ตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์
MULTIPURPOSE DIGITAL MODULATOR

โดย
นายนพดล โอภาพันธ์
นายศิริพงษ์ เรืองฤทธิ์
นายอดิศร ตันฉาวร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

สารบัญ

ตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์
A MULTIPURPOSE DIGITAL MODULATOR

โดย

นายนพดล โอภาพันธ์
นายศิริพงษ์ เรืองฤทธิ์
นายอดิศร ตันถาวร

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

บทคัดย่อ

สถาปัตยกรรมของตัวมอดูเลทสัญญาณดิจิตอล (Digital Modulator) ที่ใช้ในปัจจุบันจะมีลักษณะแตกต่างกันขึ้นอยู่กับชนิดของการมอดูเลทแบบต่างๆทำให้ขาดความยืดหยุ่นในการใช้งาน บทความนี้ได้นำเสนอตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์ (Multipurpose Digital Modulator) ที่สามารถทำการมอดูเลทได้หลายแบบโดยใช้ซอฟท์แวร์ (Soft-ware) ในการทำงานตามหลักการของตัวแยกสัญญาณ (Resolver) และตัวสร้างสัญญาณตัวอย่าง (Sample Generator)

Abstract

Architecture of digital modulators widely used at the present are vary with various types of modulation schemes, then lack of flexibility in any applications. This paper presents a multipurpose digital modulator that can operate many kinds of digital modulations by using software to process in the method of the resolver and the sample generator.

ปริญญานิพนธ์ปีการศึกษา 2536

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง ตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์

คณะผู้จัดทำ

1. นายนพดล โอภาพันธ์ 33100157
2. นายศิริพงษ์ เรืองฤทธิ์ 33100374
3. นาย อดิศร ตันถาวร 33100477

_____ อาจารย์ที่ปรึกษา

(รศ.ดร. กอบชัย เดชหาญ)

1. บทนำ

ในระบบการสื่อสารโทรคมนาคม ที่ใช้ในปัจจุบันมีความต้องการเพิ่มขึ้นอย่างมากทั้งทางคุณภาพ ปริมาณและความยืดหยุ่นในการใช้งาน ด้วยเหตุนี้ระบบการสื่อสารแบบดิจิตอล (Digital Communication Systems)ที่ให้ความถูกต้องแม่นยำมากกว่าจึงเข้ามาแทนที่ระบบการสื่อสารแบบอนาลอก (Analog Communication Systems) และในการส่งสัญญาณหรือข้อมูลในระบบต่างๆ เช่น ระบบโทรศัพท์ ระบบสื่อสารดาวเทียมภาคพื้นดิน หรือระบบสื่อสารดาวเทียมอวกาศ (Aeronautical or Land Mobile Satellite Systems) เป็นต้น จำเป็นต้องใช้เทคนิคการมอดูเลทสัญญาณดิจิตอลที่ละเอียดและซับซ้อน ทำให้ต้องมีการศึกษาและค้นคว้าเทคนิคในการมอดูเลทแบบต่างๆ เพื่อพัฒนาให้ทันกับความต้องการของตลาดการสื่อสารที่เป็นอยู่

แต่จะเห็นได้ว่า ตัวมอดูเลทสัญญาณดิจิตอลที่มีอยู่นั้นยังขาดความยืดหยุ่นในการใช้งาน เช่น อุปกรณ์สื่อสารทางคอมพิวเตอร์ที่เรียกว่าโมเด็ม (Modem) โดยทั่วไปจะสามารถส่งสัญญาณที่ได้จากการมอดูเลทสัญญาณดิจิตอลเพียงรูปแบบเดียว ทำให้ไม่สามารถรับส่งสัญญาณกับโมเด็มที่มีรูปแบบการมอดูเลทแบบอื่นได้ หรือ ในระบบการสื่อสารใหญ่ๆ ที่ไม่สามารถติดต่อสื่อสารกับระบบย่อยที่มีการมอดูเลทแบบต่างๆได้ดังนั้นจึงได้พัฒนาตัวมอดูเลทเอนกประสงค์ที่สามารถส่งสัญญาณจากการมอดูเลทได้หลายรูปแบบ ทำให้มีความยืดหยุ่นในการใช้งานกับระบบต่างๆ ซึ่งในบทความนี้ได้นำเสนอตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์โดยใช้ซอฟท์แวร์ในการสร้างและมอดูเลทสัญญาณเบสแบนด์ (Baseband Signals)แทนการใช้ฮาร์ดแวร์ (Hardware) ที่มีอยู่เดิม

2. ทฤษฎี

2.1 สมการทั่วไปและเงื่อนไขการมอดูเลชั่น

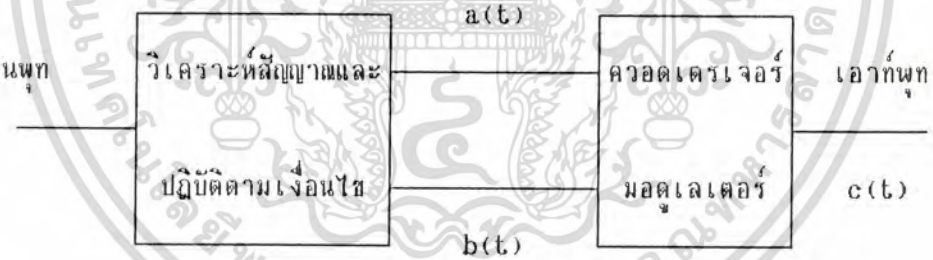
สมการทั่วไปของการมอดูเลชั่น

$$c(t) = a(t)\cos(w_c t) + b(t)\sin(w_c t) \qquad (1)$$

โดยที่

$c(t)$	คือ สัญญาณผลรวมของสัญญาณรูปไซน์และโคไซน์ด้วยขนาดต่าง ๆ
$\cos(w_c t), \sin(w_c t)$	คือ สัญญาณคลื่นพาห์ที่มีความถี่(w_c)เท่ากัน แต่มีเฟสต่างกัน 90°
$a(t), b(t)$	คือ แอมปลิจูดของสัญญาณโคไซน์และไซน์ตามลำดับ
w_c	คือ ความถี่เชิงมุมของคลื่นพาห์

เราสามารถออกแบบอุปกรณ์ให้สร้างสัญญาณตามสมการ (1) หลักการทำงานโดยทั่วไปของอุปกรณ์แสดงได้ตามแผนผังข้างล่าง

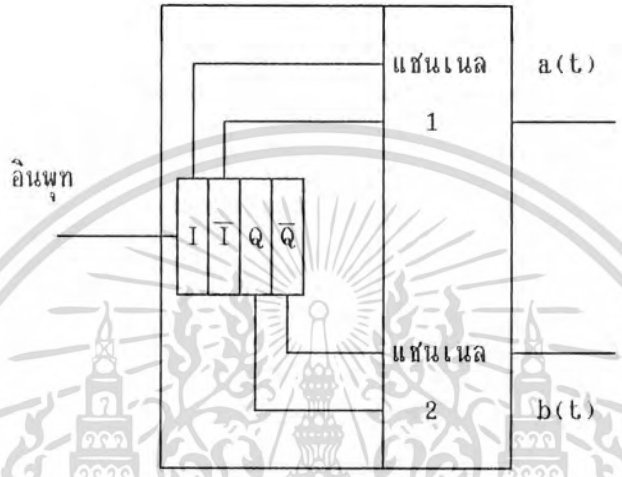


รูปที่ 2.1

เมื่อทำการเปลี่ยนรูปแบบการมอดูเลท ส่วนวิเคราะห์สัญญาณและปฏิบัติตามเงื่อนไขต้องสร้างขึ้นมาใหม่เพื่อให้ตรงตามเงื่อนไขของการมอดูเลทแต่ละแบบ เงื่อนไขจะเป็นตัวกำหนดสัมประสิทธิ์ a(t), b(t) และทำให้เกิดดิจิตอลมอดูเลชั่นแบบต่าง ๆ พื้นฐานสำคัญที่จะนำเราไปสู่หลักการมอดูเลชั่นเอนกประสงค์ก็คือ การที่สามารถวิเคราะห์และเข้าใจเงื่อนไขการมอดูเลทแต่ละแบบได้ ในที่นี้จะนำมาแสดงเพียง 16 ควอดเดรเจอร์แอมปลิจูดมอดูเลชั่น (Sixteen Quadrature Modulation (16QAM)) ส่วนเงื่อนไขการมอดูเลทแบบอื่นดูได้จากบทแทรก

16 ควอดเดรเจจอร์แอมปลิจูดมอดูเลชั่นเป็นการเข้ารหัส M-ary เมื่อ M = 16 แสดงว่ามีอินพุทบิท 4 บิท ข้อมูลทางด้านอินพุทเป็นกลุ่มของบิทข้อมูล 4 บิท ดังนั้นจะทำการวิเคราะห์สัญญาณที่เข้ามาครั้งละ 4 บิท เอาท์พุทที่ได้จะมี 16 ค่าที่ต่างกันตามเงื่อนไขที่คล้ายกับ 8 ควอดเดรเจอร์แอมปลิจูดมอดูเลชั่น

วงจรวิเคราะห์และปฏิบัติตามเงื่อนไข



รูปที่ 2.2

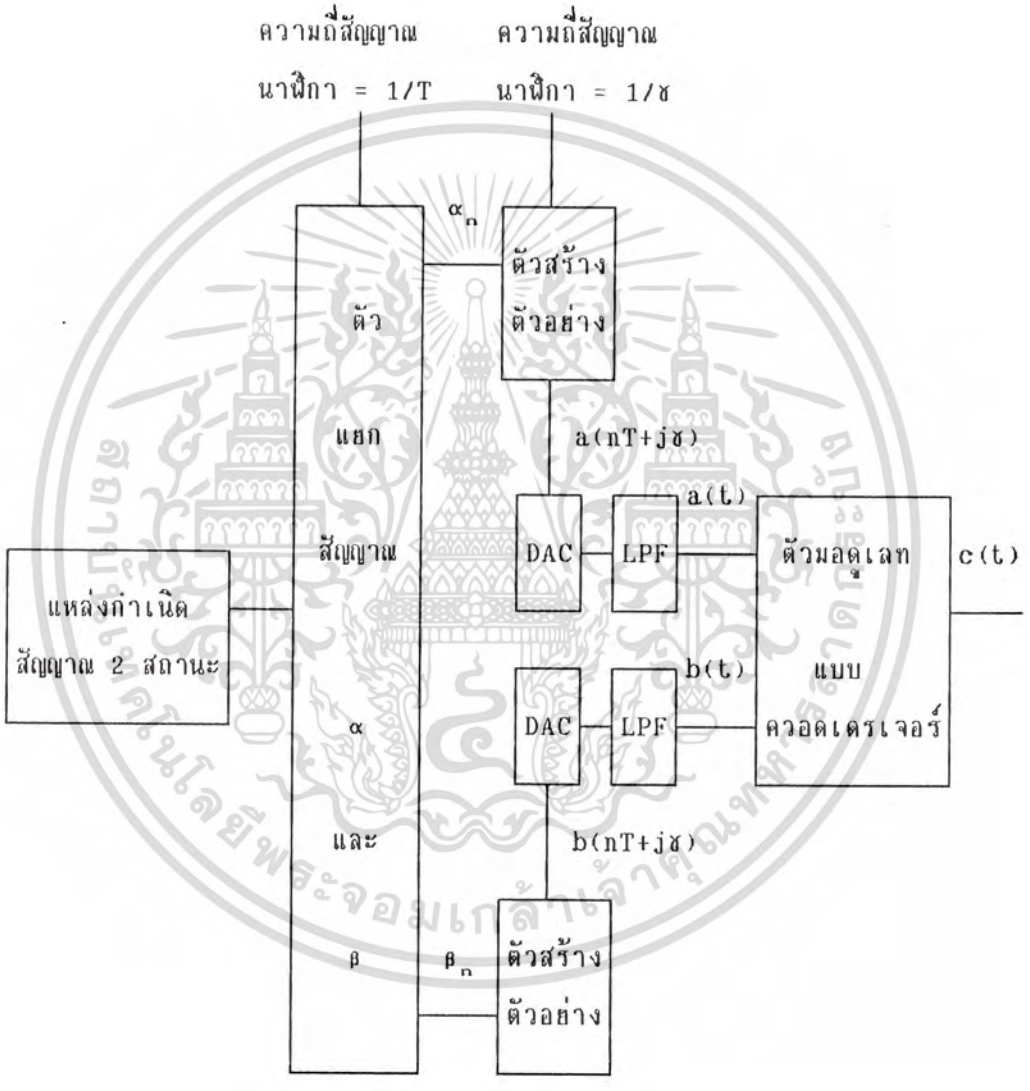
ตารางที่ 2.1

ลอจิกของI	ลอจิกของ$\bar{I}$	a(t)	ลอจิกของQ	ลอจิกของ$\bar{Q}$	b(t)
0	0	-x v	0	0	-x v
0	1	-y v	0	1	-y v
1	0	+x v	1	0	+x v
1	1	+y v	1	1	+y v

จากรูปที่ 2.2 สัญญาณอินพุทที่รับเข้ามาจะถูกแยกออกเป็น 2 แชนเนล เพื่อนำไปวิเคราะห์ตามเงื่อนไขดังตารางที่ 2.1 ซึ่งแสดงให้เห็นว่า ลอจิกของบิท I และ $\bar{I}$ เป็นเงื่อนไขในการกำหนดค่าของ a(t) ส่วนลอจิกของบิท Q และ $\bar{Q}$ เป็นเงื่อนไขในการกำหนดค่าของ b(t) ลอจิกของ I และ Q ใช้ในการกำหนดค่าของ a(t) และ b(t) ตามลำดับว่าเป็นบวกหรือลบ ลอจิกของ $\bar{I}$ และ $\bar{Q}$ ใช้ในการกำหนด a(t) และ b(t) ตามลำดับว่ามีขนาดเท่ากับ x volt หรือ y volt

2.2 หลักการของตัวมอดูเลทเอนกประสงค์

การทำงานของฮาร์ดแวร์มีแผนผังแสดงส่วนประกอบของตัวมอดูเลทเอนกประสงค์ดังรูป 2.3



รูปที่2.3

2.2.1 การสร้างสัญญาณ a(t) และ b(t)

รูปร่างของสัญญาณพัลส์ (Pulse) มีผลต่อสเปคตรัมของสัญญาณ ถึงแม้ว่าโดยทั่วไปแล้ว a(t) และ b(t) จะมีความสัมพันธ์กัน (Correlate) แต่ก็สามารถกรองสัญญาณแยกออกจากกันได้ ถ้าให้ h(t) แทนรูปร่างพัลส์ที่ใช้แสดงข่าวสารดิจิตอล รูปสัญญาณของ a(t) และ b(t) สามารถแสดงได้ดังสมการที่ 2

$$a(t) = \Sigma \alpha_i h(t - iT)$$

$$b(t) = \Sigma \beta_i h(t - iT) \qquad (2)$$

โดยที่ α_i และ β_i เป็นสัมประสิทธิ์ ที่มีค่าขึ้นอยู่กับสัญญาณจากแหล่งกำเนิดที่เวลา iT โดยกำหนดให้ T แทนคาบเวลาของสัญญลักษณ์ และ i แทนดัชนีนับสัญญลักษณ์ (symbol-count index) ตัวอย่างเช่น สัญญาณของ 8PSK จะมีคาบเวลา T ของสัญญลักษณ์เป็นสามเท่าของคาบเวลาบิท (bit duration) ซึ่งถ้าไม่มีการจำกัดแถบความถี่ของสัญญาณแล้ว h(t) จะถูกจำกัดด้วยคาบเวลา T ทั้งนี้เพื่อหลีกเลี่ยงไม่ให้เกิดการรบกวนกันของสัญญาณระหว่างสัญญลักษณ์ (Inter Symbol Interference,ISI) แต่ถ้ามีการจำกัดแถบความถี่แล้ว สัญญาณ a(t) และ b(t) จะถูกรบกวนจากสัญญลักษณ์ข้างเคียงมากกว่า 1 สัญญลักษณ์

เพราะว่าในทางปฏิบัติ สัญญาณ a(t) และ b(t) จะถูกจำกัดด้านแถบความถี่ ดังนั้นจะต้องถูกสุ่มตัวอย่าง (Sampling) ด้วยอัตราการสุ่มตัวอย่าง (Sampling Rate) ที่เพียงพอ ถ้ากำหนดให้ ϑ แทนคาบเวลาในการสุ่มตัวอย่าง ดังนั้น ค่าสัญญาณตัวอย่าง (Samples) ของ a(t) และ b(t) ที่คาบเวลาที่ n จะมีรูปสมการเป็นดังนี้

$$a(nT + j\vartheta) = \Sigma \alpha_i h[j\vartheta + (n - i)T]$$

$$b(nT + j\vartheta) = \Sigma \beta_i h[j\vartheta + (n - i)T] \qquad (3)$$

โดยที่ j เป็นเลขจำนวนเต็มชี้ตำแหน่งในการสุ่มตัวอย่างภายใน 1 คาบเวลา ถ้าให้สุ่มตัวอย่าง L ค่าต่อ 1 สัญญลักษณ์ จะได้ว่า $1 < j < L$

ในทางปฏิบัติการรบกวนกันระหว่างสัญญลักษณ์จะถูกจำกัดด้วยจำนวนจำกัดของสัญญลักษณ์ ถ้าให้ความยาวของการรบกวนระหว่างสัญญลักษณ์ในเทอมของสัญญลักษณ์เป็น 2M ดังนั้นจะสามารถเขียนสมการที่ 3 ได้เป็น

$$a(nT+j\delta) = \sum_{k=-M}^{M-1} \alpha_{-k+n} h(j\delta+kT) \qquad (4ก)$$

$$b(nT+j\delta) = \sum_{k=-M}^{M-1} \beta_{-k+n} h(j\delta+kT) \qquad (4ข)$$

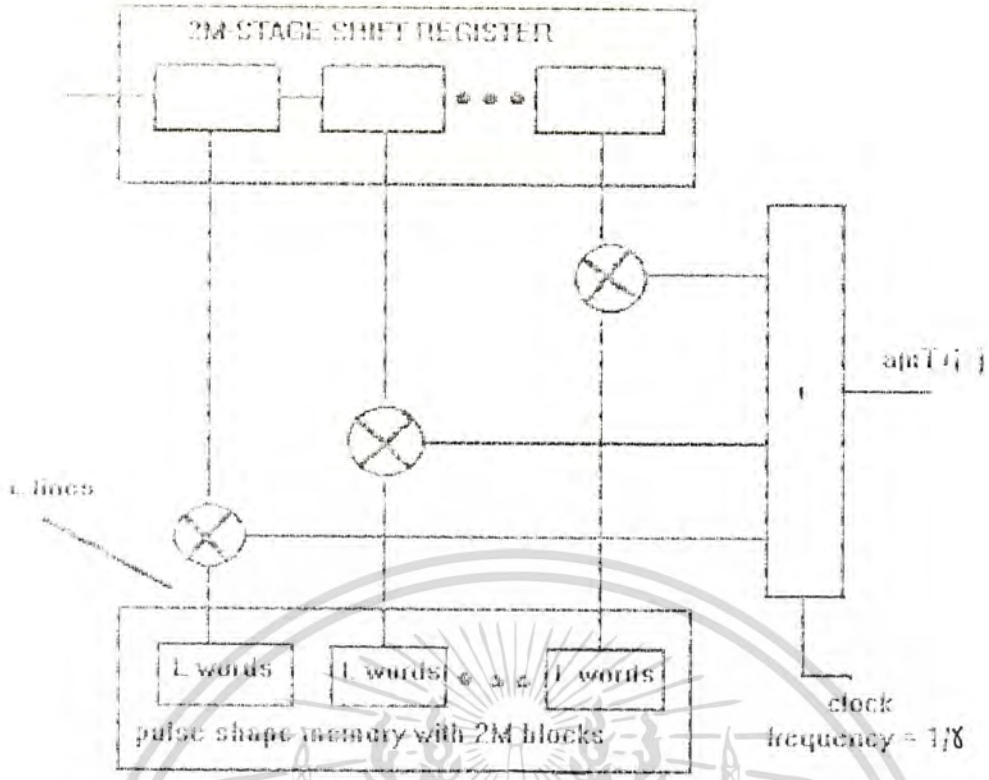
สมการที่ (4ก) และ (4ข) แสดงถึงแต่ละค่าของสัญญาณตัวอย่าง ของสัญญาณ a(t) และ b(t) ซึ่งเป็นผลลัพธ์มาจากการคูณ 2M ครั้ง จะเห็นว่า ต้องคูณ 2ML ครั้ง ต่อ 1 สัญญลักษณ์เพื่อสร้างสัญญาณ a(t),สำหรับ b(t) ก็เช่นเดียวกัน ดังนั้นจำนวนครั้งทั้งหมดของการคูณใน 1 วินาที ในการสร้างค่าสัญญาณตัวอย่างของสัญญาณ a(t) และ b(t) เท่ากับ (4ML)/T ซึ่งการเลือกค่า M, L และ ตัวประกอบต่าง ๆ ที่เหมาะสมจะทำให้มีความถูกต้องมากยิ่งขึ้น

2.2.2 สถาปัตยกรรมของระบบ

จากทฤษฎีของการมอดูเลทสัญญาณดิจิตอล (Digital Modulation) แบบต่าง ๆ สามารถเขียนเป็นแผนผังส่วนประกอบต่าง ๆ ของตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์ได้ดังรูปที่ 2.3

จากแผนผังดังรูป ข้อมูล 2 สถานะ (binary) จากแหล่งกำเนิดถูกป้อนเข้าวงจรแยกสัญญาณ (resolver) เพื่อหาค่าสัมประสิทธิ์ α_n และ β_n ซึ่งให้ค่าออกมาในอัตรา 1/T แล้วป้อนให้กับตัวสร้างสัญญาณตัวอย่าง (sample generator) จะได้สัญญาณตัวอย่างของ a(t) และ b(t) เป็น a(nT+jδ) และ b(nT+jδ) ตามลำดับ ในอัตราเท่ากับ 1/δ สัญญาณเหล่านี้ถูกเปลี่ยนเป็นสัญญาณต่อเนื่อง โดยผ่านตัวเปลี่ยนสัญญาณตัวเลขเป็นสัญญาณต่อเนื่อง (Digital-to-Analog Convertor (DAC)) และถูกผ่านไปยังวงจรกรองความถี่ต่ำ (Low Pass Filter (LPF)) แล้วนำไปมอดูเลทกับคลื่นพาห์ในตัวมอดูเลทแบบควอดเดรเจอร์ (Quadrature modulator) สัญญาณที่ได้ออกมาคือ c(t) ที่พร้อมจะส่งออกไป

ตัวแยกสัญญาณเพื่อหาค่าสัมประสิทธิ์ α และ β นั้น ในทางอุปกรณ์จะเป็นหน่วยความจำประเภท ROM โดยจะเก็บค่าสัมประสิทธิ์ α และ β ของการมอดูเลทไว้ทุกแบบ โดยการพิจารณาค่า α และ β ขึ้นอยู่กับสัญญาณไบนารีที่เข้ามาและการเลือกชนิดของการมอดูเลท



รูปที่ 2.4 แสดงบล็อคไดอะแกรมของตัวสร้างสัญญาณตัวอย่าง

เนื่องจากฟังก์ชันของตัวสร้างสัญญาณตัวอย่างของทั้งสองตัวเหมือนกัน เพราะฉะนั้นจะอธิบายเพียงตัวบนเท่านั้น ตัวสร้างสัญญาณตัวอย่างประกอบด้วยชิฟรีจิสเตอร์ 2M ชั้นเก็บค่าที่เป็นปัจจุบันของ α_n จำนวน 2M ค่าและมีหน่วยความจำขนาด 2M บล็อคสำหรับเก็บรูปร่างพัลส์ แต่ละบล็อคของหน่วยความจำนี้ประกอบด้วย L เวิร์ด (word) ซึ่งแต่ละเวิร์ดจะเก็บค่าสัญญาณตัวอย่างของรูปร่างพัลส์ โดยที่ขนาดของเวิร์ดขึ้นอยู่กับผลของการจัดระดับสัญญาณ (Quantization) ชิฟรีจิสเตอร์จะถูกควบคุมโดยสัญญาณนาฬิกาความถี่ 1/T และหน่วยความจำของรูปร่างพัลส์จะถูกควบคุมด้วยสัญญาณนาฬิกาความถี่ 1/δ โดย T/δ = L ตัวคูณจะทำงาน 1/δ ครั้งต่อวินาที และตัวบวกจะทำการรวมสัญญาณที่เข้ามา 2M ค่าเพื่อส่งไปเป็นค่าสัญญาณตัวอย่างของ a(t)

2.2.3 การวิเคราะห์ค่าสัมประสิทธิ์ α และ β

โดย α เป็นตัวกำหนดสัมประสิทธิ์ของ $\cos w_c t$ ส่วน β เป็นตัวกำหนดสัมประสิทธิ์ของ $\sin w_c t$ ในที่นี้จะอธิบายถึงการหาค่า α และ β ของการมอดูเลทแบบ PSK และ QAM ซึ่งเป็นรูปแบบที่นิยมใช้กัน ส่วนรูปแบบอื่นที่ไม่ได้กล่าวถึงก็สามารถหาค่า α และ β โดยใช้วิธีการพิจารณาเช่นเดียวกับ PSK และ QAM

กรณีของ PSK ถ้าพิจารณาจากเอาท์พุทของ PSK แบบต่าง ๆ จะพบว่า เอาท์พุทของ PSK ในแต่ละแบบนั้นจะมีค่าแอมปลิจูดของเอาท์พุทเท่ากันทุกค่า แต่ละค่าจะต่างกันเนื่อง

จากมุม ϕ ที่ต่างกัน (ทำมุมกับแกนอ้างอิงในทิศทางของมุม $+\phi$ หรือมุม $-\phi$) เพราะฉะนั้นจะได้ข้อสรุปว่า สัมประสิทธิ์ที่เป็นตัวกำหนดเฟเซอร์ค่าต่าง ๆ หรือ แผนผังคอนสเตลเลชั่น (constellation) ตำแหน่งต่าง ๆ จะเป็นฟังก์ชั่นของไซนูซอยดอล (sinusoidal) ของมุม ϕ เพียงอย่างเดียว กรณีของ QAM เมื่อพิจารณาค่าเอาท์พุทของ QAM จะพบว่าสัมประสิทธิ์ที่เป็นตัวกำหนดเฟเซอร์ค่าต่าง ๆ หรือคอนสเตลเลชั่นตำแหน่งต่าง ๆ จะเป็นฟังก์ชั่นไซนูซอยดอลของมุม ϕ และฟังก์ชันของแอมปลิจูดด้วย

มุม ϕ ในการวิเคราะห์นี้ คือมุมเล็กระหว่างเฟเซอร์กับแกนอ้างอิง เฟเซอร์จะทำมุมกับแกนอ้างอิง 2 แกนที่ตั้งฉากกัน ซึ่งจะทำให้เกิดมุม 2 มุม โดยจะใช้มุมที่เล็กระหว่างมุม 2 มุมนั้นเป็นมุม ϕ ดังรูปที่ 2.5 จะเห็นว่า มุม #1 เล็กกว่ามุม #2 เพราะฉะนั้นมุม #1 จึงเป็นมุม ϕ

รูปที่ 2.5 แสดงมุม ϕ ที่เกิดขึ้น

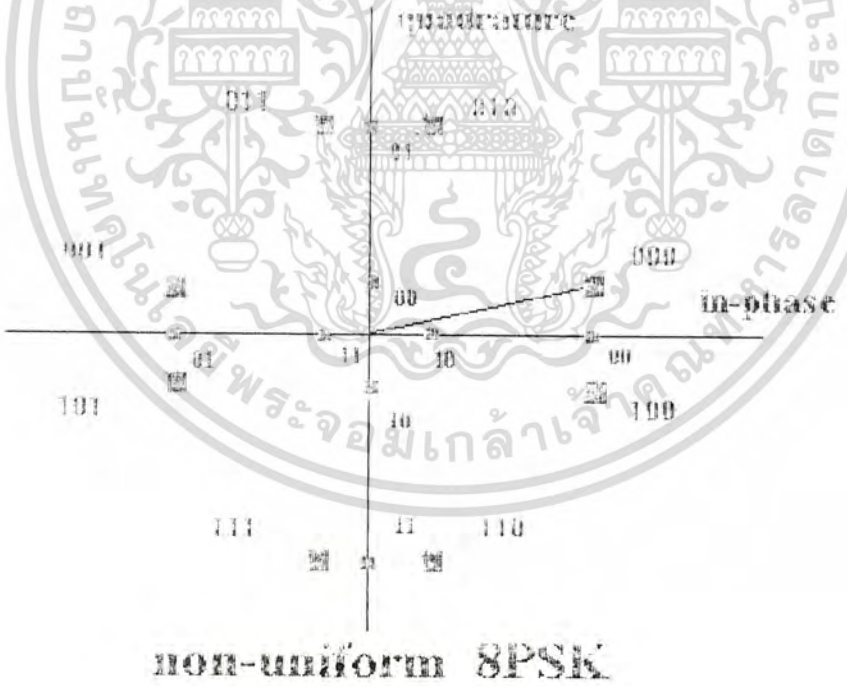
จากสมการ

$$c(t) = a(t)\cos w_c t + b(t)\sin w_c t$$

เมื่อจัดให้อยู่ในฟังก์ชันของ sine หรือ cosine และ ถ้าแกนอ้างอิงอินเฟสเป็น $\cos w_c t$ เฟสที่ได้ในรูป $w_c t-\phi$ จะคือมุมที่เปลี่ยนจาก $w_c t$ ไปทางซ้ายของแกนอ้างอิงเท่ากับ ϕ ส่วนเฟสที่ได้ในรูป $w_c t+\phi$ จะคือมุมที่เปลี่ยนจาก $w_c t$ ไปทางขวาของแกนอ้างอิงเท่ากับ ϕ แต่ถ้าแกนอ้างอิงอินเฟสเป็น $\sin w_c t$ เฟสที่ได้ในรูป $w_c t-\phi$ จะคือมุมที่เปลี่ยนจาก $w_c t$ ไปทางขวาของแกนอ้างอิงเท่ากับ ϕ ส่วนเฟสที่ได้ในรูป $w_c t+\phi$ จะคือมุมที่เปลี่ยนจาก $w_c t$ ไปทางซ้ายของแกนอ้างอิงเท่ากับ ϕ วิธีการพิจารณามุม ϕ

จะไม่ใช่ค่าเฟสที่ชิฟไปจากแกนอ้างอิง $\cos w_c t$ หรือ $\sin w_c t$ เพียงแกนใดแกนหนึ่งแต่จะเป็นการคิดเฟสที่ชิฟจากแกนอ้างอิงทั้ง 4 แกน ซึ่งจะได้ คอนสเตลเลชั่นไดอะแกรมแบบนอนยูนิฟอร์ม

ค่า α และ β ของการมอดูเลทแบบใดนั้น จะขึ้นอยู่กับเงื่อนไขเบื้องต้นที่กำหนดไว้ ถึงแม้ว่าเป็นการมอดูเลทแบบเดียวกันแต่ถ้าการกำหนดเงื่อนไขเบื้องต้นต่างกัน ค่า α และ β ที่ได้ก็จะต่างกันด้วย ดังนั้นการใช้ค่า α และ β เป็นตัวกำหนดรูปแบบการมอดูเลทจะมีความยืดหยุ่นเป็นอย่างมาก ในรูปแบบการมอดูเลทแบบเดียวกันสามารถกำหนด α และ β เพื่อให้ได้รูปแบบของเอาท์พุท (constellation หรือ ตำแหน่งของกลุ่มบิทต่างๆ) เป็นไปตามที่ต้องการได้ ไม่จำกัดเพียงรูปแบบใดรูปแบบหนึ่งเหมือนวิธีการโดยทั่วไป ในที่นี้จะยกตัวอย่างค่า α, β และคอนสเตลเลชั่นไดอะแกรมของ 8 PSK ส่วน α, β และคอนสเตลเลชั่นไดอะแกรมของการมอดูเลทแบบอื่นดูได้จากบทแทรก



non-uniform 8PSK

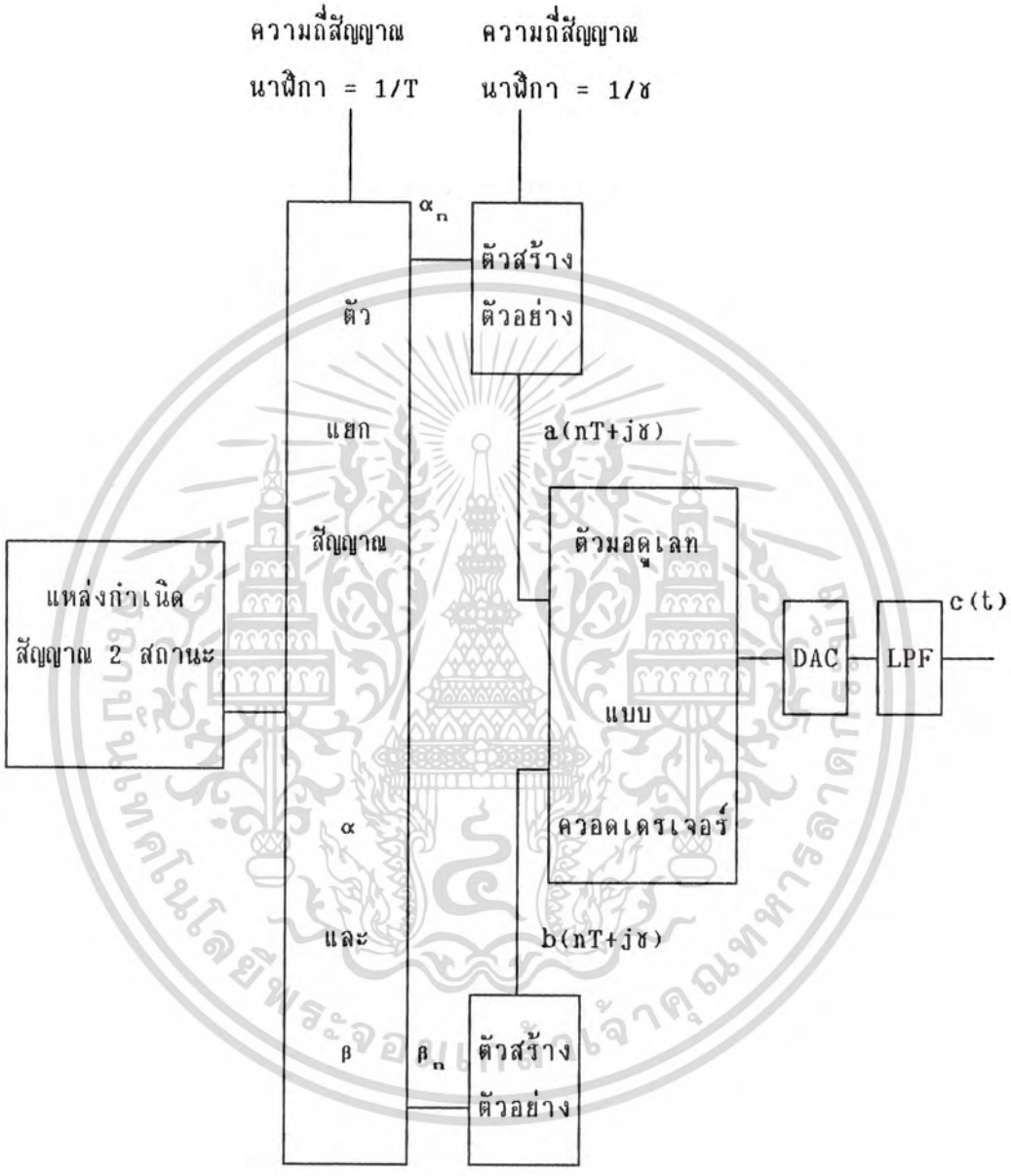
รูปที่ 2.6

TWO-BIT STATE	α_1	β_1
00	COS θ	SIN θ
01	-COS θ	COS θ
10	SIN θ	-SIN θ
11	-SIN θ	-COS θ

8 PSK

ตาราง2.2

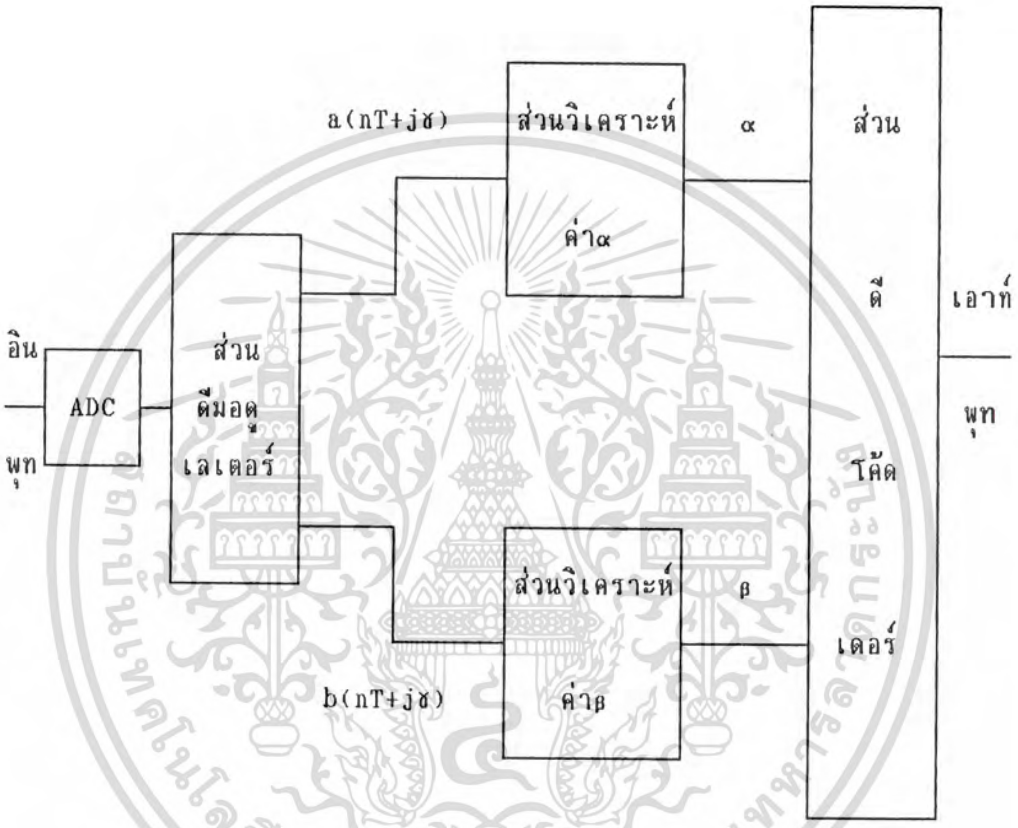
ปริญญานิพนธ์นี้ได้นำซอฟท์แวร์มาทำงานแทนอุปกรณ์ฮาร์ดแวร์ ซึ่งหลักการทำงานของซอฟท์แวร์เป็นไปตามแผนผังรูปที่ 2.7 จากแผนผังส่วนของซอฟท์แวร์จะเริ่มจากแหล่งกำเนิดสัญญาณสองสถานะจนถึงส่วนมอดูเลทแบบควอดเดรเจอร์หลังจากนั้นสัญญาณที่ได้จะนำไปผ่านวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก (Digital to Analog Converter) และวงจรกรองความถี่ต่ำผ่าน (Low Pass Filter) การทำงานในส่วนของควอดเดรเจอร์จะใช้คลื่นพาห์ที่เป็นดิจิตอลมอดูเลทกับสัญญาณข่าวสารดิจิตอล การทำงานในลักษณะนี้จะเป็นแบบนอนรีลไทม์ (Non-real Time) ซึ่งข้อมูลต่างๆจะถูกนำไปเก็บไว้ในหน่วยความจำก่อนแล้วจึงนำมาผ่านการจัดการข้อมูลเพื่อส่งสัญญาณต่อไป



รูปที่ 2.7

2.2.4 ส่วนประกอบทางด้านรับ

ทางด้านรับจะมีแผนผังส่วนประกอบต่างๆดังนี้



รูปที่ 2.8

ขั้นตอนการทำงานของด้านรับจะทำงานตรงข้ามกับด้านส่งดังนี้ สัญญาณที่เข้ามาทางด้านรับจะผ่านตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล นำไปผ่านส่วนตีมอดูเลเตอร์ได้ค่า a(nT+jϑ) และ b(nT+jϑ) เพื่อนำไปวิเคราะห์หาค่า α และ β จากนั้นจะผ่านส่วนดีโค้ดเดอร์ได้สัญญาณข้อมูลดิจิตอลเหมือนกับทางด้านส่ง

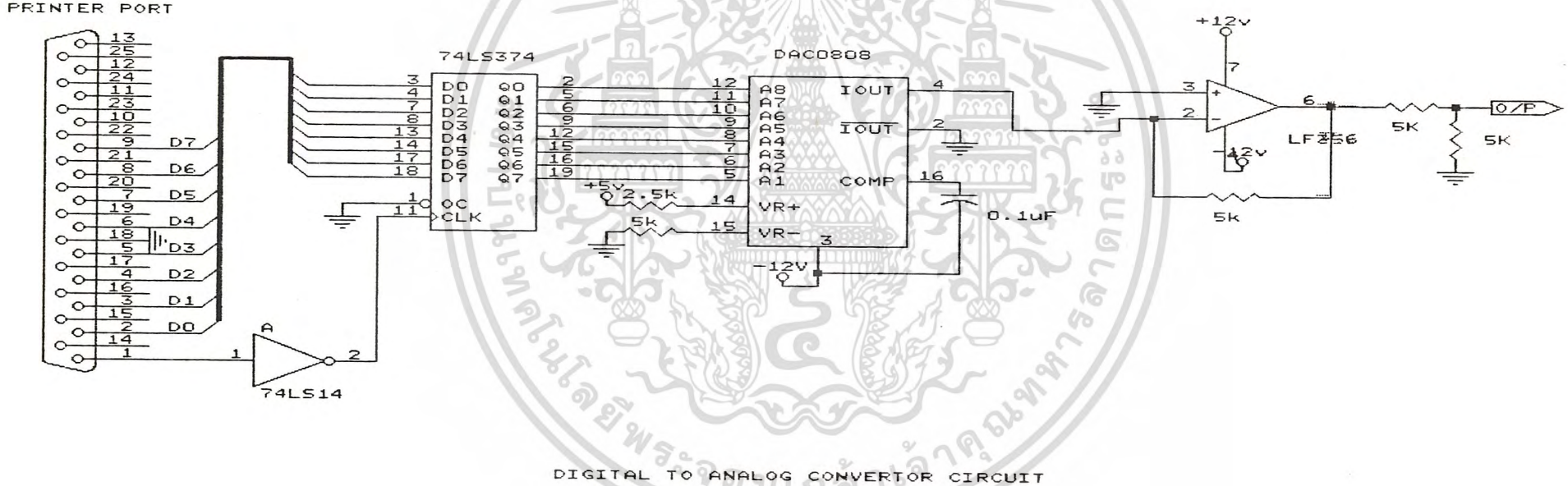
3.ผลการทดลองและการวิเคราะห์ผล

ในการทดลองได้ทำการส่งไฟล์สัญญาณผ่านวงจรดังรูป 3.1 โดยใช้โปรแกรม OUTPORT.C ซึ่งไฟล์สัญญาณที่จะส่งได้จากการมอดูเลทไฟล์ข้อมูลเข้ากับคลื่นพาห์ในโปรแกรม MODULATE.C สัญญาณที่ส่งไปในสายส่งสามารถแสดงได้ดังรูป 3.26-3.32 สัญญาณที่ส่งมาจะผ่านวงจรทางด้านรับดังรูป 3.2 ซึ่งสัญญาณที่รับได้จะนำไปเปรียบเทียบกับสัญญาณก่อนส่งแสดงในรูป 3.33 ซึ่งจะเห็นได้ว่าเกิดการเลื่อนทางเฟส (phase shift) และการลดขนาดของสัญญาณ ซึ่งอาจจะเกิดจากข้อจำกัดของอุปกรณ์ที่ใช้ในวงจร เช่น ไอซี ADC ที่ใช้มีความเร็วในการแปลงสัญญาณ (conversion time) น้อยกว่าของไอซี DAC, ออปแอมป์ที่ใช้รักษาระดับสัญญาณอ้างอิง (V_{REF}) ของADC ไม่มีเสถียรภาพเท่าที่ควร และความไม่สอดคล้องของเวลาในการสุ่มสัญญาณระหว่างด้านส่งกับด้านรับ

สัญญาณที่รับได้จะนำมาดีมอดูเลทโดยโปรแกรม RING.C และนำมาเข้าโปรแกรมกรองความถี่ต่ำผ่าน (Low Pass Filter) จะได้สัญญาณ a(t) และ b(t) ซึ่งจะนำมาเปรียบเทียบกับสัญญาณ a(t) และ b(t) ก่อนที่จะส่งแสดงดังรูป 3.34 และ 3.35ซึ่งจะเห็นได้ว่าสัญญาณที่ดีมอดูเลทได้มีสัญญาณรบกวน , มีความคลาดเคลื่อนทางเฟส และการลดขนาดของสัญญาณ อันเนื่องมาจากโปรแกรมดีมอดูเลทและกรองความถี่ต่ำผ่านยังไม่ดีพอ

หลังจากได้ a(t)และb(t)แล้วจะนำไปเข้าโปรแกรม DEMOD.C เพื่อทำการถอดรหัสสัญญาณให้ได้ข้อมูลตามที่ต้องการ

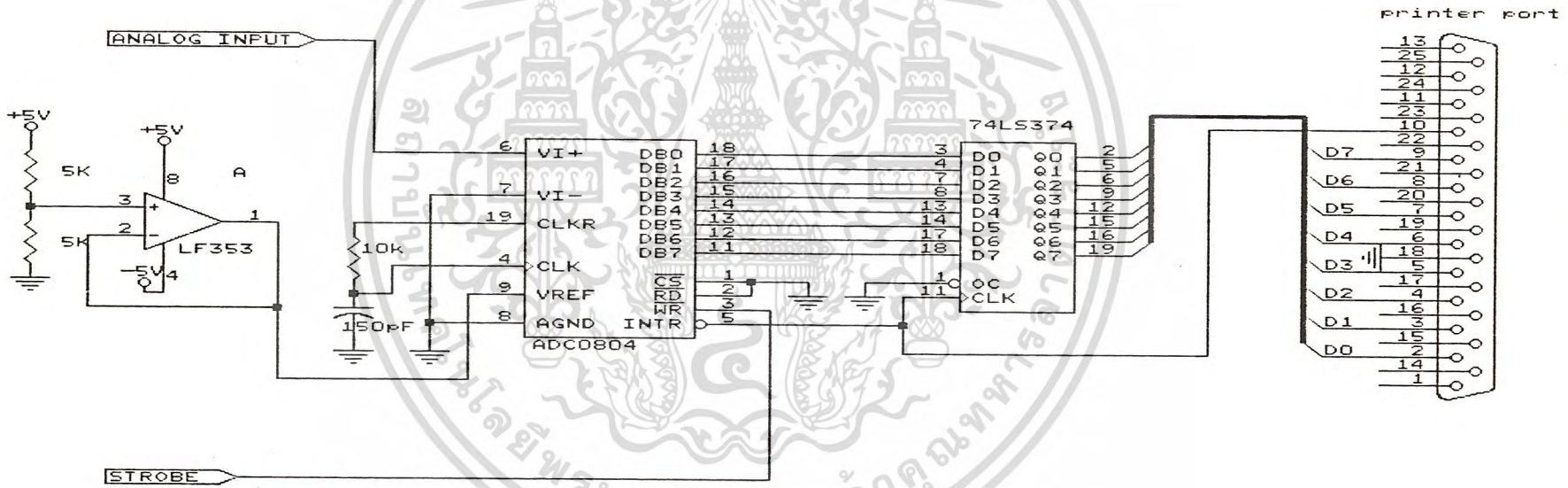
โดยในการทดลองนี้ได้แสดงตัวอย่างของการเปรียบเทียบสัญญาณที่มอดูเลทแบบ QPSK ซึ่งจากการทดลองแสดงให้เห็นว่าทฤษฎีที่ได้เสนอมานี้มีความเป็นไปได้ที่จะนำไปประยุกต์ใช้งานได้จริง



DIGITAL TO ANALOG CONVERTOR CIRCUIT

รูปที่ 3.1

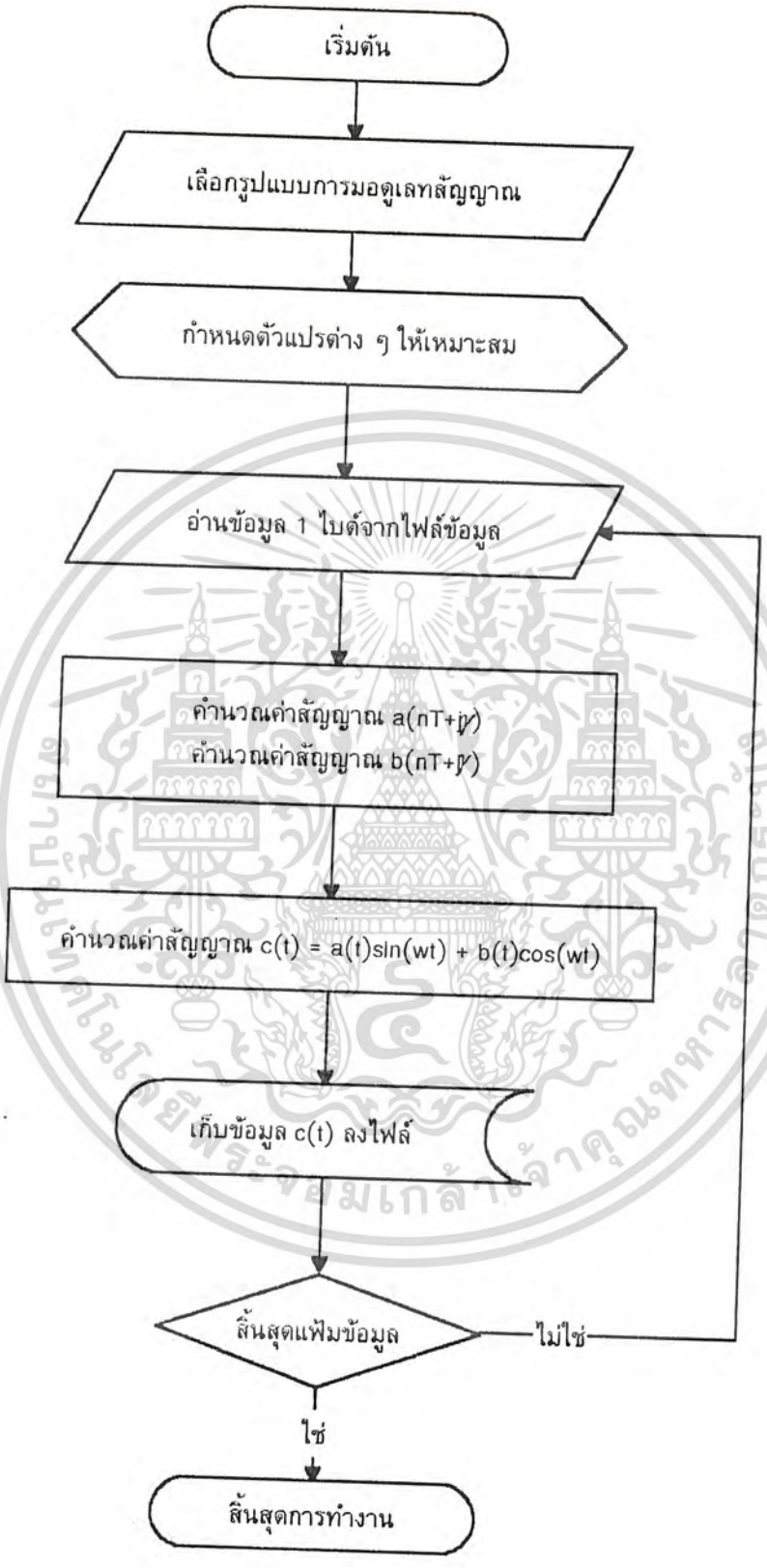
Size	Document Number	REV
A		
Date: March 21, 1994	Sheet of	



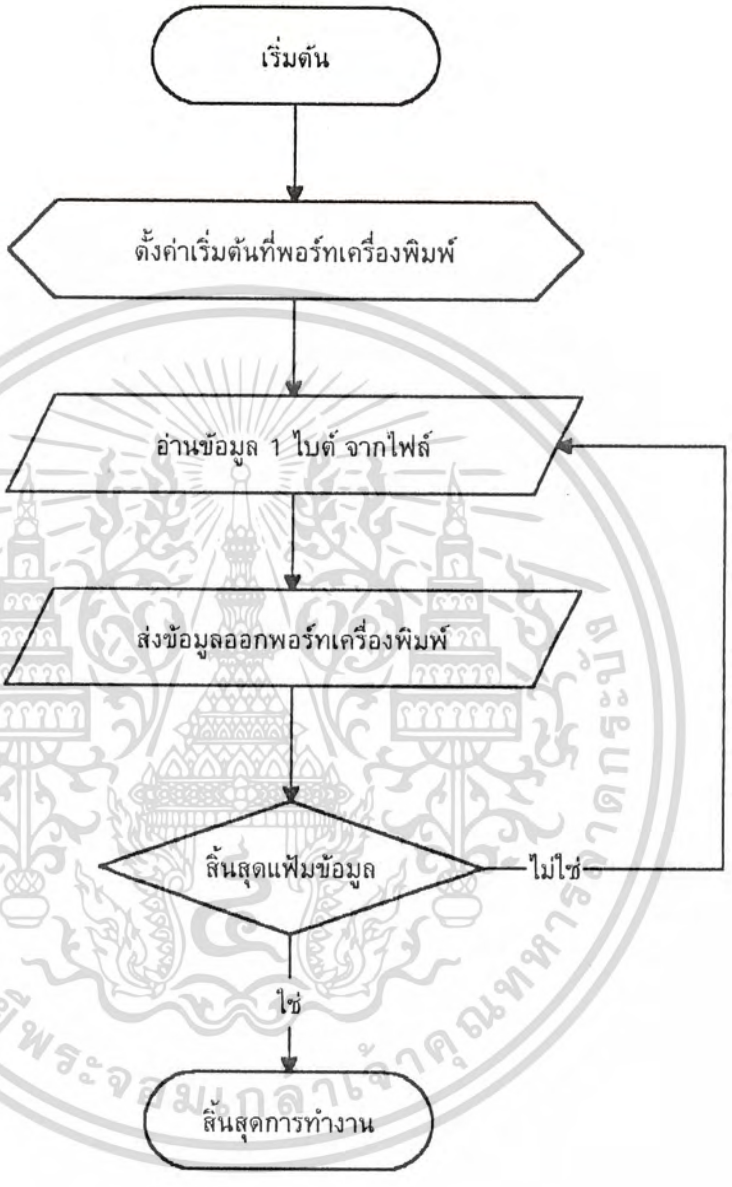
ANALOG TO DIGITAL CONVERTOR CIRDUIT

รูปที่ 3.2

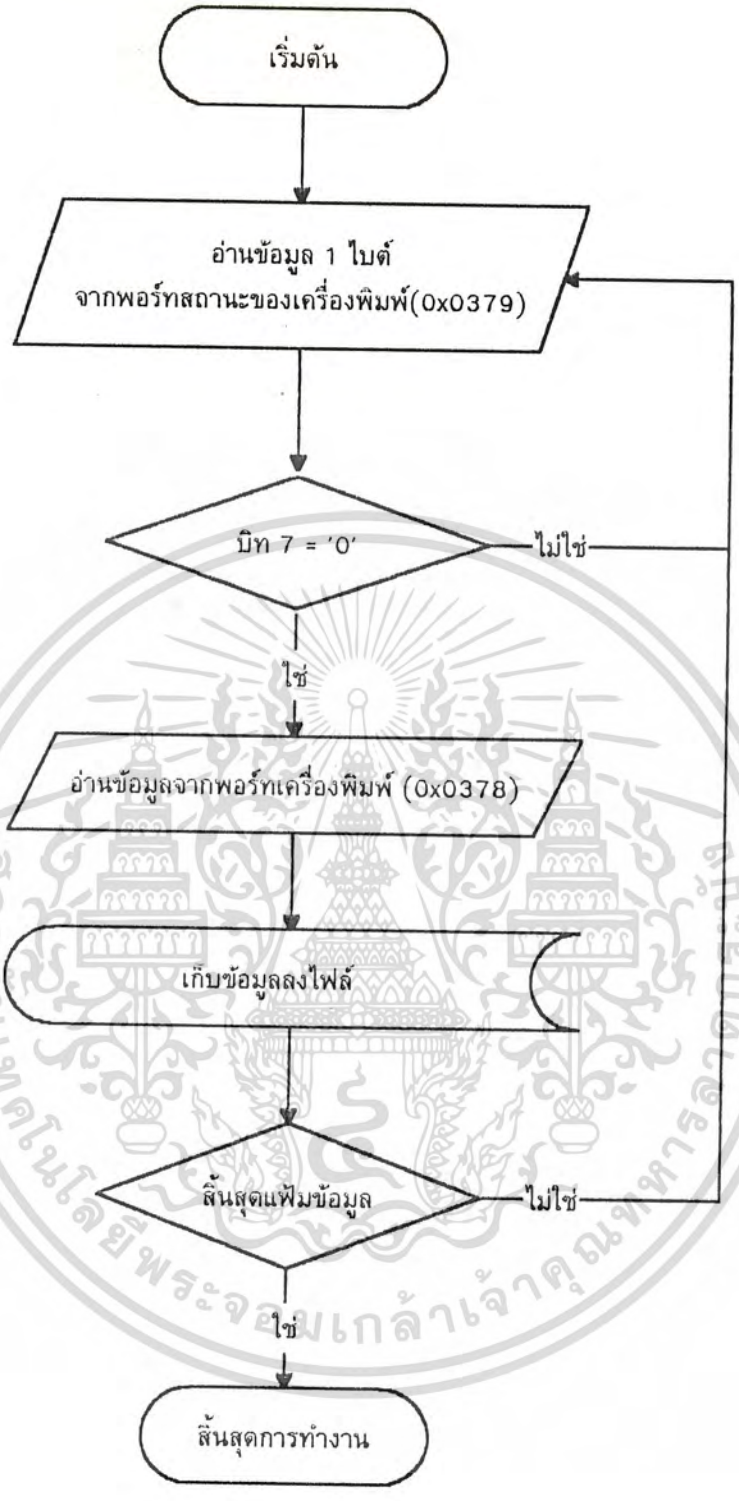
Size	Document Number	REV
A		
Date: March 21, 1994	Sheet of	



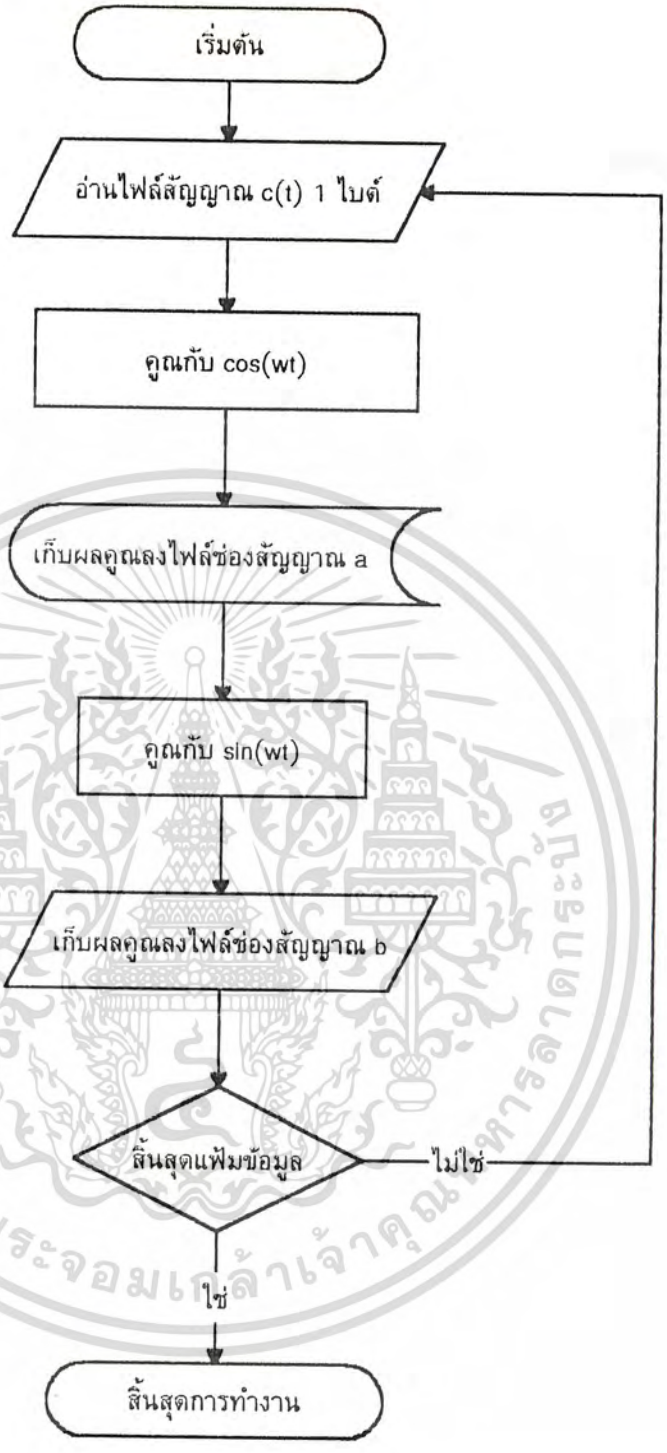
รูปที่ 3.3 การทำงานของโปรแกรม MODULATE.C



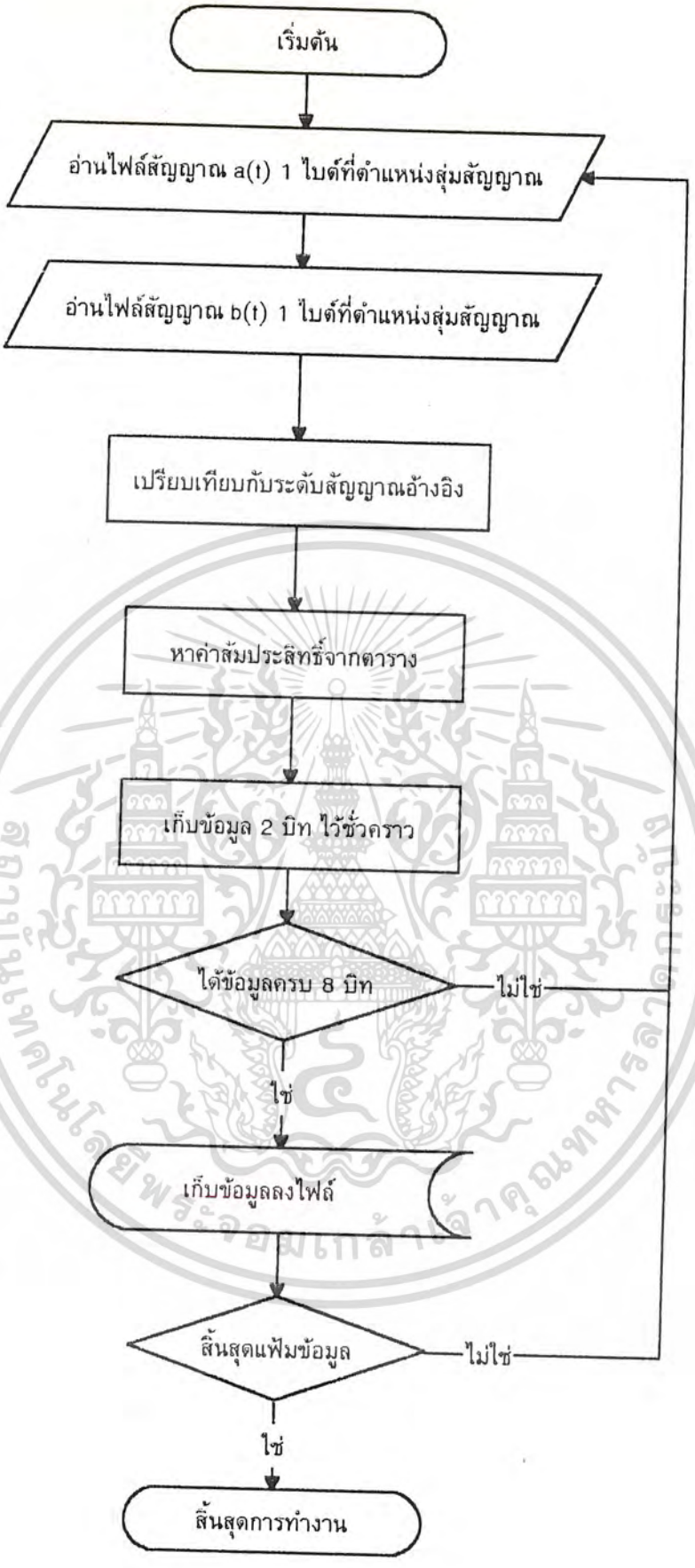
รูปที่ 3.4 แสดงการทำงานของโปรแกรม OUTPORT.C



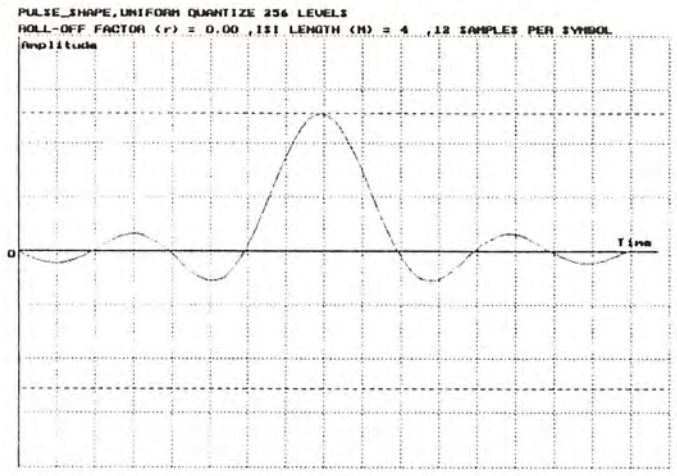
รูปที่ 3.5 แสดงการทำงานของโปรแกรม INPORT.C



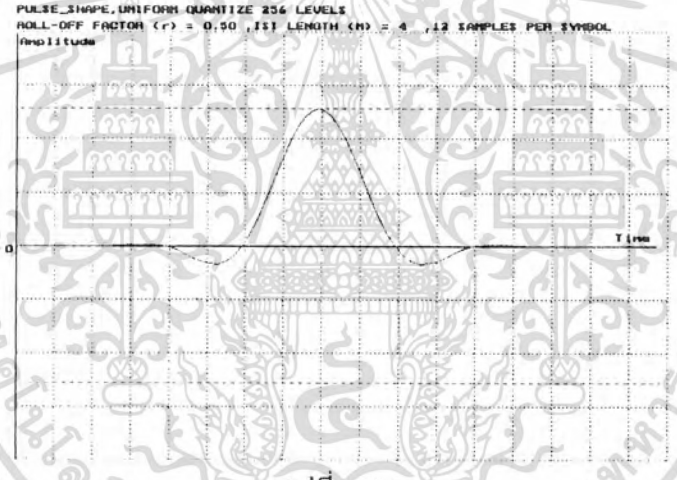
รูปที่ 3.6 การทำงานของโปรแกรม RING.C



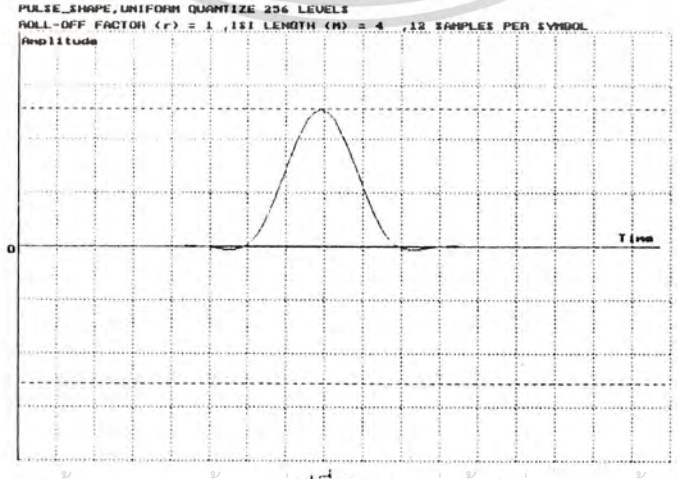
รูปที่ 3.7 การทำงานของโปรแกรม DEMOD.C



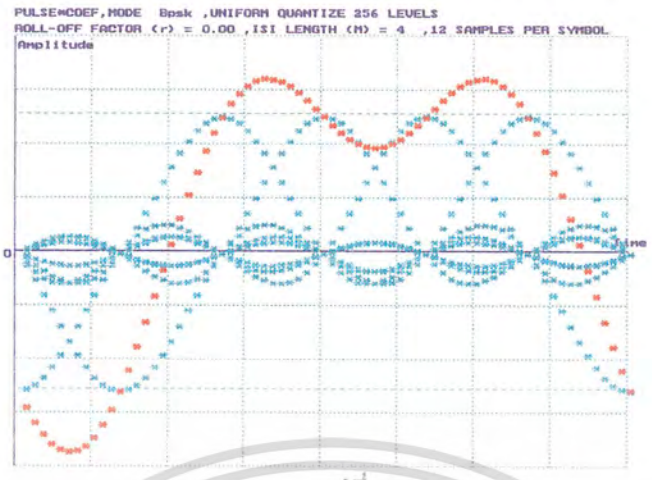
รูปที่ 3.8



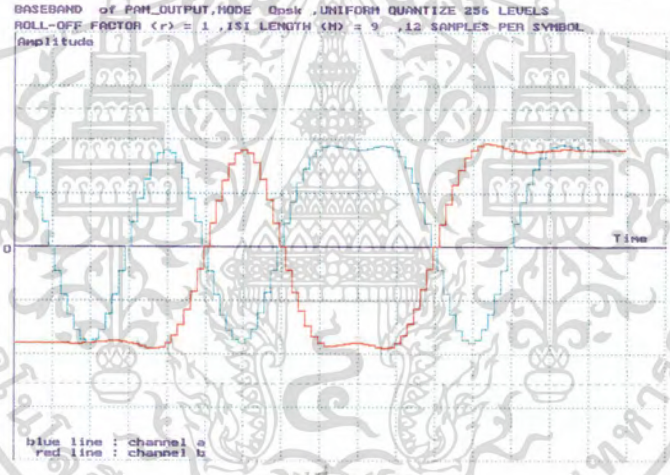
รูปที่ 3.9



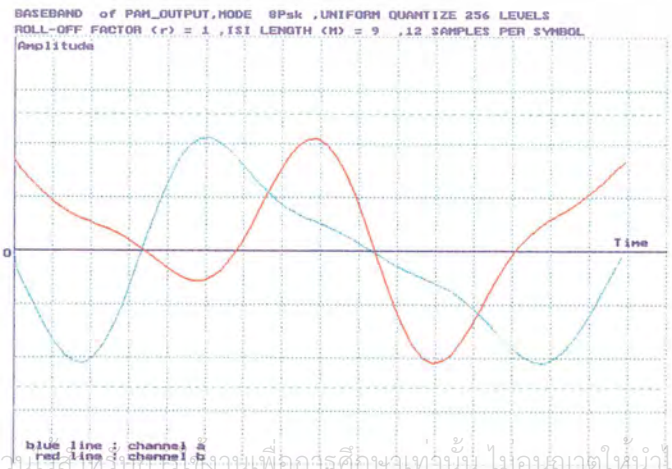
รูปที่ 3.10



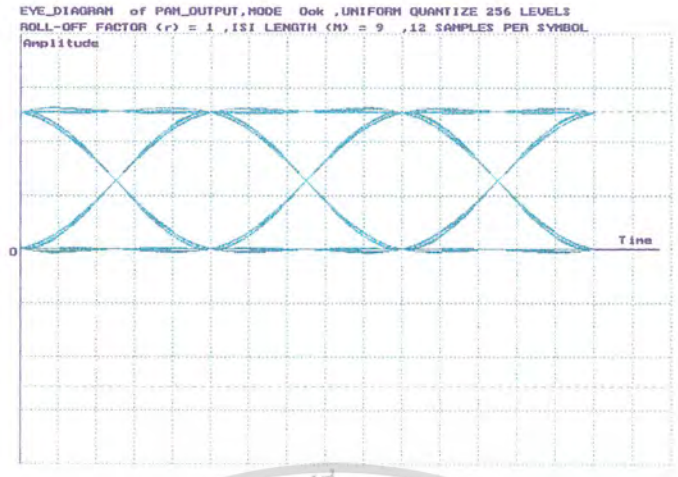
รูปที่ 3.11



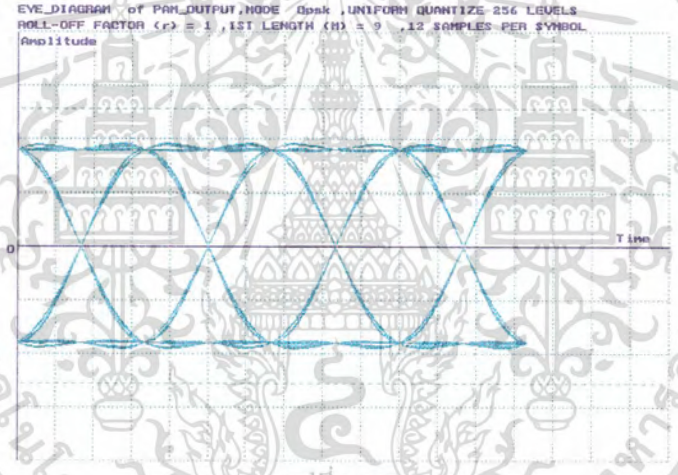
รูปที่ 3.12



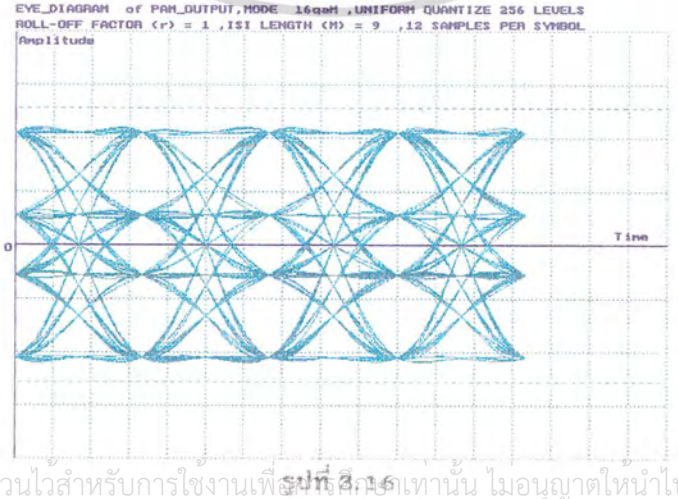
รูปที่ 3.13



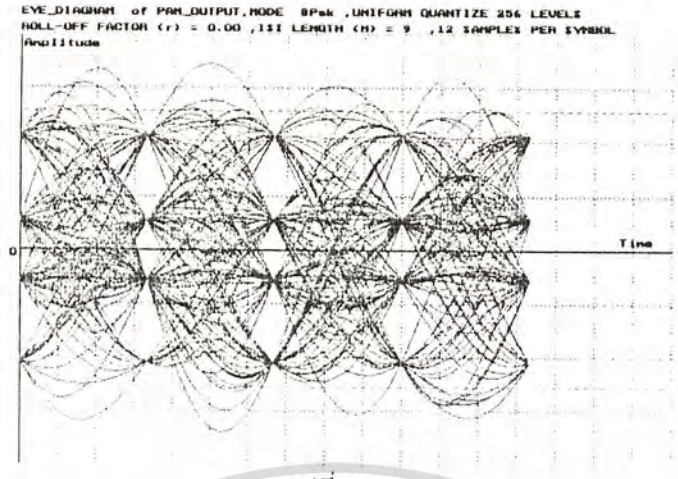
รูปที่ 3.14



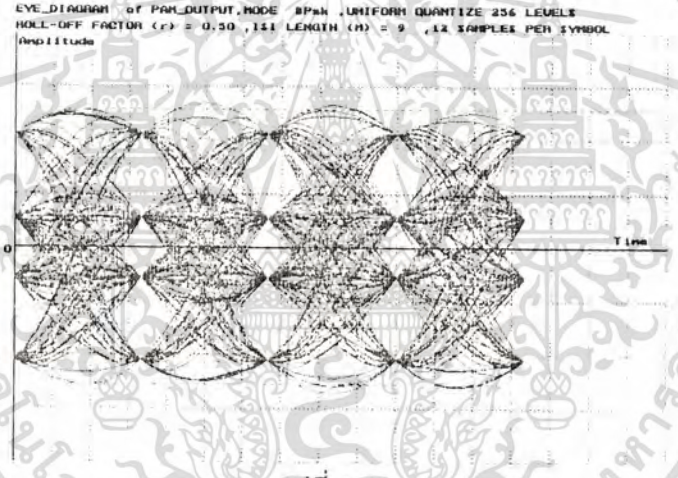
รูปที่ 3.15



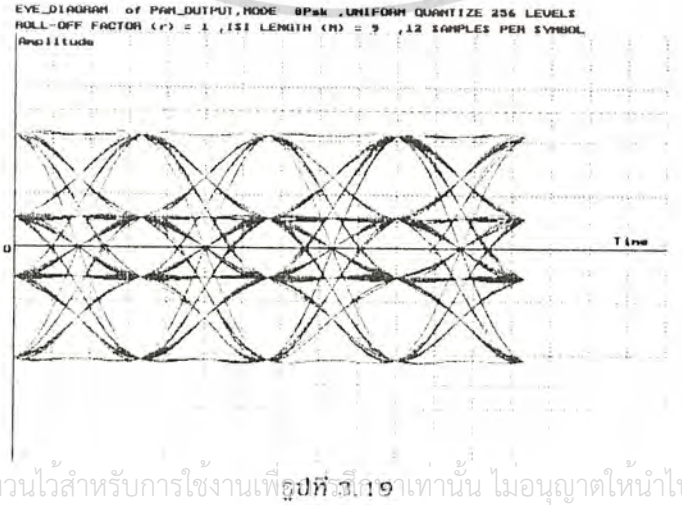
รูปที่ 3.16



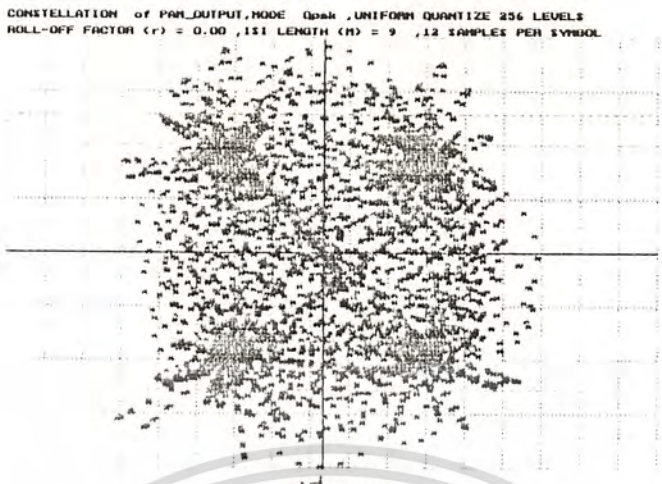
รูปที่ 3.17



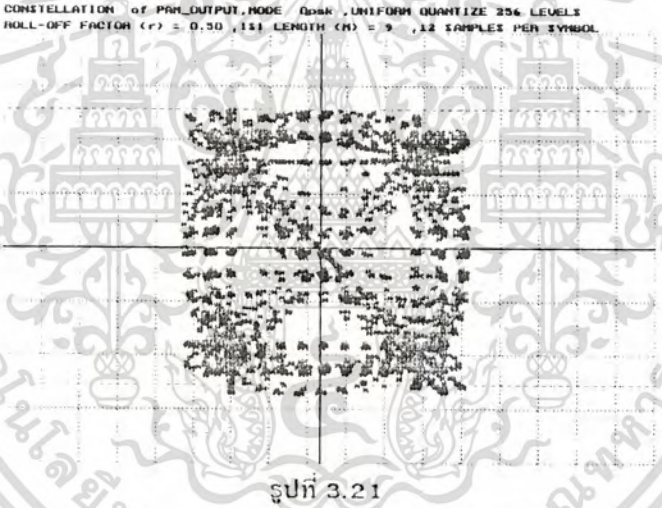
รูปที่ 3.18



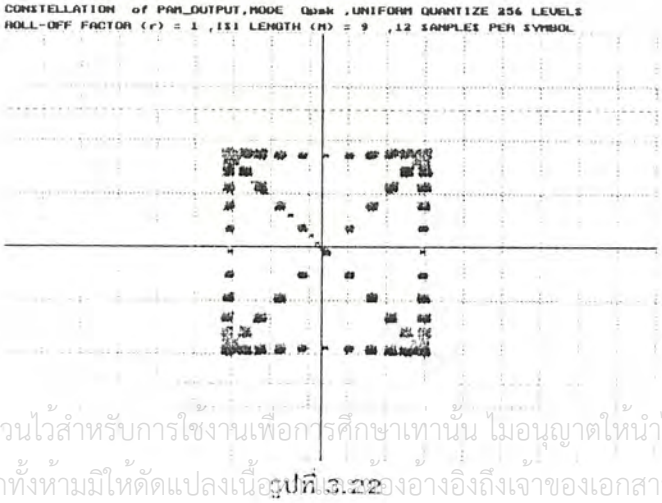
รูปที่ 3.19



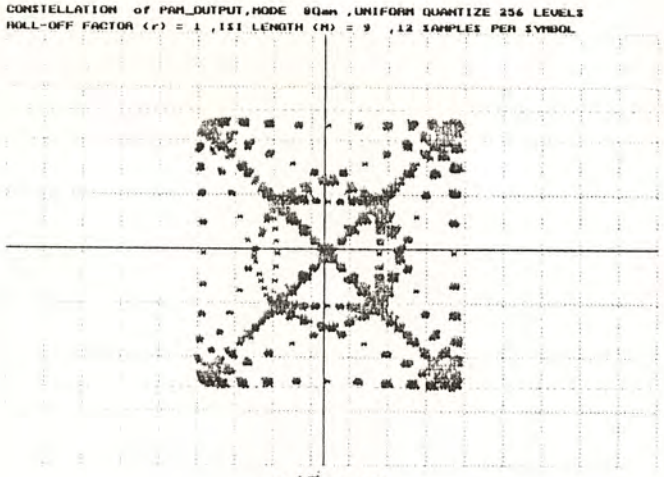
รูปที่ 3.20



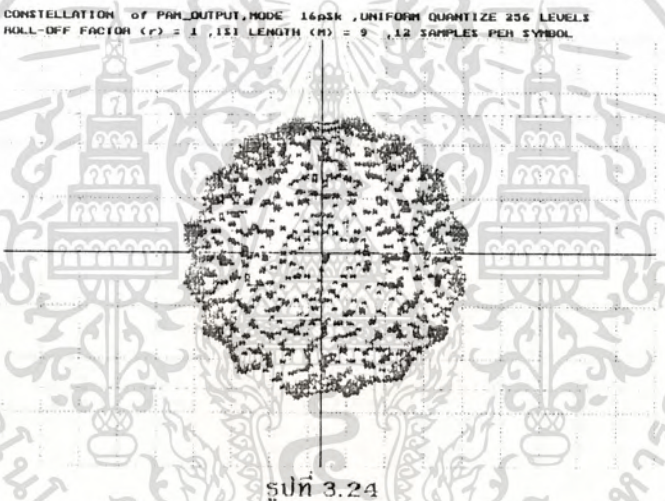
รูปที่ 3.21



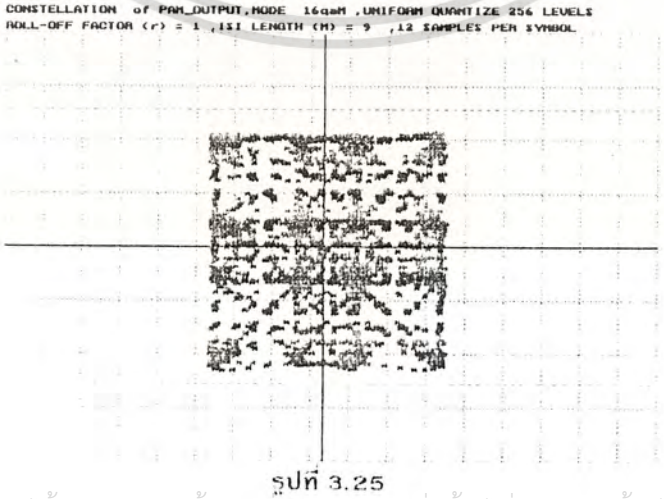
รูปที่ 3.22



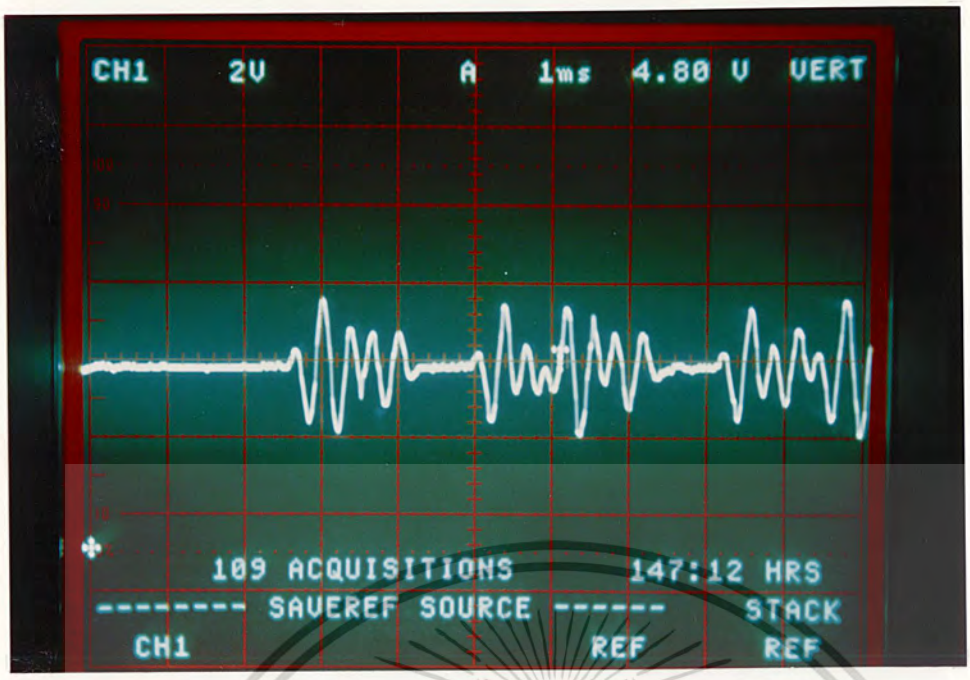
รูปที่ 3.23



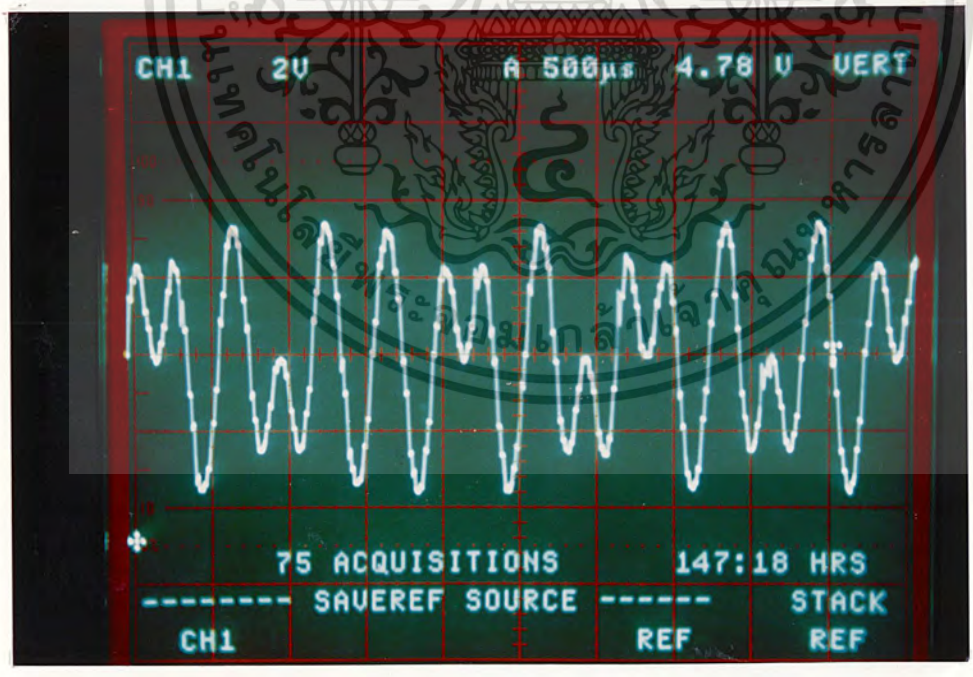
รูปที่ 3.24



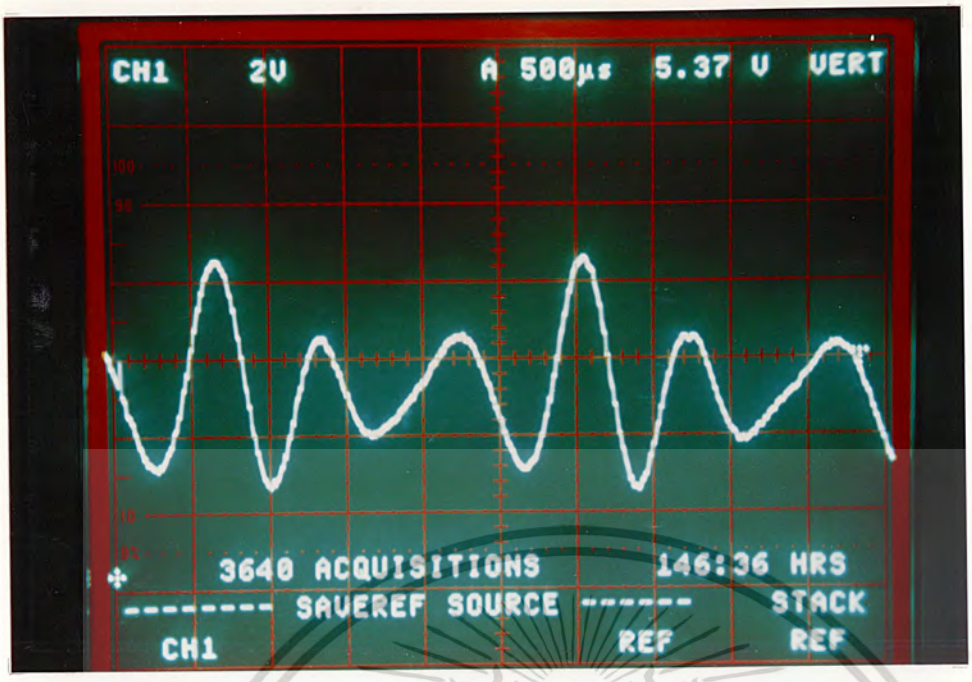
รูปที่ 3.25



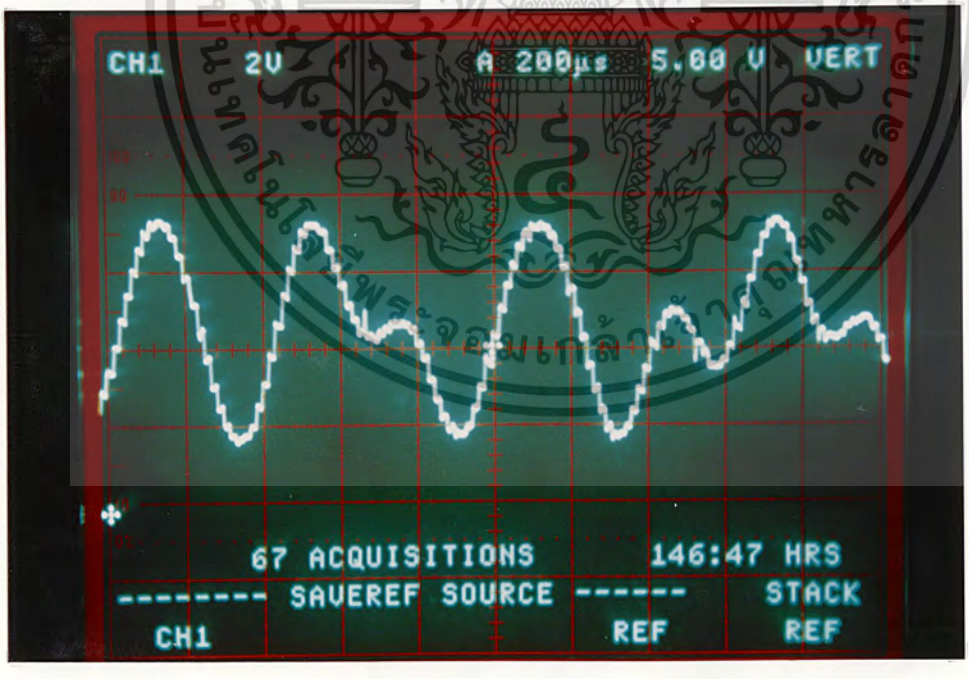
รูปที่ 3.26 แสดงรูปสัญญาณ c(t) แบบ OOK



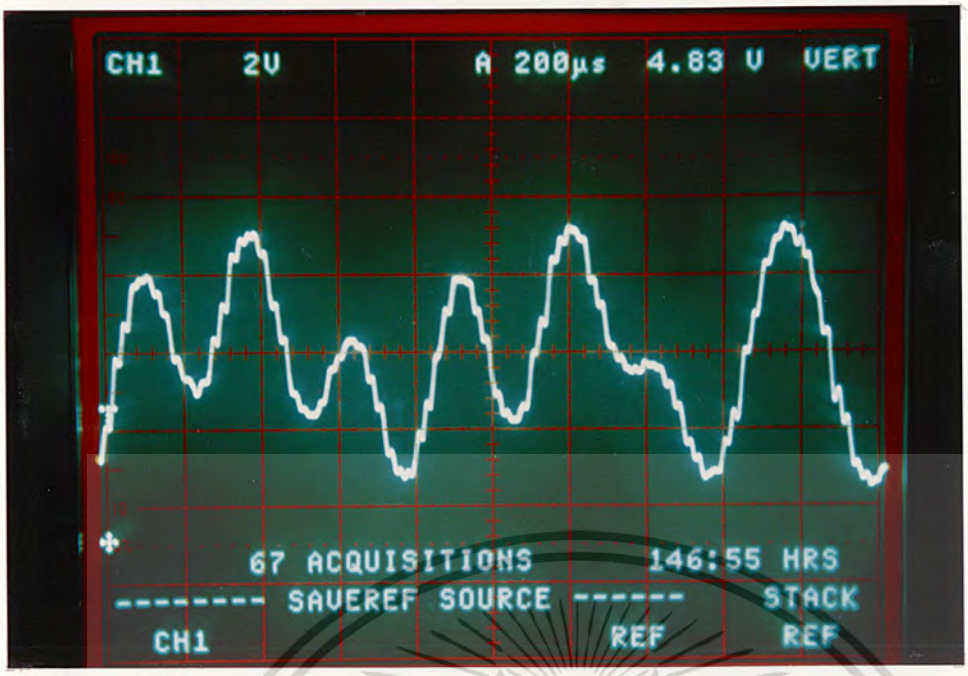
รูปที่ 3.27 แสดงรูปสัญญาณ c(t) แบบ BPSK



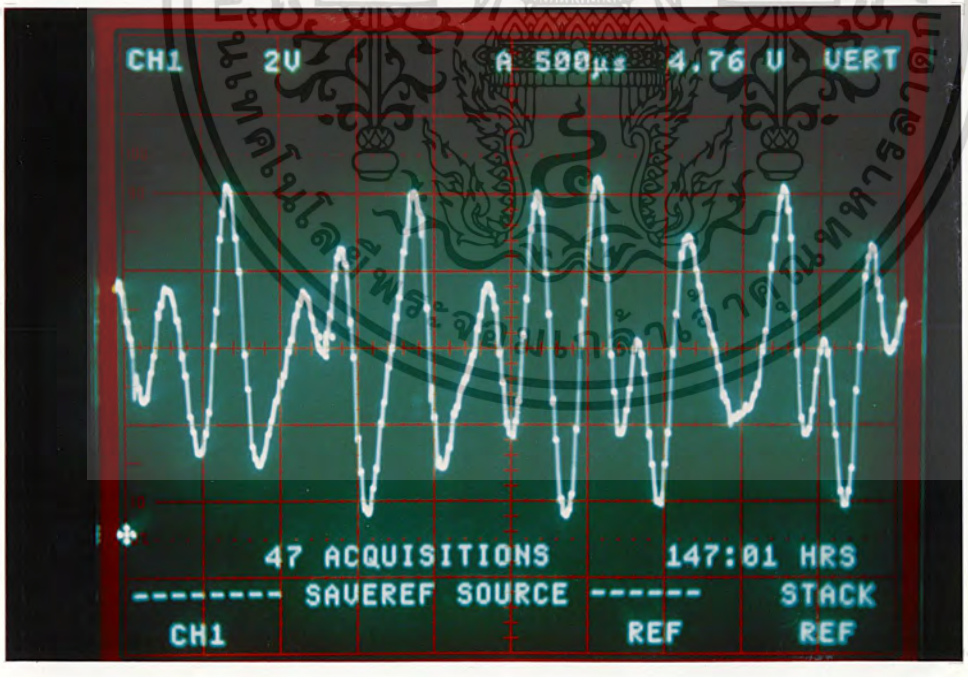
รูปที่ 3.28 แสดงรูปสัญญาณ c(t) แบบ QPSK



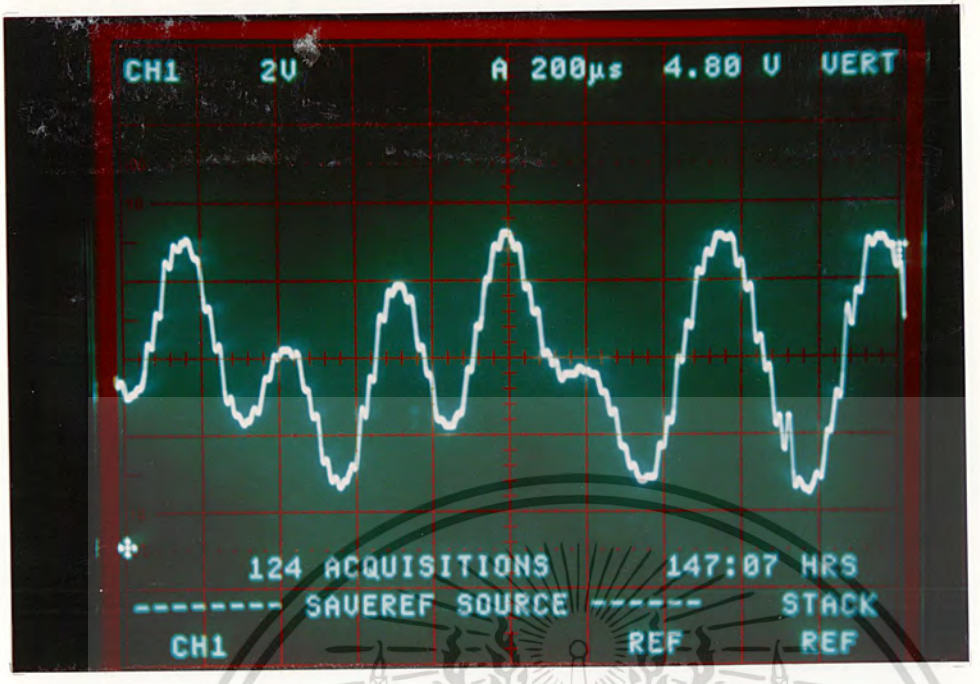
รูปที่ 3.29 แสดงรูปสัญญาณ c(t) แบบ 8PSK



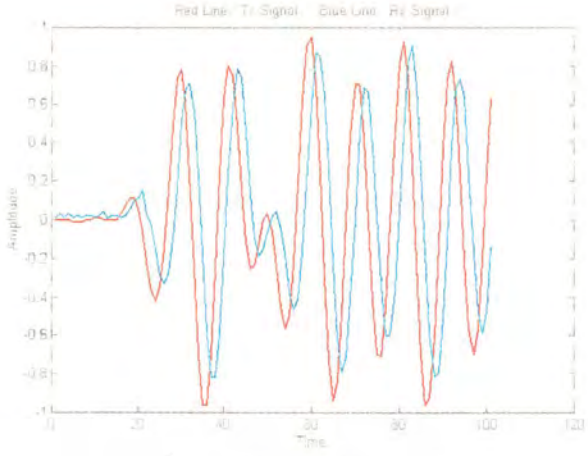
รูปที่ 3.30 แสดงรูปสัญญาณ c(t) แบบ 16PSK



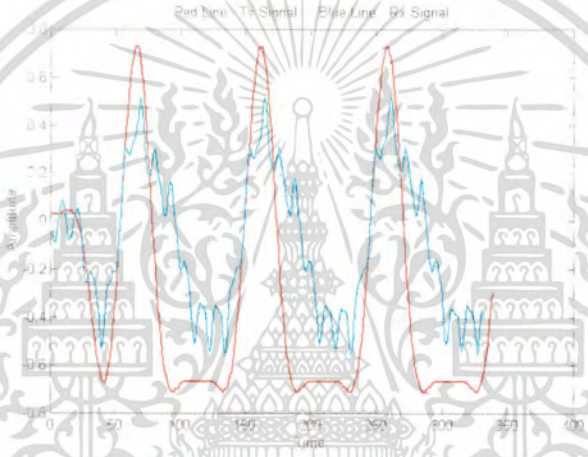
รูปที่ 3.31 แสดงรูปสัญญาณ c(t) แบบ 8QAM



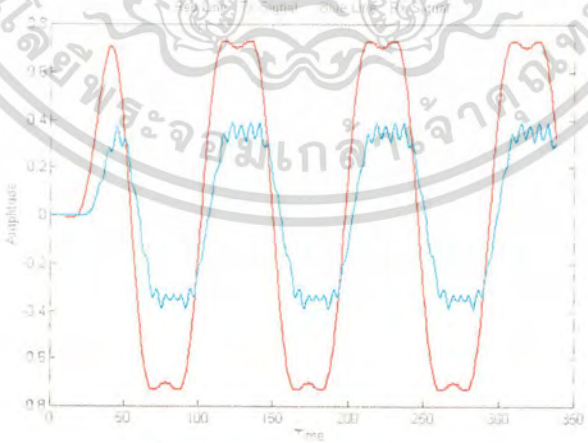
รูปที่ 3.32 แสดงรูปสัญญาณ c(t) แบบ 16QAM



รูปที่ 3.33 เปรียบเทียบสัญญาณ c(t)



รูปที่ 3.34 เปรียบเทียบสัญญาณ a(t)



รูปที่ 3.35 เปรียบเทียบสัญญาณ b(t)

4.สรุป

การมอดูเลทสัญญาณดิจิตอลเอนกประสงค์โดยใช้ซอฟท์แวร์ทำงานนั้น จะทำให้มีความยืดหยุ่นในการใช้งานได้ดี เนื่องจากสามารถขยายการใช้งานได้ง่ายกว่าอุปกรณ์ทางฮาร์ดแวร์ โดยเพิ่มโปรแกรมการทำงานตามที่ต้องการได้ นอกจากนี้ยังไม่ต้องสิ้นเปลืองค่าใช้จ่ายในการบำรุงรักษาอุปกรณ์อีกด้วย

จากผลการทดลอง ซอฟท์แวร์ที่สร้างขึ้นสามารถทำงานตามที่ต้องการคือการส่งสัญญาณที่มอดูเลทแล้วในรูปแบบต่างๆ (สัญญาณดิจิตอล) ออกมาผ่านวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก สัญญาณอนาลอกที่ได้ออกมาจะผ่านวงจรกรองความที่ต่ำผ่าน ได้สัญญาณเอาท์พุทตามที่ต้องการ และทางด้านรับนำสัญญาณไปผ่านวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลได้สัญญาณดิจิตอลไปผ่านการดีโค้ดเดอร์ เพื่อวิเคราะห์สัญญาณข้อมูลออกมา

5.แนวทางการพัฒนา

จากผลการทดลองจะเห็นว่าสามารถทำงานให้ผลตรงตามหลักการที่ได้เสนอไว้ข้างต้น ดังนั้นเราสามารถนำไปประยุกต์ใช้งานด้านอื่นๆ อาทิเช่น

นำไปใช้งานด้านสื่อสารข้อมูลด้วยเครื่องคอมพิวเตอร์ โดยการเชื่อมโยงเข้ากับโครงข่ายโทรศัพท์ ซึ่งจะทำให้ผู้ที่ใช้ตัวมอดูเลทสัญญาณดิจิตอลเอนกประสงค์นี้สามารถสื่อสารข้อมูลกับเครื่องคอมพิวเตอร์ในระบบที่ใช้รูปแบบการมอดูเลทแบบอื่นได้ หรือ นำไปใช้ในการสื่อสารข้อมูลแบบไร้สายซึ่งจะต้องออกแบบเครื่องส่งและเครื่องรับสัญญาณวิทยุให้เหมาะสมกับการใช้งาน

6. บทแทรก

6.1 เงื่อนไขการมอดูเลทแบบต่างๆ

6.1.1 ฟรีเควนซีชิพคีอิ้ง (Frequency Shift Keying (FSK))

ฟรีเควนซีชิพคีอิ้งเป็นรูปแบบของการมอดูเลทเชิงมุมที่มีเอ็นเวลลอปคงที่(constant-envelope) ซึ่งคล้ายกับการมอดูเลทความถี่ (Frequency Modulation (FM)) ต่างกันตรงที่สัญญาณที่นำไปมอดูเลทเป็นไบนารีพัลส์สตรีม (binary pulse stream) ซึ่งเป็นค่าระหว่างดิสครีทโวลเตท (discrete voltage) 2 ระดับ ที่มีการเปลี่ยนแปลงของรูปคลื่น (wave form) ค่อนข้างจะต่อเนื่อง

สำหรับฟรีเควนซีชิพคีอิ้งแบบไบนารี่(Binary Frequency Shift Keying(BFSK)) สัญญาณทางด้านอินพุทที่ป้อนเข้ามาจะถูกนำไปวิเคราะห์ และทำตามเงื่อนไขที่กำหนด และนำสัญญาณที่ได้ไปเข้าควอดเดรเจอร์มอดูเลเตอร์ (Quadrature modulator) จะได้สัญญาณเอาท์พุทที่เป็นไปได้ 2 ค่า การวิเคราะห์สัญญาณและเงื่อนไขตามตารางที่ 6.1

ตาราง 6.1

ภาวะลอจิกของอินพุท	a(t)	b(t)
0	cos(Δωt)	sin(Δωt)
1	cos(Δωt)	-sin(Δωt)

6.1.2 ไบนารีเฟสชิพคีอิ้ง (Binary Phase Shift Keying(BPSK))

สัญญาณเอาท์พุทมีเฟสที่เป็นไปได้ 2 ค่า ซึ่งเฟสค่าหนึ่งแทนลอจิก 0 เฟสอีกค่าหนึ่งแทนลอจิก 1 โดยทางด้านอินพุทจะวิเคราะห์สัญญาณทีละบิท เมื่อสัญญาณดิจิตอลทางด้านอินพุทเปลี่ยนไป ทำให้เฟสเอาท์พุทเปลี่ยนไประหว่างเฟส 2 เฟสที่ต่างกัน 180° การวิเคราะห์สัญญาณและเงื่อนไขเป็นไปตามตาราง 6.2

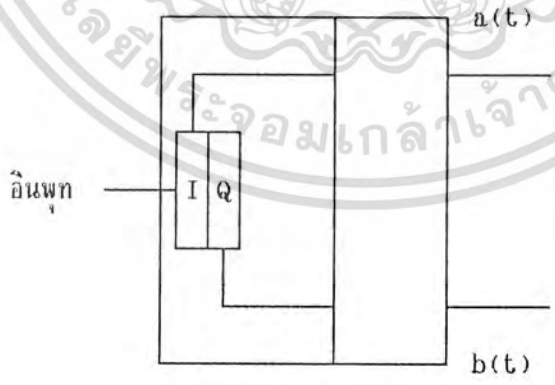
ตาราง 6.2

ภาวะลอจิกของอินพุท	a(t)	b(t)
0	-1 v	0
1	+1 v	0

6.1.3 ควอเดอร์นารีเฟสชิพคีอิ้ง (Quaternary Phase Shift Keying(QPSK))

ควอเดอร์นารีเฟสชิพคีอิ้งเป็นการเข้ารหัส M-ary[5] เมื่อ M = 4 แสดงว่ามีอินพุทบิท 2 บิท ข้อมูลอินพุทได้จากการรวมกลุ่มของบิท 2 บิท เรียกว่า ไดบิท (dibits) ดังนั้นทางด้านอินพุทจะวิเคราะห์สัญญาณทีละ 2 บิท จะได้เอาท์พุท 4 ค่าที่มีเฟสต่างกันแต่มีขนาดเท่ากัน ในแต่ละครั้งของการเปลี่ยนแปลงไดบิทอินพุทจะทำให้เกิดการเปลี่ยนแปลงเฟสทางด้านเอาท์พุท การวิเคราะห์สัญญาณและเงื่อนไขเป็นดังนี้

วงจรวิเคราะห์และปฏิบัติตามเงื่อนไข



รูปที่ 6.1

ตาราง 6.3

ภาวะลอจิกของบิทI	a(t)
0	-1 v
1	+1 v

ภาวะลอจิกของบิทQ	b(t)
0	-1 v
1	+1 v

จากตารางเห็นได้ว่า ภาวะลอจิกของบิท I และ Q จะเป็นตัวกำหนดค่า a(t) และ b(t) ตามลำดับ

6.1.4 8 เฟสชิพคีอิ้ง (8 Phase Shift Keying (8PSK))

8 เฟสชิพคีอิ้งเป็นการเข้ารหัส M-ary เมื่อ M = 8 แสดงว่ามีบิทอินพุท 3 บิท ซึ่งมูลอินพุทเป็นการรวมกลุ่มของบิทสามบิทเรียกไตรบิท (tribits) ดังนั้นทางด้านอินพุทจะทำการวิเคราะห์สัญญาณทีละ 3 บิท จะได้เอาท์พุท 8 ค่าที่มีเฟสต่างกันแต่มีขนาดเท่ากัน ในแต่ละครั้งของการเปลี่ยนแปลงทริบิทจะทำให้เกิดการเปลี่ยนแปลงเฟสทางด้านเอาท์พุท

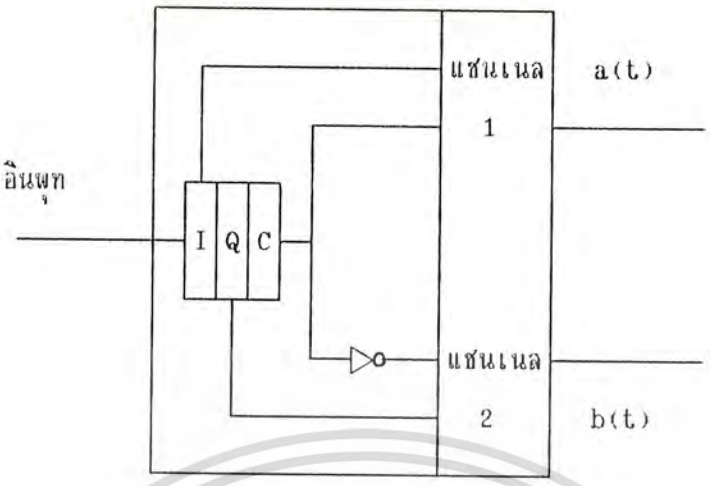
จากรูปที่ 6.2 เห็นได้ว่าสัญญาณไตรบิทที่รับเข้ามาจะถูกแยกออกเป็น 2 แชนเนล เพื่อนำไปวิเคราะห์ตามเงื่อนไขดังตาราง 6.4

ตารางที่ 6.4

ลอจิกของI	ลอจิกของC	a(t)
0	0	-x v
0	1	-y v
1	0	+x v
1	1	+y v

ลอจิกของQ	ลอจิกของ$\overline{C}$	b(t)
0	1	-y v
0	0	-x v
1	1	+y v
1	0	+x v

วงจรวิเคราะห์และปฏิบัติตามเงื่อนไข



รูปที่ 6.2

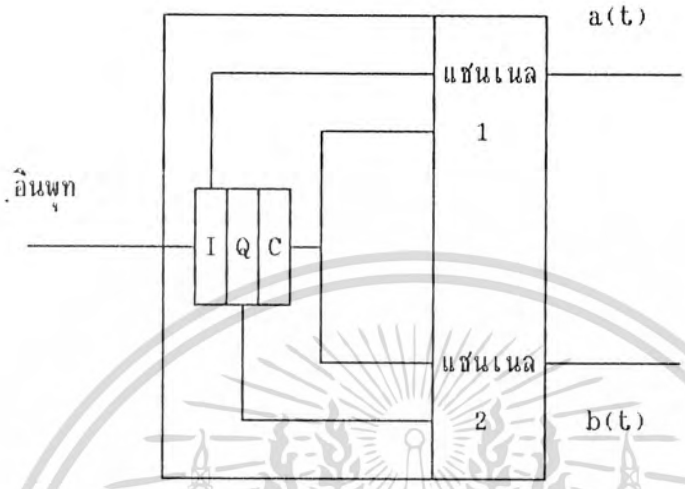
จากตารางที่ 6.4 จะเห็นได้ว่า ลอจิกของ I และ C เป็นเงื่อนไขของการกำหนดค่า a(t) ส่วนลอจิกของ Q และ $\overline{C}$ เป็นเงื่อนไขของการกำหนดค่า b(t) โดย I และ Q แสดงเครื่องหมายของ a(t) และ b(t) ตามลำดับว่าเป็นค่าบวกหรือลบ Q แสดงขนาดของ a(t) และ b(t) ว่ามีค่าเป็น x volt หรือ y volt เงื่อนไขเหล่านี้ทำให้เกิดค่าทางด้านเอาท์พุท 8 ค่าที่มีเฟสต่างกันแต่มีขนาดเท่ากัน

6.1.5 8 ควอดเดรเจอร์แอมปลิจูดมอดูเลชั่น (Eight Quadrature Amplitude Modulation (8QAM))

8 ควอดเดรเจอร์แอมปลิจูดมอดูเลชั่นเป็นการเข้ารหัส M-ary เมื่อ M = 8 เหมือนกับ 8 เฟสชิพคีอิ้ง ต่างกันตรงเงื่อนไขที่ใช้ในการวิเคราะห์สัญญาณและสัญญานเอาท์พุท ซึ่งเอาท์พุทเป็นสัญญาณที่มีขนาดไม่คงที่

จากรูปที่ 6.3 จะเห็นได้ว่าสัญญาณไตรบิทที่รับเข้ามาจะถูกแยกออกเป็น 2 แชนเนล เพื่อนำไปวิเคราะห์ตามเงื่อนไขดังตารางที่ 6.5 ซึ่งแสดงให้เห็นว่าลอจิกของ I และ C เป็นเงื่อนไขในการกำหนดค่าของ a(t) ส่วนลอจิกของ Q และ C เป็นเงื่อนไขในการกำหนดค่าของ b(t) ลอจิกของ I และ Q ใช้กำหนด a(t) และ b(t) ตามลำดับว่ามีค่าเป็นบวกหรือลบ ลอจิก C ใช้กำหนด a(t) และ b(t) ว่ามีค่าเป็น x หรือ y volt

วงจรวิเคราะห์และปฏิบัติตามเงื่อนไข



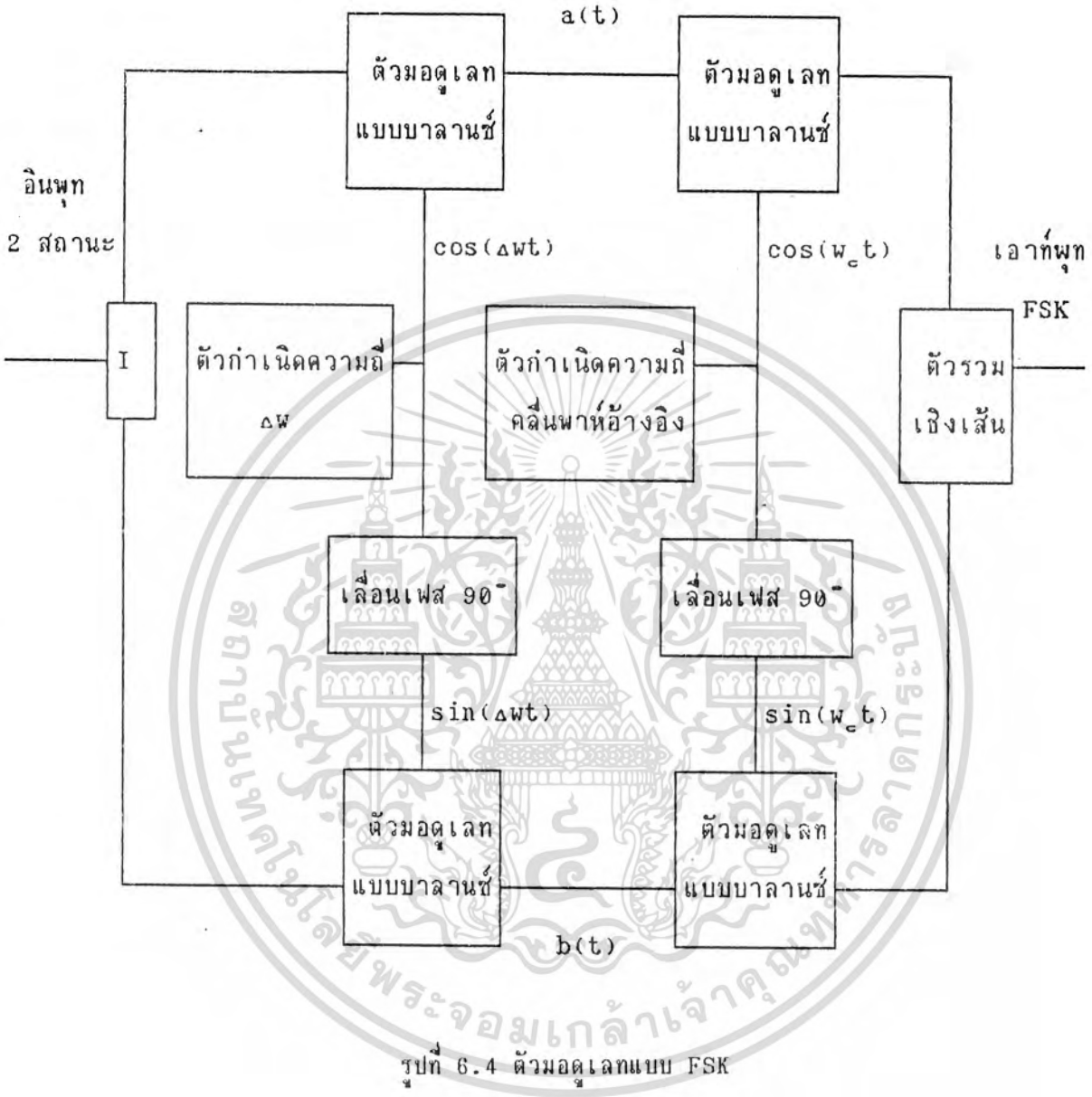
รูปที่ 6.3

ตารางที่ 6.5

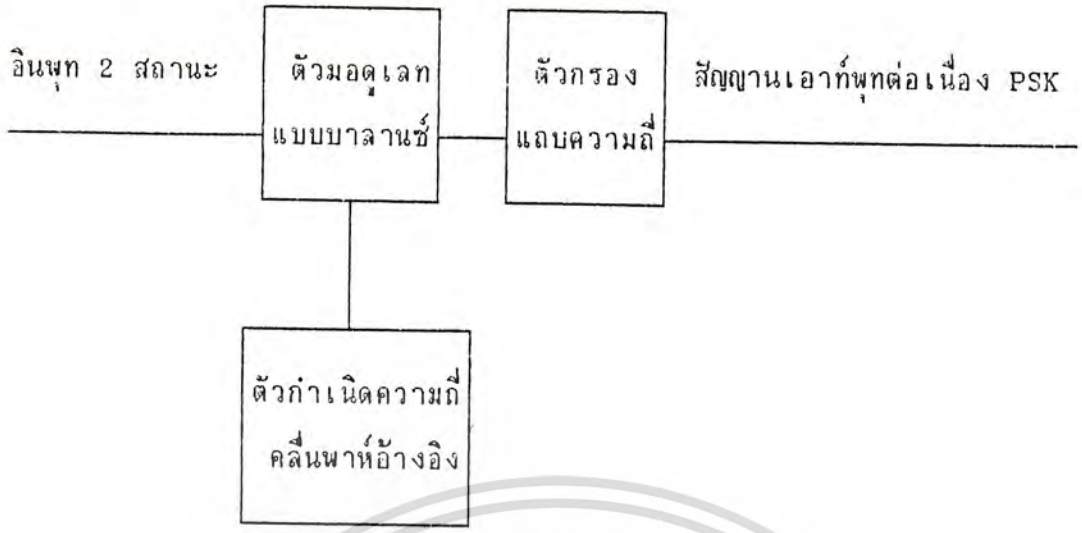
ลอจิกของI	ลอจิกของC	a(t)
0	0	-x v
0	1	-y v
1	0	+x v
1	1	+y v

ลอจิกของQ	ลอจิกของC	b(t)
0	0	-x v
0	1	-y v
1	0	+x v
1	1	+y v

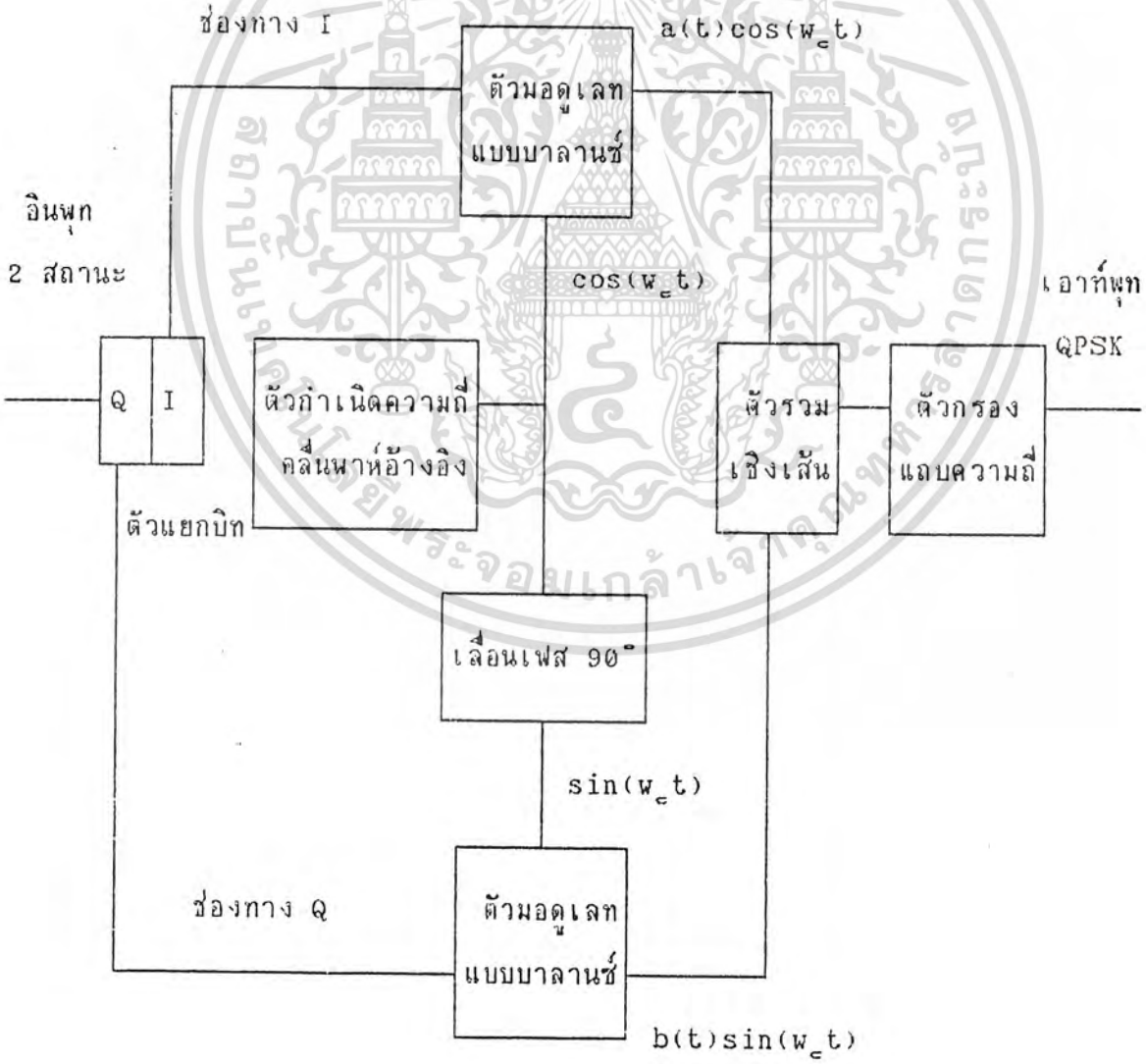
6.2 แผนผังแสดงการมอดูเลทแบบดิจิตอล



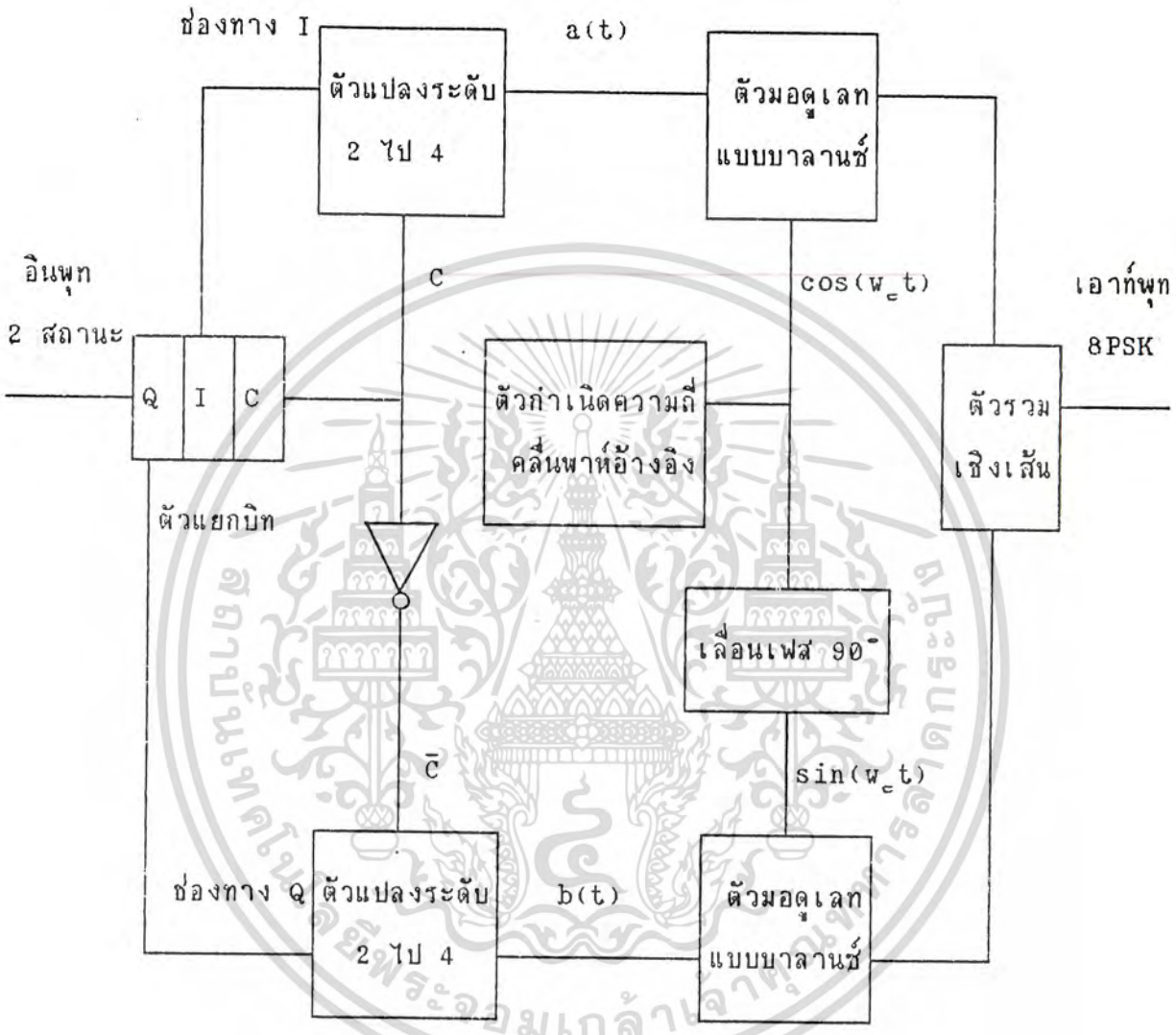
รูปที่ 6.4 ตัวมอดูเลทแบบ FSK



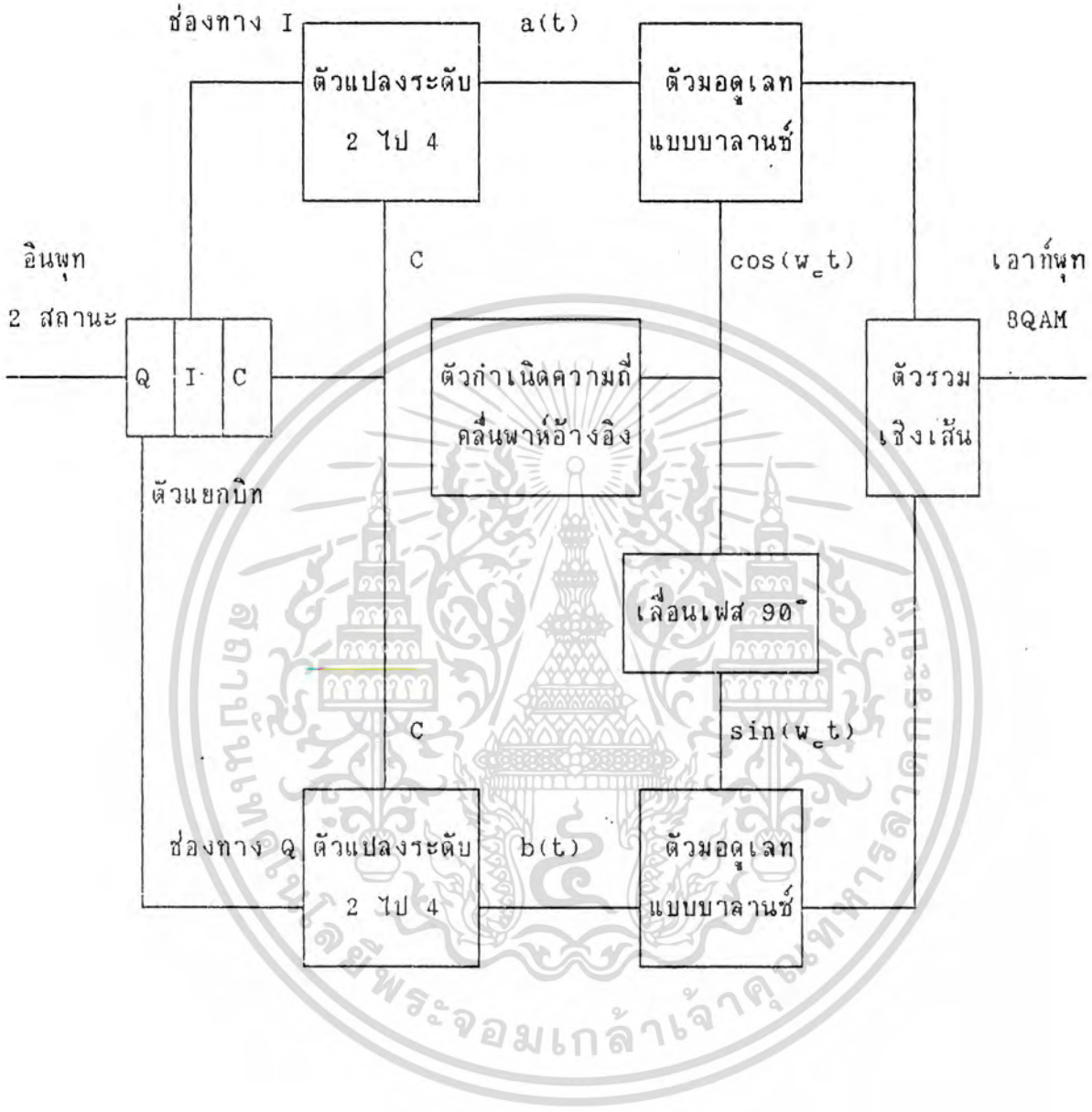
รูปที่ 6.5 ตัวมอดูเลทแบบ BPSK



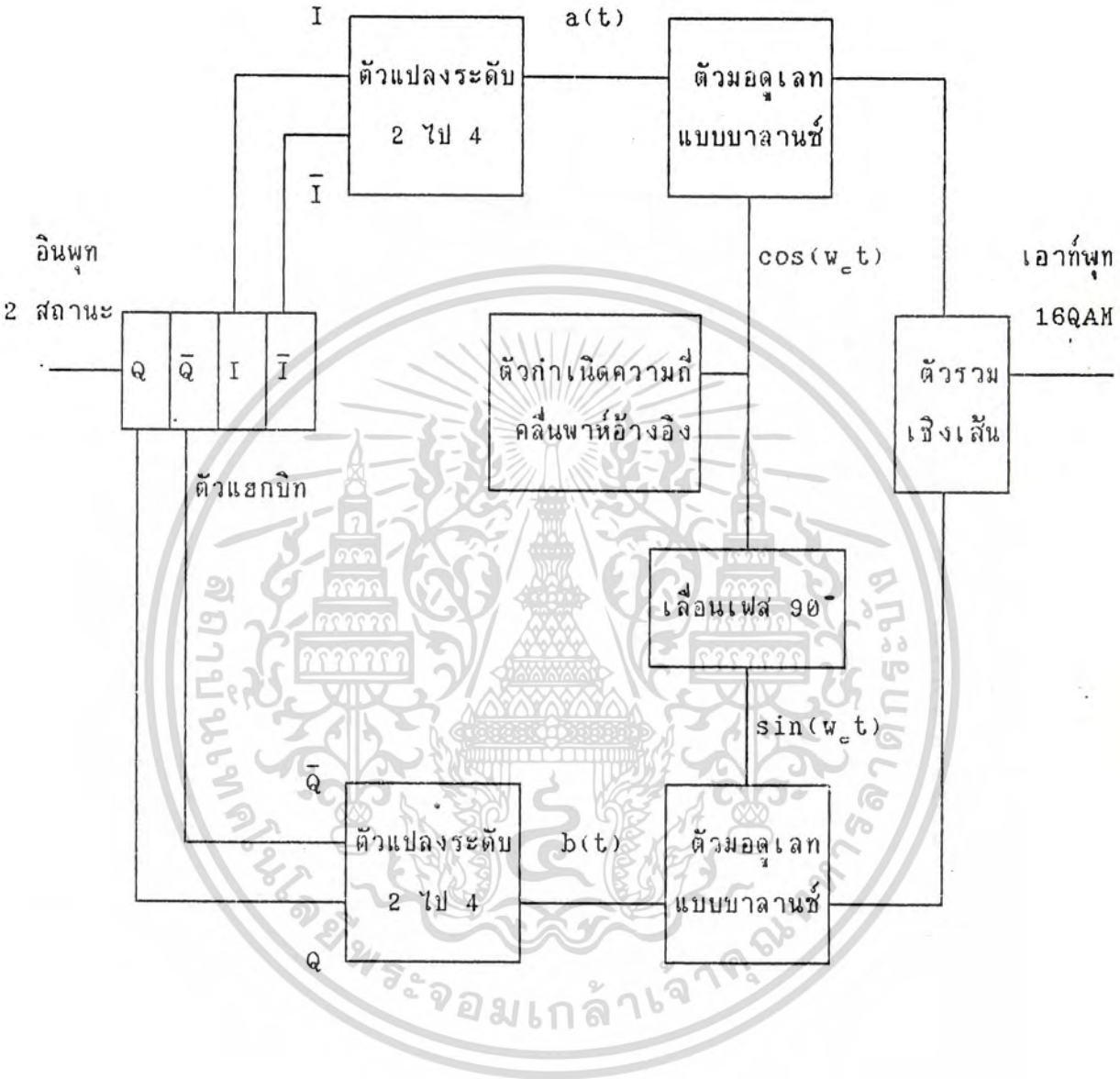
รูปที่ 6.6 ตัวมอดูเลทแบบ QPSK



รูปที่ 6.7 ตัมอดูเลทแบบ 8PSK



รูปที่ 6.8 ตัวมอดูเลทแบบ 8QAM



รูปที่ 6.9 ตัวมอดูเลทแบบ 16QAM

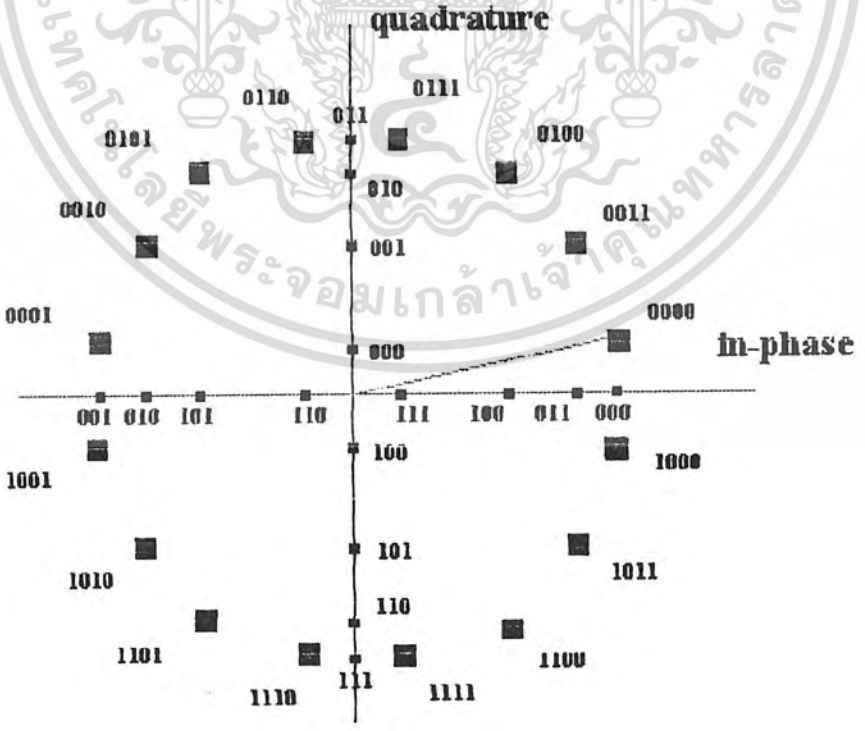
6.3 ค่าของ α และ β ของการมอดูเลท

THREE-BIT STATE	α_i	β_i
000	SIN 0	COS 0
001	SIN 3θ	- COS θ
010	COS 3θ	- COS 3θ
011	COS θ	COS 3θ
100	- SIN θ	SIN 3θ
101	- SIN 3θ	- SIN 3θ
110	- COS 3θ	- SIN θ
111	- COS θ	SIN θ

30 อาจเป็น n0 < 45 ; n เป็นจำนวนเต็มมากกว่า 1

16 PSK

ตาราง 6.6



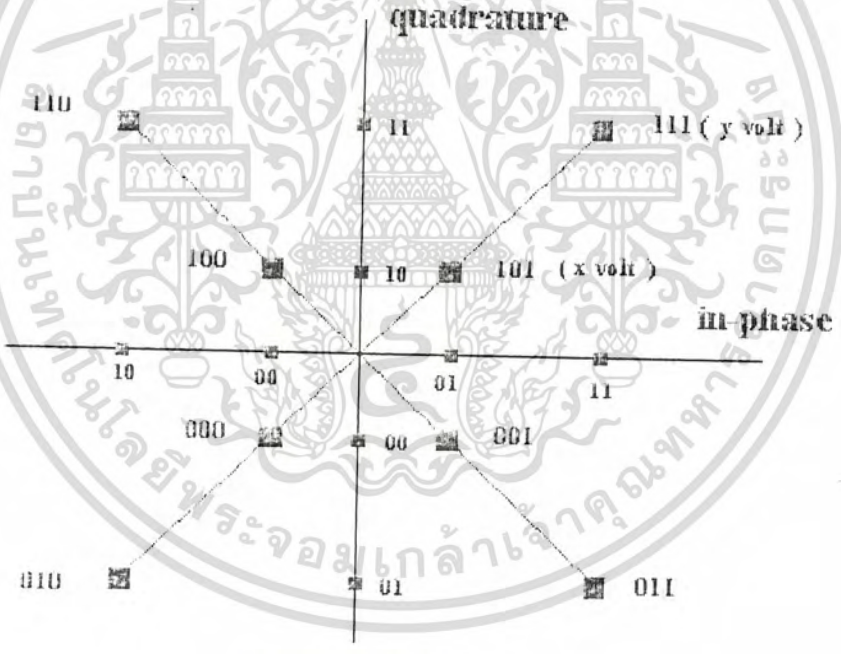
16PSK

รูปที่ 6.10

TWO-BIT STATE	α_i	β_i
00	-X COS 45°	-X SIN 45°
01	-Y COS 45°	X SIN 45°
10	X SIN 45°	-Y COS 45°
11	Y SIN 45°	Y COS 45°

8 QAM

ตาราง 6.7



8QAM

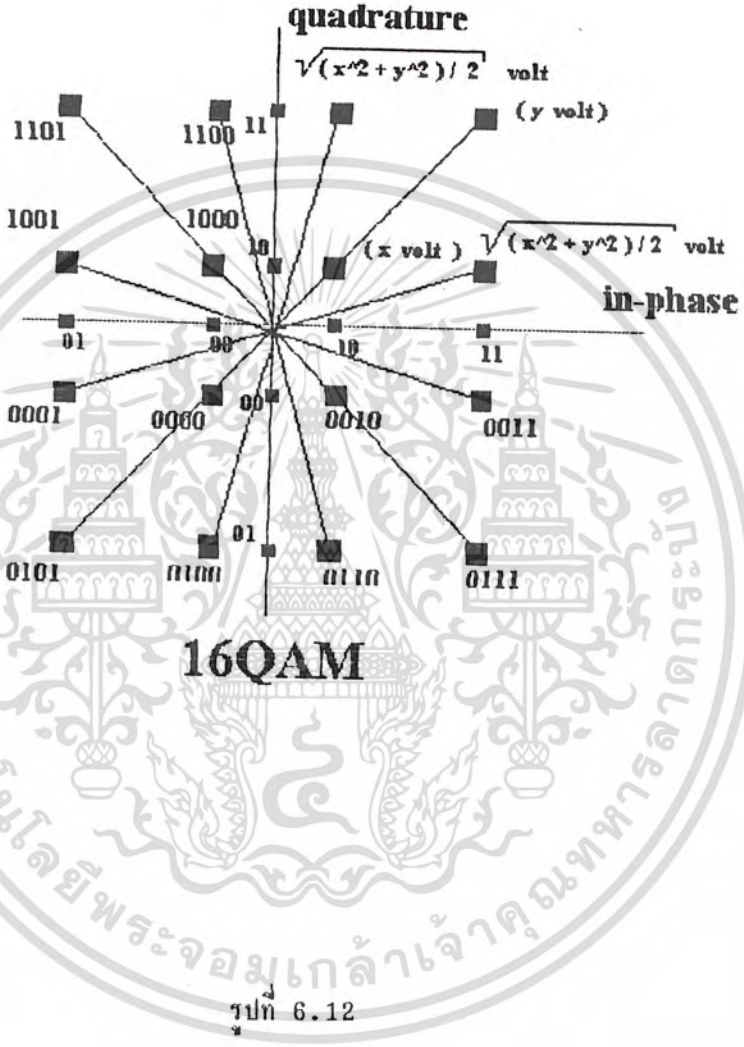
รูปที่ 6.11

$Q\bar{Q}I\bar{I}$	α_i	β_i
0000	$-X\cos 45^\circ$	$-X\sin 45^\circ$
0001	$-\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$	$-\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$
0010	$-X\cos 45^\circ$	$X\sin 45^\circ$
0011	$-\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$	$\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$
0100	$-\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$	$-\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$
0101	$-Y\cos 45^\circ$	$-Y\sin 45^\circ$
0110	$\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$	$\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$
0111	$-Y\cos 45^\circ$	$Y\sin 45^\circ$
1000	$X\cos 45^\circ$	$-X\sin 45^\circ$
1001	$\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$	$-\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$
1010	$X\cos 45^\circ$	$X\sin 45^\circ$
1011	$\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$	$\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$
1100	$\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$	$-\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$
1101	$Y\cos 45^\circ$	$-Y\sin 45^\circ$
1110	$\sqrt{\frac{(x^2+y^2)}{2}}\cos\theta$	$\sqrt{\frac{(x^2+y^2)}{2}}\sin\theta$
1111	$Y\cos 45^\circ$	$Y\sin 45^\circ$

* $\tan\theta = x/y \quad ; \quad x < y$

16 QAM

ตาราง 6.8



รูปที่ 6.12

6. โปรแกรมภาษา C

```
/*****************************************************************
* MODULATE.C : show graph of c(t) = a(t)sin(wt) + b(t)*cos(wt) *
*****************************************************************/
#include <stdlib.h>
#include <process.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <graphics.h>
#include "popup.c"

#define  PI  3.141592654
#define  ESC 0x1B

#define  QPSKsin  0.707  /* angle is 45 degree */
#define  QPSKcos  0.707  /* angle is 45 degree */

#define  EQAMx      (0.541*0.707)
#define  EQAMxsin   (0.541*0.707)
#define  EQAMy      (1.307*0.707)
#define  EQAMysin   (1.307*0.707)

#define  EPSKx 0.22
#define  EPSKy 0.821

#define  SPSKw 0.195
#define  SPSKx 0.5556
#define  SPSKy 0.8314
#define  SPSKz 0.9807

#define  SQAMxcos  (0.311*0.707)
#define  SQAMxsin  (0.311*0.707)
#define  SQAMycos  (1.161*0.707)
#define  SQAMysin  (1.161*0.707)
#define  SQAMxycos (0.850*0.9659)
#define  SQAMxysin (0.850*0.2588)

void     set_sym(char type) ;
void     shift(double new_co,char channel) ;
void     update(int i) ;
void     setgraph() ;
void     drawgrid() ;
void     display(double value,char channel) ;
int      mod(double pam_a,double pam_b,int j,int L,char channel) ;
int      quantize(double output) ;
int      popup_menu(char *menu_name[],char *hot_key,int num ) ;
double   coef(int k,char channel) ;
double   pulse_shape(int j,int k) ;
double   binary(char type,int i) ;
double   quad(int i,char channel) ;
double   octa(char type,int i,char channel) ;
double   sixteen(char type,int i,char channel) ;

char     mode, type, resolve, graph_type, observe, colorbk;
int      j, k, L, M, maxx, maxy, array[8], limitx ,
         step = 25, step_width = 10 ;
float    r ;
```

```
double   *shift_register_a,*shift_register_b ,factor = 1;
unsigned long n = 0 ;
union    { char byte  ;
           struct { unsigned int single_bit7: 1 ;
                    unsigned int single_bit6: 1 ;
                    unsigned int single_bit5: 1 ;
                    unsigned int single_bit4: 1 ;
                    unsigned int single_bit3: 1 ;
                    unsigned int single_bit2: 1 ;
                    unsigned int single_bit1: 1 ;
                    unsigned int single_bit0: 1 ;
                  } binary ;
           struct { unsigned int dibit3: 2 ;
                    unsigned int dibit2: 2 ;
                    unsigned int dibit1: 2 ;
                    unsigned int dibit0: 2 ;
                  } quad ;
           struct { unsigned int tribit2: 2 ;
                    unsigned int tribit1: 3 ;
                    unsigned int tribit0: 3 ;
                  } octa ;
           struct { unsigned int quadbit1: 4 ;
                    unsigned int quadbit0: 4 ;
                  } sixteen ;
         } ch ;
char     *menu_mode[]       = { " Ook " ,
                               " Bpsk " ,
                               " Qpsk " ,
                               " 8Psk " ,
                               " 16pSk " ,
                               " 8Qam " ,
                               " 16qaM "
                             } ,
         *menu_resolve[]    = { "PULSE_SHAPE" ,
                               "PULSE*COEF" ,
                               "PAM_OUTPUT"
                             } ,
         *menu_observe[]    = { "BASEBAND " ,
                               "EYE_DIAGRAM " ,
                               "CONSTELLATION "
                             } ,
         *menu_graph_type[] = { "STAIR_CASE  " ,
                               "DOTTED LINE" ,
                               "SMOOTHED   "
                             } ,
         *menu_source[]     = { "\"DATA.OUT\"",
                                " OPEN NEW FILE " ,
                                " RANDOM DATA "
                             } ,
         *menu_colorbk[]    = { " WHITE " ,
                                " BLACK "
                             } ;
/*****************************************************************
* function main :
*****************************************************************/
void main()
{
  int    count,i,j,temp,loop,gmode,errorcode,gdriver = DETECT ;
  double p = 0,result_a = 0, result_b = 0, pam_output_a = 0 ,
         pam_output_b = 0 ,co_a = 0,co_b = 0;
```

```
    FILE   *fp ;   /* data file to mod */
    FILE   *fpointer; /* data file alreay mod */
    char   temp_char,filename[30], *fname, source, channel = 'a';

    clrscr() ;
    sound(500) ;   delay(250) ;  nosound() ;
    sound(1000) ;  delay(350) ;  nosound() ;
    sound(1500) ;  delay(350) ;  nosound() ;
    r = 1.0 ;
    M = 9 ;
    L=24;
    step_width = 2 ;
    step += step_width ;
    printf("\n\t SELECT BACKGROUND COLOR = ") ;
    colorbk = popup_menu(menu_colorbk,"wb",2) ;

    resolve = '2' ;
    if( resolve != '0' )                    /* not pulse_shape */
    {
      printf("\n\t DIGITAL MODULATION MODE = ") ;
      mode = popup_menu(menu_mode,"obqpsqm",7) ;
      observe = '0'; /***baseband ***/
      fname = searchpath("ref.tx") ;
      if(fname == NULL)
      {
        printf("\n\t CAN'T FIND FILE \"DATA.OUT\"") ;
        exit(0) ;
      }
    }
    if ((resolve != '1') &&(observe != '2'))
       graph_type = '2';
    else                                    /* PULSE * COEF */
       graph_type = '1' ;                   /* DOTTED LINE */

  /************** initial graphic mode*************************/
    initgraph(&gdriver, &gmode, "") ; // request auto detection
    errorcode = graphresult() ;
    if (errorcode != grOk)                  /* an error occurred */
    {
      printf("Graphics error: %s\n",grapherrormsg(errorcode));
      printf("Press any key to halt:") ;
      getch() ;
      exit(1) ;                           /* return with error code */
    }
    maxx   = getmaxx() ;
    maxy   = getmaxy() ;
    limitx = L * floor( floor( (maxx-48) / step_width )/ L) ;

    if (colorbk == '0')
       setbkcolor( 15 ) ;                   /* white background */
    else
       setbkcolor( 0 ) ;                    /* black background */
    setgraph() ;

    if (source != '2')
       fp = fopen(fname,"rb") ; /* open file to read to send */

  // allocate memory 2M block as a shift register 2M stage
     shift_register_a=(double *)calloc((2 * M),sizeof(double));
```

```
    if (observe != '1')
     shift_register_b=(double *)calloc((2 * M),sizeof(double));

    switch(mode)
    {
       case '0' : { loop = 8 ; type = 'b' ; } break ;
       case '1' : { loop = 8 ; type = 'b' ; } break ;
       case '2' : { loop = 4 ; type = 'q' ; } break ;
       case '3' : { loop = 3 ; type = 'o' ; } break ;
       case '4' : { loop = 2 ; type = 's' ; } break ;
       case '5' : { loop = 3 ; type = 'o' ; } break ;
       case '6' : { loop = 2 ; type = 's' ; } break ;
       default  : printf("\n\t*** UNKNOWN MODE SELECTED ***") ;
                  exit(1);
                  }
      fpointer = fopen("c.mod","wb") ;

    while( (temp=(source=='2') ? random(512) : getc(fp))!=EOF )
    {
      ch.byte = temp ;
      set_sym(type) ;// call funtion to set new character

      for (i = 0 ; i < loop ; i++,n++)
      {
        update(i) ;
        for( j=1;j < L+1;j++,pam_output_a=0,pam_output_b=0 )
        {
         for ( k = -M ; k < M ; k++ )
         {
          p = pulse_shape(j,k) ;   /* find pulse value */
          co_a = coef(k,'a') ;     /* find coef of channel a*/
          result_a = p * co_a;
          pam_output_a += result_a ;

          if ((observe != '1') && (resolve == '2'))
          {
           co_b = coef(k,'b') ;   /* find coef of channel b*/
           result_b = p * co_b ;
           pam_output_b += result_b ;
          }
         }
         temp = mod(pam_output_a,pam_output_b,j,L,'b') ;
         display(temp,'b') ; /* c(t) in ch b dash line */
         temp = temp + 128;
         putc(temp,fpointer) ;
        }
      }
    }  /* end of file or user terminate */
    fclose(fp) ;
    fclose(fpointer) ;
    free (shift_register_a) ;
    free (shift_register_b) ;
    moveto(maxx-200,maxy-20) ;
    setcolor(colorbk == '1' ? 0 : 8) ;
    outtext("PRESS ESC TO EXIT...") ;
    while (getch() != ESC)
            getch() ;
    clearviewport() ;
    closegraph() ;
} /* end main and exit program */
```

```
/*************************************************************
function popup_menu : put menu onto screen and get value *
require             : menu name,hot key and number  menu *
return              : character of selected item          *
*************************************************************/
int popup_menu(char *menu_name[],char *hot_key,int num )
{
  int temp, cursor_x, cursor_y ;

  /* call funtion popup in file popup.c */

  cursor_x = wherex() ;
  cursor_y = wherey() ;
  temp  = popup(menu_name,hot_key,num,cursor_y-1,cursor_x,1) ;
  if (temp == -1)          /* user press escape button */
       {
            gotoxy(cursor_x,cursor_y) ;
        printf("\n\t*** PROGRAM TERMINATED BY USER ***") ;
            exit(0) ;
       }
  gotoxy( cursor_x,cursor_y ) ;     /* user selected one item */
  /* put item selected on screen at (cursor_x,cursor_y) */
  puts( menu_name[temp] ) ;
  return temp + 48 ;        /* convert integer to character */
}

/*************************************************************
* function set_sym : assign value in 1 charactor to array[] *
* require          : (char)type; (extern)array[],(struct)ch *
* return           : -                                      *
*************************************************************/
void set_sym(char type)
{
  switch(type)
   {
    case 'b' : { array[0] = ch.binary.single_bit0 ;
                 array[1] = ch.binary.single_bit1 ;
                 array[2] = ch.binary.single_bit2 ;
                 array[3] = ch.binary.single_bit3 ;
                 array[4] = ch.binary.single_bit4 ;
                 array[5] = ch.binary.single_bit5 ;
                 array[6] = ch.binary.single_bit6 ;
                 array[7] = ch.binary.single_bit7 ;
               } break ;
    case 'q' : { array[0] = ch.quad.dibit0 ;
                 array[1] = ch.quad.dibit1 ;
                 array[2] = ch.quad.dibit2 ;
                 array[3] = ch.quad.dibit3 ;
               } break ;
    case 'o' : { array[0] = ch.octa.tribit0 ;
                 array[1] = ch.octa.tribit1 ;
                 array[2] = ch.octa.tribit2 ;
               } break ;
    case 's' : { array[0] = ch.sixteen.quadbit0 ;
                 array[1] = ch.sixteen.quadbit1 ;
               } break ;
    default  : printf("\n default mode in function set_sym");
                          exit(1) ;
   }
}
```

```
/*********************************************************
* function update : call shift() and send parameter      *
* require         : (int) i, (ext)(char) mode            *
* return          : -                                    *
*********************************************************/
void update(int i)
{
  switch(mode)
  {
    case '0' : shift( binary('0',i),'a' ) ;
               break ;
    case '1' : shift( binary('1',i),'a' ) ;
               break ;
    case '2' : { shift( quad(i,'a'),'a' ) ;
                 if (observe != '1')
                   shift( quad(i,'b'),'b' ) ;
               }
               break ;
    case '3' : { shift( octa('3',i,'a'),'a' ) ;
                 if (observe != '1')
                   shift( octa('3',i,'b'),'b' ) ;
               }
               break ;
    case '4' : { shift( sixteen('4',i,'a'),'a' ) ;
                 if (observe != '1')
                   shift( sixteen('4',i,'b'),'b' ) ;
               }
               break ;
    case '5' : { shift( octa('6',i,'a'),'a' ) ;
                 if (observe != '1')
                   shift( octa('6',i,'b'),'b' ) ;
               }
               break ;
    case '6' : { shift( sixteen('7',i,'a'),'a' ) ;
                 if (observe != '1')
                   shift( sixteen('7',i,'b'),'b' ) ;
               }
               break ;
    default  : printf("\t default mode in function update") ;
               exit (1) ;
    }
}

/*********************************************************
* function shift : shift new coefficient into shift register *
* require        : (double)new_coefficient,              *
                   (ext)(double)*(shift_register_a),     *
                   *(shift_register_b)                   *
* return         : -                                     *
*********************************************************/
void shift(double new_coefficient,char channel)
{
  int count ;

  for (count = 0 ; count < (2 * M)-1 ; count++)
  {
    if(channel == 'a')
      *(shift_register_a+count)=*(shift_register_a+count+1) ;
    else
      *(shift_register_b+count)=*(shift_register_b+count+1) ;
```

```
  }

  if (channel == 'a')
        *(shift_register_a + count) = new_coefficient ;
  else
        *(shift_register_b + count) = new_coefficient ;

} /* return to funtion update */

/*******************************************************
* function pulse_shape : calculate sampling of pulse *
* require              : (ext)(float)r,(int)j,(int)k *
* return               : (double)p                   *
*******************************************************/
double pulse_shape(int j,int k)
{
  register double temp,p ;

  if ((k != -1) || (j != L))
  {
   temp = (k + (double)j/(double)L)   ;
   if ((r == 1) && (fabs(temp) == 0.5))
       return 0.5 ;
   p = ((sin(PI * temp)) * (cos(r*PI * temp) ))/(PI * temp) ;
   temp = 1-(4*r*r*temp*temp) ;
   if (temp == 0)
      return 0 ;
   return p/temp ;
  }
  else
     return 1 ;
}

/*************************************************************
* function coef : find index  from M and k                   *
*                                                            *
* require       : (int)k; (ext)(double)*shift_register_a,    *
                  (int) M ,*shift_register_b,                *
* return        : (double)reg_cont in stage pointed by index *
*                                                            *
*************************************************************/
double coef(int k,char channel)
{
  double reg_cont ;

  if (channel == 'a')
        reg_cont = *(shift_register_a + M - k - 1) ;
  else
        reg_cont = *(shift_register_b + M - k - 1) ;

  return reg_cont ;
}  /* return to funtion sum */

/*************************************************************
* function binary : store coefficient in mode OOK and BPSK   *
* require         : (char)type;(int)i; (ext)array[],channel  *
* return          : (double)table[index]                     *
*************************************************************/
double binary(char type,int i)
{
```

```
  switch(type)
  {
/*OOK*/   case '0' : if (array[i] == 0) return  0 ;
                     else              return  1 ;
/*BPSK*/  case '1' : if (array[i] == 0) return  1 ;
                     else              return -1 ;
          default  : printf("\ndefault type in binary()");
                     exit(1) ;
  }
}

/*************************************************************
* function quad : store value of coefficient in mode QPSK *
* require       : (int)i; (ext)array[],channel              *
* return        : (double)table[index]                      *
*************************************************************/
double quad(int i,char channel)
{
 int    index ;
 double table[8] = { -QPSKsin,-QPSKcos,    /* 00 */
                     -QPSKsin, QPSKcos,    /* 01 */
                      QPSKsin,-QPSKcos,    /* 10 */
                      QPSKsin, QPSKcos,    /* 11 */
                   } ;

 index = (2 * array[i]) + ( (channel == 'a') ? 0 : 1 ) ;
 return table[index] ;
}

/*************************************************************
* function octa : store coefficient in mode 16QPSK and 16QAM *
* require       : (char)type;(int)i; (ext)array[],channel    *
* return        : (double)table[index]                       *
*************************************************************/
double octa(char type,int i,char channel)
{
 int index ;
 double table[32]={-EPSKx,-EPSKy,-EQAMx,-EQAMx,    /* 000  */
                    -EPSKx, EPSKy,-EQAMx, EQAMx,    /* 001  */
                     EPSKx,-EPSKy,-EQAMy,-EQAMy,    /* 010  */
                     EPSKx, EPSKy,-EQAMy, EQAMy,    /* 011  */
                    -EPSKy,-EPSKx, EQAMx,-EQAMx,    /* 100  */
                    -EPSKy, EPSKx, EQAMx, EQAMx,    /* 101  */
                     EPSKy,-EPSKx, EQAMy,-EQAMy,    /* 110  */
                     EPSKy, EPSKx, EQAMy, EQAMy     /* 111  */
                   };

 index=( 4*array[i])+((type=='3')?0:2)+((channel=='a')?0:1 ) ;
 return table[index] ;           /* type 3 = 8psk */
}

/*************************************************************
function sixteen : store coefficient in mode 16QPSK and 16QAM *
require          : (char)type;(int)i; (ext)array[],channel    *
return           : (double)table[index]                       *
*************************************************************/
double sixteen(char type,int i,char channel)
{
 int index ;
 double table[64]={SPSKw, SPSKz,-SQAMxcos ,-SQAMxsin,  // 0000
```

```
                    SPSKw,-SPSKz,-SQAMxysin,-SQAMxycos, // 0001
                    SPSKx, SPSKy,-SQAMxcos , SQAMxsin,  // 0010
                    SPSKx,-SPSKy,-SQAMxysin, SQAMxycos, // 0011
                    SPSKy, SPSKx,-SQAMxycos,-SQAMxysin, // 0100
                    SPSKy,-SPSKx,-SQAMycos ,-SQAMysin,  // 0101
                    SPSKz, SPSKw,-SQAMxycos, SQAMxysin, // 0110
                    SPSKz,-SPSKw,-SQAMycos , SQAMysin,  // 0111
                   -SPSKw, SPSKz, SQAMxcos ,-SQAMxsin,  // 1000
                   -SPSKw,-SPSKz, SQAMxysin,-SQAMxycos, // 1001
                   -SPSKx, SPSKy, SQAMxcos , SQAMxsin,  // 1010
                   -SPSKx,-SPSKy, SQAMxysin, SQAMxycos, // 1011
                   -SPSKy, SPSKx, SQAMxycos,-SQAMxysin, // 1100
                   -SPSKy,-SPSKx, SQAMycos ,-SQAMysin,  // 1101
                   -SPSKz, SPSKw, SQAMxycos, SQAMxysin, // 1110
                   -SPSKz,-SPSKw, SQAMycos , SQAMysin   // 1111
                    } ;

 index=( 4*array[i])+((type=='4')?0:2) + ((channel=='a')?0:1 );
 return table[index] ;      /* type 4 = 16psk */
}

/*************************************************************
* function setgraph : set graphic screen, axises, border     *
* require           : (extern)resolve                        *
* return            : -                                      *
*************************************************************/
void setgraph()
{
  char *temp_string ='\0' ;
  int  dec, sign, count ;

  drawgrid() ;          /* draw grid */
  setcolor(9) ;         /* blue text */
  moveto (25,15) ;

  if (resolve == '2')               /* PAM_OUTPUT */
       {
            outtext( menu_observe[(int)observe - 48]) ;
            outtext(" of ") ;
       }

  outtext( menu_resolve[(int)resolve - 48] ) ;

  if (resolve != '0')
       {
            outtext( ",MODE " ) ;
        outtext( menu_mode[(int)mode - 48] );
       }

  outtext( ",UNIFORM QUANTIZE 256 LEVELS") ;
  moveto( 25, 30 ) ;
  outtext( "ROLL-OFF FACTOR (r) = " ) ;
  temp_string = fcvt( r, 2, &dec, &sign ) ;

  if ( dec<=0 )
       outtext( "0." ) ;

  if (r == 0.0)
    outtext("0.00") ;
  else
```

```
        if (r == 1)
           outtext("1") ;
        else
           {
             for( count = dec ; count <0 ; count++ )
                {
                 outtext( "0" ) ;
                 outtext( temp_string ) ;
                }
           }
     itoa( M, temp_string, 10 ) ;
     outtext( " ,ISI LENGTH (M) = " ) ;
     outtext( temp_string ) ;
     outtext( "  ," ) ;
     itoa( L, temp_string, 10 ) ;
     outtext( temp_string ) ;
     outtext( " SAMPLES PER SYMBOL" ) ;

     if (observe != '2')              /* not constellation */
           {
           . outtextxy( 15, (maxy/2),observe == '2' ? "" : "0" ) ;
             outtextxy( maxx - 50, (maxy/2) - 12, "Time") ;
             outtextxy( 28, 45, "Amplitude" ) ;
           }

     if (observe == '0')        /* baseband signal */
      if (graph_type == '1')          /* DOTTED LINE */
        {
          outtextxy( 10, maxy - 30, "* : channel a" ) ;
          outtextxy( 10, maxy - 20, "+ : channel b" ) ;
        }
      else
        {
          outtextxy( 10, maxy - 25, "solid line : channel a" ) ;
          outtextxy( 10, maxy - 15, " dash line : channel b" ) ;
        }


     setlinestyle(0,1,1) ;
     setcolor (3) ;                   /* curve */
     moveto( 25, maxy/2 ) ;           /* origin */
}

/*************************************************************
* function drawgrid : draw grids                             *
* require           : (ext)maxx,maxy,step_width,observe,L    *
* return            : -                                      *
*************************************************************/
void drawgrid()
{
  int countx,county,temp ;

  setcolor(3) ;                                 /*  grid */
  setlinestyle(1,1,1);       /* dotted-line,1 pixel width */

  for (countx = 25 ; countx < maxx ; countx += 8*L)
       line(countx,40,countx,maxy-40) ;/* vertical grid */

  for (county = 50 ; county < maxy ; county += 50)
  {                                     /* horizontal grid */
```

```
    temp = maxx-25-( 3*L*floor( (maxx-25) / (3*L) ) ) ;
    line(25,( (maxy/2)-county), maxx-temp,((maxy/2)-county));
    line(25,( (maxy/2)+county), maxx-temp,((maxy/2)+county));
   }

   setlinestyle(3,1,1) ;                          /* dash line */
   line (25,( (maxy/2)-128), maxx-temp,((maxy/2)-128) ) ;
   line (25,( (maxy/2)+128), maxx-temp,((maxy/2)+128) ) ;
   setlinestyle(0,1,2) ;
   setcolor(9) ;

   if (observe == '2')                        /* constellation */
        line(325,40,325,maxy-40) ;
   else
       line(25,40,25,maxy-40) ;                  /* vertical axis */

   line(25,(maxy/2),maxx-13,(maxy/2)) ;    /* horizontal axis */
}

/*************************************************************
* function display : display output to screen in graph mode *
* require          : (double)value;(ext)(int) step,         *
*                    step_width,limitx,j,n                   *
*                    resolve,maxx,maxy,graph_type,channal    *
* return           : this funtion not return value          *
*************************************************************/
void display(double value,char channel)
{
  static int first = 0 , countline = 0;
  static int ypos = 239, aposx = 25, aposy = 239,
             bposx = 25,bposy = 239,x,y ;
  static double count_step = 0, count = 0 ;
  int    quantized_value = 0 ,tempcolor ;

  quantized_value = value ;

 if (observe != '2')     /* not constellation*/
 {
  if (count_step >= limitx)
  {
   if ((resolve == '1')||(observe == '0'))
   {
     setcolor(9) ;          /* blue text */
     outtextxy(maxx-250,maxy-25,"PRESS ANY KEY TO CONTINUE") ;
     clearviewport() ;
     setgraph() ;
     aposx = 25 ;
     bposx = 25 ;
   }
   else                              /* EYE DIAGRAM */
      {
       if (first < 3)
       {
         clearviewport() ;
         first++ ;
         setgraph() ;
       }
      }
  moveto(25,ypos) ;                  /* origin */
  step = 25 ;                        /* re-initial */
```

```
 step += step_width ;          /* first step */
 count_step = 0;
}                               /* end check limit */

if (resolve == '1')      /* PULSE * COEF */
{
  outtextxy(step,(maxy / 2)-quantized_value,(k==M)?"+":"*") ;
  ypos = gety() ;
  if ( count < 2 * M )
     count ++ ;
  else
     {
       count = 0 ;
       count_step ++ ;
       step += step_width ;
     }
}       /* end PULSE * COEF */
else
 {                                      /* PAM_OUTPUT */
   switch (graph_type)
   {
     case '0' : {  /* stair-case */
                  if (observe == '0')  /* baseband */
                    if (channel == 'a')
                      moveto(aposx,aposy) ;
                    else
                      moveto(bposx,bposy) ;
                  lineto(step,gety()) ;      /* STAIR-CASE */
                  lineto(getx(),(maxy/2)-quantized_value);
                  setlinestyle(0,1,1) ;
                 } break ;
     case '1' : { setcolor(channel == 'b' ? 4: 2);
                  outtextxy(step,(maxy/2) - quantized_value,
                            (channel=='a') ? "*" : "+") ;
                }
                break ;
     case '2' : {    /* smoothed line */
                   if (observe == '0')  /* baseband */
                     if (channel == 'a')
                       moveto(aposx,aposy) ;
                     else
                       moveto(bposx,bposy) ;
                   lineto(step,(maxy/2)-quantized_value) ;
                   setlinestyle(0,1,1) ;
                }
                break ;
     default:printf("\ndefault graph_type in display_graph");
                exit(1) ;
   }     /* end switch */
   if (observe == '0')           /* baseband */
      if (channel == 'b')
      {
        count_step ++ ;
        step += step_width ; /* next x position */
        bposx = getx() ;
        bposy = gety() ;
      }
      else      /* chnnel a */
        {
          aposx = getx() ;
```

```
                aposy = gety() ;
              }
            count_step ++ ;
            step += step_width ;  /* next x position */
            ypos = gety() ;       /* next y cursor */
        }                         /* end PAM */
    }                             /* end baseband */
  else
  {                             /* constellation */
    if (channel == 'a')
        x = quantized_value ;
    else          /* channel b */
        {
          y = quantized_value ;
          putpixel(325-x,240-y,4);
        }
  }
} /* end display */

/*************************************************************
* function quantize : calculate value for display on graph *
* require           : (double)output                        *
* return            : (int)quantized output                 *
*************************************************************/
int quantize(double output)
{
 return output >= 0 ? floor(fabs(output/sqrt(2.0))*128) :
                    -floor(fabs(output/sqrt(2.0)) * 128 ) ;
}

/*************************************************************
* function mod : modulate,quantize and store in file        *
* require      : pam_output_a,pam_output_b,(ext)j,k,L,M     *
* return       : -                                          *
*************************************************************/
int mod(double pam_a,double pam_b,int j,int L,char channel)
{
      double  c,mod_a,mod_b,time,sine,cosi ;
    int     temp=0;

      time  = (double)2*j/(double)L ;
      sine  = sin(2*PI*time) ;
    cosi  = cos(2*PI*time);
      mod_a = pam_a*cosi ;
      mod_b = pam_b*sine;

      c = mod_a + mod_b ;
    temp = quantize(c) ;
      return(temp) ;
}
```

```
/*************************************************************
*DEMOD.C-source code for demodulation from baseband files  *
*        "abb.mod","bbb.mod"                               *
*************************************************************/
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <math.h>
#define  QPSKsin  0.707  /* angle is 45 degree */
#define  QPSKcos  0.707  /* angle is 45 degree */

int    quad(int a,int b) ;
int    threshold(int data);
char   setsym(int array[]);
int    coef(int a,int b);
int    filesize(void);
int   loop;
const L = 24 ;
FILE  *fpa,*fpb,*fp;
/*****************************MAIN*************************/
void main()
{
   unsigned long count;
   int temp,i,j,a,b,loop=4,delay;
   char databyte,mode ='2';
   int array[] ={0,0,0,0,0,0,0,0};
   static int M=1 ;

   clrscr();
   printf("\n\n\t\tDELAY:");scanf("%d",&delay);
   flength_a = filelength(fileno(fpa));
   fpa = fopen("abb.mod","rb");
   fpb = fopen("bbb.mod","rb");
   fp  = fopen("outbb.mod","wb");
   for(count =0;count<=delay;count++){
        temp = getc(fpa);
   }
   for(count=0;count<=flength_a+delay;){
      for(i=0;i<loop;i++,M++){        // 4 symbols in 1 'a'
         for(j=0;j<L;j++){          // 24 samples per symbol
            a = getc(fpa);
            b = getc(fpb);
            count++;
         }
         if(M>=9){        // first 8 symbols are not data
           temp = coef(a,b,type);
           array[i] = temp;
         }
      }
      if(M >9){
        databyte = setsym(array,type);
        putc(databyte,fp);
        putch(databyte);
      }
   }
   getch();
   fclose(fp);
   fclose(fpb);
   fclose(fp);
}
```

```
/*************************************************************
* function : assign value of bits in 1 charactor from array[] *
* require  : (extern)array[],(struct)ch                       *
* return   : (char)ch.byte                                    *
*************************************************************/
char setsym(int array[])
{
  int dibit[4];
  union { char byte  ;
          struct { unsigned int dibit3: 2 ;
                   unsigned int dibit2: 2 ;
                   unsigned int dibit1: 2 ;
                   unsigned int dibit0: 2 ;
                 } quad ;
        } ch ;

  ch.quad.dibit0        = array[0] ;
  ch.quad.dibit1        = array[1] ;
  ch.quad.dibit2        = array[2] ;
  ch.quad.dibit3        = array[3] ;
  return ch.byte;
}


/*************************************************************
function coef : return coefficient of each modulation mode *
*************************************************************/
int coef(int a,int b)
{
  return quad(a,b);
}


/*****************************************
function threshold : define minimum level *
*****************************************/
int threshold(int data)
{
  int temp;

  temp = (data/2) ;
  return temp;
}


/*****************************************
function quad : return coefficient of QPSK *
*****************************************/
int quad(int a,int b)
{
 int    temp,index,tempa,tempb;
 double table[4] = {0,1,2,3};

 tempa = (a-128);
 tempb = (b-128);
 index = ( (tempa > 0) ? 2 : 0 )+( (tempb > 0)? 1 : 0) ;
 temp = table[index];
 return temp ;
}
```

```
/*****************************************************************
RING.C-source code : multiply bandpass  c.mod  with carrier
                     sin(wt) and cos(wt),
                     output baseband files are abp.mod and
                     bbp.mod
*****************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <string.h>
#include <process.h>
#include <io.h>

#define PI 3.141592654

int quan(double volt);
int demod(int c,int j,int L,int channel);

void main()
{
  int    j=1,L=24,i,c=0;
  FILE   *fp,*fpa,*fpb;
  double time,temp,a,b;
  int    fsize=0;

  clrscr();

  fp  = fopen("c.mod","rb");
  fpa = fopen("abp.mod","wb");
  fpb = fopen("bbp.mod","wb");

  fsize = filelength(fileno(fp));
  c = getc(fp);

  while( j <= fsize )
       {
            a = demod(c,j,L,'a');
            b = demod(c,j,L,'b');
            putc(a,fpa);
            putc(b,fpb);
            c = getc(fp);
            j++;
       }

      fclose(fp);
      fclose(fpa);
      fclose(fpb);
}

int quan(double output)
{
 return output >= 0 ? floor(fabs(output/sqrt(2.0)) * 128) :
}
```

```
/*****************************************************************
* function demod : multiply c.mod with sin and cos byte by byte
*
* return : bandpass signal in file abp.mod and bbp.mod
*****************************************************************/
int demod(int c,int j,int L,int channel)
{
   double  time,sine,cosi,tempc,temp,integer ;

   time=(fmod(j,L)==0)?1:modf((double)j*2/(double)L,&integer);
   sine  = sin(2*PI*time);
   cosi  = cos(2*PI*time);
   tempc = (c-128)*sqrt(2.0)/128;

   if (channel == 'a')
      temp = tempc*cosi;
   else
      temp = tempc*sine;

   return(quan(temp)+128);
}
```

```
/*************************************************************
* INPORT.C : receive data byte from printer port by ADC *
*            synchronize by checking status port        *
*************************************************************/

#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <bios.h>

#define DATA_PORT 0x0378
#define STAT_PORT 0x0379
#define CTRL_PORT 0x037A

main()
{
      int b,c,s,count=0 ;
      FILE *fp;

      clrscr() ;
      fp = fopen("c.rx","wb") ;
      printf("\tTHIS IS DATA READ FROM PRINTER PORT\n");

      outportb(DATA_PORT,0xFF);

      for(b=0;b<=filesize;b++)
        {
          s = inportb(STAT_PORT);
          s = s&0x40;
          while(s==0x40)
            {
                s =inportb(STAT_PORT);
                s=s & 0x40 ;
            }
            c = inportb(DATA_PORT);
              //delay loop
            for(count=0;count<11500;count++){}

            if(c>=255)
                c =254;
            if(c<=0)
                c = 1 ;
            printf("%c",c);
            putc(c,fp);
      }
     fclose(fp) ;
}
```

```
/*****************************************************************
* OUTPORT.C : source code to send data from file c.mod
*             to printer port byte-by-byte
*****************************************************************/


#include <dos.h>
#include <conio.h>
#include <stdio.h>

#define ESC 0x1B

#define DATA_PORT 0x0378
#define CTRL_PORT 0x037A

main()
{
  int data_out = '\x0';
  unsigned long count,loop;
  FILE *fp;

  clrscr();
  biosprint(1,' ',0);   /*initial LPT1,send 0 to reset pin16 */
  data_out=0xFF;

  fp = fopen("c.mod","rb"); /* file to send  out to DATA_PORT */

  data_out = getc(fp);
  outportb(CTRL_PORT,0);
  outportb(CTRL_PORT,4);      /* reset output device*/
  loop = 2000;

  while(data_out != EOF)
  {
    outportb(DATA_PORT,data_out);

    for(count = 0 ; count < loop ; count++) { }

    outportb(CTRL_PORT,5);      /* 0101   strobe H to L */

    for(count = 0 ; count < (loop/1000) ; count++) { }

    outportb(CTRL_PORT,4);      /* 0100   strobe L to H */

    data_out=getc(fp);

    putch(data_out);
  }
  fclose(fp);
}
```

เอกสารอ้างอิง

[1] F. Davarian and J.T. Sumida, "A Multipurpose Digital Modulator." IEE Communication Magazine, pp. 36-45, Feb. 1987.

[2] W.F.Tomasi, "Advanced Electronic Communications Systems," Prentice-Hall, 1987.

[3] R.A.Meyers, "Encyclopedia of Telecommunications,"Academic Pres, 1989.

[4] W.Schweber, "Electronic Communication Systems," Prentice-Hall, 1991.

[5] H.Stark, F.Tuteur, and J.B.Anderson, "Modern Electrical Communications," Prentice-Hall, 1988.

[6] H.W.Couch, "Digital and Analog Communication Systems," Macmillan, 1987.

[7] H.B.Killen, " Digital Communications with Fiber Optic and Satellite Applications," Prentice-Hall, 1988.

[8] M.S.Roden,"Digital Communication System Design,"Prentice-Hall, 1988.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดีตามเป้าหมายที่ตั้งไว้อันเนื่องมาจากความร่วมมือเป็นอย่างดีของเพื่อนร่วมงานในกลุ่ม เพื่อนหลายคนคอยช่วยเหลือและให้กำลังใจ สำหรับบุคคลที่สำคัญเป็นอย่างยิ่ง ที่ทำให้ปริญญานิพนธ์นี้ประสบผลสำเร็จ คืออาจารย์ที่ปรึกษา ผศ.ดร.กอบชัย เดชหาญ ซึ่งให้คำปรึกษาและกำลังใจตลอดมา รวมทั้งรุ่นพี่อีกสองท่านที่คอยช่วยในการแก้ไขปัญหาในการทำงาน คือ พี่นพรัตน์ สิงโตโรจน์ และ พี่คงคา คูณพันธ์ จึงขอขอบพระคุณทุกท่านมา ณ ที่นี้ด้วย