การมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง โดยใช้การ์ด TMS320C5x DSP Starter Kit

Quadrature Phase Shift Keying with TMS320C5x DSP Starter Kit

โดย

นางสาวปวีณา ครำมไพบูลย์

นายพรชัย สถิตพงศ์สถาพร

นายสุรพจน์ ฤทธิ์ฉิม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

การมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง โดยใช้การ์ด TMS320C5x DSP Starter Kit

Quadrature Phase Shift Keying with TMS320C5x DSP Starter Kit

โดย

นางสาวปวีณา คร้ามไพบูลย์

นายพรชัย สถิตพงศ์สถาพร

นายสุรพจน์ ฤทธิ์ฉิ้ม

อาจารย์ที่ปรึกษา

อาจารย์เทอดศักดิ์ ลิ่วหาทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง
โดยใช้การ์ด TMS320C5x DSP Starter Kit

Quadrature Phase Shift Keying with the TMS320C5x DSP Starter Kit

ผู้จัดทำ

1. นางสาวปวีณา คร้ามไพบูลย์ 37014251
2. นายพรชัย สถิตพงศ์สถาพร 37014274
3. นายสุรพจน์ ฤทธิ์ฉิ้ม 37014529

(อาจารย์เทอดศักดิ์ ลิ่วหาทอง)
อาจารย์ที่ปรึกษา

การมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง

โดยใช้การ์ด TMS320C5x DSP Starter Kit

Quadrature Phase Shift Keying with the TMS320C5x DSP Starter Kit

1. นางสาวปวีณา คร้ามไพบูลย์ 37014251
2. นายพรชัย สถิตพงศ์สถาพร 37014274
3. นายสุรพจน์ ฤทธิ์ฉิ้ม 37014529

โครงงานได้รับการตรวจสอบแน่นอนแล้ว พร้อมที่จะทำการสอบได้

(อาจารย์เทอดศักดิ์ ลิ่วหาทอง)

อาจารย์ที่ปรึกษา

กิตติกรรมประกาศ

ในระหว่างการดำเนินงานของโครงงานนี้ได้รับความอนุเคราะห์จากผู้มีพระคุณหลายท่าน จนสามารถดำเนินการเสร็จลุล่วงไปด้วยดี โดยเฉพาะอาจารย์ที่ปรึกษาที่ให้คำปรึกษาและเอาใจใส่เป็นอย่างดี ขอขอบคุณผู้มีพระคุณทุกท่านไว้ ณ ที่นี้

(นางสาวปวีณา คร้ามไพบูลย์)

(นายพรชัย สถิตพงศ์สถาพร)

(นายสุรพจน์ ฤทธิ์ฉิม)

การมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง
โดยใช้การ์ด TMS320C5x DSP Starter Kit

นางสาวปวีณา คร้ามไพบูลย์
นายพรชัย สถิตพงศ์สถาพร
นายสุรพจน์ ฤทธิ์ฉิ้ม
อ. เทอดศักดิ์ ลิ่วหาทอง(อาจารย์ที่ปรึกษา)
ปีการศึกษา 2540

บทคัดย่อ

โครงงานนี้เป็นการรับส่งข้อมูลจากตัวส่งไปยังตัวรับโดยใช้การมอดูเลชั่นและดีมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิง (Quadrature Phase Shift Keying : QPSK) ซึ่งดิจิตอลมอดูเลชั่นแบบหนึ่งที่สามารถส่งข้อมูลได้ 2 บิท ไปกับสัญญาณพาหะ 1 ลูกคลื่น โดยที่ความถี่ของสัญญาณพาหะจะเปลี่ยนไปตามข้อมูล ดังนี้

- สัญญาณจะเป็นเฟส 0° เมื่อข้อมูลเป็น 00
- สัญญาณจะเป็นเฟส 90° เมื่อข้อมูลเป็น 01
- สัญญาณจะเป็นเฟส 180° เมื่อข้อมูลเป็น 11
- สัญญาณจะเป็นเฟส 270° เมื่อข้อมูลเป็น 10

ในการประมวลผลสัญญาณในขั้นตอนต่างๆได้ประยุกต์ ใช้การประมวลผลแบบเชิงเลข (Digial Signal Processing : DSP) โดยใช้บอร์ด TMS320C5x DSP Starter Kit ซึ่งเป็นบอร์ดที่มีคำสั่งและฟังก์ชันที่สนับสนุนการประมวลผลแบบดิจิตอล

Quadrature Phase Shift Keying with TMS320C5x DSP Starter Kit

Miss. Paweena Krampaibul
Mr. Ponchai Satipongsataporn
Mr. Surapod Ridchim
Mr. Thurdsuk Leauhatong(Advisor)

1997

Abstract

This thesis is about sending signal from the transmitter to the receiver by using the digital algorithm of modulation and demodulation which called 'Quadrature Phase Shift Keying' or 'QPSK' . Its advantage is that it can transfer 2 bits data at a time. The carrier signal will be shift to response the input data.

- Signal Phase is 0°, when data are 00.
- Signal Phase is 90°, when data are 01.
- Signal Phase is 180°, when data are 11.
- Signal Phase is 270°, when data are 10.

The processing that is used in this thesis is 'Digital Signal Processing (DSP)' by using the TMS320C5x DSP Starter Kit which has many functions that support the digital signal processing.

สารบัญ

หน้า

หน้า

สารบัญรูปภาพ

หน้า

หน้า

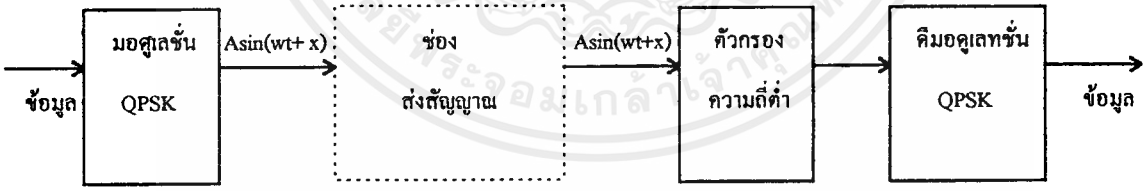
หน้า

บทที่ 1

บทนำ

การส่งสัญญาณข้อมูลจากจุดหนึ่งไปยังจุดหนึ่งเนื่องจากสัญญาณข้อมูลมีกำลังต่ำจึงจำเป็นต้องมีการมอดูเลทสัญญาณข้อมูลกับคลื่นพาหะ โดยโครงงานนี้เลือกใช้วิธี Quadrature phase shift keying (QPSK) ซึ่งเป็นดิจิตอลมอดูเลชั่นที่แอมพลิจูดและความถี่จะคงที่ แต่เฟสของสัญญาณจะเปลี่ยนไปตามข้อมูลที่ต้องการส่ง และสามารถส่งข้อมูล 2 บิทไปพร้อมกันคือ เฟสของคลื่นพาหะ $A\cos(wt + x)$ จะเปลี่ยนไปตามสถานะของสัญญาณ ดังนี้

เฟส x เป็น 0	กรณีข้อมูลเป็น 0 0
เฟส x เป็น $\pi/2$	กรณีข้อมูลเป็น 0 1
เฟส x เป็น π	กรณีข้อมูลเป็น 1 1
เฟส x เป็น $3\pi/2$	กรณีข้อมูลเป็น 0 0



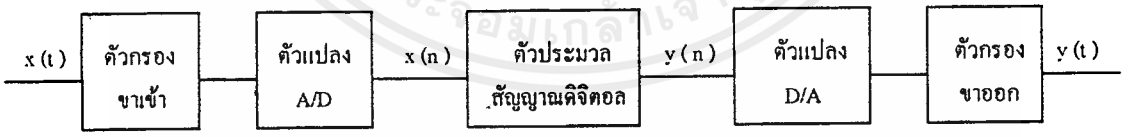
รูปที่ 1.1 บล็อกไดอะแกรมแสดงการทำงานโดยรวมของโครงงานนี้

จากรูปที่ 1.1 โครงงานนี้จะประกอบไปด้วยส่วนหลัก 4 ส่วน คือ

1. ส่วนมอดูเลชั่น เป็นส่วนที่ทำการรวมข้อมูลเข้ากับคลื่นพาหะที่จะมีเฟสเปลี่ยนไปตามลักษณะของบิทข้อมูล 2 บิทดังกล่าวข้างต้น โดยจะทำการดีเทคข้อมูลดิจิตอลทีละ 2 บิท แล้วทำการแปลงรหัส 2 บิทนั้นไปเป็นการเลื่อนของเฟสของคลื่นพาหะ
2. ส่วนช่องส่งสัญญาณ เป็นสื่อกลางในการขอเคลื่อนย้ายข้อมูลจากตัวส่งไปยังตัวรับผ่านทางสายส่ง
3. ส่วนตัวกรองความถี่ต่ำ เป็นส่วนที่ทำหน้าที่กำจัดสัญญาณรบกวนที่อาจเกิดขึ้นเนื่องมาจากการเคลื่อนย้ายข้อมูล
4. ส่วนดีมอดูเลชั่น เป็นส่วนที่ทำการดีเทคข้อมูลออกจากคลื่นพาหะในขั้นแรกจะทำกาตรวจจับความถี่ของคลื่นพาหะเพื่อนำมาเปรียบเทียบว่า สัญญาณที่รับเข้ามามีเฟสเท่าใดแล้วจึงแปลงกับไปเป็นข้อมูลดิจิตอล

โดยโครงงานนี้จะใช้บอร์ด TMS320C5x DSP Stater Kit เพื่อประยุกต์ใช้งานในลักษณะดังกล่าวข้างต้น

สัญญาณที่นิยมใช้ใน DSP มักจะแปลงมาจากสัญญาณอนาลอกซึ่งถูกแซมเปิลในช่วงเวลาคงที่และถูกแปลงให้อยู่ในรูปสัญญาณดิจิตอล



รูปที่1.2 แสดงบล็อกไดอะแกรมแสดงระบบประมวลผลสัญญาณดิจิตอล (DSP)

ปัจจุบัน DSP ถูกนำมาใช้ในงานที่เคย ทำด้วยอนาลอกมาก่อนและนำมาประยุกต์ใช้กับงานที่แต่เดิมทำด้วยอนาลอกนั้นยุ่งยากหรือไม่สามารถ ทำได้ด้วยวิธีทางอนาลอก ข้อได้เปรียบของDSP มีดังนี้

- มีความแม่นยำสูงโดยจะพิจารณาได้จากจำนวนบิทที่ใช้
- ไม่แปรตามอุณหภูมิ
- เนื่องจากใช้เทคโนโลยีของสารกึ่งตัวนำจึงทำให้กินไฟต่ำ ขนาดเล็ก ราคาถูก และมีความเร็วสูง
- ระบบ DSP สามารถทำการเขียนโปรแกรมใหม่ เพิ่มฟังก์ชันการใช้งานอื่นๆได้โดยไม่ต้องแก้ไขส่วนฮาร์ดแวร์
- DSP สามารถใช้แสดงฟังก์ชันที่ไม่สามารถเกิดขึ้นได้ในการประมวลผลสัญญาณอนาลอก เช่น ผลตอบสนองเชิงเส้น เป็นต้น
- เหมาะสำหรับอุปกรณ์ที่ข้อมูลอยู่ในรูปสัญญาณดิจิตอลอยู่แล้ว เช่น ผลลัพธ์จากคอมพิวเตอร์
- การรับส่งข้อมูลหรือสัญญาณดิจิตอลทำได้แน่นอนกว่า ทั้งนี้เนื่องจากสัญญาณดิจิตอลมีแค่ 2 ระดับคือ 0 และ 1 เท่านั้น ถ้าหากรูปคลื่นสัญญาณผิดเพี้ยนไปก็สามารถแก้ไขและสร้างขึ้นมาใหม่ให้เหมือนเดิมได้ง่าย
- การประมวลผลสัญญาณดิจิตอลทำได้ง่าย ทั้งนี้เนื่องจากขั้นตอนวิธี (algorithm)ในการประมวลสัญญาณมักประกอบด้วยการบวก การลบ การคูณ การหาร และการเลื่อนตัวเลขเท่านั้น
- ระบบประมวลผลดิจิตอลสามารถทำเป็นแบบระบบแบ่งกันใช้เวลา (time -sharing) ได้

การใช้งาน DSP ก็ยังมีข้อเสียเปรียบอยู่ แต่ข้อด้อยเหล่านี้สามารถทำให้ลดน้อยลงเรื่อยๆได้เมื่อเทคโนโลยีพัฒนาขึ้น

- มีปัญหาการเชื่อมโยง (interfacing) กับระบบการประมวลสัญญาณอนาลอกทั้งนี้เนื่องมาจากระบบที่สร้างขึ้นมาใหม่ต้องต่อเชื่อมโยงกับระบบเดิมให้ได้ ซึ่งอาจทำให้ระบบประมวลสัญญาณซับซ้อนขึ้น
- แถบปฏิบัติงานของระบบประมวลสัญญาณประมวลดิจิตอลต่ำกว่าแถบปฏิบัติงานของระบบประมวลผลอนาลอกมาก ข้อจำกัดนี้เนื่องมาจากอุปกรณ์ที่ใช้หรือประกอบขึ้นเป็นระบบการประมวลสัญญาณดิจิตอล เช่น วงจรเกต วงจรชิฟต์รีจิสเตอร์ วงจรสุ่มและคงค่าสัญญาณ(sampling and hold circuit หรือ S/H) หรือ วงจร A/D และ D/A เป็นต้น

วงจรเหล่านี้ต่างมีความเร็วสูงสุดในการทำงานจำกัดอยู่ค่าหนึ่ง ซึ่งในปัจจุบันยังมีค่าต่ำมากจึงเป็นผลทำให้ระบบประมวลผลสัญญาณดิจิตอลมีความเร็วต่ำ เช่น ถ้าใช้ไมโครโปรเซสเซอร์ที่มีสัญญาณนาฬิกาขนาด 1 MHz เมื่อนำไปสร้างระบบประมวลผลสัญญาณดิจิตอล จะใช้ได้สัญญาณที่มีความถี่สูงสุดประมาณ 10 - 20 KHz

- ความยาวจำกัดของคำ ในการใช้งาน DSP นั้นจะใช้จำนวนบิทจำกัดมีผลทำให้เกิดสัญญาณกระเพื่อมขึ้น

ขั้นตอนการแปลงสัญญาณอนาลอกไปเป็นสัญญาณดิจิตอล

ก่อนที่จะทำการประมวลผลด้วย DSP สัญญาณจะต้องอยู่ในรูปสัญญาณดิจิตอลเสียก่อนแต่เนื่องจากสัญญาณโดยทั่วไปจะเป็นสัญญาณอนาลอก ซึ่งต้องมีกระบวนการแปลงสัญญาณเหล่านี้ให้เป็นสัญญาณดิจิตอลซึ่งมีขั้นตอนดังนี้

- สัญญาณที่มีแบนด์วิดท์จำกัดจะถูกแซมเปิล ซึ่งเป็นการแปลงสัญญาณอนาลอกไปเป็นสัญญาณเต็มหน่วย (discrete signal) ที่มีการเปลี่ยนแปลงแอมพลิจูดอย่างต่อเนื่อง (discrete-time continuous amplitude signal)
- ขนาดแอมพลิจูดของสัญญาณเต็มหน่วยจะถูกปรับค่าให้มีค่าใดค่าหนึ่งใน 2^B ค่า (เมื่อ B เป็นจำนวนบิทที่ใช้แสดงการแซมเปิล ใน ADC
- ค่าแอมพลิจูดของสัญญาณเต็มหน่วยจะถูกแทนด้วยเลขฐานสอง ที่มีความยาว B บิต

บทที่ 2

TMS320C50

TMS320C50 เป็นโปรเซสเซอร์ที่ทำงานด้าน DSP (Digital Signal Processing) ซึ่งตระกูล TMS320 จะมีอยู่หลายเบอร์ เช่น TMS320C50 , TMS320C51 และ TMS320C53 ซึ่งเป็นการพัฒนาสถาปัตยกรรมของ ' C25 ให้ดีขึ้น ทั้งในด้านความเร็วและฟังก์ชันการทำงานอื่นๆ

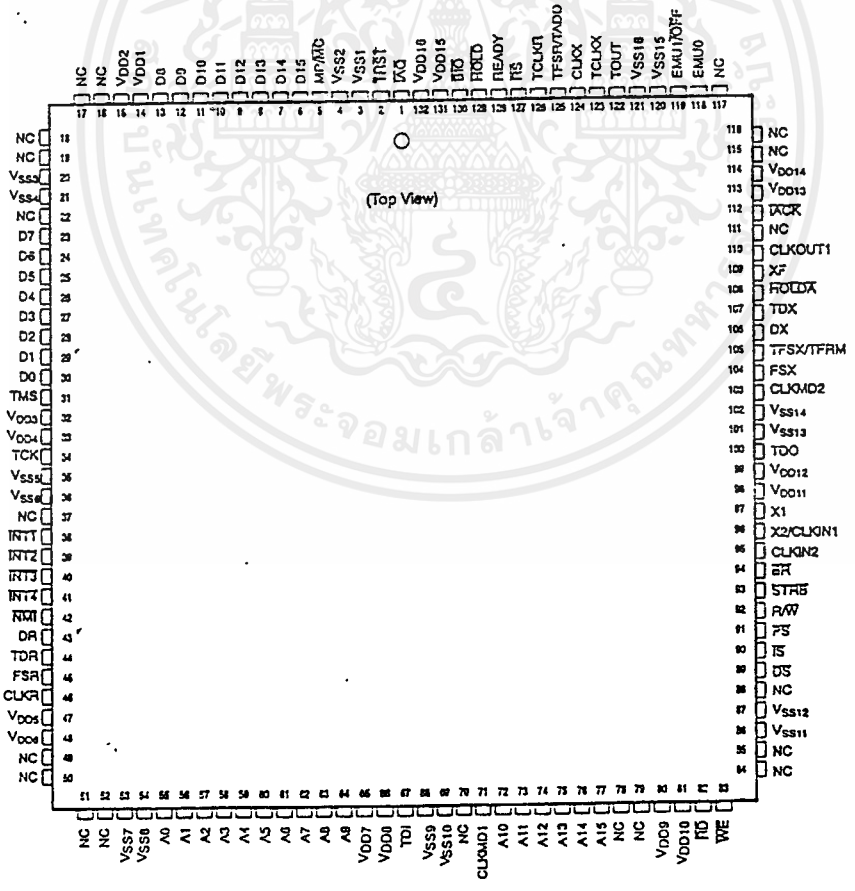
2.1 จุดเด่นของ TMS320C50x

- มีแรมบนบอร์ดขนาด 10 K เวิร์ด
- มีรอมขนาด 2K x 16 บิทสำหรับโปรแกรมบูท
- มีแรมสำหรับ โปรแกรม/ข้อมูล ขนาด 9K x 16 บิท
- ขยายหน่วยความจำภายนอกได้ถึง 224K x 16 บิท
- ทำงานด้วยความเร็ว 30 - 50 ns ต่อหนึ่งคำสั่ง
- มี ALU (Arithmetic Logic Unit) , ACC (Accumulator) และ ACCB (Accumulator buffer) ขนาด 32 บิท
- PLU (Parallel Logic Unit) ขนาด 16 บิท
- มีรีจิสเตอร์ 8 ตัว
- สามารถอ้างแอดเดรสแบบเซอร์คูลาร์ (circular) โดยการใช้เซอร์คูลาร์บัฟเฟอร์ (circular buffer)
- พอร์ตอนุกรมที่รับส่งแบบฟูลดูเพล็ก (Full duplex) สำหรับ 'C50 กับอุปกรณ์อื่นๆ
- TDM (Time-division multiple) ของพอร์ตอนุกรม
- กำหนดสัญญาณนาฬิกา โดยใช้ไทม์เมอร์ (Timer) หรือ เคาน์เตอร์(Counter)
- สามารถผลิตสัญญาณนาฬิกา โดยการหารสัญญาณนาฬิกาที่จ่ายให้ซีพียู
- I/O พอร์ตขนาด 16 K และมี 16 ตำแหน่ง สำหรับการเข้าถึงหน่วยความจำ
- สามารถทำงานแบบไปป์ไลน์(Pipeline)

- ใช้เทคโนโลยีของซีมอส(CMOS) และใช้ไฟเลี้ยง 5 โวลท์
- คำสั่งโปรแกรมที่ใช้จะช่วยให้เขียนโปรแกรมด้าน DSP ได้ง่ายขึ้น
- สถาปัตยกรรมของ TMS320C50 สร้างขึ้นมาเพื่อความเร็วในการทำงานด้าน DSP และเพื่อให้การทำงานของบัสไม่ขึ้นต่อกันจึงแยกเป็นบัสโปรแกรม(program bus) และ บัสข้อมูล (data bus) ออกจากกัน โดยบัสโปรแกรมจะเป็นทางเข้าของรหัสคำสั่งและโอเปอร์แรนด์ของคำสั่ง ส่วนบัสข้อมูลจะเชื่อมต่อโดยตรงระหว่างหน่วยความจำที่ใช้เก็บข้อมูลกับวงจรประมวลผล เช่น ALU

2.2 ตำแหน่งขาและหน้าที่การทำงานของTMS320C50

ตำแหน่งขาแสดงดังรูปที่ 2.1 และหน้าที่การทำงานของTMS320C50แสดงดังตารางที่ 2.1



รูปที่ 2.1 แสดงขาของTMS320C50x

ตารางที่ 2.1 แสดงตำแหน่งขาและหน้าที่การทำงานของTMS320C50

สัญญาณ	ขา	สถานะ	การทำงาน
กลุ่มบัสแอดเดรสและบัสข้อมูล(Adress and Data bus)			
A15(MSB) A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0(LSB)	77 76 75 74 73 72 64 63 62 61 60 59 58 57 56 55	I/O/Z	เป็นบัสแบบขนาน (Parallel Address Bus) ใช้สำหรับชี้ตำแหน่งของหน่วยความจำข้อมูลและหน่วยความจำโปรแกรม หรือ I/O ภายนอก เมื่ออยู่ในโฮลด์โหมด (hold mode) จะมีสถานะเป็นอิมพิแดนซ์สูง (High Impedance)
D15(MSB) D14 D13	6 7 8	I/O/Z	เป็นบัสข้อมูลแบบขนาน (parallel data bus) ใช้ส่งผ่านข้อมูลระหว่างซีพียูหลัก (core CPU) กับหน่วยความจำข้อมูลฝโปรแกรมภายนอก หรือ อุปกรณ์ I/O เมื่อไม่มี

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0	9 10 11 12 13 23 24 25 26 27 28 29 30		เอาท์พุทสัญญาณเหล่านี้จะเป็นอิมพีแดนซ์สูง
กลุ่มสัญญาณควบคุมหน่วยความจำ			
DS PS IS	89 91 90	O/Z	เลือกหน่วยความจำข้อมูล/โปรแกรมและ I/O ปกติมีสถานะสูงแต่เมื่อเป็นสถานะต่ำจะเป็นการติดต่อกับภายนอกเมื่อขา OFF อยู่ในสถานะต่ำจะอยู่ในสถานะอิมพีแดนซ์สูง
READY	128	I	สัญญาณข้อมูลพร้อม (Data ready input) ใช้แสดงเมื่ออุปกรณ์ภายนอกส่งข้อมูลเรียบร้อยแล้ว และเมื่อยังทำงานไม่เสร็จ (READY=0) จะต้องรอ 1 ไซเคิลและเช็คขา READY อีกครั้ง ในสภาวะปกติขา READY จะทำงานหลังจากที่มีสัญญาณ BR
R/W	92	I/O/Z	สัญญาณอ่าน/เขียน เป็นสัญญาณควบคุมการอ่านและเขียนข้อมูลจะเป็นสถานะอิมพีแดนซ์สูงเมื่ออยู่ในโหมดโฮลด์

STRB	93	I/O/Z	สัญญาณสโตรบ (strob signal) ปกติมักอยู่ในสถานะสูง จะเป็นสถานะต่ำเพื่อบอกการอ้างอิงกับบัสภายนอก เป็นอิมพีแดนซ์สูงเมื่ออยู่ในโฮลด์โหมด
RD	82	O/Z	สัญญาณเลือกอ่าน (read select) ขานี้จะทำงานเมื่อมีการอ่าน จะต่อโดยตรงกับ OE ของอุปกรณ์ภายนอก สัญญาณนี้จะใช้ในการอ่านค่าหน่วยความจำโปรแกรม/ข้อมูล และ I/O ภายนอกทั้งหมด เป็นอิมพีแดนซ์สูงเมื่ออยู่ในโฮลด์โหมด
WE	83	O/Z	สัญญาณเขียน (write enable) จะใช้สำหรับการเขียนค่าในหน่วยความจำโปรแกรม/ข้อมูล และ I/Oภายนอกทั้งหมด เป็นอิมพีแดนซ์สูงเมื่ออยู่ในโหมดโฮลด์
กลุ่มสัญญาณมัลติโปรเซสซิง (Multiprocessing)			
HOLD	129	I	สัญญาณโฮลด์ (hold input) เป็นสัญญาณที่ใช้เพื่อแสดงว่ากำลังมีการติดต่อกับบัสตำแหน่ง, บัสข้อมูล และบัสควบคุม เมื่อได้รับการตอบรับจาก'C50 (acknowledge) จะอยู่ในสภาวะอิมพีแดนซ์สูง
HOLDA	108	O/Z	สัญญาณตอบรับสัญญาณโฮลด์ (hold acknowledge signal) ใช้แสดงว่าวงจรอยู่ในสถานะโฮลด์ (hold state) ทั้งบัสตำแหน่ง, บัสข้อมูลและบัสควบคุมอยู่ในสภาวะอิมพีแดนซ์สูง
BR	94	I/O/Z	สัญญาณการขอใช้บัส (bus request signal) แสดงเมื่อมีการติดต่อกับหน่วยความจำข้อมูล สัญญาณจากขานี้ใช้กับหน่วยความจำข้อมูลที่ว่างได้ 32K เวิร์ด เมื่อขา HOLDA อยู่ในสถานะต่ำสัญญาณนี้ใช้กับ DMA ของแรมภายนอก BR จะเป็นสถานะต่ำเมื่อติดต่อกับแรมภายนอก
IOQ	1	O/Z	สัญญาณรับคำสั่ง (instruction acquisition signal) จะแสดงสถานะต่ำเมื่อมีตำแหน่งของคำสั่งอยู่บนแอดเดรสบัส

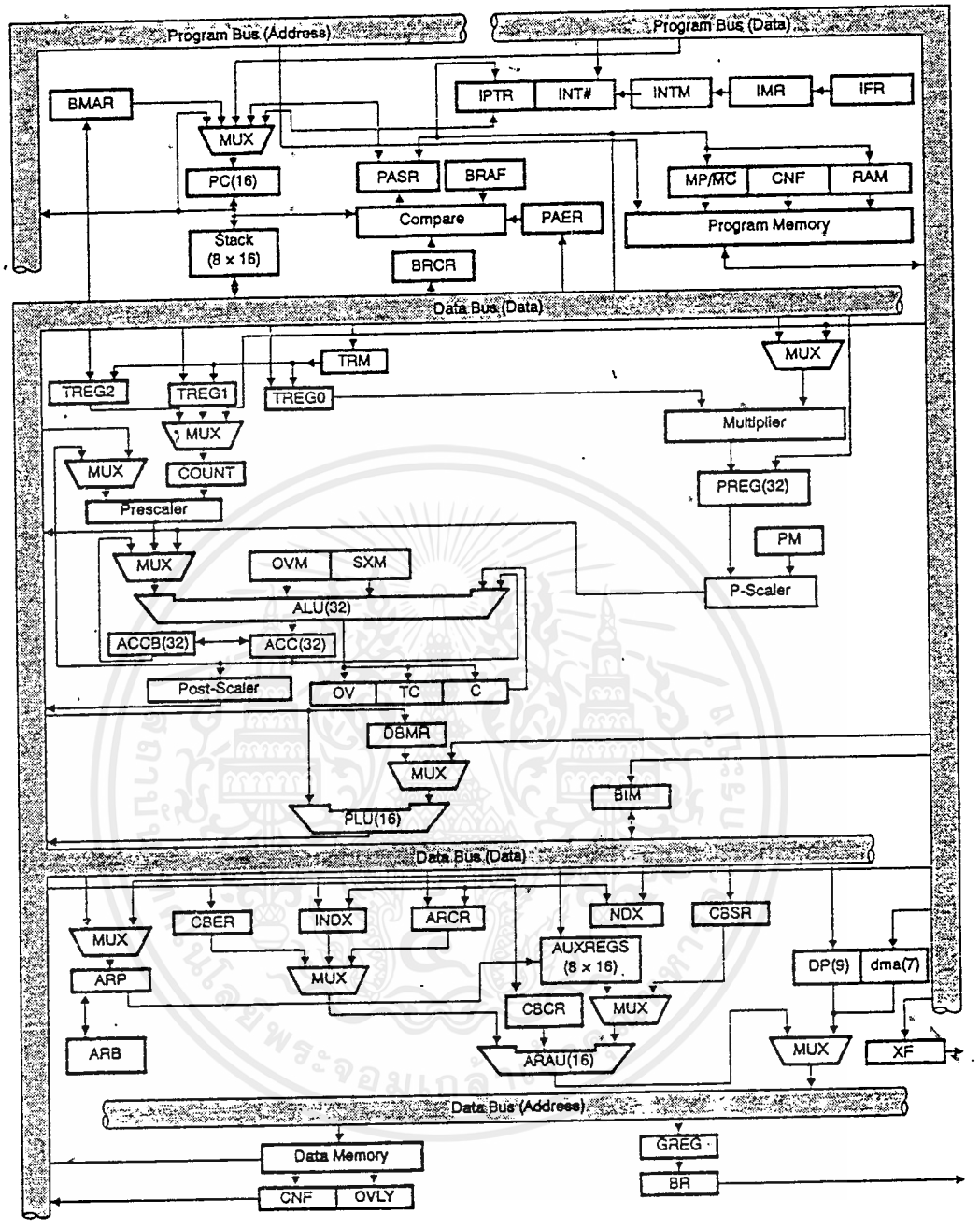
BIO	130	I	สัญญาณควบคุมบรานซ์ (branch control input) จะเป็นสถานะต่ำเมื่อมีการทำงานในคำสั่งเงื่อนไข
XF	109	O/Z	สัญญาณติดต่อภายนอก (external flag output) ถูกเซตให้เป็นสถานะสูง หรือ สถานะต่ำ โดยคำสั่งพิเศษหรือโดยการโหลดค่าในสเตตรีจิสเตอร์ (state register (ST1)) เมื่อรีเซตขานี้จะเป็นสถานะสูง
LACK	112	O/Z	สัญญาณตอบรับอินเทอร์รัพต์ (interrupt acknowledge signal) แสดงค่าเมื่อรับค่าอินเทอร์รัพต์และโปรแกรมเคาท์เตอร์จะนำค่า อินเทอร์รัพต์เวกเตอร์(interrupt vector) ซึ่งกำหนดโดย A15-A0
การอินนิเชียลไลซ์ (Initialization) , อินเตอร์รัพต์ , คำสั่งรีเซต			
INT4 INT3 INT2 INT1	41 40 39 38	I	สัญญาณอินเทอร์รัพต์จากผู้ใช้ภายนอก (external user interrupt input) กำหนดโดยรีจิสเตอร์ควบคุมอินเตอร์รัพต์ (interrupt mask register) และบิทอินเตอร์รัพต์โหมด (intereupt mode bit) สามารถรีเซตผ่านรีจิสเตอร์บอกอินเทอร์รัพต์ (interrupt flag register)
NMI	42	I	สัญญาณนอน-มาส์กอะเบิลอินเทอร์รัพต์ (non-maskable interrupt) เป็นอินเทอร์รัพต์ภายนอกไม่สามารถควบคุมโดย INTM หรือ MRเมื่อ NMI ทำงานจะเกิดอินเทอร์รัพต์
MC/MP	5	I	ขาเลือกโหมดไมโครโปรเซสเซอร์/ไมโครคอมพิวเตอร์ (microprocessor / microcomputer mode select pin) ถ้าเป็นสถานะต่ำ (microcomputer mode) จะทำให้โปรแกรมรอมภายในส่งไปยังหน่วยความจำโปรแกรม (program memmory space) ในโหมดไมโครโปรเซสเซอร์
สัญญาณออสซิเลเตอร์/ไทม์เมอร์ และ CLKIN1/2			
CLKOUT1	110	O/Z	สัญญาณนาฬิกาส่งออก (master clock output signal หรือ

			CLKIN2 frequency) มีค่าไซเคิลเท่ากับอัตราแมชชีนไซเคิล (machine-cycle) ของซีพียู
CLKMD1 CLKMD2	71 103	I	CLKMD1 CLKMD2 0 0 สัญญาณนาฬิกาภายนอกเป็นสัญญาณนาฬิกาเข้าจากขา X2/CLKIN ทำให้ออสซิเลเตอร์และ PLL disable 0 1 สำหรับตรวจสอบ 1 0 เป็นสัญญาณนาฬิกาอินพุต (input clock) สำหรับ CLKIN2 ทำให้ออสซิเลเตอร์ภายในหยุดทำงานและ PLL ทำงานแทน 1 1 เป็นสัญญาณนาฬิกาอินพุตสำหรับขา X2/CLKIN1 ทำให้ออสซิลเลเตอร์ภายในทำงานและPLLภายในไม่ทำงาน
X2/CLKIN	122	I	ขาอินพุตสำหรับออสซิลเลเตอร์ภายใน (input pin to internal oscillator from crystal) ถ้าออสซิลเลเตอร์ภายในไม่ถูกใช้ สัญญาณนาฬิกาจะเป็นอินพุตสำหรับอุปกรณ์บนขานี้ แมชชีนไซเคิลภายในจะมีค่าเป็นครึ่งหนึ่งของสัญญาณนาฬิกานี้
X1	97	O	เป็นขาเอาท์พุตของออสซิเลเตอร์ภายในส่งกลับไปที่คริสตัลถ้าไม่ใช้ออสซิลเลเตอร์ภายในจะไม่มีการต่อขานี้
CLKIN2	95	I	เป็นอินพุทให้สัญญาณนาฬิกาเพื่อขับอัตราแมชชีนไซเคิล

TOUT	122	O	เอาท์พุตไทม์เมอร์(timer output) ขานี้ให้สัญญาณพัลส์เมื่อไทม์เมอร์ภายใน(on-chip timer) นับถึง 0 ความกว้างพัลส์เท่ากับ CLKOUT1 ไซเคิล
สัญญาณพอร์ตอนุกรม			
CLKR CLKRT	46 126	I I	เป็นขาที่รับสัญญาณนาฬิกาจากข้างนอกเพื่อกำหนดให้การรับข้อมูล (DR/TDR) เข้าไปเก็บไว้ที่ RSR (serial port receive shift register) แต่ถ้าขานี้ไม่ใช้สามารถที่จะใช้เป็นขาอินพุต IN0 ของ SPC/TSPC รีจิสเตอร์ได้
CLKX TCLKX	124 123	I/O/Z I/O/Z	เป็นขาที่แสดงสัญญาณนาฬิกาจากข้างนอกเพื่อกำหนดให้ DR/TDR ส่งข้อมูลไปที่ DX/TDX CLKX จะเป็นอินพุตถ้า MCM บิทที่การควบคุมพอร์ตอนุกรม (serial port control) มีค่าเป็น 0 และมีความถี่เป็น (CLKOUT1) / 4 เมื่อ MCM เป็น 1 ถ้าขานี้ไม่ใช้สามารถที่จะทำเป็นอินพุตของบิท IN1 ของ SPC/TSPC รีจิสเตอร์
DR TDR	43 44	I I	เป็นขาเพื่อรับสัญญาณข้อมูล ซึ่งเมื่อรับมาแล้วจะเก็บไว้ที่ (serial port receive shift register)
DX TDX	106 107	O/Z	เป็นขาเพื่อส่งสัญญาณข้อมูล ซึ่งเมื่อรับมาแล้วจะเก็บไว้ที่ (serial port receive shift register)
FSR TFSR/TFRM	104 105	I/O/Z I/O/Z	เป็นขาเพื่อส่งสัญญาณการพร้อมของเฟรม (Frame synchronization) สำหรับรับสัญญาณอินพุต TFSR จะเป็นได้ทั้งอินพุต/เอาท์พุต (TADD) เมื่อพอร์ตอนุกรมอยู่ในโหมด TDM
TCK	34	I	สัญญาณนาฬิกาตรวจสอบ JTAG (JTAG test clock) เป็นสัญญาณนาฬิกาแบบฟรี-รันนิ่ง (free-running) ซึ่งมีค่าดิวตี้ไซเคิล (duty cycle 50%) การเปลี่ยน TAP (test access port) ซึ่งเป็นอินพุตสัญญาณนาฬิกาจะควบคุมผ่าน

			TAP รีจิสเตอร์ (instruction register) หรือเลือกการทดลองรีจิสเตอร์ข้อมูล (data register) ที่ขอบขาขึ้นของ TCK
TDI	67	I	การเปลี่ยนTAP จะปรากฏเป็นสัญญาณเอาท์พุตที่ขอบขาลงของTCK

ฮาร์ดแวร์ภายในของ 'C50x สามารถปฏิบัติการคำสั่งที่หน่วยประมวลผลอื่นต้องใช้ซอฟแวร์ช่วย ยกตัวอย่างเช่น 'C50x จะมีฮาร์ดแวร์ที่สามารถทำการคูณข้อมูล 16 x 16 บิท ได้ภายในไซเคิลเดียว , การเลื่อนข้อมูล และการอ้างอิงแอดเดรส ซึ่งจะเห็นว่าโปรเซสเซอร์ตัวนี้มีศักยภาพในการประมวลผลอย่างมาก คำอธิบายฮาร์ดแวร์ของ 'C50x ได้แสดงไว้ในตารางที่ 2.2 และบล็อกไดอะแรมแสดงฮาร์ดแวร์ภายในของ 'C50x แสดงดังรูปที่ 2.2



รูปที่ 2.2 แสดงฮาร์ดแวร์ภายในของTMS320C50

ตารางที่ 2.2 แสดงฮาร์ดแวร์ภายในของ TMS320C5x

หน่วย	สัญลักษณ์	การทำงาน
Accumulator	ACC(32) ACCH(16) ACCL(16)	มีขนาด 32 บิท แบ่งออกเป็น 2 ส่วน คือ ACCH (accumulator high) และ ACCL (accumulator low)
Accumulator Buffer	ACCB(32)	เป็นรีจิสเตอร์รที่ใช้เก็บข้อมูลของแอคคูมูเลเตอร์ไว้ชั่วคราว สามารถเป็นอินพุทให้กับ ALU ได้
Arithmetic Unit	ALU	เป็นหน่วยประมวลผลทางคณิตศาสตร์ขนาด 32 บิท
Auxiliary Register Logic Unit	ARAU	เป็นหน่วยคำนวณที่ไม่คิดเครื่องหมายขนาด 16 บิท ไว้คำนวณแอดเดรสโดยอ้อม
Auxiliary Register Compare	ARCR(16)	รีจิสเตอร์ขนาด 16 บิท ใช้จำกัดการเปรียบเทียบแอดเดรสแบบโดยอ้อม
Auxiliary Register File	AUXREGS	รีจิสเตอร์ขนาด 16 บิท จำนวณ 8 ตัว (AR0 - AR7) ใช้เป็นตัวอ้างแอดเดรสโดยอ้อม เก็บข้อมูล หรือเก็บค่าเดิมของARP
Auxiliary Register Buffer	ARB(3)	รีจิสเตอร์ขนาด 3 บิท เก็บค่าเดิมของ ARP โดยค่าทั้ง 3 บิทจะถูกเก็บอยู่ใน ST1
Auxiliary Register Pointer	ARP(3)	รีจิสเตอร์ขนาด 16 บิท เก็บค่าแอดเดรสสำหรับการเคลื่อนย้ายบล็อกข้อมูล
Block Move Address Register	BMAR(16)	รีจิสเตอร์ 16 บิท เก็บค่าแอดเดรสสำหรับการเคลื่อนย้ายบล็อก

หน่วย	สัญลักษณ์	การทำงาน
Block Repeat Active Flag	BRAF(1)	แฟลกขนาด 1 บิท ใช้แสดงสถานะการทำงานของคำสั่งซ้ำบล็อก จะมีค่าเป็น 1 เมื่อกำลังปฏิบัติคำสั่งและเป็น 0 เมื่อ BRCR รีจิสเตอร์ลดค่าลงต่ำกว่าศูนย์
Block Repeat Address End Register	PAER(16)	รีจิสเตอร์ขนาด 16 บิทอ้างอิงหน่วยความจำ ใช้เก็บค่าแอดเดรสตำแหน่งสุดท้ายของส่วนที่จะทำซ้ำ
Block Repeat Address Start Register	PASR(16)	รีจิสเตอร์ขนาด 16 บิทที่ใช้เก็บค่าแอดเดรสเริ่มต้นของส่วนที่ถูกทำซ้ำ
Block Repeat Counter Register	BRCR(16)	รีจิสเตอร์ขนาด 16 บิทที่เก็บจำนวนครั้งในการทำซ้ำ
Bus Interface Module	BIM	เป็นบัฟเฟอร์อินเตอร์เฟสที่ใช้เป็นทางผ่านของข้อมูลระหว่างข้อมูลภายในกับบัสโปรแกรม
Bus Request	BR	สัญญาณที่บอกให้ทราบว่าข้อมูลได้ถูกส่งไปยังหน่วยความจำหลัก
Carry	C	เป็นบิทที่ใช้เป็นตัวทดที่ได้จาก ALU โดยจะเก็บไว้ใน ST1
Central Arithmetic Logic Unit	CALU	เป็นกลุ่มที่ประกอบด้วย ALU , มัลติพลิเยอร์ (multiplier) , ACC scaling shifter
Circular Buffer Control Register	CBCR(8)	รีจิสเตอร์ขนาด 8 บิท ที่ใช้ enable / disable เซอร์คูลาร์บัฟเฟอร์ และบอกว่ารีจิสเตอร์ตัวใดที่จะถูกส่งไปยังเซอร์คูลาร์บัฟเฟอร์
Circular Buffer End Address	CBER(16) CBER1(16) CBER2(16)	รีจิสเตอร์ขนาด 16 บิท แสดงแอดเดรสสุดท้ายของเซอร์คูลาร์บัฟเฟอร์

หน่วย	สัญลักษณ์	การทำงาน
Circular Buffer Start Address	CBSR(16) CBSR1(16) CBSR2(16)	รีจิสเตอร์ขนาด 16 บิท แสดงแอดเดรสเริ่มต้นของเซอร์คูลาร์บัฟเฟอร์
Compare of Program Address	COMPARE	เป็นตัวเปรียบเทียบค่าปัจจุบันใน PC กับค่าใน PAER ถ้ามีค่าเท่ากันจะทำการโหลดค่าใน PASR ไปเก็บใน PC
Configure RAM	CNF	เป็นตัวบอกว่าบล็อกในแรมถูกส่งไปยังส่วนของโปรแกรมหรือส่วนของข้อมูล
Data Bus	DATA	บัสข้อมูลขนาด 16 บิท
Data Memory	DATA MEMORY	เป็นส่วนของหน่วยความจำที่ใช้เก็บข้อมูล
Data Memory Address Bus	DATA ADDRESS	บัสขนาด 16 บิท ที่ส่งค่าแอดเดรสสำหรับการอ้างอิงหน่วยความจำข้อมูล
Data Memory Address Immediate Register	dma(7)	รีจิสเตอร์ขนาด 16 บิท เก็บค่าแอดเดรสภายใน 128 เวิร์ด ในหนึ่งเพจ
Data Memory Page Pointer	DP(9)	รีจิสเตอร์ขนาด 9 บิท เก็บค่าแอดเดรสของเพจปัจจุบัน(1 เพจ = 128 เวิร์ด)
Data RAM Map Bit	RAM(1)	บิทนี้จะแสดงก็ต่อเมื่อซิงเกิลแอดเดรสแรม ถูกส่งลงในส่วนของข้อมูล
Direct Data Memory Address Bus	DRB(16)	บัสขนาด 16 บิท ที่ส่งค่าแอดเดรสโดยตรงของหน่วยความจำข้อมูล

หน่วย	สัญลักษณ์	การทำงาน
Dynamic Bit Manipulation Register	DBMR(16)	รีจิสเตอร์อ้างอิงหน่วยความจำขนาด 16 บิท ใช้เป็นมาสก์อินพุทให้กับ PLU
Dynamic Bit Pointer	TREG2(4)	รีจิสเตอร์ขนาด 4 บิท ที่เก็บค่าบิทของพอยน์เตอร์ซึ่งเปลี่ยนแปลงค่าได้ สำหรับการใช้คำสั่ง BITT
Dynamic shift Count	TREG1(5)	รีจิสเตอร์ขนาด 5 บิท ที่เก็บจำนวนอินพุตข้อมูลที่เข้า ALU
External Flag	XF(1)	เป็นบิทที่ใช้ควบคุมระดับของเฟลกภายนอกและเก็บอยู่ใน ST1
Global Memory Allocation Regisster	GREG(8)	รีจิสเตอร์ขนาด 8 บิท สำหรับระบุขนาดของข้อมูลแบบโกลบอล
Hold Mode	HM(1)	บิทที่เก็บอยู่ใน ST1 และเป็นตัวตัดสินว่า CPU จะหยุดหรือทำงานต่อไปดี
Index Register	INDX(16	รีจิสเตอร์ 16 บิท ใช้ในการอ้างอิงแอดเดรสโดยอ้อมได้
Index Register Enable	NDX(1)	เป็นตัวตัดสินว่าจะทำการแก้ไขหรือทำการเขียนค่าให้ AR0
Interrupt Flag Register	IFR(16)	รีจิสเตอร์ขนาด 16 บิท ใช้เก็บสถานะอินเทอร์รัพต์ที่เข้ามา
Interrupt Mask Bit	INTM(1)	เป็นตัวระบุการอนุญาติการอินเทอร์รัพต์
Interrupt Number	INT#(4)	เก็บจำนวนการอินเทอร์รัพต์เฉพาะ ที่ส่งไปยัง CPU
Interrupt Pointer	IPTR(5)	ใช้บิท 5 บิท เพื่อชี้ไปยัง 2 K เพจ ที่ใช้เก็บอินเทอร์รัพต์เวคเตอร์ บิททั้งห้านี้จะเก็บอยู่ใน PMST

หน่วย	สัญลักษณ์	การทำงาน
Interrupt Mask Register	IMR(16)	รีจิสเตอร์ขนาด 16 บิท ใช้สำหรับมาสก์อินเทอร์รัพต์
Microcall Stack	MCS(15 - 0)	สแตกขนาด 1 เวิร์ด ที่ใช้เก็บค่าของ PFC ชั่วคราวในขณะที่ PFC กำลังถูกใช้ระบุทำแหน่งของหน่วยความจำข้อมูลด้วยวิธีเคลื่อนย้ายบล็อก
Microprocessor/ Microcomputer Mode	MP/MC	บิทนี้อยู่ใน PMST รีจิสเตอร์ใช้แสดงการส่งข้อมูลไปยังตำแหน่งที่ว่างของส่วนโปรแกรม
Moutiplexer	MUX	มัลติเพลกเซอร์ (bus miltiplexer) ใช้เลือกแหล่งที่มาของโอเปอร์แรนด์สำหรับบัสหรือหน่วยปฏิบัติการ
Multiplier	MULTIPLIER	มัลติไพล์เออร์ขนาด 16 x 16 บิท
Overflow Flag	OV(1)	บิทนี้เก็บอยู่ใน ST0 ใช้แสดงการเกิดโอเวอร์โฟล์ ในการประมวลผลทางคณิตศาสตร์ของ ALU
Overflow Mode	OVM(1)	เป็นตัวตัดสินว่าโอเวอร์โฟล์ที่เกิดขึ้นนั้นเป็นแบบใด ค่านี้จะเก็บอยู่ใน ST0
Overlay to Data Space	OVLY(1)	เป็นตัวตัดสินว่าจะสามารถอ้างอิงตำแหน่งในที่ว่างที่เก็บค่าแอดเดรสได้หรือไม่ บิทนี้อยู่ใน PMST
Parallel Logic Unit	PLU	เป็นตัวประมวลผลตรรกะที่แยกต่างหากจาก CALU โดยข้อมูลรับมาจากคำสั่งโดยตรงหรือรับจาก DBMR
Prefetch Counter	PCF(15 - 0)	รีจิสเตอร์เคาท์เตอร์ขนาด 16 บิท จะเก็บค่าแอดเดรสของคำสั่งที่เพิ่งเฟชมาไว้ก่อน (prefetch)
Prescale Counter Register	COUNT(4)	รีจิสเตอร์ขนาด 4 บิท ซึ่งเก็บค่าที่ใช้ในการกระทำ prescaling

หน่วย	สัญลักษณ์	การทำงาน
Product Register	PREG(32)	รีจิสเตอร์ขนาด 32 บิท ใช้เก็บผลลัพธ์ที่ได้จากการคูณ
Program Bus	PROG DATA	รีจิสเตอร์ขนาด 16 บิท ใช้สำหรับคำสั่ง route
Program Counter	PC(16)	รีจิสเตอร์ขนาด 16 บิท ใช้ระบุแอดเดรสของหน่วยความจำโปรแกรม
Program Memory	PROGRAM MEMORY	หน่วยความจำที่ใช้เก็บโปรแกรม
Program Memory Address Bus	PROG ADDRESS	บัสขนาด 16 บิท ใช้ส่งค่าแอดเดรสของโปรแกรม
Prescalling Shifter	PRESCALER	ตัวเลื่อนซ้าย 0 - 16 บิท ใช้สำหรับprescale ข้อมูลที่เข้าไปใน ALU
Postscalling Shifter	POSTSCALER	ตัวเลื่อนซ้ายขนาด 0 - 7 บิท ใช้สำหรับ postscale ข้อมูลที่ออกจาก CALU
Product Shifter	P-SCALER	ตัวเลื่อนซ้ายขนาด 0 , 1 หรือ 4 บิทที่สามารถเลื่อนบิทเครื่องหมายพิเศษได้
Producter Mode	PM(2)	มีขนาด 2 บิท ใช้ระบุโหมดการเลื่อนผลลัพธ์
Repeat Counter	RPTC(16)	รีจิสเตอร์เคาท์เตอร์ขนาด 16 บิท ใช้ควบคุมการปฏิบัติการซ้ำของคำสั่งหนึ่งคำสั่ง
Sign Extension Mode	SXM(1)	ใช้ควบคุมว่าการปฏิบัติการทางคณิตศาสตร์จะเป็นแบบมีเครื่องหมายหรือไม่ ค่านี้จะเก็บใน ST1
Stack	STACK	ฮาร์ดแวร์ สแตกขนาด 8 x 16 บิท เก็บค่า PC ในระหว่างการอินเตอร์รัพต์หรือการเรียกใช้โปรแกรมย่อย

หน่วย	สัญลักษณ์	การทำงาน
Status Register	ST0 , ST1 PMST	รีจิสเตอร์ขนาด 16 บิท ทั้งสามตัวใช้เก็บบิทแสดงสถานะและบิทควบคุม
Temporary Multiplicand	PREGO(16)	รีจิสเตอร์ขนาด 16 บิท เก็บค่าชั่วคราวของข้อมูลที่ใช้ในการคูณ
Temporary Register Enable	TRM(1)	เป็นตัวแสดงว่าคำสั่ง LT(A,D,P,S) ได้ทำการโหลดค่าทั้งสามของ TREG(0,1,2)
Test/Control Flag	TC(1)	บิทที่ใช้เก็บค่าผลลัพธ์ที่ได้จากการทดสอบของ ALU หรือ PLU

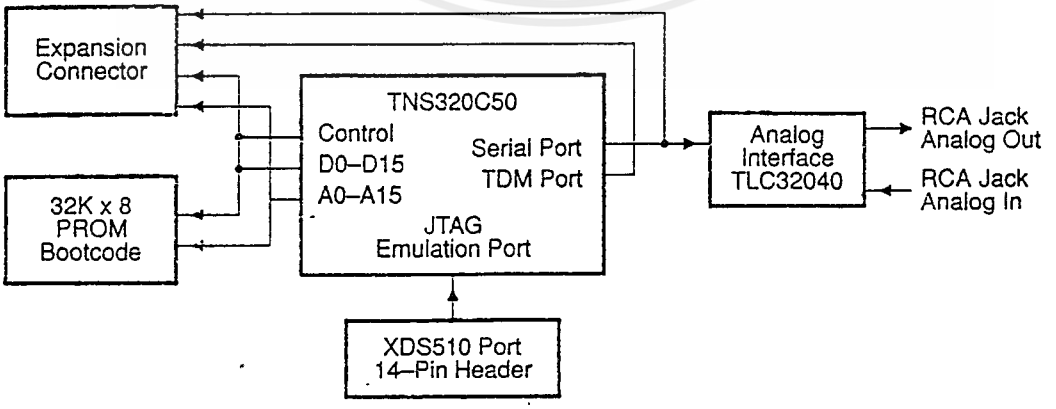
บทที่ 3

โครงสร้างของ TMS 320C50 DSP Starter Kit (50'DSK)

3.1 ลักษณะทั่วไปของบอร์ด

- ใช้โปรเซสเซอร์เบอร์ TMS320C50
- คำสั่งใช้เวลาประมาณ 50 ns (instruction cycle time)
- 32K-byte PROM
- สามารถใช้กับสัญญาในช่วงความถี่เสียง โดยผ่านทาง TLC32040
- ใช้มาตรฐาน RCA คอนเนคเตอร์ สำหรับ อนาลอกอินพุท และเอาท์พุท เพื่อให้สามารถต่อเข้ากับไมโครโฟน และลำโพงได้โดยตรง
- ในการติดต่อกับการควบคุม มีคอนเนคเตอร์ XDS510 ติดต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม
- สามารถขยายI/O บัสได้เพื่อใช้สำหรับการออกแบบภายนอก

ในรูปที่ 3.1 แสดงบล็อกไดอะแกรมของบอร์ด ซึ่งประกอบไปด้วย โฮสอินเตอร์เฟส , อนาลอกอินเตอร์เฟส และ อีมูเลชั่นพอร์ต ทำให้สามารถติดต่อกับ พีซีได้โดยผ่านทาง RS232 นอกจากนี้ยังมี PROM ขนาด 32K bytes ที่ใช้เก็บเคอร์แนลโปรมแกรมไว้สำหรับการบูท



รูปที่ 3.1 บล็อกไดอะแกรมของ TMS320C5x DSK

ส่วนของอนาลอกอินเตอร์เฟส (Analog Interface) ใช้ TLC32040 ซึ่งเป็นวงจรอินเตอร์เฟสสัญญาอนาลอก (analog interface cirvuit : AIC) ที่มี RCA คอนเน็กเตอร์ 2 ตัวสำหรับ อินพุท และเอาท์พุท

จากรูปที่ 3.1 การทำงานเริ่มต้นโดยการบูทโลดเดอร์ (Bootloader) ซึ่งบรรจุบนพรอมบูทโลดเดอร์ (PROM Bootloader) ขนาด 32K เบอร์ 27PC256FM จะทำการบูท (Boot) บอร์ด เป็นการเซ็ตค่าที่จำเป็นสำหรับ TMS ซึ่งใช้เป็นดีบักเกอร์ (debugger) ของ DSK

เมื่อทำการรีเซต เริ่มต้นที่ตำแหน่ง 000H โดยขารีเซต หรือ ที่ *BR* จะเป็นสภานะต่ำ (low) ทำให้เรียกเคอร์เนลโปรแกรม (Kernel program) โดยใช้ขา *BIO* และ XF ของ TMS ในการติดต่อกับ PC ทาง RS232

ในการติดต่อเมื่อ $\overline{BIO}$ เป็นสภานะต่ำ (low) จะเป็นการแสดงว่าเริ่มติดต่อ RS232 ซึ่งจะเป็นการกำหนดบิทเริ่มต้น สำหรับการคำนวณ โดยเริ่มที่ 1 บิทเริ่มต้น + 7 บิทข้อมูล แล้วหารด้วย 8 จะได้อัตราบอด (Baud Rate) ที่คำนวณจากคำสั่ง NOP ในโครงงานนี้กำหนดอัตราบอดอยู่ที่ 9600 bit/sec

3.1.1 วงจรส่วน CPU TMS320C50

ขา $MP/\overline{MC}$ ของ TMS302C50 ต่อลงกราวด์เพื่อให้อยู่ในโหมดไมโครคอมพิวเตอร์ ในส่วนของอินเตอร์รัพต์ *INT1 - INT4* และ *NMI* จะแอคทีฟในสถานะต่ำ (low) ต้องมีตัวต้านทาน ต่อเข้ากับไฟเลี้ยงไว้ ส่วนของการติดต่อกับ RS232 จะต่อที่ *BIO* และ *XF* ซึ่งจะกล่าวอีกทีในการอธิบายส่วนของการติดต่อกับ PC ส่วนสัญญานาฬิกาของระบบจะใช้คริสตอลขนาด 40 MHz ต่อเข้าที่ *CLKIN* ในส่วนของ *CLKOUT1* จะต่อไปยัง TLC32040

การติดต่อกับหน่วยความจำจะใช้แอดเดรสบัส A0-A15 จะอ่านเขียนโดยกำหนดที่ขา *R/W* , *RD* , *WE* และ *STRB*

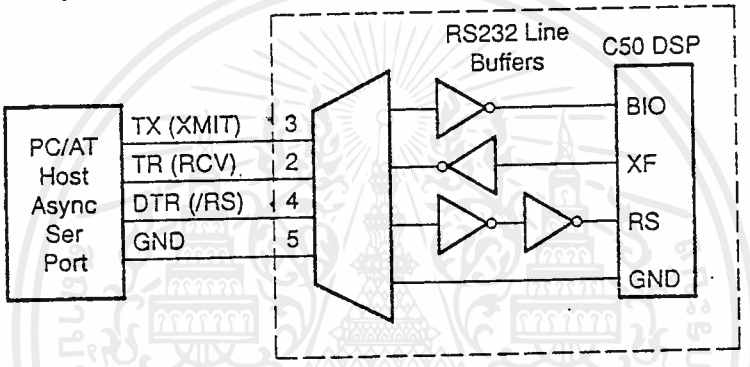
ในการติดต่อกับ TLC32040 เป็นการติดต่อแบบอนุกรมดดยมีขา DR สำหรับรับข้อมูลและขา DX สำหรับส่งข้อมูลโดยกำหนดสัญญาที่ CLKR และ CLX ในการติดต่อจะติดต่อเป็นเฟรม โดยมีการกำหนดที่ FSR และ FSX กำหนดเฟรมในการรับและเฟรมในการส่งตามลำดับ

3.1.2 วงจรส่วนการติดต่อกับ RS232

การติดต่อกับ PC ผ่านทางพอร์ตอนุกรม ในโครงงานนี้จะใช้ COM2 มีขั้นตอนดังนี้ ข้อมูล TTL มาต่อกับ *INT2* และ *BIO* เข้าที่ TMS320C50 อีกที

ในการส่งข้อมูลจาก TMS ไปยัง PC จะผ่านที่ขา TX โดยมี IC เบอร์ 75188 เปลี่ยนระดับสัญญาณ TTL เป็นสัญญาณ RS232 อีกที

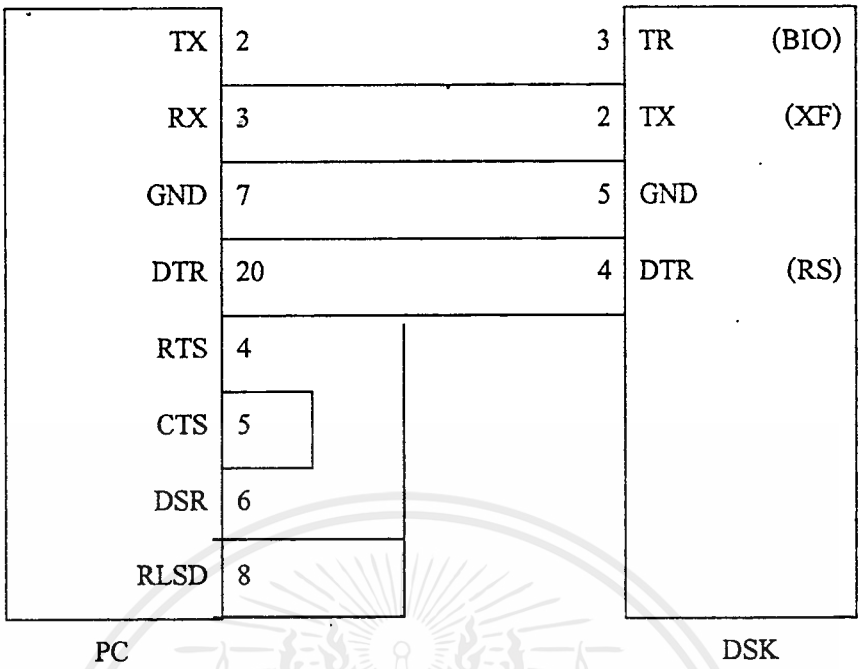
ส่วนขา DTR จะใช้เป็นขาสำหรับรีเซตบอร์ด ซึ่งมี 75189X2 สำหรับเปลี่ยนระดับสัญญาณ การต่อเพิ่มเติมในส่วนของการติดต่อทาง RS232 จะมีการต่อดังรูปที่ 3.2



DTR : data terminal ready

รูปที่ 3.2 แสดงการติดต่อกับ RS232

การรับสัญญาณจาก RS232 มาเป็น TTL และ จาก TTL มาเป็น RS232 จะใช้ IC 75189 และ 75188 ตามลำดับ โดยที่ปกติแล้ว ระดับ RS232 +12 V จะเป็น logic 0 และ -12 V จะเป็น logic 1 ซึ่งจะต่อดังรูปที่ 3.3



รูปที่ 3.3 สัญญาณในการติดต่อกับ RS232

จากผลการทดลองเมื่อไม่มีการรับส่งจะได้

TX เป็น (-10V) logic 1

TR เป็น (-10V) logic 1

DTR เป็น (+10V) logic0

DSR เป็น (+10V) logic 0

3.1.3 วงจรส่วนที่ติดต่อกับ TLC32040

จากที่ได้กล่าวมาแล้ว การติดต่อระหว่าง TMS320C50 กับ TLC32040 จะผ่านทางพอร์ตอนุกรม โดยขาที่ใช้จะเป็น *DX*, *DR*, *FSR*, *FSX* และ *CLK* สัญญาณนาฬิกาของ TLC32040 จะได้มาจาก TMS32040 ทาง *TOUT*

ในการติดต่อกับสัญญาณอนาลอกภายนอกจะเข้ามาทาง *AIN+* หรือ *AUXIN* $\pm$ เข้ามายัง TLC32040 ในการส่งข้อมูลอนาลอก จะส่งออกทาง *OUT+*, *OUT-*

3.1.4 วงจรส่วนที่ติดต่อกับหน่วยความจำ

เนื่องจากบอร์ดมี PROM เบอร์ 27PC256 ซึ่งเก็บบูทโหลดเดอร์ไว้ และส่วนของการติดต่อจะใช้บัสแอดเดรส A0-A14 และบัสข้อมูล D0-D7ในการอ่านใช้ขา $\overline{RD}$ และ $\overline{OE}$ โดยการเลือกชิพที่ $\overline{CE}$ ทาง $\overline{BR}$

ในส่วนของ RAM ที่ใช้ จะใช้ภายใน TM320C50 เอง ซึ่งมีขนาด 9K สำหรับหน่วยความจำโปรมแกรม และ 9K สำหรับหน่วยความจำข้อมูลซึ่งจะอธิบายอีกครั้งในหัวข้อหน่วยความจำ

3.1.5 วงจรส่วนไฟเลี้ยงและหัวต่อขยายบอร์ด

บอร์ดทำงานโดยใช้ไฟขนาด $\pm$5 V โดย เรคติไฟ (rectified) ไฟเข้ามา 9V จากหม้อแปลงในการทำให้แรงดัน คงที่จะใช้ IC LM805 สำหรับไฟบวก และใช้ LM7905 สำหรับไฟลบ ส่วนการขยายบอร์ด หรือส่วนที่สามารถต่อกับภายนอกได้จะมีถึง 5 ส่วน (JP1-JP5) ซึ่งจะเป็นการต่อกับขาของ TMS320C50 และ TLC32040 ทั้งหมด

3.2 ข้อกำหนดต่างๆที่ใช้ในบอร์ด

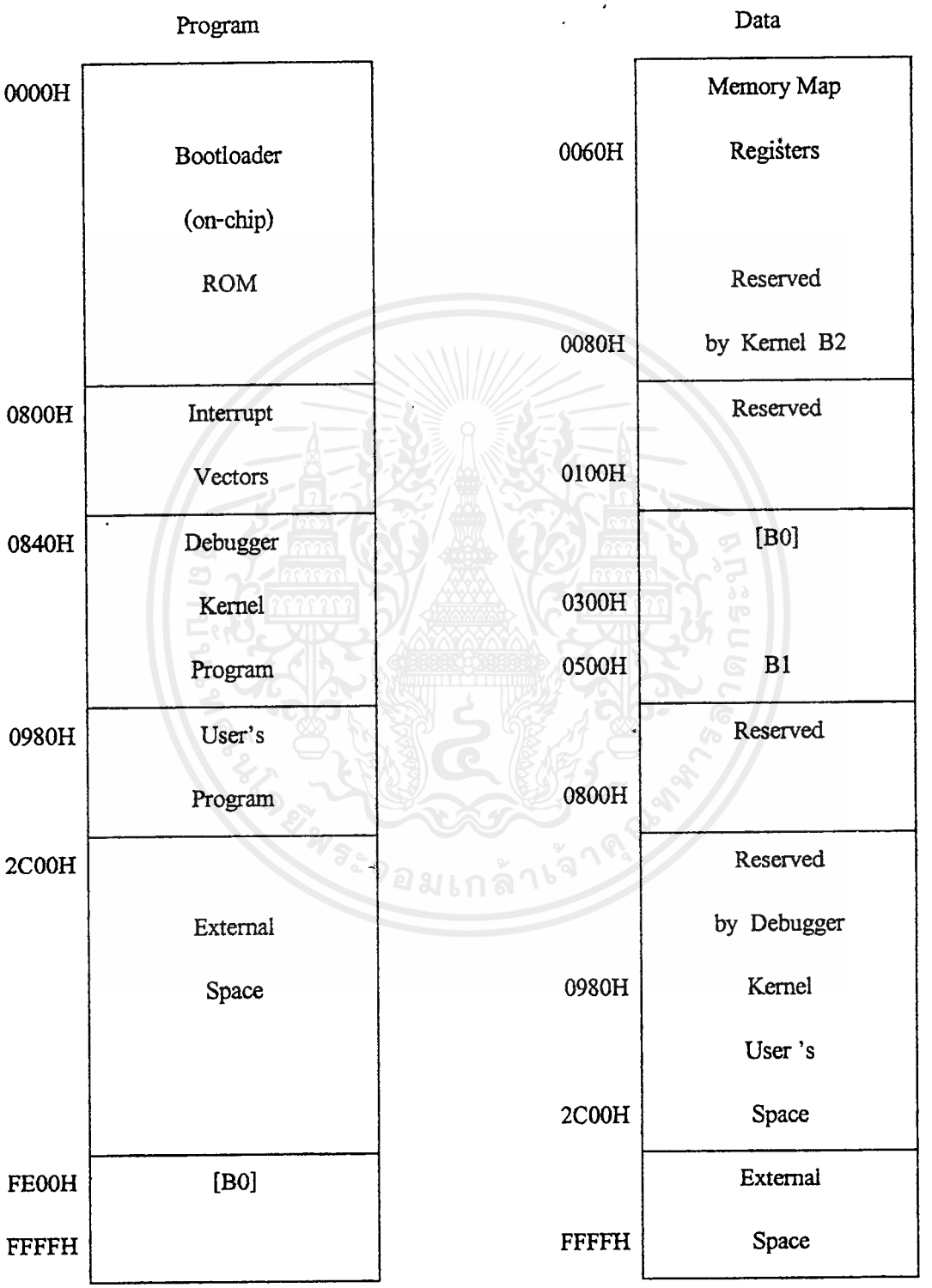
1. RAM พื้นที่ B2 เก็บ ซีพียูรีจิสเตอร์ (CPU register)
2. SARAM ขนาด 384 เวิร์ด เก็บค่าที่ใช้ติดต่อทาง RS232
3. ซีพียูต้องอยู่ในโหมดไมโครคอมพิวเตอร์เพราะจะใช้ติดต่อหรือบูทโหลดเดอร์ที่โปรแกรมภายใน
4. สงวน $\overline{INT2}$ สำหรับติดต่อกับ RS232 ที่ขา $\overline{BIO}$
5. ซอฟแวร์อินเตอร์รัพต์ (TRAP) จะใช้สำหรับซิงเกิลสเตรป (single step) ของดีบักเกอร์

หน่วยความจำใน C50' DSK นั้น จะแบ่งออกเป็นหน่วยความจำสำหรับโปรแกรม และหน่วยความจำสำหรับข้อมูล ซึ่งในแต่ละแอดเดรส ก็จะถูกใช้งานต่างๆ ตามรูปที่ 3.4 และผู้ใช้สามารถใช้หน่วยความจำภายนอกขนาด 10K ได้โดยไม่จำเป็นต้องต่อภายนอก

TLC32040 บนบอร์ด จะมีลักษณะดังนี้

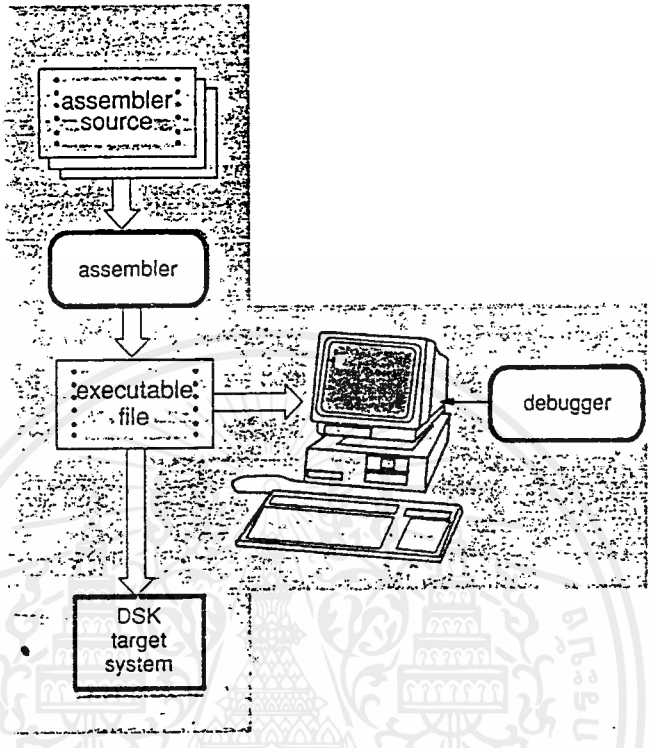
- ใช้ในการแปลง A/D และ D/A ขนาด 14 บิท
- สามารถเปลี่ยนแปลงการแซมปลิ้งทั้งหมดของ A/D ,D/A และความถี่ฟิลเตอร์ได้

การติดต่อระหว่าง AIC และ TMS320C50 จะผ่านทางพอร์ตอนุกรมภายในบอร์ด



รูปที่ 3.4 แผนภาพแสดงหน่วยความจำภายใน C'50 DSK

การต่อใช้งาน 'C50 DSK กับPC จะใช้ขา XF และ BIOที่ต่อพอร์ตอนุกรม RS232 ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 แสดงการต่อระหว่าง DSK และ PC โดยผ่านทาง RS232

นอกจากนี้ DSK ยังมีแอสเซมเบอร์ และ ดีบักเกอร์ ของมันเอง ซึ่งจะมีประโยชน์มากเนื่องจากภาษาแอสเซมบลีที่ใช้ใน DSK นั้นจะสนับสนุนการทำงานด้านประมวลผลสัญญาณ (Singnal Processing) แล ดีบักเกอร์ก็มีความสามารถที่จะทำงานในแบบทีละขั้น (single-step) และเบรคพอยท์เพื่อช่วยในการแก้ไขโปรแกรม

3.3 การสร้างโปรแกรมเพื่อใช้กับDSK

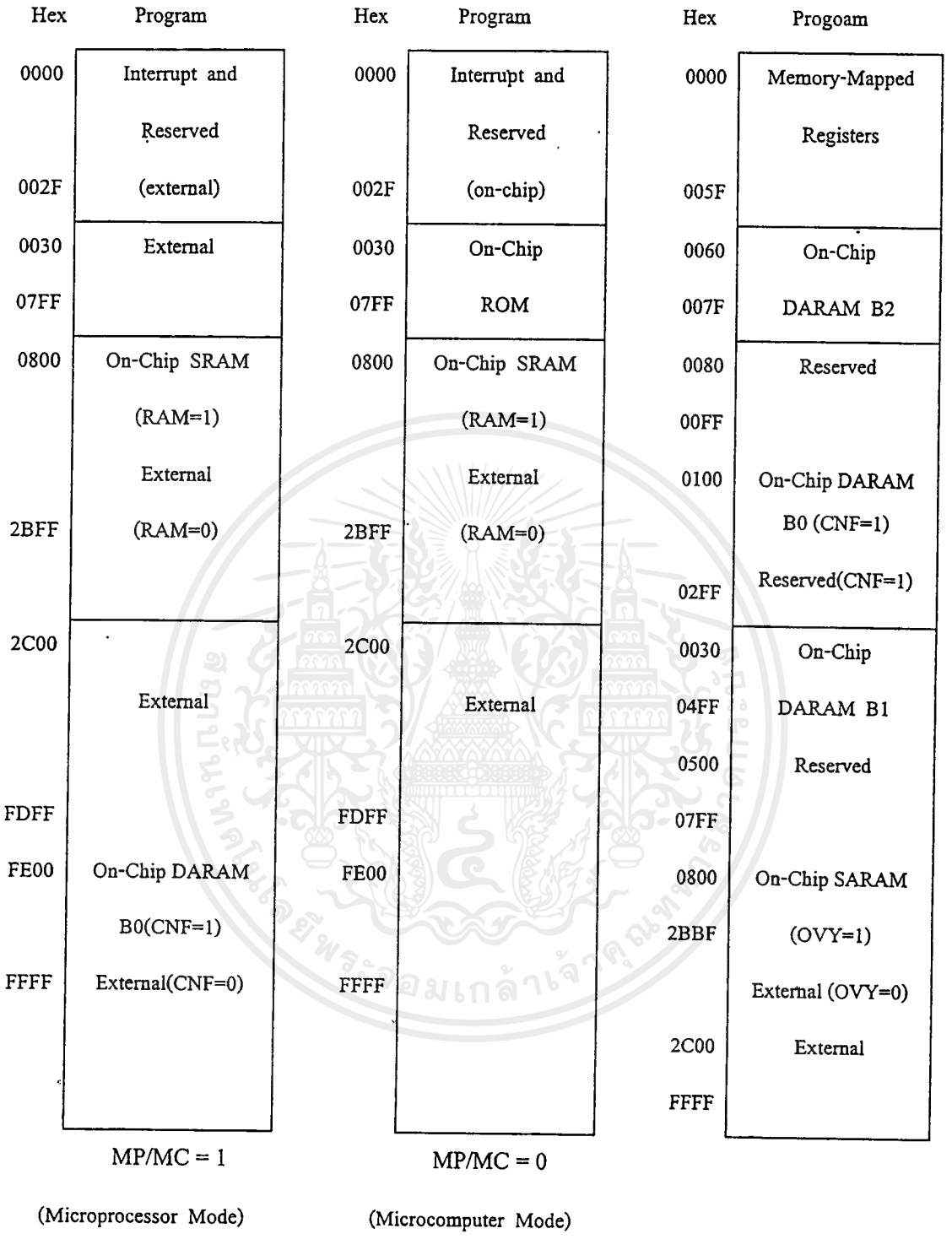
1. ทำการสร้างไฟล์หลัก(source file) สำหรับโปรแกรม เช่น exam.asm

   แปลงไฟล์หลัก โดยใช้ DSK แอสเซมเบอร์ ซึ่งมีคำสั่ง : dsk5a eaxm.asm

2. หากต้องการตรวจแก้ไขโปรแกรมโดยใช้ดีบักเกอร์ ก็ทำได้โดยโปรมแกรม dsk5.exeซึ่งได้ให้มากับบอร์ดแล้ว

3.4 หน่วยความจำ(memory)

หน่วยความจำทั้งหมดที่ TMS320C50 สามารถอ้างอิงได้ทั้งหมดเป็น 224K x 16 บิท โดยจะประกอบด้วย program memory 64K , local data memory 64K , global datamemory 32K และในส่วนที่ใช้อ้างอิง I/O port 64K โดยที่การออกแบบ TSM320C50 ใช้สถาปัตยกรรมแบบฮาร์วาร์ด(เป็นการแยกบัสข้อมูลและบัสโปรแกรมออกจากกัน) มีรอมในตัว 2K เวิร์ด มีแรมแบบ SARAM (single-access RAM) 9K และ DARAM (dual-access RAM) 1056 เวิร์ด ในส่วนของDARAM นั้น จะถูกแบ่งออกเป็น 3 ส่วน แบ่งเป็น block0 (B0) มี 512 เวิร์ด (ตำแหน่ง 0100h-02FFh ใน local data memory หรือ 0FE00h-0FFFFh ใน program space) , block1(B1) มี 512 เวิร์ด ที่ ตำแหน่ง 0300h-04FFh ใน local data memory และ block(B2) มี 32 เวิร์ด ที่ตำแหน่ง 0060h-007Fh ใน local data memory ดังรูปที่ 3.6



รูปที่ 3.6 แสดงการใช้งานหน่วยความจำภายในเมื่ออยู่ในโหมดไมโครโปรเซสเซอร์และไมโครคอนโทรเลอร์

3.5 หน่วยความจำโปรแกรม (Program memory)

ในหน่วยความจำโปรแกรม สามารถขยายได้ถึง 64K แต่ในการใช้หน่วยความจำภายในบอร์ดจะมีข้อดีกว่าคือจะไม่มีสภาวะรอ(wait state) โดยที่การกำหนดสถานะต่างๆของ C50 จะกำหนดตามตารางที่ 3.1

ตารางที่ 3.1 แสดงการกำหนดค่าสำหรับใช้หน่วยความจำโปรแกรม

CNF	RAM	*MP/$\overline{MC}$*	ROM	SARAM	DRAM B0	OFF-CHIP
0	0	0	0000-07FF			0800-FFFF
0	0	1				0000-FFFF
0	1	0	0000-07FF	2000-23FF		2C00-FFFF
0	1	1		2000-23FF		0000-FFFF
						2C00-FFFF
1	0	0	0000-07FF		FE00-FFFF	0800-FDFF
1	0	1			FE00-FFFF	0000-FDFF
1	1	0	0000-07FF	2000-23FF	FE00-FFFF	2C00-FDFF
1	1	1		2000-23FF	FE00-FFFF	0000-07FF
						2C00-FDFF

การใช้งานของการอินเทอร์รัพต์จะต้องกำหนดเวกเตอร์ของแอดเดรสในหน่วยความจำโปรแกรมซึ่งอินเทอร์รัพต์เวกเตอร์ แสดงดังตารางที่ 3.2

ตารางที่ 3.2 แสดงตำแหน่งของอินเทอร์รัพต์เวคเตอร์

Name	Location		Priority	Function
	Dec	Hex		
$\overline{RS}$	0	0	1(highest)	External reset signal
$\overline{NMI}$	36	24	2	Nonmaskable interrupt
$\overline{INT1}$	2	2	3	External user interrup#1
$\overline{INT2}$	4	4	4	External user interrup#2
$\overline{INT3}$	6	6	5	External user interrup#3
TINT	8	8	6	External user interrup
RINT	10	A	7	Serial port receive interrupt
XINT	12	C	8	Serial port transmit interrupt
TRNT	14	E	9	TDM port recevice interrupt
TXNT	16	10	10	TDM port transmit interrupt
$\overline{INT4}$	18	12	11	External user interrupt#4
-----	20-33	14-21	N/A	Reserved
TRAP	34	22	N/A	Trap instruction vector
------	38-39	26-27	N/A	Reseved
------	40-63	28-3F	N/A	Software interrupt

3.6 หน่วยความจำแบบ local data (Local Data Memory)

ใน 'C50 จะมีแรมแบบ SARAM อยู่ 9K และ DARAM อยู่ 1056 word ซึ่งสามารถแสดงรายละเอียดการเซ็ตค่าของบิทควบคุมหน่วยความจำ(Control state Register) ได้ตามตารางที่ 3.3

การเซ็ตบิทรีจีสเตอร์ควบคุมหน่วยความจำเพื่อใช้หน่วยความจำข้อมูล

ตารางที่ 3.3 แสดงการกำหนดค่าการใช้งาน Local Memory

CNF	OVLY	DARAM B0	DARAM B1	DARAM B2	SARAM	Off-Chip
0	0	100h-2FFh	300h-4FFh	60h-7Fh		80h-FFFFh
0	1	100h-2FFh	300h-4FFh	60h-7Fh	800-2BFFh	2C00-FFFFh
1	0	-	300h-4FFh	60h-7Fh		2C00-FFFFh
1	1	-	300h-4FFh	60h-7Fh	800-2BFFh	80h-FFFFh

เนื่องจากการติดต่อกับรีจีสเตอร์ต่างๆ และการส่งค่าจะทำบนเพจ 0 ซึ่งจะเป็นการใช้แบบส่งแบบรีจีสเตอร์(Register Memory Map) ดังตารางที่ 3.4

ตารางที่ 3.4 แสดงตำแหน่งแอดเดรสข้อมูลเพจ 0

Name	Address		Description
	Dec	Hex	
Core Processor Memory-Mapped Registers			
--------	0-3	0-3	Reserved
IMR	4	4	Interrupt Mask Register
GREG	5	5	Global Memory Allocation Register
IFR	6	6	Interrupt Flag Register
PMST	7	7	Processor Mode Status Register
RPTC	8	8	Repeat Counter Register
BRCE	9	9	Block Repeat Counter Register

Name	Address		Description
	Dec	Hex	
PASR	10	A	Block Repeat Program Address Start Register
PAER	11	B	Block Repeat Program Address End Register
TRGE0	12	C	Temporary Register Used for Multiplicand
TRGE1	13	D	Temporary Register Used for Dynamic Shift Count (5 bits only)
TREG2	14	E	Temporary Register Used as Bit Pointer In Dynamic Bit Test (4 bits only)
DBMR	15	F	Dynamic Bit Manipulation Register
AR0	16	10	Auxiliary Register Zero
AR1	17	11	Auxiliary Register One
AR2	18	12	Auxiliary Register Two
AR3	19	13	Auxiliary Register Three
AR4	20	14	Auxiliary Register Four
AR5	21	15	Auxiliary Register Five
AR6	22	16	Auxiliary Register Six
AR7	23	17	Auxiliary Register Seven
INDX	24	18	Index Register
ARCR	25	19	Auxiliary Register Compare Register
CBSR1	26	1A	Circular Buffer 1 Start Register
CBSR1	27	1B	Circular Buffer 1 End Register
CBSR2	28	1C	Circular Buffer 2 Start Register
CBSR2	29	1D	Circular Buffer 2 End Register
CBCR	30	1E	Circular Buffer Control Register

Name	Address		Description
	Dec	Hex	
	31	1F	Block Move Address Register
Peripheral Memory-Mapped Registers			
ORR	32	20	Data Receive Register
DXR	33	21	Data Transmit Register
SPC	34	22	Serial Port Control Register
-----	35	23	Reserved
Peripheral Memory-Mapped Registers (Continued)			
TIM	36	24	Timer Register
PRO	37	25	Period Register
TCR	38	26	Timer Control Register
----	39	27	Reserved
PDWSR	40	28	Program/Data S/W Wait-state Register
IOWSR	41	29	I/O Port S/W Wait-State Register
CWSR	42	2A	Control S/W Wait-State Register
----	43-47	2B-2F	Reserved for Test/Emulation
TRCV	48	30	TDM Data Receive Register
TOXR	49	31	TDM Data Transmit Register
TSPC	50	32	TDM Serial Port Control Register
TCSR	51	33	TDM Channel Select Register

Name	Address		Description
	Dec	Hex	
TRTA	52	34	Received/Transmit Address Register
TRAD	53	35	Received Address Register
------	54-79	36-4F	Reserved
Memory-mapped I/O Ports			
PA0	80	50	I/O Port 80
PA1	81	51	I/O Port 81
PA2	82	52	I/O Port 82
PA3	83	53	I/O Port 83
PA4	84	54	I/O Port 84
PA5	85	55	I/O Port 85
PA6	86	56	I/O Port 86

3.6.1 Auxillary Register (AR0-AR7)

auxiliary register ทั้ง 8 ตัวสามารถเก็บค่าได้ 16 บิท สามารถเข้าถึงได้จาก CALU และสามารถเปลี่ยนแปลงค่าได้จาก ARAU หรือ PLU ความสามารถขั้นแรกของการใช้ auxiliry register เป็นการเก็บค่าที่อยู่ของข้อมูลทั้งหมด 16 บิท นอกจากนี้ยังสามารถใช้เก็บค่าต่างๆ หรือจะใช้เป็นตัวนับก็ได้

3.6.2 Auxiliary Register Compare Register (ARCR)

auxiliary register compare register เป็นตัวเก็บค่า 16 บิทที่ใช้ในการกำหนดขอบเขตการเปรียบเทียบ โดยที่ ARCR เป็นตัวเปรียบเทียบกับตัวที่ AR ที่ถูกเลือกไว้จากคำสั่ง CMPR และผลจากการเปรียบเทียบจะถูกเก็บไว้ใน TC บิทใน ST1

3.6.3 Index Register (INDX)

เป็นรีจิสเตอร์ที่ ARAU ใช้ในการแก้ไขแอดเดรสโดยทางอ้อมไปยังรีจิสเตอร์ช่วย(Auxiliary Register)

3.6.4 Circular Buffer Register (CBSR1 , CBER1 , CBSR2 , CBER2 , CBCR)

อุปกรณ์ 'C5X จะสนับสนุนกระบวนการทำงานพร้อมกันของ circular buffer ด้วยการควบคุมจากผู้ใช้ โดยที่ตัวเก็บค่าเริ่มต้น 2 ตัวของ circular buffer (CBSR1 และ CBSR2) ชี้ตำแหน่งข้อมูลมีค่า 16 บิท เก็บค่าเริ่มต้นของ circular buffer ส่วนตัวเก็บค่าตัวสุดท้าย 2 ตัวของ circular buffer(CBER1 และ CBER2) ชี้ตำแหน่งสุดท้าย 2 ตัวของ circular buffer และ circular buffer control register (CBCR) ควบคุมการทำงานของ circular buffer

3.6.5 Block Move Address Register (BMAR)

เป็นตัวเก็บค่าการย้ายตำแหน่งข้อมูลมีค่า 16 บิท เก็บค่าที่อยู่ที่ใช้ในการย้ายกลุ่มของข้อมูลและกระบวนการของ การคูณ/การสะสม

3.6.6 Repeat Registers (RPTC,BRCR,PASR,และ PAER)

เป็นรีจิสเตอร์ที่ใช้เก็บจำนวนครั้งในการกระทำคำสั่งซ้ำหนึ่งคำสั่ง ค่าในรีจิสเตอร์นี้จะถูกโหลดด้วยคำสั่ง RPT และ RPTZ

3.6.7 Interrupt Register (IMR,IFR)

interrupt mask register(IMR) ใช้ในเวลาที่ต้องการทำเครื่องหมายกับการขัดจังหวะเฉพาะในมาส์กอะเบิลอินเทอร์รัพต์ ส่วนอินเทอร์รัพต์เฟลกรีจิสเตอร์ (IFR) เป็นตัวบอกถึงสถานะต่างๆของการอินเทอร์รัพต์

การกำหนดค่าใน IMR

15.....9	8	7	6	5	4	3	2	1	0
Reserved	$\overline{\text{INT 4}}$	TXNT	TRNT	XINT	RINT	TINT	$\overline{\text{INT3}}$	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$

รูปที่ 3.7 แสดงค่าต่างๆที่เก็บใน IMR

3.6.8 Global Memory Allocation Register (GREG)

global memory allocation register จะเป็นตัวจัดสรรส่วนที่ว่างให้เป็นหน่วยความจำโกบอล

3.6.9 Dynamic Bit Manipulation Register (DBMR)

ใช้เชื่อมกับ PLU เพื่อเป็นไดนามิกมาส์กรีจิสเตอร์

3.6.10 Temporary Register (TREG0,TREG1,TREG2)

TREG0 จะได้เก็บค่าของการคูณในกระบวนการคูณ สามารถนำมาใช้ได้ในด้วยคำสั่ง LT ,LTA , LTD , LTP , LTS , SQRA , SQRS , MAC , MACD , MADS , and MADD ในส่วน TREG1 จะเก็บค่า dynamic shift count และ TREG2 เก็บค่า dynamic bit address ของคำสั่ง BITT

3.6.11 Processor Mode Status Register (PMST)

เป็นตัวควบคุมลักษณะการใช้งานหน่วยความจำของ 'C5x การกำหนดค่าใน PMST

15.....11	10	9	8	7	6	5	4	3	2	1	0
IPTR	*0*	*0*	*0*	*AVIS*	*0*	*OVLY*	*RAM*	*MP/$\overline{MC}$*	*NDX*	*IRM*	*BRAT*

รูปที่ 3.8 แสดงค่าที่เก็บใน PMST

3.6.12 Serial Port Register (DRR,DXR,SPC)

SPC จะบรรจุส่วนควบคุมและสถานะต่างๆของพอร์ตอนุกรม ใน TDM Serial Port Register,(TRCV , TDXR , TSPC , TCSR , TRTA , TRAD)

3.6.13 Time Registers (TIM,PRD,TCR)

การใช้งานในส่วนของไทม์เมอร์จะเกี่ยวข้องกับรีจิสเตอร์ 3 ตัวด้วยกัน โดย TIM รีจิสเตอร์จะเก็บจำนวนที่นับได้ในปัจจุบัน PRD รีจิสเตอร์บอกถึงคาบเวลาของไทม์เมอร์ และ TCR (Timer Control Register) เป็นตัวควบคุมการทำงานของไทม์เมอร์

รีจิสเตอร์ควบคุมไทม์เมอร์ (TCR)

15-12	11	10	9-6	5	4	3-0
Resered	SOFT	FREE	PSC	TRB	TSS	TDDR

รูปที่ 3.9 แสดงค่าต่างๆที่เก็บอยู่ใน TCR

3.6.14 Software Wait-State Registers (PDWSR,IOWSR,CWSR)

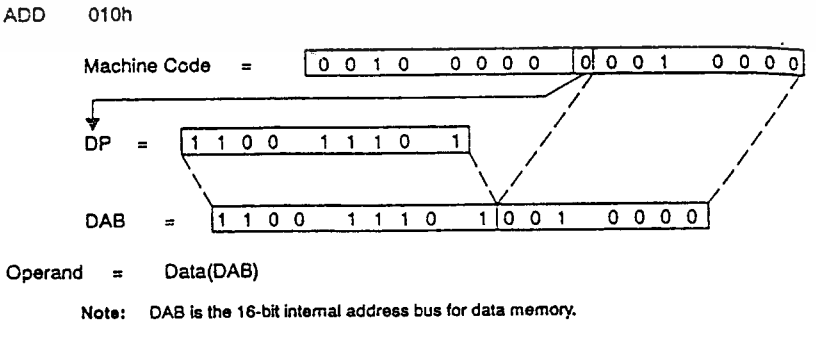
PDWSR เป็นตัวนับค่าสถานะการรอสำหรับ 16 K ทั้ง 4 บล็อก ของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล IOWSR เก็บค่าการนับสถานะการรอของที่ว่าง I/O ทั้ง 16 ส่วน CWSR เป็นรีจิสเตอร์ควบคุมไว้กำหนดระยะการรอว่าจะเป็น (0, 1, 2 หรือ 3) หรือ (0,1,3 หรือ 7)

3.7 โหมดการอ้างอิงถึงหน่วยความจำ(Memory Addressing Mode)

การอ้างอิงแอดเดรสสามารถอ้างได้ 64K สำหรับหน่วยความจำโปรแกรมและ 96K สำหรับหน่วยความจำข้อมูลซึ่ง 'C50 มีวิธีการอ้างอิงแอดเดรส ได้

3.7.1 Direct Addressing Mode

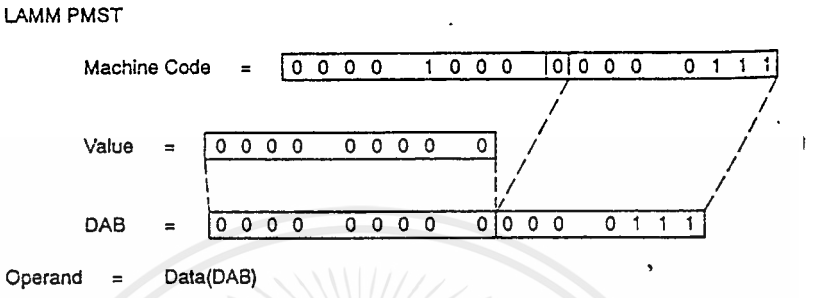
โอเปอร์แรนด์จะถูกเฟชมาจากพื้นที่ที่เกี่ยวกับหน่วยความจำข้อมูล เช่น บัสข้อมูล การคำนวณวิธีนี้จะนำค่าใน DP และ 7 บิทต่ำของคำสั่งมารวมกัน



รูปที่ 3.10 แสดงตัวอย่างการอ้างอิงหน่วยความจำแบบโดยตรง

3.7.2 Memory - Mapped Addressing Mode

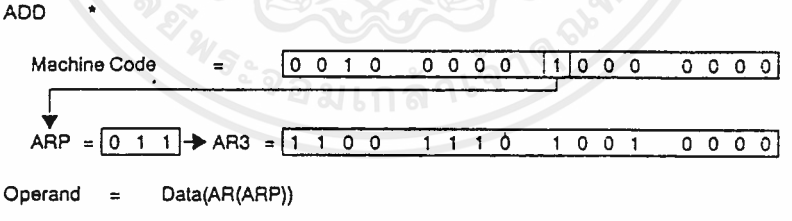
คล้ายกับ direct addressing mode แต่ค่า 9 บิทสูงของค่าแอดเดรสที่คำนวณได้จะถูกบังคับให้เป็น 0 แทนที่จะโหลดค่าจาก DP หรือจากรีจิสเตอร์ช่วย



รูปที่ 3.11 แสดงตัวอย่างการอ้างอิงหน่วยความจำแบบ Memory - Mapped

3.7.3 Indiirect Addressing Mode

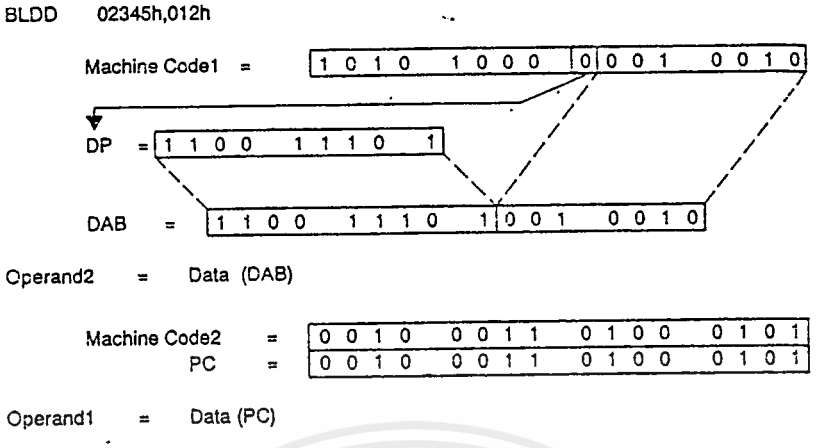
รีจิสเตอร์ชั่วคราว 16 บิท ที่ถูกเลือก AR(ARP) จะเป็นตัวกำหนดค่าแอดเดรสของหน่วยความจำข้อมูลผ่านทาง AFB



รูปที่ 3.12 แสดงตัวอย่างการอ้างอิงหน่วยความจำแบบโดยอ้อม

3.7.4 Long Immidiate Addressing Mode

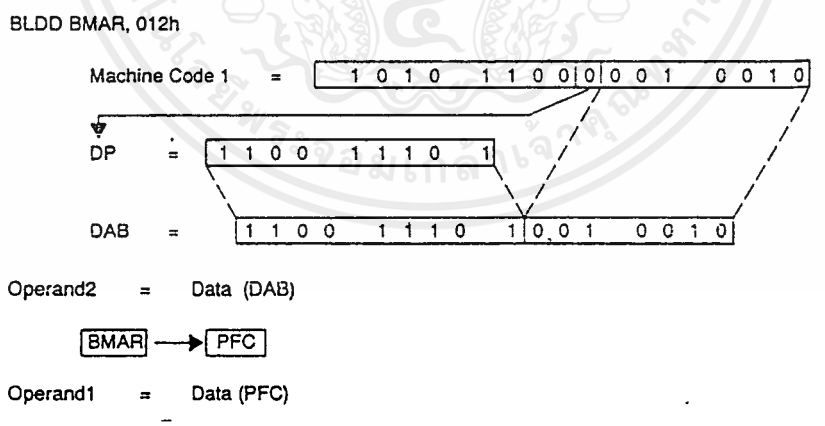
แอดเดรสจะถูกกำหนดด้วยเวิร์ดที่สองของคำสั่ง ในกรณีนี้ PAB จะถูกใช้ในการพรีเฟช (prefetch) PFC(prefetch counter) จะถูกเก็บลง MCS(microcall stack) และค่าตำแหน่งจะถูกโหลดลงใน PFC



รูปที่ 3.13 แสดงตัวอย่างการอ้างอิงหน่วยความจำแบบ Long Immidiate

3.7.5 Registered Block Memory Addressing Mode

คล้ายกับ long immediate addressing mode แต่จะมาจาก BMAR รีจิสเตอร์



รูปที่ 3.14 แสดงตัวอย่างการอ้างอิงหน่วยความจำแบบ Registered Block Memory

3.8 Global Memory

เป็นการใช้ memory ร่วมกันของโปรเซสเซอร์มากกว่า 1 ตัว เมื่อมีการใช้ Global Memory จะทำให้มีการแบ่งกันระหว่างหน่วยความจำที่เป็น Global และ Local โดยที่ส่วนที่เป็น local จะใช้ในส่วนที่ของตัวโปรเซสเซอร์เองและในส่วนที่เป็น global เป็นส่วนที่ใช้ในการติดต่อกับโปรเซสเซอร์อื่นๆ ทำให้เป็นการง่ายในการใช้ข้อมูลร่วมกันในการประมวลผลแบบหลายโปรเซสเซอร์ โดยที่จะควบคุมการทำงานของ global memory จาก GREG register มีรายละเอียดดังตารางที่ 3.5

ตารางที่ 3. 5 แสดงลักษณะของ Global Data Memory

GREG Value	Local Memory		Global Memory	
	Range	#Word	Range	#Words
00000XX	0h-0FFFFh	65,536	-	0
10000000	0h-0FFFFh	32,768	08000h-0FFFFh	32,768
11000000	0h-0BFFFh	49,152	0C000h-0FFFFh	16,384
11100000	0h-0DFFFh	57,344	0E000h-0FFFFh	8,192
11110000	0h-0EFFFh	61,440	0F000h-0FFFFh	4,096
11111000	0h-0F7FFh	63,488	0F800h-0FFFFh	2,048
11111100	0h-0FBFFh	64,512	0FC00h-0FFFFh	1,024
11111110	0h-0BFFh	65,024	0FE00h-0FFFFh	512
11111111	0h-0EFFh	65,280	0FF00h-0FFFFh	256

3.9 Input/output Space

'C50 สามารถให้ I/O port จากภายนอกใช้ตำแหน่งข้อมูลของหลาย I/O ports ไว้ที่ตำแหน่งเดียวกัน ทั้ง65536 พอร์ตสามารถที่จะใช้เป็น IN หรือ OUT ก็ได้ ดังตัวอย่าง

```
IN      DAT7 , 0FFFEh

OUT     DAT7 , 0FFFFh
```

3.10 อุปกรณ์เสริม (Peripherals)

TMS32C50 มีการอินเตอร์เฟสกับอุปกรณ์ภายนอกได้ทั้งหมด 7 วิธี ได้แก่

1. พอร์ตอนุกรม (serail port)
2. พอร์ตอนุกรมแบบแบ่งเวลา (TDM serail port)
3. ไทม์เมอร์ (Timer)
4. รีจิสเตอร์ใช้โปรแกรมสภาวะการรอ (software-programable wait state)
5. พอร์ตติดต่อภายนอก (I/O port)
6. รีจิสเตอร์หารสัญญาณนาฬิกา (divide-by-one clock)
7. XF และ BIO

ซึ่งทั้งหมดจะควบคุมผ่านรีจิสเตอร์ควบคุม โดยการใช้วิธีการอ้างหน่วงความจำในพจ 0 ตามตารางที่ 3.4 ซึ่งเป็นการกำหนดค่าต่างของรีจิสเตอร์ควบคุมเก็บไว้ในหน่วยความจำ และในการใช้อุปกรณ์เสริมต่างส่วนที่สำคัญที่มีส่วนในการควบคุมการใช้งานอุปกรณ์เสริมให้ทำงานได้ตามความต้องการก็คือ อินเทอร์รัพต์รีจิสเตอร์ ซึ่งมีรายละเอียดดังที่จะกล่าวต่อไป

3.10.1 อินเทอร์รัพต์

ในอุปกรณ์ 'C50 สามารถมี 4 makable user interrupts (INT1- INT4) จากภายนอก ในการอินเทอร์รัพต์การทำงานของโปรเซสเซอร์ มี 1 นอนมาส์กอะเบิลอินเทอร์รัพต์ (NMI) การอินเทอร์รัพต์จากภายในสามารถทำได้จาก serial port (RINT และ XINT)

TSM320C50 มีอินเทอร์รัพต์ทั้งหมด 16 อินเทอร์รัพต์ (INT16-INT1) แต่จะไม่ใช้พร้อมกันทั้งหมด

การรีเซตจะเป็นแบบ nonmarkable external interrupt ซึ่งจะเกิดขึ้นเมื่อมีการรีเซต ดังนี้

1. CNF = 0 หมายถึง การใช้หน่วยความจำข้อมูลบนชิพ
2. PC = 0000H
3. INTM = 1 และ IFR = 0 ทำให้อินเทอร์รัพต์ทุกตัวไม่ทำงาน
4. บิทสถานะ (status bit) จะเป็นดังนี้

   0 ---> OV 1 ---> XF 1 ---> SXM 0 ---> PM 1 ---> HM

   0 ---> BRAI 0 ---> TRM 0 ---> NDX 0 ---> CENM1 0 ---> CENM2

   0 ---> IPTR 0 ---> OVLY 0 ---> AVIS 0 ---> RAM 0 ---> BIG

   0 ---> CNF 1 ---> INTM MP/MC(pin) ---> PMST(MP/MC) 1 ---> C
5. GREG = 000000
6. RPTC = 00
7. IACK จะแสดงสภาวะการเก็บการรีเซต

ในการควบคุมการอินเทอร์รัพต์จะมี IFR (Interrupt Flag Register) แสดงสภาวะการอินเทอร์รัพต์และ IMR(Interrupt Mask Register) แสดงการ mask การอินเทอร์รัพต์ ดังรูปที่ 3.15

รีจิสเตอร์อินเทอร์รัพต์แฟลก

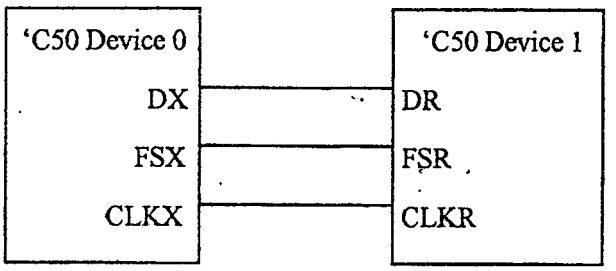
15 – 9	8	7	6	5	4	3	2	1	0
Reserved	$\overline{INT4}$	TXNT	TRNT	XINT	RINT	TINT	$\overline{INT3}$	$\overline{INT2}$	$\overline{INT1}$

ในการกำหนดให้ใช้อินเทอร์รัพต์จะต้องทำการเคลียร์ที่ INTM บิทด้วย

รูปที่ 3.15 รีจิสเตอร์อินเทอร์รัพต์แฟลก

3.10.2 Serial port

Serial port ของ TMS320C50 จะเป็นแบบฟลูดูเพล็ก คือ สามารถรับและส่วภายในเวลาเดียวกัน ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกอื่นๆ ได้ ไม่ว่าจะเป็น CODEC A/D หรือ ระบบอื่นๆ ในการต่อขาต่างๆ จะเป็นตามรูปที่ 3.16



รูปที่ 3.16 แสดงการต่อพอร์ตอนุกรมกับพอร์ตภายนอก

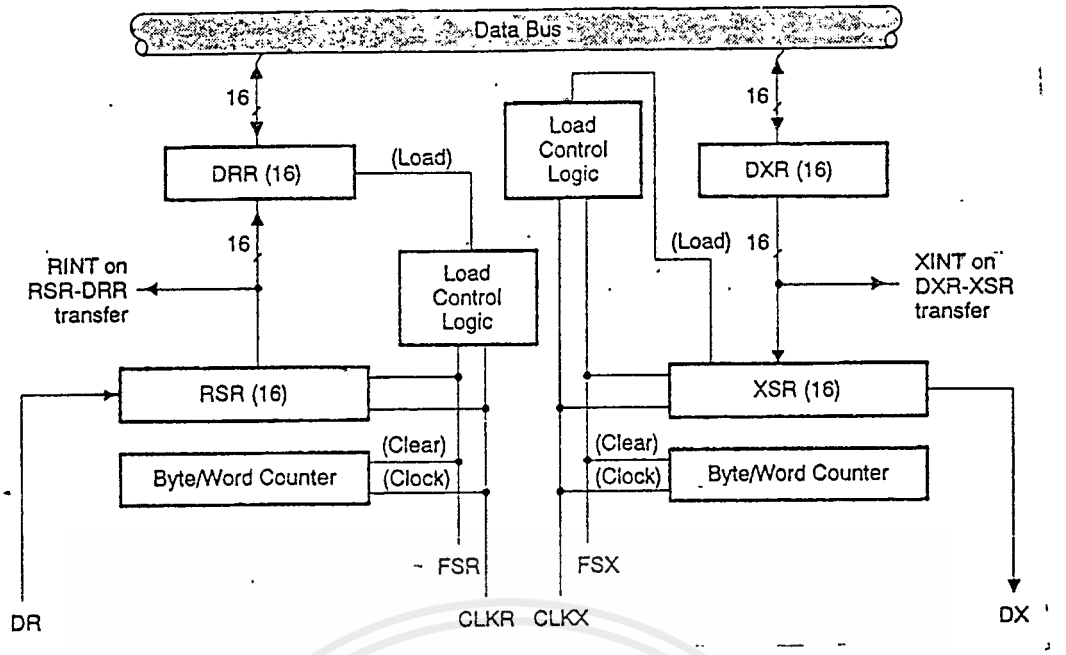
ตารางที่ 3.6 แสดงขาสัญญาของพอร์ตอนุกรม

ขาสัญญาณ	ความหมาย
CLKX	สัญญาณนาฬิกาที่ใช้ในการส่ง
CLKR	สัญญาณนาฬิกาที่ใช้ในการรับ
DX	สัญญาณข้อมูลอนุกรมในการส่ง
DR	สัญญาณข้อมูลอนุกรมในการรับ
FSX	สัญญาณการซิงโครไนซ์เฟรมการส่ง
FSR	สัญญาณการซิงโครไนซ์เฟรมการรับ

ในการควบคุมพอร์ตอนุกรมจะมีรีจิสเตอร์ที่จำเป็นตามตารางที่ 3.7

ตารางที่ 3.7 แสดงการควบคุมพอร์ตอนุกรมผ่านทางรีจิสเตอร์พอร์ตอนุกรม

รีจิสเตอร์	ความหมาย
SPC	รีจิสเตอร์ควบคุม
DXR	รีจิสเตอร์ส่งข้อมูล
DRR	รีจิสเตอร์รับข้อมูล
XSR	รีจิสเตอร์เลื่อนการส่ง
RSR	รีจิสเตอร์เลื่อนการรับ

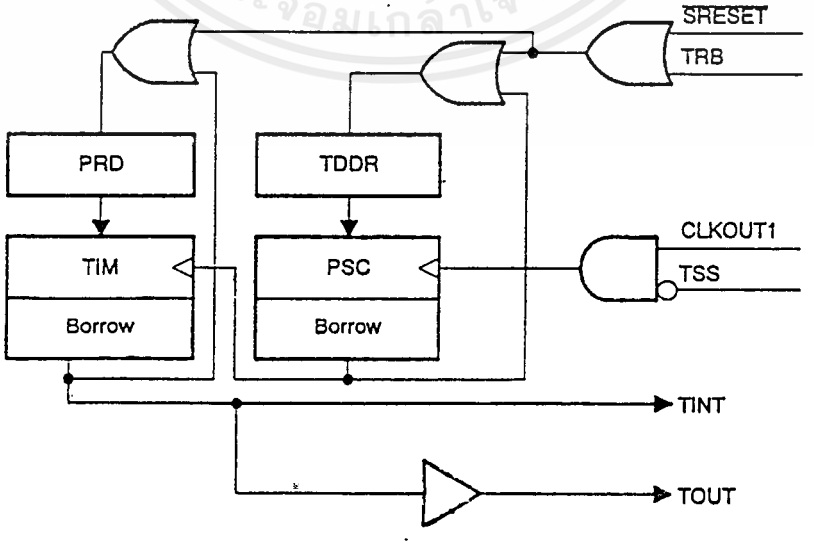


รูปที่ 3.17 แสดงบล็อกไดอะแกรมการทำงานของพอร์ตอนุกรม

ในการส่งข้อมูลทำได้โดยการเขียนข้อมูลลงใน DXR (Data transmit register) โดย XSR (Transmit Shift Register) จะเป็นตัวเลื่อนบิทตามสัญญาณนาฬิกาจนครบไปตามขา DX ส่วนการรับข้อมูลและรับทางขาDR เข้ามาที่ RSR(Receive Shift Register) เมื่อครบแล้วจะทำการอินเทอร์รัพต์บอกให้ทราบ แล้วสามารถ รับข้อมูลโดยการอ่านข้อมูลจาก DRR (Data Receive Register)

3.10.3 ไทม์เมอร์ (Timer)

ในการใช้ไทม์เมอร์ของ TMS320C50 ซึ่ งจะสามา รถตั้ งคาบเวลาของสัญญาณนาฬิกาได้โดยสัญญาณจะออกที่ขา Tout ของ TMS320C50



รูปที่ 3.18 บล็อกไดอะแกรมของไทม์เมอร์

ซึ่งการคำนวณคาบเวลา จะเป็นไปตามสูตร

$$T = \frac{1}{t_e \times (TDDR + 1) \times (PRD + 1)}$$

ซึ่ง t_e = คาบเวลาของสัญญาณ นาฬิกาที่จ่ายให้ที่ขา CLKOUT1

TDDR = Timer Divide Down Ratio

PRD = Period Register ซึ่งเป็นรีจิสตอร์ที่สามารถใส่ค่าคาบเวลาได้

ในการทำงานจะต้องมีการใส่ค่าต่างๆ ลงบน TCR (Time Contol Register) ดังรูปที่ 3.19

รีจิสเตอร์ควบคุมไทม์เมอร์ (TCR)

15-12	11	10	9-6	5	4	3-0
Resered	SOFT	FREE	PSC	TRB	TSS	TDDR

รูปที่ 3.19 แสดงการกำหนดค่าในรีจิสเตอร์ควบคุมไทม์เมอร์

3.10.4 การเขียนโปรแกรมและคำสั่ง

การกำหนดสถานะเริ่มต้นสำหรับ TMS320C50

ทุกครั้งที่มีการใช้งาน TMS320C50 จำเป็นต้องมีการกำหนอสถานะเริ่มต้น (intialize) ตัวซีพียู ก่อนเสมอ ซึ่งต้องทำการเตรียมสภาวะของ โปรเซสเซอร์ให้พร้อม โดยปกติจะทำไว้เมื่อเกิดการรีเซต ซึ่งการรีเซตของ TMS320C50จะเกิดขึ้นเมื่อ ขา $\overline{RS}$ อยู่ในสถานะต่ำ ซึ่งจะทำให้ IPTP (interrupt poniter) ซึ่งอยู่ใน PMST (processor mode status register) เกิดการเคลียร์ ทำให้เกิดการmapเวกเตอร์ ไปที่ page 0 ในการเกิด interrupt นั้นเมื่อ รีเซตจะทำให้ disable

ในการกำหนดสถานะเริ่อต้นสำหรับโปรเซสเซอณ์นั้นจะประกอบด้วย

- การ map หน่วยความจำและ รีจิสเตอร์ควบคุม
- การเซตโครงสร้างของ อินเตอร์รัพต์
- โหมดควบคุมต่างๆ (OVM, SXM , PM, AVIS , NDX, TRM)
- การควบคุมหน่วยความจำ (RAM, OVLY < CNF)
- กำหนด รีจิสเตอร์ช่วย (Auxiliary register)และ ตัวชี้รีจิสเตอร์ช่วย

- กำหนดตัวชี้เพจข้อมูล (Data page poniter:DP)
- การกำหนดค่าใน PMST (Processor mode status register)

การกำหนดค่าใน PMST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPTR					*0*	*0*	*0*	*AVIS*		*0*	*OVLY*	*RAM*	*MP/ $\overline{MC}$*	*NDX*	*IRM*	*BRAT*

รูปที่ 3.20 แสดงการกำหนดค่า ใน PMST

ค่า IPTP เราจะกำหนดที่ IPTP = 100008 หรือเป็นเพจที่ 1 ซึ่งตรงกับโปรแกรมตำแหน่ง 0800h ส่วนการกำหนดการใช้ RAM ภายใน จะต้องใช้ RAM 9K ซึ่งอยู่ในชิป กำหนด RAM OVLY เป็น "1" RAM ที่ใช้จะเป็น SARAM โดยที่หน่วยความจำโปรแกรมอยู่ที่ 0800h-2BFFh และหน่วยความจำอยู่ที่ 0800h-28FFh

ในส่วนของการกำหนดอินเตอร์เวกเตอร์รัพต์เวกเตอร์ (interrupt vectcr) ในโครงงานนี้ มีการกำหนดดังนี้

- Reset 0800h การรีเซตภายนอก
- $\overline{INT1}$ 0802h สัญญาณอินเตอร์รัพต์จากผู้ใช้ภายนอกหมายเลข 1
- $\overline{INT2}$ 0804h สัญญาณอินเตอร์รัพต์จากผู้ใช้ภายนอกหมายเลข 2
- $\overline{INT3}$ 0806h สัญญาณอินเตอร์ รั พต์จากผู้ใช้ภายนอกหมายเลข 3
- TINT 0808h สัญญาณอินเตอร์รัพต์ไทมเมอร์ภายใน
- RINT 080Ah สัญญาณอินเตอร์รัพต์สำหรับการรับค่าจากพอร์ตอนุกรม
- XINT 080Ch สัญญาณอินเตอร์รัพต์สำหรับการส่งค่าจากพอร์ตอนุกรม
- $\overline{INT4}$ 0812h สัญญาณอินเตอร์รัพต์จากผู้ใช้ภายนอก
- TRAP 0822h คำสั่ง ซอผท์แวร์ แทรป (Software trap instuction)
- NMI 0824h นอนมาสเคเบิ้ลอินเตอร์รัพต์ (Nonmaskable interrupt)

ในโครงงานนี้เรากำหนด interrupt จาก IMR (interrupt mask register) ซึ่งจะกำหนดดังนี้

การกำหนดค่าใน IMR

15……..9	8	7	6	5	4	3	2	1	0
Reserved	$\overline{\text{INT 4}}$	TXNT	TRNT	XINT	RINT	TINT	$\overline{\text{INT3}}$	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$

รูปที่ 3.21 แสดงการกำหนดค่าใน PMST

ก่อนการใช้อินเตอร์รัพต์จะต้องไม่ mask อินเตอร์รัพต์ ตัวนั้นๆ โดยใช้คำสั่ง clrc INTM ส่วน IMM จะแอคทีฟ ที่ " 1 "

<u>ตัวอย่าง</u>

```
setc  INTM              ; กำหนด INT1, XINT
SPLK  # 03IH ,IMR       ; RINT แอคทีฟ
clrc  INTM
```

บทที่ 4

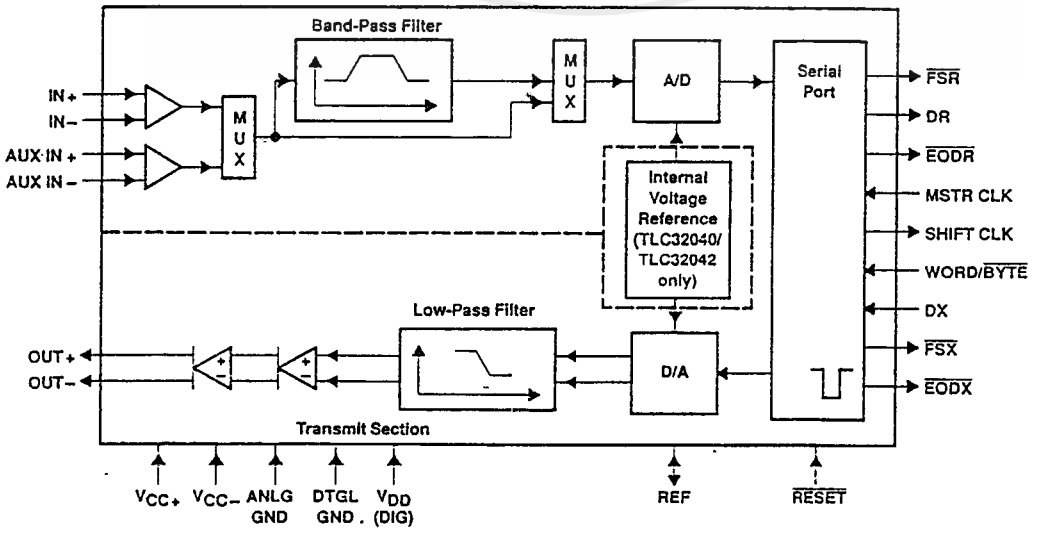
วงจรอินเตอร์เฟสสัญญาณอนาลอก TLC32040

TLC320C40 เป็นชิพซีมอส (CMOS Chip)ที่ใช้เชื่อมต่อกับ TMS320C50

4.1 ลักษณะสำคัญของ TLC320C40 มีดังนี้

- ใช้เทคโนโลยีการผลิต Advanced LinCMOS™ Silicon-Gate Process
- ความละเอียดของADC และ DACเป็น 14 บิท
- สามารถเปลี่ยนอัตราแซมปลิ้งของ ADC และ DAC ได้ถึง 19,200 ครั้ง/วินาที
- มี Switched-Capacitor Input Filter และ Output-Reconstruction Filter
- มีพอร์ตอนุกรมสำหรับติดต่อโดยตรงกับ TMS3211 , TMS320C17 , TMS32020 และ TMS320C25 DSP
- สามารถปรับตัวอัตราการแปลงของ ADC และ DAC ทั้งแบบซิงโครนัส หรือ อะซิงโครนัสโดยใช้ซอฟท์แวร์ควบคุม

4.2 ฟังก์ชันบล็อกไดอะแกรม (Function Block Diagram)



รูปที่ 4 .1 แสดงฟังก์ชันบล็อกไดอะแกรมของ TLCC32040

4.3 ตำแหน่งขาและหน้าที่การทำงานของแต่ละขา

ตารางที่ 4.1 แสดงตำแหน่งขาและหน้าที่การทำงานของแต่ละขา

ชื่อ	หมายเลข	I/O	คำอธิบาย
ANLG GND	17,18		กราวด์อนาลอก(แยกกับกราวด์ดิจิตอล)
AUX IN+	24	I	นอน-อินเวอร์ติงออกซิลลารีอินพุท (Noninvertimg auxiliary Input)
AUX IN-	23	I	อินเวอร์ติง อกซิลลาริอนพุท (Inverting auxiliary Input)
DGTL GND	9		กราวด์ดิจิตอล
DR	5	O	ใช้สำหรับส่ง เอาท์พุท ADC จาก AIC (analog interface circuit)ไปยัง 'C50 ผ่านทางพอร์ตอนุกรม(ต้องซิงค์กับ SHIFT CLK)
DX	12	I	ใช้สำหรับรับอินพุท DAC หรือ คำสั่งการควบคุมจาก 'C50 ซึ่งการส่งผ่านพอร์ตอนุกรมจะต้องซิงค์กับ SHIFT CLK
$\overline{\text{EODR}}$	3	O	สัญญาณหยุดรับข้อมูล (End of data receive) : ในการติดต่อผ่านพอร์ตอนุกรมในโหมดเวิร์ด (word-mode) สัญญาณ $\overline{\text{EODR}}$จะอยู่ในสถานะต่ำ ทันทีเมื่อทั้ง 16 บิท ของเอาท์พุท A/D ได้ถูกส่งจาก AIC ไปยัง 'C50 ซึ่งสามารถใช้สัญญานี้ในการอินเทอร์รัพต์ไมโครโปรเซสเซอร์ ให้ทราบว่าสิ้นสุดการติดต่อแล้ว หรือใช้ สโตรบ และให้รีจิสเตอร์เลื่อนข้อมูลออก (enable external serisl-to-parallel shift register) ก็ได้ แต่ถ้าเป็นโหมดไบต์ (byte-mode) สัญญาณ $\overline{\text{EODR}}$ จะอยู่ในสถานะต่ำ หลังจากไบต์แรกได้ส่งไปยัง 'C50 แล้ว และยังคงรักษาสถานะต่ำจนกระทั่งไบต์ที่สองได้ส่งไปทั้งนี้เพื่อให้รู้ว่าไบต์แรกหรือไบต์ที่สองออกไป
$\overline{\text{EODX}}$	11	O	สัญญานหยุดส่งข้อมูล (End of data transmit) ก็คล้ายกับ $\overline{\text{EODR}}$ ซึ่งจะบอกให้ทรายว่าการติดต่อจาก 'C50ไปยัง AIC นั้นเสร็จแล้วทั้งในโหมดเวิร์ด และโหมดไบต์ ก็คล้ายกับ $\overline{\text{EODR}}$

ชื่อ	หมายเลข	I/O	คำอธิบาย
$\overline{FSR}$	4	O	สัญญาณซิงค์การรับ (Frame sync transmit) : ในการติดต่อทางพอร์ตอนุกรม $\overline{FSR}$ จะมีสถานะต่ำตลอดการส่งจาก AIC ไปยัง 'C50 (โดยส่งผ่านขา DR) ซึ่งบิทแรกที่จะส่งต้องพร้อมอยู่ที่ขาDRก่อน $\overline{FSR}$ จะต่ำ
$\overline{FSX}$	14	O	สัญญาณซิงค์การส่ง (Frame sync transmit) : เมื่อสัญญาณนี้อยู่ในสถานะต่ำ พอร์ตอนุกรม 'C50 จะส่งบิทไปยังAICโดยมาขา DX ในการติดต่ออนุกรมทุกโหมด $\overline{FSR}$ จะมีสถานะต่ำตลอดการส่ง
IN+	26	I	นอน-อินเวอร์ติง อินพุท (Non-inverting input)
IN-	25	I	อินเวอร์ติง อินพุท (Inverting input)
MSTR CLK	6	I	สัญญาณนาฬิกามาสเตอร์ (master clock) จะใช้ในการควบคุมทุกส่วนภายใน AIC ไม่ว่าจะเป็นสัญญาณนาฬิกาเลื่อน (shift clock) สัญญาณนาฬิกาควบคุมฟิลเตอร์ (switched-capacitor filter cock), A/D และ D/A ไทม์มิ่ง
OUT+	22	O	นอน-อินเวอร์ติง เอาท์พุท (Noninverting output)
OUT-	21	O	อินเวอร์ติง เอาท์พุท (Inverting output)
REF	8	I/O	สำหรับ TLC32040 และ TLC32042 แรงดันอ้างอิงภายในจะถูกส่งออกมาที่ขานี้ แต่ถ้าเป็น TLC32040 , TLC32041และ TLC32042 แรงดันอ้างอิงจากภายนอกจะถูกต่อเข้าที่ขานี้
$\overline{RESET}$	2	I	รีเซตจะทำการตั้งค่า TA , TA' , TB , RA , RA' , RB และรีจิสเตอร์ควบคุม (control register)ให้เป็นค่าดีฟอลต์ (default) ซึ่งจะบอกภายหลัง
SHUFT CLK	10	O	สัญญาณนาฬิกาเลื่อนเกิดจากการหารความถี่ของสัญญาณนาฬิกามาสเตอร์ด้วย 4 ซึ่งสัญญาณนี้จะใช้ในการติดต่อทางพอร์ตอนุกรม
VDD	7		ไฟเลี้ยงวงจรดิจิตอล (Digital supply voltage) 5 V $\pm$ 5%
Vcc+	20		ไฟเลี้ยงวงจรอนาลอกด้านบวก 5 V $\pm$ 5%

ชื่อ	หมายเลข	I/O	คำอธิบาย
Vcc-	19		ไฟเลี้ยงวงจรอนาลอกด้านลบ -5 V ± 5%
WORD/ $\overline{BYTE}$	13	O	ขานี้จะทำงานร่วมกับรีจิสเตอร์ควบคุมเพื่อใช้ในการเลือกโหมดการติดต่ออนุกรม ซึ่งมี 4 แบบดังนี้ **การติดต่อแบบ อะซิงโครนัส** <u>ในโหมดไบต์</u> (WORD/ *BYTE* = low) พอร์ตอนุกรมจะติดต่อโดยตรงกับ 'C50 และจะติดต่อ ทีละ 8 บิท 2 ครั้ง ซึ่งมีขั้นตอนการทำงานดังนี้ 1. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะต่ำ 2. 8 บิทแรกถูกส่งออกไป หรือรับเข้ามา 3. $\overline{EODX}$ หรือ $\overline{EODR}$อยู่ในสถานะต่ำ 4. $\overline{FSX}$ หรือ $\overline{FSR}$ ประมาณ 4 สัญญาณนาฬิกาเลื่อน แล้ว อยู่ในสถานะต่ำ 5. 8 บิท ต่อมา(ไบต์ที่สอง) ถูกส่งหรือรับเข้ามา 6. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะสูง 7. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะสูง <u>ในโหมดเวิร์ด</u> พอร์ตอนุกรมจะต่อตรงกับพอร์ตอนุกรมของ'C50 และมีการติดต่อครั้งเดียว 16 บต ซึ่งมีขั้นตอนการทำงานดังนี้ 1. *FSX* หรือ *FSR* อยู่ในสถานะต่ำ 2. 16บิทถูกส่งหรือรับเข้ามา 3. *FSX* หรือ *FSR* อยู่ในสถานะสูง 4. *EODX*หรือ*EODR*อยู่ในสถานะต่ำ

ชื่อ	หมายเลข	I/O	คำอธิบาย
			การติดต่อแบบซิงโครนัส ในการณีนี้ แบนด์พาส ฟิสเตอน์ (bandpsaa fiter) และอัตราการแปลง A/D (A/D conversion timing) จะถูกกำหนดจาก TX เคาน์เตอร์ A , TX เคาน์เตอร์ B และ TA , TA' และ TB แทนส่วนในการติดต่อทั้งในแบบโหมดเวิร์ดและโหมดไบต์นั้นมีขั้นตอนเหมือนกับการติดต่อแบบซิงโครนัส

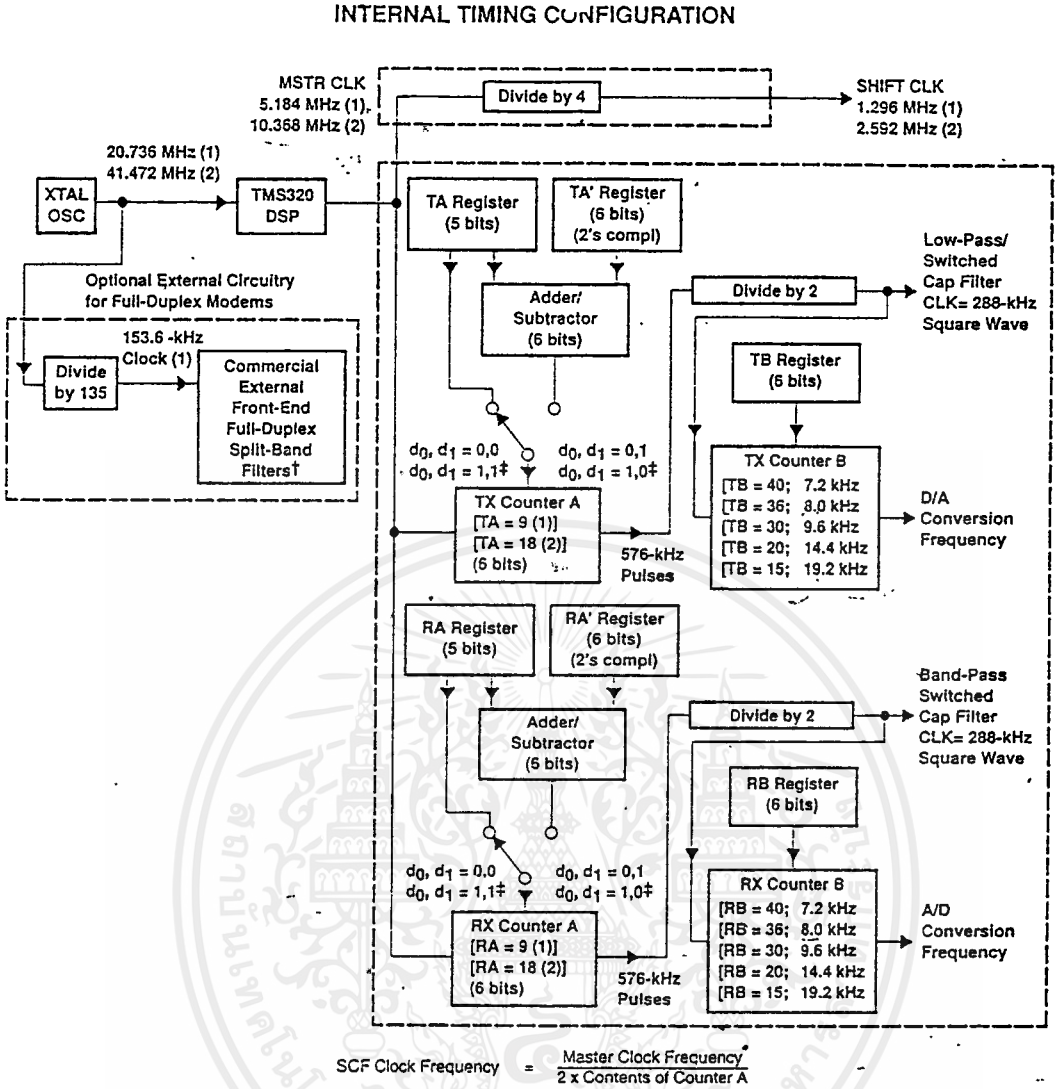
4.4 การทำงานของ TLC32040

4.4.1 อินพุททางอนาลอก (analog input)

สำหรับ อนาลอกอินพุทจะมี 2 กลุ่ม คือ IN+ , IN- และ AUX IN+ , AUX IN- ซึ่งสามารถเลือกใช้กลุ่มใดกลุ่มหนึ่งโดยจะใช้แบบ ดิฟเฟอเรนเชียล (differential) หรือ ซิงเกิล-เอนด์ (simgle-emded) และค่าเกน สำหรับอินพุท IN+ , IN- , AUX IN+ และ AUX IN- สามารถจะใช้โปรแกรมตั้งค่าได้ (มี 3 ค่า คือ 1 , 2 หรือ 4) การเลือกใช้กลุ่มอินพุทใดจะเลือกโดยใช้ซอฟท์แวร์ควบคุม

4.4.2 A/D bandpass filter , A/D bandpass filter clocking และ A/D conversion timing

สำหรับ A/D แบนด์พาส ฟิลเตอร์ (A/D bandpass filter) เราสามารถที่จะเลือกใช้หรือไม่ใช่ก็ได้โดยใช้ ซอฟท์แวร์ควบคุม ความถี่ของสัญญาณนาฬิกาควบคุมฟิลเตอร์ (fliter clock) จะเป็นตัวกำหนด ทรานเฟอร์ฟังก์ชัน (transfer function) ของฟิลเตอร์โดยจะคิดอัตราส่วนจากความถี่ สัญญาณนาฬิกาควบคุมฟิลเตอร์ 28 KHz ที่ความถี่ต่ำที่เริ่มมีลักษณะเป็นความถี่สูงผ่านจะมีความถี่เป็น 300 Hz อัตรการแปลง D/A ก็หาได้จากความถี่ ที่หาร 228 KHz ด้วย RX เคาน์เตอร์ B



รูปที่ 4.2 แสดงไทม์มิงภายในของ TLC32040C

4.4.3 เอาท์พุททางอนาลอก (analog output)

อนาลอกเอาท์พุทจะมีเพาเวอร์แอมพลิไฟร์ (power anplifier) ที่มีเอาท์พุททั้งแบบนอนอินเวอร์ติง (noninverting) และ แบบอินเวอร์ติง (inverting) เนื่องจากมีแอมพลิไฟล์ทำให้เอาท์พุท สามารถขับ ทรานฟอร์เมอร์ชนิดไฮ-บริด (transformer hybrid) หรือ โหลดอิมพีแดนซ์ต่ำได้ โดยใช้แบบดิฟเฟอเรนเชียล หรือ ซิงเกิล-เอน

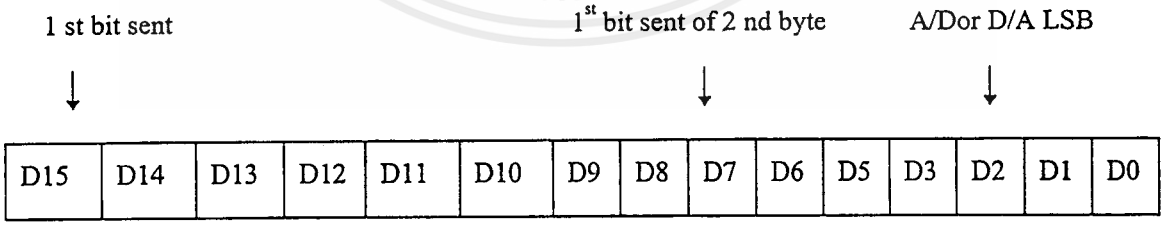
4.4.4 วงจรกรองความถี่ต่ำของ D/A , วงจรกรองความถี่แบบแบนด์พาสของD/A ,สัญญาณนาฬิกาควบคุม วงจรกรองความถี่ต่ำ และอัตรการแปลงสัญญาณดิจิตอลเป็นอนาลอก(D/A lowpass filter , D/A lowpass filter clocking และ D/A conversion timing)

เช่นเดียวกับ A/D ฟิลเตอร์โดยทรานเฟอร์ฟังก์ชันของฟิลเตอณีถูกกำหนดจากอัตราส่วนกัยความถี่ 228 kHz และอัตราการแปลง D/A ก็หาได้จากความถี่ 228 kHz หารด้วยTXเคาน์เตอร์ B

4.4.5 การต่อกลับ (loopback)

การต่อกลับจะให้ผู้ใช้ตรวจสอบวงจรโดย OUT+ และ OUT- จะต่อเข้าภายในกับIN+ และ IN- ดังนั้นบิท DAC(d15-d2)จะถูกส่งไปยังขา DX และเปรียบเทียบกับบิท ADC ที่รับมาจากขา DR ซึ่งโดยปกติจะต้องมีค่าเท่ากัน (ในทางปฏิบัติอาจไม่เท่ากันก็ได้ ในการตรวจสอบ ถ้าใช้ขาIN+ และ IN- สัญญาณภายนอกที่ต่อกับ IN+ , IN- จะไม่มีผลแต่ถ้าใช้ AUX IN+ , AUX IN- สัญญาณภายนอกจะถูกรวมกับ OUT +และ OUT-สำหรับการควบคุมการต่อกลับ จะทำโดยการตั้งค่าในรีจิเตอร์ควบคุม

4.4.6 รูปแบบของบิทใน AIC DR หรือ DX เวิร์ด



รูปที่ 4.3 แสดงรูปแบบบิทใน AIC DR หรือ DX word

4.4.7 รูปแบบของบิทใน AIC DX (AIC DX data word format section)

d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0

รูปที่ 4.4 แสดงรูปแบบการฟอร์แมทใน AIC DX data word

ในรูปแบบการส่งครั้งแรก (Primary DX serial communication protocak (d0-d15)) นั้นเราจะใช้ d1 และ d0 เป็นตัวเลือก ส่วน d2-d15 จะถูกส่งไปยังรีจิสเตอร์ควบคุมการแปลง D/A (D/A comverter register) ซึ่งจะแบ่งการทำงานออกเป็นกรณีต่างๆ ได้ดังนี้คือ

ตารางที่ 4.2 แสดงการกำหนดค่าใน d1 และ d0 ในการทำงานในโหมดต่างๆ ของ AIC

d1	d0	การทำงาน
0	0	จะทำการโหลดค่าในรีจิสเตอร์ TA และ RA ไปยัง TX และ RX เคาน์เตอร์A ตามลำดับ และทำการโหลด TX และ RX เคาน์เตอร์ B ด้วยค่าในรีจิสเตอร์ TB และ RB
0	1	TX และ RX เคาน์เตอร์A จะถูกโหลดด้วยค่า TA + TA' และ RA + RA' แต่จะทำการโหลด TX และ RX เคาน์เตอร์B ด้วยค่าในรีจิสเตอร์ TB และ RB
1	0	TX และ RX เคาน์เตอร์ A จะถูกโหลดด้วยค่า TA - TA' และ RA - RA' แต่จะทำการโหลด TX และ RX เคาน์เตอร์ B ด้วยค่าในรีจิสเตอร์ TB และ RB
1	1	TX และ RX เคาน์เตอร์ A จะถูกโหลดด้วยค่าในรีจิสเตอร์ TA และ RA ส่วน TX และ RX เคาน์เตอร์ B จะถูกโหลดด้วยค่าในรีจิสเตอร์ TB และ RB หลังจากนั้นก็จะรอเวลาประมาณ 4 สัญญาณนาฬิกาเลื่อน ก็จะทำการส่งครั้งที่สอง (secondary communication) เพื่อไปโปรแกรม AIC ให้ทำงานตามต้องการ

4.4.8 รูปแบบของบิทในการส่งครั้งที่สอง (Secondary DX serial communication protocol)

ตารางที่ 4.3 แสดง secondary DX serial communication protocal

XX	to TA register	X X	to RA register	0 0	d13 และ d6 เป็น MSR (เป็นบวก)
X	to TA' register	X	to RA' register	0 1	d14 และ d7 เป็นบิทแบบ 2'S
X	to TB register	X	to RB register	1 0	d14 และ d7 เป็น MSB (เป็นบวก)
X X X X X X X d7 d6 d5 d4 d3 d2 Control Register				1 1	d2 = 0/1 ลด/เพิ่มตัวกรองผ่านความถี่ d3 = 0/1 disable/enableฟังก์ชันloopback d4 = 0/1 disable/enableขาAUX IN+ และขา AUX IN- d5 = 0/1 บิทกำหนดการรับส่งแบบ อะซิงโครนัส/ซิงโครนัส d6,d7 เป็นบิทควบคุม gain	

4.4.9 ฟังก์ชัน Reset

ฟังก์ชันรีเซตจะเป็นการตั้งค่าอัตราการเปลี่ยนแปลง A/D และD/A ให้เป็น 8KHz ถ้าใช้สัญญาณ นาฬิกามาสเตอร์ 5.184 MHz โดยค่าในรีจิสเตอร์ จะถูกตั้งค่าเริ่มต้นเป็น

TA = RA = 9

TA' = RA' = 1

TB = RB = 24

และบิทในรีจิสเตอร์ควบคุมเป็น d7 = 1 , d6 = 1 , d5 = 1 , d4 = 0 , d3 = 0 , d2 = 1

4.4.10 ข้อบังคับในการกำหนด ค่ารีจิสเตอร์ของ AIC

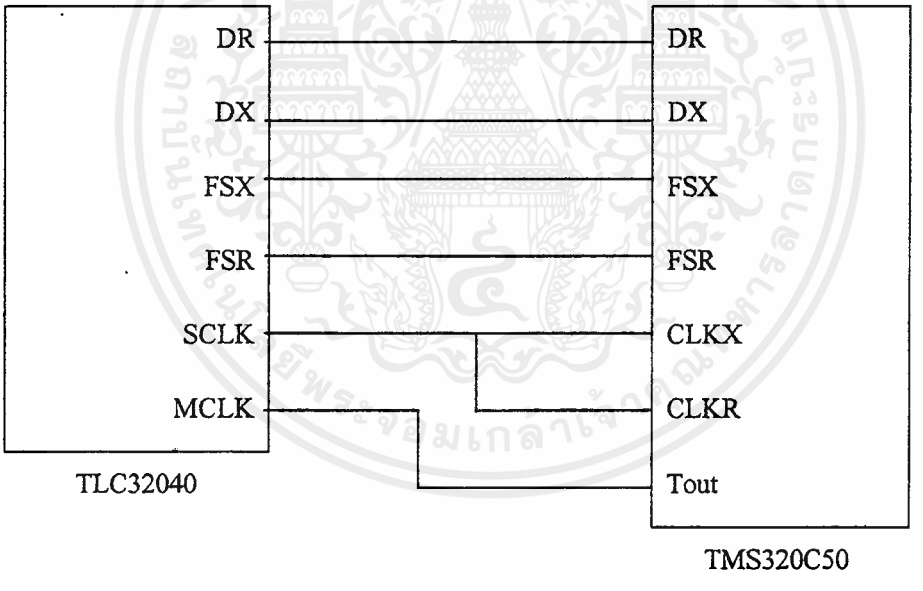
ตารางที่ 4.4 แสดงเงื่อนไขต่างๆที่มีผลต่อการตอบสนอง AIC

เงื่อนไข	การตอบสนองของAIC
1. ถ้า TA + TA' = 0 หรือ 1 หรือ TA - TA' = 0 หรือ 1	จะทำการโหลดค่าในรีจิสเตอร์ TA ลงใน TX เคาน์เตอร์ A แทน
2. ถ้า TA+TA' < 0	MODULO 64 จะถูกใช้เพื่อให้แน่ใจว่า ค่าที่เป็นบวกถูกโหลดไปไว้ที่ TX เคาน์เตอร์ A แทน เช่น TA + TA' + 40 HEX โหลดไปไว้ที่ TX เคาน์เตอร์ A
3. ถ้า RA + RA' = 0 หรือ 1 RA + RA' = 0 หรือ 1	จะทำการโหลดค่ารีจิสเตอร์ RA ลงในRX เคาน์เตอร์A
4. ถ้า RA + RA' = 0 หรือ 1	MODULO 64 ถูกใช้เหมือนข้อ 2 เพียงแต่โหลดไปที่ RX เคาน์เตอร์ A
5. ถ้า TA = 0 หรือ 1 หรือ RA = 0 หรือ 1	AIC หยุดทำงาน
6. ถ้า TA < 4 ในโหมดเวิร์ด TA < 5 ในโหมดไบต์ RA < 4 ในโหมดเวิร์ด RS < 5 ในโหมดไบต์	การติดต่อทางพอร์ตอนุกรมของAIC ไม่ทำงาน

7. ถ้า TB = 0 หรือ 1	โหลดค่า TBใหม่ ด้วยค่า 24 HEX
8. ถ้า RB = 0 หรือ 1	โหลดค่า RBใหม่ ด้วยค่า 24 HEX
9. ถ้า AIC และ DSP ไม่สามารถติดต่อกันได้	จะเก็บค่า DAC Output ไว้

4.5 การอินแตอร์เฟสระหว่างTMS320C50 dy[TLC32040

4.5.1 การใช้อินแตอร์ของพอร์ตอนุกรม



รูปที่ 4.5 แสดงการเชื่อมต่อระหว่าง TLC32040 กับTM320C50

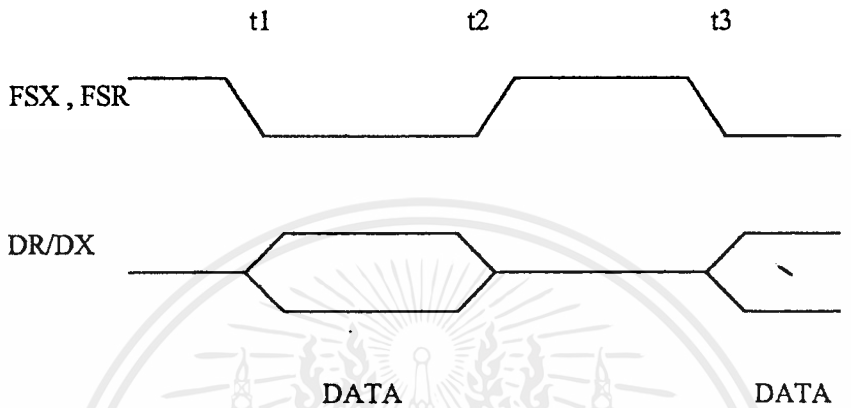
4.5.2 TMS320C50

การส่ง โดยการเขียน DXR จะเป็นการขออินเทอร์รัพต์ของ XINT(โดยไม่มีการมาร์ค)

การรับ เมื่อ RSR -----> DRR เต็มจะเกิดอินเทอร์รัพต์ของ RINT (โดยไม่มีการมาร์ค)

4.5.3 TLC32040

เนื่องจากการส่งระหว่าง TLC กับ TMS เป็นแบบซิงโครนัส TLC32040 จะส่งและรับทุกๆเฟรม ซิงโครนัส (Frame Synchonous)



รูปที่ 4.6 แสดงช่วงเวลาในการส่งและรับข้อมูลระหว่าง TLC32040 กับ TMS320C50

เนื่องจากความถี่แซมปลิ้งสูงสุดเท่ากับ 19.2 KHz เพราะฉะนั้นข้อมูลต้องมาก่อน (ภายใน t2 - t1) t2 - t1 ต้องไม่น้อยกว่า 52.08 us ข้อมูลที่มาจาก TLC32040 แล้วทำการประมวลผลทางคณิตศาสตร์แล้วส่งกลับมาที่ TLC32040 อีกครั้ง ต้องใช้เวลาน้อยกว่า 1/fs จึงจะสามารถได้สัญญาณที่มีความถี่เข้าเท่ากับความถี่ออก เนื่องจาก t3 - t1 ช่องของข้อมูล (16bit) ใช้เวลาประมาณ (1/sclk) x 16 ประมาณเท่ากับ 6.17 us เพราะฉะนั้น 6.17 us เป็นเวลาที่ใช้ในการส่ง 1 ข้อมูล จะต้องใช้เวลาเหลือประมวลผล (52.08 - 6.17) = 45.91 us หรือประมาณ (45.91 us)/(50 ns ต่อ 1 คำสั่ง) = 9.18 คำสั่ง

<u>สรุป</u> สามารถเขียนโปรแกรมถึงกว่า 900 คำสั่ง เพื่อประมวลผลข้อมูล 1 ข้อมูล ก่อนที่จะเกดการอินเทอร์รัพต์อีกครั้ง แต่เราใช้ไดถึงเพราะ fs มักจะน้อยกว่า 19.2 KHz

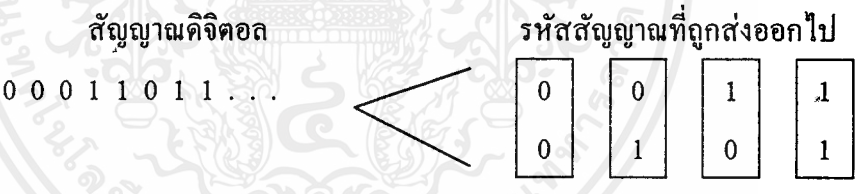
บทที่ 5

Quadrature Phase shift Keying

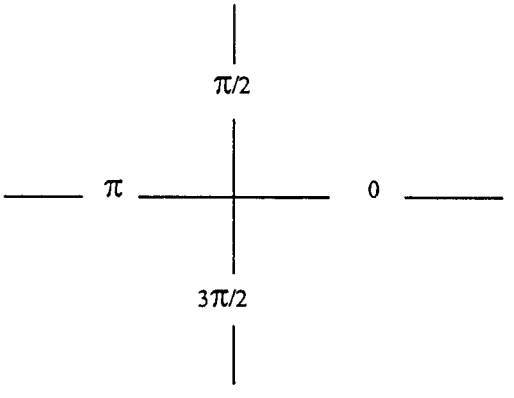
ในวิธีการ QPSK สัญญาณ 2 บิทจะถูกส่งไปพร้อมๆกัน โดยลำดับสัญญาณดิจิตอลที่ส่งถูกแบ่งเป็นบล็อก 2 บิทและแต่ละรหัสบล็อกประกอบด้วย 2 บิทถูกส่งไปพร้อมๆกัน ในกรณีนี้จะมีการรวมรหัส 4 คู่ด้วยกันคือ 00 01 10 และ 11 คู่ของรหัสทั้ง 4 คู่นี้จะเทียบกับ initial phase 4 เฟส ของสัญญาณ QPSK

รูปที่ 5.1 แสดงความสัมพันธ์ระหว่างรหัส และ initial phase ใน Gray code ซึ่งใช้ใน QPSK ดังนี้

- เฟส 0 แทนรหัส 00
- เฟส $\Pi/2$ แทนรหัส 01
- เฟส Π แทนรหัส 11
- เฟส $3\Pi/2$ แทนรหัส 10



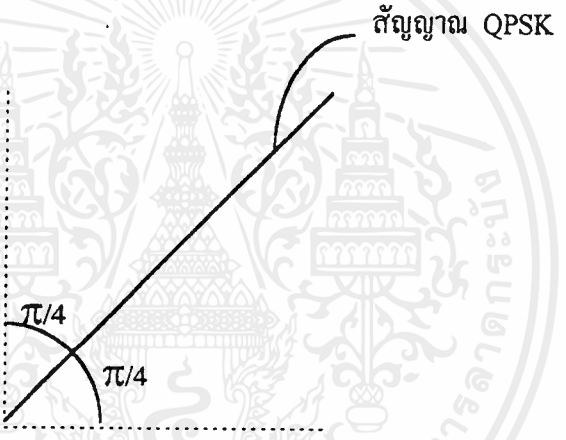
รูปที่ 5.1 แสดงรหัส 4 คู่ของสัญญาณ QPSK



รูปที่ 5.2 Gray Code ที่ใช้ใน QPSK

จะเห็นว่ารหัสระหว่างสองเฟสข้างเคียงหนึ่งในสองบิทจะเหมือนกันในทุกๆกรณีดังนั้นโดยการใช้ Gray Code จะมีเพียงบิทเดียวที่เกิดผิดพลาด แม้ว่าสัญญาณที่รับได้ถูกดีมอดูเลทผิดไปเป็นเฟสข้างเคียง

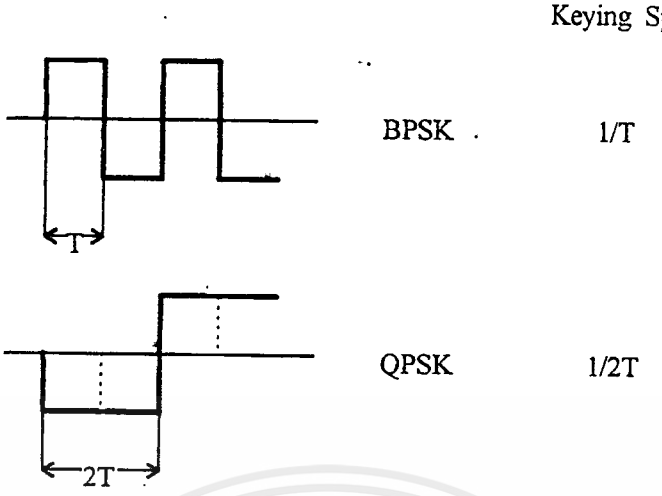
เมื่อพิจารณาหลักการของการดีมอดูเลทสัญญาณ โดยเวคเตอร์ที่แสดงโดยเป็นเส้นทึบ เป็นสัญญาณที่รับได้ สัญญาณนี้สามารถพิจารณาได้ว่าเป็นสัญญาณผลลัพธ์ที่ได้การรวมสัญญาณ ที่ตั้งฉากคู่หนึ่ง ซึ่งแสดงด้วยเส้นประ นั่นคือการดีโมดูเลทสัญญาณ ก็เท่ากับการดีโมดูเลทสัญญาณสองสัญญาณ ดังนั้นสัญญาณ ที่ได้รับคือ การดีเทคแบบ โดยการใช้คลื่นพาหะอ้างอิงที่ตั้งฉากกับคู่หนึ่ง ซึ่งได้โดยการเลื่อนเฟสของสัญญาณที่รับไปเท่ากับนั่นคือ สัญญาณที่ตั้งฉากกันแต่ละสัญญาณถูกการดีเคแบบ แยกกัน



รูปที่ 5.3 สัญญาณ QPSK ถูกชิฟ $\Pi/4$

ลักษณะของ QPSK

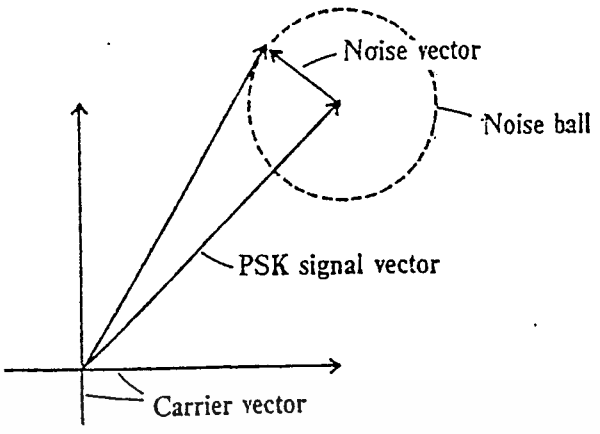
ให้จำนวนบิทที่จะต้องส่งในหนึ่งหน่วยเวลาคือ บิทเรท (bit ratc) มีค่าเท่ากันทั้งในกรณีของ BPSK และ QPSK สำหรับ QPSK บิทจะถูกส่ง 2 บิทในแต่ละครั้ง ดังนั้นความถี่ของการเปลี่ยนเฟสคลื่นพาหะขึ้นกับรหัส นั่นคือ kcying spccd จะเป็นครึ่งหนึ่งเมื่อเทียบกับกรณี BPSK เมื่อช่วงระยะเวลาของบิทแต่ละตัวเป็น T ดังนั้น keying speed จะมีค่าเท่ากับ 1/T สำหรับ BPSK และจะมีค่าเท่ากับ 1/2T สำหรับกรณีของ QPSK



รูปที่ 5.4 แสดง Keying Speed ในการส่งสัญญาณ

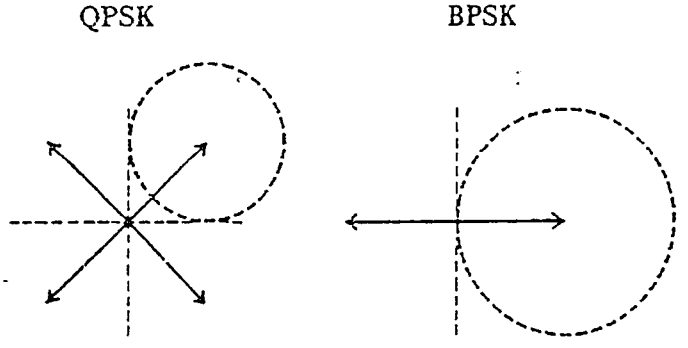
โดยทั่วไปแล้วแถบกว้างความถี่ที่ถูกครอบครองของสัญญาณ PSK จะเพิ่มขึ้นเป็นสัดส่วนกับ keying speed ฉะนั้นเมื่อให้อัตราการส่งบิทคงที่ keying speed ในกรณี QPSK จะเป็นครึ่งหนึ่งของ BPSK และแถบกว้างความถี่ที่ถูกครอบครองของสัญญาณ QPSK จะเป็นครึ่งหนึ่งของสัญญาณ BPSK ในทางตรงกันข้ามเมื่อให้แถบกว้างความถี่เท่ากัน จำนวนบิทซึ่งสามารถส่งได้ในกรณีของ QPSK จะเป็นสองเท่าของ BPSK ดังนั้นจะเห็นว่า QPSK มีข้อดีกว่า BPSK ถ้ามองจากผลในการใช้แถบความถี่คลื่น

รูปที่ 5.5 แสดงเวคเตอร์ไดอะแกรมของสัญญาณ PSK เมื่อมีเสียงรบกวนเข้ามาทำให้แอมพลิจูด และเฟสเปลี่ยนไปจะเห็นว่าทิศทางของเวคเตอร์เสียงรบกวนเปลี่ยนไปได้ทุกทิศทาง เวคเตอร์เสียงรบกวนที่มีขนาดคงที่และหมุนไปทุกทิศทางเป็นวงกลมนี้เรียกว่า “noise ball” เมื่อเวคเตอร์ผลรวมระหว่าง PSK เวคเตอร์และเวคเตอร์เสียงรบกวนข้าม (carrier vector) ไปทางใดทางหนึ่งเฟสที่ส่งก็จะถูกดีโมดูเลทผิดไปเป็นเฟสข้างเคียงเพราะฉะนั้นเมื่อ noise ball มีค่ามาก การผิดพลาดของบิทก็จะเกิดขึ้น



รูปที่ 5.5 PSK เวคเตอร์และ noisc เวคเตอร์

ลองเปรียบเทียบขนาดของ noisc ball สูงสุดที่ไม่ทำให้เกิดความผิดพลาดที่บิท (bit crror) ขึ้นใน BPSK และ QPSK โดยที่ความยาวของเวคเตอร์สัญญาณมีขนาดเท่ากัน ซึ่งหมายความว่ากำลังส่งสัญญาณทั้ง QPSK และ BPSK มีกำลังส่งเท่ากัน เราจะเห็นระยะระหว่างสองเวคเตอร์สัญญาณในกรณี QPSK จะสั้นกว่าในกรณี BPSK และช่วงขนาดของ noise ball ที่ ไม่ทำให้เกิดการผิดพลาดของบิทในกรณี QPSK จะแคบกว่า ดังนั้นถ้ากำลังในการส่งสัญญาณเท่ากันและย่านกว้างความถี่เท่ากัน อัตราการผิดพลาดของบิทในกรณี QPSK จะสูงกว่า กรณี BPSK แต่จำนวนข้อมูลข่าวสารที่ส่งในกรณีQPSK จะมากเป็น 2 เท่าของ BPSK โดยลักษณะเช่นนี้ถ้าต้องการใช้ย่านความถี่ให้ได้ผลมากที่สุดจึงควรใช้ QPSK มากกว่า BPSK แต่ถ้าจะให้อัตราการผิดพลาดของบิทเกิดขึ้นเท่ากัน ก็จำเป็นที่จะต้องให้กำลังในการส่งสัญญาณในกรณี มากกว่ากรณี



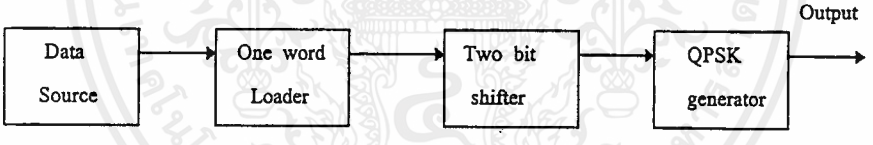
รูปที่ 5.6 เปรียบเทียบขนาดของ noisc ball ที่ไม่ทำให้เกิดการผิดพลาดของบิท

บทที่ 6
หลักการมอดูเลชั่นและดีมอดูเลชั่น
แบบ QPSKที่ใช้ในโครงงาน

ในโครงงานนี้ได้ทำการส่งข้อมูลจากตัวส่งไปยังตัวรับ โดยใช้หลักการมอดูเลชั่น และดีมอดูเลชั่นแบบ QPSK ที่เฟสของสัญญาณพาหะจะเปลี่ยนไปตามข้อมูล 2 บิท ที่ได้อธิบายไว้แล้วในบทที่ 5 ในบทที่ 6 นี้จะกล่าวถึงวิธีการทั้งหมดที่ได้นำมาประยุกต์ใช้ในการมอดูเลทและดีมอดูเลท สัญญาณในโครงงานนี้

6.1 หลักการมอดูเลชั่นแบบ QPSK ของตัวส่ง

ส่วนประกอบ

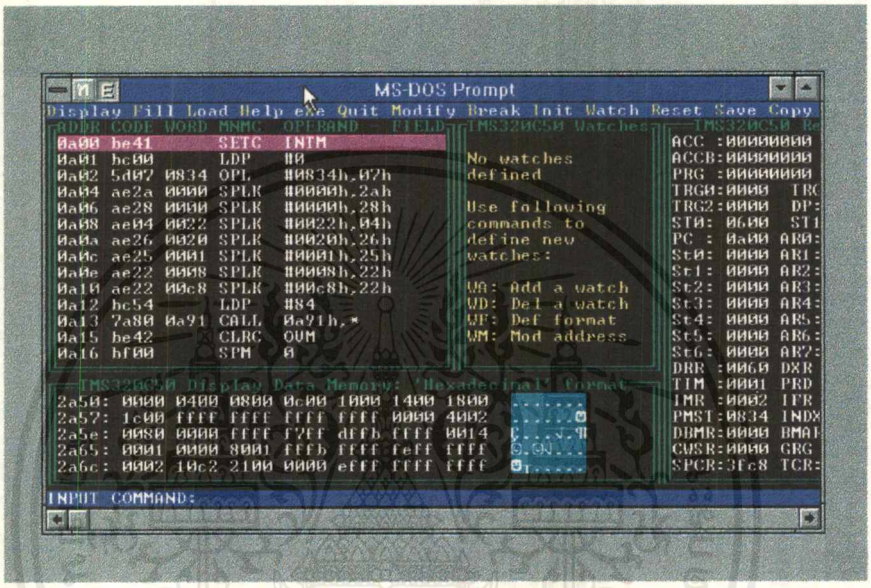


รูปที่ 6.1 บล็อกไดอะแกรมแสดงส่วนประกอบต่างๆของส่วนมอดูเลชั่นของตัวส่ง

1. **Data Source** เป็นชุดข้อมูลที่ต้องการจะส่งซึ่งจะถูกเก็บไว้ในหน่วยความจำ
2. **One word Loader** เป็นส่วนที่ทำหน้าที่โหลดข้อมูลขนาด 1 เวิร์ด จากหน่วยความจำมาเก็บไว้ในบัฟเฟอร์เพื่อใช้ในการประมวลผลขั้นต่อไป
3. **Two bit Shifter** เป็นส่วนที่ใช้เลื่อนข้อมูลเข้าประมวลผล (มอดูเลทกับสัญญาณพาหะซายน์) ทีละ 2 บิท ตามหลักการของการมอดูเลชั่นแบบ QPSK
4. **QPSK generator** เป็นส่วนที่ทำการสร้างสัญญาณพาหะซายน์ที่มีเฟสเปลี่ยนไปตามข้อมูล 2 บิทโดยอาศัยหลักการของ QPSK

6.1.1 แหล่งข้อมูล(Data Source)

คือ ข้อมูลที่ต้องการจะส่งไปยังตัวรับ ซึ่งจะต้องถูกนำมามอดูเลทกับสัญญาณพาหะซายน์ด้วยวิธี QPSK เสียก่อนจึงจะส่งไปยังตัวรับได้ ข้อมูลที่ต้องการส่งจะถูกเก็บไว้ในหน่วยความจำของตัวส่ง



รูปที่ 6.2 ภาพแสดงตำแหน่ง 2a50h - 2a57h ในหน่วยความจำที่ใช้เก็บข้อมูลของสัญญาณรูปขั้นบันไดที่จะนำไปมอดูเลทกับสัญญาณพาหะในส่วนถัดไป

6.1.2 ส่วนโหลดข้อมูล 1 เวิร์ด (One word Loader)

เป็นส่วนที่ทำหน้าที่ดึงข้อมูลจากหน่วยความจำมาทีละ 1 เวิร์ดเพื่อนำไปใช้ในการประมวลผลในส่วนถัดไป

6.1.3 ส่วนเลื่อน 2 บิท (Two bit Shifter)

เป็นส่วนที่ใช้แยกข้อมูล 1 เวิร์ด ออกมาทีละ 2 บิท โดยจะพิจารณา 2 บิทท้าย (LSB) ก่อน การแยกข้อมูล 2 บิทท้ายออกมาทำได้โดยนำไป AND กับ 0003h แล้วจึงนำข้อมูล 2 บิทที่ได้ไปทำการมอดูเลทกับสัญญาณพาหะซายน์จากนั้นจะทำการเลื่อนข้อมูลไปทางขวา 2 บิทถัดจะเลื่อนไปอยู่ที่ 2 บิทท้ายแทน ทำซ้ำขั้นตอนที่กล่าวมาจนครบ 16 บิท

6.1.4 ส่วนสร้างสัญญาณ QPSK (QPSK Generator)

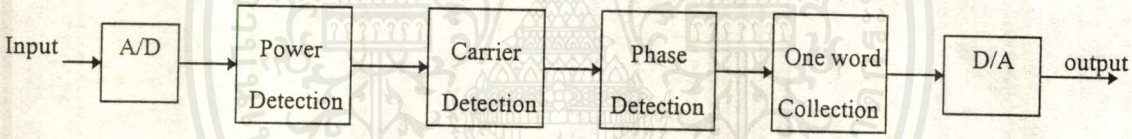
ข้อมูล 2 บิทจากส่วนเลื่อนจะถูกเปรียบเทียบกับค่าที่กำหนดไว้ในโปรแกรมโดยถ้า 2 บิทดังกล่าวเป็น

- 00 รีจิสเตอร์จะถูกโหลดด้วยตำแหน่งแอดเดรสเริ่มต้นของตารางซายน์เฟส 0°
- 01 รีจิสเตอร์จะถูกโหลดด้วยตำแหน่งแอดเดรสเริ่มต้นของตารางซายน์เฟส 90°
- 11 รีจิสเตอร์จะถูกโหลดด้วยตำแหน่งแอดเดรสเริ่มต้นของตารางซายน์เฟส 180°
- 10 รีจิสเตอร์จะถูกโหลดด้วยตำแหน่งแอดเดรสเริ่มต้นของตารางซายน์เฟส 270°

จากนั้นก็จะทำการโหลดข้อมูลจากตารางซายน์(มีขนาด 1 ลูกครึ่ง) ของเฟสที่ต้องการจะส่งออกไปทาง D/A

6.2 หลักการดีมอดูเลชั่นแบบ QPSK ของตัวรับ

ส่วนประกอบ

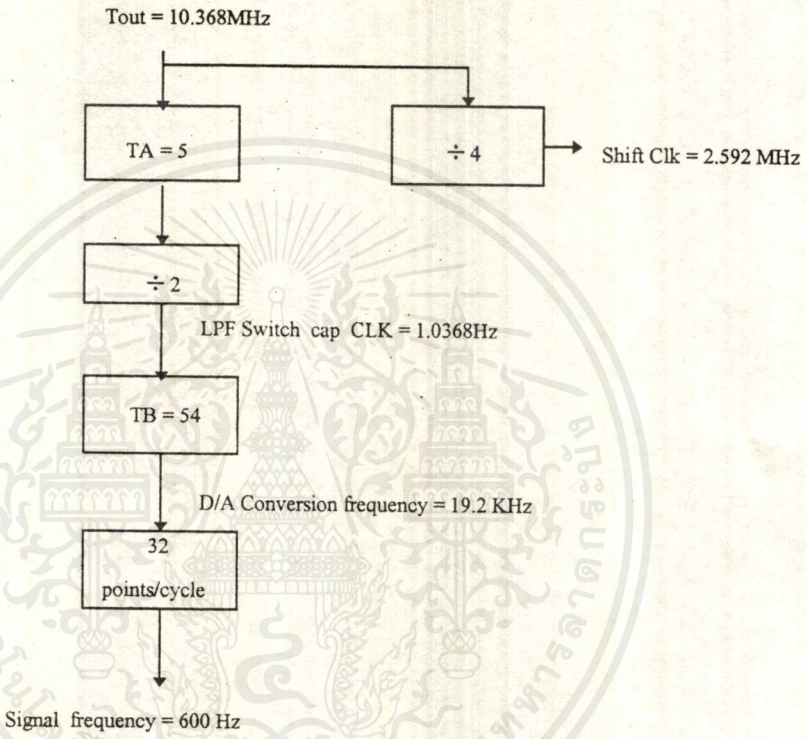


รูปที่ 6.3 แสดงส่วนประกอบของภาครับ

1. **A/D** เป็นตัวสุ่ม(sampling) สัญญาณอนาลอกให้อยู่ในรูปรหัสดิจิตอล
2. **Power Detection** เป็นส่วนที่ใช้หาจุดเริ่มต้นของสัญญาณอินพุทที่เข้ามา
3. **Carrier Detection** เป็นส่วนที่ใช้สังเคราะห์สัญญาณซายน์ที่มีความถี่เป็น 4 เท่าของสัญญาณอินพุทจากสัญญาณอินพุทที่รับเข้ามาเพื่อใช้เป็นฐานเวลาในการดีเทคเฟสของสัญญาณ
4. **Phase Detection** เป็นส่วนที่ใช้ดีเทคเฟสเพื่อให้ได้ข้อมูลซึ่งอยู่ในรูปเลขฐาน ส่วน 2 บิท ตามหลักการของ QPSK
5. **ส่วนสะสมข้อมูล** จะทำการเก็บข้อมูลที่ละ 2 บิทจนครบ 16 บิท (1 word)
6. **D/A** ทำการแปลงข้อมูลดิจิตอล เป็นสัญญาณ Analog เพื่อส่งไปแสดงผลที่ออสซิลโลสโคป

6.2.1 ส่วน A/D และ D/A

A/D เป็นตัวที่ใช้สุ่มสัญญาณอนาลอกให้อยู่ในรูปดิจิตอลเพื่อให้สามารถนำไปประมวลผลด้วยโปรแกรมได้ ส่วน D/A จะแปลงข้อมูลดิจิตอลที่ได้จากการประมวลผลกลับไปเป็นสัญญาณอนาลอก ในโครงงานนี้ใช้ A/D และD/A เบอร์ TLC32040C ของบอร์ด TMS320C50 DSP Starter Kit ซึ่งมีหลักการหาอัตราการสุ่มดังรูปข้างล่างนี้



รูปที่ 6.4 บล็อกไดอะแกรมแสดงการคำนวณหาค่าความถี่ของสัญญาณที่ออกจากตัวส่ง

สมการคำนวนหาค่าความถี่ของจุดต่างๆแสดงได้ดังนี้

1. Shift CLK $= \frac{Tout}{4} = \frac{1.0368\,MHz}{4} = 2.592$ MHz
2. LP Switch cap CLK $= \frac{(X-tal)/2}{TA} = \frac{1.0368\,MHz/2}{5} = 1.0368$ MHz
3. D/A CLK $= \frac{LP.switch.cap.CLK}{TB} = \frac{1.0368\,MHz}{54} = 19.2$ KHz
4. Signal frequency $= \frac{D/A.CLK}{32} = \frac{19.2\,KHz}{32} = 600$ Hz
5. ข้อมูล 2 บิทแทนด้วยสัญญาณ 1.5 ไซเคิล จะได้ 1.33 boud
6. อัตราการส่งข้อมูล = Signal Frequency x 1.33 = 800 bps

6.2.2 ส่วนเช็คจุดเริ่มต้น(Power detection)

เนื่องจากการดีมอดูเลทสัญญาณแบบQPSK นั้นจะต้องทำการดีเทคเฟสของสัญญาณที่รับเข้ามาให้ได้จึงจะสามารถแปลงกลับไปเป็นข้อมูลตามเดิมได้ ซึ่งจำเป็นจะต้องดีเทคจุดเริ่มต้นที่แท้จริงของสัญญาณที่รับเข้ามาไม่เช่นนั้นแล้วเฟสของสัญญาณที่ดีเทคได้จะผิดพลาดไปทำให้ไม่สามารถแปลงกลับไปเป็นข้อมูลตามเดิมได้

ในโครงงานนี้จะใช้หลักการเปรียบเทียบพลังงานของสัญญาณที่รับมาได้ 8 จุดแรกกับค่าพลังงานอ้างอิง (ในขั้นต้นจะสามารถคำนวณค่าพลังงานอ้างอิงจากผลรวม 8 จุดแรกของค่ากำลังสองของสัญญาณที่จะถูกส่งออกมาจากตัวส่ง) โดยถ้าพลังงานของสัญญาณที่รับเข้ามามีค่าน้อยกว่าพลังงานอ้างอิงก็จะถือว่าไม่ใช่สัญญาณที่ต้องการก็จะไม่ทำการประมวลผลในส่วนถัดไปและจะทำหักค่าของจุดแรกทิ้งไป จากนั้นค่อยนำไปรวมกับค่ากำลังสองของจุดถัดไปเพื่อให้ครบ 8 จุด ถึงจะนำไปเปรียบเทียบกับพลังงานอ้างอิงอีกครั้ง ทำเช่นนี้จนกว่าจะพบจุดเริ่มต้น (จุดที่ทำให้พลังงานของ 8 จุดแรกมีค่ามากกว่าพลังงานอ้างอิง) ซึ่งวิธีนี้สามารถลดปัญหาที่เกิดจากสัญญาณรบกวนได้เนื่องจากสัญญาณรบกวนจะมีแอมพลิจูดต่ำซึ่งเมื่อนำมาหาค่าพลังงานแล้วจะต่ำกว่าค่าพลังงานอ้างอิง

โดยมีสมการหาค่าพลังงานดังนี้

$$P = \sum_{N=1}^{8} (X_N)^2$$

เมื่อ P : พลังงานของข้อมูล 8 จุดแรก

X_N : ข้อมูลในแต่ละจุด

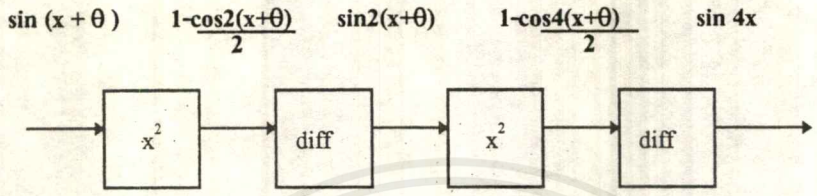
N : ลำดับที่ของจุด

6.2.3 ส่วนดีเทคสัญญาณพาหะ(Carrier detection)

ในส่วนดีเทคเฟสนั้น จะต้องแบ่งสัญญาณอินพุท 1 ลูกคลื่นออกเป็น 4 ส่วน ซึ่งในส่วนของโปรแกรมที่เขียนขึ้นนั้นจะกำหนดค่าความถี่ของสัญญาณพาหะและอัตราการสุ่มของตัว A/D และตัว D/Aไว้ค่าหนึ่งซึ่งเหมือนกันทั้งตัวรับและตัวส่ง แต่เพื่อให้สามารถนำไปประยุกต์ใช้กับสัญญาณพาหะหลายความถี่ โดยตัวรับจะต้องดีเทคความถี่ของตัวส่งให้ได้ก่อนถึงจะสามารถดีเทคเฟสได้ ในโครงงานนี้จึงนำเสนอวิธีการสังเคราะห์สัญญาณที่มีความถี่เป็น 4 เท่าของสัญญาณพาหะจากสัญญาณอินพุทที่รับเข้ามาเพื่อนำไปใช้ในการแบ่งส่วนของสัญญาณอินพุท 1 ลูกคลื่นออกเป็น 4 ส่วนต่อไป

จากสัญญาณอินพุทซายน์ที่มีเฟสเปลี่ยนไปตามการมอดูเลชั่นแบบ QPSKที่รับมานั้น สามารถกำจัดเฟสให้เหลือเฉพาะความถี่ โดยผลลัพธ์ที่ได้จะเป็นสัญญาณซายน์ที่มีความถี่เป็น 4 เท่าของสัญญาณอินพุท(สัญญาณพาหะ)ซึ่งมีหลักการดังนี้

หลักการ สร้าง sin 4x



รูปที่ 6.5 บล็อกแสดงหลักการสังเคราะห์ SIN 4X

จากสัญญาณอินพุท $\sin(x+\theta)$

ยกกำลัง : $\sin^2(x+\theta) = \frac{1-\cos(x+\theta)}{2}$

diff : $\frac{d[1-\cos 2(x+\theta)]/2}{dx} = \frac{-2*[-(\sin 2(x+\theta)]}{2}$

$= \sin 2(x+\theta)$

ยกกำลังสอง : $\sin^2(2x+2\theta) = \frac{1-\cos 4(x+\theta)}{2}$

diff : $\frac{d[1-\cos 4(x+\theta)]/2}{dx} = \sin 4(x+\theta)$

จากหลักการของ QPSK จะได้สัญญาณซายน์ที่มีเฟสของสัญญาณแตกต่างกัน 4 แบบ ดังนี้

$\theta = 0, 90, 180, 360$ หรือ

$\theta = 90n$ เมื่อ $n = 0, 1, 2, \ldots$

และเนื่องจาก

$$\sin(4x + 4\theta) = \sin(4x + 4*90n)$$
$$= \sin(4x + 360n)$$
$$= \sin(4x + 0)$$
$$= \sin 4x \quad \#$$

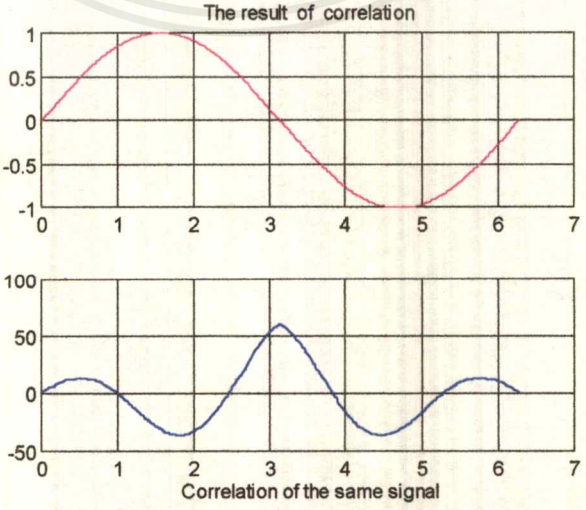
6.2.4 ส่วนดีเทคเฟส (Phase detection)

จากหนังสือ Principle of Signal and System ของ Fred J. Taylor กล่าวถึง coorelation ไว้ว่าเป็นตัวที่ใช้วัดความเหมือนกันของสัญญาณสองสัญญาณ(Correlation is a measure of the similarity existing between two signals.) การ correlation แสดงได้ดังสมการข้างล่างนี้

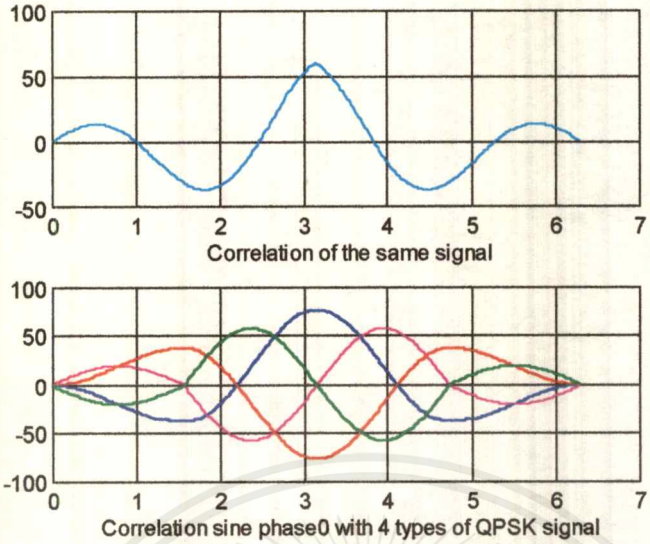
$$R_{fg}(x) = \int_{-\infty}^{\infty} f(t)g(t+x) \quad dt$$

เมื่อ $R_{fg}(x)$: cross correlation function ของฟังก์ชัน f(t) กับ g(t)

จากสมการจะพบว่าค่าผลคูณ (inner product) จะมีค่าสูงสุดเมื่อสัญญาณทั้งสองทับกันพอดีนั่นคือผลรวมของผลคูณแอมพลิจูดที่เวลาเดียวกันของสัญญาณทั้งสองจะมีค่าสูงสุดเมื่อมันซ้อนทับกันพอดี ด้วยหลักการที่ว่านี้สามารถนำไปประยุกต์ใช้ในการดีเทคเฟสของสัญญาณได้



รูปที่ 6.6 แสดงผลลัพธ์การcorrelation ระหว่างสัญญาณที่เหมือนกัน



เส้นสีน้ำเงิน : ผลการ correlation ระหว่าง sine เฟส 0° กับ square เฟส 0°

เส้นสีม่วง : ผลการ correlation ระหว่าง sine เฟส 0°กับ square เฟส 90°

เส้นสีแดง : ผลการ correlation ระหว่าง sine เฟส 0° กับ square เฟส 180°

เส้นสีเขียว : ผลการ correlation ระหว่าง sine เฟส 0° กับ square เฟส 270°

รูปที่ 6.7 ผลลัพธ์จากการ correlation ระหว่างสัญญาณที่เหมือนกันเปรียบเทียบกับผลที่ได้จากการcorrelation ระหว่างสัญญาณซายน์เฟส 0°และสัญญาณซายน์ที่มีเฟส 0°, 90°, 180°และ270°

เนื่องจากสัญญาณอินพุทที่เข้ามาเป็นสัญญาณซายน์ สัญญาณที่จะนำมา correlation ก็จะต้องเป็นสัญญาณซายน์ด้วย แต่ในทางปฏิบัติการสร้างสัญญาณซายน์ทำได้ยาก จึงนำสัญญาณอินพุทไป correlation กับสัญญาณรูปสี่เหลี่ยมแทนเนื่องจากสร้างได้ง่ายกว่าอีกทั้งสัญญาณรูปสี่เหลี่ยมก็คือสัญญาณรูปซายน์หลายๆความถี่มารวมกันนั่นเอง แสดงผลการ correlation กับสัญญาณรูปสี่เหลี่ยมได้ดังนี้

จากสมการ correlation

$$R_{fg}(x) = \int_{-\infty}^{\infty} f(t)g(t+x) \quad dt$$

สมการสัญญาณอินพุทซายน์

$$G(t) = \sin(t + \theta_2)$$

และสมการสัญญาณรูปสี่เหลี่ยม

$$F(t) = Ao + \sum_{n=-\infty}^{\infty} An\cos(nw_0t) + \sum_{n=-\infty}^{\infty} Bn\sin nw_0t)$$

ทำการ integrate inner product ระหว่าง สัญญาณรูปสี่เหลี่ยม กับ input (QPSK)

$$R_{fg}(x) = \int F(t) * G(t+x)\ dt$$

$$= \int [A_0 + \sum_{n=-\infty}^{\infty} A_n\cos(nw_0t+\theta_1) + \sum_{n=-\infty}^{\infty} Bn\sin(nw_0t+\theta_1)]\sin(nw_0t+\theta_2)dt$$

จากสมบัติ Orthogonal ของ sine และ cos ที่ว่า

$$\int_{-T/2}^{T/2} \cos(mw_0t)\ dt = 0 \quad ; \text{for } m \neq 0$$

$$\therefore \quad R_{fg}(x) = \int \sum_{n=-\infty}^{\infty} [Bn\sin(nw_0t+\theta_1)\sin(nw_0t+\theta_2)]\ dt$$

จากสมการตรีโกณมิติ

$$2\sin A\sin B = \cos(A+B) + \cos(A+B)$$

$$\therefore \text{จะได้} \quad R_{fg}(x) = Cn \int \sum_{n=-\infty}^{\infty} [\cos(nw_0t+\theta_1+w_0t+\theta_2) + \cos(nw_0t+\theta_1+w_0t+\theta_2)]\ dt$$

กรณีที่ความถี่เท่ากัน พิจารณาเฉพาะ เฟส

$$= Cn \int \sum_{n=-\infty}^{\infty} [\cos(2w_0t+\theta_1+\theta_2) + \cos(\theta_1-\theta_2)]\ dt$$

$$= Cn \int \sum_{n=-\infty}^{\infty} \cos(2w_0t + \theta_1 + \theta_2)\ dt + Cn \int \sum_{n=-\infty}^{\infty} Cn \cos(\theta_1 - \theta_2)\ dt$$

จาก $\int_{-T/2}^{T/2} \cos(mw_0t)\ dt = 0 \qquad \text{for } m \neq 0$

$$\therefore \qquad R_{fg} = Cn \int \sum_{n=-\infty}^{\infty} \cos(\theta_1 - \theta_2)\ dt$$

เมื่อทั้งสองความถี่เท่ากันแล้ว ก็มาพิจารณาที่ เฟส

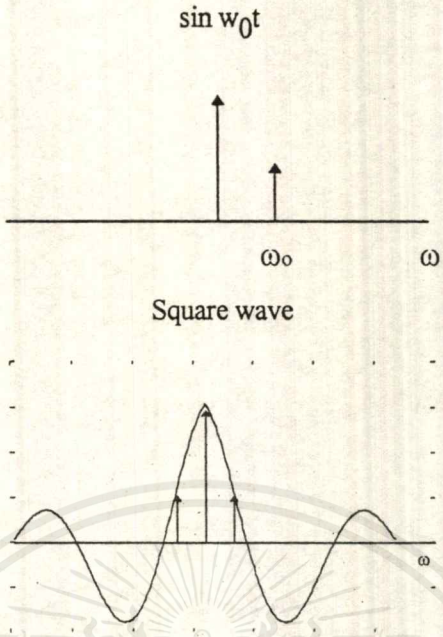
ถ้า $\theta_1 = \theta_2$ ผลลัพธ์ => + สูงสุด

$\theta_1 = -\theta_2$ ผลลัพธ์ => - ต่ำสุด

ต่างกัน $90^o, 270^o$ ผลลัพธ์ => 0

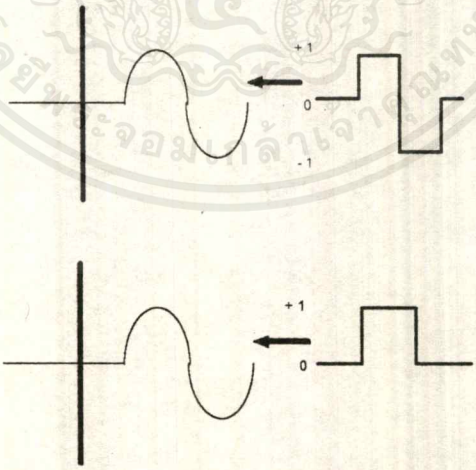
การเลือกใช้สัญญาณรูปสี่เหลี่ยมทำการcorrelation กับสัญญาณอินพุทซายน์นั้นสามารถเขียนโปรแกรมได้ง่ายกว่าแต่อาจพบปัญหาเรื่องสัญญาณรบกวนเพราะเมื่อเปรียบเทียบสเปคตรัมความถี่ของ สัญญาณทั้งสองแล้วพบว่าสัญญาณรูปสี่เหลี่ยมจะมีสัญญาณรบกวนได้มากกว่าเนื่องจากมีแบนด์วิทธ์ที่กว้างกว่า

จากที่กล่าวมาข้างต้นสรุปได้ว่าการcorrelation ใช้วัดความเหมือนของสัญญาณสองสัญญาณและถ้าหากสองสัญญาณเหมือนกันกราฟผลลัพธ์ของการcorrelationจะมีค่าแอมพลิจูดสูงสุดที่จุดศูนย์กลางซึ่งก็คือค่า inner product ในขณะที่สัญญาณทั้งสองซ้อนทับกันพอดี ด้วยหลักการนี้ในโครงงานจึงเลือกใช้การหา inner product ของสัญญาณทั้งสองในขณะที่มันซ้อนทับกันพอดีแทนที่จะต้องหา correlation ซึ่งเป็นการยุ่งยากโดยไม่จำเป็น โดยจะนำสัญญาณอินพุทซายน์ไป inner product กับสัญญาณรูปสี่เหลี่ยมที่มีความถี่เดียวกับสัญญาณอินพุทซายน์



รูปที่ 6.8 แสดงสเปคตรัมความถี่ของสัญญาณซายน์และสัญญาณรูปสี่เหลี่ยม

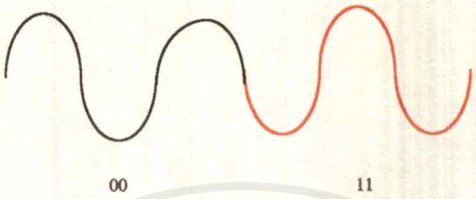
ในส่วนของโปรแกรมจะใช้หลักการของ inner product ระหว่างสัญญาณอินพุทกับสัญญาณรูปสี่เหลี่ยมทั้ง 4 เฟส จากนั้นก็จะเปรียบเทียบค่า inner product ที่ได้จากสัญญาณอินพุทกับเฟสใดให้ค่าสูงสุดก็แสดงว่าสัญญาณอินพุทมีเฟสตรงกับสัญญาณรูปสี่เหลี่ยมเฟสนั้น



รูปที่ 6.9 ภาพแสดงการ Correlation ระหว่างสัญญาณอินพุทซายน์กับสัญญาณรูปสี่เหลี่ยม

ในทางปฏิบัติแล้วไม่สามารถใช้ Correlation ในการดีเทคเฟสได้เนื่องจากในกรณีที่สัญญาณ QPSK ที่ได้จากข้อมูล 2 บิทติดกันออกมาเป็นรูปซายน์ เช่น 0011 ซึ่งจะได้เฟส 90 ต่อ

กับ เฟส 0 เป็นรูปซายน์ออกมา ซึ่งในกรณีนี้ Correlation จะไม่สามารถแยกความแตกต่างของเฟสทั้ง 4 ได้ จึงเลือกใช้การหาค่า Inner product ระหว่างสัญญาณอินพุทที่ต้องการตรวจสอบเฟสกับสัญญาณสี่เหลี่ยมที่สร้างขึ้นทั้ง 4 เฟส ขณะที่สัญญาณทั้งสองซ้อนทับกันพอดี แล้วนำมาเปรียบเทียบว่าค่า Inner Product ที่ได้จากเฟสใดมีค่าสูงสุด ก็แสดงว่าสัญญาณอินพุทมีเฟสตรงกับเฟสนั้น

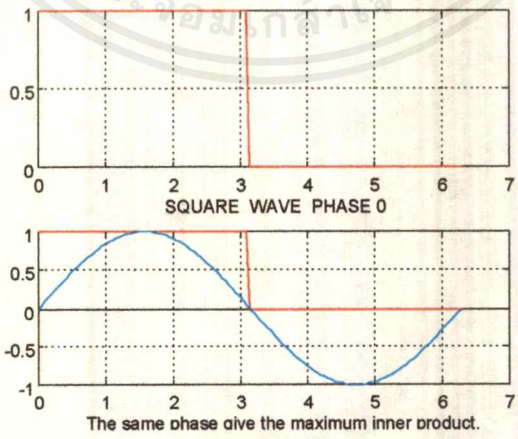


รูปที่ 6.10 ภาพแสดงกรณีที่ไม่สามารถใช้ Correlation ดีเทคเฟสได้

วิธีการที่ใช้หาค่า Inner product ในโปรแกรม

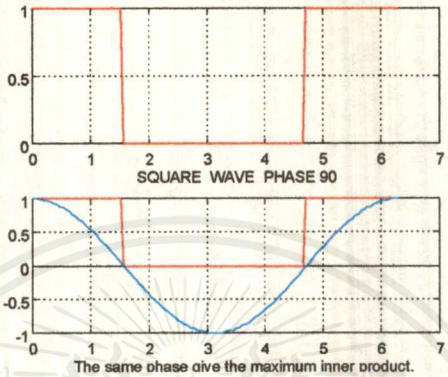
1. แบ่งสัญญาณอินพุทซายน์ 32 จุดออกเป็น 4 ส่วน
2. หาผลบวกของสัญญาณซายน์ ในแต่ละส่วน
3. ทำการจำแนกค่าผลบวกที่ได้ให้ตรงกับแต่ละเฟสดังนี้

- เฟส 0° ค่า Inner product จะได้จากผลบวกของสัญญาณซายน์ในส่วนที่ 1 และส่วนที่ 2 เนื่องจากสัญญาณรูปสี่เหลี่ยมเฟส 0^{o} มีแอมพลิจูดเป็น 1 ในส่วนที่ 1 และส่วนที่ 2 ส่วนส่วนที่3 และส่วนที่ 4 นั้นสัญญาณรูปสี่เหลี่ยมมีค่าเป็นศูนย์ และค่า Inner product จะมากสุดเมื่อสัญญาณทั้งสองมีเฟสตรงกัน



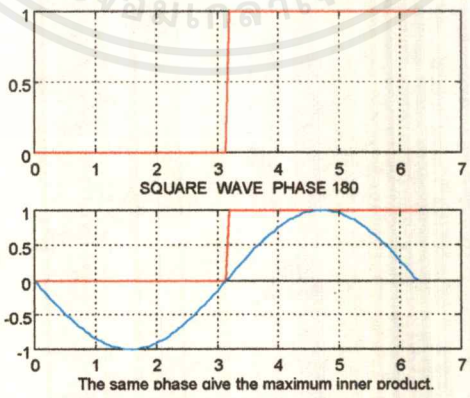
รูปที่ 6.11 แสดงสัญญาณรูปสี่เหลี่ยมเฟส0^{o} และ กรณีที่นำไป Inner product กับสัญญาณอินพุทซายน์แล้วได้ค่าสูงสุด

- เฟส 90° ค่า Inner product จะได้จากผลบวกของสัญญาณซายน์ในส่วนที่ 1 และส่วนที่ 4 เนื่องจากสัญญาณรูปสี่เหลี่ยมเฟส 0^{o} มีแอมพลิจูดเป็น 1 ในส่วนที่ 1 และส่วนที่ 4 ส่วนส่วนที่ 2 และส่วนที่ 3 นั้นสัญญาณรูปสี่เหลี่ยมมีค่าเป็นศูนย์และค่า Inner product จะมากสุดเมื่อสัญญาณทั้งสองมีเฟสตรงกัน



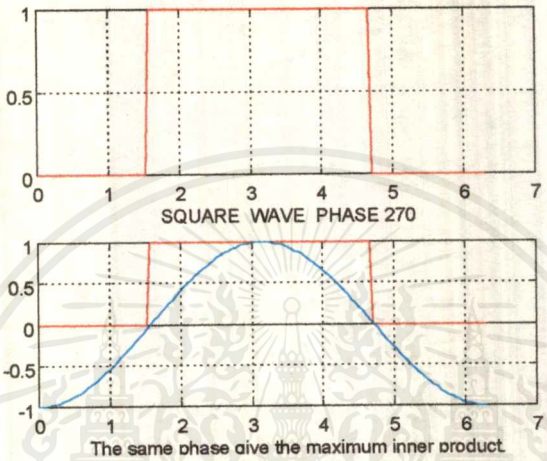
รูปที่ 6.12 แสดงสัญญาณรูปสี่เหลี่ยมเฟส 90^{o} และ กรณีที่นำไป Inner product กับสัญญาณอินพุท ซายน์แล้วได้ค่าสูงสุด

-เฟส 180° ค่า Inner product จะได้จากผลบวกของสัญญาณซายน์ในส่วนที่ 3 และส่วนที่ 4 เนื่องจากสัญญาณรูปสี่เหลี่ยมเฟส 0^{o} มีแอมพลิจูดเป็น 1 ในส่วนที่ 3 และส่วนที่ 4 ส่วนส่วนที่ 1 และส่วนที่ 2 นั้นสัญญาณรูปสี่เหลี่ยมมีค่าเป็นศูนย์และค่า Inner product จะมากสุดเมื่อสัญญาณทั้งสองมีเฟสตรงกัน



รูปที่ 6.13 แสดงสัญญาณรูปสี่เหลี่ยมเฟส180^{o} และ กรณีที่นำไป Inner product กับสัญญาณอินพุทซายน์แล้วได้ค่าสูงสุด

- เฟส 270° ค่า Inner product จะได้จากผลบวกของสัญญาณซายน์ในส่วนที่ 2 และส่วนที่3 เนื่องจากสัญญาณรูปสี่เหลี่ยมเฟส 0^{o} มีแอมพลิจูดเป็น 1 ในส่วนที่ 2 และส่วนที่ 3 ส่วนส่วนที่ 1 และส่วนที่ 4 นั้นสัญญาณรูปสี่เหลี่ยมมีค่าเป็นศูนย์และค่า Inner product จะมากสุดเมื่อสัญญาณทั้งสองมีเฟสตรงกัน



รูปที่ 6.14 แสดงสัญญาณรูปสี่เหลี่ยมเฟส0^{o} และ กรณีที่นำไป Inner product กับสัญญาณอินพุทซายน์แล้วได้ค่าสูงสุด

6.2.5 ส่วนสะสมข้อมูล (One word Collection)

เนื่องจากในการส่งหนึ่งครั้งของตัวส่ง จะส่งข้อมูล 2 บิท มากับสัญญาณซายน์ลูกคลื่นครึ่งจึงจะต้องทำการรับและดีเทคข้อมูล 8 ครั้ง จึงจะได้ข้อมูลครบ 1 จุด (16 บิท) ของชุดข้อมูลที่ทำการส่งมา

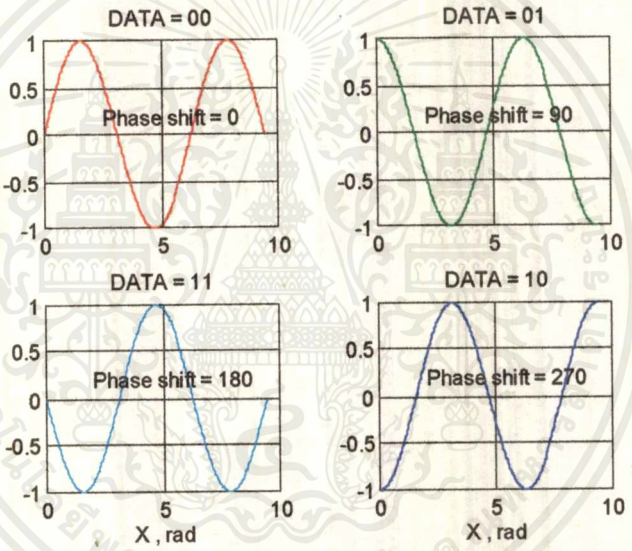
บทที่ 7

การศึกษาและจำลองการทำงานด้วยโปรแกรม MATLAB

เพื่อทดสอบหลักการและวิธีการต่างๆที่คิดขึ้นว่าจะนำไปใช้ได้จริงเพียงใดจึงได้จำลองการทำงานในส่วนต่างๆ โดยใช้โปรแกรม MATLAB ซึ่งแบ่งออกเป็น 2 ส่วนหลัก ดังนี้

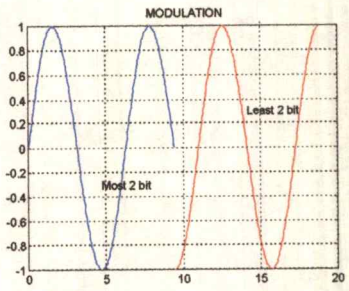
1. ใช้ MATLAB จำลองการทำงานในส่วนของภาคส่ง

1.1 สร้างสัญญาณซายน์ที่มีเฟสเปลี่ยนไปตามข้อมูล 2 บิทไปกับสัญญาณซายน์ขนาดหนึ่งลูกครึ่ง 4 แบบ ตามหลักการของ QPSK



รูปที่ 7.1 แสดงสัญญาณซายน์ที่ถูกมอดูเลทด้วยวิธี QPSK

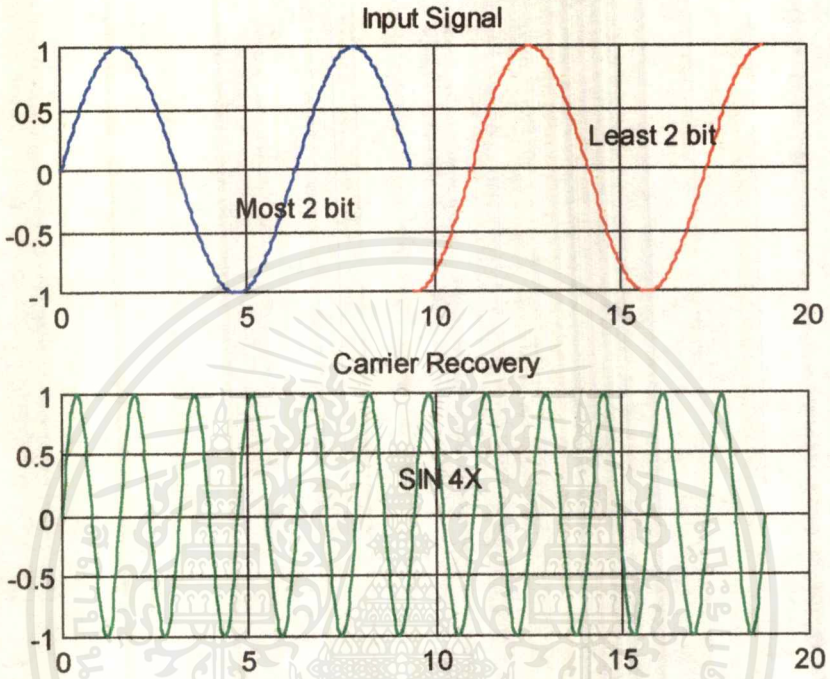
1.2 มอดูเลทข้อมูล 4 บิท กับสัญญาณซายน์ด้วยวิธี QPSK



รูปที่ 7.2 แสดงสัญญาณซายน์ที่ถูกมอดูเลทด้วยข้อมูล ' 0010 '

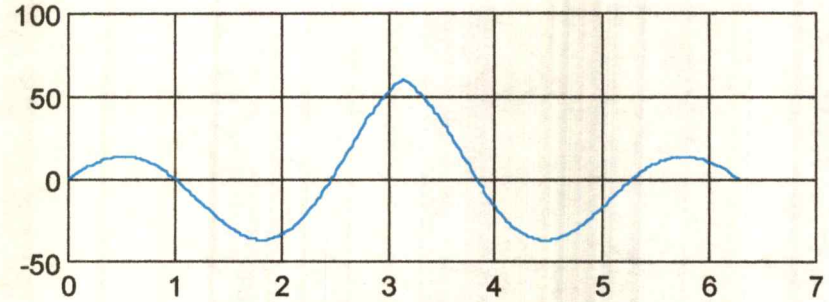
2. ใช้ MATLAB ศึกษาและจำลองการทำงานในส่วนของภาครับดังนี้

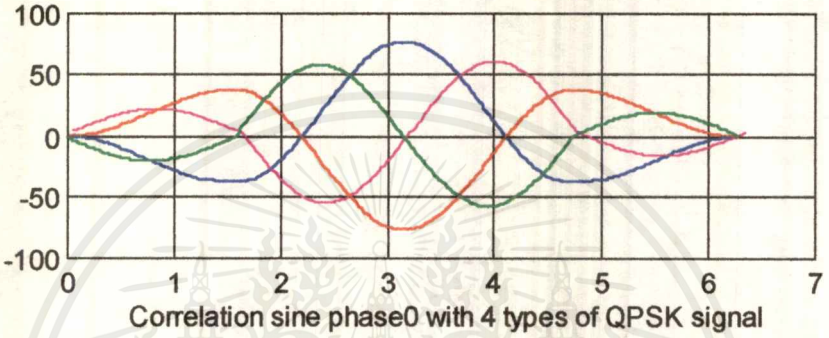
2.1 สังเคราะห์สัญญาณซายน์ที่มีความถี่เป็น 4 เท่าของสัญญาณพาหะ



รูปที่ 7.3 แสดงการสังเคราะห์สัญญาณ sin4x จากสัญญาณอินพุท

2.2 ทดสอบหลักการ correlation เพื่อใช้ในการดีเทคเฟส โดยการนำสัญญาณซายน์ที่มีเฟสต่างกัน 4 แบบ ตามหลักการของ QPSK ได้แก่ เฟส 0^{o} , เฟส 90^{o} , เฟส 180^{o} และ เฟส 270^{o} มาทำการ correlation กับสัญญาณซายน์ที่สร้างขึ้นในตัวส่งซึ่งจะต้องมีความถี่เดียวกันกับสัญญาณอินพุท





เส้นสีน้ำเงิน : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ sine เฟส 0^{o}

เส้นสีม่วง : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ sine เฟส 90^{o}

เส้นสีแดง : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ sine เฟส 180^{o}

เส้นสีเขียว : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ sine เฟส 270^{o}

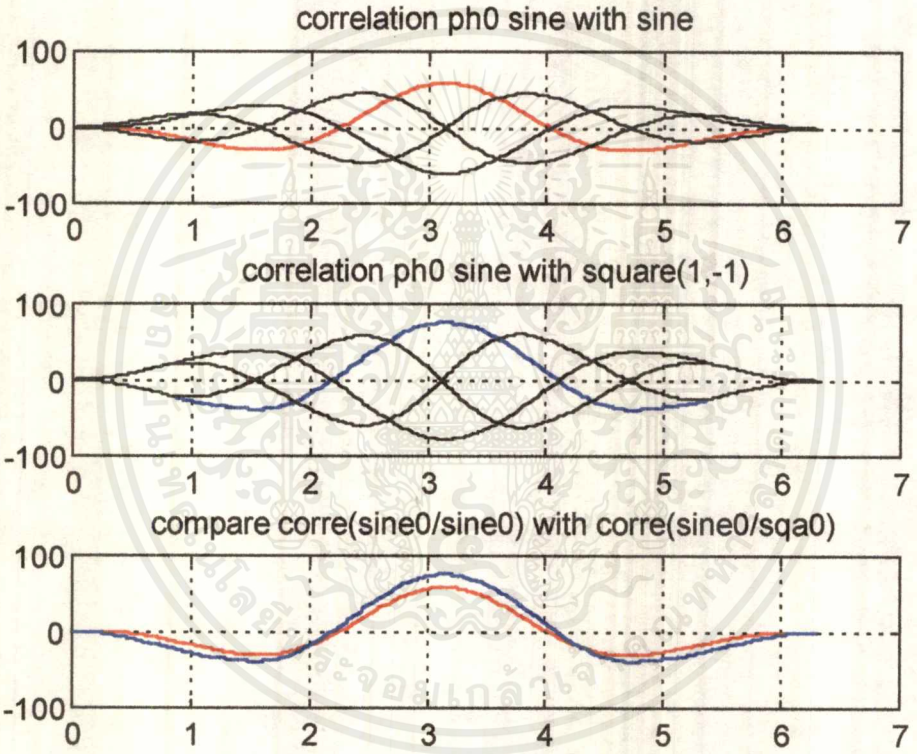
รูปที่ 7.4 แสดงผลลัพธ์ที่ได้จากการcorrelation ระหว่างสัญญาณที่เหมือนกันและผลการ correlationที่ได้จากสัญญาณซายน์เฟส 0^{o} กับสัญญาณอินพุทซายน์ที่ถูกมอดูเลทแบบ QPSKทั้ง 4 แบบ

จะพบว่าถ้าหากสัญญาณที่นำมา correlate กันมีลักษณะเหมือนกันกราฟผลลัพธ์ที่ได้จะมีค่าสูงสุด ณ จุดศูนย์กลางซึ่งก็คือค่าที่ได้จากการ inner product ของสัญญาณทั้งสองในขณะที่มันซ้อนทับกันพอดี

2.3 ทดสอบการ correlation ระหว่างสัญญาณอินพุทซายน์ ทั้ง 4 แบบ กับ สัญญาณรูปสี่เหลี่ยม(square wave) 3 แบบ ดังนี้

- แอมพลิจูด ±1 Vpp
- แอมพลิจูด 0 - 1 V
- แอมพลิจูด -1 - 0 V

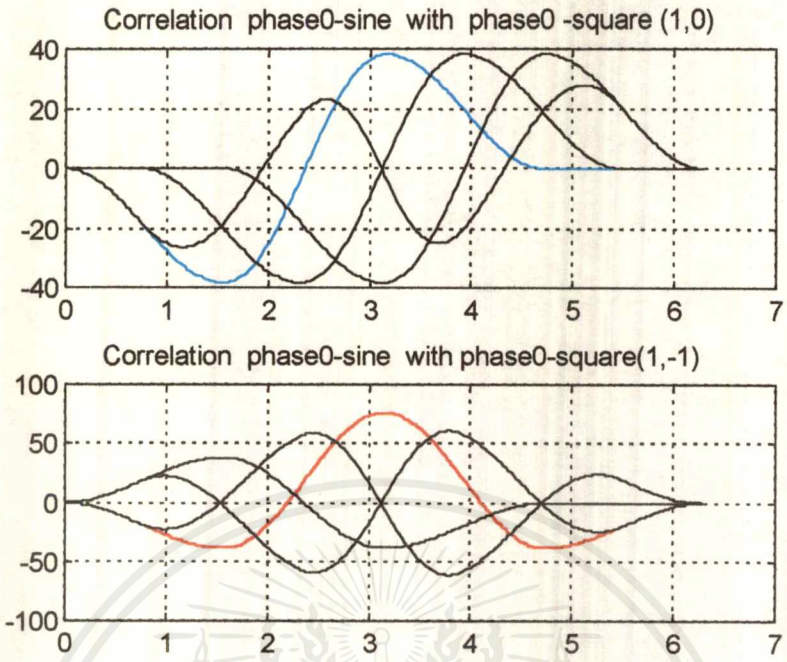
ได้ผลลัพธ์แสดงดัง 3 รูปด้านล่าง ตามลำดับ



เส้นสีแดง : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ sine เฟส 0^{o}

เส้นสีน้ำเงิน : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ square(1.-1) เฟส 0^{o}

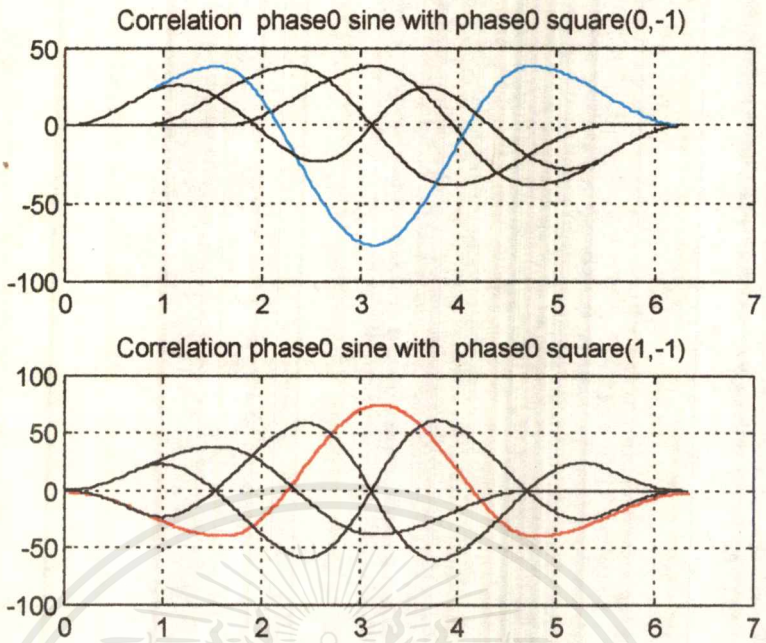
รูปที่ 7.5 แสดงการเปรียบเทียบผลการcorrelation ที่ได้จากสัญญาณซายน์และสัญญาณรูปสี่เหลี่ยมที่มีแอมพลิจูด ±1 Vpp



เส้นสีเขียว : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ square (1,0) เฟส 0^{o}

เส้นสีแดง : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ square (1,-1) เฟส90^{o}

รูปที่ 7.6 แสดงการเปรียบเทียบผลการcorrelation ที่ได้จากสัญญาณซายน์และสัญญาณรูปสี่เหลี่ยมที่มีแอมพลิจูด 0 - 1 V



เส้นสีเขียว : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ square (0,-1) เฟส 0^{o}

เส้นสีแดง : ผลการ correlation ระหว่าง sine เฟส 0^{o} กับ square (1,-1) เฟส 0^{o}

รูปที่ 7.7 แสดงการเปรียบเทียบผลการcorrelation ที่ได้จากสัญญาณซายน์และสัญญาณรูปสี่เหลี่ยมที่มีแอมพลิจูด -1 - 0 V

พบว่าผลลัพธ์ที่ได้จากการ correlation ระหว่างสัญญาณอินพุทซายน์ กับสัญญาณซายน์และกับสัญญาณรูปสี่เหลี่ยม ± 1 Vpp นั้นมีลักษณะใกล้เคียงกันจึงสามารถใช้สัญญาณรูปสี่เหลี่ยมแทนสัญญาณซายน์ได้และเมื่อพิจารณาผลลัพธ์ที่ได้จากการ correlation ระหว่างสัญญาณอินพุท ซายน์ กับสัญญาณรูปสี่เหลี่ยมที่มีแอมพลิจูด 0 - 1 V ซึ่งสามารถเขียนโปรแกรมหาค่า inner product ได้ง่ายกว่าของสัญญาณรูปสี่เหลี่ยมแบบแรก นั้นพบว่ามีลักษณะที่แตกต่างกันอยู่บ้างแต่ผลลัพธ์ที่ได้ในกรณีที่สัญญาณทั้งมีเฟสตรงกันยังคงมีค่าแอมพลิจูดสูงสุดที่จุดกึ่งกลางเหมือนเดิม ซึ่งสอดคล้องกับหลักการที่กล่าวไว้ในบทที่ 6 ดังนั้นในส่วนของโปรแกรมจริงจะใช้วิธีการเปรียบเทียบค่า inner product ของสัญญาณอินพุทซายน์กับสัญญาณรูปสี่เหลี่ยมเฟส 0^{o} , 90^{o}, 180^{o} และ 270^{o} ถ้าผลลัพธ์ที่ได้จากเฟสใดมีค่า สูงสุดแสดงว่าสัญญาณอินพุทจะมีเฟสตรงกับสัญญาณสี่เหลี่ยมเฟสนั้น

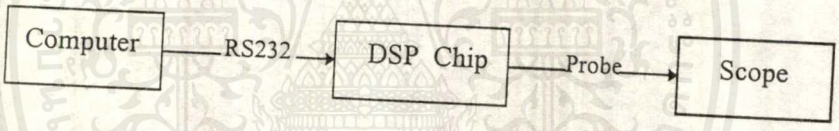
บทที่ 8

ผลการทดลอง

ในโครงงานนี้ได้เขียนโปรแกรมที่ใช้ในการมอดูเลชั่นและดีมอดูเลชั่นแบบ QPSK โดยจะ
ทำการส่งข้อมูล 2 บิท มากับสัญญาณซายน์ 1 ลูกครึ่ง โดยจะแบ่งการทดลองออกเป็น 2 ส่วน
หลักคือ

8.1 การทดลองส่วนมอดูเลชั่นในตัวส่ง

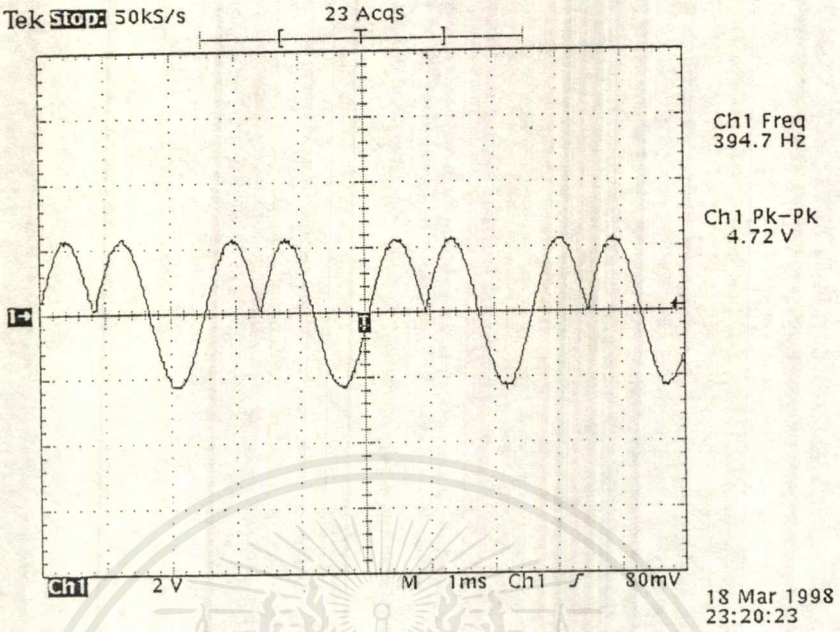
ทำการรันโปรแกรมของตัวส่ง แล้วนำสโครปจับที่เอาท์พุทของบอร์ดจะได้สัญญาณซายน์
ที่มีเฟสแตกต่างกันไปตามข้อมูลที่ต้องการส่ง บล็อกไดอะแกรมแสดงการทดลองแสดงไว้ดังรูป
ข้างล่าง



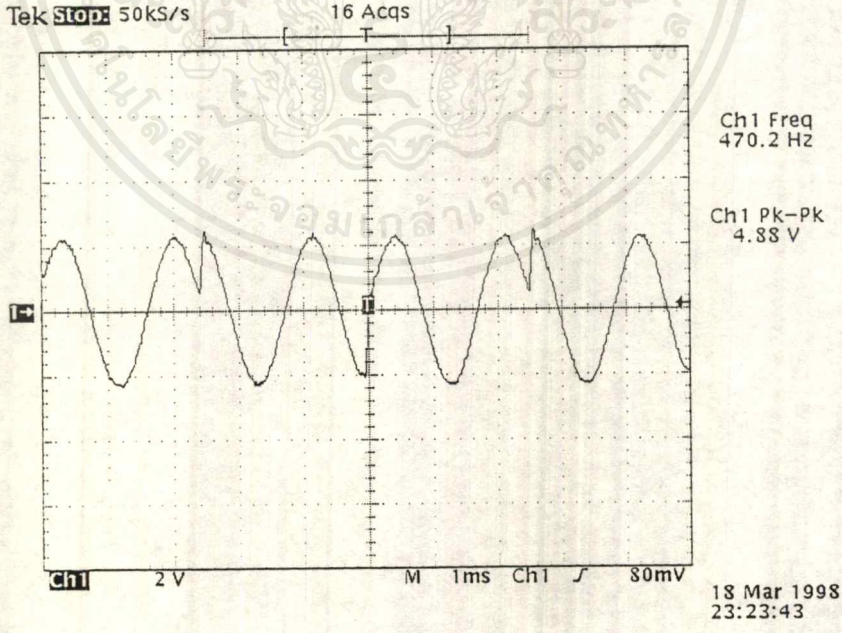
รูปที่ 8.1 บล็อกไดอะแกรมแสดงการทดลองโปรแกรมมอดูเลชั่นของตัวส่ง

ผลลัพธ์ที่ได้จากตัวส่งสามารถแสดงออกทางออสซิลโลสโคปได้ดังรูปข้างล่างนี้

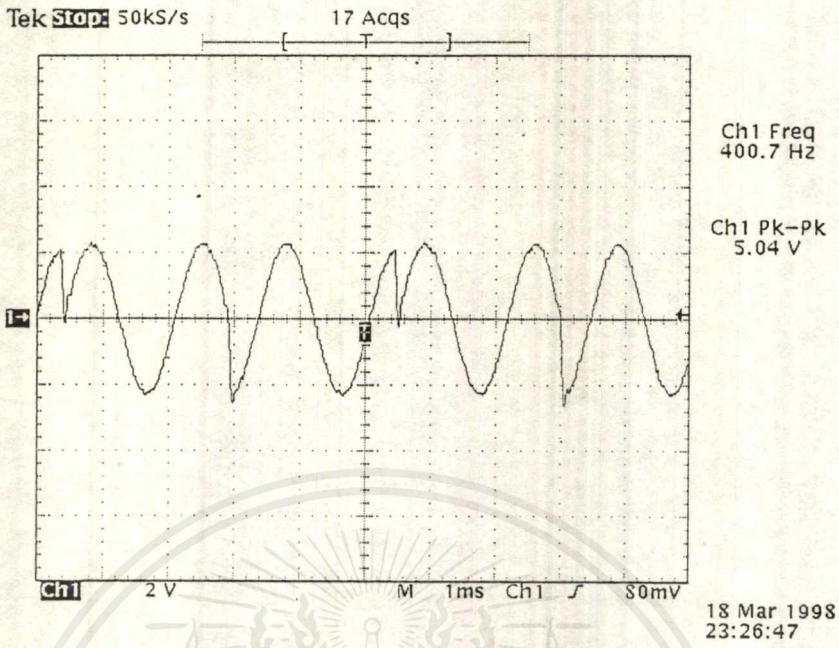
8.1.1 ทดลองโปรแกรมที่ตัวส่งจะวนลูปส่งข้อมูล 16 บิทซ้ำกันไป ใช้สัญญาณพาหะความถี่ 590 Hz โดยมีจำนวนจุดต่อลูกคลื่น = 32 จุด ความถี่ cut off ของ LPF ขาออก = 12.85 KHz, TA = 5 , TB = 53 สามารถส่งข้อมูลด้วยอัตรา 800 bit/s



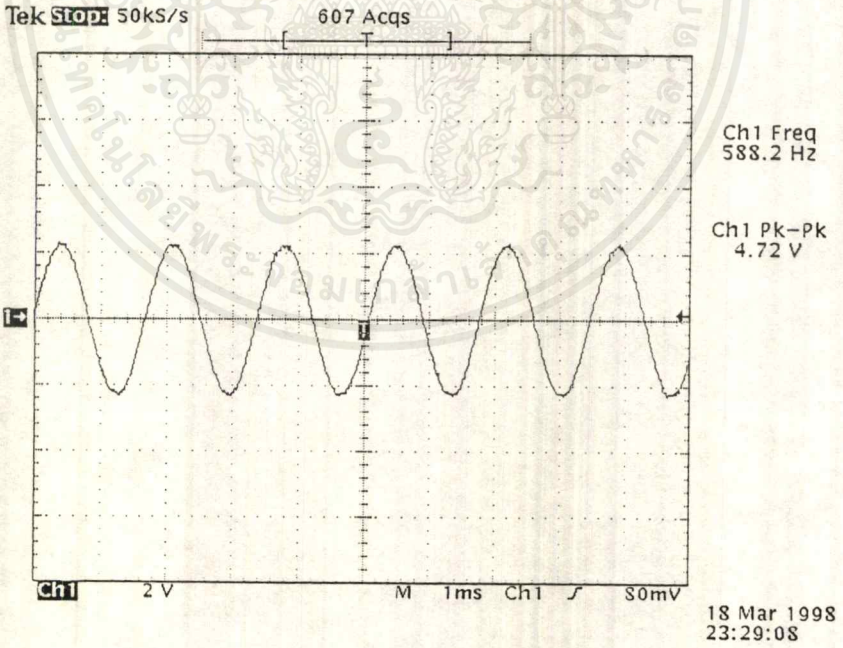
รูปที่ 8.2 แสดงภาพของสัญญาณที่ได้จากตัวส่งเมื่อสัญญาณพาหะถูกมอดูเลทด้วยข้อมูล "0000h"



รูปที่ 8.3 แสดงภาพของสัญญาณที่ได้จากตัวส่งเมื่อสัญญาณพาหะถูกมอดูเลทด้วยข้อมูล "1111h"



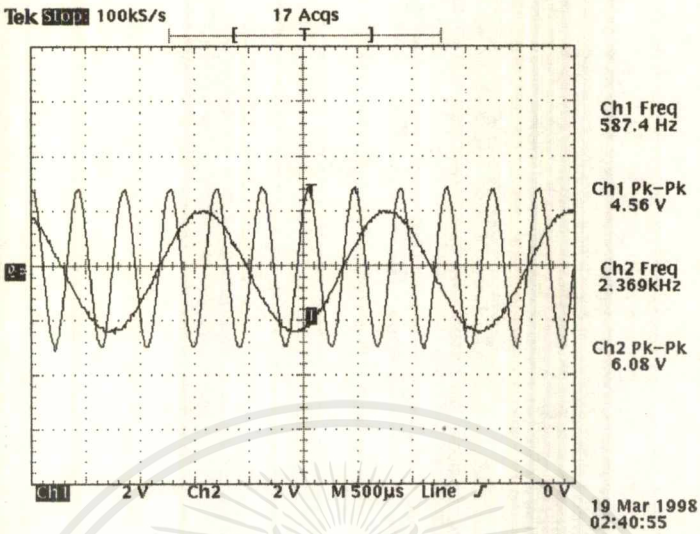
รูปที่ 8.4 แสดงภาพของสัญญาณที่ได้จากตัวส่งเมื่อสัญญาณพาหะถูกมอดูเลทด้วยข้อมูล “2222h”



รูปที่ 8.5 แสดงภาพของสัญญาณที่ได้จากตัวส่งเมื่อสัญญาณพาหะถูกมอดูเลทด้วยข้อมูล “3333h”

8.2 การทดลองส่วนดีมอดูเลชั่นในตัวรับ

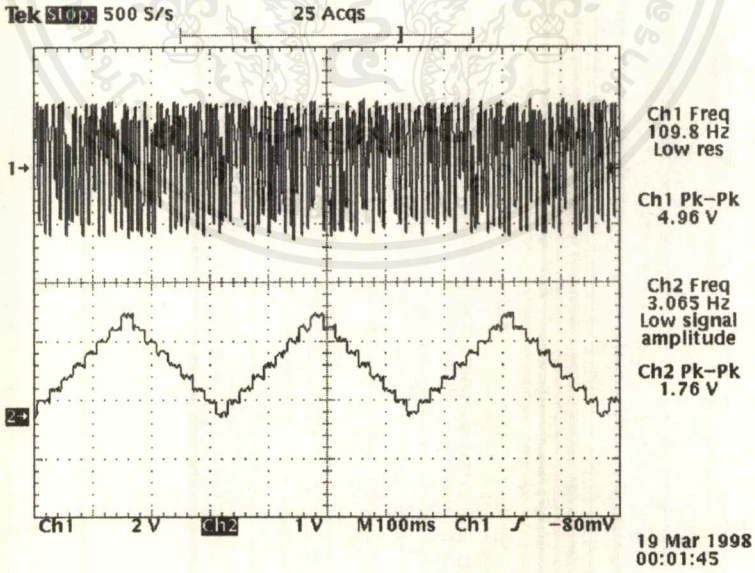
8.2.1 ทดลองโปรแกรมสังเคราะห์ sin4x



รูปที่ 8.6 แสดงสัญญาณที่มีความถี่เป็นสี่เท่าของสัญญาณพาหะที่สังเคราะห์ได้เปรียบเทียบกับสัญญาณอินพุท

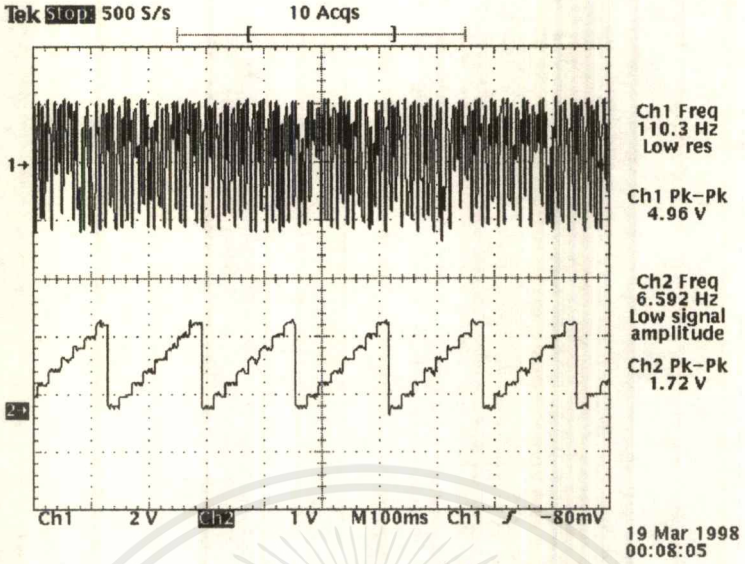
8.2.2 ทดลองการรับส่งข้อมูลจากตัวส่งไปยังตัวรับ

- การทดลองส่งข้อมูลรูปสามเหลี่ยมที่เก็บไว้ในหน่วยความจำของตัวรับไปยังตัวส่ง



รูปที่ 8.7 CH1 :แสดงสัญญาณอินพุทที่รับจากตัวส่ง
CH2 :แสดงสัญญาณรูปสามเหลี่ยมที่ดีเทคได้จากสัญญาณอินพุทที่รับมาจากตัวส่ง

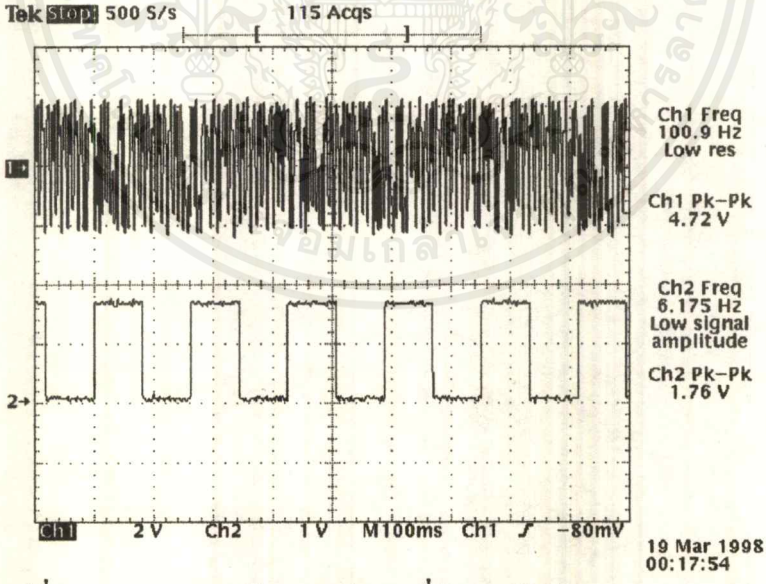
- การทดลองส่งข้อมูลรูปขั้นบันไดที่เก็บในหน่วยความจำของตัวส่งไปยังตัวรับ



รูปที่ 8.8 CH1 :แสดงสัญญาณอินพุทที่รับจากตัวส่ง

CH2 :แสดงสัญญาณขั้นบันไดที่ดีเทคได้จากสัญญาณอินพุทที่รับมาจากตัวส่ง

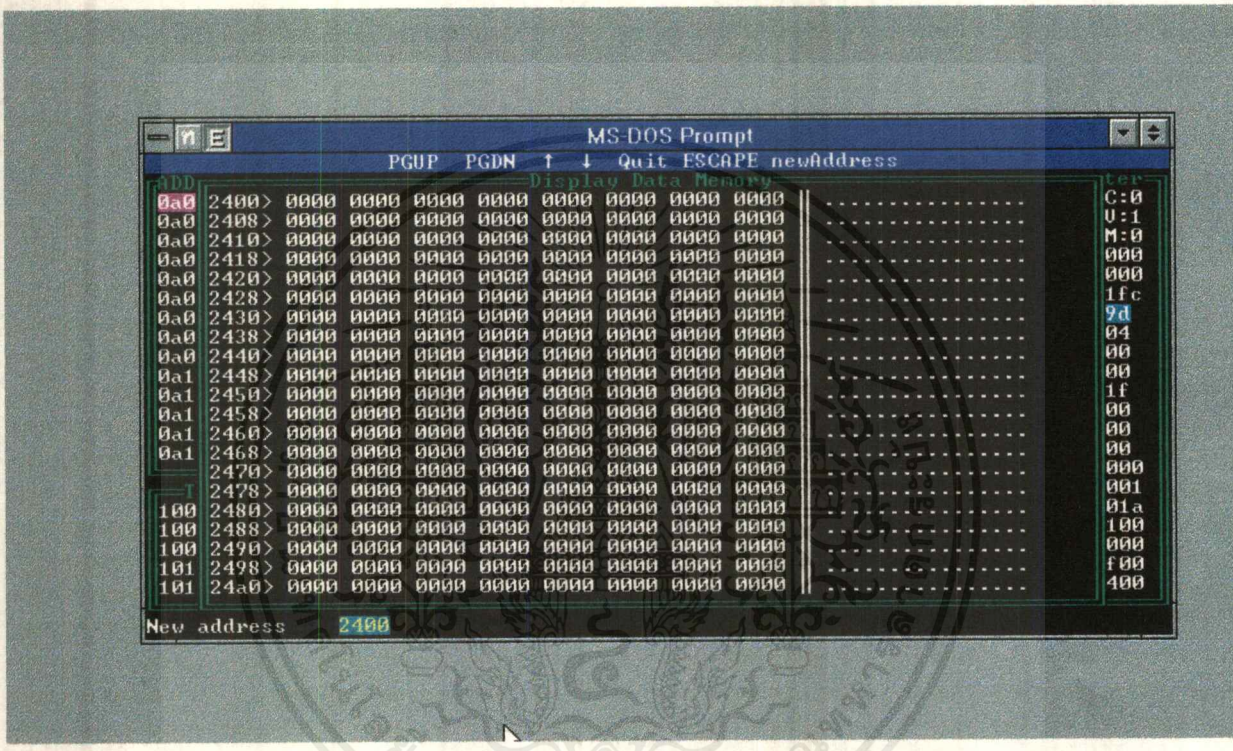
- การทดลองส่งข้อมูลรูปขั้นบันไดที่เก็บในหน่วยความจำของตัวส่งไปยังตัวรับ



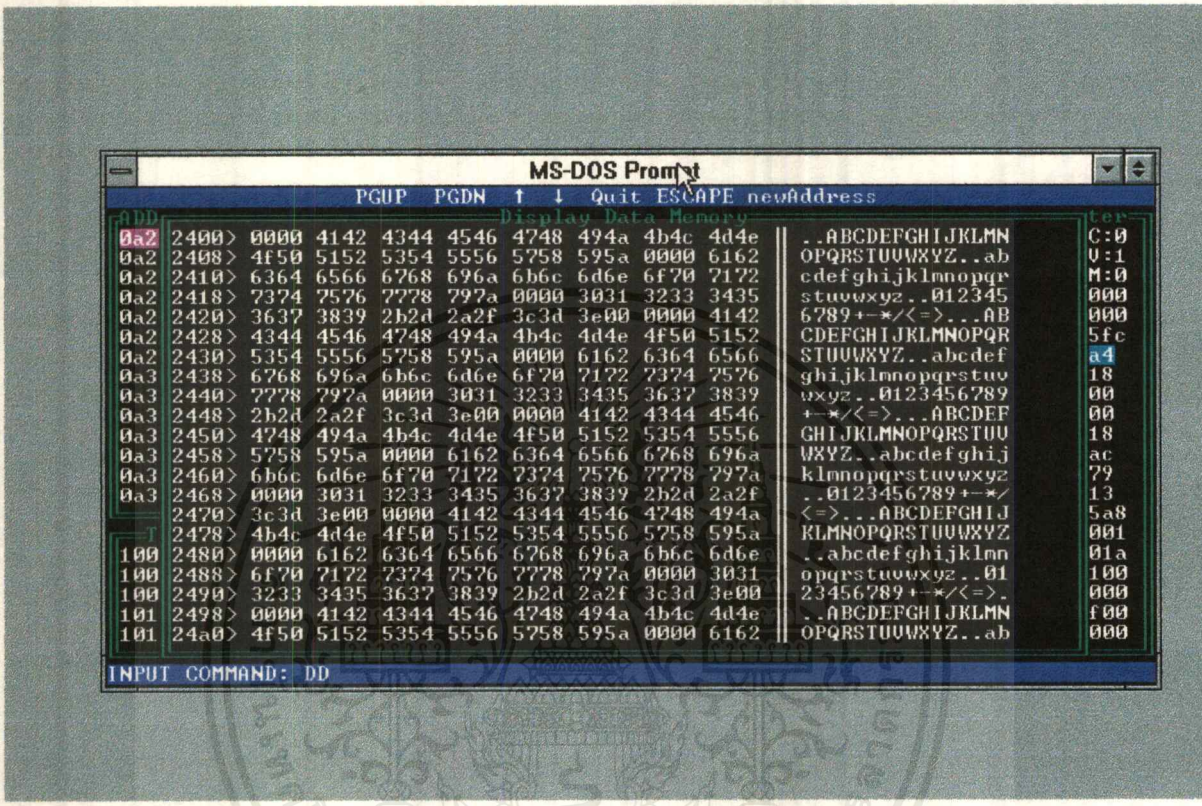
รูปที่ 8.9 CH1 :แสดงสัญญาณอินพุทที่รับจากตัวส่ง

CH2 :แสดงสัญญาณสี่เหลี่ยมที่ดีเทคได้จากสัญญาณอินพุทที่รับมาจากตัวส่ง

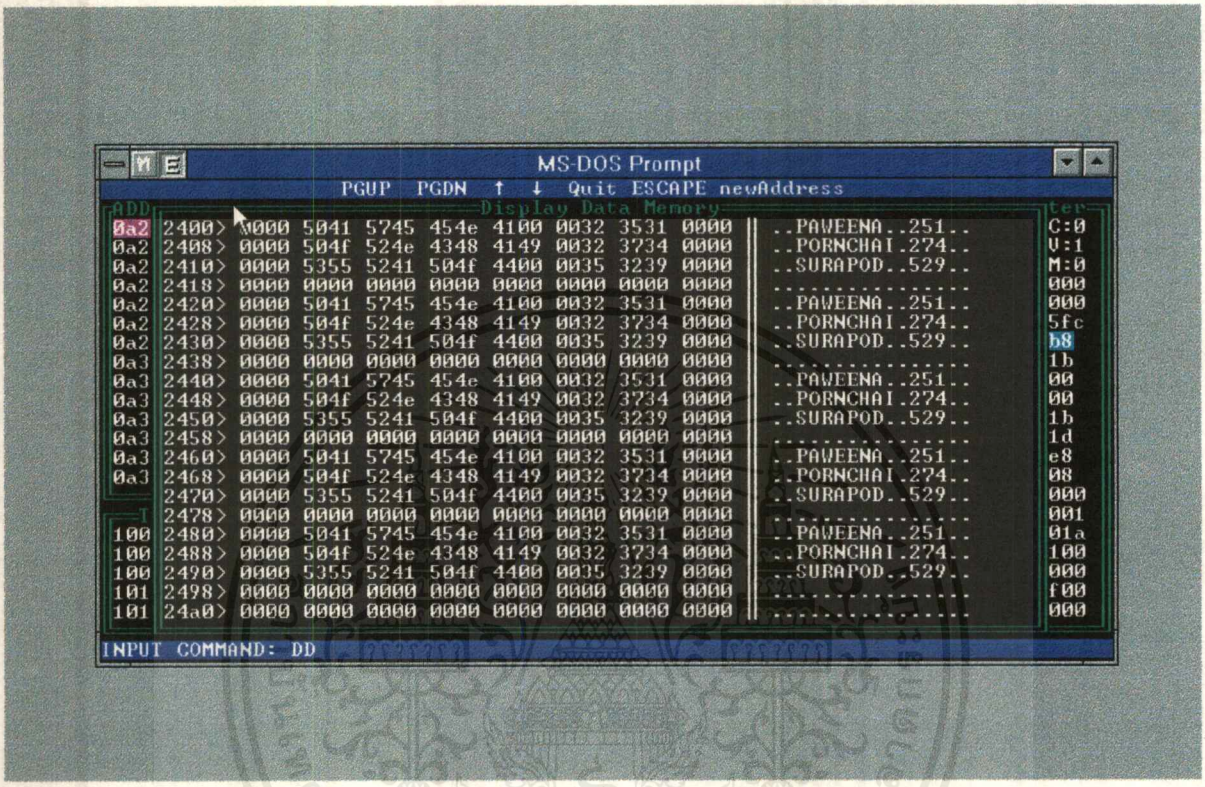
- การรับส่งข้อมูลที่เป็นตัวอักษรตั้งแต่ A - Z , a - z และสัญลักษณ์อื่นๆ



รูปที่ 8.10 ภาพแสดงหน่วยความจำก่อนที่ตัวรับจะรับข้อมูลจากตัวส่ง



รูปที่ 8.11 ภาพแสดงหน่วยความจำของตัวรับหลังจากที่รับข้อมูล A-Z , a-z และ สัญญลักษณ์อื่นๆมาจากตัวส่ง



รูปที่ 8.12 ภาพแสดงหน่วยความจำของตัวรับหลังจากที่ได้รับข้อมูลจากตัวส่งแล้ว

บทที่ 9
สรุปและวิจารณ์

โครงงานนี้สามารถรับส่งข้อมูลจากตัวส่งไปยังตัวรับโดยใช้หลักการมอดูเลชั่นและดีมอดูเลชั่นแบบควอเดรเจอร์เฟสชิฟคีย์อิงได้สำเร็จ โดยสามารถทำการรับส่งข้อมูลที่เป็นอักษรภาษาอังกฤษได้ครบทั้ง 26 ตัว ทั้งตัวเล็กและตัวใหญ่รวมทั้งสัญญลักษณ์อื่นด้วย ความผิดพลาดของข้อมูลจะเกิดเมื่อรันโปรแกรมตัวมอดูเลทเป็นเวลานาน ค่าเฟสที่ดีเทคได้จะผิดพลาดไปเนื่องมาจากค่าอินพุทที่ได้จากตัวสุ่มจะรวมเอาจุดปลาย 4-5 จุดของสัญญาณลูกก่อนหน้านี้มารวมด้วยทำให้ได้ค่า inner product ของสัญญาณไม่ถูกต้อง ส่งผลให้เฟสที่ดีเทคได้ไม่ตรงกับที่เป็นจริงข้อมูลที่ถูกแปลงกลับมาจึงผิด

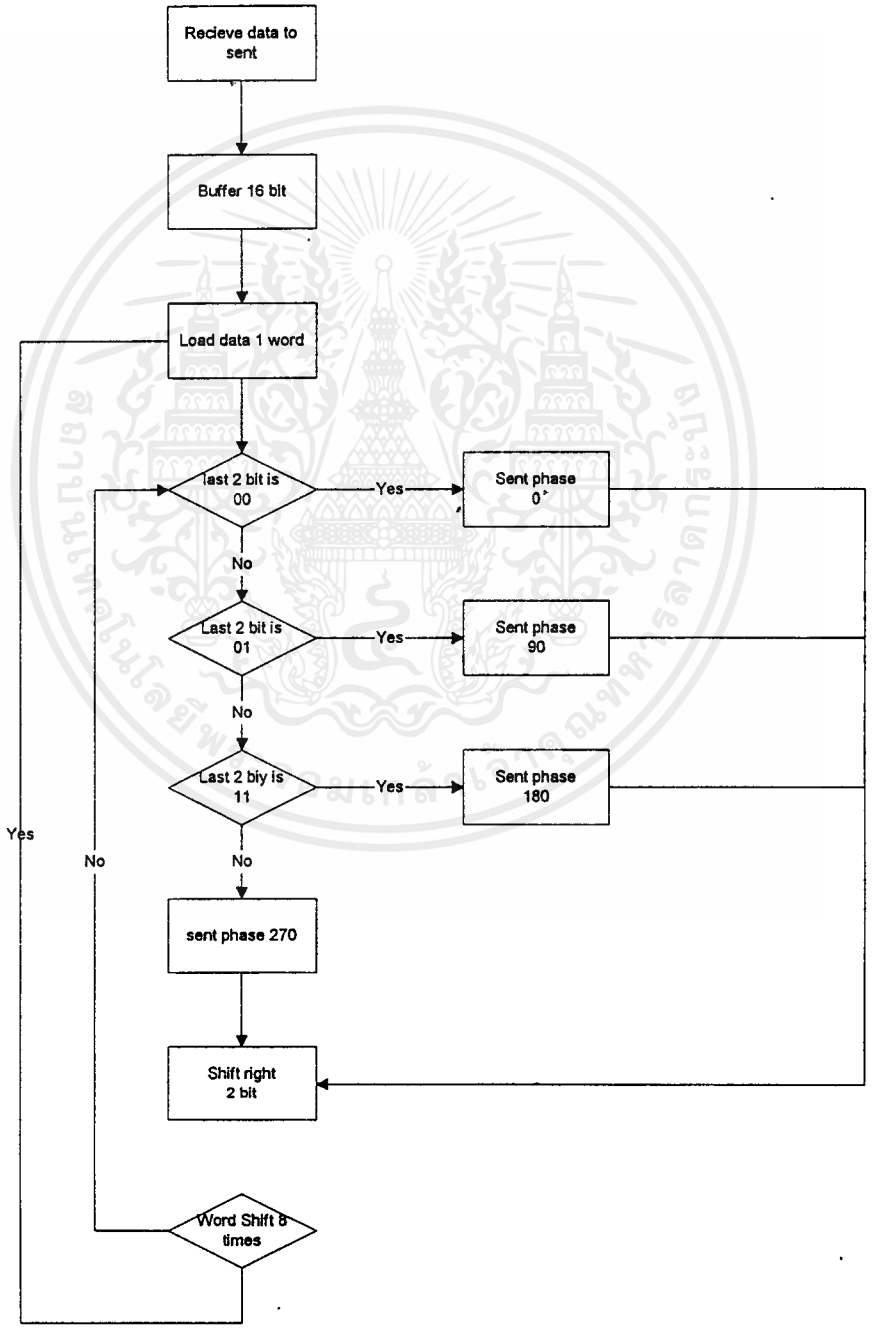
สัญญาณพาหะที่ได้จากตัวส่งจะมีความถี่ประมาณ 600 Hz โดยสามารถส่งข้อมูลด้วยอัตรา 800 bits/s ซึ่งเป็นอัตราส่งที่ค่อนข้างช้าการแก้ไขทำได้โดยใช้ A/D และ D/A จากภายนอกแทน ซึ่งในโครงงานนี้ได้ทดลองออกแบบและต่อวงจร AIC ขึ้นใหม่ รูปวงจรแสดงในภาคผนวก ข

แนวทางในการพัฒนาในส่วนของโครงงานนี้เพื่อให้ใช้งานกับความถี่ของสัญญาณพาหะได้กว้างขึ้นจึงได้สร้างส่วนสังเคราะห์ sin 4x ไว้เพื่อใช้เป็นฐานเวลาในการแบ่งสัญญาณอินพุทซายน์ 1 ลูกคลื่นออกเป็น 4 ส่วนเพื่อนำไปใช้ในการหาค่า inner product ต่อไป ซึ่งในกรณีดังกล่าวจะสามารถขยายแบนด์วิทธ์ของสัญญาณพาหะได้มากขึ้น

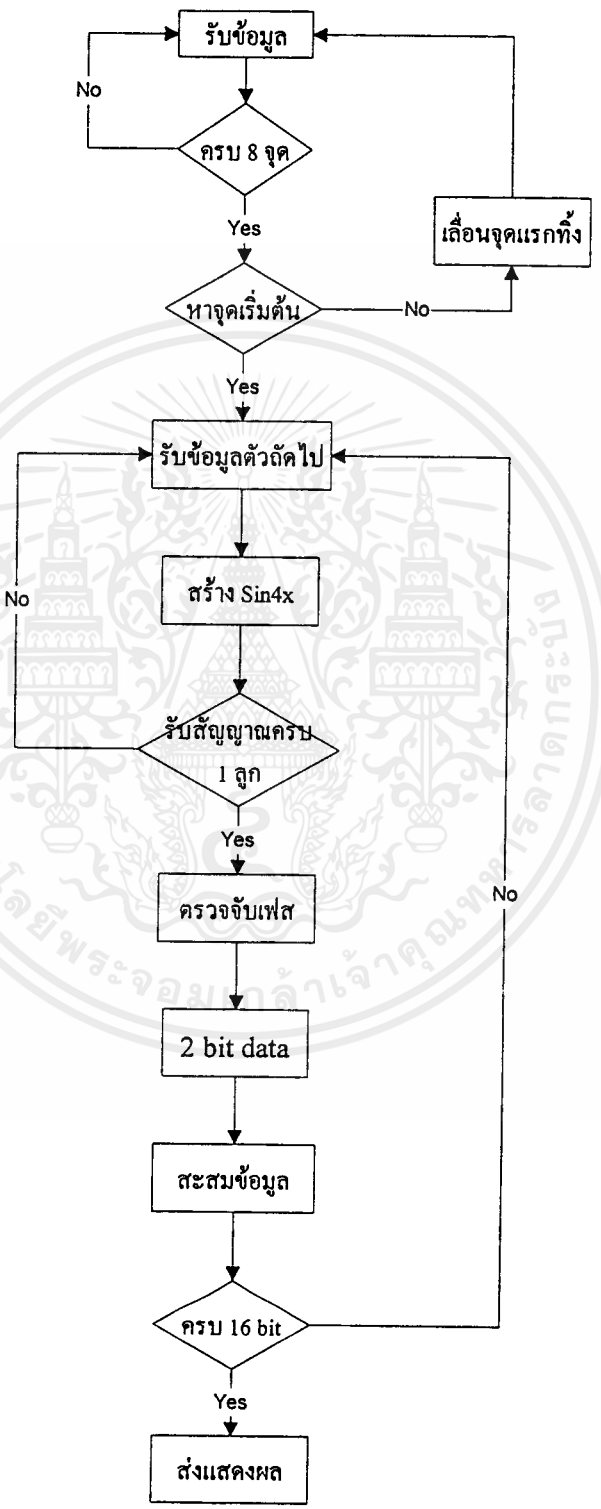
ภาคผนวก

ภาคผนวก ก
ส่วนโปรแกรมของโครงงาน

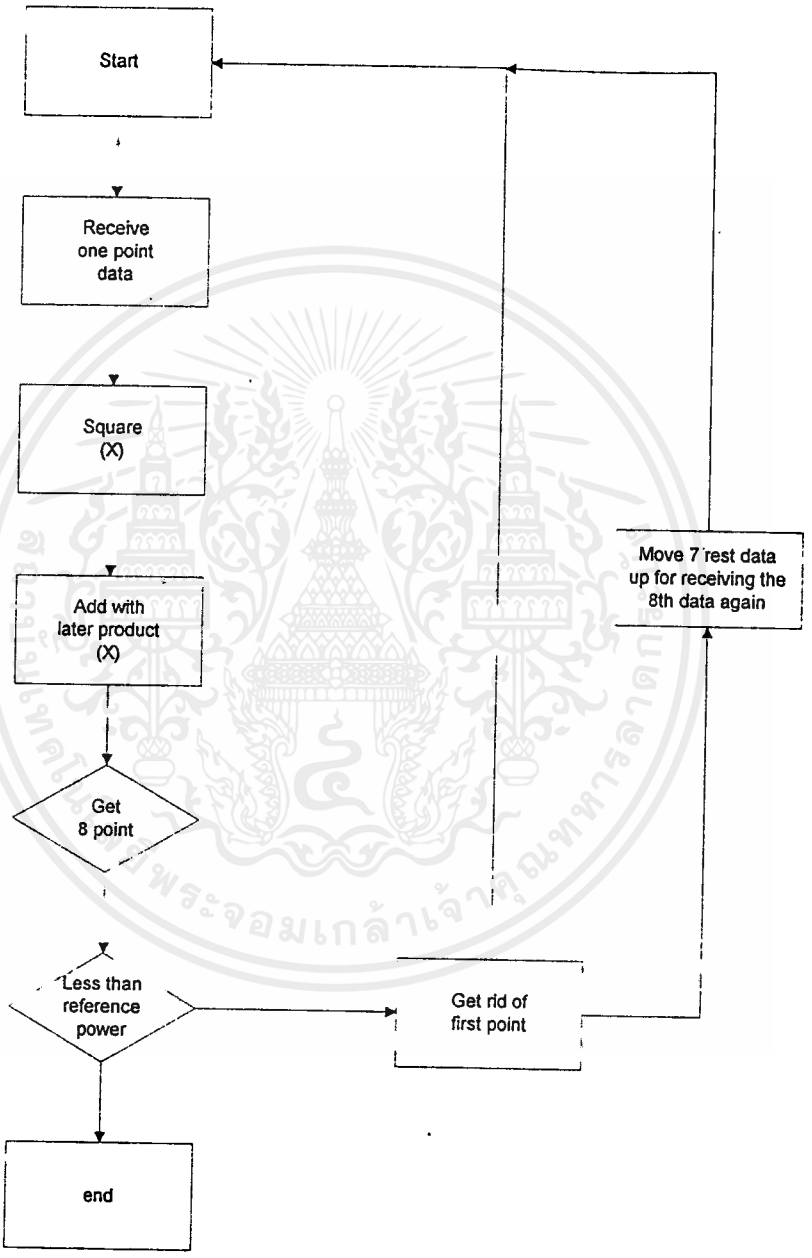
แผนภูมิที่ 1 แสดงการทำงานของโปรแกรมมอดูเลชั่นของตัวส่ง



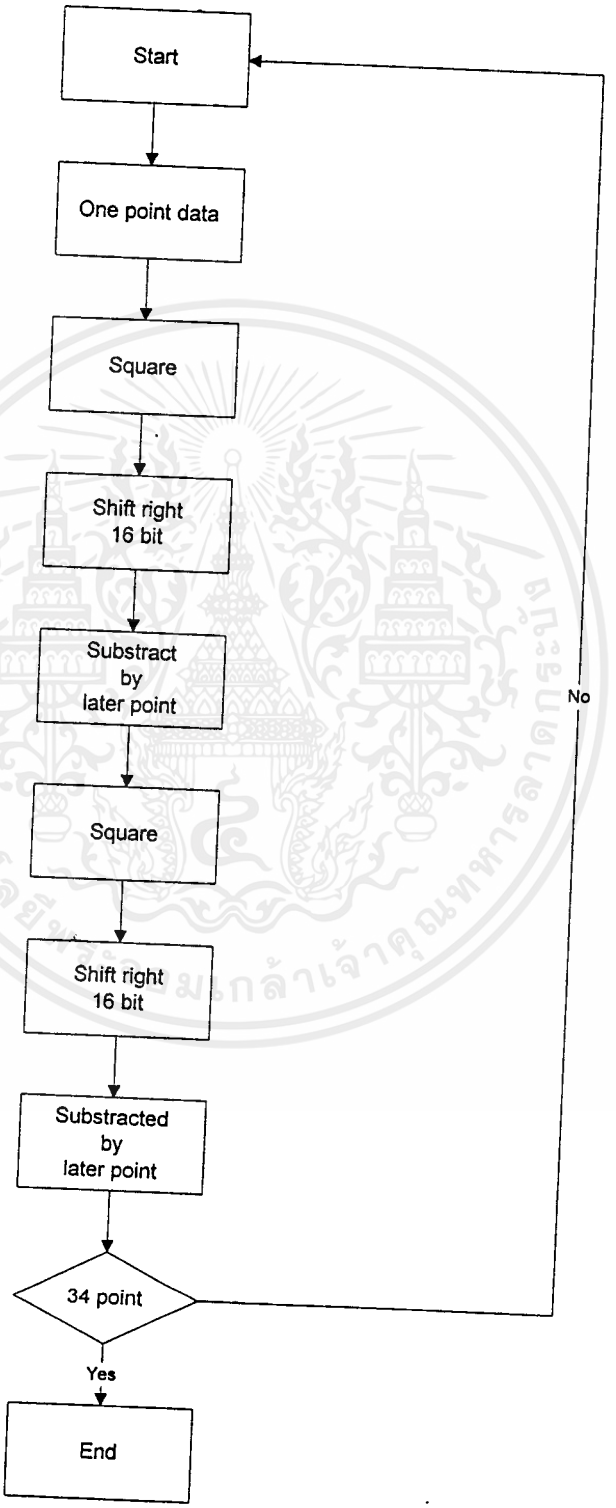
แผนภูมิที่ 2 แสดงการทำงานของโปรแกรมมอดูเลชั่นของตัวรับ



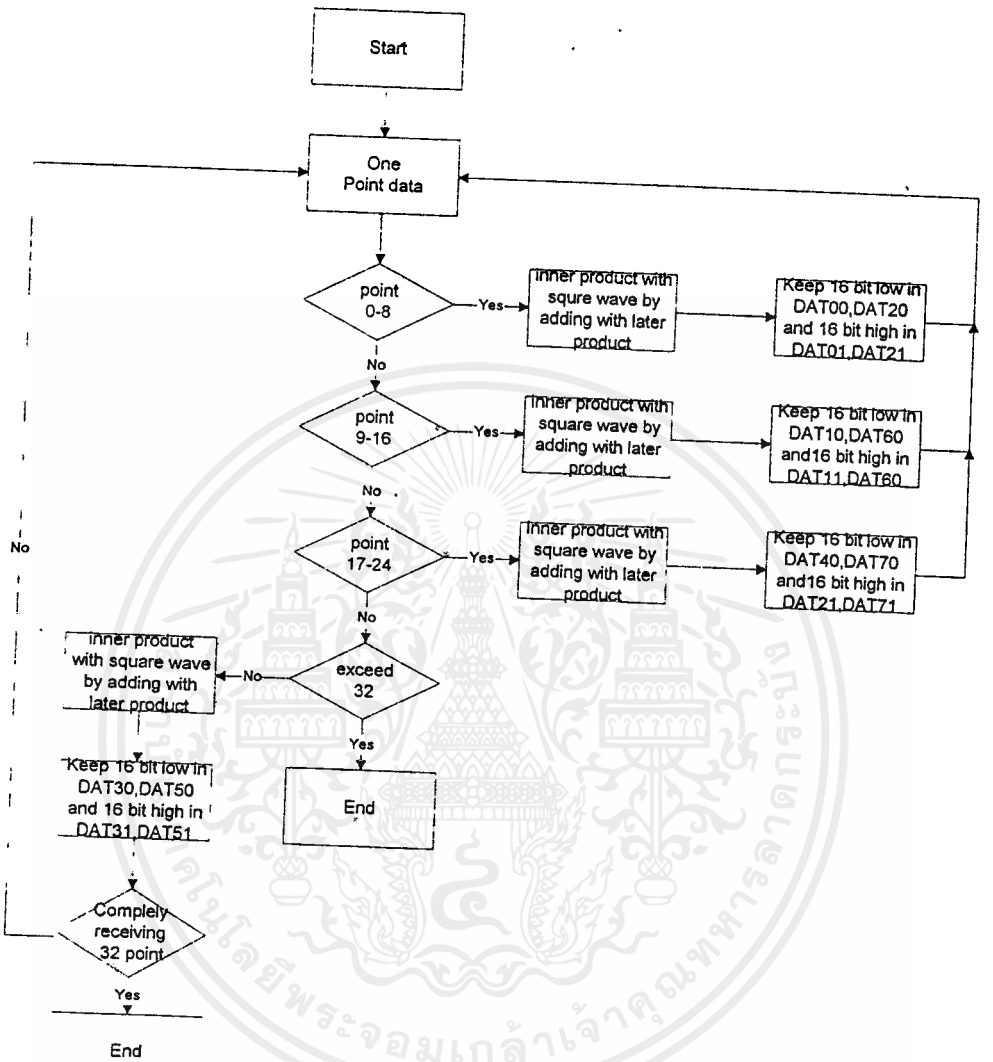
แผนภูมิที่ 3 แสดงส่วนเช็คจุดเริ่มต้นของสัญญาณด้วยวิธีเช็คพลังงาน (Power Deection)



แผนภูมิที่ 4 แสดงส่วนสังเคราะห์ sin4x



แผนภูมิที่ 5 แสดงส่วนที่ใช้หาค่า Inner Product เพื่อใช้ดีเทคเฟส



DAT00,DAT20 : 16 bit low of inner product of 1^{th} - 8^{th} point

DAT01,DAT21 : 16 bit high of inner product of 1^{th} - 8^{th} point

DAT10,DAT60 : 16 bit low of inner product of 9^{th} - 16^{th} point

DAT11,DAT61 : 16 bit low of inner product of 9^{th} - 16^{th} point

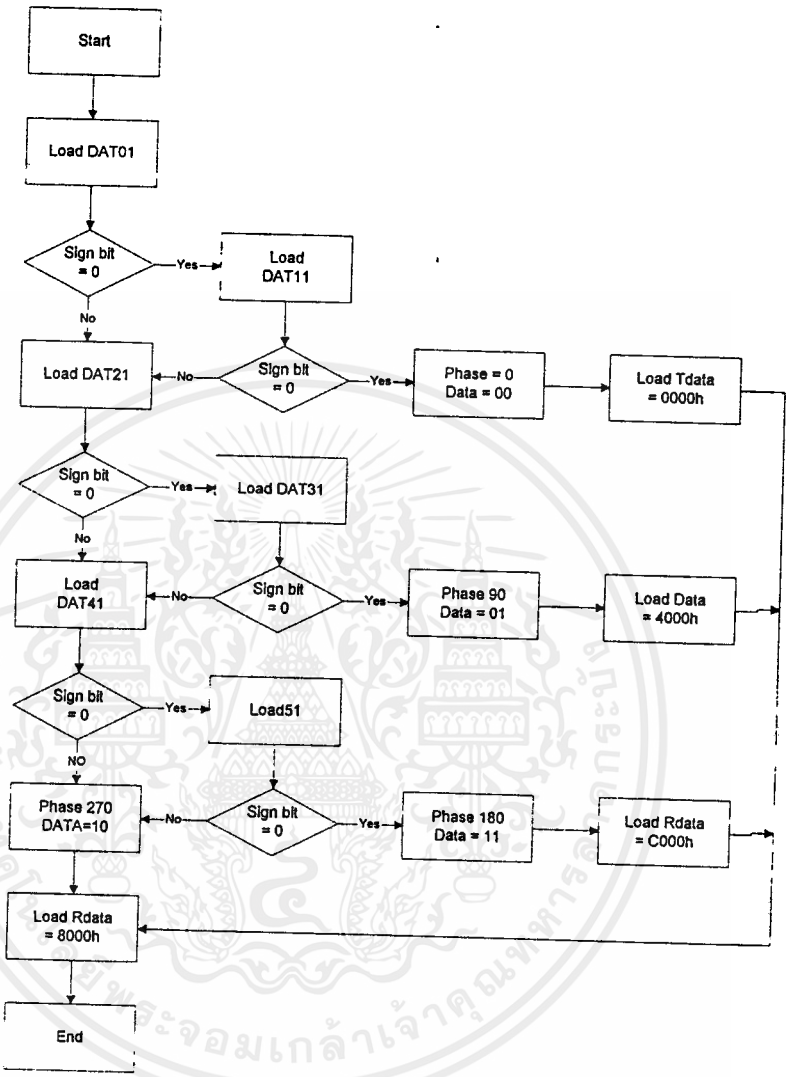
DAT40,DAT70 : 16 bit low of inner product of 17^{th} - 24^{th} point

DAT41,DAT71 : 16 bit low of inner product of 17^{th} - 24^{th} point

DAT30,DAT50 : 16 bit low of inner product of 25^{th} - 32^{th} point

DAT31,DAT71 : 16 bit low of inner product of 25^{th} - 32^{th} point

แผนภูมิที่ 6 แสดงส่วนดีเทค เฟส



RDATA : 16 bit memmory for keeping 2 bits detected data

DAT01 : 16 bit high of inner product of 1^{th} - 8^{th} point

DAT11 : 16 bit hight of inner product of 1^{th} - 8^{th} point

DAT21 : 16 bit high of inner product of 9^{th} - 16^{th} point

DAT31 : 16 bit high of inner product of 9^{th} - 16^{th} point

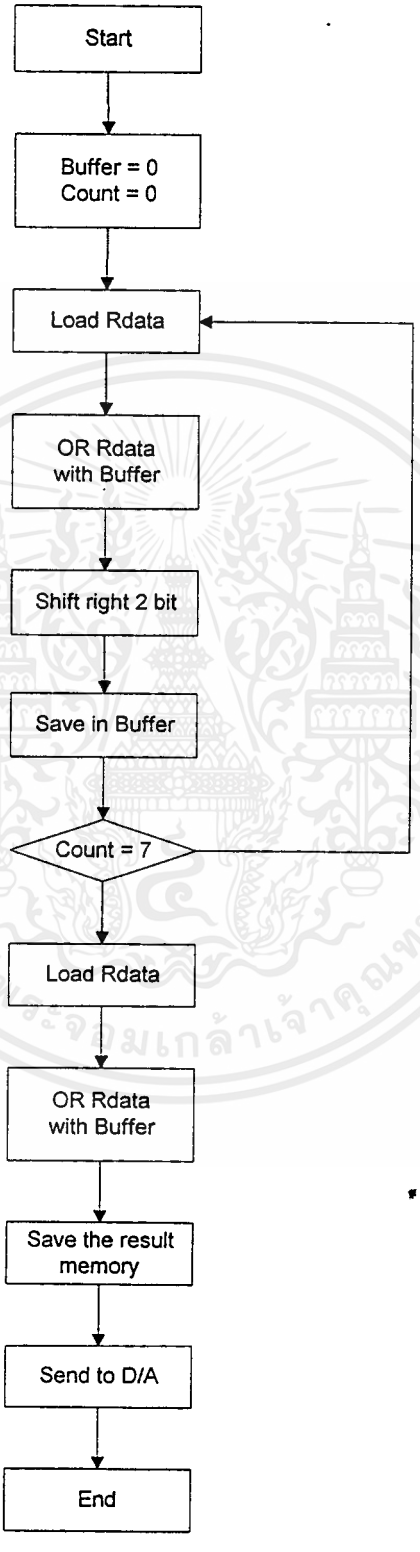
DAT41 : 16 bit high of inner product of 17^{th} - 24^{th} point

DAT51 : 16 bit hight of inner product of 17^{th} - 24^{th} point

DAT61 : 16 bit high of inner product of 25^{th} - 32^{th} point

DAT71 : 16 bit high of inner product of 25^{th} - 32^{th} point

แผนภูมิที่ 7 แสดงส่วน One Word Collection



Buffer : 16 bit memmory for collectiong one word data

RDATA : 16 bit memmory for keeping 2 bits detected data

โปรแกรมมอดูเลชั่นในส่วนภาคส่ง

```
* GENERRATE TRIANGLE -> MODULATE BY QPSK METHOD -> TRANSMIT
*************************************************************************
*                              CONSTANT                                 *
*************************************************************************
TA          .set    5           ; LP SCF  frequency  = 12.5 KHz
RA          .set    9           ; BP SCF  frequency  = 6.9 KHz
TB          .set    53          ; D/A Conversion  Frequency  =  18.87 KHz
RB          .set    29          ; A/D Conversion Frequency  =  18.87 KHz
AIC_CTR     .set    9h          ; d7,d6 = 00  >>> input gain = 1
                                ; d5 = 1   >>> synchronous transmit recieve sections
                                ; d4 = 0   >>> disables the AUX IN+ and AUX IN- pin
                                ; d3 = 0   >>> disables loopback function
                                ; d2 = 1   >>> inserts the bandpass filter
*************************************************************************
*                              VARIABLE                                 *
*************************************************************************
        .ds   02a00h
NEWDATA     .word   1
BUFFER      .word   0000h
SCOUNT      .word   0
SHF_C       .word   8
WORDTEMP    .word   0
```

```
*************************************************************************
*                          SINE TABLE                                   *
*************************************************************************
        .ds     02A80h
Phase0          .word   0,1065,2089,3033,3861,4540,5045,5355
                .word   5460,5355,5045,4540,3861,3033,2089,1065
                .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355
                .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065
                .word   0,1065,2089,3033,3861,4540,5045,5355
                .word   5460,5355,5045,4540,3861,3033,2089,1065

Phase90         .word   5460,5355,5045,4540,3861,3033,2089,1065
                .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355
                .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065
                .word   0,1065,2089,3033,3861,4540,5045,5355
                .word   5460,5355,5045,4540,3861,3033,2089,1065
                .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355

Phase180        .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355
                .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065
                .word   0,1065,2089,3033,3861,4540,5045,5355
                .word   5460,5355,5045,4540,3861,3033,2089,1065
                .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355
                .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065

Phase270        .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065
                .word   0,1065,2089,3033,3861,4540,5045,5355
                .word   5460,5355,5045,4540,3861,3033,2089,1065
                .word   0,-1065,-2089,-3033,-3861,-4540,-5045,-5355
                .word   -5460,-5355,-5045,-4540,-3861,-3033,-2089,-1065
```

```
.word   0,1065,2089,3033,3861,4540,5045,5355

        .mmregs

*************************************************************************
*                          Set up the ISR vector                        *
*************************************************************************
        .ps     080Ch
;rint:  B       RECEIVE         ; Serial port receive interrupt RINT.
xint:   B       TRANSMIT        ; Serial port transmit interrupt XINT.

*************************************************************************
*                    TMS32C05X  INITIALIZATION                          *
*************************************************************************
        .ps 0a00h
        .entry
START:  SETC  INTM              ; Disable interrupts
        LDP   #0
        OPL   #0834h,PMST
        SPLK  #0h,CWSR          ; Set software wait state to 0
        SPLK  #0h,PDWSR                 ; Set Program and Data wait state to 0
        SPLK  #022h,IMR         ; Using XINT syn TX & RX
        SPLK  #020h,TCR         ; set timer to generate 10 MHz clock to AIC
        SPLK  #01h,PRD          ; set period register to 1
        SPLK  #08h,SPC          ; set serial port for non continuous mode
        SPLK  #0C8h,SPC         ; enable serial port
        LDP   #84
        CALL  AICINIT           ; initialize AIC and enable interrupts
```

```
*************************************************************************
*                This routine enables serial port rx interrupts & configures *
*               TLC32040 for the frame sync. When RINT is triggered, read a *
*               dummy data word from the AIC then generate a sine wave to *
*               send out.                                                  *
*************************************************************************

        CLRC    OVM             ; OVM = 0
        SPM     0               ; PM = 0
        LDP     #84
        LACC    #2A80h
        SAMM    AR1
        LACC    #2AC0h
        SAMM    ARCR
        LACC    #0
        SAMM    DXR
        MAR     *,AR3
        LAR     AR3,#2900h
        CLRC    INTM            ; enable interrupt
        LDP     #84
        LACC    *               ; load data from memmory
        MAR     *+
        SACL    WORDTEMP
buffloop: LACC  NEWDATA
        ADD     #0
        NOP
        NOP
        BCND    olddata,EQ
        LACC    WORDTEMP
        AND     #0003h          ; transmit 2 LSB first.
```

```
        ADD     #0
        BCND    cond2,NEQ           ; if this 2 bit = 00 , send Phase 0°
        LACC    #Phase0
        SAMM    AR1
        NOP
        NOP
        B       endaddr
cond2:  SUB     #1
        BCND    cond3,NEQ           ; if this 2 bit = 01 , send Phase 90°
        LACC    #Phase90
        SAMM    AR1
        NOP
        NOP
        B       endaddr
cond3:  SUB     #1
        BCND    cond4,NEQ           ; if this 2 bit = 10 , send Phase 270°
        LACC    #Phase270
        SAMM    AR1
        NOP
        NOP
        B       endaddr
cond4:  LACC    #Phase180           ; if this 2 bit = 11 , send Phase 180°
        SAMM    AR1
        NOP
        NOP

endaddr: LAMM   AR1
        ADD     #48
        SAMM    ARCR
        LACC    #0
```

```
        SACL    NEWDATA
        LACC    WORDTEMP
        CLRC    C
        ROR
        ROR                             ; shift right for the next two bits
        SACL    WORDTEMP
        LACC    SHF_C
        SUB     #1
        SACL    SHF_C
        BCND    olddata,NEQ             ; if send completely (16 bit), load
        LACC    SCOUNT                  ; new data.
        ADD     #1
        SACL    SCOUNT
        SUB     #8
        BCND    NOTFIN,NEQ
        LACC    #0000
        SACL    BUFFER
        SACL    WORDTEMP
        SACL    SCOUNT
        B       AGAIN

NOTFIN: LACC    BUFFER
        ADD     #400h,0
        SACL    BUFFER
        SACL    WORDTEMP
AGAIN:  LACC    BUFFER
        SACL    BUFFER
        LACC    #8
        SACL    SHF_C
        NOP
```

```
        NOP
olddata: IDLE
        B       buffloop

;------- end of main program ----------
TRANSMIT:   LDP     #84
            LACC    *+,2        ;LOAD ACC WITH DATA TO SEND
            SAMM    DXR         ; SEND DATA
            CMPR    0           ; CHECK AR1 = LAST POINT ?
            BCND    OUT,NTC     ; NO : GOTO OUT_LOOP
            LACC    #1
            SACL    NEWDATA
OUT:        RETE

*********************************************************************************
*                          AIC Initialization                                   *
*********************************************************************************

AICINIT: MAR    *,AR0
        LACC    #0
        SAMM    DXR             ; Reset  AIC  by enable grobal memory and
        LACC    #080H           ;  load data in grobal memory to ACC 10000
        SAMM    GREG            ; times. This will make  BR pin low about
        LAR     AR0,#0FFFFh     ; 0.5ms. Since  AIC reset pin connect with
        RPT     #10000          ; BR, hence AIC will be in reset state as long
        LACC    *,0,AR0         ; as grobal memory communication has take
                                ;  place.
       LACC     #0              ; When  reset  finish , disables grobal memory
        SAMM    GREG
        SETC    SXM
```

```
        LACC    #TA,9           ; Initialized TA and RA register
        ADD     #RA,2
        CALL    AIC_2ND
        LACC    #TB,9           ; Initialized TB and RB register
        ADD     #RB,2
        ADD     #02h
        CALL    AIC_2ND
        LACC    #AIC_CTR,2      ; Initialized control register
        ADD     #03h
        CALL    AIC_2ND
        RET

AIC_2ND:LDP     #0
        SACH    DXR
        CLRC    INTM
        IDLE
        ADD     #6h,15
        SACH    DXR
        IDLE
        SACL    DXR
        IDLE
        LACL    #0
        SACL    DXR             ; make sure the word got sent
        IDLE
        SETC    INTM
        RET
        .end
```

ส่วนโปรแกรมดีมอดูเลชั่นของภาครับ

```
** RECEIVE "CONTINUE"  SIGNAL(START NO SIGNAL)
*-> TRANSMIT   THE DETECTED DATA

*************************************************************************
*                              CONSTANT                                 *
*************************************************************************
        TA          .set   5          ; LP SCF frequency = 12.5 KHz
        RA          .set   5          ; BP SCF frequency = 12.5 KHz
        TB          .set   53         ; D/A Conversion Frequency =  18.87 KHz
        RB          .set   53         ; A/D Conversion Frequency =  18.87 KHz
      . AIC_CTR     .set   .8         ; d7,d6 = 00  >>> input gain = 1
                                      ; d5 = 1   >>> synchronous transmit recieve sections
                                      ; d4 = 0   >>> disables the AUX IN+ and AUX IN-
                                      ; d3 = 0   >>> disables loopback function
                                      ; d2 = 1   >>> inserts the bandpass filter

*************************************************************************
*                                DATA                                   *
*************************************************************************
        .ds 2a50h
        DAT00       .word 0
        DAT01       .word 0
        DAT10       .word 0
        DAT11       .word 0
        DAT20       .word 0
        DAT21       .word 0
        DAT30       .word 0
```

```
DAT31       .word 0
DAT40       .word 0
DAT41       .word 0
DAT50       .word 0
DAT51       .word 0
DAT60       .word 0
DAT61       .word 0
DAT70       .word 0
DAT71       .word 0
RDATA       .word 0
DBUFFER     .word 0
FIXDAT      .word 0
COUNT       .word 8
PWBIT       .word 1
TAR0        .word 1
TAR1        .word 0
TAR2        .word 0
REF         .word 2000
RFLAG       .word 0
TFLAG       .word 0
INIT_F      .word 0
TEST        .word 0

*****************************************************************************
*                          Set up the ISR vector                            *
*****************************************************************************
        .ps     080ch
xint:   B       TRANSMIT        ; Serial port transmit interrupt XINT.
        .ps     080ah
rint:   B       RECEIVE         ; Serial port receive interrupt RINT
```

```
*************************************************************************
*                    TMS32C05X  INITIALIZATION                          *
*************************************************************************
        .ps     0a00h
        .entry
        .mmregs
STRT:   SETC    INTM            ; Disable interrupts
        LDP     #0
        OPL     #0834h,PMST
        SPLK    #0h,CWSR        ; Set software wait state to 0
        SPLK    #0h,PDWSR       ; Set Program and Data wait state to 0
        SPLK    #032h,IMR       ; Using XINT , RIXT syn TX & RX
        SPLK    #020h,TCR       ; set timer to generate 10 MHz
                                ; clock to AIC
        SPLK    #01h,PRD        ; set period register to 1
        SPLK    #08h,SPC        ; set serial port for non continuous mode.
        SPLK    #0C8h,SPC       ; enable serial port
        SPLK    #0100h,INDX
        LDP     #84
        CALL    AICINIT         ; initialize AIC and enable
                                ; interrupt.
        LAR   AR1,#2800h
        LAR   AR4,#28FFh        ; for receiving data in receive routine
        LAR   AR5,#2400h
        LAR   AR6,#2500h
        LACC  #28FFh
        SAMM  ARCR
        NOP
        NOP
        CLRC  INTM              ; enable interrupts
```

```
*************************************************************************
*                              MAIN                                     *
*************************************************************************
STRTLP: NOP
        NOP
        IDLE
        NOP
        NOP
        BIT    RFLAG,15
        BCND   STRTLP,NTC
        LACC   TAR0                    ; count the number of data cycle
        ADD    #1
        SACL   TAR0
        LACC   TAR2
        ADD    #1
        SACL   TAR2
        LACC   #0
        SACL   RFLAG
        B      STRTLP

*************************************************************************
*                          INTERRUPT ROUTINE                            *
*************************************************************************
TRANSMIT:   BIT    INIT_F,15
            BCND   OUT,NTC
OUT:        RETE
*************************************************************************
RECEIVE:    LACC   #28FFh
            SAMM   ARCR
            MAR    *,AR4
```

```
        LAMM  DRR
        NOP
        NOP
        SACL  *
        CMPR  0
        BCND  GETRID,TC
        LACC  #2908h
        SAMM  ARCR
        NOP
        NOP
        CMPR  1                    ; if AR4 >= 2308h , jump to PART2
        BCND PART2,NTC
*****************************************************************************
*                        POWER  DETECTION                                   *
*****************************************************************************
        BIT    PWBIT,15
        BCND  SKIPPW,NTC
        CLRC  SXM
        NOP
        NOP
        MAR   *,AR4
        LACC  DAT01,16             ; shift to ACCH
        SACB                       ; ACCB = (DAT01)0000
        LACL  DAT00                ; ACC = 0000(DAT00)
        ORB                        ; ACC = (DAT01)(DAT00)
        SACB
        LACL  *
        ADDB
        SACL  DAT00
        SACH  DAT01
```

```
        SACL  DAT20
        SACH  DAT21
        SPM   0
        ZAP
        SQRA  *+,AR1
        PAC
        CLRC  SXM
        RPT   #15
        SFR
        SACL  *+,0,AR4          ; start at 2800h
        ADD   TAR1
        SACL  TAR1
        CMPR  0
        BCND  BACK,NTC
        MAR   *,AR0
        LACC  REF
        SAMM  ARCR              ; ARCR = reference power
        NOP
        NOP
        LACC  TAR1
        SAMM  AR0
        NOP
        NOP
        CMPR  1
        BCND  PRECAL,NTC        ; if power > = reference power ,
SHIFT:  LAR   AR4,#2900h        ; receive 9th point.
        LAR   AR1,#2800h
        LAR   AR7,#8
        MAR   *,AR4
        LACC  DAT01,16
```

```
        SACB
        LACL  DAT00
        ORB
        SACB
        LACC  *+,0,AR1
        EXAR
        SBB
        SACL  DAT00
        SACH  DAT01
        SACL  DAT20
        SACH  DAT21
        LACC  *+,0,AR7
        SACB
        LACC  TAR1
        SBB                          ; ACC-ACB -> ACC
        SACL  TAR1
PSHIFT: MAR   *-,AR1
        BLDD  *-,#2A70h
        BLDD  #2A70h,*+
        MAR   *+,AR4
        BLDD  *-,#2A71h
        BLDD  #2A71h,*+
        MAR   *+,AR7
        BANZ  PSHIFT
        LAR   AR1,#2807h
        LAR   AR4,#2907h
        B     BACK
SKIPPW: MAR *,AR4
        LACC DAT01,16
        SACB
```

```
        LACL  DAT00
        ORB
        SACB
        LACL  *
        ADDB
        SACL  DAT00
        SACH  DAT01
        SACL  DAT20
        SACH  DAT21
        SPM   0
        ZAP
        SQRA  *+,AR1
        PAC                     ; send product to ACC
        RPT   #15               ; decrease data by devide 2^16
        SFR
        SACL  *+,0,AR4
        B     BACK

PART2:  MAR   *,AR4             ; separate sine wave into 4 parts
        LACC  #2910h            ; if it is 9th - 16th , jump to ZONE2
        SAMM  ARCR
        NOP
        NOP
        CMPR  1
        BCND  ZONE2,TC
        LACC  #2918h
        SAMM  ARCR              ; if it is 17th - 24th , jump to ZONE3
        NOP
        NOP
```

```
        CMPR  1
        BCND  ZONE3,TC
        LACC  #2920h
        SAMM ARCR                    ; if it is 15th - 32nd , jump to ZONE4
        NOP
        NOP
        CMPR  1
        BCND  SIN2X,NTC

ZONE4:LACC  DAT31,16
        SACB
        LACL  DAT30
        ORB
        SACB
        LACC  *
        ADDB
        SACL  DAT30
        SACH  DAT31
        SACL  DAT50
        SACH  DAT51
        B     SIN2X

ZONE3: LACC DAT41,16
        SACB
        LACL  DAT40
        ORB
        SACB
        LACC  *
        ADDB
        SACL  DAT40
```

```
        SACH  DAT41
        SACL  DAT70
        SACH  DAT71
        B     SIN2X

ZONE2:LACC  DAT11,16
        SACB
        LACL  DAT10
        ORB
        SACB
        LACC  *
        ADDB
        SACL  DAT10
        SACH  DAT11
        SACL  DAT60
        SACH  DAT61

SIN2X: LACC  #2922h
        SAMM ARCR
        SPM   0
        ZAP
        SQRA  *+,AR1
        PAC
        RPT   #15
        SFR
        SACL  *+,0,AR4
        CMPR  0
        BCND  CHECK,NTC
```

```
*************************************************************************
*                    BIT DETECTION FOR FINDING PHASE                    *
*************************************************************************
        MAR   *,AR6
        LACC  #0000h
GOON0:bit     DAT01,13
        NOP
        NOP
        BCND  GOON90,TC                    ; TC=1 -> NEG
        BIT   DAT11,13
        NOP
        NOP
        BCND  GOON90,TC
        LACC  #0000
        SACL  RDATA
        SACL  *
        B     COLLECT
GOON90:     BIT   DAT21,13
            NOP
            NOP
            BCND  GOON18,TC                ; TC=1 -> NEG
        BIT   DAT31,13
        NOP
        NOP
        BCND  GOON18,TC
        LACC  #4000h
        AND   #0C000h
        SACL  RDATA
        SACL  *
        B     COLLECT
```

```
GOON18: BIT  DAT41,13
        NOP
        NOP
        BCND GOON27,TC                    ; TC=1 -> NEG
        BIT  DAT51,13
        NOP
        NOP
        BCND GOON27,TC
        LACC #0C000h
        AND  #0C000h
        SACL RDATA
        SACL *
        B    COLLECT
GOON27: LACC #8000h
        AND  #0C000h
        SACL RDATA
        SACL *
*************************************************************************
*                          COLLECT DATA                                 *
*************************************************************************
COLLECT: MAR *+,AR7
        CLRC SXM
        LACC COUNT
        SAMM ARCR
        LAR  AR7,TAR0
        NOP
        NOP
        CMPR 1
        BCND COMPLETE,NTC
        MAR  *,AR5
```

```
        CLRC  C
        LACC  #0000h
        LACL  RDATA
        OR    DBUFFER
        ROR
        ROR
        SACL  DBUFFER
        SACL  *
        B     GETDAT
COMPLETE:  CMPR  0
           BCND  GETDAT,NTC
           MAR   *,AR5
           LACL  RDATA
           OR    DBUFFER
           SACL  DBUFFER
           SACL  *
           LACC  *,2
           SAMM  DXR
           MAR   *+
           SACL  FIXDAT
           LACC  #0000h
        SACL  DBUFFER
        SACL  TAR0
        LACC  #2408h
        SAMM  ARCR
        NOP
        NOP
        CMPR  0
        BCND  GETDAT,NTC
        LAR   AR6,#2500h
```

```
        LAR   AR5,#2400h
        NOP
        NOP
GETDAT: LACC  #1                ; if receive until 32 points , set RFLAG = 1
        SACL  RFLAG
        B     BACK

CHECK:CMPR 1
        BCND  BACK,TC
        LACC  #2930h
        SAMM ARCR
        NOP
        NOP
        CMPR  0
        BCND  BACK,NTC

SETINT: LAR   AR4,#2900h
        LAR   AR1,#2800h
        LACC  #0
        SACL  TAR1
        SACL  DAT00
        SACL  DAT01
        SACL  DAT10
        SACL  DAT11
        SACL  DAT20
        SACL  DAT21
        SACL  DAT30
        SACL  DAT31
        SACL  DAT40
        SACL  DAT41
```

```
        SACL  DAT50
        SACL  DAT51
        SACL  DAT60
        SACL  DAT61
        SACL  DAT70
        SACL  DAT71
        B     BACK

GETRID: MAR *+
        B     BACK

PRECAL:     LAR   AR4,#2908h
            LAR   AR1,#2808h
            LACC  #0
            SACL  PWbit

BACK: MAR   *,AR1
      RETE

*************************************************************************
*                          AIC  INITIALIZATION                          *
*************************************************************************
AICINIT:    MAR   *,AR0
            LACC  #0
            SAMM  DXR           ; Reset AIC by enable grobal memory and
            LACC  #080H         ; load data in grobal memory to ACC
            SAMM  GREG          ; times. This will make BR pin low about
            LAR   AR0,#0FFFFh   ; 0.5ms . Since AIC reset pin conect with
```

```
        RPT    #10000          ; BR , hence AIC will be in reset state as long
        LACC   *,0,AR0         ; as grobal memory communication has take
                               ; place.
        LACC   #0              ; When reset finish ,disables grobal memmory.
        SAMM GREG
        SETC   SXM
        LACC   #TA,9           ; Initialized TA and RA register
        ADD    #RA,2
        CALL   AIC_2ND
        LACC   #TB,9           ; Initialized TB and RB register
        ADD    #RB,2
        ADD    #02h
        CALL   AIC_2ND
        LACC   #AIC_CTR,2      ; Initialized control register
        ADD    #03h
        CALL   AIC_2ND
        LDP    #84
        SPLK   #0FFFFh,INIT_F
        RET

AIC_2ND: LDP   #0
        SACH   DXR
        CLRC   INTM
        IDLE
        ADD    #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH   DXR
        IDLE
        SACL   DXR
        IDLE
        LACL   #0
```

```
SACL   DXR          ; make sure the word got sent
IDLE
SETC   INTM
RET
.end
```

โปรแกรมสังเคราะห์ซายน์

```
* FIND SIN4X TRANSMIT(INPUT FROM MEMORY)
*************************************************************************
*                                CONSTANT                               *
*************************************************************************
        TA          .set  5         ; LP SCF frequency = 12.5 KHz
        RA          .set  9         ; BP SCF frequency = 6.9 KHz
        TB          .set  53        ; D/A Conversion Frequency =  18.87 KHz
        RB          .set  29        ; A/D Conversion Frequency =  19.2 KHz
        AIC_CTR .set  9h            ; d7,d6 = 00  >>> input gain = 1
                                    ; d5 = 1   >>> synchronous transmit recieve sections
                                    ; d4 = 0   >>> disables the AUX IN+ and AUX IN- pin
                                    ; d3 = 0   >>> disables loopback function
                                    ; d2 = 1   >>> inserts the bandpass filter
*************************************************************************
        .ds 2A50h
TEST    .word 0
*************************************************************************
*                              SINE TABLE                               *
*************************************************************************
        .ds 2900h
phase0      .word 0,1598,3134,4550,5791,6810,7567,8033
            .word 8190,8033,7567,6810,5791,4550,3134;1598
            .word 0,-1598,-3134,-4550,-5791,-6810,-7567,-8033
            .word -8190,-8033,-7567,-6810,-5791,-4550,-3134,-1598
            .word 0,1598
        .ds 2930h
phase90     .word 8190,8033,7567,6810,5791,4550,3134,1598
```

```
        .word 0,-1598,-3134,-4550,-5791,-6810,-7567,-8033
        .word -8190,-8033,-7567,-6810,-5791,-4550,-3134,-1598
        .word 0,1598,3134,4550,5791,6810,7567,8033
        .word 8190,8033

*************************************************************************
*                        Set up the ISR vector                          *
*************************************************************************
        .ps   080Ch
xint:   B     TRANSMIT          ;0C; Serial port transmit interrupt XINT.
*************************************************************************
*                      TMS32C05X INITIALIZATION                         *
*************************************************************************
        .ps 0a00h
        .entry
        .mmregs
STRT:   SETC  INTM              ; Disable interrupts
        LDP   #0
        OPL   #0834h,PMST
        SPLK  #0h,CWSR          ; Set software wait state to 0
        SPLK  #0h,PDWSR         ; Set Program and Data wait state to 0
        SPLK  #022h,IMR         ; Using XINT syn TX & RX
        SPLK  #020h,TCR         ; set timer to generate 10 MHz clock to AIC
        SPLK  #01h,PRD          ; set period register to 1
        SPLK  #08h,SPC          ; set serial port for non continuous mode
        SPLK  #0C8h,SPC         ; enable serial port
        LDP   #84
        CALL  AICINIT           ; initialize AIC and enable interrupts
```

```
*************************************************************************
*                          MAIN  PROGRAM                                *
*************************************************************************
        LAR    AR3,#2820h
        LAR    AR1,#2400h
        LAR    AR0,#2990h
        LAR    AR7,#34
SQ2:    MAR    *,AR7
        LAMM PMST
        OR     #2h
        SAMM PMST          ; TRM=1 LOAD TREG0 NO AFFECT ON TREG1,TREG2
        CLRC   SXM
SQLOOP: MAR *-,AR0
        SPM 0
        ZAP
        SQRA   *+,AR1
        PAC                ; SEND PRODUCT TO ACC
        RPT    #12
        SFR
        SACL   *+,0,AR7    ; KEEP IN MEM ROCATED BY AR1
        BANZ   SQLOOP
*************************************************************************
*                          DIFFERENTIATE1                               *
*************************************************************************
        BIT    TEST,15
        BCND   DIFF2,TC
        LAR    AR1,#2400h
        LAR    AR2,#2600h
        LAR    AR7,#33
        MAR    *,AR1
```

```
        LACC  *+,0,AR7
        SACB
DFLOOP: MAR *-,AR1
        LACC  *,0,AR2
        SBB
        SACL  *,0
        LT    *
        MPY   #5
        PAC
        SACL  *+,0,AR1
        LACC  *+,0,AR7
        SACB
        BANZ  DFLOOP
*****************************************************************************
*                              SQUARE2                                      *
*****************************************************************************
        LAR   AR0,#2600h
        LAR   AR1,#2400h
        LAR   AR7,#34
        LACC  #1
        SACL  TEST
        B     SQ2
*****************************************************************************
*                           DIFFERENTIATE2                                  *
*****************************************************************************
DIFF2:  LAR   AR1,#2400h
        LAR   AR2,#2600h
        LAR   AR7,#33
        MAR   *,AR1
        LACC  *+,0,AR7
```

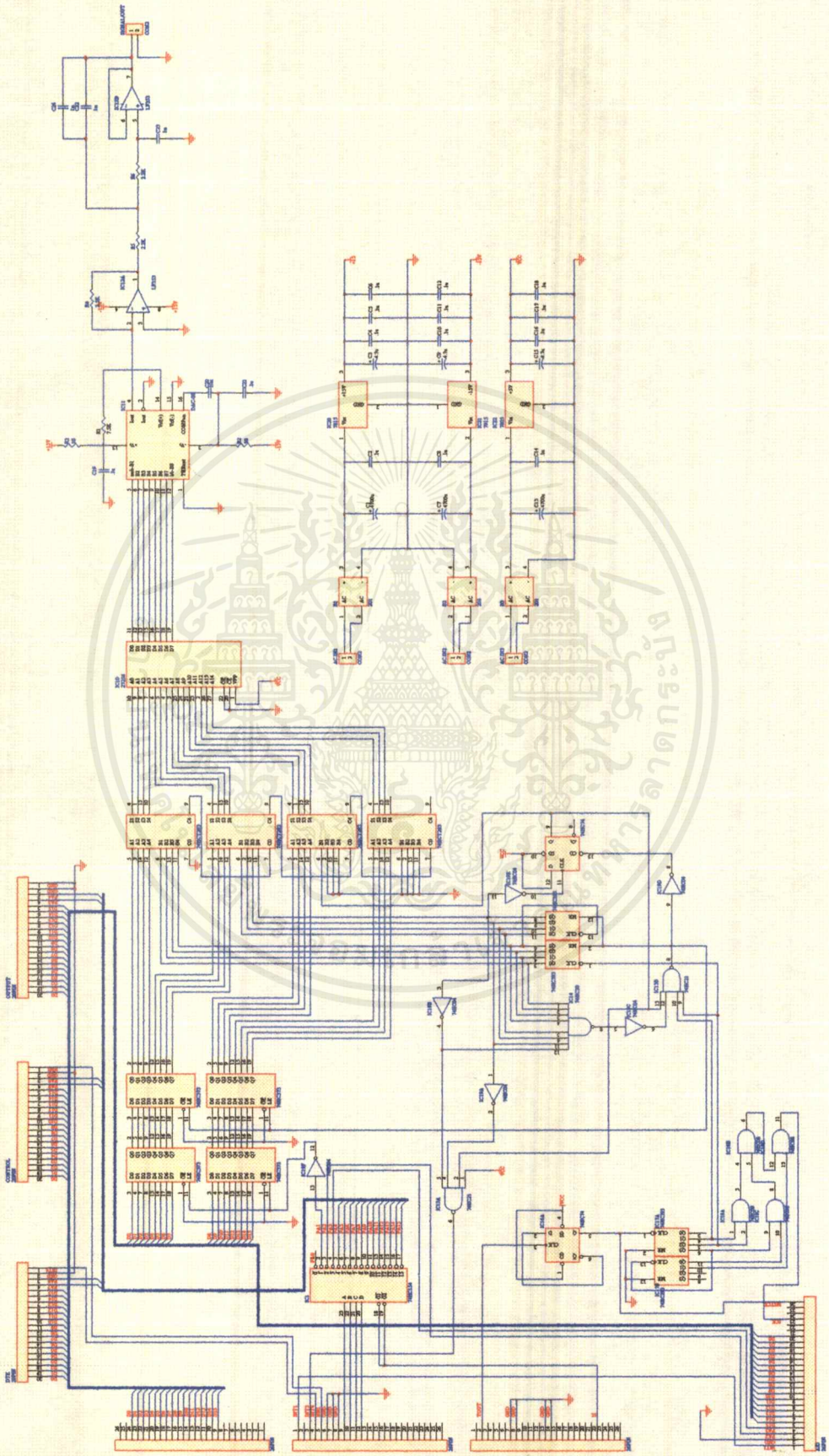
```
        SACB
        CLRC  SXM
DF2LOOP: MAR *-,AR1
        LACC  *,0,AR2
        SBB
        SACL  *,0
        LT    *
        MPY   #5              ; DIVIDE BY 0.196
        PAC
        SFR
        SACL  *+,0,AR1
        LACC  *+,0,AR7
        SACB
        BANZ  DF2LOOP
        LDP   #84
*****************************************************************************
*                Transmit part program (To show output on oscilloscope)     *
*****************************************************************************
        LACC  #2600h
        SAMM  AR1
        ADD   #32
        SAMM  ARCR
        LACC  #0
        SAMM  DXR
        CLRC  INTM
mainloop IDLE
        NOP
        NOP
        B     mainloop
;------- end of main program ----------
```

```
TRANSMIT:
        LDP    #84
        MAR    *,AR1
        LACC   *+,2            ; LOAD ACC WITH DATA TO SEND
        SAMM   DXR             ; SEND DATA
        CMPR   0               ; CHECK AR1 = LAST POINT ?
        BCND   OUT,NTC         ; NO : GOTO OUT_LOOP
        LACC   #2600h
        SAMM   AR1
        ADD    #32
        SAMM   ARCR
OUT:    RETE
*******************************************************************************
*                              AIC Initialization                             *
*******************************************************************************
AICINIT:
      MAR      *,AR0
      ;------------------------
      LACC     #0
      SAMM     DXR             ; Reset AIC by enable grobal memory and
      LACC     #080H           ; load data in grobal memory to ACC 10000
      SAMM     GREG            ; times.This will make BR pin low about
      LAR      AR0,#0FFFFh     ; 0.5ms .Since AIC reset pin conect with
      RPT      #10000          ; BR,hence AIC wil be in reset state as long
      LACC     *,0,AR0         ; as grobal memory communication has take place.
      LACC     #0              ; When reset finish, disables grobal memmory.
      SAMM     GREG
      ;------------------------
      SETC     SXM
      LACC     #TA,9           ; Initialized TA and RA register
```

```
        ADD     #RA,2
        CALL    AIC_2ND
        ;-----------------------
        LACC    #TB,9           ; Initialized TB and RB register
        ADD     #RB,2
        ADD     #02h
        CALL    AIC_2ND
        ;-----------------------
        LACC    #AIC_CTR,2      ; Initialized control register
        ADD     #03h
        CALL    AIC_2ND
        RET

AIC_2ND:
        LDP     #0
        SACH    DXR
        CLRC    INTM
        IDLE
        ADD     #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH    DXR
        IDLE
        SACL    DXR
        IDLE
        LACL    #0
        SACL    DXR             ; make sure the word got sent
        IDLE
        SETC    INTM
        RET
        .end
```

ภาคผนวก ข
รูปวงจร AIC



เอกสารอ้างอิง

[1] ศ.ดร. วัลลภ สุรกำพลธร , "การประมวลผลสัญญาณเชิงเลข , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , พ.ศ. 2533

[2] ณรงค์ เหมกรณ์ , "การสื่อสารดาวเทียม" , โรงพิมพ์คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , พ.ศ. 2533

[3]"TMS320C5x DSP Starter Kit" , Texas Instrument Incorporated , 1994

[4]"TMS320C5x User's Guide" , Texas Instrument Incorporated , 1994

[5]Simoh Haykin , "Digital Communication" , Wiley , 1988

[6]Emmanvel C. Ifeachor & Barie W. Jervis , "Digital Signal Processing" , ADDISON - WESLEY PUBLISHING COMPANY

[7]Fred J. Taylor , "Principle of signal and System"