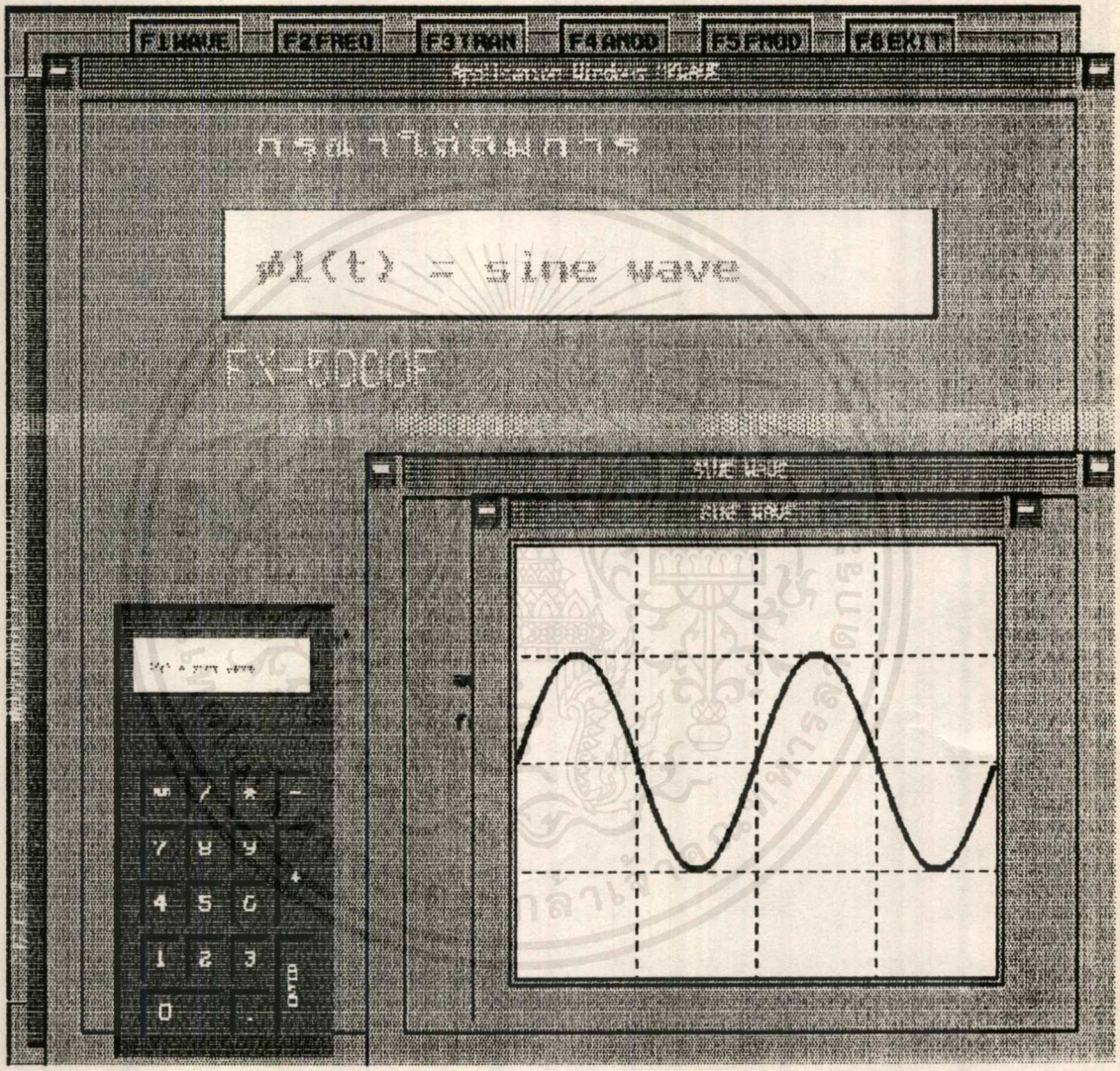
Robert M. Gagliardi Introduction to Communications Engineering. second editor: A Wiley-Interscience Publication 1988

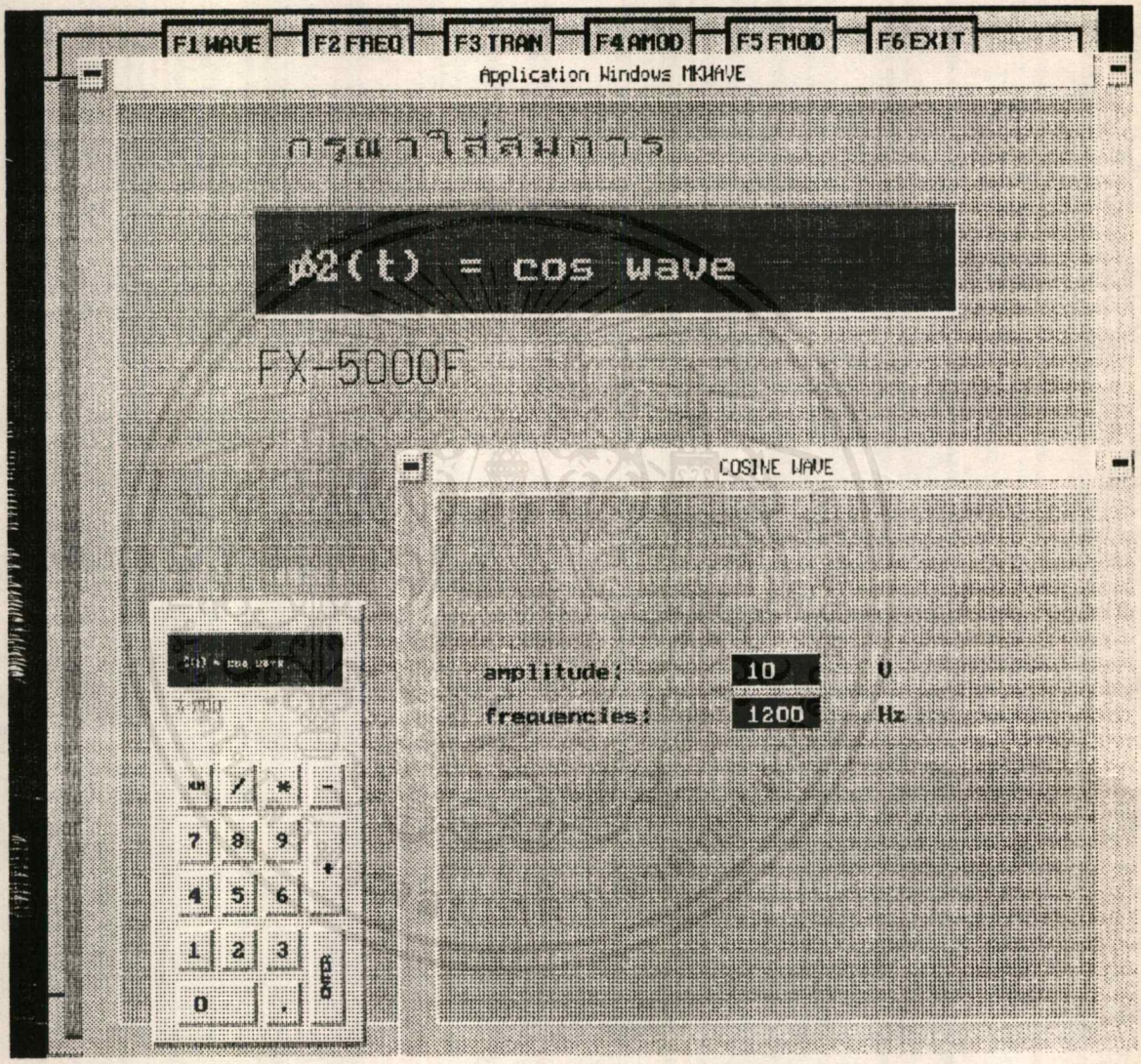
Paul M. Embree, Bruce Kimble C Language Angorithms for Digital Signal Processing: Prentice-Hall, Inc. Publication 1991

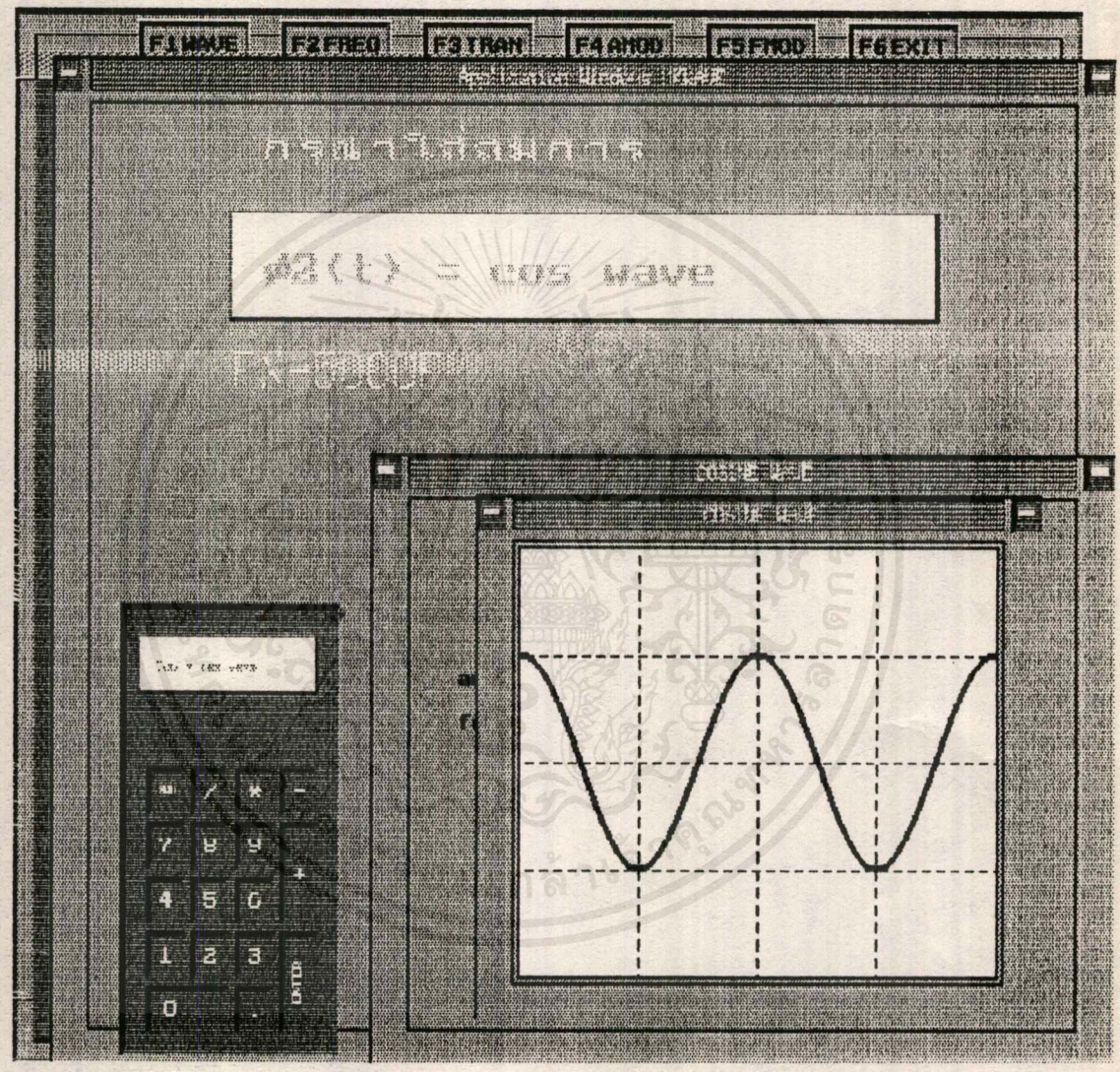
ภาคผนวก ก. SOFTWARE PROGRAMMING

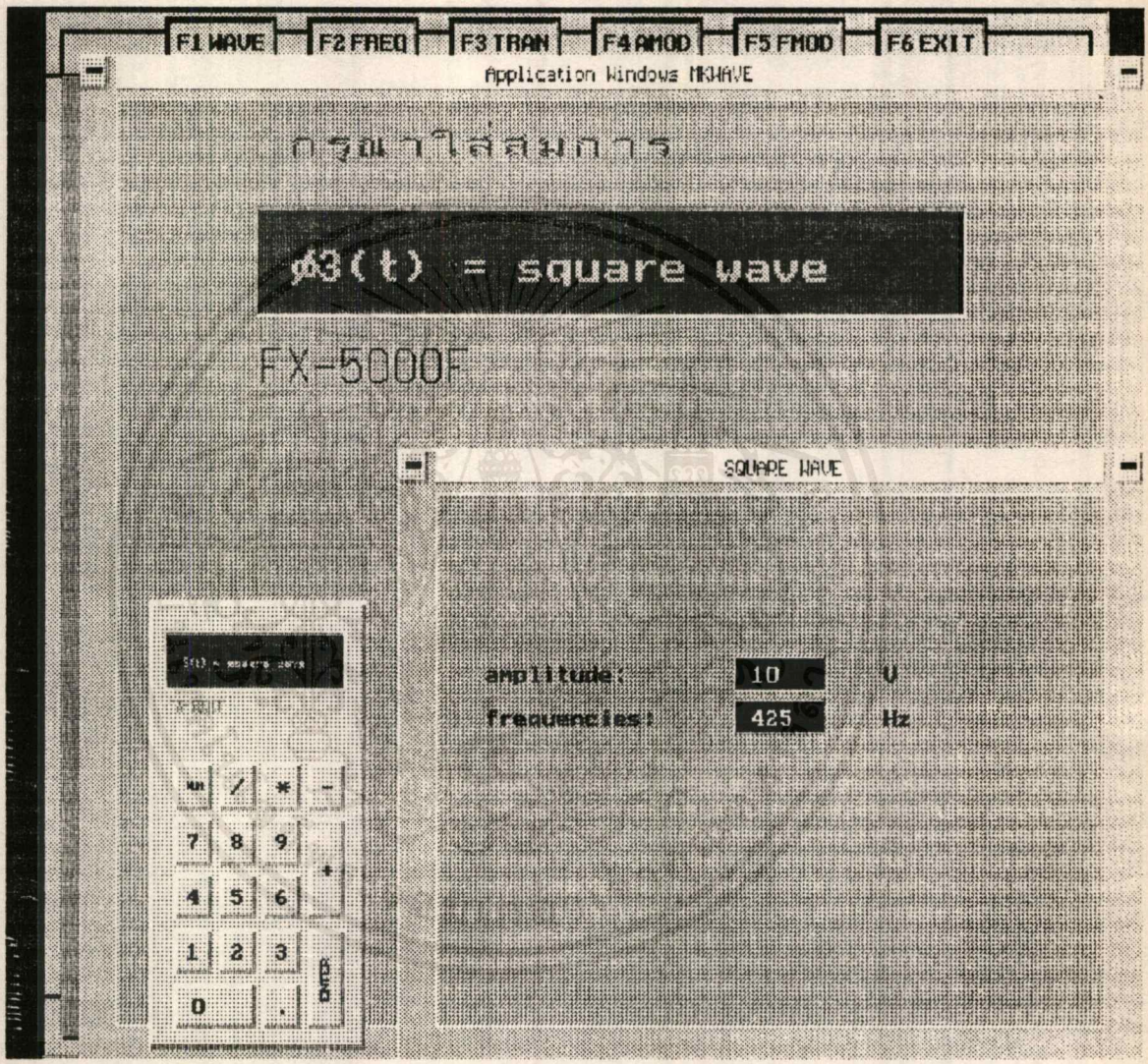


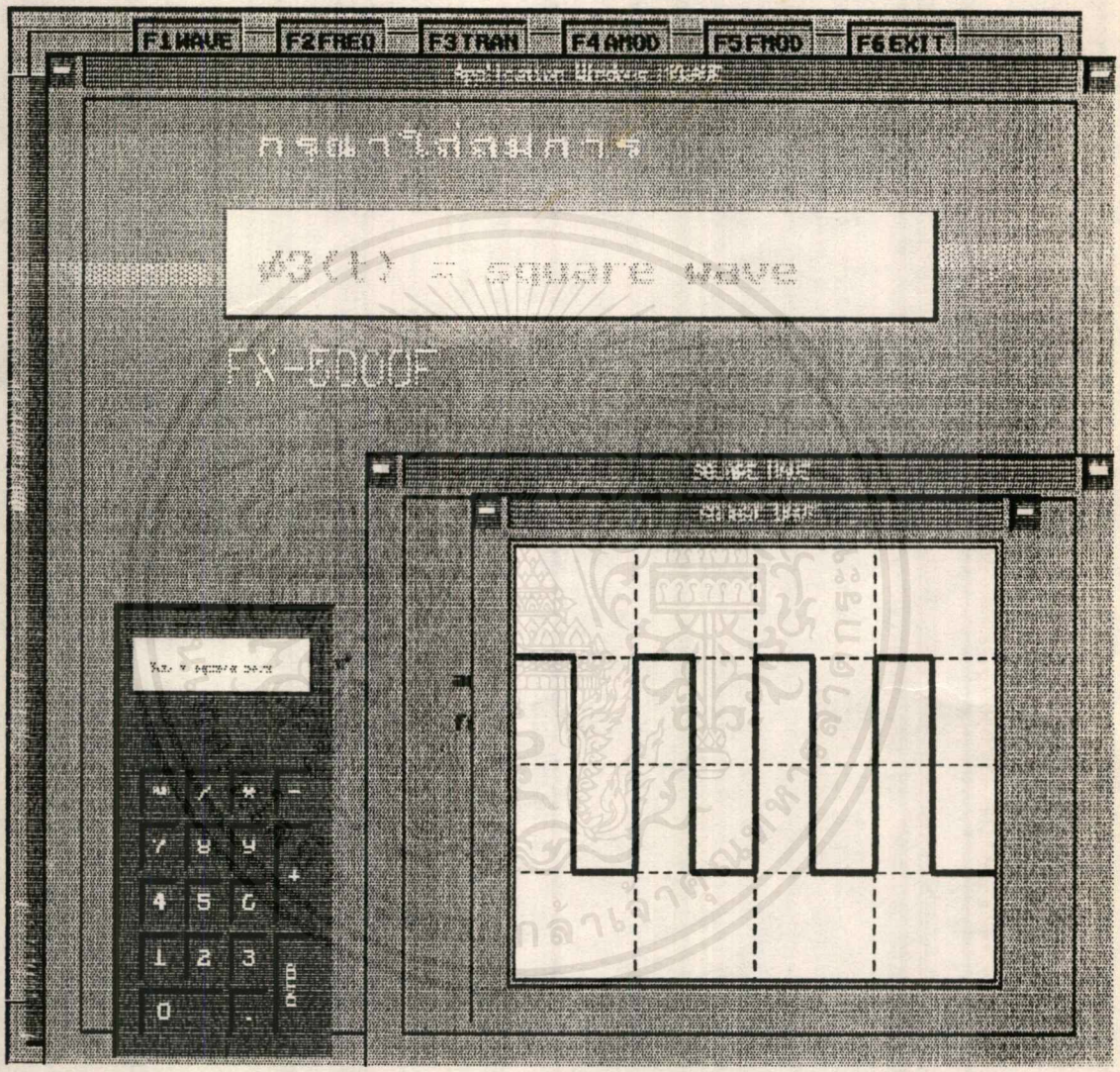


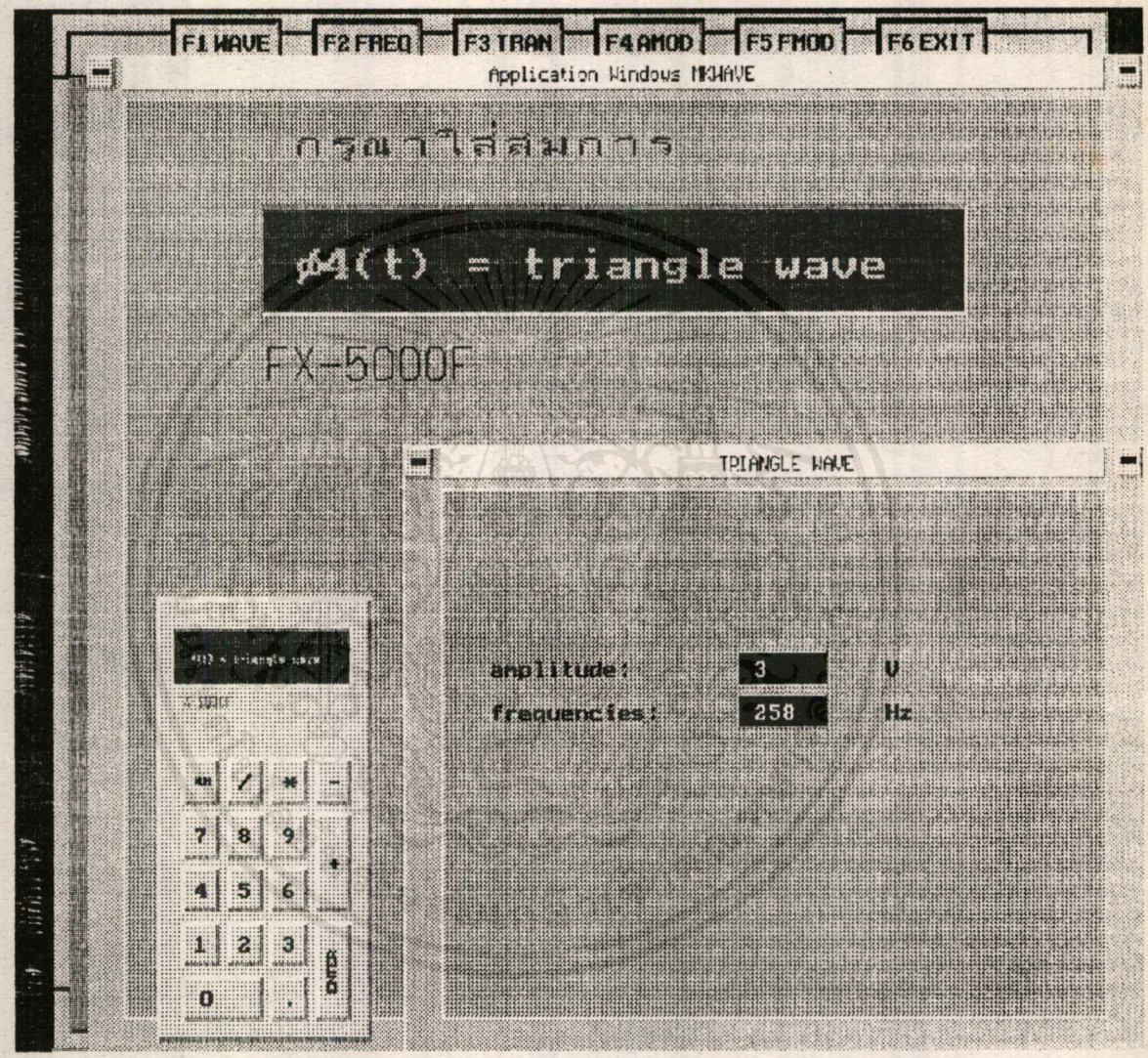


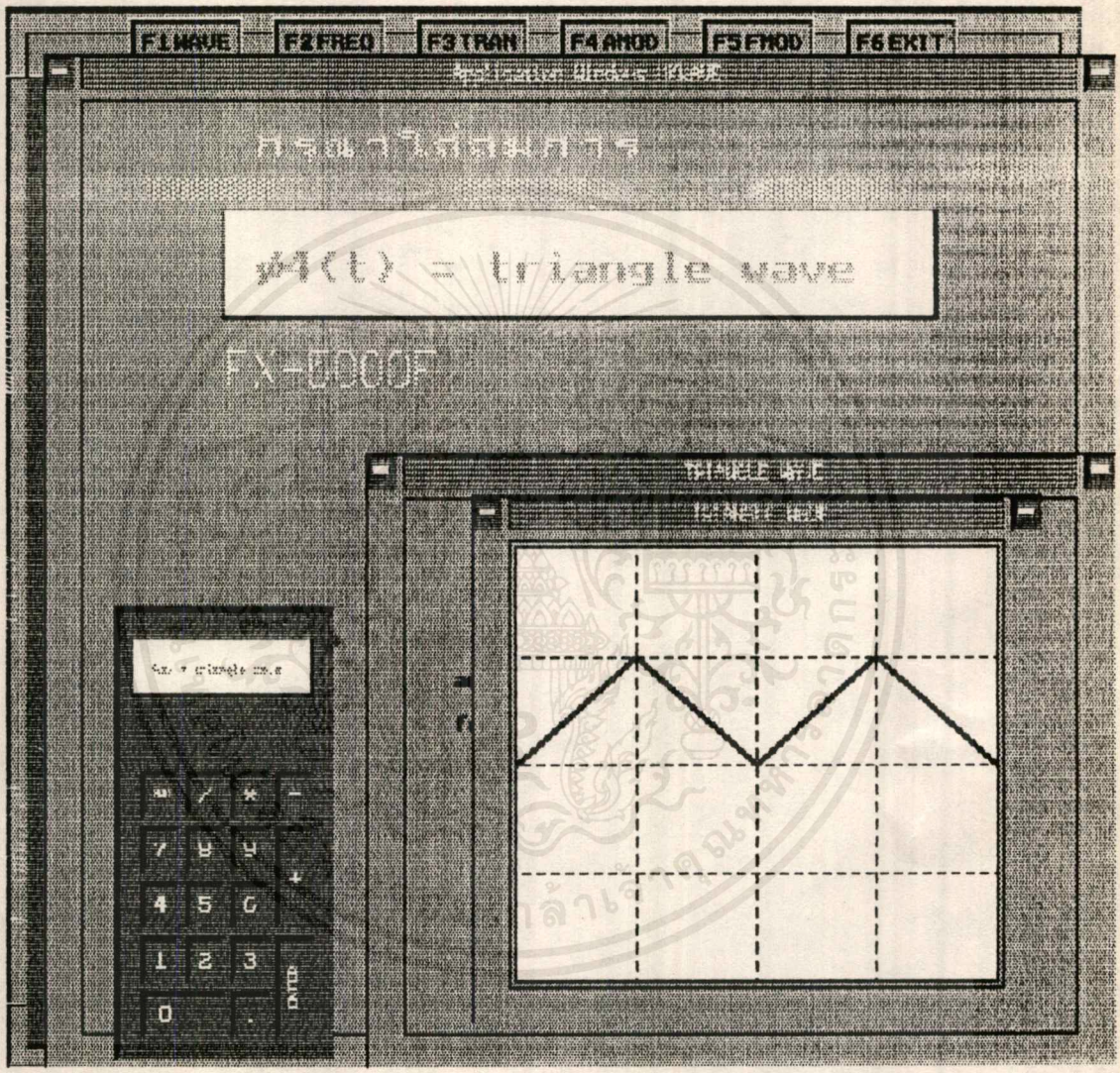


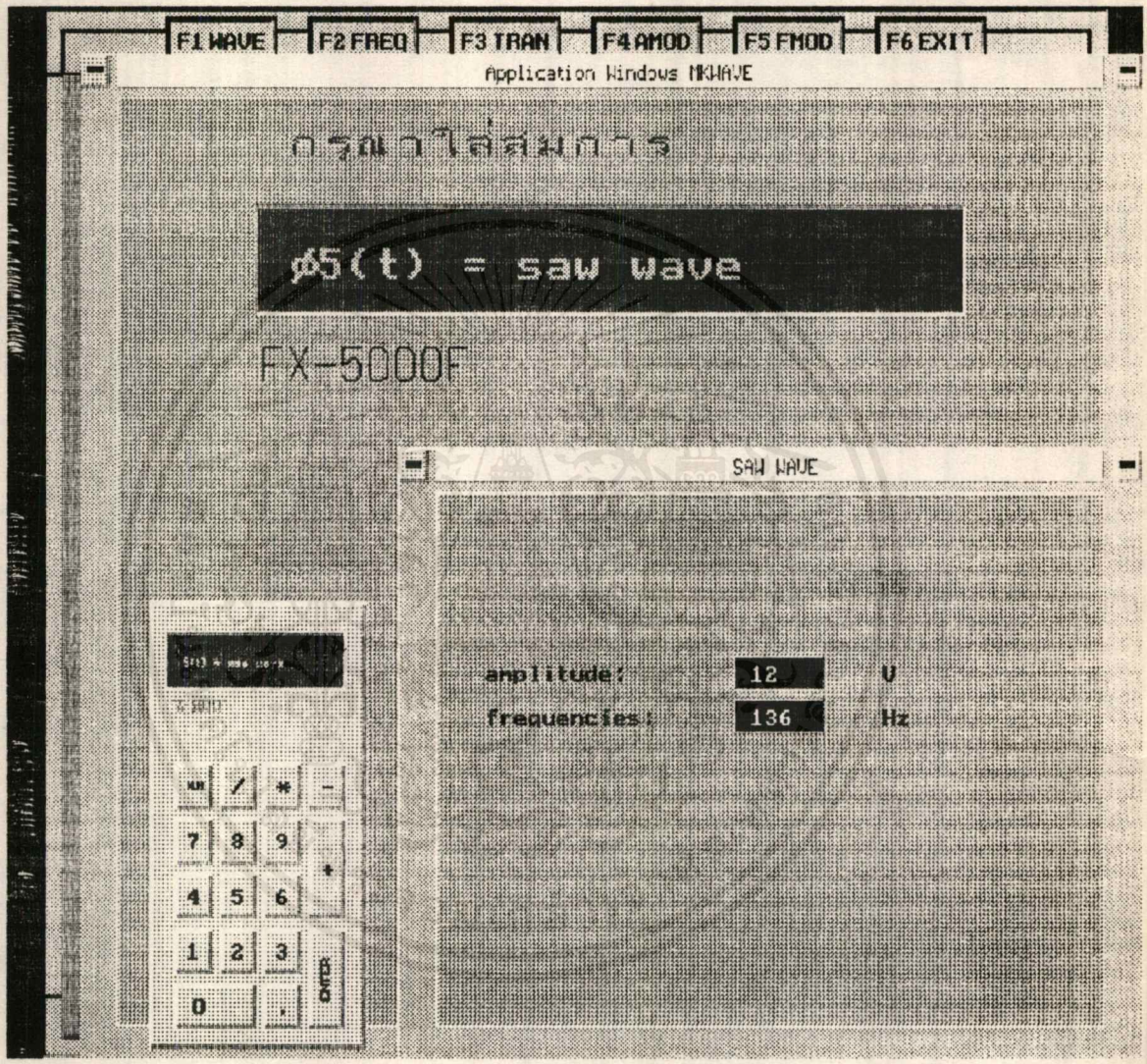


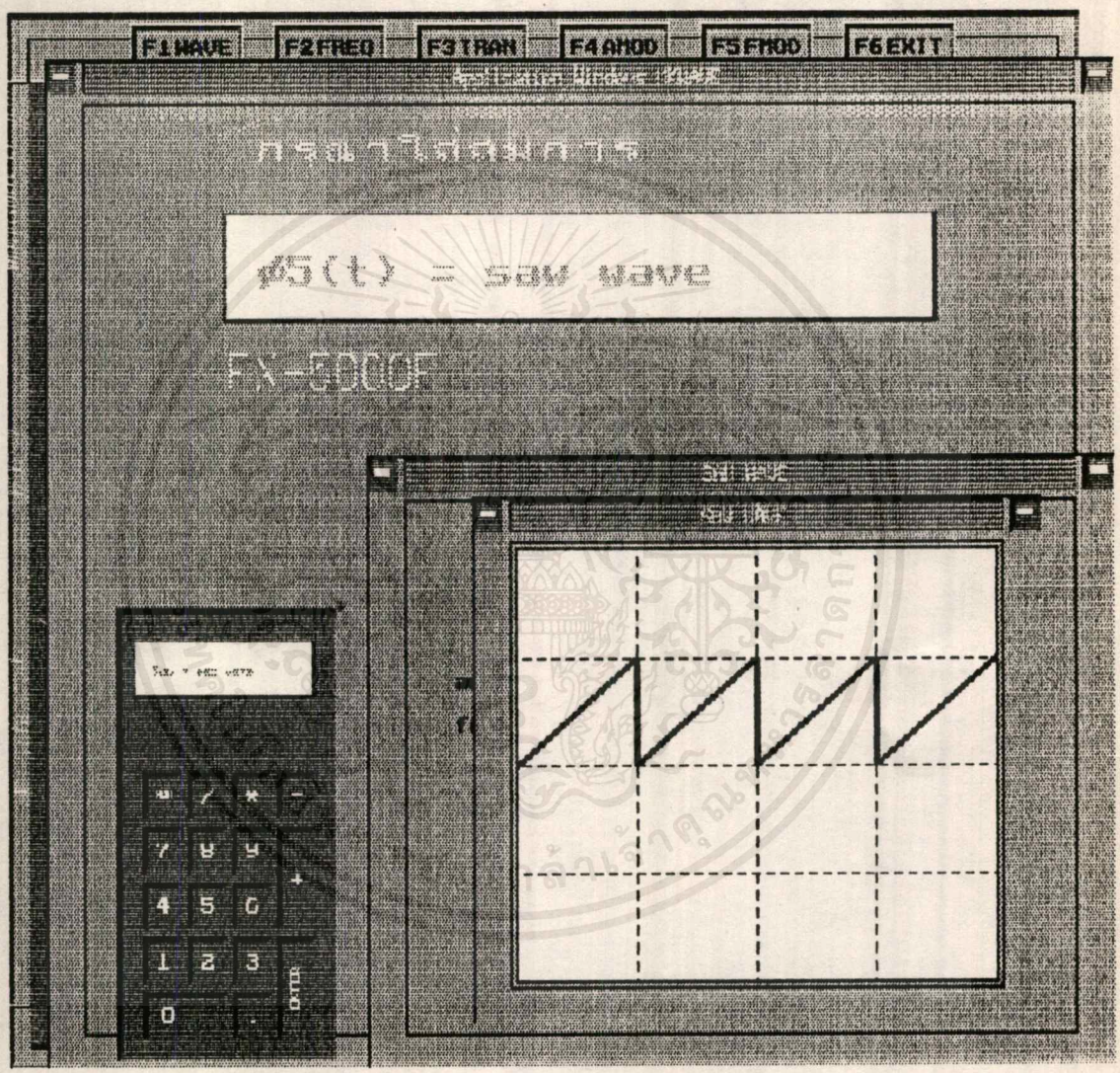


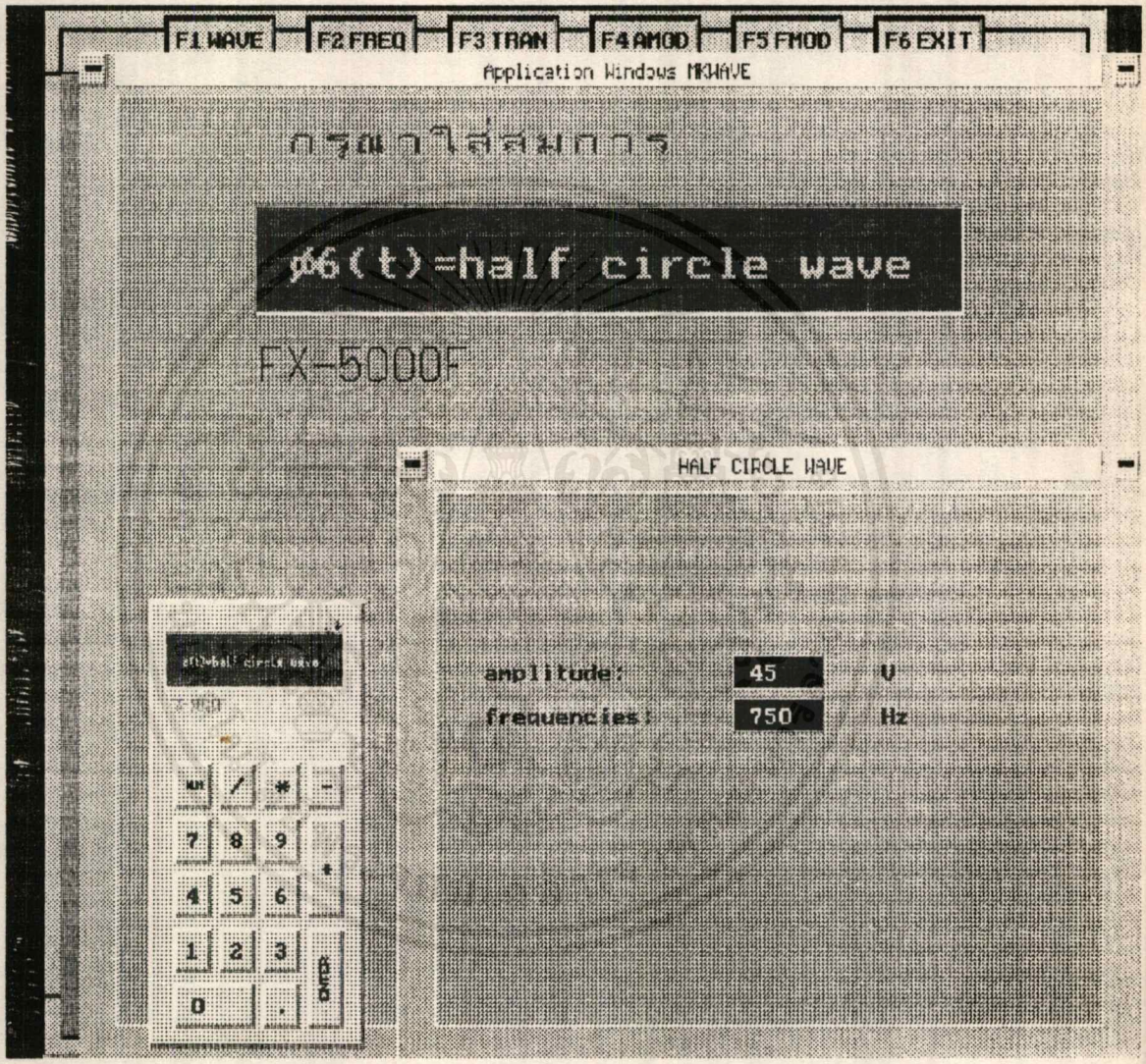


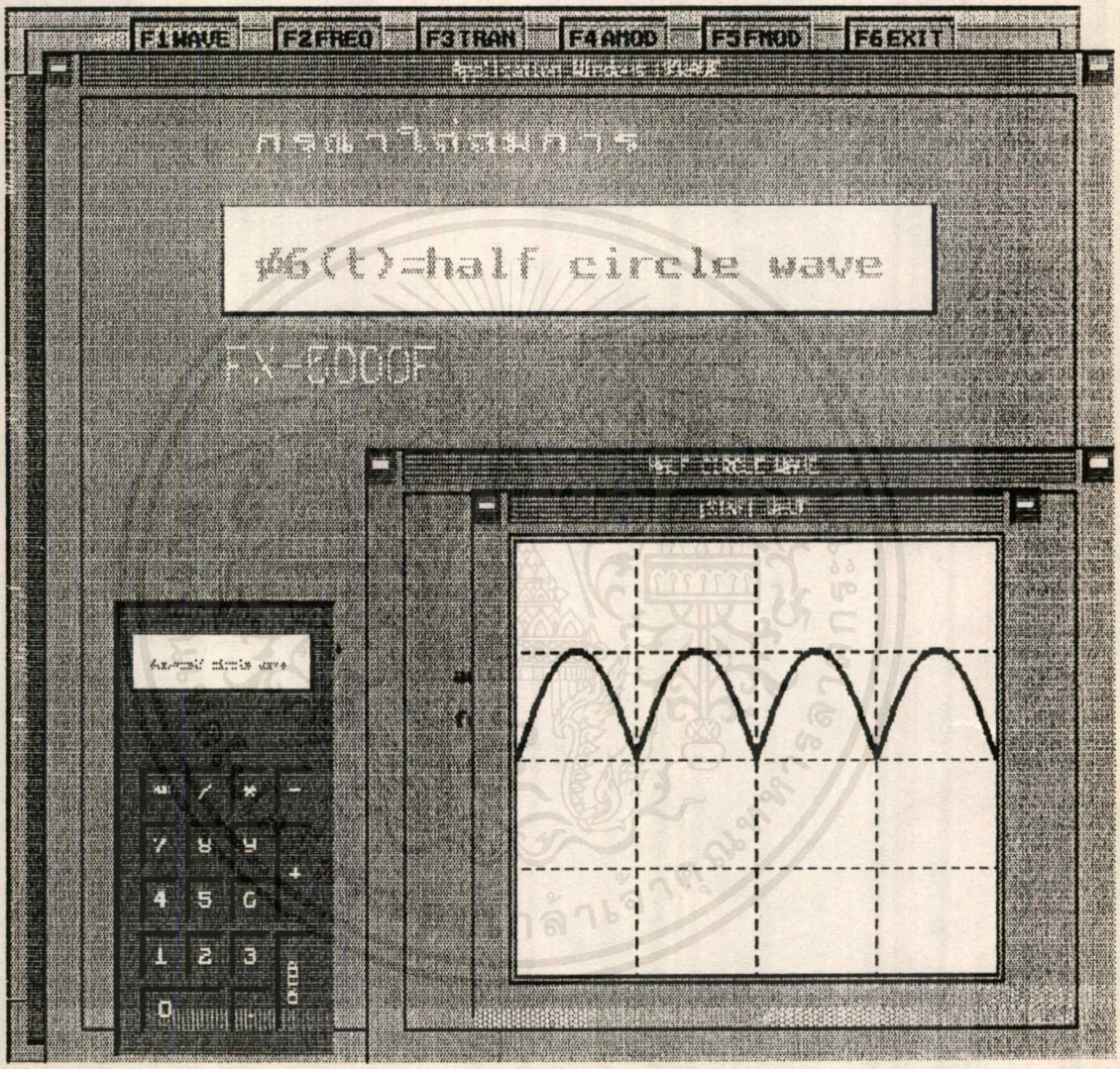


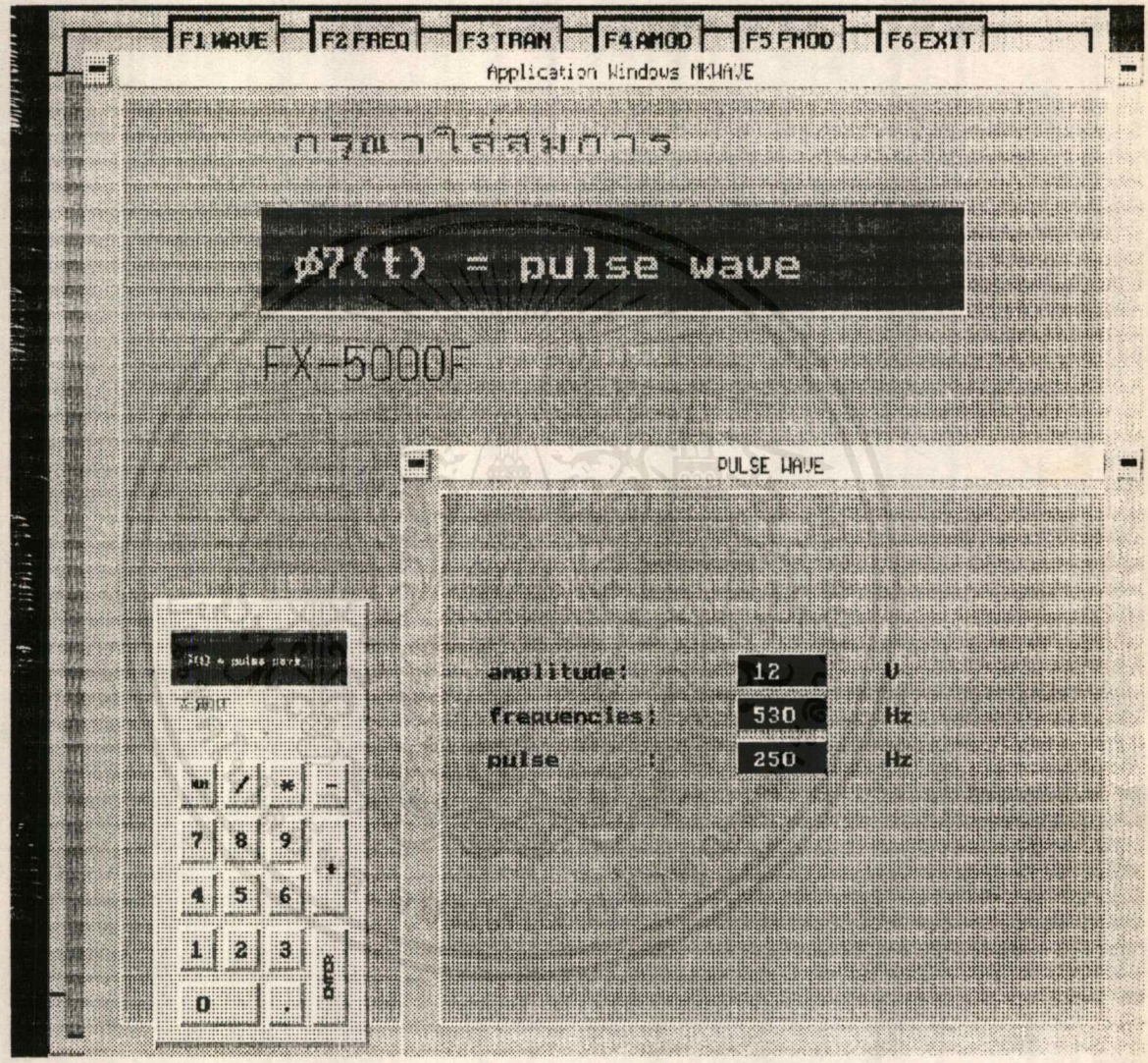


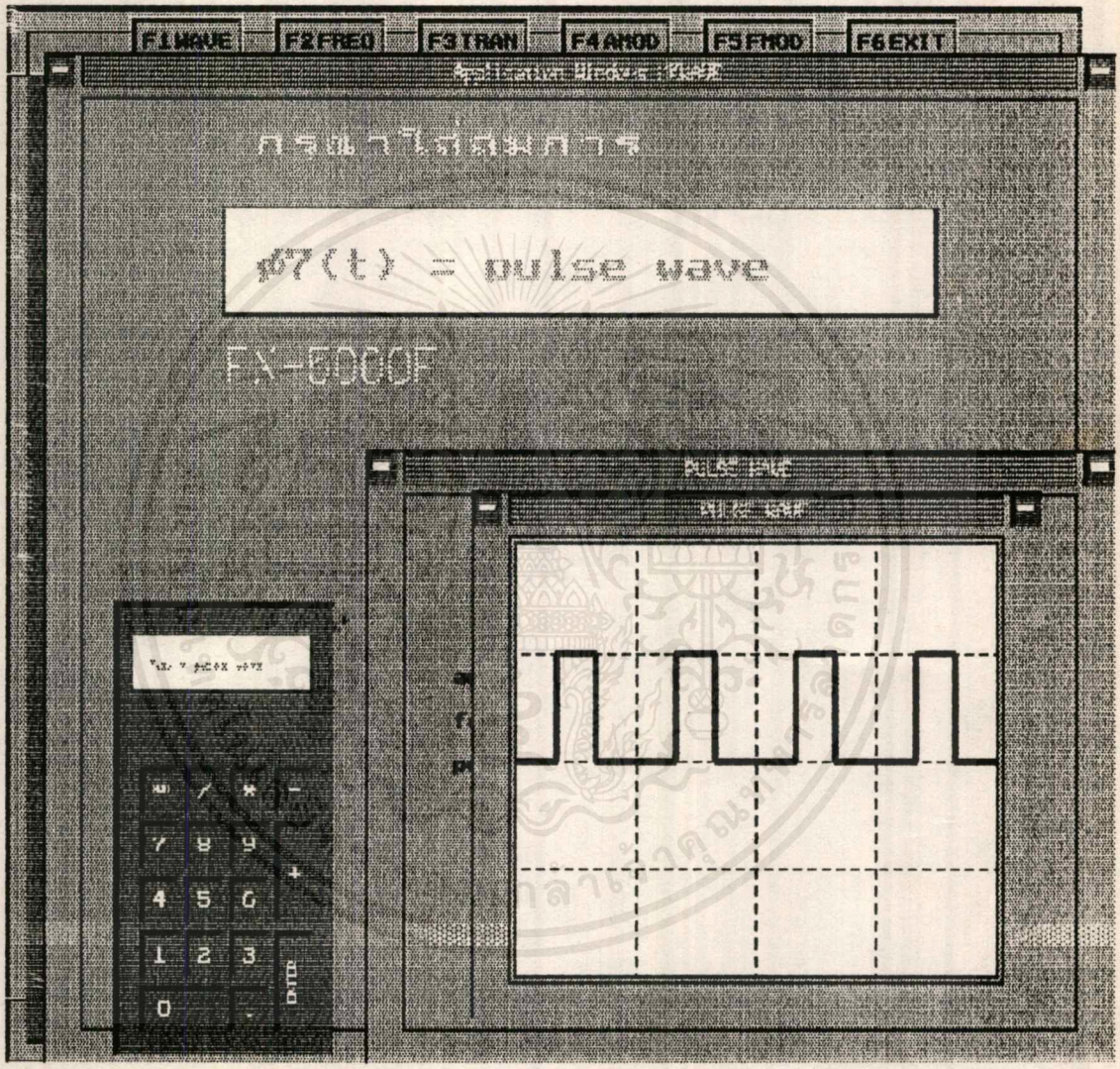


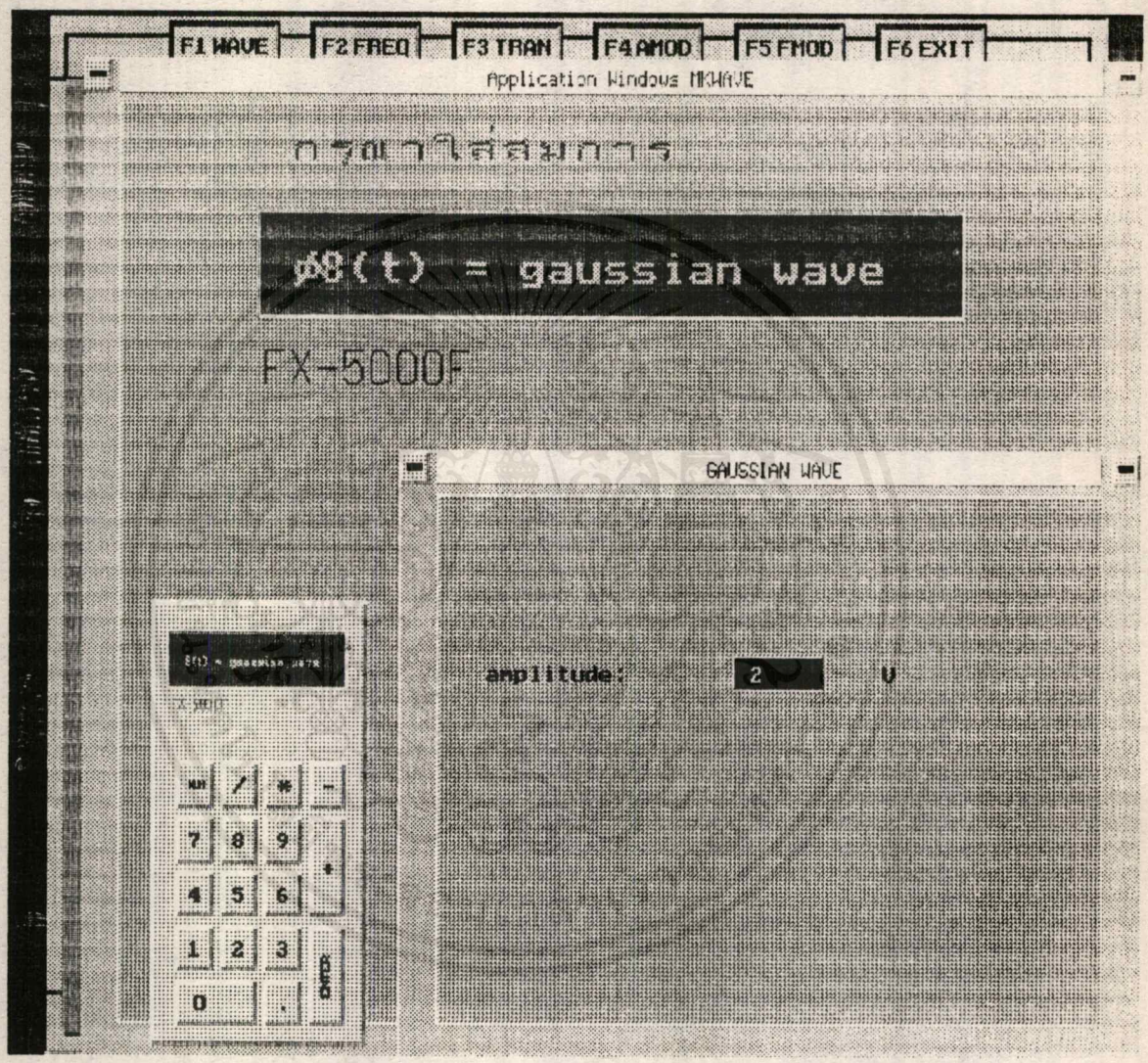


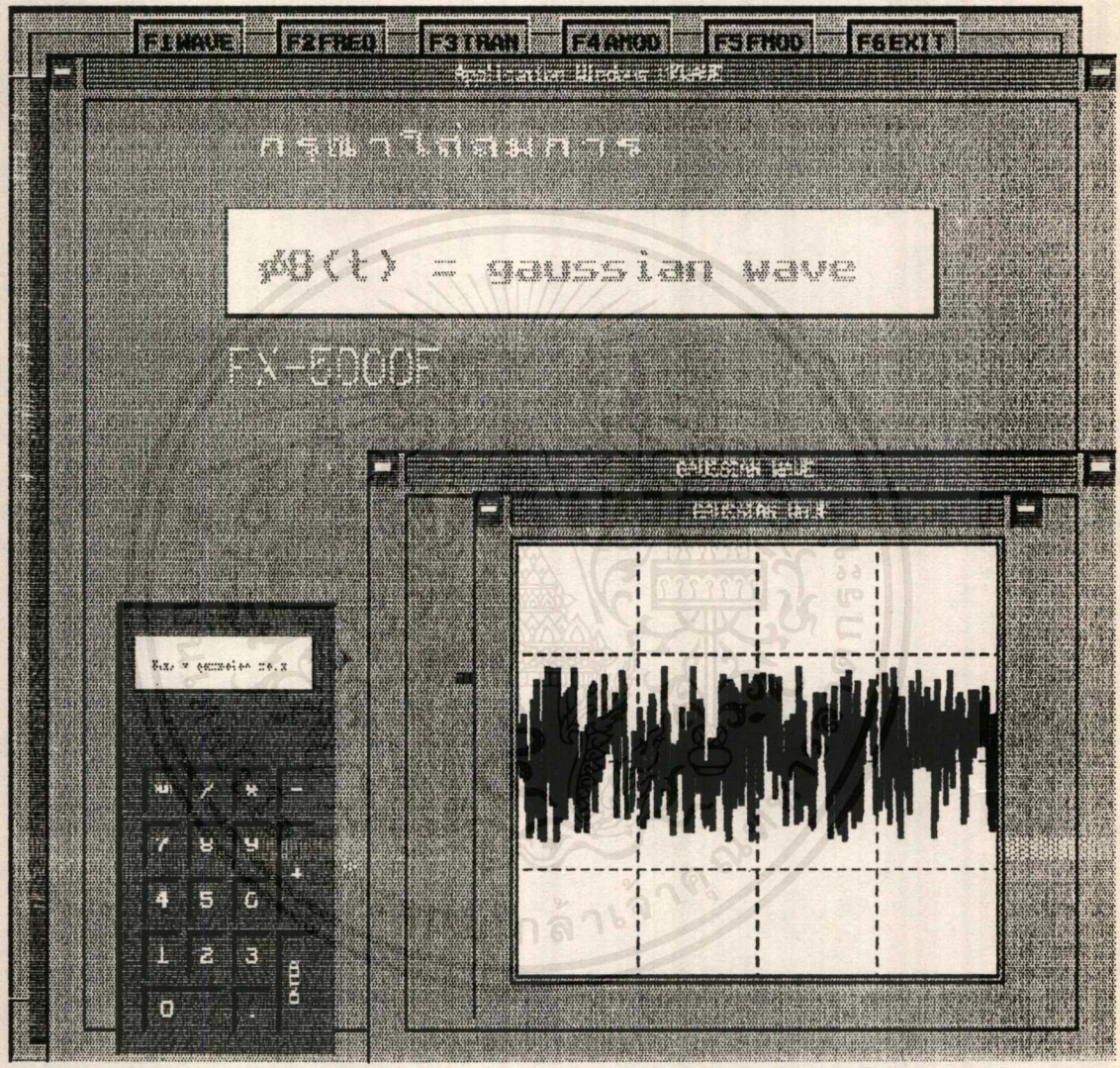




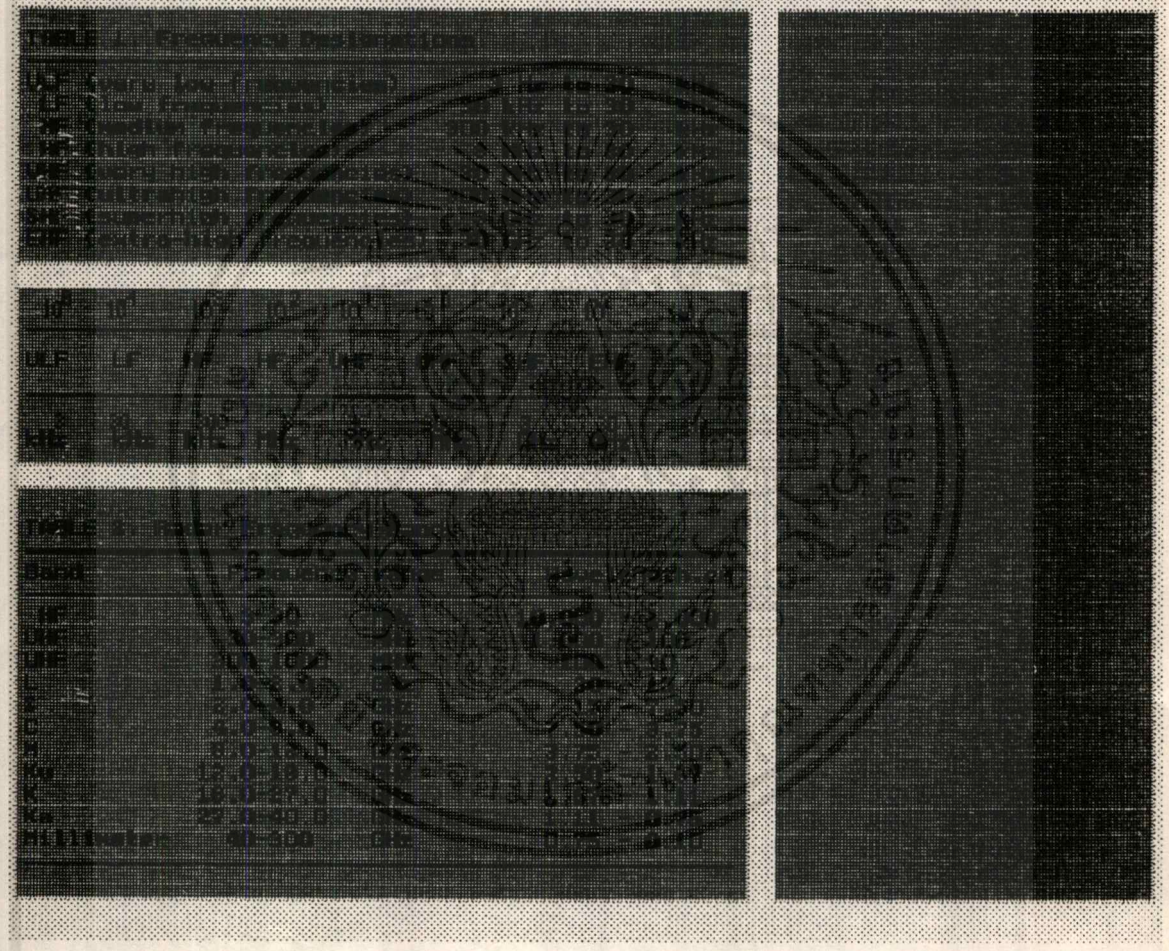








BROADCAST FREQUENCY BANDS



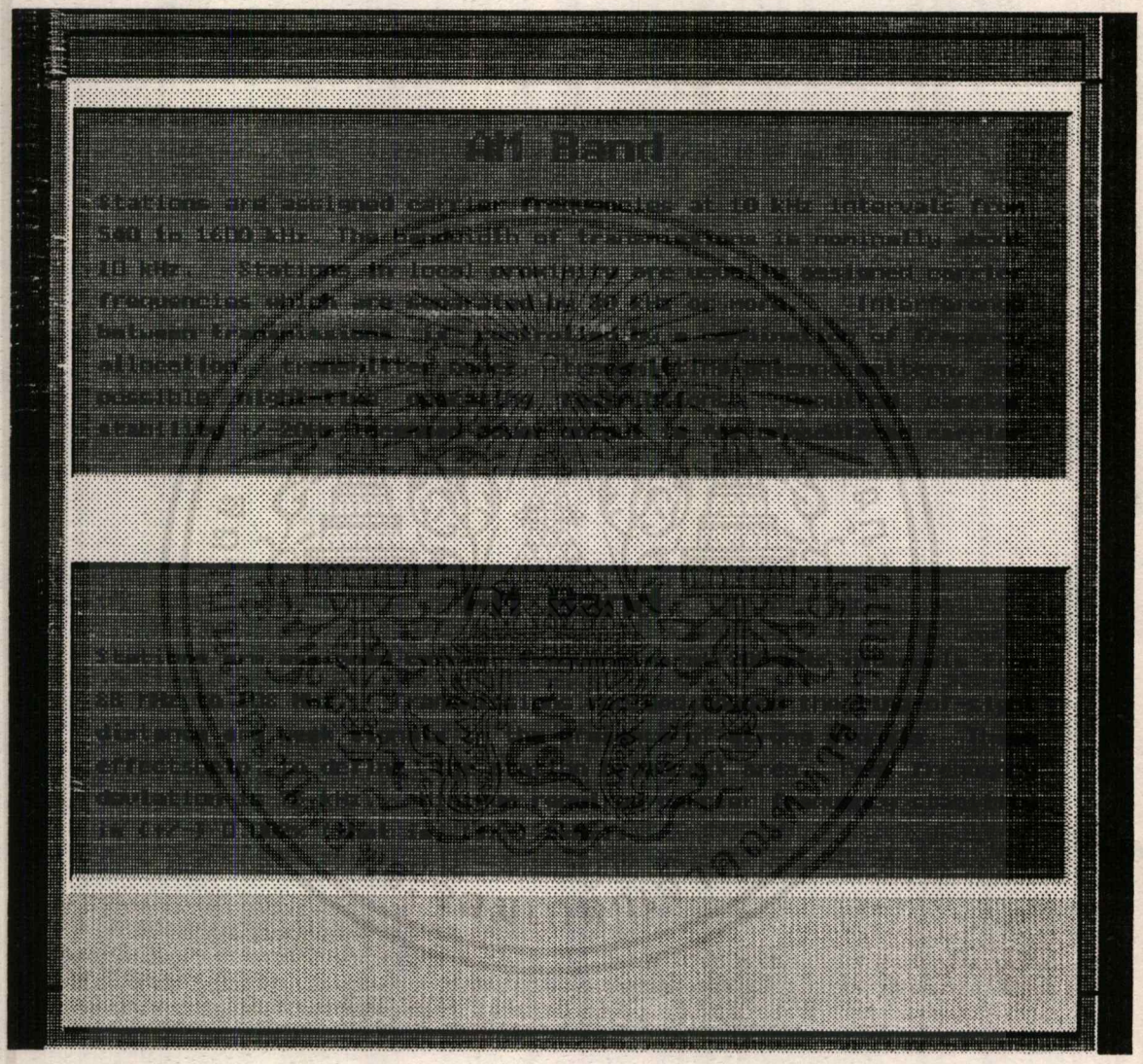


TABLE 3. VHF Television station Allocations

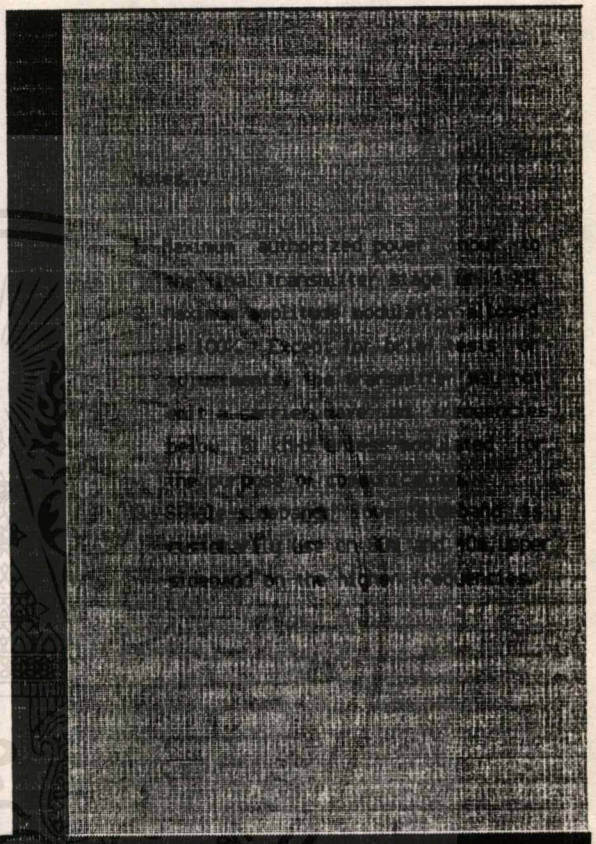
Channel number	Frequency span (MHz)	Video carrier (MHz)
1		Not used
2	54 - 60	55.25
3	60 - 66	61.25
4	66 - 76	67.25
5	76 - 82	77.25
6	82 - 88	83.25
		(Standard FM) band 88-108
7	174 - 180	175.25
8	180 - 186	181.25
9	186 - 192	187.25
10	192 - 198	193.25
11	198 - 204	199.25
12	204 - 210	205.25
13	210 - 216	211.25

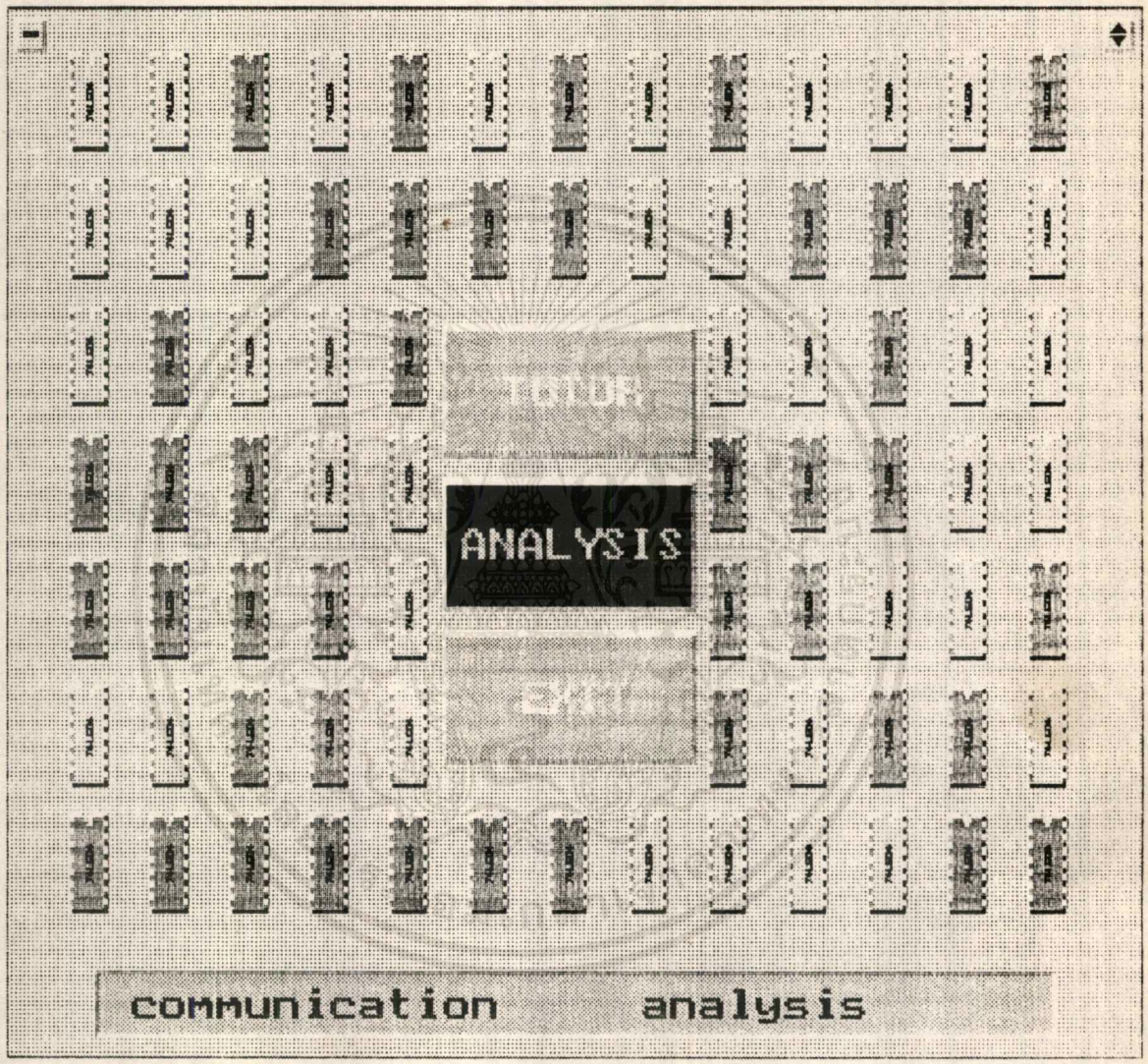
TABLE 4. UHF Television station Allocations

Channel number	Freq-span (MHz)	Video carrier (MHz)	Channel number	Freq-span (MHz)	Video carrier (MHz)
14	470-476	471.25	42	638-644	639.25
15	476-482	477.25	43	644-650	645.25
16	482-488	483.25	44	650-656	651.25
17	488-494	489.25	45	656-662	657.25
18	494-500	495.25	46	662-668	663.25
19	500-506	501.25	47	668-674	669.25
20	506-512	507.25	48	674-680	675.25
21	512-518	513.25	49	680-686	681.25
22	518-524	519.25	50	686-692	687.25
23	524-530	525.25	51	692-698	693.25
24	530-536	531.25	52	698-704	699.25
25	536-542	537.25	53	704-710	705.25
26	542-548	543.25	54	710-716	711.25
27	548-554	549.25	55	716-722	717.25
28	554-560	555.25	56	722-728	723.25
29	560-566	561.25	57	728-734	729.25
30	566-572	567.25	58	734-740	735.25
31	572-578	573.25	59	740-746	741.25
32	578-584	579.25	60	746-752	747.25
33	584-590	585.25	61	752-758	753.25
34	590-596	591.25	62	758-764	759.25
35	596-602	597.25	63	764-770	765.25
36	602-608	603.25	64	770-776	771.25
37	608-614	609.25	65	776-782	777.25
38	614-620	615.25	66	782-788	783.25
39	620-626	621.25	67	788-794	789.25
40	626-632	627.25	68	794-800	795.25
41	632-638	633.25	69	800-806	801.25

TABLE 5. Amateur Bands

Band designation	Frequency allocation(MHz)
160	1.800-2.000
80	3.500-4.000
40	7.000-7.300
20	14.000-14.350
15	21.000-21.450
10	28.000-29.700
6	50.000-54.0
2	144-148
	220-225
	420-450
	1,215-1,300
	2,300-2,450
	3,300-3,500
	5,650-5,925
	10,000-10,500
	24,000-24,500
	48,000-50,000
	71,000-76,000
	165,000-170,000
	240,000-250,000
	above 300,000





```
/*****************************************************************************
                    King Mongkut's Institute of Technology
                                KMITL
                The Communication Systems Analysis   VERSION 1.01
-----------------------------------------------------------------------------
          Software for Communication System Analysis VERSION 1.01
       Communications System are concerned with the transmittion
      of the information from a source to a sink DIGITAL & ANALOG
      communications systems. This program is require a persional
              computer which has an VGA or higher display
-----------------------------------------------------------------------------
FILE NAME :  CCAD.C
progammer  : WEERASAK  NAMVONG
Create       : 25 / Feb / 1992
Update       : 19 / may / 1992
Purpose      : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/

main()
{
        opengraph();                    /*  initial graphics mode          */
        ccadtitle();                    /*  logo TITLE window program      */
        closegraph();                    /*  retun to text mode             */
}

/*********************************<end of ccad.c>*****************************/
```

```
/*****************************************************************************
FILE NAME     : ALLKEY.H
progammer     : WEERASAK  NAMVONG
Create        : 25 / Feb / 1992
Update        : 19 / may / 1992
Purpose       : Prototype function for CCAD.EXE
Comment       : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/

#define     BS      8
#define     CR      13
#define     AQ      16
#define     AW      17
#define     AE      18
#define     AR      19
#define     AT      20
#define     AY      21
#define     AU      22
#define     AI      23
#define     AO      24
#define     AP      25
#define     ESC     27
#define     AA      30
#define     AS      31
#define     AD      32
#define     AF      33
#define     AG      34
#define     AH      35
#define     AJ      36
#define     AK      37
#define     AL      38
#define     AZ      44
#define     AX      45
#define     AC      46
#define     AV      47
#define     AB      48
#define     AN      49
#define     AM      50
#define     F1      59
#define     F2      60
#define     F3      61
#define     F4      62
#define     F5      63
#define     F6      64
#define     F7      65
#define     F8      66
#define     F9      67
#define     F10     68
#define     Home    71
#define     UP      72
#define     PgUp    73
#define     LEFT    75
```

```
#define         RIGHT       77
#define         End         79
#define         DOWN        80
#define         PgDn        81
#define         Ins         82
#define         Del         83
#define         SF1         84
#define         SF2         85
#define         SF3         86
#define         SF4         87
#define         SF5         88
#define         SF6         89
#define         SF7         90
#define         SF8         91
#define         SF9         92
#define         SF10        93
#define         CF1         94
#define         CF2         95
#define         CF3         96
#define         CF4         97
#define         CF5         98
#define         CF6         99
#define         CF7         100
#define         CF8         101
#define         CF9         102
#define         CF10        103
#define         AF1         104
#define         AF2         105
#define         AF3         106
#define         AF4         107
#define         AF5         108
#define         AF6         109
#define         AF7         110
#define         AF8         111
#define         AF9         112
#define         AF10        113
int             getkey();

/*****************************<END OF ALLKEY.H>*****************************/
```

```
/*****************************************************************************
FILE NAME : GET.H
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/

/* get.h - prototypes for get.c functions and common macros */

        extern char *get_string(char *);
        extern int get_int(char *,int,int);
        extern double get_float(char *,double,double);


/* MIN, MAX, ROUND macros */

#define MAX(a,b)    (((a) > (b)) ? (a) : (b))
#define MIN(a,b)    (((a) < (b)) ? (a) : (b))
#define ROUND(a)    (((a) < 0) ? (int)((a)-0.5) : (int)((a)+0.5))

/*****************************<END OF GET.H>*****************************/
```

```
/*****************************************************************************
FILE NAME     : DFT.H
progammer     : WEERASAK  NAMVONG
Create        : 25 / Feb / 1992
Update        : 19 / may / 1992
Purpose       : Prototype function for CCAD.EXE
Comment       : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/

/* dft.h - function prototypes and structures for dft and fft functions */

/* COMPLEX STRUCTURE */

typedef struct {
        float real, imag;
} COMPLEX;

/* function prototypes for dft and inverse dft functions */

        extern void fft(COMPLEX *,int);
        extern void ifft(COMPLEX *,int);
        extern void dft(COMPLEX *,COMPLEX *,int);
        extern void idft(COMPLEX *,COMPLEX *,int);
        extern void rfft(float *,COMPLEX *,int);
        extern void ham(COMPLEX *,int);
        extern void han(COMPLEX *,int);
        extern void triang(COMPLEX *,int);
        extern void black(COMPLEX *,int);
        extern void harris(COMPLEX *,int);
        extern int log2(unsigned int);

/*****************************<END OF DFT.H>*****************************/
```

```
/*****************************************************************************
FILE NAME     : DISK.H
progammer     : WEERASAK  NAMVONG
Create        : 25 / Feb / 1992
Update        : 19 / may / 1992
Purpose       : Prototype function for CCAD.EXE
Comment       : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/

/* disk.h - function prototypes and structures for diskio.c functions */

/* DSP INFORMATION STRUCTURE FOR DSP DATA FILES */

typedef struct {
        unsigned char type;             /* data type 0-7 as defined below      */
        unsigned char element_size;     /* size of each element                */
        unsigned short int records;     /* number of records                   */
        unsigned short int rec_len;     /* number of elements in each record   */
        char *name;                     /* pointer to file name                */
        FILE *fp;                       /* pointer to FILE structure           */
} DSP_FILE;

/* defines for data type used in DSP data file header and structure */

#define UNSIGNED_CHAR   0
#define UNSIGNED_INT    1
#define UNSIGNED_LONG   2
#define FLOAT           3
#define SIGNED_CHAR     4
#define SIGNED_INT      5
#define SIGNED_LONG     6
#define DOUBLE          7

/* function prototypes for DSP disk file functions */

  extern DSP_FILE *open_read(char *);
  extern DSP_FILE *open_write(char *,int,int,int);
  extern void read_record(char *,DSP_FILE *);
  extern void write_record(char *,DSP_FILE *);
  extern void seek_record(int,DSP_FILE *);
  extern float *read_float_record(DSP_FILE *);
  extern char *read_trailer(DSP_FILE *);
  extern char *append_trailer(char *,DSP_FILE *);
  extern int write_trailer(char *,DSP_FILE *);

/*****************************<END OF DISK.H>*****************************/
```

```
/*****************************************************************************
FILE NAME : INITIAL.C
progammer  : WEERASAK NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang

*****************************************************************************
   Graphics Library for CCAD.C Its softwave for communication Systems
   Analysis                  <- initial.c ->
*****************************************************************************
-----------------------------------------------------------------------------
| bgiobj cga        | bgiobj trip     | tlib graphics +cga +egavga  |
| bgiobj egavga     | bgiobj litt     | +herc +pc3270 +ibm8514      |
| bgiobj herc       | bgiobj sans     | +trip +litt +sans +goth     |
| bgiobj att        | bgiobj goth     |                             |
| bgiobj pc3270     |                 |                             |
| bgiobj ibm8514    |                 |                             |
-----------------------------------------------------------------------------
*****************************************************************************/
#include <graphics.h>
#include <stdlib.h >
#include <stdio.h  >
#include <stdarg.h >
#include <conio.h  >
#include <ctype.h  >
#include <dos.h    >
#include <alloc.h  >
#include <string.h >

int     GraphDriver = DETECT, GraphMode, ErrorCode = 0;
void Initialize()
{
        initgraph(&GraphDriver, &GraphMode, " ");
        ErrorCode = graphresult();
        if(ErrorCode != grOk)
        {
                printf(" Graphics System Error: %s\n",
                grapherrormsg(ErrorCode));
                exit(1);
        }
}

void RegisterDriverF()
{
        if(registerfarbgidriver(CGA_driver_far    ) < 0)      exit(1);
        if(registerfarbgidriver(EGAVGA_driver_far ) < 0)      exit(1);
        if(registerfarbgidriver(Herc_driver_far   ) < 0)      exit(1);
        if(registerfarbgidriver(ATT_driver_far    ) < 0)      exit(1);
        if(registerfarbgidriver(PC3270_driver_far ) < 0)      exit(1);
        if(registerfarbgidriver(IBM8514_driver_far ) < 0)     exit(1);
```

```
}

void RegisterFontF()
{
        if(registerfarbgifont(triplex_font_far    ) < 0) exit(1);
        if(registerfarbgifont(small_font_far    ) < 0)  exit(1);
        if(registerfarbgifont(sansserif_font_far  ) < 0)          exit(1);
        if(registerfarbgifont(gothic_font_far   ) < 0)  exit(1);
}

void opengraph()
{
        RegisterDriverF();
        RegisterFontF();
        Initialize();
}

void Pause()
{
        if(kbhit) getch();
        getch();
}

void speaker(int f,int t)
{
        sound(f);
        delay(t);
        nosound();
}

void bbx0(int sx,int sy,int ex,int ey,int c_tl,int c_br,int c_mid)
{
        setfillstyle(1,c_mid);
        bar(sx,sy,ex,ey);
        setfillstyle(1,c_tl);
        bar(sx,sy,sx,ey);
        bar(sx,sy,ex,sy);
        setfillstyle(1,c_br);
        bar(sx,ey,ex,ey);
        bar(ex,sy,ex,ey);
        setcolor(0);
}

void bbx1(int sx,int sy,int ex,int ey,int c_tl,int c_br,int c_mid)
{
        setfillstyle(1,c_mid);
        bar(sx,sy,ex,ey);
        setfillstyle(1,c_tl);
        bar(sx,sy,sx+1,ey);
        bar(sx,sy,ex,sy+1);
        setfillstyle(1,c_br);
        bar(sx+1,ey-1,ex,ey);
        bar(ex-1,sy+1,ex,ey);
        setcolor(0);
```

```
}

void bbx(int sx,int sy,int ex,int ey,int c_tl,int c_br,int c_mid)
{
        setfillstyle(1,c_mid);
        bar(sx,sy,ex,ey);
        setfillstyle(1,c_tl);
        bar(sx,sy,sx+2,ey);
        bar(sx,sy,ex,sy+2);
        setfillstyle(1,c_br);
        bar(sx+2,ey-2,ex,ey);
        bar(ex-2,sy+2,ex,ey);
        setcolor(0);
}

void bbxtri(int sx,int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+2+n,sx+8+n,sy+2+n,0,0,0);
                bbx0(sx+8-n,sy+12-n,sx+8+n,sy+12-n,0,0,0);
        }
}

void bbxtrii(int sx,int sy)
{
        int n;
        bbx0(sx,sy,sx+16,sy+15,8,15,14);
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+2+n,sx+8+n,sy+2+n,0,0,0);
                bbx0(sx+8-n,sy+12-n,sx+8+n,sy+12-n,0,0,0);
        }
}

void bbxesc(int sx,int sy)
{
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        bbx1(sx+4,sy+6,sx+12,sy+8,0,0,0);
}

void bbxesci(int sx,int sy)
{
        bbx0(sx,sy,sx+16,sy+15,8,15,14);
        bbx1(sx+4,sy+6,sx+12,sy+8,0,0,0);
}

upkey(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        sx -= 1;   sy -=1;
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+5+n,sx+8+n,sy+5+n,0,0,0);
```

```
        }
}

downkey(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        sx -= 1;   sy -=1;
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+10-n,sx+8+n,sy+10-n,0,0,0);
        }
}

leftkey(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        sx -= 1;   sy -=1;
        for(n=0;n<=4;n++) {
                bbx0(sx+5+n,sy+8-n,sx+5+n,sy+8+n,0,0,0);
        }
}

rightkey(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,15,8,7);
        sx -= 1;   sy -=1;
        for(n=0;n<=4;n++) {
                bbx0(sx+11-n,sy+8-n,sx+11-n,sy+8+n,0,0,0);
        }
}

upkeyi(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,8,15,14);
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+5+n,sx+8+n,sy+5+n,4,4,4);
        }
}

downkeyi(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,8,15,14);
        for(n=0;n<=4;n++) {
                bbx0(sx+8-n,sy+10-n,sx+8+n,sy+10-n,4,4,4);
        }
}


leftkeyi(int sx, int sy)
{
```

```
        int n;
        bbx1(sx,sy,sx+16,sy+15,8,15,14);
        for(n=0;n<=4;n++) {
                bbx0(sx+5+n,sy+8-n,sx+5+n,sy+8+n,4,4,4);
        }
}

rightkeyi(int sx, int sy)
{
        int n;
        bbx1(sx,sy,sx+16,sy+15,8,15,14);
        for(n=0;n<=4;n++) {
                bbx0(sx+11-n,sy+8-n,sx+11-n,sy+8+n,4,4,4);
        }
}

writetext(int sx,int sy,int color,char font,int direct,
        int size,char *text)
{
        settextstyle(font,direct,size);
        setcolor(color);
        outtextxy(sx,sy,text);
}

writetextxy(int xloc, int yloc, int color, char font, int direct,
                int size, char *fmt, ...)
{
        va_list argptr;
        char str[140];
        struct textsettingstype textinfo;
        va_start(argptr, format);
        vsprintf(str, fmt, argptr);
        gettextsettings(&textinfo);
        settextstyle(font,direct,size);
        setcolor(color);
        outtextxy(xloc, yloc, str);
        va_end(argptr);
}

void special(int t)
{
        int n;
        for(n = 1; n <= t; n++) {
                sound(random(3000));
                sound(random(5000));
                sound(random(1000));
        }
        nosound();
}

/*************************<end of initial.i>*****************************/
```

```
/*****************************************************************************
FILE NAME : thai.c
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#include <ctype.h>

void fontthai(void);
int  byte_font[5120];

F1(int x,int y,int e,int color)
{
        int     a[9] = { 128, 64, 32, 16, 8, 4, 2, 1, 0 };
        int     m=0,i, k, n;
        while( m < 20)
        {
                k = m + e*20;
                for(i=0;i<=7;i++)
                {
                        n = ( a[i] & byte_font[k] );
                        if(n != 0)
                        {
                                putpixel(x+i,y+m,color);
                        }
                }
                m++;
        }
}

F2(int x,int y,int e,int color)
{
        int     a[9] = { 128, 64, 32, 16, 8, 4, 2, 1, 0 };
        int     m = 0, i, k, n;
        while(m < 20)
        {
                k = m + e*20;
                for(i = 0;i <= 7; i++)
                {
                        n = ( a[i] & byte_font[k] );
                        if(n != 0)
                        {
                                putpixel(x+2*i-1,y+m,color);
                                putpixel(x+2*i,y+m,color);
```

```
                        }
                }
                m++;
        }
}

F3(int x,int y,int e,int color)
{
        int     a[9] = { 128, 64, 32, 16, 8, 4, 2, 1, 0 };
        int     m = 0, i, k, n;
        while(m < 20)
        {
                k = m + e*20;
                for(i = 0;i <= 7; i++)
                {
                        n = ( a[i] & byte_font[k] );
                        if(n != 0)
                        {
                                putpixel(x+3*i-2,y+2*m,color);
                                putpixel(x+3*i-1,y+2*m,color);
                                putpixel(x+3*i,y+2*m,color);
                                putpixel(x+3*i+1,y+2*m,color);
                                putpixel(x+3*i+2,y+2*m,color);
                                putpixel(x+3*i-2,y+2*m+1,color);
                                putpixel(x+3*i-1,y+2*m+1,color);
                                putpixel(x+3*i,y+2*m+1,color);
                                putpixel(x+3*i+1,y+2*m+1,color);
                                putpixel(x+3*i+2,y+2*m+1,color);
                        }
                }
                m++;
        }
}

void fontthai(void)
{
        FILE  *infile;
        int   pos = 0;
        int   ch;

        infile = fopen("CCAD.OVL","rb");
        if (!infile)
        {
                writetext(20,20,4,0,0,2,"Can't Open file CCAD.OVL");
                sleep(1);
                closegraph();
                exit(1);
        }
        rewind(infile);
        while (pos++ <=5120)
        {
                ch = getc(infile);
                byte_font[pos] = ch;
        }
```

```
        fcloseall();
}

writethai(int sx,int sy,int color,int size,char *text)
{
        int i,j,k;
        i = j = 0;
        while(text[i] != '\0')
        {
                k = text[i]+256;
                if(k > 256)
                        k = toascii(text[i]);
                if((k==209)||(k>=211 && k<=219)||(k>=231 && k<=238))
                {
                        j = j;
                }
                else
                {
                        j+=10;
                }
                switch(size) {
                        case 1 : F1(sx+j,sy,k,color);   break;
                        case 2 : F2(sx+2*j,sy,k,color);  break;
                        case 3 : F3(sx+3*j,sy,k,color);  break;
                }
                i++;
        }
}
/*******************************<end of thai.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : SONG.C
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include      "DOS.H"
#define C     262
#define D     294
#define E     330
#define F     350
#define G     392
#define A     440
#define B     467
#define CH    524
#define DH    588
#define EH    660
#define FH    700
#define GH    785
#define AH    880
#define BH    934
#define REST  25000
#define QQE   937.5
#define Q     375
#define EN    187.5
#define ES    287.25
#define S     93.75
#define EP    187.5
#define QP    375
#define BLANK      5
#define MAXNOTE    109
#define speed      0.866

void kmitsong(void);
void kmitsong(void)
{
        int     i = 0;
int melody[] = {
A,DH,CH,A,G,F,D,G,F,D,C,A,DH,CH,G,A,DH,CH,FH,DH,A,CH,F,G,A,G,
C,A,G,F,D,C,F,G,A,G,A,CH,G,A,G,A,DH,CH,A,REST,A,G,D,F,
B,CH,B,CH,REST,CH,REST,DH,CH,B,CH,A,G,F,REST,G,REST,G,REST,G,C,G,CH,A,G,E,D,E,G,E,D,C
REST,
A,DH,CH,A,G,F,D,G,F,D,C,A,DH,CH,G,A,DH,CH,FH,DH,A,CH,A,G,D,F
};
int del[] = {
EN,EN,QQE,EN,EN,EN,EN,EN,EN,EN,Q,EN,EN,ES,S,EN,EN,EN,EN,EN,EN,EN,EN,EN,Q,
EN,EN,QQE,EN,EN,EN,ES,S,ES,S,Q,EN,EN,ES,S,EN,EN,EN,S,BLANK,S,EN,EN,QQE,
ES,S,EN,Q,20,Q,EP,ES,S,EN,EN,EN,EN,EN,EP,S,BLANK,S,BLANK,EN,EN,EN,EN,EN,EN,ES,S,EN,EN
,EN,EN,Q,QP,
EN,EN,QQE,EN,EN,EN,EN,EN,EN,EN,Q,EN,EN,ES,S,EN,EN,EN,EN,EN,EN,EN,EN,EN,EN,QQE
```

```
};
        while(!bioskey(1)){
                sound(melody[i]);
                delay(del[i]*speed);
                if(i < MAXNOTE) i++;
                if(i == MAXNOTE){
                nosound();
                i = 0;
                }
        }
        nosound();
        bioskey(0);
}
/*********************************<END OF SONG.C>*****************************/
```

```
/*****************************************************************************
FILE NAME : title.c
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <stdlib.h>

char    *welcome[37] = {
        "W","e","l","c","o","m","e"," ","t","o"," ","A","n","a","l","y",
        "s","i","s"," ","S","y","s","t","e","m"," ","P","r","o","g","r",
        "a","m","!","!","!"
};
char    *kmit[2] = {"-¦-¦-¦_+ñ_¦_+-+++-¿--_í+_-","+-¦í+-¦-º"};
char    *hotel[6] = {"¦++-","_¦+--_+ñ-",
                     "¦++-+-í¦+_","¦--¦º",
                     "++-¦++","ßí_¦í-¦_¦¦+"
};
char    *kisdakorn[2] = { "--¿-+-_í-¦-í+","í+_--í-+" };
char    *project[2] = { "½-++_ß¦+_¦+_ñ+---_+-¦¦-+_---+",
                        "SOFTWARE FOR COMMUNICATION SYSTEM ANALYSIS"
};
char    *engineering[] = {"ñ¦-¦++¦í++-+--¦+_"};
int     posx=50, posy=460, count;

void ccadtitle(void)
{
        int x1,x2;
        fontthai();
        bbx0(0,0,639,479,11,11,9);
        bbxesc(5,5);        bbxtri(618,5);
        bbx1(50,25,590,170,15,8,7);
        writethai(226,51,8,2,kmit[1]);
        writethai(225,50,14,2,kmit[1]);
        writethai(70,30,12,2,kmit[0]);
        bbx0(200,80,440,160,15,0,3);
        writetext(264,35,8,1,0,20,"c");
        settextstyle(1,0,20);
        setcolor(4);
        for(x1 = 265;x1 <= 285;x1++)
        {
                outtextxy(x1, 35, "c");
        }
        settextstyle(1,0,3);
        setcolor(12);
        speaker(800,10);
        for(x2=328;x2>=325;x2--)
        {
                outtextxy(x2,109, "CAD");
        }
```

```
        writetext(349,136,0,2,0,4,"R");
        writetext(348,136,14,2,0,4,"R");
        setcolor(8);
        circle(350,141,5);
        setcolor(14);
        circle(350,141,6);
        settextstyle(4,0,5);
        setcolor(10);
        bbx1(260,380,370,440,15,8,3);
        writetext(286,403,0,0,0,2,"V1.0");
        writetext(285,403,6,0,0,2,"V1.0");
        settextstyle(4,0,5);
        setcolor(13);
        outtextxy(20,300, "Communications System Analysis");
        outtextxy(180, 250, " Software for ");
        kmitsong();
        bbx0(260,380,370,440,8,15,14);
        writetext(286,403,0,0,0,2,"V1.0");
        writetext(285,403,3,0,0,2,"V1.0");
        for(count = 0; count <= 36; count++) {
        writetext(posx+15*count,posy-6,random(15),0,0,2,welcome[count]);
                speaker(950,50);
                writetext(posx+15*count,posy-6,11,0,0,2,welcome[count]);
        }
        bbxesci(5,5);       bbxtrii(618,5);
        for(count=1;count <= 3;count++) {
                speaker(1000,90);
                speaker(500,100);
        }
        bbxesc(5,5);        bbxtri(618,5);
        bbx1(260,380,370,440,15,8,3);
        writetext(286,403,0,0,0,2,"V1.0");
        writetext(285,403,6,0,0,2,"V1.0");
        delay(100);
/*********************************<P2>*************************************/
        bbx0(0,0,639,479,11,11,3);
        bbxesc(5,5);        bbxtri(618,5);
        bbx1(50,5,590,150,15,8,7);
        bbx1(50,400,590,470,15,8,7);
        writethai(71,411,0,2,kmit[0]);
        writethai(226,436,0,2,kmit[1]);
        writethai(70,410,12,2,kmit[0]);
        writethai(225,435,12,2,kmit[1]);
        bbx0(200,110,440,140,15,0,6);
        bbx0(150,170,490,240,8,15,0);
        writethai(165,175,14,1,hotel[0]);
        writethai(250,175,14,1,hotel[1]);
        writethai(165,195,14,1,hotel[2]);
        writethai(251,195,14,1,hotel[3]);
        writethai(165,215,14,1,hotel[4]);
        writethai(251,215,14,1,hotel[5]);
        writetext(390,175,14,4,0,1,"34131171");
        writetext(390,195,14,4,0,1,"34131172");
        writetext(390,215,14,4,0,1,"34131173");
```

```
        writetext(68,78,4,1,0,2,project[1]);
        writethai(65,37,11,2,project[0]);
        writethai(233,114,14,1,engineering[0]);
        bbx1(150,260,490,290,14,8,2);
        writethai(188,263,0,1,kisdakorn[0]);
        writethai(355,263,0,1,kisdakorn[1]);
        bbxesci(5,5);       bbxtrii(618,5);
        sleep(1);
        speaker(1000,90);
        bbxesc(5,5);        bbxtri(618,5);
        Select();
}
/*****************************<end of title.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : SELECT.C
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <graphics.h >
#include <stdlib.h  >
#include <stdio.h   >
#include <stdarg.h  >
#include <conio.h   >
#include <ctype.h   >
#include <dos.h     >

char    *sec[3] = {
        "TUTOR", "ANALYSIS",  " EXIT "
};

char    *infor[3] = {
        "COMMUNICATION   TUTORIAL",
        "communication   analysis",
        "EXIT to DOS"
};

int     mation[3] = {
        70,70,240
};

int     barsec[3] = {
        145,215,285
};

int     barX[3] = {
        280,256,274
};

int     funC[2] = {
        514, 600
};

int     funR[6] = {
        20, 80, 140, 200, 260, 320
};

int     funX[6] = {
        542,542,542,542,542,542
};
int     funY[6] = {
        38,98,158,218,278,338
};
```

```
int     blockY[9] = {
        20,60,100,140,180,220,260,300,340
};

int     blockX[8] = {
        10,70,130,190,250,310,370,430
};

int     sx1=245, ex1=388;
int     begin, start;


void Select()
{
        int key;
        backg();
        bbx(sx1-3,barsec[0]-11,ex1+3,barsec[2]+75,7,7,7);
        bbxesc(5,5);        bbxtri(618,5);
        bbx(sx1,barsec[0],ex1,barsec[0]+60,15,8,4);
        bbx(sx1,barsec[1],ex1,barsec[1]+60,15,8,3);
        bbx(sx1,barsec[2],ex1,barsec[2]+60,15,8,3);
        bbx1(50,440,590,470,8,15,3);
        writetext(barX[0],barsec[0]+23,3,0,0,2,sec[0]);
        writetext(barX[1],barsec[1]+23,6,0,0,2,sec[1]);
        writetext(barX[2],barsec[2]+23,6,0,0,2,sec[2]);
        writetext(mation[0],448,14,0,0,2,infor[0]);
        begin = 0;
        do{
                moveto(640,480);
                key = getch();
                if(key==0) {
                        key=getch();
                        if((key==72) || (key==80)) {
                                check(key);
                        }
                }
                if(key == 27) {
                        bbxesci(5,5);       bbxtrii(618,5);
                        delay(500);
                        ccadtitle();
                        break;
                }
                else {
                        if(key==13) {
                bbxtrii(618,5);
                bbx1(sx1,barsec[begin],ex1,barsec[begin]+60,8,15,14);
                writetext(barX[begin],barsec[begin]+23,3,0,0,2,
                                                sec[begin]);
                bbx1(50,440,590,470,8,15,3);
                writetext(mation[begin],448,14,0,0,2,infor[begin]);
                delay(200);
                        switch(begin) {
                                case 0 : tutor();   break;
```

```
                                        case 1 : analysis();  break;

                                        case 2 : closegraph(); exit(1);
                                                 break;
                                        }
                                }
                        }

                } while((key != 13));
                setcolor(15);
        }

        backg()
        {
                int     i,      j;
                bbx0(0,0,639,479,13,13,7);
                bbxesci(5,5);       bbxtrii(618,5);
                for(i=1;i<=7;i++) {
                for(j=1;j<=13;j++) {
                                ic(j*45-9,i*58-38,random(2)+7);
                        }
                }
        }

        /* checking key when the button is pressed */
        int check(int key)
        {
                if(key==72) {
                speaker(1000,10);
                bbx(sx1,barsec[begin],ex1,barsec[begin]+60,15,8,3);
                writetext(barX[begin],barsec[begin]+23,6,0,0,2,sec[begin]);
                begin = (begin == 0 ) ? 2 : begin - 1;
                bbx(sx1,barsec[begin],ex1,barsec[begin]+60,15,8,4);
                writetext(barX[begin],barsec[begin]+23,3,0,0,2,sec[begin]);
                bbx1(50,440,590,470,8,15,3);
                writetext(mation[begin],448,14,0,0,2,infor[begin]);
                }
                if(key==80) {
                speaker(1000,10);
                bbx(sx1,barsec[begin],ex1,barsec[begin]+60,15,8,3);
                writetext(barX[begin],barsec[begin]+23,6,0,0,2,sec[begin]);
                begin = (begin == 2 ) ? 0 : begin + 1;
                bbx(sx1,barsec[begin],ex1,barsec[begin]+60,15,8,4);
                writetext(barX[begin],barsec[begin]+23,3,0,0,2,sec[begin]);
                bbx1(50,440,590,470,8,15,3);
                writetext(mation[begin],448,14,0,0,2,infor[begin]);
                }
        }

        ic(int x,int y,int color)
        {
                int n;
                bbx1(x,y,x+20,y+45,15,0,color);
                bbx1(x+7,y,x+13,y+1,7,15,7);
```

```
        bbx1(x+8,y,x+12,y+2,7,15,7);
        bbx1(x+8,y,x+12,y+3,7,15,7);
        bbx1(x+9,y,x+11,y+4,7,15,7);
        bbx1(x+9,y,x+11,y+5,7,15,7);
        writetext(x+8,y+10,0,2,1,1,"74LS04");
        for(n=1;n<=7;n++) {
                bbx1(x-1,y+(6*n-3),x+1,y+6*n,15,8,7);
                bbx0(x+19,y+(6*n-3),x+21,y+6*n,15,8,7);
        }
}
/*****************************<end of select.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : TUTOR.C
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <process.h>
#include <math.h>
#include "dft.h"
#include "get.h"
#include "disk.h"
#include "allkey.h"

void tutor(void);
void calwave(void);

char    *com[8] = {"F1","F2","F3","F4","F5","F6","F7"};
char    *mand[8] = {
        "-WAVE","-TCF","-FCT","-AM","-FM","-PLOT","-RETURN"
};
char    *key[20] = {"","/","*","-","7","8","9","+","4","5",
                    "6","","1","2","3","","0","",".",""
};
int     status;

void tutor(void)
{
        setviewport(0,0,getmaxx(),getmaxy(),0);
        bbx0(0,0,639,479,7,7,7);
        bbx0(0,0,639,33,9,9,9);
        bbx0(0,0,34,479,1,1,1);
        bbx0(605,0,639,479,1,1,1);
        bbx0(0,464,639,479,9,9,9);
        bbxesc(1,1);        bbxesc(623,1);
        bbxtri(17,1);       bbxtri(606,1);
        press();
        bbx0(35,33,604,463,15,15,7);
        writetext(60,13,4,0,0,1,com[0]);
        writetextxy(75,13,11,0,0,1,mand[0]);
        writetext(140,13,4,0,0,1,com[1]);
        writetextxy(155,13,11,0,0,1,mand[1]);
        writetext(215,13,4,0,0,1,com[2]);
        writetextxy(230,13,11,0,0,1,mand[2]);
        writetext(285,13,4,0,0,1,com[3]);
        writetextxy(300,13,11,0,0,1,mand[3]);
        writetext(350,13,4,0,0,1,com[4]);
```

```
        writetextxy(365,13,11,0,0,1,mand[4]);
        writetext(420,13,4,0,0,1,com[5]);
        writetextxy(435,13,11,0,0,1,mand[5]);
        writetext(500,13,4,0,0,1,com[6]);
        writetextxy(515,13,11,0,0,1,mand[6]);
        writetext(260,469,11,0,0,1,"Alt-X to Exit");
        while(!bioskey(1)) {
                switch(getkey()) {
                        case F1 : cwave();          break;
                        case F2 : tcf();            break;
                        case F3 : fct();            break;
                        case F4 : amo();            break;
                        case F5 : fmo();            break;
                        case F6 : print();          break;
                        case F7 : ret();            break;
                        case CR : help0();          break;
                        case ESC: Select();         break;
                        case AX : closegraph();     exit(1);
                        default : help2();          break;
                }
        }
        bioskey(0);
        analysis();
}

cwave()
{
        writetext(60,13,12,0,0,1,com[0]);
        press1();       delay(90);
        writetext(60,13,4,0,0,1,com[0]);
        press();
        calwave();
}

tcf()
{
        writetext(140,13,12,0,0,1,com[1]);
        press1();       delay(90);
        writetext(140,13,4,0,0,1,com[1]);
        press();
}

fct()
{
        writetext(215,13,12,0,0,1,com[2]);
        press1();       delay(90);
        writetext(215,13,4,0,0,1,com[2]);
        press();
}

amo()
{
        writetext(285,13,12,0,0,1,com[3]);
        press1();       delay(90);
```

```
        writetext(285,13,4,0,0,1,com[3]);
        press();
        ammod();
}

fmo()
{
        writetext(350,13,12,0,0,1,com[4]);
        press1();       delay(90);
        writetext(350,13,4,0,0,1,com[5]);
        press();
        fmmod();
}

print()
{
        writetext(420,13,12,0,0,1,com[5]);
        press1();       delay(90);
        writetext(420,13,4,0,0,1,com[5]);
        press();
        plot();
        tutor();
}

ret()
{
        writetext(500,13,12,0,0,1,com[6]);
        press1();       delay(90);
        Select();
}

void calwave(void)
{
char    *equation[9] = {"_1(t) = sine wave",  "_2(t) = cos wave",
                "_3(t) = square wave",        "_4(t) = triangle wave",
                "_5(t) = saw wave",           "_6(t)=half circle wave",
                "_7(t) = pulse wave",         "_8(t) = gaussian wave",
                "       exit"                                          };
        int     x, y, keys;
        window31(40,20);
        writetext(230,4,14,2,0,4,"Application Windows MKWAVE");

        bbx(40,250,162,455,15,8,7);
        bbx0(50,265,150,290,8,15,14);
        for(y = 1; y <= 5; y++) {
                for(x = 1; x <= 4; x++) {
                        bbx1(25*x+30,25*y+300,25*x+50,25*y+320,15,8,7);
                        writetext(25*x+37,25*y+307,0,0,0,1,key[(y-1)*4+x-1]);
                }
        }
        writetext(52,295,8,2,0,2,"FX-5000F");
        bbx1(55,425,100,445,15,8,7);
        bbx1(130,350,150,395,15,8,7);
        bbx1(130,400,150,445,15,8,7);
```

```
writetext(137,370,0,0,0,1,key[7]);
writetext(65,432,0,0,0,1,key[16]);
writetext(62,332,0,2,0,1,"NUM");
writetext(135,410,0,2,1,3,"ENTER");
writethai(100,30,14,2,"_++ª-¦-_--_-+");
writetext(104,130,0,2,0,8,"FX-5000F");
bbx1(100,70,500,120,8,15,14);
writetext(120,90,3,0,0,2,equation[0]);
writetext(60,275,3,2,0,2,equation[0]);
status = 0;
do{
        moveto(640,480);
        keys = getch();
        if(keys==0) {
                keys=getch();
                if((keys==72) || (keys==80)) {
                        if(keys==72) {
                                speaker(1000,10);
                        status = (status == 0 ) ? 8 : status - 1;
                                bbx0(52,267,148,288,14,14,14);
                                bbx1(105,75,495,115,14,14,14);
                        writetext(120,90,3,0,0,2,equation[status]);
                        writetext(60,275,3,2,0,2,equation[status]);
                        }
                        if(keys==80) {
                                speaker(1000,10);
                        status = (status == 8 ) ? 0 : status + 1;
                                bbx0(52,267,148,288,14,14,14);
                                bbx1(105,75,495,115,14,14,14);
                        writetext(120,90,3,0,0,2,equation[status]);
                        writetext(60,275,3,2,0,2,equation[status]);
                        }
                }
        }
        if(keys == 27) {
                tutor();
                break;
        }
        else {
                if(keys==13) {
                switch(status) {
                        case 0 : si();      break;
                        case 1 : co();      break;
                        case 2 : sq();      break;
                        case 3 : tr();      break;
                        case 4 : sa();      break;
                        case 5 : ha();      break;
                        case 6 : pu();      break;
                        case 7 : gu();      break;
                        case 8 : tutor();   break;
                        }
                }
        }
```

```
        } while((keys != 13));
        setcolor(15);
        tutor();
}
/*****************************<end of tutor.c>*************************/
```

```
/*****************************************************************************
FILE NAME : demo.c
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <graphics.h>
#include <alloc.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "allkey.h"

void    pauses(void);
void    window31(int x, int y);
void    *ReadImage(char *filename);
void    tutor(void);
void    calwave(void);

char    *com[7] = {"F1","F2","F3","F4","F5","F6"};
char    *mand[7] = {
        "WAVE","FREQ","TRAN","AMOD","FMOD","EXIT"
};
char    *key[20] = {"","/","*","-","7","8","9","+","4","5",
                    "6","","1","2","3","","0","",".",""
};
int     status;

void tutor(void)
{
        viewdemo();
        botton(90,5,150,25);
        botton(170,5,230,25);
        botton(250,5,310,25);
        botton(330,5,390,25);
        botton(410,5,470,25);
        botton(490,5,550,25);
        writetext(96,12,4,0,0,1,com[0]);
        writetext(114,12,1,0,0,1,mand[0]);
        writetext(176,12,4,0,0,1,com[1]);
        writetext(194,12,1,0,0,1,mand[1]);
        writetext(256,12,4,0,0,1,com[2]);
        writetext(274,12,1,0,0,1,mand[2]);
        writetext(336,12,4,0,0,1,com[3]);
        writetext(354,12,1,0,0,1,mand[3]);
        writetext(416,12,4,0,0,1,com[4]);
        writetext(434,12,1,0,0,1,mand[4]);
        writetext(496,12,4,0,0,1,com[5]);
```

```
        writetext(514,12,1,0,0,1,mand[5]);
        bbx1(150,150,490,370,15,8,7);
        writetext(290,180,9,4,0,2,"DEMO");
        writetext(165,220,3,1,0,1,"COMMUNICATION SYSTEM ANALYSIS");
        writetext(210,280,3,2,0,4,"KING MONGKUT'S INSTITUTE OF TECHNOLOGY");
        writetext(290,300,11,2,0,4,"LADKRABANG");
        while(!bioskey(1)) {
                switch(getkey()) {
                        case F1 : cwave();          break;
                        case F2 : freqs();          break;
                        case F3 : trans();          break;
                        case F4 : ammods();         break;
                        case F5 : fmmods();         break;
                        case F6 : ret();            break;
                        case CR : kmitsong();       break;
                        case ESC: ret();            break;
                        case AX : closegraph();     exit(1);
                        default : break;
                }
        }
        bioskey(0);
        analysis();
}

viewdemo()
{
        setviewport(0,0,getmaxx(),getmaxy(),0);
        bbx1(0,0,getmaxx(),getmaxy(),0,0,0);
        bbx1(20,0,620,getmaxy(),11,8,3);
        setcolor(0);
        line(30,10,610,10);
        line(30,10,30,30);
        line(610,10,610,30);
        line(20,30,620,30);
        line(20,450,620,450);
        setcolor(11);
        line(31,11,609,11);
        line(31,11,31,30);
        line(609,11,609,30);
        line(21,31,619,31);
        line(21,451,619,451);
        bbx1(31,31,610,470,8,11,8);
}

botton(int sx,int sy,int ex,int ey)
{
        setfillstyle(1,3);
        bar(sx,sy,ex,ey);
        setfillstyle(1,11);
        bar(sx,sy,sx+1,ey);
        bar(sx,sy,ex,sy+1);
        setfillstyle(1,8);
        bar(sx+1,ey-1,ex,ey);
        bar(ex-1,sy+1,ex,ey);
```

```
        setcolor(0);
        rectangle(sx-1,sy-1,ex+1,ey+1);
}

botton1(int sx,int sy,int ex,int ey)
{
        setfillstyle(1,7);
        bar(sx,sy,ex,ey);
        setfillstyle(1,8);
        bar(sx,sy,sx+1,ey);
        bar(sx,sy,ex,sy+1);
        setfillstyle(1,11);
        bar(sx+1,ey-1,ex,ey);
        bar(ex-1,sy+1,ex,ey);
        setcolor(0);
        rectangle(sx-1,sy-1,ex+1,ey+1);
}

cwave()
{
        botton1(90,5,150,25);
        writetext(96,13,12,0,0,1,com[0]);
        writetext(114,13,9,0,0,1,mand[0]);
        delay(200);
        botton(90,5,150,25);
        writetext(96,12,4,0,0,1,com[0]);
        writetext(114,12,1,0,0,1,mand[0]);
        calwave();
}

freqs()
{
        botton1(170,5,230,25);
        writetext(176,13,12,0,0,1,com[1]);
        writetext(194,13,9,0,0,1,mand[1]);
        delay(100);
        botton(170,5,230,25);
        writetext(176,12,4,0,0,1,com[1]);
        writetext(194,12,1,0,0,1,mand[1]);
        freqsdemo();
}

trans()
{
        botton1(250,5,310,25);
        writetext(256,13,12,0,0,1,com[2]);
        writetext(274,13,9,0,0,1,mand[2]);
        delay(100);
        botton(250,5,310,25);
        writetext(256,12,4,0,0,1,com[2]);
        writetext(274,12,1,0,0,1,mand[2]);

}
```

```
ammods()
{
        botton1(330,5,390,25);
        writetext(336,13,12,0,0,1,com[3]);
        writetext(354,13,9,0,0,1,mand[3]);
        delay(100);
        botton(330,5,390,25);
        writetext(336,12,4,0,0,1,com[3]);
        writetext(354,12,1,0,0,1,mand[3]);

}

fmmods()
{
        botton1(410,5,470,25);
        writetext(416,13,12,0,0,1,com[4]);
        writetext(434,13,9,0,0,1,mand[4]);
        delay(100);
        botton(410,5,470,25);
        writetext(416,12,4,0,0,1,com[4]);
        writetext(434,12,1,0,0,1,mand[4]);

}

ret()
{
        botton1(490,5,550,25);
        writetext(496,13,12,0,0,1,com[5]);
        writetext(514,13,9,0,0,1,mand[5]);
        delay(100);
        botton(490,5,550,25);
        writetext(496,12,4,0,0,1,com[5]);
        writetext(514,12,1,0,0,1,mand[5]);
        Select();
}

void calwave(void)
{
char    *equation[9] = {"_1(t) = sine wave",  "_2(t) = cos wave",
                "_3(t) = square wave",        "_4(t) = triangle wave",
                "_5(t) = saw wave",           "_6(t)=half circle wave",
                "_7(t) = pulse wave",         "_8(t) = gaussian wave",
                "      exit"                                        };
        int     x, y, keys;
        window31(40,20);
        writetext(230,4,14,2,0,4,"Application Windows MKWAVE");
        bbx(40,250,162,455,15,8,7);
        bbx0(50,265,150,290,8,15,14);
        for(y = 1; y <= 5; y++) {
                for(x = 1; x <= 4; x++) {
                        bbx1(25*x+30,25*y+300,25*x+50,25*y+320,15,8,7);
                        writetext(25*x+37,25*y+307,0,0,0,1,key[(y-1)*4+x-1]);
                }
        }
```

```
writetext(52,295,8,2,0,2,"FX-5000F");
bbx1(55,425,100,445,15,8,7);
bbx1(130,350,150,395,15,8,7);
bbx1(130,400,150,445,15,8,7);
writetext(137,370,0,0,0,1,key[7]);
writetext(65,432,0,0,0,1,key[16]);
writetext(62,332,0,2,0,1,"NUM");
writetext(135,410,0,2,1,3,"ENTER");
writethai(100,30,14,2,"i++|-¶-_--i-+");
writetext(104,130,0,2,0,8,"FX-5000F");
bbx1(100,70,500,120,8,15,14);
writetext(120,90,3,0,0,2,equation[0]);
writetext(60,275,3,2,0,2,equation[0]);
status = 0;
do{
        moveto(640,480);
        keys = getch();
        if(keys==0) {
                keys=getch();
                if((keys==72) || (keys==80)) {
                        if(keys==72) {
                                speaker(1000,10);
                        status = (status == 0 ) ? 8 : status - 1;
                                bbx0(52,267,148,288,14,14,14);
                                bbx1(105,75,495,115,14,14,14);
                        writetext(120,90,3,0,0,2,equation[status]);
                        writetext(60,275,3,2,0,2,equation[status]);
                        }
                        if(keys==80) {
                                speaker(1000,10);
                        status = (status == 8 ) ? 0 : status + 1;
                                bbx0(52,267,148,288,14,14,14);
                                bbx1(105,75,495,115,14,14,14);
                        writetext(120,90,3,0,0,2,equation[status]);
                        writetext(60,275,3,2,0,2,equation[status]);
                        }
                }
        }
        if(keys == 27) {
                tutor();
                break;
        }
        else {
                if(keys==13) {
                switch(status) {
                        case 0 : si();      break;
                        case 1 : co();      break;
                        case 2 : sq();      break;
                        case 3 : tr();      break;
                        case 4 : sa();      break;
                        case 5 : ha();      break;
                        case 6 : pu();      break;
                        case 7 : gu();      break;
                        case 8 : tutor();   break;
```

```
                                }
                        }
                }

        } while((keys != 13));
        setcolor(15);
        tutor();
}

freqsdemo()
{
 int i;
setviewport(0,0,getmaxx(),getmaxy(),0);
bbx0(0,0,getmaxx(),getmaxy(),15,1,7);
bbx1(5,5,getmaxx()-5,50,11,8,3);
bbx0(5,60,410,175,11,8,3);
bbx0(5,185,410,265,11,8,3);
bbx0(5,275,410,460,11,8,3);
bbx0(425,60,getmaxx()-5,460,11,8,3);
writetext(30,185,0,2,0,4,
"5   4    3      2      1      0      -1      -2");
writetext(20,190,0,2,0,4,
"10  10   10    10     10     10   10   10   (m)");
writetext(20,240,0,2,0,4,
" 3  30   300   3     30     300   3    30   (f)");
writetext(20, 15,14,0,0,3,"BROADCAST FREQUENCY BANDS");
writetext(10, 70,0,0,0,1,"TABLE 1. Frequency Designations");
writetext(10, 90,0,0,0,1,"VLF (very low frequencies)    3 Hz to 30  kHz");
writetext(10,100,0,0,0,1," LF (low frequencies)      30 kHz to 30  kHz");
writetext(10,110,0,0,0,1," MF (medium frequencies)   300 kHz to 30  kHz");
writetext(10,120,0,0,0,1," HF (high frequencies)     3 MHz to 30  kHz");
writetext(10,130,0,0,0,1,"VHF (very high frequencies)  30 MHz to 30  kHz");
writetext(10,140,0,0,0,1,"UHF (ultrahigh frequencies)  300 MHz to 30  kHz");
writetext(10,150,0,0,0,1,"SHF (superhigh frequencies)  3 GHz to 30  kHz");
writetext(10,160,0,0,0,1,"EHF (extra-high frequencies)  30 GHz to 30  kHz");
writetext(10,290,0,0,0,1,"TABLE 2. Radar Frequency Bands");
writetext(10,215,4,0,0,1,"VLF  LF  MF  HF  VHF  UHF  SHF  EHF");
writetext(10,250,0,0,0,0,"kHz  kHz kHz MHz  MHz  MHz  GHz GHz");
writetext(10,310,0,0,0,1,"Band       Frequency range     Wavelength,cm");
writetext(10,330,0,0,0,1," HF         3-30   MHz     10,000 - 1,000");
writetext(10,340,0,0,0,1,"VHF        30-300  MHz      1,000 - 100");
writetext(10,350,0,0,0,1,"UHF       300-1000 MHz       100 - 30");
writetext(10,360,0,0,0,1,"L         1.0-2.0  GHz       30 - 15");
writetext(10,370,0,0,0,1,"S         2.0-4.0  GHz       15 - 7.50");
writetext(10,380,0,0,0,1,"C         4.0-8.0  GHz      7.50 - 3.75");
writetext(10,390,0,0,0,1,"X         8.0-12.0 GHz      3.75 - 2.50");
writetext(10,400,0,0,0,1,"Ku       12.0-18.0 GHz      2.50 - 1.67");
writetext(10,410,0,0,0,1,"K        18.0-27.0 GHz      1.67 - 1.11");
writetext(10,420,0,0,0,1,"Ka       27.0-40.0 GHz      1.11 - 0.75");
writetext(10,430,0,0,0,1,"Millimeter  40-300  GHz      0.75 - 0.10");
writetext(10, 80,0,0,0,1,"-----------------------------------------");
writetext(10,200,0,0,0,1,"-----------------------------------------");
writetext(10,230,0,0,0,1,"-----------------------------------------");
writetext(10,300,0,0,0,1,"-----------------------------------------");
```

```
writetext(10,320,0,0,0,1,"-----------------------------------------------");
writetext(10,440,0,0,0,1,"-----------------------------------------------");

pauses();
viewdemo();
bbx1(33,33,608,408,15,8,7);
bbx1(35,45,603,215,11,8,3);
bbx1(35,255,603,400,11,8,3);
writetext(260,55,14,0,0,2,"AM Band");
writetext(260,265,4,0,0,2,"FM Band");
writetext(50,85,1,0,0,1,
"Stations are assigned carrier frequencies at 10 kHz intervals from");
writetext(50,100,1,0,0,1,
"540 to 1600 kHz. The bandwidth of transmissions is nominally about");
writetext(50,115,1,0,0,1,
"10 kHz.  Stations in local proximity are usually assigned carrier");
writetext(50,130,1,0,0,1,
"frequencies which are separated by 30 kHz or more.  Interference");
writetext(50,145,1,0,0,1,
"between transmissions is controlled by a combination of frequncy");
writetext(50,160,1,0,0,1,
"allocation, transmitter power, transmitting antenna pattern, and");
writetext(50,175,1,0,0,1,
"possible night-time operating restrictions.  Required carrier");
writetext(50,190,1,0,0,1,
"stability +/-20Hz;locensed power output is for unmodulated carrier");
writetext(50,295,5,0,0,1,
"Stations are assigned carrier frequencies at 200 kHz intervals from");
writetext(50,315,5,0,0,1,
"88 MHz to 108 MHz.  Transmissions recived beyond the line-of-sight");
writetext(50,330,5,0,0,1,
"distance are week signals in the presence of strong signals.  These");
writetext(50,345,5,0,0,1,
"effectshelp to define the station broadcast area.  Peak frequency");
writetext(50,360,5,0,0,1,
"deviation is 75 kHz.  Minimum required carrier frequency stability");
writetext(50,375,5,0,0,1,
"is (+/-) 0.002% (that is, (+/-) 2kHz).");

pauses();
bbx0(0,0,getmaxx(),getmaxy(),0,0,0);
bbx0(0,0,480,370,15,1,7);
writetext(50,70,14,0,0,1,"TABLE 3. VHF Television statation Allocations");
writetext(50, 80,0,0,0,1,"-----------------------------------------");
writetext(50,110,0,0,0,1,"-----------------------------------------");
writetext(50, 90,4,0,0,1,"Channel  Frequency span       Video carrier");
writetext(50,100,4,0,0,1,"number      (MHz)              (MHz)");
writetext(50,120,1,0,0,1," 1                        Not used");
writetext(50,130,1,0,0,1," 2         54 - 60          55.25");
writetext(50,140,1,0,0,1," 3         60 - 66          61.25");
writetext(50,150,1,0,0,1," 4         66 - 76          67.25");
writetext(50,160,1,0,0,1," 5         76 - 82          77.25");
writetext(50,170,1,0,0,1," 6         82 - 88          83.25");
writetext(290,180,1,0,0,1,"(Standard FM)");
```

```
writetext(295,190,1,0,0,1,"band 88-108");
writetext(50,200,1,0,0,1," 7       174 - 180        175.25");
writetext(50,210,1,0,0,1," 8       180 - 186        181.25");
writetext(50,220,1,0,0,1," 9       186 - 192        187.25");
writetext(50,230,1,0,0,1,"10       192 - 198        183.25");
writetext(50,240,1,0,0,1,"11       198 - 204        199.25");
writetext(50,250,1,0,0,1,"12       204 - 210        205.25");
writetext(50,260,1,0,0,1,"13       210 - 216        211.25");

pauses();
viewdemo();
writetext(40,70,14,0,0,1,"TABLE 4. UHF Television statation Allocations");
writetext(40, 80,0,0,0,1,
"-------------------------------------------------------------");
writetext(40,110,0,0,0,1,
"-------------------------------------------------------------");
writetext(40, 90,1,0,0,1,
"Channel Freq-span  Video carrier   Channel  Freq-span   Video carrier");
writetext(40,100,1,0,0,1,
"number  (MHz)      (MHz)       number   (MHz)       (MHz)");

for(i=1;i<=28;i++) {
        writetextxy(40,120+10*i,11,0,0,1,
        "%d     %d-%d      %d.25       %d      %d-%d       %d.25",
        13+i,464+6*i,470+6*i,465+6*i,41+i,632+6*i,638+6*i,633+6*i);
}

pauses();
bbx0(0,0,getmaxx(),getmaxy(),0,0,0);
bbx0(0,0,320,370,15,1,7);
bbx0(350,0,639,370,11,8,3);
writetext(30, 70,14,0,0,1,"TABLE 5. Amateur Bands");
writetext(30, 80,0,0,0,1,"-------------------------");
writetext(30,110,0,0,0,1,"-------------------------");
writetext(30, 90,3,0,0,1,"Band          Frequency");
writetext(30,100,3,0,0,1,"designation     allocation(MHz)");
writetext(30,120,4,0,0,1,"160           1.800-2.000");
writetext(30,130,4,0,0,1," 80           3.500-4.000");
writetext(30,140,4,0,0,1," 40           7.000-7.300");
writetext(30,150,4,0,0,1," 20          14.000-14.350");
writetext(30,160,4,0,0,1," 15          21.000-21.450");
writetext(30,170,4,0,0,1," 10          28.000-29.700");
writetext(30,180,4,0,0,1,"  6          50.000-54.0");
writetext(30,190,4,0,0,1,"  2            144-148");
writetext(30,200,4,0,0,1,"               220-225");
writetext(30,210,4,0,0,1,"               420-450");
writetext(30,220,4,0,0,1,"              1,215-1,300");
writetext(30,230,4,0,0,1,"              2,300-2,450");
writetext(30,240,4,0,0,1,"              3,300-3,500");
writetext(30,250,4,0,0,1,"              5,650-5,925");
writetext(30,260,4,0,0,1,"             10,000-10,500");
writetext(30,270,4,0,0,1,"             24,000-24,500");
writetext(50,280,4,0,0,1,"             48,000-50,000");
writetext(30,290,4,0,0,1,"             71,000-76,000");
```

```
writetext(30,300,4,0,0,1,"            165,000-170,000");
writetext(30,310,4,0,0,1,"            240,000-250,000");
writetext(30,320,4,0,0,1,"             above 300,000");
writetext(390, 70,4,2,0,4,"Notes...");
writetext(390,100,1,2,0,4,"1. Maximum authorized power input to");
writetext(390,115,1,2,0,4,"   the final transmitter stage is 1 kW");
writetext(390,130,1,2,0,4,"2. Maximum amplitude modulation allowed");
writetext(390,145,1,2,0,4,"   is 100%. Except for brief tests or");
writetext(390,160,1,2,0,4,"   adjustments, the transmitter may not");
writetext(390,175,1,2,0,4,"   emit a carrier wave on frequencies");
writetext(390,190,1,2,0,4,"   below 51 kMHz unless modulated for");
writetext(390,205,1,2,0,4,"   the purpose of communication.");
writetext(390,220,1,2,0,4,"3. Single-sideband: lower sideband is");
writetext(390,235,1,2,0,4,"   customarily use on 80m and 40m,upper");
writetext(390,250,1,2,0,4,"   sideband on the higher frequencies.");
pauses();
tutor();
}

void pauses(void)
{
        while(kbhit()) getch();
        getch();
}

void *ReadImage(char *filename)
{
        unsigned xsize,ysize;
        unsigned long size;
        FILE *f1=fopen(filename,"r");
        void far *tempimage;

        xsize=fgetc(f1)|(fgetc(f1)<<8);
        ysize=fgetc(f1)|(fgetc(f1)<<8);
        size=imagesize(0,0,xsize,ysize);
        tempimage=(void far*)farmalloc((long)size);
        if(!tempimage) {
                beep();
                calwave();
        }
        rewind(f1);
        fread(tempimage,size,1,f1);
        fclose(f1);
        return(tempimage);
        farfree(tempimage);
}

tektronix()
{
        int i;
        setviewport(0,0,639,479,0);
        bbx0(0,0,639,479,7,7,7);
        bbx1(50,20,590,460,15,8,7);
        bbx0(70,40,570,440,15,15,0);
```

```
        bbxesc(5,5);        bbxtri(618,5);
        setlinestyle(1,1,NORM_WIDTH);
        for(i=1;i<=9;i++) {
                setcolor(2);
                line(71,50*i+40,569,50*i+40);
                line(50*i+70,41,50*i+70,439);
                writetextxy(555,50*i+30,10,2,0,2,"-%d",10*i);
        }
        for(i=0;i<=4;i++) {
                writetextxy(75,100*i-55,10,2,0,2,"%d",125-25*i);
        }
        setcolor(10);
        line(71,340,569,340);
        line(320,41,320,439);
        bbx0(70,390,90,440,10,10,0);
        bbx0(570,390,550,440,10,10,0);
        writetextxy(75,395,10,2,1,5,"LIN %");
        writetextxy(555,395,10,2,1,5,"LOGdB");
        writetext(70,28,1,0,0,1,"Tektronix");
        writetext(160,28,0,0,0,1,"2710");
        writetext(320,28,0,2,0,4,"10kHz - 1.8GHz   50ohm");
        setlinestyle(SOLID_LINE,1,NORM_WIDTH);
}

void window31(int x, int y)
{
        setviewport(x,y,getmaxx(),getmaxy(),0);
        bbx0(1,1,getmaxx(),getmaxy(),15,15,3);
        bbxesc(1,1);        bbxesc(623-x,1);
        bbx0(22,21,getmaxx()-x-21,getmaxy()-y-15,15,15,9);
        bbx0(22,2,getmaxx()-x-21,16,15,8,6);
}

/*****************************<end of demo.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : ANALYSIS.C
progammer : WEERASAK NAMVONG
Create    : 25 / Feb / 1992
Update    : 19 / may / 1992
Purpose   : Prototype function for CCAD.EXE
Comment   : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include      <graphics.h>
#include      <stdlib.h>
#include      <stdio.h>
#include      <string.h>
#include      <process.h>
#include      <math.h>
#include      "dft.h"
#include      "get.h"
#include      "disk.h"
#include      "allkey.h"

void    analysis(void);
void    plot(void);

char    *gettext(char *title_text);
char    *peak[3];
char    *data1[7] = {"F1","F2","F3","F4","F5","F6","F7"};
char    *data2[7] = {
        "-MAKE","-MIX","-MOD","-PLOT","-AM","-FM","-RETURN"
};
int     px = 512,aapx=500;
double  xf,aaxf;
float   *value,*valu,aamm,bbmm,amx,bmx;
float   lin1,lin2,*lin;

void analysis(void)
{
        setviewport(0,0,getmaxx(),getmaxy(),0);
        bbx0(0,0,639,479,7,7,7);
        bbx0(0,0,639,33,9,9,9);
        bbx0(0,0,34,479,1,1,1);
        bbx0(605,0,639,479,1,1,1);
        bbx0(0,464,639,479,9,9,9);
        bbxesc(1,1);      bbxesc(623,1);
        bbxtri(17,1);     bbxtri(606,1);
        press();
        bbx0(35,33,604,463,15,15,7);
        writetext(60,13,4,0,0,1,data1[0]);
        writetextxy(75,13,11,0,0,1,data2[0]);
        writetext(145,13,4,0,0,1,data1[1]);
        writetextxy(160,13,11,0,0,1,data2[1]);
        writetext(225,13,4,0,0,1,data1[2]);
        writetextxy(240,13,11,0,0,1,data2[2]);
        writetext(300,13,4,0,0,1,data1[3]);
```

```
        writetextxy(315,13,11,0,0,1,data2[3]);
        writetext(380,13,4,0,0,1,data1[4]);
        writetextxy(395,13,11,0,0,1,data2[4]);
        writetext(445,13,4,0,0,1,data1[5]);
        writetextxy(460,13,11,0,0,1,data2[5]);
        writetext(500,13,4,0,0,1,data1[6]);
        writetextxy(515,13,11,0,0,1,data2[6]);

        writetext(260,469,11,0,0,1,"Alt-X to Exit");
        while(!bioskey(1)) {
                switch(getkey()) {
                        case F1 : mkwav();      break;
                        case F2 : mix();        break;
                        case F3 : mod();        break;
                        case F4 : draws();      break;
                        case F5 : am();         break;
                        case F6 : fm();         break;
                        case F7 : help1();      break;
                        case CR : help0();      break;
                        case ESC: Select();     break;
                        case AX : closegraph(); exit(1);
                        default : help2();  break;
                }
        }
        bioskey(0);
        Select();
}

mkwav()
{
        writetext(60,13,12,0,0,1,data1[0]);
        press1();       delay(90);
        writetext(60,13,4,0,0,1,data1[0]);
        press();
        mkwave();
}

mix()
{
        writetext(145,13,12,0,0,1,data1[1]);
        press1();       delay(90);
        writetext(145,13,4,0,0,1,data1[1]);
        press();
        mixer();
}

mod()
{
        writetext(225,13,12,0,0,1,data1[2]);
        press1();   delay(90);
        writetext(225,13,4,0,0,1,data1[2]);
        press();
        modulation();
}
```

```
am()
{
        writetext(380,13,12,0,0,1,data1[4]);
        press1();       delay(90);
        writetext(380,13,4,0,0,1,data1[4]);
        press();
        ammod();
}

fm()
{
        writetext(445,13,12,0,0,1,data1[5]);
        press1();       delay(90);
        writetext(445,13,4,0,0,1,data1[5]);
        press();
        fmmod();
}

draws()
{
        writetext(300,13,12,0,0,1,data1[3]);
        press1();       delay(90);
        writetext(300,13,4,0,0,1,data1[3]);
        press();
        plot();
        analysis();
}

press()
{
        upkey(17,17);         upkey(606,17);
        downkey(17,464);  downkey(606,464);
        leftkey(1,17);        leftkey(1,464);
        rightkey(623,17);   rightkey(623,464);
}

press1()
{
        upkeyi(17,17);         upkeyi(606,17);
        downkeyi(17,464);  downkeyi(606,464);
        leftkeyi(1,17);        leftkeyi(1,464);
        rightkeyi(623,17);   rightkeyi(623,464);
}

help0()
{
        bbxtrii(17,1);       bbxtrii(606,1);
        kmitsong();
        bbxtri(17,1);        bbxtri(606,1);
        bbx0(35,33,604,463,15,15,7);
}
```

```
help1()
{
        bbxesci(1,1);       bbxesci(623,1);
        delay(150);
        bbxesc(1,1);        bbxesc(623,1);
        Select();
}

help2()
{
        bbxesci(1,1);       bbxesci(623,1);
        bbxtrii(17,1);      bbxtrii(606,1);
        bbx0(150,115,236,135,15,8,7);
        writetext(155,120,12,2,0,4,"select F1...F7");
        sleep(1);
        bbxesc(1,1);        bbxesc(623,1);
        bbxtri(17,1);       bbxtri(606,1);
        bbx0(35,33,604,463,15,15,7);
}

help3()
{
        bbxesci(1,1);       bbxesci(623,1);
        delay(150);
        bbxesc(1,1);        bbxesc(623,1);
        Select();
}

void plot(void)
{
        DSP_FILE        *dsp_info,*dsp_info1,*dsp_info2;
        COMPLEX         *samp, *filt;
        int         i,j,k,length,fft_length,col,ma;
        int         mea,view,begin,center,center1,center2;
        float       *linearss,*signal,*signal1,*signal2,*log_mag1,m,a,b,cmx;
        double      amp,arg,twopi,wc,freq1,freq2,tempflt;

        bbx0(0,0,639,479,1,1,1);
        bbx0(64,36,576,444,15,15,0);
        bbxesc(5,5);        bbxtri(618,5);
        setlinestyle(DOTTED_LINE,1,NORM_WIDTH);
        setcolor(15);
        line(192,40,192,440);
        line(320,40,320,440);
        line(448,40,448,440);
        line(64,240,576,240);
        setlinestyle(SOLID_LINE,1,NORM_WIDTH);
        bbx0(100,9,540,33,15,8,0);

/* Read the input data file specified by the user */
        gotoxy(27,2);
        dsp_info = open_read(gettext("File name"));
        length = dsp_info->rec_len;
```

```
signal1 = (float *) calloc(length,sizeof(float));
if(!signal1) {
        bbx0(35,33,604-50,463,15,15,7);
        writetext(50,50,12,0,0,2,"Unable to allocate samples");
        free(signal1);
        beep();
        analysis();
}

signal2 = (float *) calloc(length,sizeof(float));
if(!signal2) {
        bbx0(35,33,604-50,463,15,15,7);
        writetext(50,50,12,0,0,2,"Unable to allocate samples");
        free(signal2);
        beep();
        analysis();
}
signal1 = read_float_record(dsp_info);
amx = -1000.0;    bmx = 1000.0;
for(i = 0; i < length; i++) {
        amx = MAX(amx,signal1[i]);
        bmx = MIN(bmx,signal1[i]);
}
free(value);
value[0] = amx;
value[1] = bmx;
value[2] = (value[0] + value[1])/2;
value[3] = value[0] - value[1];
value[4] = length;
for(i = 0; i < length; i++) {
        signal2[i] = (value[0] - signal1[i])*400/value[3] + 40;
}
xf=(double) (px)/length;
moveto(65,signal2[0]);
setcolor(14);
for(col = 0; col < length; col++) {
        lineto(dx(col)+64,signal2[col]);
}
writetextxy(25,35,12,0,0,1,"%1.2f",value[0]);
writetextxy(20,437,12,0,0,1,"%1.2f",value[1]);
writetextxy(25,235,12,0,0,1,"%1.2f",value[2]);
writetext(60,460,12,0,0,1,"0");
writetextxy(175,460,12,0,0,1,"%1.2f",value[4]/4);
writetextxy(305,460,12,0,0,1,"%1.2f",value[4]/2);
writetextxy(425,460,12,0,0,1,"%1.2f",value[4]*3/4);
writetextxy(545,460,12,0,0,1,"%1.2f",value[4]);

pauses();
bbxesci(5,5);       bbxtrii(618,5);
free(signal2);      free(value);
writetext(320,465,14,0,0,1,"Please Wait! 15 Second");
mea = log2(length);
fft_length = 1<<mea;
```

```
        samp = (COMPLEX *) calloc(fft_length, sizeof(COMPLEX));
        if(!samp){
                error();
                free(samp);
                analysis();
        }

        linearss = (float *) calloc(fft_length, sizeof(COMPLEX));
        if(!linearss){
                error();
                free(linearss);
                analysis();
        }
        for (i = 0 ; i < length ; i++) samp[i].real = signal1[i];
                han(samp,length);
        for (i = 0 ; i < length ; i++) signal1[i] = samp[i].real;


/* Find the spectrum of the data */
        fft(samp,mea);

/* do log magnitude */
        amp = 4.0/((double)length * length);
        for (i = 0 ; i < fft_length ; i++) {
                tempflt  = samp[i].real * samp[i].real;
                tempflt += samp[i].imag * samp[i].imag;
                tempflt *= amp;
                linearss[i] = tempflt;
                samp[i].real = 10 * log10(MAX(tempflt,1.e-14));
        }

/*  Copy a section of the spectrum to the magnitude file.
This allows a close up plot of a particular segment of the
spectral magnitude of the signal. */

        center = fft_length/2;
        view = fft_length;
        begin = center-view/2;
        log_mag1 = (float *) calloc(view,sizeof(float));
        if(!log_mag1){
                error();
                free(signal2);
                analysis();
        }
        for(k = 0 ; k < view ; k++){
                i = k + begin;
                if (i<0) i = 0;
                if (i>=fft_length) i = fft_length - 1;
                log_mag1[k] = samp[i].real;
        }
```

```
/* Read the input data file specified by the user */
        tektronix();
        length = view;
        lin1=-1000.0;       lin2=1000.0;
        for(i=0;i<length;i++) {
                lin1 = MAX(lin1,linearss[i]);
                lin2 = MIN(lin2,linearss[i]);
        }
        free(lin);
        lin[0] = lin1;
        lin[1] = lin2;
        lin[2] = (lin[0] + lin[1])/2;
        lin[3] = lin[0] - lin[1];

        for(i=0;i<length;i++) {
                linearss[i] = (lin[0]-linearss[i])*350/lin[3] + 80;
        }
        aaxf=(double) (aapx)/length;
        moveto(70,linearss[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(addx(col)+70,linearss[col]);
        }
        writetext(320,465,14,0,0,1,"linear scale");
        writetext(185,445,1,0,0,1,
        "<-- (-)freq    0Hz     (+)freq -->");
        getch();
        bbxesci(5,5);       bbxtrii(618,5);
        free(linearss);

/* Read the input data file specified by the user */
        tektronix();
        length = view;
        aamm = -1000.0; bbmm = 1000.0;
        for(i=0;i<length;i++) {
                aamm = MAX(aamm,log_mag1[i]);
                bbmm = MIN(bbmm,log_mag1[i]);
        }
        free(valu);
        valu[0] = aamm;
        valu[1] = bbmm;
        valu[2] = (valu[0] + valu[1])/2;
        valu[3] = valu[0] - valu[1];

        for(i=0;i<length;i++) {
                log_mag1[i] = (valu[0]-log_mag1[i])*350/valu[3] + 80;
        }
        moveto(70,log_mag1[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(addx(col)+70,log_mag1[col]);
        }
        writetext(320,465,14,0,0,1,"log scale");
        writetext(185,445,1,0,0,1,
```

```
            "<-- (-)freq    0Hz     (+)freq -->");
            getch();
            bbxesci(5,5);       bbxtrii(618,5);

            tektronix();
            aaxf=(double) (aapx)/center;
            moveto(70,log_mag1[center]);
            setcolor(14);
            for(col = 0; col < center; col++) {
                    lineto(addx(col)+70,log_mag1[col+center]);
            }
            writetext(320,465,1,0,0,1,"span (+)freqs");
            writetext(70,445,1,0,0,1,
            "0Hz      (+)freq -->");
            pauses();
            bbxesci(5,5);       bbxtrii(618,5);
            free(signal1);      free(valu);
            free(log_mag1); free(samp);
            analysis();
}

int dx(int vx)
{
            return((int) (xf*vx));
}

int addx(int vx)
{
            return((int) (aaxf*vx));
}

int getkey()
{
                    int ch01;
                            ch01 = getch();
                            if (ch01 == 0)
                                    ch01 = getch();
                            return ch01;
}

char *gettext(char *title_string)
{
   char *alpha;
   alpha = (char *) malloc(80);
   if(!alpha) {
           writetext(200,100,0,0,0,1,"String allocation error in get_string");
   }
   printf("Enter %s : ",title_string);
   gets(alpha);
   return(alpha);
}
/*****************************<end of analysis.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : dft.c
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include   <math.h>
#include   <stdlib.h>
#include   <stdio.h>
#include   "dft.h"

/*****************************************************************************

DFT.C - SOURCE CODE FOR DISCRETE FOURIER TRANSFORM FUNCTIONS

fft    In-place radix 2 decimation in time FFT
rfft   Trig recombination real input FFT
han    Hanning window
log2   Base 2 logarithm

*****************************************************************************/

/*****************************************************************************

fft - In-place radix 2 decimation in time FFT

Requires pointer to complex array, x and power of 2 size of FFT, m
(size of FFT = 2**m).  Places FFT output on top of input COMPLEX array.

void fft(COMPLEX *x, int m)

*****************************************************************************/

void fft(x,m)
  COMPLEX *x;
  int m;
{
        static COMPLEX *w;      /* used to store the w complex array */
        static int mstore = 0;  /* stores m for future reference     */
        static int n = 1;       /* length of fft stored for future   */

        COMPLEX u,temp,tm;
        COMPLEX *xi,*xip,*xj,*wptr;

        int i,j,k,l,le,windex;

        double arg,w_real,w_imag,wrecur_real,wrecur_imag,wtemp_real;

        if(m != mstore) {
```

```
/* free previously allocated storage and set new m */
                    if(mstore != 0) free(w);
                    mstore = m;
                    if(m == 0) return;              /* if m=0 then done */

/* n = 2**m = fft length */
                    n = 1 << m;
                    le = n/2;

/* allocate the storage for w */
                    w = (COMPLEX *) calloc(le-1,sizeof(COMPLEX));
                    if(!w) {
                              tektronix();
                              error();
                              free(w);
                              sleep(1);
                              analysis();
                    }

/* calculate the w values recursively */
                    arg = 4.0*atan(1.0)/le;         /* PI/le calculation */
                    wrecur_real = w_real = cos(arg);
                    wrecur_imag = w_imag = -sin(arg);
                    xj = w;
                    for (j = 1 ; j < le ; j++) {
                        xj->real = (float)wrecur_real;
                        xj->imag = (float)wrecur_imag;
                        xj++;
                        wtemp_real = wrecur_real*w_real - wrecur_imag*w_imag;
                        wrecur_imag = wrecur_real*w_imag + wrecur_imag*w_real;
                        wrecur_real = wtemp_real;
                    }
          }

/* start fft */
          le = n;
          windex = 1;
          for (l = 0 ; l < m ; l++) {
                    le = le/2;

/* first iteration with no multiplies */
                    for(i = 0 ; i < n ; i = i + 2*le) {
                              xi = x + i;
                              xip = xi + le;
                              temp.real = xi->real + xip->real;
                              temp.imag = xi->imag + xip->imag;
                              xip->real = xi->real - xip->real;
                              xip->imag = xi->imag - xip->imag;
                              *xi = temp;
                    }

/* remaining iterations use stored w */
                    wptr = w + windex - 1;
                    for (j = 1 ; j < le ; j++) {
```

```
                        u = *wptr;
                        for (i = j ; i < n ; i = i + 2*le) {
                                xi = x + i;
                                xip = xi + le;
                                temp.real = xi->real + xip->real;
                                temp.imag = xi->imag + xip->imag;
                                tm.real = xi->real - xip->real;
                                tm.imag = xi->imag - xip->imag;
                                xip->real = tm.real*u.real - tm.imag*u.imag;
                                xip->imag = tm.real*u.imag + tm.imag*u.real;
                                *xi = temp;
                        }
                        wptr = wptr + windex;
                }
                windex = 2*windex;
        }

/* rearrange data by bit reversing */
        j = 0;
        for (i = 1 ; i < (n-1) ; i++) {
                k = n/2;
                while(k <= j) {
                j = j - k;
                k = k/2;
    }
    j = j + k;
    if (i < j) {
                xi = x + i;
                xj = x + j;
                temp = *xj;
                *xj = *xi;
                *xi = temp;
    }
  }
}

/*****************************************************************************

rfft - trig recombination real input FFT

Requires real array pointed to by x, pointer to complex
output array, y and the size of real FFT in power of
2 notation, m (size of input array and FFT, N = 2**m).
On completion, the COMPLEX array pointed to by y
contains the lower N/2 + 1 elements of the spectrum.

void rfft(float *x, COMPLEX *y, int m)

*****************************************************************************/

void rfft(x,y,m)
  float    *x;
  COMPLEX  *y;
  int      m;
```

```
{
        static   COMPLEX  *cf;
        static   int      mstore = 0;
        int      p,num,k,index;
        float    Realsum, Realdif, Imagsum, Imagdif;
        double   factor, arg;
        COMPLEX  *ck, *xk, *xnk, *cx;

/* First call the fft routine using the x array but with
  half the size of the real fft */

        p = m - 1;
        cx = (COMPLEX *) x;
        fft(cx,p);

/* Next create the coefficients for recombination, if required */

        num = 1 << p;        /* num is half the real sequence length.  */

        if (m!=mstore){
                if (mstore != 0) free(cf);
                cf = (COMPLEX *) calloc(num - 1,sizeof(COMPLEX));
                if(!cf){
                        tektronix();
                        error();
                        free(cf);
                        analysis();
                }

                factor = 4.0*atan(1.0)/num;
                for (k = 1; k < num; k++){
                        arg = factor*k;
                        cf[k-1].real = (float)cos(arg);
                        cf[k-1].imag = (float)sin(arg);
                }
        }

/* DC component, no multiplies */
        y[0].real = cx[0].real + cx[0].imag;
        y[0].imag = 0.0;

/* other frequencies by trig recombination */
        ck = cf;
        xk = cx + 1;
        xnk = cx + num - 1;
        for (k = 1; k < num; k++){
        Realsum = ( xk->real + xnk->real ) / 2;
        Imagsum = ( xk->imag + xnk->imag ) / 2;
        Realdif = ( xk->real - xnk->real ) / 2;
        Imagdif = ( xk->imag - xnk->imag ) / 2;

        y[k].real = Realsum + ck->real * Imagsum - ck->imag * Realdif;

        y[k].imag = Imagdif - ck->imag * Imagsum - ck->real * Realdif;
```

```
        ck++;
        xk++;
        xnk--;
    }
}

/*****************************************************************************

han - Hanning window

Scales both real and imaginary parts of input array in-place.
Requires COMPLEX pointer and length of input array.

void han(COMPLEX *x, int n)

*****************************************************************************/

void han(x, n)
 COMPLEX   *x;
 int       n;
{
        int   i;
        double factor,han;

        factor = 8.0*atan(1.0)/(n-1);
        for (i = 0 ; i < n ; i++){
                han = 0.5 - 0.5*cos(factor*i);
                x->real *= han;
                x->imag *= han;
                x++;
        }
}

/*****************************************************************************
log2 - base 2 logarithm

Returns base 2 log such that i = 2**ans where ans = log2(i).
if log2(i) is between two values, the larger is returned.
int log2(unsigned int x)
*****************************************************************************/
int log2(x)
  unsigned int x;
{
        unsigned int mask,i;
        if(x == 0) return(-1);          /* zero is an error, return -1 */
        x--;                        /* get the max index, x-1     */
        for(mask = 1 , i = 0 ; ; mask *= 2 , i++) {
                if(x == 0) return(i);     /* return log2 if all zero     */
                x = x & (~mask);          /* AND off a bit               */
        }
}
/*****************************<end of dft.c>*****************************/
```

```
/*****************************************************************************
FILE NAME :  DISKIO.C
progammer  :  WEERASAK  NAMVONG
Create     :  25 / Feb / 1992
Update     :  19 / may / 1992
Purpose    :  Prototype function for CCAD.EXE
Comment    :  This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/*      DSP INFORMATION STRUCTURE FOR MANIPULATING DSP DATA FILES       */
typedef struct {
        unsigned char type;             /* data type 0-7 as defined below      */
        unsigned char element_size;     /* size of each element                */
        unsigned short int records;     /* number of records                   */
        unsigned short int rec_len;     /* number of elements in each record */
        char *name;                     /* pointer to file name                */
        FILE *fp;                       /* pointer to FILE structure           */
} DSP_FILE;

/*      FILE HEADER STRUCTURE FOR DSP DATA FILES                        */
typedef struct {
        unsigned char type;             /* data type 0-7 as defined below      */
        unsigned char element_size;     /* size of each element                */
        unsigned short int records;     /* number of records                   */
        unsigned short int rec_len;     /* number of elements in each record */
} HEADER;

/* defines for data type used ind DSP data file header and structure    */

#define UNSIGNED_CHAR   0
#define UNSIGNED_INT    1
#define UNSIGNED_LONG   2
#define FLOAT           3
#define SIGNED_CHAR     4
#define SIGNED_INT      5
#define SIGNED_LONG     6
#define DOUBLE          7

/*****************************************************************************
open_read - open a DSP data file for read

DSP_FILE *open_read(char *file_name)
*****************************************************************************/
DSP_FILE *open_read(file_name)
  char *file_name;              /* file name string      */
{
        DSP_FILE *dsp_info;
        int status;
```

```
/* allocate the DSP data file structure */
        dsp_info = (DSP_FILE *) malloc(sizeof(DSP_FILE));
        if(!dsp_info) {
          writetextxy(100,50,10,0,0,1,"Error in openread file %s",file_name);
          free(dsp_info);  beep();
          plot();
        }

/* open file for binary read and update */
        dsp_info->fp = fopen(file_name,"r+b");
        if(!dsp_info->fp) {
                writetextxy(100,50,10,0,0,1,"Error opening %s in open_read",
                file_name);
                free(dsp_info);
                beep();
                plot();
        }

/* copy and allocate file name string for the DSP_FILE structure */
        dsp_info->name = malloc(strlen(file_name) + 1);
        if(!dsp_info->name) {
                writetext(100,100,0,0,0,1,
                "Unable to allocate file_name string in open_read");
                free(dsp_info->name);
                free(dsp_info);
                beep();
        }
        strcpy(dsp_info->name,file_name);

/* read in header from file */
        status = fread((char *)dsp_info,sizeof(HEADER),1,dsp_info->fp);
        if(status != 1) {
            writetextxy(100,100,0,0,0,1,"Error reading header of file %s",
            file_name);
            beep();
        }

/* return pointer to DSP_FILE structure */
        return(dsp_info);
        free(dsp_info);
        free(dsp_info->name);
}

/*****************************************************************************
open_write - open a DSP data file for write

DSP_FILE *open_write(char *file_name,int type,int records,int rec_len)
  file_name     pointer to file name string
  type          type of DSP data (0-7 specified in defines)
  records       number of records of data to be written
  rec_len       number of elements in each record
*****************************************************************************/
DSP_FILE *open_write(file_name,type,records,rec_len)
  char *file_name;                  /* file name string */
```

```
  int type;                          /* data type 0-7                    */
  unsigned short int records;        /* number of records to be written */
  unsigned short int rec_len;        /* elements in each record          */
{
        DSP_FILE *dsp_info;
        int status;

/* allocate the DSP data file structure */
        dsp_info = (DSP_FILE *) malloc(sizeof(DSP_FILE));
        if(!dsp_info) {
           writetextxy(70,70,12,0,0,1,
           "Error in open_write: structure allocation, file %s",file_name);
           free(dsp_info);          beep();
           analysis();
        }

/* set the basics */
        dsp_info->type = (unsigned char)type;
        dsp_info->records = records;
        dsp_info->rec_len = rec_len;

/* set element size from data type */
        switch(type) {
                case 0:
                case 4:
                        dsp_info->element_size = sizeof(char);
                        break;
                case 1:
                case 5:
                        dsp_info->element_size = sizeof(short int);
                        break;
                case 2:
                case 6:
                        dsp_info->element_size = sizeof(long int);
                        break;
                case 3:
                        dsp_info->element_size = sizeof(float);
                        break;
                case 7:
                        dsp_info->element_size = sizeof(double);
                        break;
                default:
                        writetextxy(70,70,12,0,0,1,
                        "Unsupported data type, file %s",file_name);
                        beep();
                        analysis();
        }

 /* open file for binary write */
        dsp_info->fp = fopen(file_name,"wb");
        if(!dsp_info->fp) {
                writetextxy(70,50,11,0,0,1,
                "Error opening %s in open_write",file_name);
                free(dsp_info);
```

```
                beep();
                analysis();
        }

/* copy and allocate file name string for the DSP_FILE structure */
        dsp_info->name = malloc(strlen(file_name) + 1);
        if(!dsp_info->name) {
                writetext(70,50,11,0,0,1,
                "Unable to allocate file_name string in open_write");
                free(dsp_info->name);
                free(dsp_info);
                beep();
                analysis();
        }
        strcpy(dsp_info->name,file_name);

/* write header to file */
        status = fwrite((char *)dsp_info,sizeof(HEADER),1,dsp_info->fp);
        if(status != 1) {
                writetextxy(70,50,11,0,0,1,
                "Error writing header of file %s",file_name);
                free(dsp_info);
                free(dsp_info->name);
                beep();
                analysis();
        }

/* return pointer to DSP_FILE structure */
        return(dsp_info);
        free(dsp_info);
        free(dsp_info->name);
}

/*****************************************************************************
read_record - read one record of DSP data file

void read_record(char *ptr,DSP_FILE *dsp_info)
  ptr       pointer to previously allocated memory to put data
  dsp_info  pointer to DSP data file structure
*****************************************************************************/
void read_record(ptr,dsp_info)
  char *ptr;              /* pointer to some type of data */
  DSP_FILE *dsp_info;
{
        int status;

        if(!dsp_info) {
                writetext(100,100,0,0,0,1,
                "Error in DSP_FILE structure passed to read_record");
                beep();
                analysis();
        }

        status = fread(ptr,dsp_info->element_size,
```

```
                dsp_info->rec_len,dsp_info->fp);
        if(status != dsp_info->rec_len) {
                writetextxy(100,100,0,0,0,1,"Error in read_record, file %s",
                dsp_info->name);
                beep();
                analysis();
        }
}
/*****************************************************************************
write_record - write one record of DSP_FILE data

void write_record(char *ptr,DSP_FILE *dsp_info)
  ptr       pointer to data to write to disk (type in dsp_info)
  dsp_info  pointer to DSP data file structure
*****************************************************************************/
void write_record(ptr,dsp_info)
  char *ptr;              /* pointer to some type of data */
  DSP_FILE *dsp_info;
{
        int status;

        if(!dsp_info) {
                writetext(100,100,0,0,0,1,
                "Error in DSP_FILE structure passed to write_record");
                beep();
                analysis();
        }

        status = fwrite(ptr,dsp_info->element_size,
                dsp_info->rec_len,dsp_info->fp);
        if(status != dsp_info->rec_len) {
                writetextxy(100,100,0,0,0,1,
                "Error write_record, file %s",dsp_info->name);
                beep();  analysis();
        }
}

/*****************************************************************************
seek_record - seek to the beginning of a record of DSP data file

void seek_record(int record_num,DSP_FILE *dsp_info)
  record_num   integer record number to seek in DSP data file
  dsp_info     pointer to DSP data file structure
*****************************************************************************/
void seek_record(record_num,dsp_info)
  int record_num;         /* record number to seek */
  DSP_FILE *dsp_info;
{
        long int position,bytecount;
        if(!dsp_info) {
                writetext(100,100,0,0,0,1,
                "Error in DSP_FILE structure passed to seek_record");
                beep();
                analysis();
```

```
        }

/* get the number of bytes to the beginning of the record of data */
        bytecount = (long)dsp_info->element_size * (long)dsp_info->rec_len
                                                    * (long)record_num;
        bytecount += sizeof(HEADER);        /* add on the header length */

/* find the end of file location */
        fseek(dsp_info->fp,0L,2);           /* seek to end of file */
        position = ftell(dsp_info->fp);

/* check for errors in position */
        if(position <= 0L || position < bytecount) {
                writetextxy(100,100,0,0,0,1,
                "Error in locating record in file %s",dsp_info->name);
                beep();             analysis();
        }
        fseek(dsp_info->fp,bytecount,0);      /* move to record */
}

/*****************************************************************************
read_float_record - read one record of DSP data file and convert
        to float array of values.

float *read_float_record(DSP_FILE *dsp_info)
*****************************************************************************/
float *read_float_record(dsp_info)
  DSP_FILE *dsp_info;
{
        void read_record();
        static long int prev_size = 0; /* previous size in bytes */
        static double *buf;         /* input buffer to read data in */

        float *out;                 /* return output pointer */
        float *out_ptr;

        long int byte_size;         /* current size in bytes */
        int i,length;

        length = dsp_info->rec_len;

        byte_size = (long)length*dsp_info->element_size;

/* check to see if we have to allocate the input buffer */
        if(byte_size != prev_size) {

                if(prev_size != 0) free(buf);  /* free old buffer */

/* allocate input buffer area cast to double for worst case alignment */
                buf = (double *) calloc(length,dsp_info->element_size);

                if(!buf) {
                        writetext(100,100,0,0,0,1,
                        "Allocation error in input buffer");
```

```
                        free(buf);
                        beep();
                        analysis();
                }

                prev_size = byte_size;     /* lattest size */
        }

/* allocate the output pointer only if conversion required */
        if(dsp_info->type != FLOAT) {
                out = (float *) calloc(length,sizeof(float));
                if(!out) {
                        writetext(100,100,0,0,0,1,
                        "Allocation error in read_float_record");
                        free(buf);
                        free(out);
                        beep();
                        analysis();
                }
        }

/* read the record into buf */
        read_record((char *)buf,dsp_info);

/* perform conversion to floating point */
        out_ptr = out;
        switch(dsp_info->type) {
                case UNSIGNED_CHAR: {
                        unsigned char *uc_ptr;
                        uc_ptr = (unsigned char *)buf;
                        for(i = 0 ; i < length ; i++)
                                *out_ptr++ = (float)(*uc_ptr++);
                }
                break;
                case SIGNED_CHAR: {
                        char *sc_ptr;
                        sc_ptr = (char *)buf;
                        for(i = 0 ; i < length ; i++)
                                *out_ptr++ = (float)(*sc_ptr++);
                }
                break;
                case SIGNED_INT: {
                        int *si_ptr;
                        si_ptr = (int *)buf;
                        for(i = 0 ; i < length ; i++)
                                *out_ptr++ = (float)(*si_ptr++);
                }
                break;
                case UNSIGNED_INT: {
                        unsigned int *ui_ptr;
                        ui_ptr = (unsigned int *)buf;
                        for(i = 0 ; i < length ; i++)
                        *out_ptr++ = (float)(*ui_ptr++);
                }
```

```
                break;
                case UNSIGNED_LONG: {
                        unsigned long *ul_ptr;
                        ul_ptr = (unsigned long *)buf;
                        for(i = 0 ; i < length ; i++)
                                *out_ptr++ = (float)(*ul_ptr++);
                }
                break;
                case SIGNED_LONG: {
                        long *sl_ptr;
                        sl_ptr = (long *)buf;
                        for(i = 0 ; i < length ; i++)
                        *out_ptr++ = (float)(*sl_ptr++);
                }
                break;
                case FLOAT:
                        out = (float *) buf;    /* no conversion          */
                        prev_size = 0;          /* force next allocation */
                break;
                case DOUBLE: {
                        double *d_ptr;
                        d_ptr = buf;
                        for(i = 0 ; i < length ; i++)
                                *out_ptr++ = (float)(*d_ptr++);
                }
                break;
        }

        return(out);            /* return converted pointer */
        free(buf);
        free(out);
}
/*****************************<end of diskio.c>*****************************/
```

```
/*****************************************************************************
FILE NAME : MKWAVE.C
progammer  : WEERASAK  NAMVONG
Create     : 25 / Feb / 1992
Update     : 19 / may / 1992
Purpose    : Prototype function for CCAD.EXE
Comment    : This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*****************************************************************************/
#include        <graphics.h>
#include        <stdlib.h>
#include        <stdio.h>
#include        <string.h>
#include        <alloc.h>
#include        <conio.h>
#include        <math.h>
#include        "allkey.h"
#include        "disk.h"
#include        "get.h"
#include        "filter.h"
#include        "dft.h"

void    si(void);
void    co(void);
void    sq(void);
void    tr(void);
void    sa(void);
void    ha(void);
void    pu(void);
void    gu(void);
void    mixer(void);
void    ammod(void);
void    fmmod(void);
void    mkwave(void);
void    modulation(void);

char    *CHR;
double  FLOA,axf;
float   *values,amm,bmm,linear1,linear2,*linear;
int     INT,keyin,sam,apx=500;
float  *sig11,*sig22;

void mkwave(void)
{
        DSP_FILE *dsp_info;
        int i,status,nfreqs,length;
        float *signal;
        float freqs[10];            /* max 10 frequencies              */
        char prompt[10];            /* used to prompt user for input*/

        float *wave();
        window31(43,42);
        writetext(230,4,14,2,0,4,"Application Windows MKWAVE");
```

```
        writetext(50,100,12,0,0,1,
        "Enter number of samples to generate [2...10000] :");
        getscanf(455,100,515,6,0,0,1,"%s",CHR);
        length=INT;

        writetext(50,120,12,0,0,1,
        "Enter number of frequencies in sum [1...10] :");
        getscanf(455,120,515,6,0,0,1,"%s",CHR);
        nfreqs=INT;

        for(i = 0 ; i < nfreqs ; i++) {
        writetextxy(50,140+20*i,12,0,0,1,
        "Enter frequency #%d  [0...0.5] :",i);
                getscanf(455,140+20*i,515,6,0,0,1,"%s",CHR);
                freqs[i] = FLOA;
        }

        signal = wave(length, freqs, nfreqs);

        writetext(50,160+20*nfreqs,12,0,0,1,
        "Enter file name  :");
        getscanf(330,160+20*nfreqs,515,6,0,0,1,"%s",CHR);

        dsp_info = open_write(CHR,FLOAT,1,length);
        if(!dsp_info)  {
                beep();          analysis();
        }
        write_record((char *)signal,dsp_info);
        free(signal);
        analysis();
}

void mixer(void)
{
        DSP_FILE *dsp_info33;
        int i,length1,length2;
        double *signals;

        float *fwave();
        window31(43,42);
        writetext(230,4,14,2,0,4,"Application Windows MIXER");

        writetext(50,100,12,0,0,1,"Enter  MIX  File name  #1 :");
        getscanf(350,100,515,6,0,0,1,"%s",CHR);
        sig11 = fwave(CHR);
        length1 = sam;

        writetext(50,120,12,0,0,1,"Enter  MIX  File name  #2 :");
        getscanf(350,120,515,6,0,0,1,"%s",CHR);
        sig22 = fwave(CHR);
        length2 = sam;

        signals = (double *) calloc(sam,sizeof(float));
        if(!signals) {
```

```
                error();
                free(signals);
                analysis();
        }
        if(length1!=length2) {
                bbx0(35,33,604-50,463,15,15,7);
                writetext(50,50,12,0,0,2,"Unable to allocate samples");
                beep();         free(signals);
                analysis();
        }
        if(length1==length2) {
                for(i = 0; i < sam; i++)  {
                        signals[i] = sig11[i] + sig22[i];
                }
        }
        writetext(50,200,12,0,0,1,"Enter file name  :");
        getscanf(350,200,515,6,0,0,1,"%s",CHR);

        dsp_info33 = open_write(CHR,FLOAT,1,sam);
        if(!dsp_info33) {
                beep();         analysis();
        }
        write_record((char *)signals,dsp_info33);
        free(signals);

        analysis();
}

void modulation(void)
{
        DSP_FILE *dsp_info33;
        int i,length1,length2;
        double *signals;

        float *fwave();
        window31(43,42);
        writetext(227,4,14,2,0,4,"Application Windows MODULATION");

        writetext(50,100,12,0,0,1,"Enter SIGNAL File name  :");
        getscanf(350,100,515,6,0,0,1,"%s",CHR);
        sig11 = fwave(CHR);
        length1 = sam;

        writetext(50,120,12,0,0,1,"Enter  MOD   File name  :");
        getscanf(350,120,515,6,0,0,1,"%s",CHR);
        sig22 = fwave(CHR);
        length2 = sam;
        signals = (double *) calloc(sam,sizeof(float));
        if(!signals) {
                bbx0(35,33,604-50,463,15,15,7);
                writetext(50,50,12,0,0,2,"Unable to allocate samples");
                beep();         free(signals);
                analysis();
        }
```

```
        if(length1!=length2) {
                bbx0(35,33,604-50,463,15,15,7);
                writetext(50,50,12,0,0,2,"Unable to allocate samples");
                beep();                 free(signals);
                analysis();
        }
        if(length1==length2) {
                for(i = 0; i < sam; i++)  {
                        signals[i] = sig11[i] * sig22[i];
                }
        }
        writetext(50,200,12,0,0,1,"Enter file name  :");
        getscanf(350,200,515,6,0,0,1,"%s",CHR);

        dsp_info33 = open_write(CHR,FLOAT,1,sam);
        if(!dsp_info33) {
                beep();                 analysis();
        }
        write_record((char *)signals,dsp_info33);
        free(signals);
        analysis();
}

void ammod(void)
{
        DSP_FILE        *dsp_info,*dsp_info1,*dsp_info2;
        COMPLEX         *samp, *filt;
        int             i,j,k,length,fft_length,col,ma;
        int             mea,view,begin,center,center1,status;
        float           *linears,*signal,*signal1,*signal2,*log_mag1,m,a,cmx;
        double          amp,arg,twopi,wc,freq1,freq2,tempflt;

        window31(43,42);
        writetext(225,4,14,2,0,4,"Application Windows AM SYSTEM");

        writetext(80,130,5,1,0,2,
        "Re{_AM}= (1+   cos    ) cos");
        writetext(160,130,13,1,0,2,
        "A   m    (Wm)    (Wc)");

        writetext(40,200,12,0,0,1,
        "Enter number of AMPLITUDE of signal [1...100] mV :");
        getscanf(455,200,515,6,0,0,1,"%s",CHR);
        a=FLOA;

        writetext(40,220,12,0,0,1,
        "Enter PERCCENTAGE of modulation    [1...150] % :");
        getscanf(455,220,515,6,0,0,1,"%s",CHR);
        m = FLOA/100;

        writetext(40,240,12,0,0,1,
        "Enter input FREQUENCY baseband   [1...15000] Hz :");
        getscanf(455,240,515,6,0,0,1,"%s",CHR);
        freq2=FLOA;
```

```
writetext(40,260,12,0,0,1,
"Enter input FREQUENCY carrier   [540...1600] kHz:");
getscanf(455,260,515,6,0,0,1,"%s",CHR);
freq1=FLOA;

signal1 = (float *) calloc(1000,sizeof(float));
if(!signal1) {
        error();
        free(signal1);
        analysis();
}
twopi = 0.025/freq2;
wc = twopi*freq1*1000;
arg = twopi*freq2;
amm=-1000.0;    bmm=1000.0;
for (i = 0 ; i < 1000 ; i++) {
        signal1[i]=a*cos(wc*i)*(1+m*cos(arg*i));
        amm = MAX(amm,signal1[i]);
        bmm = MIN(bmm,signal1[i]);
}
free(values);
values[0] = amm;
values[1] = bmm;
values[2] = (values[0] + values[1])/2;
values[3] = values[0]-values[1];

signal2 = (float *) calloc(1000,sizeof(float));
if(!signal2) {
        error();
        free(signal2);
        analysis();
}

for(j=0;j<1000;j++) {
        signal2[j] = (values[0]-signal1[j])*400/values[3] + 40;
}
setviewport(0,0,639,479,0);
bbx0(0,0,639,479,1,1,1);
bbx0(69,36,571,444,15,15,0);
bbxesc(5,5);        bbxtri(618,5);
setlinestyle(1,1,NORM_WIDTH);
setcolor(15);
line(195,40,195,440);
line(320,40,320,440);
line(445,40,445,440);
line(71,240,569,240);
writetext(80,10,12,0,0,2,"KENV");
writetext(136,10,12,0,0,2,"VOOD");
writetext(240,13,12,2,0,5,"DIGITAL STORAGE OSCILLOSCOPE");
writetext(485,10,12,2,0,6,"CS - 8010");
setlinestyle(SOLID_LINE,1,NORM_WIDTH);

axf=(double) (apx)/1000;
moveto(70,signal2[0]);
```

```
        setcolor(14);
        for(col = 0; col < 1000; col++) {
                lineto(dxx(col)+70,signal2[col]);
        }
        writetextxy(10,45,11,0,0,2,"mV");
        writetextxy(590,440,11,0,0,2,"S");
        writetextxy(5,35,14,0,0,1,"%1.1f",values[0]);
        writetextxy(5,437,14,0,0,1,"%1.1f",values[1]);
        writetextxy(5,235,14,0,0,1,"%1.1f",values[2]);
        writetext(60,460,14,0,0,1,"0");
        writetextxy(182,460,14,0,0,1,"%g",1/freq2);
        writetextxy(310,460,14,0,0,1,"%g",2/freq2);
        writetextxy(438,460,14,0,0,1,"%g",3/freq2);
        writetextxy(566,460,14,0,0,1,"%g",4/freq2);
        getch();
        free(signal2);  free(values);

        writetext(320,465,12,0,0,1,"Please Wait! 15 Second");
        length=1000;
        mea = log2(length);
        fft_length = 1<<mea;

        samp = (COMPLEX *) calloc(fft_length, sizeof(COMPLEX));
        if(!samp){
                error();
                free(samp);
                analysis();
        }

        linears = (float *) calloc(fft_length, sizeof(COMPLEX));
        if(!linears){
                error();
                free(linears);
                analysis();
        }
        for (i = 0 ; i < length ; i++) samp[i].real = signal1[i];
                han(samp,length);
        for (i = 0 ; i < length ; i++) signal1[i] = samp[i].real;


/* Find the spectrum of the data */
        fft(samp,mea);

/* do log magnitude */

        amp = 4.0/((double)length * length);
        for (i = 0 ; i < fft_length ; i++) {

                tempflt  = samp[i].real * samp[i].real;
                tempflt += samp[i].imag * samp[i].imag;
                tempflt *= amp;
                linears[i] = tempflt;
                samp[i].real = 10 * log10(MAX(tempflt,1.e-14));
        }
```

```
/*  Copy a section of the spectrum to the magnitude file.
This allows a close up plot of a particular segment of the
spectral magnitude of the signal. */

        center = fft_length/2;
        view = fft_length;
        begin = center-view/2;
        log_mag1 = (float *) calloc(view,sizeof(float));
        if(!log_mag1){
                error();
                free(signal2);
                analysis();
        }
        for(k = 0 ; k < view ; k++){
                i = k + begin;
                if (i<0) i = 0;
                if (i>=fft_length) i = fft_length - 1;
                log_mag1[k] = samp[i].real;
        }


/* Read the input data file specified by the user */
        tektronix();
        length = view;
        linear1=-1000.0;  linear2=1000.0;
        for(i=0;i<length;i++) {
                linear1 = MAX(linear1,linears[i]);
                linear2 = MIN(linear2,linears[i]);
        }
        free(linear);
        linear[0] = linear1;
        linear[1] = linear2;
        linear[2] = (linear[0] + linear[1])/2;
        linear[3] = linear[0] - linear[1];

        for(i=0;i<length;i++) {
                linears[i] = (linear[0]-linears[i])*350/linear[3] + 80;
        }
        axf=(double) (apx)/length;
        moveto(70,linears[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(dxx(col)+70,linears[col]);
        }
        writetext(320,465,14,0,0,1,"linear scale");
        writetext(185,445,1,0,0,1,
        "<-- (-)freq    0Hz    (+)freq -->");
        getch();
        bbxesci(5,5);       bbxtrii(618,5);
        free(linears);

/* Read the input data file specified by the user */
        tektronix();
        length = view;
```

```
        amm = -1000.0; bmm = 1000.0;
        for(i=0;i<length;i++) {
                amm = MAX(amm,log_mag1[i]);
                bmm = MIN(bmm,log_mag1[i]);
        }
        free(values);
        values[0] = amm;
        values[1] = bmm;
        values[2] = (values[0] + values[1])/2;
        values[3] = values[0] - values[1];

        for(i=0;i<length;i++) {
                log_mag1[i] = (values[0]-log_mag1[i])*350/values[3] + 80;
        }
        moveto(70,log_mag1[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(dxx(col)+70,log_mag1[col]);
        }
        writetext(320,465,14,0,0,1,"log scale");
        writetext(185,445,1,0,0,1,
        "<-- (-)freq    0Hz    (+)freq -->");
        getch();
        bbxesci(5,5);      bbxtrii(618,5);

first:  tektronix();
        axf=(double) (apx)/center;
        moveto(70,log_mag1[center]);
        setcolor(14);
        for(col = 0; col < center; col++) {
                lineto(dxx(col)+70,log_mag1[col+center]);
        }
        writetext(320,465,14,0,0,1,"(+)freqs");
        bbxesci(5,5);      bbxtrii(618,5);
        center1 = center/8;
        axf=(double) (apx)/center1;
        status = 0;
        do {
                keyin = getch();
                if(keyin == 0) {
                        keyin = getch();
                        if((keyin == 75) || (keyin == 77)) {

                                if(keyin == 75) {
                                status = (status == 0 ) ? 7 : status - 1;
                                tektronix();
                                moveto(70,log_mag1[center1*status]);
                                setcolor(14);
                                for(col = 0; col < center1; col++) {
                                        i = center1*status+col;
                                        lineto(dxx(col)+70,log_mag1[i]);
                                }
                                writetextxy(320,465,14,0,0,1,
                                        "span (%d)8freqs",status+1);
```

```
                                        bbxesci(5,5);       bbxtrii(618,5);
                                        }

                                        if(keyin == 77) {
                                        status = (status == 7 ) ? 0 : status + 1;
                                        tektronix();
                                        moveto(70,log_mag1[center1*status]);
                                        setcolor(14);
                                        for(col = 0; col < center1; col++) {
                                                i = center1*status+col;
                                                lineto(dxx(col)+70,log_mag1[i]);
                                        }
                                        writetextxy(320,465,14,0,0,1,
                                                "span (%d)8freqs",status+1);
                                        bbxesci(5,5);       bbxtrii(618,5);
                                        }
                                }
                        }
                        if(keyin == 27 || keyin == 72) {
                                goto first;
                        }
                        if(keyin == 13) {
                                bbxesci(5,5);       bbxtrii(618,5);
                                free(signal1);      free(values);
                                free(log_mag1); free(samp);
                                analysis();
                        }
                } while((keyin != 13));
                pauses();
                analysis();
}

void fmmod(void)
{
        DSP_FILE        *dsp_info,*dsp_info1,*dsp_info2;
        COMPLEX     *samp, *filt;
        int         i,j,k,length,fft_length,col,ma,status,status1;
        int         mea,view,begin,center,center1,center2,keyout,dirf;
        float       *linears,*signal,*signal1,*signal2,*log_mag1,m,a,b,cmx;
        double      amp,arg,twopi,wc,freq1,freq2,tempflt;

        window31(43,42);
        writetext(225,4,14,2,0,4,"Application Windows FM SYSTEM");

        writetext(80,130,5,1,0,2,
        "Re{_FM}= cos(    + sin      )");
        writetext(165,130,13,1,0,2,
        "A    (Wc)  B    (Wm)");

        writetext(50,200,12,0,0,1,
        "Enter number of AMPLITUDE of signal [1...100]mV :");
        getscanf(455,200,515,6,0,0,1,"%s",CHR);
        a=FLOA;
```

```
        writetext(50,220,12,0,0,1,
        "Enter modulation index         [β = 0...12.1] :");
        getscanf(455,220,515,6,0,0,1,"%s",CHR);
        b = FLOA;

        writetext(50,240,12,0,0,1,
        "Enter carrier FREQUENCY      [88...108]MHz :");
        getscanf(455,240,515,6,0,0,1,"%s",CHR);
        freq1=INT;

        writetext(50,260,12,0,0,1,
        "Enter baseband FREQUENCY      [1...15000]Hz :");
        getscanf(455,260,515,6,0,0,1,"%s",CHR);
        freq2=INT;

        signal1 = (float *) calloc(1000,sizeof(float));
        if(!signal1) {
                error();
                free(signal1);
                analysis();
        }
        twopi = 0.0000025;          /*  calculate 2*PI */
        wc = twopi*freq2;
        arg = 1000000*twopi*freq1;
        amm=-1000.0;    bmm=1000.0;
        for (i = 0 ; i < 1000 ; i++) {
                signal1[i]=a*cos((wc*i)+b*sin(arg*i));
                amm = MAX(amm,signal1[i]);
                bmm = MIN(bmm,signal1[i]);
        }
        free(values);
        values[0] = amm;
        values[1] = bmm;
        values[2] = (values[0] + values[1])/2;
        values[3] =  values[0] - values[1];

        signal2 = (float *) calloc(1000,sizeof(float));
        if(!signal2) {
                error();
                free(signal2);
                analysis();
        }

        for(j=0;j<1000;j++) {
                signal2[j] = (values[0]-signal1[j])*400/values[3] + 40;
        }
span:
        setviewport(0,0,639,479,0);
        bbx0(0,0,639,479,1,1,1);
        bbx0(69,36,571,444,15,15,0);
        bbxesc(5,5);        bbxtri(618,5);

        writetext(80,10,12,0,0,2,"KENV");
        writetext(136,10,12,0,0,2,"VOOD");
```

```
writetext(240,13,12,2,0,5,"DIGITAL STORAGE OSCILLOSCOPE");
writetext(485,10,12,2,0,6,"CS - 8010");
bbx0(69,36,571,444,15,15,0);
bbxesc(5,5);        bbxtri(618,5);
setlinestyle(1,1,NORM_WIDTH);
setcolor(15);
line(195,40,195,440);
line(320,40,320,440);
line(445,40,445,440);
line(71,240,569,240);
setlinestyle(SOLID_LINE,1,NORM_WIDTH);

axf=(double) (apx)/1000;
moveto(70,signal2[0]);
setcolor(14);
for(col = 0; col < 1000; col++) {
        lineto(dxx(col)+70,signal2[col]);
}
writetextxy(10,45,11,0,0,2,"mV");
writetextxy(590,440,11,0,0,2,"S");
writetextxy(5,35,14,0,0,1,"%1.1f",values[0]);
writetextxy(5,437,14,0,0,1,"%1.1f",values[1]);
writetextxy(5,235,14,0,0,1,"%1.1f",values[2]);
writetext(60,460,14,0,0,1,"0");
writetextxy(182,460,14,0,0,1,"%g",1/freq1);
writetextxy(310,460,14,0,0,1,"%g",2/freq1);
writetextxy(438,460,14,0,0,1,"%g",3/freq1);
writetextxy(566,460,14,0,0,1,"%g",4/freq1);

status1 = 1;
do {
        keyout = getch();
        if(keyout == 0) {
                keyout = getch();
                if((keyout == 75) || (keyout == 77)) {

                        if(keyout == 75) {
                        status1 = (status1 == 8 ) ? 1 : status1 + 1;
bbx0(0,300,639,479,1,1,1);
bbx0(69,36,571,444,15,15,0);
bbxesc(5,5);        bbxtri(618,5);
setlinestyle(1,1,NORM_WIDTH);
setcolor(15);
line(195,40,195,440);
line(320,40,320,440);
line(445,40,445,440);
line(71,240,569,240);
setlinestyle(SOLID_LINE,1,NORM_WIDTH);
moveto(70,signal2[0]);
setcolor(14);
for(col = 0; col < 1000; col++) {
        dirf = (int) (col /pow(2,status1));
        lineto(dxx(col)+70,signal2[dirf]);
}
```

```
writetextxy(10,45,11,0,0,2,"mV");
writetextxy(590,440,11,0,0,2,"S");
writetextxy(5,35,14,0,0,1,"%1.1f",values[0]);
writetextxy(5,437,14,0,0,1,"%1.1f",values[1]);
writetextxy(5,235,14,0,0,1,"%1.1f",values[2]);
writetext(60,460,14,0,0,1,"0");
writetextxy(182,460,14,0,0,1,"%g",status1/freq1);
writetextxy(310,460,14,0,0,1,"%g",2*status1/freq1);
writetextxy(438,460,14,0,0,1,"%g",3*status1/freq1);
writetextxy(566,460,14,0,0,1,"%g",4*status1/freq1);
                        bbxesci(5,5);       bbxtrii(618,5);
                        }

                        if(keyout == 77) {
                        status1 = (status1 == 1 ) ? 8 : status1 - 1;

bbx0(0,300,639,479,1,1,1);
bbx0(69,36,571,444,15,15,0);
bbxesc(5,5);        bbxtri(618,5);
setlinestyle(1,1,NORM_WIDTH);
setcolor(15);
line(195,40,195,440);
line(320,40,320,440);
line(445,40,445,440);
line(71,240,569,240);
setlinestyle(SOLID_LINE,1,NORM_WIDTH);
moveto(70,signal2[0]);
setcolor(14);
for(col = 0; col < 1000; col++) {
        dirf = (int) (col /pow(2,status1));
        lineto(dxx(col)+70,signal2[dirf]);
}
writetextxy(10,45,11,0,0,2,"mV");
writetextxy(590,440,11,0,0,2,"S");
writetextxy(5,35,14,0,0,1,"%1.1f",values[0]);
writetextxy(5,437,14,0,0,1,"%1.1f",values[1]);
writetextxy(5,235,14,0,0,1,"%1.1f",values[2]);
writetext(60,460,14,0,0,1,"0");
writetextxy(182,460,14,0,0,1,"%g",status1/freq1);
writetextxy(310,460,14,0,0,1,"%g",2*status1/freq1);
writetextxy(438,460,14,0,0,1,"%g",3*status1/freq1);
writetextxy(566,460,14,0,0,1,"%g",4*status1/freq1);
                        bbxesci(5,5);       bbxtrii(618,5);
                        }
                }
        }
        if(keyout == 27 || keyout == 72) {
                bbxesci(5,5);       bbxtrii(618,5);
                goto span;
        }
        if(keyout == 13) {
                bbxesci(5,5);       bbxtrii(618,5);
                free(signal2);  free(values);
                goto output;
```

```
                }
        } while((keyout != 13));

output:
        writetext(320,465,12,0,0,1,"Please Wait! 15 Second");
        length=1000;
        mea = log2(length);
        fft_length = 1<<mea;

        samp = (COMPLEX *) calloc(fft_length, sizeof(COMPLEX));
        if(!samp){
                error();
                free(samp);
                analysis();
        }

        linears = (float *) calloc(fft_length, sizeof(COMPLEX));
        if(!linears){
                error();
                free(linears);
                analysis();
        }
        for (i = 0 ; i < length ; i++) samp[i].real = signal1[i];
                han(samp,length);
        for (i = 0 ; i < length ; i++) signal1[i] = samp[i].real;


/* Find the spectrum of the data */
        fft(samp,mea);

/* do log magnitude */
        amp = 4.0/((double)length * length);
        for (i = 0 ; i < fft_length ; i++) {
                tempflt  = samp[i].real * samp[i].real;
                tempflt += samp[i].imag * samp[i].imag;
                tempflt *= amp;
                linears[i] = tempflt;
                samp[i].real = 10 * log10(MAX(tempflt,1.e-14));
        }

/*  Copy a section of the spectrum to the magnitude file.
This allows a close up plot of a particular segment of the
spectral magnitude of the signal. */

        center = fft_length/2;
        view = fft_length;
        begin = center-view/2;
        log_mag1 = (float *) calloc(view,sizeof(float));
        if(!log_mag1){
                error();
                free(signal2);
                analysis();
        }
        for(k = 0 ; k < view ; k++){
```

```
                i = k + begin;
                if (i<0) i = 0;
                if (i>=fft_length) i = fft_length - 1;
                log_mag1[k] = samp[i].real;
        }


/* Read the input data file specified by the user */
        tektronix();
        length = view;
        linear1=-1000.0;   linear2=1000.0;
        for(i=0;i<length;i++) {
                linear1 = MAX(linear1,linears[i]);
                linear2 = MIN(linear2,linears[i]);
        }
        free(linear);
        linear[0] = linear1;
        linear[1] = linear2;
        linear[2] = (linear[0] + linear[1])/2;
        linear[3] = linear[0] - linear[1];

        for(i=0;i<length;i++) {
                linears[i] = (linear[0]-linears[i])*350/linear[3] + 80;
        }
        axf=(double) (apx)/length;
        moveto(70,linears[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(dxx(col)+70,linears[col]);
        }
        writetext(320,465,14,0,0,1,"linear scale");
        writetext(185,445,1,0,0,1,
        "<-- (-)freq     0Hz     (+)freq -->");

        getch();
        bbxesci(5,5);       bbxtrii(618,5);
        free(linears);

/* Read the input data file specified by the user */
        tektronix();
        length = view;
        amm = -1000.0; bmm = 1000.0;
        for(i=0;i<length;i++) {
                amm = MAX(amm,log_mag1[i]);
                bmm = MIN(bmm,log_mag1[i]);
        }
        free(values);
        values[0] = amm;
        values[1] = bmm;
        values[2] = (values[0] + values[1])/2;
        values[3] = values[0] - values[1];

        for(i=0;i<length;i++) {
                log_mag1[i] = (values[0]-log_mag1[i])*350/values[3] + 80;
```

```
        }
        moveto(70,log_mag1[0]);
        setcolor(14);
        for(col = 0; col < length; col++) {
                lineto(dxx(col)+70,log_mag1[col]);
        }
        writetext(320,465,14,0,0,1,"log scale");
        writetext(185,445,1,0,0,1,
        "<-- (-)freq    0Hz     (+)freq -->");
        pauses();
        bbxesci(5,5);       bbxtrii(618,5);

first:  tektronix();
        axf=(double) (apx)/center;
        moveto(70,log_mag1[center/2]);
        setcolor(14);
        for(col = 0; col < center; col++) {
                lineto(dxx(col)+70,log_mag1[col+center/2]);
        }
        writetext(320,465,14,0,0,1,"Span1 freqs");
        bbxesci(5,5);       bbxtrii(618,5);

        center1 = length/8;
        axf=(double) (apx)/center1;
        status = 7;
        do {
                keyin = getch();
                if(keyin == 0) {
                        keyin = getch();
                        if((keyin == 75) || (keyin == 77)) {

                                if(keyin == 75) {
                                status = (status == 0 ) ? 7 : status - 1;
                                tektronix();
                                moveto(70,log_mag1[center1*status]);
                                setcolor(14);
                                for(col = 0; col < center1; col++) {
                                        i = center1*status+col;
                                        lineto(dxx(col)+70,log_mag1[i]);
                                }
                                writetextxy(320,465,14,0,0,1,
                                        "span (%d)8freqs",status+1);
                                bbxesci(5,5);       bbxtrii(618,5);
                                }

                                if(keyin == 77) {
                                status = (status == 7 ) ? 0 : status + 1;
                                tektronix();
                                moveto(70,log_mag1[center1*status]);
                                setcolor(14);
                                for(col = 0; col < center1; col++) {
                                        i = center1*status+col;
                                        lineto(dxx(col)+70,log_mag1[i]);
                                }
```

```
                                writetextxy(320,465,14,0,0,1,
                                        "span (%d)8freqs",status+1);
                                bbxesci(5,5);       bbxtrii(618,5);
                                }
                        }
                }
                if(keyin == 27 || keyin == 72) {
                        goto first;
                }
                if(keyin == 13) {
                        bbxesci(5,5);       bbxtrii(618,5);
                        free(signal1);      free(values);
                        free(log_mag1); free(samp);
                        analysis();
                }
        } while((keyin != 13));
        pauses();
        analysis();
}

void si(void)
{
        window31(220,200);
        writetext(185,4,14,2,0,4,"SINE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:          V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:        Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.sin"),COPY_PUT);

        pauses();
        calwave();
}

void co(void)
{
        window31(220,200);
        writetext(185,4,14,2,0,4,"COSINE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:          V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:        Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.cos"),COPY_PUT);
```

```
/*************************************************************************
                        King Mongkut's Institute of Technology
                                    KMITL
                        The Communication Systems Analysis   VERSION 1.01
-------------------------------------------------------------------------------
                        Software for Communication System Analysis VERSION 1.01
                    Communications System are concerned with the transmittion
                    of the information from a source to a sink DIGITAL & ANALOG
                    communications systems. This program is require a persional
                    computer which has an VGA or higher display
-------------------------------------------------------------------------------
FILE NAME :  CCAD.CPP
progammer  :  WEERASAK  NAMVONG
Create       :  25 / Feb / 1992
Update       :  19 / may / 1992
Purpose      :  Prototype function for CCAD.EXE
Comment      :  This utility for Communications System Analysis

Copyright (c) 1993 King Mongkut's Institute of Technology Ladkrabang
*************************************************************************/

#include <alloc.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <graphics.h>
#include <math.h>

void Pause();
void *SaveImage(int left,int top,int right,int bottom,unsigned *size,char *filename);
void *ReadImage(char *filename);

void Pause()
{
        while(kbhit()) getch();
        getch();
}

void *SaveImage(int left,int top,int right,int bottom,unsigned *size,char *filename)
{
        void far *image;

        FILE *fl=fopen(filename,"w");
        *size=imagesize(left,top,right,bottom);
        image=farmalloc(*size);
        getimage(left,top,right,bottom,image);
        putimage(left,top,image,XOR_PUT);
        fwrite(image,*size,1,fl);
        fflush(fl);
        fclose(fl);
        farfree(image);
}
```

```
        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.saw"),COPY_PUT);

        pauses();
        calwave();
}

void ha(void)
{
        window31(220,200);
        writetext(160,4,14,2,0,4,"HALF CIRCLE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:             V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:           Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.haf"),COPY_PUT);

        pauses();
        calwave();
}

void pu(void)
{
        window31(220,200);
        writetext(180,4,14,2,0,4,"PULSE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:             V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:           Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        writetext(50,140,12,0,0,1,
        "pulse    :             Hz");
        getscanf(200,140,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.pul"),COPY_PUT);

        pauses();
        calwave();
}

void gu(void)
{
        window31(220,200);
        writetext(175,4,14,2,0,4,"GAUSSIAN WAVE");
        writetext(50,100,12,0,0,1,
```

```
        "amplitude:             V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.gau"),COPY_PUT);

        pauses();
        calwave();
}

float   *fwave(char *file)
{
        DSP_FILE *dsp_info;
        float *sign;
        dsp_info = open_read(file);
        sam = dsp_info->rec_len;
        sign = (float *) calloc(sam,sizeof(float));
        if(!sign) {
                error();
                free(sign);
                analysis();
        }
        sign = read_float_record(dsp_info);
        return(sign);
}

float   *wave( length, freq, number)
        int length;             /* number of samples to generate        */
        float *freq;            /* pointer to frequency array           */
        int number;             /* number of frequencies to generate    */
{
        int i, j;
        double arg,twopi;
        float *samp;                    /* pointer to samples           */

        samp = (float *) calloc(length,sizeof(float));
        if(!samp) {
                error();
                free(samp);
                analysis();
        }

        twopi = 8.0*atan(1.0);      /*  calculate 2*PI          */
        for (j = 0 ; j < number ; j++) {
                arg = twopi * freq[j];
                for (i = 0 ; i < length ; i++)
                        samp[i] += sin(i*arg);
        }
        for (i=0; i<length; i++)
                samp[i] = samp[i]/number;
        return(samp);
}
```

```
getscanf(int x1,int y,int x2,int color,char font,int direct,int size)
{
        int     i;
        char    *gtext="a";
con:    bbx0(x1-10,y-5,x2,y+10,15,8,1);
        free(CHR);
        CHR = malloc(120);
        for(i=0;i<=119;i++)*(CHR+i)=0;
        while(*gtext!='\r'){
                while(!bioskey(1));
                *gtext = bioskey(0);
                if(*gtext!='\r') {
                if(*gtext==AF9)  analysis();
                if(*gtext!=BS) {
                        strcat(CHR,gtext);
                        writetextxy(x1,y,color,font,
                                direct,size,"%s",CHR);
                        }
                if(*gtext==BS)  goto con;
                }
        }
        *gtext = 'a';
        INT=atoi(CHR);
        FLOA=atof(CHR);
}

error()
{
                kmitsong();
}

beep()
{
                kmitsong();
}

int dxx(int vx)
{
        return((int) (axf*vx));
}
/*<*****************************<end of mkwave.c>*************************/
```

```
        pauses();
        calwave();
}

void sq(void)
{
        window31(220,200);
        writetext(185,4,14,2,0,4,"SQUARE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:          V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:        Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.squ"),COPY_PUT);

        pauses();
        calwave();
}

void tr(void)
{
        window31(220,200);
        writetext(180,4,14,2,0,4,"TRIANGLE WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:          V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:        Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);

        setviewport(0,0,getmaxx(),getmaxy(),0);
        putimage(280,220,ReadImage("ccad.tri"),COPY_PUT);

        pauses();
        calwave();
}

void sa(void)
{
        window31(220,200);
        writetext(190,4,14,2,0,4,"SAW WAVE");
        writetext(50,100,12,0,0,1,
        "amplitude:          V");
        getscanf(200,100,240,6,0,0,1,"%s",CHR);

        writetext(50,120,12,0,0,1,
        "frequencies:        Hz");
        getscanf(200,120,240,6,0,0,1,"%s",CHR);
```

```
void *ReadImage(char *filename)
{
        unsigned xsize,ysize,size;
        FILE *f1=fopen(filename,"r");
        void *tempimage;

        xsize=fgetc(f1)|(fgetc(f1)<<8);
        ysize=fgetc(f1)|(fgetc(f1)<<8);
        size=imagesize(0,0,xsize,ysize);
        tempimage=farmalloc(size);
        rewind(f1);
        fread(tempimage,size,1,f1);
        fclose(f1);
        return(tempimage);
}

windows31(int x, int y)
{
        setviewport(x,y,getmaxx(),getmaxy(),0);
        bbx0(1,1,getmaxx(),getmaxy(),15,15,3);
        bbxesc(1,1);bbxesc(623-x,1);
        bbx0(22,21,getmaxx()-x-21,getmaxy()-y-15,15,15,9);
        bbx0(22,2,getmaxx()-x-21,16,15,8,6);
}

main()
{
        opengraph();
        s1();
        s2();
        s3();
        s3();
        s4();
        s5();
        s6();
        s7();
        s8();
        closegraph();
}

s1()
{
        int i;
        double wc;
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"SINE WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
```

```
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);
        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);

        wc=16.0*atan(1.0)/268;
        moveto(26,123);
        for(i = 1; i < 268; i++) {
                lineto(26+i,123-49*sin(wc*i));
        }

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.sin");
        putimage(10,10,ReadImage("ccad.sin"),COPY_PUT);
}

s2()
{
        int i;
        double wc;
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"COSINE WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);
        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);

        wc=16.0*atan(1.0)/268;
        moveto(26,74);
        for(i = 1; i < 268; i++) {
                lineto(26+i,123-49*cos(wc*i));
        }

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.cos");
        putimage(10,10,ReadImage("ccad.cos"),COPY_PUT);
}
```

```
s3()
{
        int i,j;
        double wc;
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"|SINE| WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);
        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);

        wc=16.0*atan(1.0)/268;
        moveto(26,123);
        for(j=0;j<=3;j++) {
                for(i = 1; i <= 67; i++) {
                        lineto((67*j)+26+i,123-49*sin(wc*i));
                }
        }

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.haf");
        putimage(10,10,ReadImage("ccad.haf"),COPY_PUT);
}

s4()
{
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"SAW WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);

        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);
        moveto(26,123);
```

```
        lineto(93,74);
        lineto(93,123);
        lineto(160,74);
        lineto(160,123);
        lineto(227,74);
        lineto(227,123);
        lineto(294,74);

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.saw");
        putimage(10,10,ReadImage("ccad.saw"),COPY_PUT);
}

s5()
{
        unsigned Size;

        windows31(320,240);
        writetext(125,4,14,2,0,4,"TRIANGLE WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);

        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);
        moveto(26,123);
        lineto(93,74);
        lineto(160,123);
        lineto(227,74);
        lineto(294,123);

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.tri");
        putimage(10,10,ReadImage("ccad.tri"),COPY_PUT);
}

s6()
{
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"SQUARE WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
```

```
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);

        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);
        moveto(26,74);
        lineto(58,74);
        lineto(58,172);
        lineto(93,172);
        lineto(93,74);
        lineto(125,74);
        lineto(125,172);
        lineto(160,172);
        lineto(160,74);
        lineto(192,74);
        lineto(192,172);
        lineto(227,172);
        lineto(227,74);
        lineto(259,74);
        lineto(259,172);
        lineto(294,172);
        Pause();

        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.squ");
        putimage(10,10,ReadImage("ccad.squ"),COPY_PUT);
}

s7()
{
        unsigned Size;

        windows31(320,240);
        writetext(130,4,14,2,0,4,"PULSE WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);
        setcolor(12);
        setlinestyle(0,1,THICK_WIDTH);

        moveto(26,123);
        lineto(48,123);
```

```
        lineto(48,74);
        lineto(70,74);
        lineto(70,123);
        lineto(115,123);
        lineto(115,74);
        lineto(137,74);
        lineto(137,123);
        lineto(182,123);
        lineto(182,74);
        lineto(204,74);
        lineto(204,123);
        lineto(249,123);
        lineto(249,74);
        lineto(271,74);
        lineto(271,123);
        lineto(294,123);
        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.pul");
        putimage(10,10,ReadImage("ccad.pul"),COPY_PUT);
}

s8()
{
        int i;
        unsigned Size;

        windows31(320,240);
        writetext(125,4,14,2,0,4,"GAUSSIAN WAVE");
        bbx0(25,24,295,221,15,15,0);
        setcolor(15);
        setlinestyle(3,1,NORM_WIDTH);
        line(93,25,93,221);
        line(160,25,160,221);
        line(227,25,227,221);
        line(26,74,294,74);
        line(26,123,294,123);
        line(26,172,294,172);
        setcolor(12);
        setlinestyle(0,2,THICK_WIDTH);

        moveto(26,123);
        for(i = 1; i < 268; i++) {
                lineto(26+i,80+random(80));
        }

        Pause();
        setviewport(0,0,getmaxx(),getmaxy(),0);
        Pause();
        SaveImage(321,241,639,479,&Size,"ccad.gau");
        putimage(10,10,ReadImage("ccad.gau"),COPY_PUT);
}
/*****************************<end of ccad.cpp>*********************************/
```

ภาคผนวก ข SUMMARY

คำสั่งกราฟิคส์ของเทอร์โบซี

ต่อไปนี้เป็นฟังก์ชั่นทั้งหมดใน graphics.lib

ฟังก์ชั่น `arc (int x,int y,int strangle,int endangle,int radius);`

การทำงาน วาดส่วนของเส้นโค้งที่จุดศูนย์กลาง (x,y) รัศมีเท่ากับ radius โดยที่มุมเริ่มต้นที่ strangle ไปสิ้นสุดที่มุม endangle

ฟังก์ชั่น `bar (int left,int top,int right,int bottom);`

การทำงาน ระบายพื้นผิวสี่เหลี่ยม สีและลวดลายที่จะใช้ระบาย กำหนดด้วยฟังก์ชั่น setfillstyle หรือ setfillpattern

ฟังก์ชั่น `bar3d (int left,int top,int right,int bottom,int depth,int topflag);`

การทำงาน วาดแท่งสี่เหลี่ยม 3 มิติและระบายพื้นผิวของสี่เหลี่ยม

ฟังก์ชั่น `circle (int x,int y,int radius);`

การทำงาน วาดวงกลมที่มีจุดศูนย์กลางที่ (x,y) มีรัศมีเท่ากับ radius โดยสีของเส้นรอบวงกำหนดได้ด้วยฟังก์ชั่น setcolor

ฟังก์ชั่น `cleardevice (viod);`

การทำงาน ลบทุกสิ่งบนหน้าจอออกหมดทั้งจอภาพ เปลี่ยนสีของจอภาพเป็นสีแบคกราวนด์ ซึ่งกำหนดด้วยฟังก์ชั่น setbkcolor

ฟังก์ชั่น `clearviewport (void);`

การทำงาน ลบภาพต่าง ๆ ในบริเวณวิวพอร์ท

ฟังก์ชั่น `closegraph (void);`

การทำงาน เปลี่ยนการทำงานของจอภาพให้กลับไปอยู่ในโหมดเดิม

ฟังก์ชั่น **detectgraph** (int far *graphdriver, int far *graphmode);

การทำงาน ตรวจสอบว่าจอภาพเป็นชนิดใด และส่งค่าไดรฟ์เวอร์ และโหมดทางกราฟิคผ่านทางตัวแปร graphdriver และ graphmode

ฟังก์ชั่น **drawpoly** (int n,int far *polypoints);

การทำงาน วาดเส้นขอบของรูปหลายเหลี่ยม n จุด โคออร์ดิเนตของจุดต่าง ๆ ถูกเก็บไว้ในตัวแปร polypoints

ฟังก์ชั่น **ellipse** (int x,int y,int stangle,int endangle, int xradius,int yradius);

การทำงาน วาดเส้นโค้งวงรี

ฟังก์ชั่น **fillellipse** (int x, int y, int xradius, int yradius);

การทำงาน วาดวงรีและระบายพื้นผิวภายใน

ฟังก์ชั่น **fillpoly**(int numpoints, int far *polypoints);

การทำงาน วาดเส้นขอบของรูปหลายเหลี่ยมและระบายพื้นผิวภายใน

ฟังก์ชั่น **floodfill**(int x, int y, int border);

การทำงาน ระบายพื้นผิวภายในขอบเขตปิด

ฟังก์ชั่น **getarccoords** (struct arccoordstype far *arccoords);

การทำงาน หาค่าโคออร์ดิเนตของส่วนโค้ง

ฟังก์ชั่น **getaspectratio**(int far *xasp, int far *yasp);

การทำงาน หาค่า **aspect ratio** ของจอภาพ

ฟังก์ชั่น `getbkcolor(void);`
การทำงาน ส่งค่าของสีแบคกราวน์

ฟังก์ชั่น `getcolor(void);`
การทำงาน ส่งค่าของสีที่ใช้วาด

ฟังก์ชั่น `getdefaultpalette ( void );`
การทำงาน หาค่าของพาเลตต์เมื่อเริ่มต้นการทำงานในโหมดกราฟิคส์

ฟังก์ชั่น `getdrivername ( void );`
การทำงาน ส่งข้อความของไดรฟเวอร์ที่เลือกใช้

ฟังก์ชั่น `getfillpattern (char far *pattern);`
การทำงาน หาลวดลายของการระบายพื้นผิวที่ผู้ใช้สร้างขึ้น

ฟังก์ชั่น `getfillsettings (struct fillsettingstype far *fillinfo);`
การทำงาน หาลวดลายและสีของการระบายพื้นผิว

ฟังก์ชั่น `getgraphmode (void);`
การทำงาน ส่งค่าของโหมดทางกราฟิคส์ที่ระบบกำลังทำงานอยู่

ฟังก์ชั่น `getimage (int left, int top, int right, int bottom,void far *bitmap);`
การทำงาน เก็บภาพในบริเวณพื้นที่สี่เหลี่ยมที่กำหนด

ฟังก์ชั่น `getlinesettings (struct linesettingstype far *lineinfo);`
การทำงาน หาลวดลายของการวาดเส้น

ฟังก์ชั่น **getmaxcolor(void);**

การทำงาน ส่งค่าหมายเลขสีสุดท้ายในพาเลตต์

ฟังก์ชั่น **getmaxmode (void);**

การทำงาน ส่งค่าของหมายเลขสูงสุดของไดรฟเวอร์ที่กำลังใช้

ฟังก์ชั่น **getmaxx(void);**

การทำงาน ส่งค่าของโคออร์ดิเนต X ที่มากที่สุด

ฟังก์ชั่น **getmaxy (void);**

การทำงาน ส่งค่าของโคออร์ดิเนต Y ที่มากที่สุด

ฟังก์ชั่น **getmodename (int mode_number);**

การทำงาน ส่งข้อความรายระเอียดของโหมดทางกราฟิคส์ที่กำหนด

ฟังก์ชั่น **getmoderange (int graphdriver, int far *lomode, int far *himode);**

การทำงาน หาค่าของหมายเลขโหมดที่มากที่สุดและน้อยที่สุดของไดรฟเวอร์ที่กำหนด

ฟังก์ชั่น **getpixel (int x, int y);**

การทำงาน ส่งค่าสีของพิกเซลที่กำหนด

ฟังก์ชั่น **getpalette (struct palettetype far *palette);**

การทำงาน หาค่าของจำนวนสีและสีต่าง ๆ ในพาเลตต์

ฟังก์ชั่น **getpalettesize (void);**

การทำงาน จำนวนสีที่ใช้ได้ในไดรฟเวอร์และโหมดที่กำลังทำงานอยู่

ฟังก์ชั่น `gettextsettings (struct textsettingstype far *texttypeinfo);`

การทำงาน หาค่าของการกำหนดและการวางข้อความในโหมดกราฟิกส์

ฟังก์ชั่น `getviewsettings(struct viewporttype far *viewport);`

การทำงาน หาค่าของการกำหนดวิวพอร์ต

ฟังก์ชั่น `getx (void);`

การทำงาน ส่งค่าของโคออร์ดิเนตทางแกน X ของ CP

ฟังก์ชั่น `gety (void);`

การทำงาน ส่งค่าของโคออร์ดิเนตทางแกน Y ของ CP

ฟังก์ชั่น `graphdefaults (void);`

การทำงาน รีเซ็ตระบบกราฟิกส์

ฟังก์ชั่น `grapherrormsg (int errorcode);`

การทำงาน ส่งข้อความแสดงความผิดพลาดของรหัสความผิดพลาด

ฟังก์ชั่น `graphresult (void);`

การทำงาน ส่งค่ารหัสความผิดพลาด

ฟังก์ชั่น `imagesize (int left, int top, int right, int bottom);`

การทำงาน ส่งค่าขนาดของภาพ (ไบต์) ในบริเวณที่กำหนด

ฟังก์ชั่น `initgraph (int far *graphdriver,*graphmode, *pathtodriver);`

การทำงาน เริ่มต้นเข้าสู่การทำงานในโหมดกราฟิกส์

ฟังก์ชั่น `installuserdriver( char far *name, int huge (*detect)(void) );`

การทำงาน รวมไดรฟเวอร์ชนิดใหม่เข้ากับระบบกราฟิกส์

ฟังก์ชั่น `installuserfont ( char far *name );`

การทำงาน รวมแบบตัวอักษรแบบใหม่เข้ากับระบบกราฟิกส์

ฟังก์ชั่น `line (int x1, int y1, int x2, int y2);`

การทำงาน ลากเส้นตรงจากจุดที่กำหนด

ฟังก์ชั่น `linerel (int dx, int dy);`

การทำงาน ลากเส้นตรงจาก CP ไปยังจุดที่ห่างออกไปเป็นระยะทางที่กำหนด

ฟังก์ชั่น `lineto (int x, int y);`

การทำงาน ลากเส้นตรงจาก CP ไปยังจุดที่กำหนด

ฟังก์ชั่น `moverel (int dx, int dy);`

การทำงาน ย้าย CP ไปยังจุดที่ห่างออกไปเป็นระยะทางที่กำหนด

ฟังก์ชั่น `moveto (int x, int y);`

การทำงาน ย้าย CP ไปยังจุดที่กำหนด

ฟังก์ชั่น `outtext (char far *textstring);`

การทำงาน แสดงข้อความออกทางจอภาพ ณ ตำแหน่งของ CP

ฟังก์ชั่น `outtextxy (int x, int y, char far *textstring);`

การทำงาน แสดงข้อความออกทางจอภาพ ณ จุดที่กำหนด

ฟังก์ชั่น	`pieslice (int x, int y, int stangle, int endangle,int radius);`
การทำงาน	วาดและระบายส่วนแบ่งของวงกลม
ฟังก์ชั่น	`putimage (int left, int top, void far *bitmap, int op);`
การทำงาน	สร้างภาพจากภาพที่เก็บไว้
ฟังก์ชั่น	`putpixel (int x, int y, int color);`
การทำงาน	เปลี่ยนสีของพิกเซล
ฟังก์ชั่น	`rectangle (int left, int top, int right, int bottom);`
การทำงาน	วาดกรอบสี่เหลี่ยม
ฟังก์ชั่น	`restorecrtmode (void);`
การทำงาน	รวมไดรฟเวอร์ในหน่วยความจำกับ GRAPH UNIT
ฟังก์ชั่น	`sector ( int X, int Y, int StAngle, int EndAngle,int XRadius, int YRadius );`
การทำงาน	วาดและระบายส่วนแบ่งของวงรี
ฟังก์ชั่น	`setactivepage(int page);`
การทำงาน	เลือกเพจที่ใช้เป็นเอาท์พุตของคำสั่งต่าง ๆ
ฟังก์ชั่น	`setallpalette(struct palettetype far *palette);`
การทำงาน	เปลี่ยนสีต่าง ๆในพาเลตต์

ฟังก์ชั่น **setaspectratio (int xasp, int yasp);**
การทำงาน กำหนด ASPECT RATIO

ฟังก์ชั่น **setbkcolor(int color);**
การทำงาน กำหนดสีแบคกราวด์ของจอภาพ

ฟังก์ชั่น **setcolor (int color);**
การทำงาน กำหนดสีของการวาดภาพ

ฟังก์ชั่น **setfillpattern (char far *upattern, int color);**
การทำงาน สร้างและกำหนดลวดลายของการระบายพื้นผิว

ฟังก์ชั่น **setfillstyle (int pattern, int color);**
การทำงาน กำหนดลวดลายและสีของการระบายพื้นผิว

ฟังก์ชั่น **setgraphbufsize (unsigned bufsize);**
การทำงาน กำหนดขนาดของบัฟเฟอร์ในการทำงานของบางคำสั่ง

ฟังก์ชั่น **setgraphmode(int mode);**
การทำงาน เปลี่ยนการทำงานของจอภาพไปเป็นโหมดกราฟิกส์

ฟังก์ชั่น **setlinestyle (int linestyle, unsigned upattern, int thickness);**
การทำงาน เลือกลักษณะของเส้นที่ใช้วาดรูป

ฟังก์ชั่น **setpalette (int colornum, int color);**
การทำงาน เปลี่ยนสีในพาเลตต์

ฟังก์ชั่น **setrgbpalette (int colornum,int red, int green, int blue);**

การทำงาน เปลี่ยนองค์ประกอบของสีในพาเลตต์

ฟังก์ชั่น **settextjustify (int horiz, int vert);**

การทำงาน กำหนดลักษณะการวางข้อความ

ฟังก์ชั่น **settextstyle (int font, int direction, int charsize);**

การทำงาน กำหนดแบบ ทิศทาง และขนาดของตัวอักษร

ฟังก์ชั่น **setusercharsize (int multx, int divx,int multy, int divy);**

การทำงาน เปลี่ยนแปลงขนาดของตัวอักษรหรือข้อความ

ฟังก์ชั่น **setviewport (int left, int top, int right, int bottom,int clip);**

การทำงาน กำหนดวิวพอร์ต

ฟังก์ชั่น **setvisualpage (int page);**

การทำงาน เลือกเพจที่จะถูกแสดงบนจอภาพ

ฟังก์ชั่น **setwritemode (int mode);**

การทำงาน กำหนดวิธีการสร้างเส้นลงบนข้อความ

ฟังก์ชั่น **textheight (char far *textstring);**

การทำงาน ส่งค่าความสูงของข้อความ

ฟังก์ชั่น **textwidth (char far *textstring);**

การทำงาน ส่งค่าความกว้างของข้อความ