การตรวจจับสัญญาณซายน์ด้วยดิจิตอลฟิลเตอร์ชนิดปรับค่าตัวแปรอัตโนมัติ โดยใช้ TMS320C50

Detect Sine Wave with Adaptive Filter Using TMS320C50

โดย

นาย กิตติ คำแก้ว

นาย จักรินทร์ เจริญจิตต์

นาย มารุต ค้ำชู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

หัวข้อปริญญานิพนธ์ การตรวจจับสัญญาณซายน์ด้วยดิจิตอลฟิลเตอร์ ชนิดปรับค่าตัวแปรอัตโนมัติ โดยใช้ TMS320C50

Project Report Detect Sine Wave with Adaptive Filter Using TMS320C50

ผู้จัดทำ นาย กิตติ คำแก้ว เลขประจำตัว 39013266

นาย จักรินทร์ เจริญจิตต์ เลขประจำตัว 39013268

นาย มารุต ค้ำชู เลขประจำตัว 39013286

อาจารย์ที่ปรึกษา ผศ. ชวลิต เบญจางคประเสริฐ

ภาควิชา เทคนิคอุตสาหกรรม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

หัวข้อปริญญานิพนธ์	**การตรวจจับสัญญาณซายน์ด้วยดิจิตอลฟิลเตอร์ ชนิดปรับค่าตัวแปรอัตโนมัติ โดยใช้ TMS320C50**
Project Report	Detect Sine Wave with Adaptive Filter Using TMS320C50
ผู้จัดทำ	**นาย กิตติ คำแก้ว เลขประจำตัว 39013266**
	นาย จักรินทร์ เจริญจิตต์ เลขประจำตัว 39013268
	นาย มารุต ค้ำชู เลขประจำตัว 39013286
อาจารย์ที่ปรึกษา	**ผศ. ชวลิต เบญจางคประเสริฐ**
ภาควิชา	**เทคนิคอุตสาหกรรม**
คณะ	**วิศวกรรมศาสตร์**
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง	
ปีการศึกษา	2541

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์นี้ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

การตรวจจับสัญญาณซายน์ด้วยดิจิตอลฟิลเตอร์ ชนิดปรับค่าตัวแปรอัตโนมัติ โดยใช้ TMS320C50

ผู้จัดทำ	นาย กิตติ	คำแก้ว	เลขประจำตัว 39013266
	นาย จักรินทร์	เจริญจิตต์	เลขประจำตัว 39013268
	นาย มารุต	ค้ำชู	เลขประจำตัว 39013286

อาจารย์ที่ปรึกษา ผศ. ชวลิต เบญจางคประเสริฐ

ปีการศึกษา 2541

.....

บทคัดย่อ

ในโครงงานนี้จะนำเสนอการนำเอาตัวกรองสัญญาณดิจิตอลชนิดปรับค่าตัวแปรอัตโนมัติ มาทำการเขียนโปรแกรมบนอุปกรณ์ประมวลผลสัญญาณแบบดิจิตอล เพื่อนำมาประมวลผลสัญญาณแบบเวลาจริง โดยมีวัตถุประสงค์ที่จะนำมาตรวจจับสัญญาณซายน์ที่มีสัญญาณรบกวนปะปนอยู่ด้วยและสามารถเปลี่ยนแปลงความถี่ตามความถี่ของสัญญาณอินพุทเปลี่ยนแปลงไปได้ โดยในขั้นแรกจะทำการทดลองจำลองการทำงานด้วยการเขียนโปรแกรมบนคอมพิวเตอร์ หลังจากนั้นจะเขียนโปรแกรมบนอุปกรณ์ประมวลสัญญาณแบบดิจิตอลเพื่อประมวลผลกับสัญญาณจริงๆ พบว่าโปรแกรมสามารถตรวจจับสัญญาณซายน์ออกจากสัญญาณรบกวนและติดตามความถี่ของการเปลี่ยนแปลงสัญญาณอินพุทได้ในระดับที่ดี

Detect Sine Wave with Adaptive Filter Using TMS320C50

By	Mr. Kitti	Kamkaew	ID 39013266
	Mr. Chakkarin	Chareanchit	ID 39013268
	Mr. Marut	Kamchoo	ID 39013286

Adviser Asst.Prof Chawalit Benjangkaprasert

Academic year 1998

.....

ABSTRACT

This project presents the Adaptive IIR Band Pass Filter. Implement for working in Real Time on Digital Signal Processing device. Aim to detect sine wave from noise and it can vary frequency to follow the input signal. The first, This experiment studies simulation by programming on the computer. Then, we are working and programming on Digital Signal Processing device. It means that the proposed program works as expected

กิตติกรรมประกาศ

การที่โครงงานการ Detect Sine Wave with Adaptive Filter Using TMS320C50 สำเร็จลงด้วยความเรียบร้อยเป็นอย่างดีนั้น ทางคณะผู้จัดทำจึงขอขอบพระคุณอย่างสูงต่อท่านทั้งหลายที่กรุณาให้ความอนุเคราะห์ ความร่วมมือ ให้คำแนะนำและคำปรึกษา และอำนวยความสะดวก โดยเฉพาะ ผศ. ชวลิต เบญจางคประเสริฐ ในการให้คำปรึกษาชี้แนะและแก้ปัญหาต่างๆ ในการจัดทำโครงงานและในการเขียนปริญญานิพนธ์และ ผศ. อรลาภ แสงอรุณ ในการเอื้อเฟื้ออุปกรณ์ต่างๆ ในการทำโครงงาน

คณะผู้จัดทำ

กิตติ คำแก้ว

จักรินทร์ เจริญจิตต์

มารุต ค้ำชู

สารบัญ

สารบัญรูป

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 บทนำ

สัญญาณคือปริมาณทางกายภาพใดๆ ซึ่งเป็น Function ของตัวแปรอิสระหนึ่งหรือมากกว่าซึ่งเปลี่ยนแปลงค่าตามเวลา,ระยะทาง,อุณหภูมิหรือความดัน การแปรค่าของสัญญาณทาง Amplitude ซึ่งเป็น Function ของตัวแปรอิสระ เราเรียกว่ารูปคลื่น (Waveform) สัญญาณสามารถเกิดขึ้นเองได้ตามธรรมชาติหรือได้จากการผลิตหรือจากการ Simulated ซึ่งสัญญาณสามารถเป็นได้ทั้งสัญญาณที่ต่อเนื่อง (Continuous Signal) และสัญญาณแบบเป็นช่วงหรือไม่ต่อเนื่อง (Discrete Signal) ถ้าสัญญาณที่เกิดขึ้นเป็น Function ของตัวแปรอิสระเพียงตัวเดียว เราเรียกว่าสัญญาณหนึ่งมิติ (1-D Signal) แต่ถ้าสัญญาณที่เกิดขึ้นเป็น Function ของตัวแปรอิสระสองตัว เรียกว่าสัญญาณสองมิติ (2-D Signal) สัญญาณหลายมิติ (Multidimension Signal) เกิดจาก Function ของตัวแปรอิสระมากกว่าหนึ่งตัว

การประมวลผลสัญญาณเกี่ยวข้องกับการนำเอาคณิตศาสตร์มาแทนสัญญาณในโดเมน (Domain) เดิมของตัวแปรที่เกิดขึ้นหรือในการแปลงโดเมน (Domain) ไปแล้วและเกี่ยวกับ Algorithmic ที่ใช้ในการถอดข้อมูลออกจากสัญญาณหลังจากผ่านการพา (Carried)

1.2 วัตถุประสงค์ของโครงงาน

1. ทำการศึกษา ทำความเข้าใจในการทำงานในทางทฤษฎีเกี่ยวกับการทำงานของ Digital Signal Processing
2. ศึกษาทำความเข้าใจระบบการทำงาน Digital Filter ทั้งจากการ Simulation และการใช้งานประมวลผลสัญญาณจริงๆ
3. ศึกษาทำความเข้าใจการทำงานพื้นฐานของ Adaptive Filter ทั้งในการทดลองโดยการ Simulation และการนำไปประมวลผลกับสัญญาณจริงๆ เพื่อที่จะสามารถนำไปประยุกต์ในการใช้งานจริงๆ ในด้านต่างในอนาคตๆ ได้
4. ศึกษาการทำงานและการโปรแกรม Chip DSP โดยนำไปเขียนโปรแกรมนำไปใช้งานจริงๆ ซึ่งจะสามารถทำความเข้าใจได้ดีกว่าการศึกษาทฤษฎีเพียงอย่างเดียวและสามารถที่จะมองเห็นประโยชน์ที่จะนำมันไปใช้งานในด้านอื่น
5. ทำการศึกษาเปรียบเทียบผลจากการ Simulation และจากการทำการทดลองจากการประมวลผลจากสัญญาณจริงๆ ว่าจะมีความเหมือนหรือแตกต่างกันมากน้อยหรือไม่อย่างไร รวมทั้งสามารถหาเงื่อนไขที่ทำให้แตกต่างกันได้
6. สามารถนำความรู้ที่ได้จากการทำโครงงานนี้เป็นพื้นฐานในการทำงานหรือประยุกต์ใช้งาน ที่เกี่ยวกับการทำงานของการประมวลผลแบบดิจิตอลต่อไปได้

1.3 ขอบเขตของโครงงาน

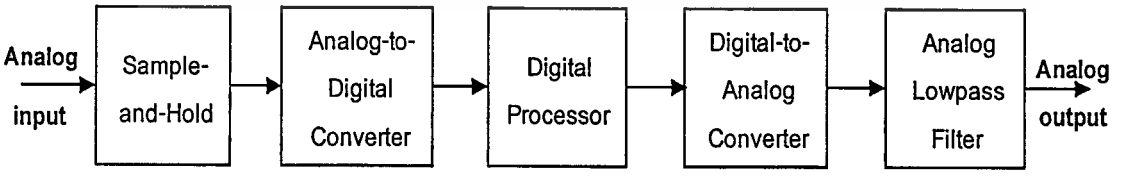
1. โครงงานนี้ยังเป็นเรื่องที่ยังทำเป็นงานวิจัยและพัฒนาอยู่โดยยังไม่สามารถที่จะนำไปใช้งานประยุกต์จริงๆ ได้ เพราะยังมีการพัฒนาหา Algorithm ที่มีการติดตามสัญญาณที่รวดเร็วกว่านี้อยู่แต่ยังมีการนำไปเขียนโปรแกรมทดลองใช้ประมวลผลกับสัญญาณจริงๆ น้อยมาก จึงทำให้เกิดแนวคิดที่จะทำโครงงานนี้ขึ้นมา
2. โครงงานนี้มิได้มีวัตถุประสงค์ในการทำการออกแบบหรือสร้าง Hardware ขึ้นมาใหม่ แต่จะมีการศึกษาเกี่ยวกับ Hardware ที่เป็นชุดพัฒนาโปรแกรมของบริษัท ผลิต Chip แต่ในการเขียนโปรแกรมภาษา Assembly นั้นจำเป็นที่จะต้องมีความรู้เกี่ยวกับ Hardware ของมันพอสมควร ซึ่งเราจึงจะทำการศึกษา Hardware เท่าที่จะสามารถนำไปใช้ในการที่จะทำความเข้าใจถึงโครงสร้างที่สำคัญที่ทำให้มีความเข้าใจพอที่จะเขียนโปรแกรมได้ซึ่งไม่ได้เจาะลึกรายละเอียดทั้งหมดเพราะจะเยอะมาก โดยในที่นี้ถือว่าชุดพัฒนาโปรแกรม TMS320C5x Starter Kit เป็นเครื่องมืออย่างหนึ่งในโครงงานนี้เท่านั้น

1.4 การประมวลผลสัญญาณดิจิตอล (Digital Signal Processing)

โดยธรรมชาติแล้วสัญญาณที่เกิดขึ้นส่วนมากจะเป็น Function ที่ต่อเนื่องของตัวแปรอิสระเทียบกับเวลาซึ่งโดยทั่วไปแล้วเราจะเรียกว่าสัญญาณอนาล็อก (Analog Signals) การประมวลผลทางดิจิตอลของสัญญาณอนาล็อก ต้องประกอบด้วยพื้นฐาน 3 ขั้นตอนคือ

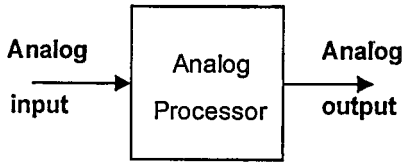
1. เปลี่ยนสัญญาณอนาล็อกให้อยู่ในรูปของสัญญาณดิจิตอล
2. ทำการประมวลผลสัญญาณดิจิตอล
3. เปลี่ยนสัญญาณดิจิตอลที่ถูกประมวลผลแล้วกลับเป็นสัญญาณอนาล็อกอย่างเดิม

ส่วนประกอบต่างๆทั้งหมดแสดงได้ในรูปที่ 1.1



รูปที่ 1.1 Block Diagram ของการประมวลผลสัญญาณอนาล็อกในแบบดิจิตอล

ในการเปรียบเทียบการประมวลผลสัญญาณอนาล็อกโดยตรงซึ่งใช้หลักการง่ายๆเพราะว่ามันยุ่งเกี่ยวกับการประมวลผลเพียงส่วนเดียวดังแสดงให้เห็นในรูปที่ 1.2 ดังนั้นจึงมักมีคำถามเกี่ยวกับประโยชน์ที่ดีกว่าของการประมวลผลสัญญาณดิจิตอลเมื่อเทียบกับการประมวลผลสัญญาณแบบอนาล็อก



รูปที่ 1.2 Block Diagram ของการประมวลผลสัญญาณอนาล็อกในแบบอนาล็อก

ข้อแตกต่างจากวงจรอนาล็อกอีกอย่างหนึ่งคือ การทำงานของวงจรดิจิตอลไม่ขึ้นอยู่กับค่าของสัญญาณดิจิตอล ดังนั้นผลลัพธ์ที่ได้จากวงจรดิจิตอลจะได้รับผลกระทบน้อยจากค่าต่างๆ ที่เป็นส่วนประกอบของวงจรเช่น ค่าของอุณหภูมิและพารามิเตอร์ภายนอกอื่นๆ

ในการประมวลผลสัญญาณดิจิตอล สัญญาณและสัมประสิทธิ์ของมันจะถูกทำให้อยู่ในรูปของเลขฐานสองจำนวนแปดบิต (binary words) ดังนั้นเราสามารถเพิ่ม wordlength ภายใต้ขอบเขตที่จำกัดได้โดยง่าย ยิ่งกว่านั้น dynamic ranges และสัมประสิทธิ์ของสัญญาณสามารถทำให้เพิ่มขึ้นโดยใช้การคำนวณทาง Floating-Point (การคำนวณเลขฐานสองหรือฐานสิบหกในรูปของเลขทศนิยม) ได้ถ้าจำเป็น

ในระหว่างการประมวลผลเราสามารถใช้คำสั่งทางดิจิตอลในการปรับแต่งคุณลักษณะของตัวประมวลผล เช่นต้องการปรับให้เป็นแบบ Adaptive Filter ซึ่งการปรับสามารถทำได้โดยการเปลี่ยนค่าสัมประสิทธิ์ต่างๆ ของ Algorithm และการประยุกต์ใช้งานอื่นๆในการปรับเปลี่ยนค่าสัมประสิทธิ์ของระบบสามารถทำได้โดยการโปรแกรม เช่นการเลือกความความถี่ต่างๆ สามารถทำได้โดยการปรับความถี่ cutoff คุณสมบัติการตอบสนองความถี่ในช่วงความถี่ที่กำหนดมีความน่าเชื่อถือโดยการใช้คำสั่งที่อยู่ในรูปแบบของดิจิตอล

คุณลักษณะดังกล่าวสามารถใช้คำสั่งทางดิจิตอลทำได้แต่คำสั่งทางอนาล็อกไม่สามารถทำได้ เช่นการประมวลผลแบบ Linear Phase และ Multirate (การเพิ่มอันดับ) วงจรดิจิตอลสามารถต่อแบบ Cascaded กันได้โดยปราศจากปัญหาใดๆไม่เหมือนกับวงจรอนาล็อก (เช่นการ matching กันของ Input & Output Impedance) และสัญญาณดิจิตอลสามารถเก็บไว้ได้นานโดยไม่มีการสูญหายของข้อมูลบนตัวกลางต่างๆ เช่น magnetic tapes , disks หรือ optical disks ดังนั้นสัญญาณที่ถูกเก็บอยู่สามารถนำมาประมวลผลภายหลังได้ เช่นในเครื่องเล่น Compact Disk , เครื่องเล่น VCD หรืออาจถูกใช้บนเครื่องคอมพิวเตอร์ทั่วๆ ไปได้

1.4.1 ข้อเสียของการประมวลผลแบบดิจิตอล

1. ถ้าสัญญาณ Input เป็นแบบอนาล็อกต้องมีการเพิ่มระบบ รวมเข้าไปในการประมวลผลดิจิตอลด้วยคือ ต้องเพิ่มอุปกรณ์ที่ใช้ในการเตรียมและส่งผ่านการประมวลผลเช่น A/D และ D/A Converters

2. แถบความถี่ที่สามารถจัดการได้ของการประมวลผลสัญญาณแบบดิจิตอล ซึ่งตอนแรกถูกกำหนดโดยวงจรแซมเปิ้ลและโฮล (Sample-and-Hold Circuit) และวงจร A/D Converter ซึ่งผลลัพธ์ที่ได้ถูกจำกัดอยู่ที่เทคโนโลยีการออกแบบ ซึ่งโดยทั่วไปสัญญาณอนาล็อกต้องการความถี่แซมเปิ้ลอย่างน้อยสองเท่าของความถี่ใช้งาน ถ้าไม่เป็นไปตามเงื่อนไขนี้ส่วนประกอบของความถี่ที่สูงกว่าครึ่งหนึ่งของความถี่แซมเปิ้ลจะหักล้างกับความถี่ที่เราใช้งานซึ่งต่ำกว่าเรียกว่าเกิดการผิดเพี้ยน (Distortion) ของสัญญาณ Input ที่เป็นสัญญาณอนาล็อก วิธีแก้ปัญหามีหลายวิธีแต่ที่นิยมคือการใช้วงจร Lowpass Filter ที่มีจุด Cutoff ต่ำกว่าครึ่งหนึ่งของความถี่ Sampling กรองสัญญาณ Input ที่เข้ามาก่อน

3. มีสาเหตุมาจากการที่ส่วนต่างๆ ของวงจรดิจิตอลถูกสร้างขึ้นมาจากอุปกรณ์ประเภท Active ซึ่งต้องใช้กำลังงานไฟฟ้าในการทำงาน ยกตัวอย่างเช่น Chip Digital Signal Processor DSP32C ของบริษัท AT&T ซึ่งมี Transistor บรรจุอยู่ภายในมากกว่า 405,000 ตัวและเกิดการสูญเสียโดยรวมประมาณ 1 Watt ซึ่งมีการประมวลผลแบบอนาล็อกหลายแบบที่ Algorithms สามารถใช้วงจรแบบ Passive ที่มีอุปกรณ์จำพวก Inductors , Capacitors และ Resisters ซึ่งไม่ต้องใช้กำลังงานเลย ยิ่งกว่านั้นอุปกรณ์แบบ Active ยังมีความน่าเชื่อถือน้อยกว่าอุปกรณ์แบบ Passive

อย่างไรก็ตาม ประโยชน์ของวงจรแบบดิจิตอลก็มีมากกว่าข้อเสียของมันและ Hardware ของการประมวลผลสัญญาณดิจิตอลก็มีราคาถูกลง จึงทำให้มีการพัฒนา Applications ของการประมวลผลสัญญาณดิจิตอลเพิ่มขึ้นอย่างรวดเร็ว

ในโครงงานนี้เราจะใช้ข้อดีของการประมวลผลแบบดิจิตอลคือการปรับตัวแปรในการคำนวณค่า Output ของ Digital filter โดยการปรับความถี่ตามความถี่หลักของสัญญาณ Input ซึ่งจะกล่าวรายละเอียดในหัวข้อที่ 2.4

1.5 การแยกระบบต่างๆ ในระบบ Discrete-Time

ระบบ LINEAR

คือระบบที่สามารถวิเคราะห์ได้ด้วยการใช้ทฤษฎี Super Position ผลตอบสนองทาง Output สามารถหาได้จากการรวมของผลตอบสนองของสัญญาณใดๆ ทางด้าน Input เมื่อ Input ประกอบจากสัญญาณใดๆ รวมกันดังแสดงได้ดังนี้

$$H[a_1x_1(n)+a_2x_2(n)] = a_1H[x_1(n)]+a_2H[x_2] = a_1y_1(n)+a_2y_2(n) \quad (1.1)$$

ระบบที่ไม่เป็นไปตามสมการนี้เราเรียกว่าระบบ NON-LINEAR

ระบบคอเซล คือระบบที่ไม่มีการนำสัญญาณ Input ในอนาคตมาทำการคำนวณด้วยซึ่งเป็นระบบที่สามารถนำไปใช้งานแบบ Real time ได้ส่วนระบบนอนคอเซล ไม่สามารถนำมาใช้งานจริงแบบนั้นได้

เราสามารถแยกระบบ LTI (Linear Time Invariant) ได้ตามลักษณะการตอบสนองอิมพัลส์ (Impulse) ของมันได้ดังนี้

FIR (Finite Impulse Response) สามารถแปลได้ว่า ระบบที่มีการตอบสนองต่ออิมพัลส์ในช่วงที่จำกัดในระบบ สำหรับระบบคอเซลสามารถพิจารณาได้ว่า

H(n) = 0 เมื่อ n < 0 n >= m

สูตรการ Convolution จะอยู่ในรูป

$$y(n) = \sum_{k=1}^{m-1} h(k).x(n-k) \qquad (1.2)$$

จะเห็นว่า Output ที่เวลาใดๆ คือการรวมเชิงเส้นของ Input X (n),X (n-1).....,X (n–m+1) ที่ถูกถ่วงน้ำหนักโดยระบบถ่วงน้ำหนักล่าสุด (Most Recent) ของสัญญาณจำนวน m ซึ่งระบบจะแสดงผลเหมือนหน้าต่างที่มองเห็น Input ที่ใหม่ที่สุดจำนวน m ตัวอย่างในการทำให้เกิด Output

IIR (Infinite Impulse Response) ระบบที่มีการตอบสนองต่ออิมพัลส์ที่มีช่วงเวลาไม่จำกัดในระบบ ดังนั้น สำหรับระบบคอเซล สามารถพิจารณาได้ว่า

H(n) = 0 ที่ n < 0

สูตรการ Convolution จะอยู่ในรูป

$$y(n) = \sum_{k=0}^{\infty} h(k).x(n-k) \qquad (1.3)$$

จะเห็นว่า output ของระบบจะเป็นการรวมเชิงเส้นของ ตัวอย่างinput X (n),X (n-1),X (n-2),... ที่ถูกถ่วงน้ำหนัก เพราะผลบวกที่เกี่ยวข้องกับตัวอย่างในปัจจุบันและอดีตทั้งหมดของ Input ดังนั้นระบบมีความจำเป็นอนันต์

1.5.1 การวิเคราะห์ระบบ Discrete-Time เมื่อแทนโดยสมการผลต่าง

ที่ผ่านมาเราสามารถแยก LTI ด้วยผลตอบสนองอิมพัลส์ของมัน ในทางกลับกันเราสามารถหา output Y(n) จากผลตอบสนองอิมพัลส์ของมันโดยผ่านการทำ Convolution

$$y(n) = \sum_{k=-\infty}^{\infty} h(k).x(n-k) \qquad (1.4)$$

นอกจากนั้นจากสมการข้างต้นได้แนะนำถึงการทำให้เป็นระบบจริงในกรณีของระบบ FIR การทำให้เป็นจริงเกี่ยวกับการบวก การคูณ และตำแหน่งของหน่วยความจำที่มีจำนวนจำกัด ผลก็คือเราจะได้ระบบ FIR โดยตรงจากสมการข้างต้น

อย่างไรก็ตามการทำให้ระบบ FIR เป็นจริงโดยสมการข้างต้นไม่สามารถทำได้ในทางปฏิบัติ เพราะว่ามันต้องการหน่วยความจำจำนวนมากในการคูณและการหาร แต่เรามีวิธีอื่นในการคำนวณซึ่งแสดงดังหัวข้อต่อไปนี้

1.5.2 ระบบ Discrete-Time แบบ Recursive และ Non-recursive

ดังที่ได้แสดงสูตรการบวก Convolution ที่ผ่านมาจะเห็นได้ว่า Output ของระบบ LTI จะถูกแสดงอยู่ในรูปของสัญญาณ Input เพียงอย่างเดียว แต่อย่างไรก็ตามจะมีหลายระบบที่จำเป็นในการต้องการแสดง Output ของระบบ ซึ่งไม่เพียงแต่ในเทอมของ Input ในปัจจุบันและในอดีตเท่านั้นแต่ยังรวมถึงเทอมที่หาได้แล้วของค่า Output ในอดีตโดยลองพิจารณาตัวอย่างต่อไปนี้

สมมุติว่าเราต้องการคำนวณค่าเฉลี่ยสะสม (Accumulative Average) ของสัญญาณ X (n) ในช่วง 0 <= k <= n ซึ่งนิยามว่า

$$y(n)=\frac{1}{n+1}\sum_{k=0}^{n}x(k).....n=0,1,... \qquad (1.5)$$

เพื่อให้สามารถคำนวณสมการข้างบนได้ถูกต้องต้องมีการเก็บค่า Input ทุกค่าคือ $x(k)$ สำหรับ 0 <= k <= n แต่เพราะว่า n เพิ่มมากขึ้นเรื่อยๆ ดังนั้นเราต้องการจำนวนหน่วยความจำที่เพิ่มขึ้นอย่างเป็นเชิงเส้นกับเวลา

อย่างไรก็ตามเราสามารถคำนวณให้มีประสิทธิภาพมากขึ้นโดยใช้ค่า Output ก่อนหน้า $y(n-1)$ โดยจัดเทอมสมการข้างบนเสียใหม่

$$\begin{aligned}(n+1)y(n)&=\sum_{k=0}^{n-1}x(k)+x(n)\\&=ny(n-1)+x(n)\end{aligned} \qquad (1.6)$$

ดังนั้น

$$y(n)=\frac{n}{n+1}y(n-1)+\frac{1}{n+1}x(n) \qquad (1.7)$$

ขณะนี้เราสามารถคำนวณ $y(n)$ อย่าง Recursive โดยคูณ Output ก่อนหน้าคือ $y(n-1)$ ด้วยค่า $\frac{n}{n+1}$ และคูณค่า Input ในปัจจุบัน $x(n)$ ด้วย $\frac{1}{n+1}$ จากนั้นทำการบวกผลคูณทั้งสองเข้าด้วยกัน ขณะนี้การคำนวณ $y(n)$ ตามสมการหลังสุดต้องการการคูณสองครั้งและการบวกอีกหนึ่งครั้งและต้องการหน่วยความจำเพียงหนึ่งตำแหน่งโดยทั่วระบบ ซึ่ง Output ของมันที่เวลา n ขึ้นอยู่กับค่า Output ในอดีตใดๆ $y(n-1), y(n-2)$ .....เรียกว่าระบบ Recursive

$$y(n_0) = \frac{n_0}{n_0+1} y(n_0-1) + \frac{1}{n_0+1} x(n_0) \qquad (1.8)$$

ในสมการที่ผ่านมา หากเราต้องการคำนวณระบบต่อ input $x(n)$ ที่ n เท่ากับ 0 คือ $y(n_0)$ เราต้องการค่า $y(n_0-1)$ และตัวอย่างของ input $x(n)$ สำหรับ $n \geq n_0$ เทอม $y(n_0-1)$ เรียกว่า เงื่อนไขเริ่มต้น (Initial Condition)

ส่วนระบบ Non-Recursive $y(n)$ จะขึ้นอยู่กับ input ในอดีตและปัจจุบันเท่านั้นแสดงโดยสมการ (1.9)

$$y(n) = F[x(n), x(n-1), \ldots x(n-m)] \qquad (1.9)$$

ซึ่งจะเหมือนกับระบบคอเซล แบบ FIR ที่ใช้สูตรผลบวก Convolution

1.6 FILTER

คือระบบที่ใช้ในตัดหรือคัดเลือกความถี่ในย่านที่ต้องการ ซึ่งมีอยู่หลายชนิดตามลักษณะการตัดหรือการเลือกความถี่ดังนี้

Low pass Filter

High pass Filter

Band pass Filter

Notch Filter

ซึ่งจะไม่กล่าวถึงรายละเอียดในทุกแบบเพราะจะเกินขอบเขตของโครงงาน (สามารถศึกษาจากหนังสือทางด้าน การประมวลผลสัญญาณทั่วไป) ในที่นี้จะพิจารณาในส่วนของ IIR Band pass Filter เพื่อนำมาใช้งานการตรวจจับสัญญาณ Band แคบๆ ซึ่งจะกล่าวถึงในบทต่อไป

บทที่ 2
ทฤษฎี Filter และ Simulation

ในการศึกษาหรือทำงานวิจัยทางวิศวกรรมในบางลักษณะไม่จำเป็นต้องทำการทดลองด้วยการกระทำกับสัญญาณจริงๆเพราะถ้าเราสามารถทราบแบบจำลองทางคณิตศาสตร์ของมัน เราก็จะสามารถศึกษาผลตอบสนองของระบบได้จากการเปลี่ยนค่าต่างในแบบจำลองทางคณิตศาสตร์ของมัน ซึ่งสามารถทำได้สะดวกกว่าการทำงานกับสัญญาณหรือทำการสร้างระบบขึ้นมาจริงๆ โดยเฉพาะสำหรับระบบขนาดใหญ่ที่มีความซับซ้อนมากการสร้างโดยมีข้อมูลไม่พอหรือไม่ครบทุกด้านอาจทำให้เกิดความเสียหายอย่างมากได้ ในอดีตการคำนวณที่มีความซับซ้อนมากจะทำได้ยากมากเพราะต้องใช้อุปกรณ์การคำนวณที่มีประสิทธิภาพสูงมาก แต่หลังจากมีการพัฒนาไมโครคอมพิวเตอร์อย่างรวดเร็วมากทำให้มีการใช้งานคอมพิวเตอร์มาใช้ในการคำนวณทางด้านนี้มากจนเป็นที่นิยมและยอมรับกันโดยทั่วไป สามารถดูผลการตอบสนองได้ละเอียดครบถ้วน (แต่ในการใช้งานจริงๆ อาจมีตัวแปรที่ไม่ทราบค่าอีกมาก) จากการคำนวณในทุกๆด้าน รวมทั้งการพัฒนาทางด้าน Software ก็มีการพัฒนามากทำให้เกิดความง่ายในการทำการ Simulation จากสมการทางคณิตศาสตร์

ในการทำงานทางด้าน Digital Signal Processing ซึ่งจะอาศัยการคำนวณทางด้านดิจิตอลดังที่กล่าวมาแล้ว เพราะฉะนั้นในเงื่อนไขของการ Simulation จะมีลักษณะการคำนวณเหมือนหรือใกล้เคียงกับการใช้งานจริงๆมากจนการ ทำงานทางด้านนี้จะสามารถมั่นใจได้ว่าเมื่อเราสามารถ Simulation ผลการทดลองออกมาได้เมื่อเรานำไปทำการประมวลผลกับสัญญาณจริงก็จะได้ผลเหมือนกันกับการ Simulation ดังนั้นในโครงงานนี้จะทำการศึกษาการ ตอบสนองทางด้านต่างๆของ Adaptive IIR Filter จาก Transfer Function ของมันทำโดยการใช้โปรแกรมสำหรับการเขียนโปรแกรมทางด้านคณิตศาสตร์และวิศวกรรมโดยเฉพาะคือโปรแกรม Matlab (ซึ่งมีคำสั่งที่สะดวกสำหรับการทำงานด้านนี้มากกว่าโปรแกรมที่ใช้ในการเขียนโปรแกรมบนคอมพิวเตอร์ ทั่วไปเช่น ภาษาBasic Pascal) หลังจากนั้นจึงค่อยนำสมการต่างๆที่ทำการ Simulation มาทำการเขียนโปรแกรมกับ Chip DSP เพื่อประมวลผลกับสัญญาณจริงๆ

2.1 อนาล็อก Filter

อนาล็อก Filter ซึ่งมีการใช้งานกันมากมายทางด้านต่างๆ ซึ่งมีแบ่งแยกตามลักษณะการใช้งานในแบบต่างๆ เช่น ลักษณะการตัดสัญญาณ , ย่านในการกรองความถี่ หรือ ลักษณะของวงจรซึ่งในที่นี้จะเลือกเอา Narrow Band pass Filter order 2 มาทำการ Simulate เพื่อดูผลการตอบสนองลักษณะต่างๆของมันทั้งนี้เพราะมันมีลักษณะของตัวแปรและลักษณะการพิจารณาที่เหมือนกัน (เนื่องจากเราจะคุ้นเคยกับ อนาล็อก Filter กันอยู่มากกว่า)

จาก Transfer Function ของ Analog Band pass Filter order 2 (Narrow Band) ตามสมการด้านล่าง

$$H_{BP}(s) = \frac{k\left(\frac{w_0}{Q}\right)s}{s^2 + \left(\frac{w_0}{Q}\right) + w_0^2} \qquad (2.1)$$

โดยตัวแปรต่างๆ ที่สามารถพิจารณาการตอบสนองลักษณะต่างๆ มีดังนี้

k คือค่า Gain ที่ความถี่กลาง

w_0 คือค่าความถี่กลาง

Q คือค่าของความชันในการ Cut ของ Filter ซึ่งสามารถกำหนดโดย

$$Q = \frac{w_0}{(w_h - w_l)} \qquad (2.2)$$

w_h คือค่าความถี่ Gain เป็น –3 dB ทางด้านสูง

w_l คือค่าความถี่ Gain เป็น –3 dB ทางด้านต่ำ

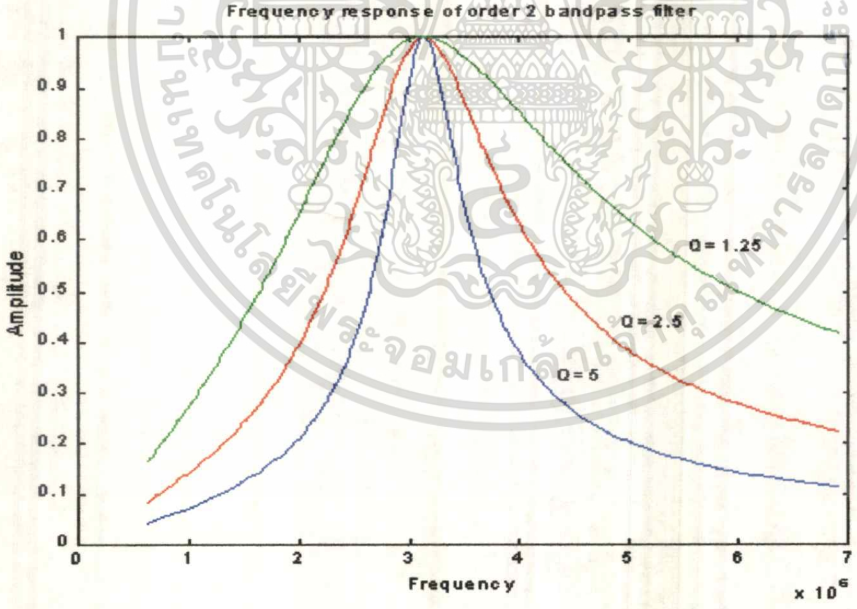
จาก Transfer Function ที่ผ่านมาสามารถทดลองเขียนโปรแกรม Simulation เพื่อทำการศึกษาว่าระบบมีการตอบสนองอย่างไร เมื่อมีการเปลี่ยนค่าตัวแปรต่างๆ ในระบบ โดยโปรแกรมสามารถแสดงได้ดังนี้คือ

```
clear;
w=2*pi*(linspace(10e4,110e4,20000));   %Frequency Range
s=(j*w);
w0=50e4*(2*pi);                        %Frequency Cut
wl=45e4*(2*pi);                        %Low  Frequency Cut
wh=55e4*(2*pi);                        %High Frequency Cut
k=1;                                   %Gain
q=w0/(wh-wl);                          %Q Factor
d=s.*(w0/q);                           %Calculate transfer function
n=s.^2+s.*(w0/q)+w0^2;
h=d./n;
```

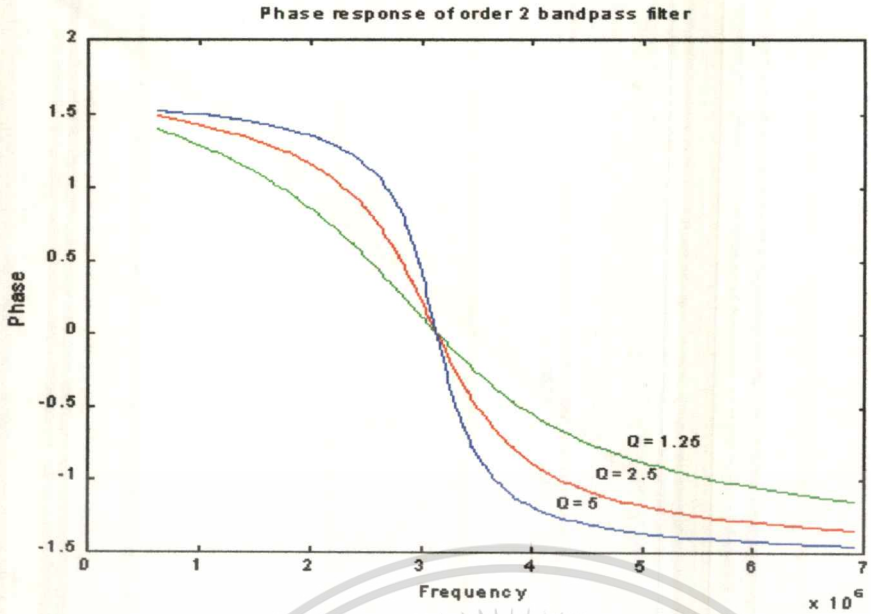
```
H=abs(h);
phase=angle(h);
subplot(211);
plot(w,H,'b');
subplot(212);
plot(w,phase,'b');
```

โปรแกรมที่แสดงนี้เป็นส่วนที่เป็นส่วนหลักในการคำนวณ ส่วนที่เป็นรายละเอียดในการ plot ค่าต่างๆ ไม่ได้แสดงไว้ทั้งหมด

รูปที่ได้จากการ Simulate แสดงได้ดังนี้



รูปที่ 2.1 การแสดง Frequency Response ที่ค่า Q ต่างๆ กัน



รูปที่ 2.2 การแสดง Phase Response ที่ค่า Q ต่างๆ กัน

จากรูปที่แสดงในส่วนของ Frequency Response ได้แสดงให้เห็นว่าค่า Q มีผลต่อการตอบสนองของความถี่ข้างๆ ของความถี่กลางจึงเหมือนกับว่าเป็นการลดค่า Band Width ของระบบ ซึ่งสามารถนำไปใช้งานในการตรวจจับสัญญาณในย่านแคบๆ ได้ดี ส่วนของผลตอบสนองทาง Phase แสดงความสัมพันธ์กับการตอบสนองทางความถี่ได้ว่ายิ่งมีการตัดความชันมากเท่าไหร่ก็จะมีการเปลี่ยนแปลงทาง Phase ที่ความถี่ใกล้กับความถี่กลางมากขึ้นเท่านั้น แต่ที่ความถี่กลางจะไม่มีการเปลี่ยนแปลงทาง Phase เลยที่ทุกๆ เงื่อนไข

2.2 Digital IIR Filter

Digital Filter นั้นมีอยู่หลายประเภท สามารถแบ่งออกได้หลายชนิด เช่นแบ่งตามลักษณะการตอบสนองอิมพัลส์(IIR,FIR) ตามสมการผลต่างซึ่งก็คือสมการใน Time Domain(Recursive,Non-Recursive) รวมทั้งการออกแบบสามารถทำได้หลายวิธี เช่นการออกแบบใน Z-Domain โดยตรงซึ่งมีหลายแบบตามลักษณะการประมาณค่าสัมประสิทธิ์แบบต่างๆ ซึ่งมีการตอบสนองแตกต่างกันตามการใช้งานลักษณะต่างๆ (เหมือนกับอนาล็อกที่มีการประมาณค่าสัมประสิทธิ์แบบต่างๆ หลายแบบเช่น Butterworth , Chebyshev) หรือออกแบบในอนาล็อก (ใน S-Domain) ก่อนแล้วทำการแปลงไปเป็นแบบดิจิตอล (Z-Domain) ซึ่งก็มีการแปลงหลายแบบอีกตามความต้องการการตอบสนองความถี่ลักษณะต่างๆ ซึ่งรายละเอียดเหล่านี้จะสามารถศึกษาได้จากหนังสือทางด้าน การประมวลผลสัญญาณดิจิตอลทั่วๆ ไป แต่ในโครงงานนี้จะนำเอา Order'2 IIR filter มาใช้งานเพราะเราต้องนำเอา Transfer Function

ของมันมาทำการหาสัมประสิทธิ์ของส่วนการปรับค่าตัวแปรต่างๆ โดยอัตโนมัติซึ่งอยู่ในส่วนของ Adaptive Filter จะกล่าวถึงรายละเอียดในหัวข้อ 2.4

จาก Transfer Function ของ IIR Digital Filter order 2 สามารถแสดงได้ดังนี้คือ

$$H_{BP}(z) = \left[\frac{1-\alpha_0}{2}\right].\left[\frac{1-z^{-2}}{1-\alpha_1(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}}\right] \qquad (2.3)$$

โดยมีตัวแปรต่างๆ ที่สามารถปรับค่าได้ดังนี้คือ

α_1 เป็นสัมประสิทธิ์ในการแสดงค่าความถี่กลาง (ใน Z-Domain) โดยกำหนดได้ดังนี้

$\alpha_1 = \cos(\omega_0)$ โดยมีค่าอยู่ระหว่าง -1 ถึง 1

α_0 เป็นสัมประสิทธิ์ในการกำหนดค่า Q-Factor มีค่าคือ $0 < \alpha_0 < 1$

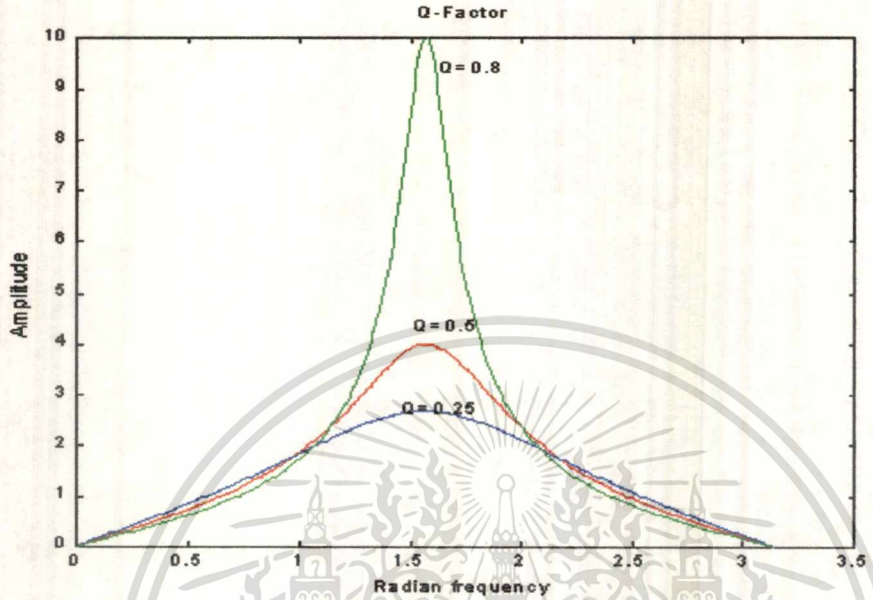
2.2.1 การตอบสนองใน Frequency Domain ของ IIR Filter

ในการวิเคราะห์การตอบสนองเชิงความถี่ของ Digital Filter เราจะทำใน Z-Domain (เหมือนของอนาล็อก จะทำ S-Domain)เพราะฉะนั้นจาก Transfer Function ข้างต้นเราจะทำการ Simulation โดยการใช้โปรแกรม Matlab ในการเขียนโปรแกรมเพื่อศึกษาการตอบสนองในการเปลี่ยนค่าตัวแปรต่างๆ ของมันโดยโปรแกรมสามารถแสดงได้ได้ดังนี้

```
clear;
w=linspace(0,pi,3140);                %specification frequency range
alp0=.5;                              %specification Q-factor
w0=(pi/2);                            %Center frequency
alp1=cos(w0);
z=exp(j*w);
k=(1-alp0)/2;
den=1-z.^(-2);                        %calculate transfer function
num=1-((alp1*(1+alp0))*z.^(-1))+alp0*z.^(-2);
h=den./num;
phase=angle(h);
H=abs(h);
subplot(211);
plot(w,H,'b');
```

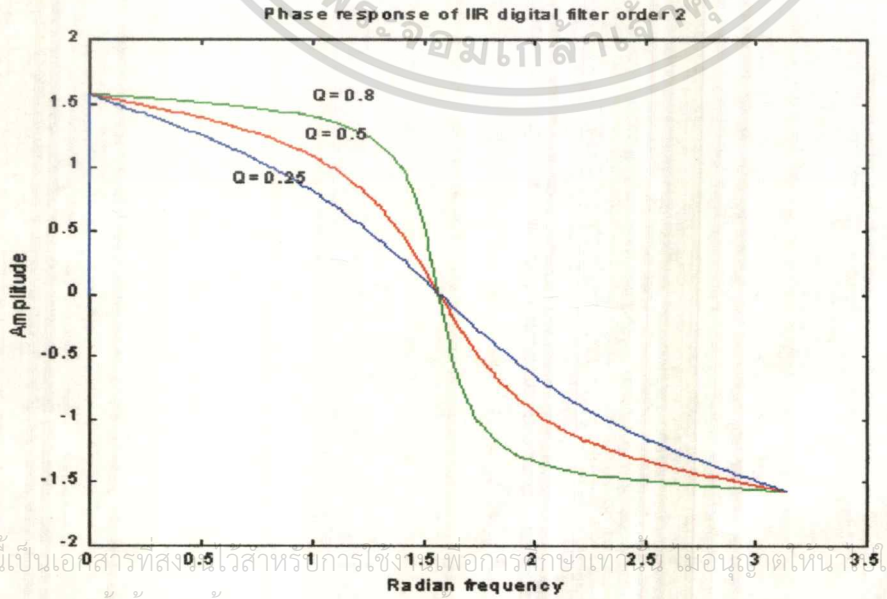
```
subplot(212);
plot(w,phase,'b');
; โปรแกรมนี้มิได้แสดงรายละเอียดในการ Plot ค่าทั้งหมดไว้
```

จากโปรแกรมที่ผ่านมาสามารถ Simulate ผลการตอบสนองต่างๆ ได้ดังนี้



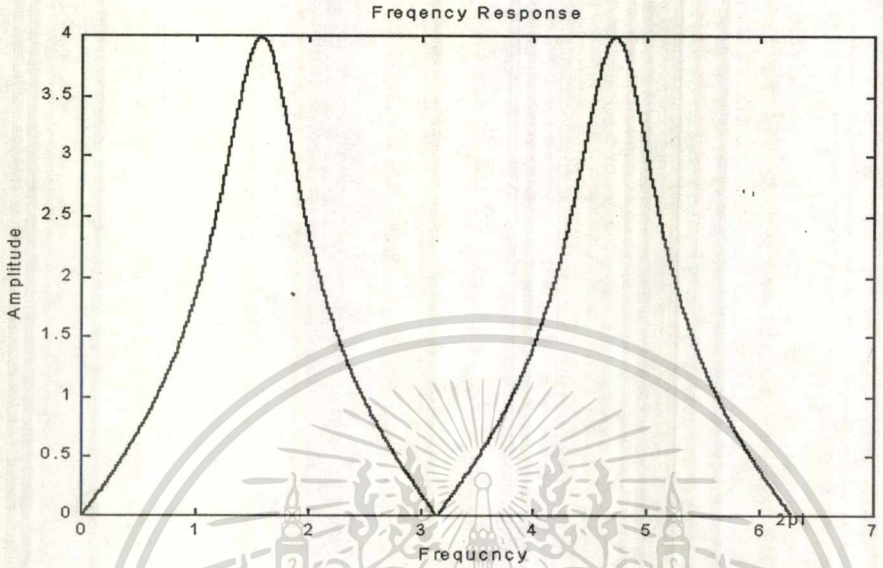
รูปที่ 2.3 การแสดงผลตอบสนองเชิงความถี่ที่ค่า Q ต่างๆ (อยู่ในช่วง 0 - π)

จากผลการตอบสนองที่อ่านได้จากกราฟสามารถแสดงได้ว่าการตอบสนองของ Digital Filter มีความคล้ายกันกับแบบ Analog Filter ที่ผ่านมา ต่างกันตรงที่ Gain ที่ความถี่กลางของ Digital Filter จะขึ้นอยู่กับค่า Q-Factor ของระบบคือ ยิ่งค่า Q สูง Gain ก็จะสูงด้วย (เป็นลักษณะจำเพาะของ Transfer Function)



รูปที่ 2.4 การแสดงผลตอบสนองเชิง Phase ที่ค่า Q ต่างๆ

ส่วนผลตอบสนองเชิง Phase ของ Digital Filter จะเห็นได้ว่ามีลักษณะเหมือนกับแบบ Analog Filter ทุกประการคือถ้ามีการตัดความถี่มากหรือมีค่า Q มากก็จะมีการเปลี่ยนแปลงทาง Phase ที่บริเวณใกล้ๆ ความถี่กลางมากขึ้นตามไปด้วยเหมือนกับ Analog Filter



รูปที่ 2.5 การตอบสนองความถี่ของ IIR Filter

จากรูปการตอบสนองเชิงความถี่ของ IIR Filter รูปนี้เป็นลักษณะการตอบสนองความถี่ที่เป็นคุณสมบัติเฉพาะของ Digital Filter ทุกแบบ ซึ่ง Function ของการตอบสนองความถี่จะเป็น Function ซ้ำคาบโดยจะซ้ำทุกๆค่า π ใน Frequency Domain (Z-Domain) ซึ่งเป็นเงื่อนไขที่สำคัญมากในการนำเอา Digital Filter ไปใช้งานในด้านต่างๆ

2.2.2 การตอบสนองใน Time Domain ของ IIR Filter

การวิเคราะห์การตอบสนองต่างๆของ Filter ในระบบบอนาล็อกจาก Transfer Function โดยการ Simulation สามารถทำสะดวกเฉพาะใน Frequency Domain เท่านั้นเพราะในระบบ S-Domain เมื่อจะทำการ Inverse Laplace Transform จะต้องทราบค่าสมการของสัญญาณ Input จึงจะสามารถทำการแปลงสมการใน S-Domain ไปเป็นสมการทาง Time Domain แล้วจึงสามารถ Simulate ผลการตอบสนองใน Time Domain ได้แต่ใน Frequency Domain ของ Digital Filter (Z-Domain) ไม่จำเป็นต้องรู้สัญญาณ Input ก็จะสามารถหาสมการใน Time Domain ได้เพราะจะเป็นสมการเดียวกับสมการที่จะนำไปใช้งานเขียนโปรแกรมจริงใน Chip DSP โดยสมการจะอยู่ในรูปมาตรฐานดังนี้

$$y(n) = a_0x(n) + a_1x(n-1) + ...a_mx(n-m) + b_1y(n-1)..... + b_my(n-m) \qquad (2.4)$$

จะเห็นได้ว่าสมการ Output ที่เวลาใดๆ $y(n)$ จะเกี่ยวสัญญาณ Input ที่เวลานั้นซึ่งก็คือที่เวลาการ Sampling เดียวกันของสัญญาณ Input และ Output และ สัญญาณ Input และ Output ที่เวลาการ Sampling ที่ผ่านมาซึ่งก็คือเทอม $y(x-1)...y(x-m)$ และ $x(n-1)....x(n-m)$ ซึ่งจะไม่เกี่ยวกับสมการใดๆ ของสัญญาณ Input แต่จะเกี่ยวกับค่าที่มีการ Sampling มาในแต่ละเวลาเท่านั้นเราจึงสามารถ Simulate หาผลตอบสนองทาง Time ได้เลยโดยจาก Transfer Function สามารถหาสมการทาง Time ได้โดยการ Inverse Z-Transform เพื่อนำมาทำการ Simulation หาลักษณะการตอบสนองต่อสัญญาณ Input แบบต่างๆ ได้ (รวมถึงการนำไปเขียนโปรแกรมลงบน Chip DSP) ซึ่งบางทีอาจเรียกว่าสมการผลต่าง

จาก Transfer Function ในสมการ (2.3)

$$H_{BP}(z)=\left[\frac{1-\alpha_0}{2}\right].\left[\frac{1-z^{-2}}{1-\alpha_1(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}}\right]=\frac{y(z)}{x(z)}$$

สามารถแยกออกเป็นองค์ประกอบของ $x(z)$ และ $y(z)$ ได้โดย

$$y(z)\left[1-\alpha_1(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}\right]=x(z)\left[(1-z^{-2}).\left(\frac{1-\alpha_0}{2}\right)\right]$$

$$y(z)-y(z)\alpha_1(1+\alpha_0)z^{-1}+y(z)z^{-2}\alpha_0=\left(\frac{1-\alpha_0}{2}\right).\left[x(z)-x(z)z^{-2}\right] \qquad (2.5)$$

ทำการแปลง Inverse Z-Transform

$$y(k)-y(k-1)\alpha_1(1+\alpha_0)+y(k-2)\alpha_0=\left(\frac{1-\alpha_0}{2}\right).\left[x(k)-x(k-2)\right]$$

$$y(k)=\left(\frac{1-\alpha_0}{2}\right).\left[x(k)-x(k-2)\right]+y(k-1)\alpha_1(1+\alpha_0)-\alpha_0 y(k-2) \qquad (2.6)$$

จากสมการ $y(k)$ เป็นผลการตอบสนองที่ค่า k ต่างๆ (ซึ่งก็คือที่เวลาการ Samplingใดๆ) $x(k)$ คือสัญญาณ Input ที่เวลาการ Sampling เดียวกันส่วน $x(k-n), y(k-n)$ คือค่า Input และ Output ที่เวลาการ Sampling n ครั้งที่ผ่านมา ดั้งนั้นจึงเป็นสมการที่จะนำไปทำการเขียนโปรแกรมเพื่อทำการ Simulation ดูการตอบสนองทางเวลา (รวมถึงการนำไปเขียนโปรแกรมจริงๆ ใน Chip DSP) โดยมีรายละเอียดเกี่ยวกับตัวแปรต่างๆ ดังนี้

α_0 เป็นสัมประสิทธิ์ในการกำหนดค่า Q-Factor ของ Filter ในที่นี้ไว้กำหนดเท่ากับ 0.8 (โดยมีค่า $0 < \alpha_0 < 1$)

α_1 เป็นสัมประสิทธิ์ในการกำหนดค่าความถี่กลางของ Filter กำหนดโดย $\alpha_1 = cos(\omega_0)$ โดย ω_0 คือความถี่กลางของ Filterใน Z-Domain โดยในที่นี้กำหนดไว้เท่ากับความถี่ของสัญญาณ Input

เพื่อดูการตอบสนองต่อการแยกสัญญาณ sine ออกจากสัญญาณรบกวนแบบ Random แบบ Gausian โดยส่วนของอัตราส่วนของสัญญาณต่อสัญญาณรบกวนของ Noise แบบ Gausain ในเทอมของกำลังงานของสัญญาณสามารถกำหนดได้จาก (จาก [3] หน้า 339,[7] หน้า148)

$$SNR = 10\log\left[\frac{v^2}{2\sigma^2}\right] \qquad (2.7)$$

V = Amplitude ของสัญญาณ Input

σ = standard deviation ของ Gausian Noise

ต้องการขนาดของสัญญาณรบกวน กำหนด $SNR = 10db$

$$\sigma = \frac{10^{\left[\log v - \frac{SNR}{20}\right]}}{\sqrt{2}}$$

$$\sigma = \frac{10^{\left[\log 1 - \frac{10}{20}\right]}}{\sqrt{2}}$$

$$= 0.2236$$

ซึ่งสามารถนำไปกำหนดสัมประสิทธิ์ของสัญญาณรบกวนในโปรแกรมที่จะทำการเขียนในหัวข้อต่อๆ ไป

สัญญาณ Input $x(k) = \sin(\omega tk) + n(k)$; t = Sampling time, k =Number of sampling point, $n(k)$ =Noise (Noise แบบ Gausian ใน Matlab สามารถกำหนดโดยคำสั่ง randn)

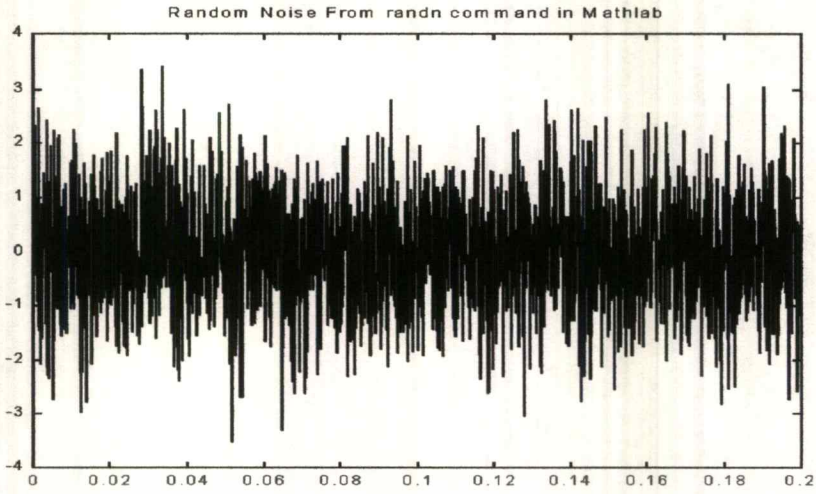
ความถี่ในการ Sampling เท่ากับ 20 kH_z (Sampling Time = 50 μs) ในการ Plot จะ Normalize ไปที่ 1

ส่วนของโปรแกรมเป็นดังนี้

```
clear;
alp1=cos(pi/4.9971);      %cutoff frequency in Z Domain (pi/4.9971)alp1=0.808
alp0=.8;                  %Q-factor
```

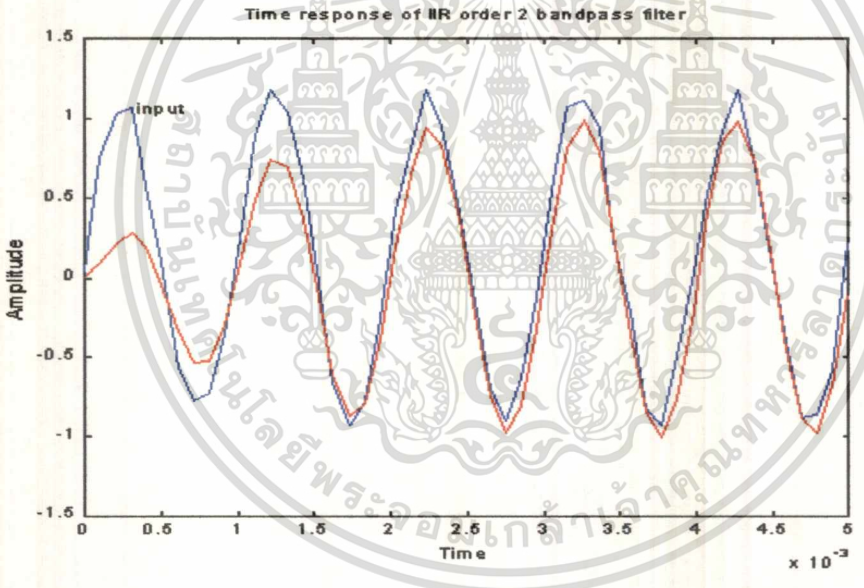
```
rp=5;                   %cycle per period
fun=1e3/rp;             %sampling factor (in this program)
fi=rp*fun;              %frequency input (1k)
sigma=0.2236;           %standard derivation factor (of noise)
k=5e1;                  %sampling point
g=k*fun;                %sampling frequency (10k)
w=linspace(0,k/g,k);
%input signal with 10db signal to noise ratio
xi=sin(w.*2*pi*fi)+sigma*randn(size(w));
xs=[0,0,(xi)];
for r=1:k;
  n=r;
  t=(r-1);
  x=xs(:,(n+2));
  x1=xs(:,(n+1));
  x2=xs(:,n);
  if t==0;
    y1=0;
    y2=0;
  else
    y2=y1;
    y1=y;
  end
  y=(((1-alp0)/2)*(x-x2)+((alp1)*(1+alp0)*y1)-(alp0*y2));
  z(r)=y;
end
plot(w,xi,'b',w,z,'r');
title('Time response of IIR order 2 bandpass filter');
xlabel('Time');
ylabel('Amplitude')
```

ผลจากการ simulate จากโปรแกรมข้างต้นได้ผลดังนี้



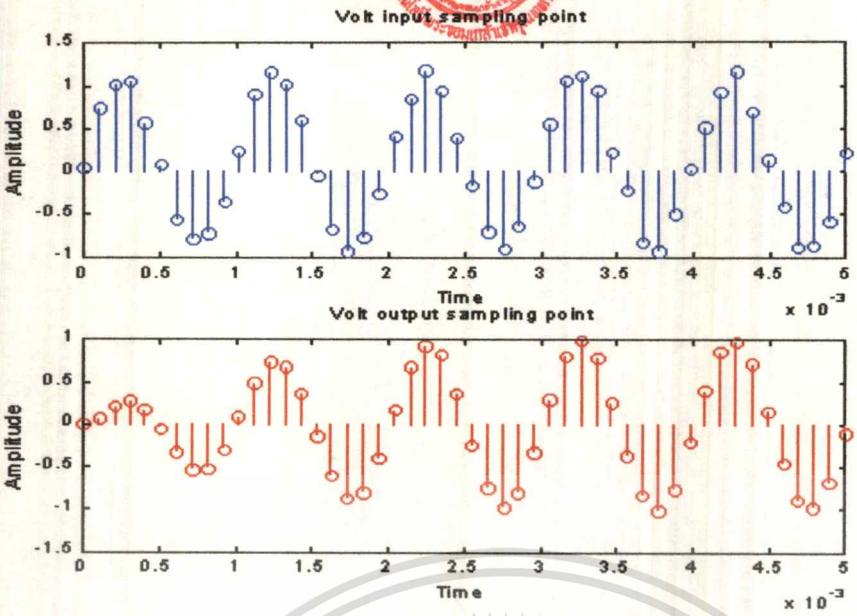
รูปที่ 2.6 สัญญาณรบกวนที่กำเนิดจากคำสั่ง randn ในโปรแกรม Matlab

ซึ่งจะใช้สัญญาณนี้เป็นตัวสัญญาณรบกวนในการ Simulation ในหัวข้อต่อๆ ไปทั้งหมด



รูปที่ 2.7 การแสดงผลการตอบสนองใน Time Domain

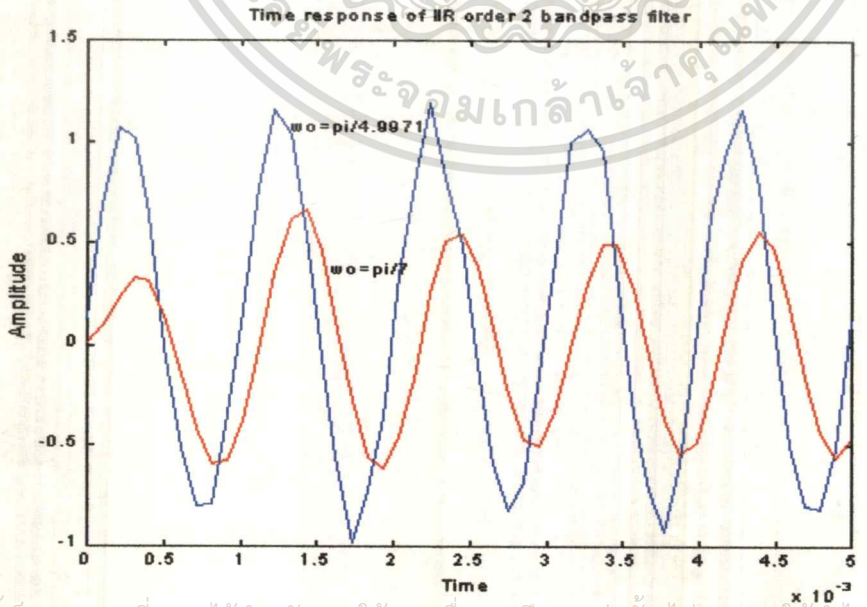
จากรูปการตอบสนองใน Time Domain เมื่อความถี่กลางของ Filter ตรงกับความถี่ของ Input จะเห็นว่า Phase ของสัญญาณ Input และ Output จะ Inphase กันหรือเท่ากัน โดยในตอนแรกจะมีการปรับขนาดเล็กน้อยเนื่องจากเงื่อนไขเริ่มต้นของระบบจะนำเอาสัญญาณ Output ที่เวลาก่อนหน้านั้นมาคิดด้วย ซึ่งตอนเริ่มต้นส่วนนี้เป็นศูนย์ทำให้มีการปรับค่าเพื่อเข้าหาค่าที่แท้จริงดังจะเห็นได้จากว่าเมื่อเวลาผ่านไปได้ระยะหนึ่งรูปคลื่นจะมีลักษณะเข้าใกล้ Input มากขึ้นจนมีขนาดเกือบเท่ากันที่ไม่เท่ากันเพราะมี สัญญาณรบกวนอยู่ด้วย แต่ที่ไม่สามารถเห็นสัญญาณรบกวนชัดเพราะค่าสัญญาณอินพุทที่กำหนดในการคำนวณนี้จะเป็นค่าที่ได้มาจากการ Sampling แล้ว



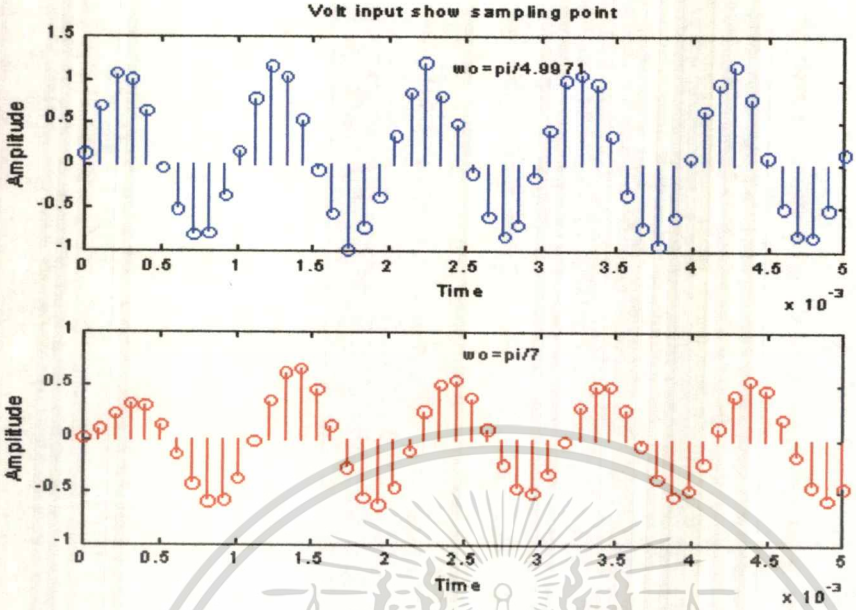
รูปที่ 2.8 การแสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

จากรูปแสดงการ Plot เพื่อการแสดงจุด Sampling Point ซึ่งเป็นลักษณะของสัญญาณแต่ละจุดจริงๆ ที่ได้จากการ Sampling (ของสัญญาณ Input) และเป็นค่าที่ส่งออกไปให้ Digital to Analog ในการใช้งานจริงซึ่งที่ Output ของ D/A จะมี Lowpass Filter ตัดองค์ประกอบทางความถี่สูงของสัญญาณ Sampling ทิ้ง ซึ่งสัญญาณที่ Output จริงๆ จะเป็นลักษณะ sine ที่ Smooth มากกว่าการ Simulation มาก

กราฟต่อไปนี้ได้มาจากการเปลี่ยนค่าความถี่กลางของ Filter (ω_0) ให้น้อยกว่าสัญญาณ Input จะเห็นว่าลักษณะของสัญญาณ Output จะมีการลดทอนรวมทั้ง Phase ก็มีการ Shift ออกไป

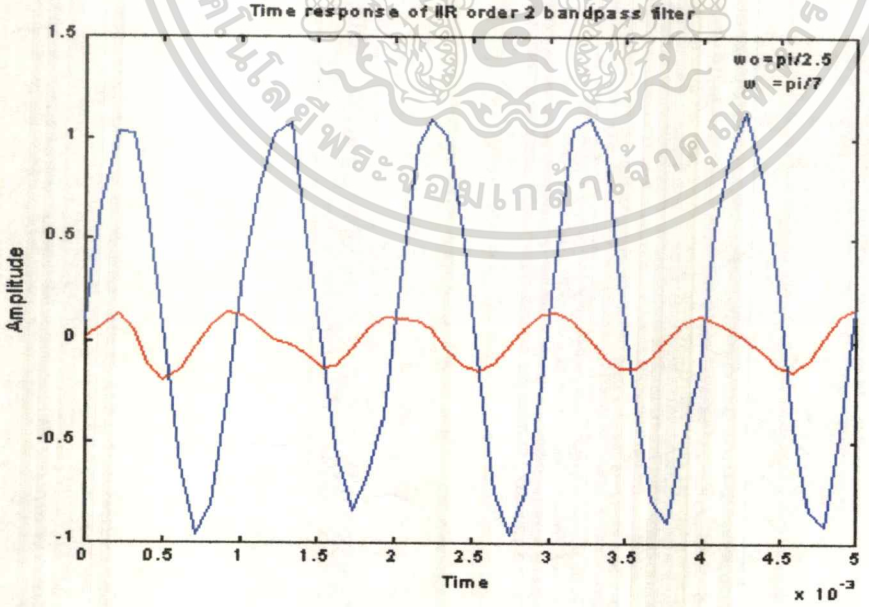


รูปที่ 2.9 การแสดงผลเมื่อมีการปรับค่า ω_0 ให้น้อยกว่าสัญญาณ Input

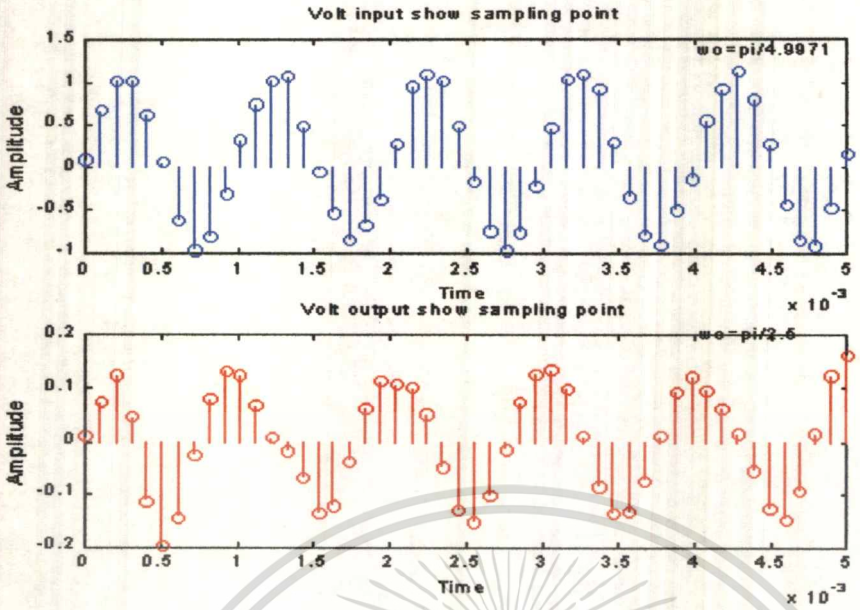


รูปที่ 2.10 การแสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

กราฟต่อไปนี้ได้มาจากการเปลี่ยนค่าความถี่กลางของ Filter (ω_0) ให้มากกว่าสัญญาณ Input จะเห็นว่าลักษณะของสัญญาณ Output จะมีการลดทอนรวมทั้ง Phase ก็มีการ Shift ออกไป



รูปที่ 2.11 การแสดงผลเมื่อมีการปรับค่า ω_0 ให้มากกว่าสัญญาณ Input



รูปที่ 2.12 การแสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

จากทั้งหมดที่ผ่านมาได้แสดงถึงรูปคลื่นของสัญญาณ Input และ Output ที่รูปแบบต่างๆ

2.3 การวิเคราะห์ผลตอบสนองความถี่จากสมการ ใน Time Domain

ในขั้นตอนนี้เราจะทำการ Simulation ผลตอบสนองเชิงความถี่ของ Digital IIR Filter จากสมการทาง Time Domain ที่ได้ทำการ Simulation ผ่านมาแล้วในหัวข้อที่ผ่านมาเพื่อดูผลการตอบสนองความถี่จากสมการทาง Time โดยจะทำการกำหนดความถี่ทางด้าน Input ค่าหนึ่งแล้วเก็บค่าแรงดันทาง Output ไว้ แล้วทำการเพิ่มความถี่ทาง Input แล้วก็เก็บค่าแรงดันทาง Output ทำอยู่อย่างนี้จนถึงค่าที่กำหนด (คล้ายเป็นการจำลองการทำงานของเครื่องมือวัดการตอบสนองของความถี่ของในทางอนาล็อก)

2.3.1 ความสัมพันธ์ระหว่างความถี่ใน Time และ Z-Domain

ก่อนอื่นจะต้องรู้ความสัมพันธ์ระหว่าง ความถี่ใน Time Domain และ Frequency Domain (Z-Domain) ก่อนซึ่งทำให้พิจารณาผลตอบสนองเชิงความถี่ได้อย่างถูกต้องโดยใน Transfer Function ที่นำมาทำการ Simulation นี้จะมีความสัมพันธ์ทางความถี่เหมือนกับการแปลง แบบ Impulse Invarious ซึ่งขึ้นอยู่กับคาบเวลาการ Sampling ตามสมการต่อไปนี้

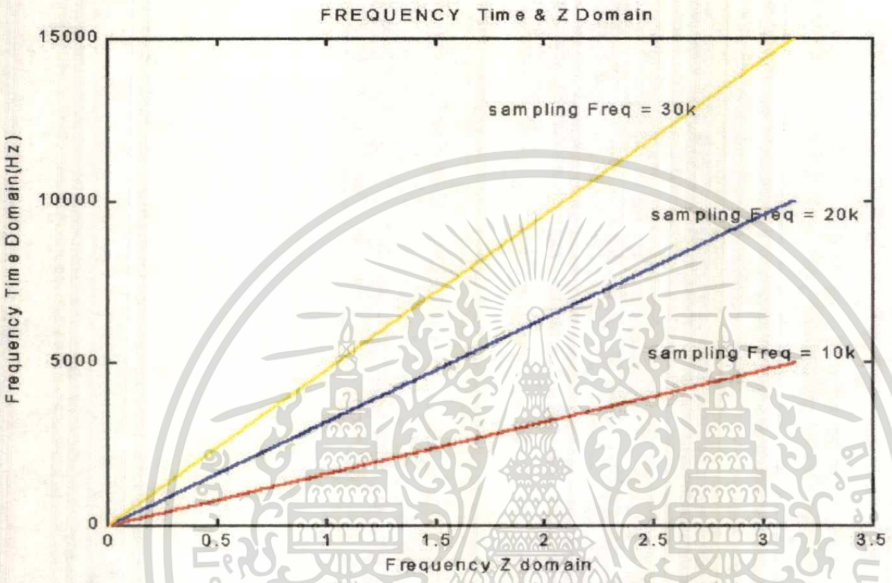
$$\omega = \frac{\Omega}{T} \qquad (2.8)$$

โดย ω =ความถี่ใน Time Domain $(2\pi f)$

Ω = ความถี่ใน Frequency Domain(Z Domain)

T = คาบเวลาการ Sampling

โดยสามารถ Plot แสดงความสัมพันธ์ระหว่างความถี่ใน Time Domain และความถี่ใน Z- Domain เมื่ออัตราการ Sampling เปลี่ยนไป สามารถแสดงได้ดังรูป



รูปที่ 2.13 แสดงความสัมพันธ์ระหว่างความถี่ใน Time Domain และ Z-Domain

โดยที่ความสัมพันธ์ดังกล่าวจะใช้ได้ในขอบเขตที่ความถี่ใน Time Domain ที่น้อยกว่าครึ่งหนึ่งของความถี่ Sampling มิฉะนั้นจะทำให้การตอบสนองความถี่ไม่เป็นไปตามเงื่อนไขที่ออกแบบไว้ คือจะเกิดการซ้อนทับของความถี่ (Aliasing)

จากนั้นจะทำการเขียนโปรแกรมที่จะ Simulation โดยใช้ Matlab ได้โดย

กำหนดค่า Q (α_0) = 0.8

ความถี่ Sampling 10 KHz

ความถี่กลางของ Band pass Filter เท่ากับ $.5\pi$

จะทำให้ได้ความถี่ใน Time Domain ซึ่งจากสมการที่ 2.8 จะได้

$$2\pi f = \frac{.5\pi}{10^{-4}}$$

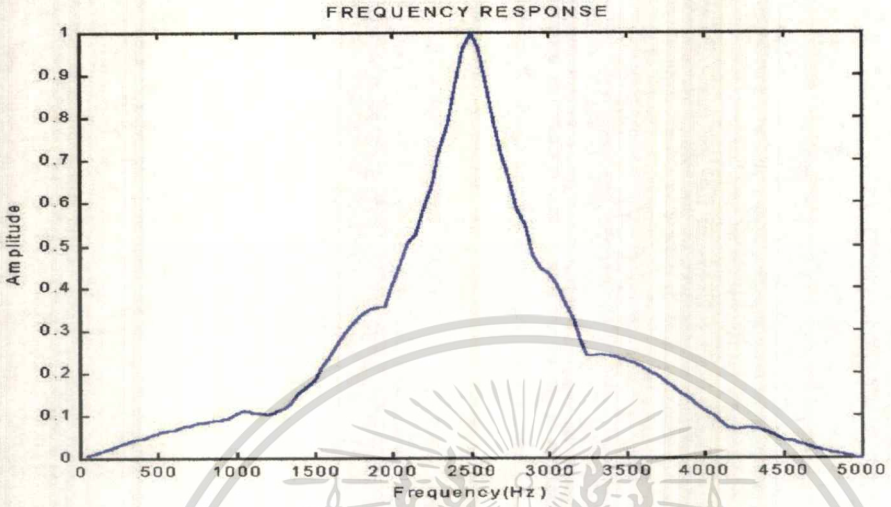
$$f = \frac{.5\pi}{2\pi 10^{-4}} = 2500Hz$$

จากนั้นสามารถเขียนโปรแกรมได้ดังนี้

```
clear;
for m=1:100;
  fi=m*5e1;          %frequency input
  sf=10e3;           %sampling   frequency
  sn=.2236;          %standard derivation factor of noise
  rp=40;
  t=0:(1/sf):(rp*(1/fi));
  wi=(2*pi*fi);
  xn=sin(t.*wi);     %input signal
  xi=[0,0,(xn)];
  alp1=cos(pi/2);    %cut off frequency
  alp0=.8;           % q factor
  ns=size(xn);
  n=ns(:,2);
  for k=1:n;
    x=xi(:,(k+2));
    x1=xi(:,(k+1));
    x2=xi(:,k);
    if k==1;
      y1=0;
      y2=0;
    else
      y2=y1;
      y1=y;
    end
    y=(((1-alp0)/2)*((x-x2))+((alp1)*(1+alp0)*y1)-(alp0*y2));
    yo(k)=y;
    xd(k)=x2;
  end
  fr(m)=max(yo);
  yo=0;
```

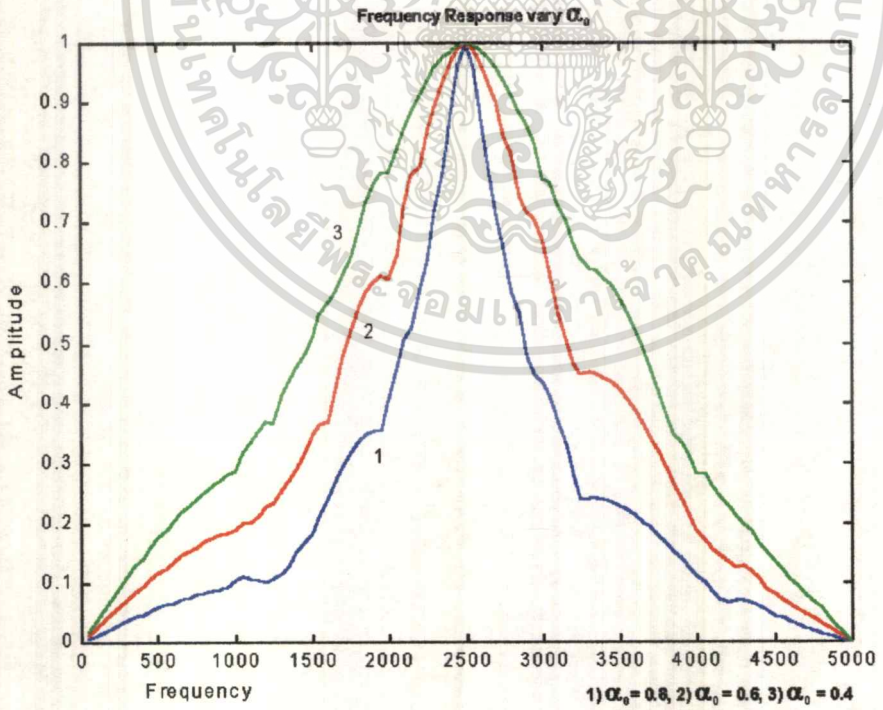
```
end
plot(f,fr)
```

ซึ่งจะได้ผลตอบสนองความถี่ดังรูป



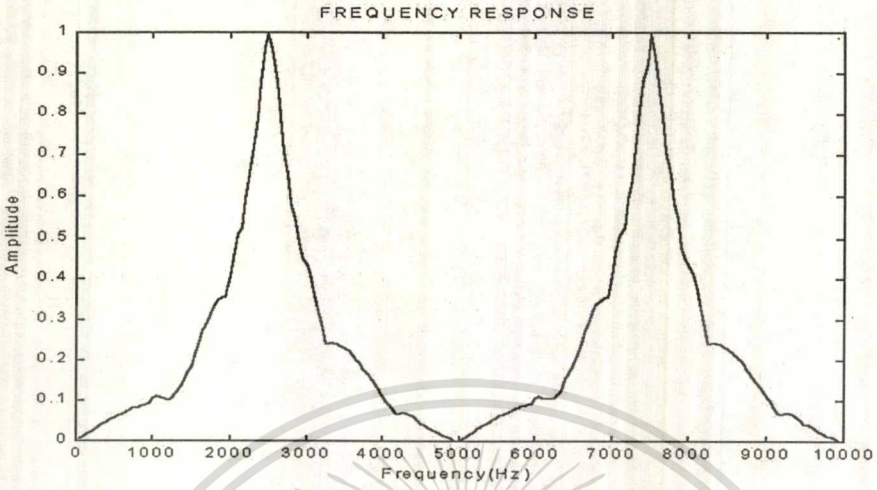
รูปที่ 2.14 แสดงผลตอบสนองทางความถี่

จากรูปจะเห็นว่าความถี่กลางของ Filter จะตรงกับที่คำนวณไว้คือ 2500 Hz



รูปที่ 2.15 การตอบสนองความถี่ของ Filter เมื่อมีการเปลี่ยนแปลงค่า Q

จากรูปจะเห็นว่าเมื่อมีการเปลี่ยนค่า Q-factor ลักษณะการตัดสัญญาณที่ความถี่ข้างๆ ของความถี่กลางจะคมขึ้นแต่การเพิ่มค่า Q-factor จะไม่มีผลกับการตอบสนองทาง Amplitude เหมือนกับการ Simulate ผลตอบสนองทางความถี่จาก สมการใน Z-Domain



รูปที่ 2.16 แสดงการตอบสนองความถี่เป็น Function ซ้ำคาบ

รูปนี้จะแสดงให้เห็นว่าการตอบสนองเชิงความถี่ของ Digital Filter จะเป็น Function ซ้ำคาบเหมือนกับที่ได้กล่าวมาแล้วในหัวข้อของการ Simulate จากสมการใน Z-Domain แต่ในสมการทาง Time นั้นจะไม่เหมือนกันตรงที่จะเป็น Function ซ้ำคาบที่ครึ่งหนึ่งของความถี่การ Sampling ซึ่งจะตรงกับเงื่อนไขของ Nyquist ซึ่งในการใช้งานนั้นจะสามารถใช้ได้ในความถี่ช่วงแรกเท่านั้นคือในช่วงไม่เกินครึ่งหนึ่งของความถี่ Sampling เพราะการตอบสนองความถี่หลังจากความถี่นี้จะผิดพลาด คือเกิดการซ้อนทับของความถี่ (Aliasing) เพราะฉะนั้นในบางครั้ง Digital Filter จะถูกเรียกว่า Band Limit Filter

2.4 Adaptive IIR Filter

คือ Filter ที่มีการปรับค่าตัวแปรต่างๆ ตามที่ต้องการได้อย่างอัตโนมัติซึ่งได้กล่าวในบทที่ 1 มาแล้วว่าเป็นลักษณะเฉพาะที่เป็นข้อดีของ การประมวลผลสัญญาณดิจิตอลที่สามารถทำได้ โดยในที่นี้จะทำการปรับค่าสัมประสิทธิ์ที่เป็นค่ากำหนดความถี่กลางตามความถี่หลักของสัญญาณทางด้าน Input ที่เข้ามาเมื่อมีการเปลี่ยนแปลงทาง Input ในที่นี้จะเป็นการปรับค่าความถี่กลางเมื่อความถี่ Input เปลี่ยนไป โดยจะใช้ IIR Band pass Filter order 2 ตามที่ได้แสดงไว้ที่ผ่านมา โดยสัมประสิทธิ์ที่จะเปลี่ยนไปคือ α_1 ต่อไปจะแทนด้วย $\alpha_1(k)$ จาก Transfer function ในสมการที่ 2.3

$$H_{BP}(z) = \left[\frac{1-\alpha_0}{2}\right]\cdot\left[\frac{1-z^{-2}}{1-\alpha_1(k)(1+\alpha_0)z^{-1}+\alpha_0 z^{-2}}\right]$$

สามารถ inverse Z-Transform เป็นสมการทาง Time Domain ได้ดังนี้

$$y(k)=\left[\frac{1-\alpha_0}{2}\right].[x(k)-x(k-2)]+y(k-1)\alpha_1(k).(1+\alpha_0)-\alpha_0 y(k-2) \quad (2.9)$$

จากสมการทาง Time Domain จะมีความสัมพันธ์กับ Z-Domain โดยการเปลี่ยนแปลงระหว่าง $y^2(k)$ กับ $H^2(e^{j\omega})$ เป็นสัดส่วนโดยตรงต่อกัน

จาก Stochastic Gradient Adaptive Algorithms ซึ่งเป็น Algorithms ในการ Update Vector ใดๆ โดยเทียบกับตัวที่ต้องการเพื่อให้สามารถให้ค่าที่ทำให้ตัวแปรที่ต้องการมีค่าสูงสุดสามารถกำหนดโดย

$$w_{k+1}=w_k-\frac{\mu_0}{2}\nabla\left[\varepsilon^2(k)\right] \quad (2.10)$$

ในที่นี้เราจะทำการ Update สัมประสิทธิ์ $\alpha_1(k)$ เทียบกับค่าของ $y(k)$ เพื่อหาค่าของ α_1 ที่ทำให้ค่า $y(k)$ มีค่าสูงสุดโดยจาก สมการ 2.10 จะได้

$$\alpha_1(k+1)=\alpha_1(k)+\frac{\mu}{2}.\frac{\partial(y^2(k))}{\partial\alpha_1(k)}$$
$$\alpha_1(k+1)=\alpha_1(k)+\mu.y(k).\frac{\partial y(k)}{\partial\alpha_1(k)} \quad (2.11)$$

โดยกำหนดให้เทอม $\frac{\partial y(k)}{\partial\alpha_1(k)}=\psi(k)$ ซึ่งจะเรียกว่าสัญญาณควบคุม Adaptive ของ $\alpha_1(k)$ เป็นพารามิเตอร์ที่ใช้ในการ Update ค่าถัดไปสามารถหาได้ดังนี้

$$\frac{\partial y(k)}{\partial\alpha_1(k)}=\psi(k)=\frac{\partial\left[\left(\frac{1-\alpha_0}{2}\right).[x(k)-x(k-2)]+y(k-1).\alpha_1(k).(1+\alpha_0)-\alpha_0.y(k-2)\right]}{\partial\alpha_1(k)}$$

$$=(1+\alpha_0)\left[\alpha_1(k)\frac{\partial y(k-1)}{\partial\alpha_1(k)}+y(k-1)\right]-\alpha_0\frac{\partial y(k-2)}{\partial\alpha_1(k)} \quad (2.12)$$

จะเห็นว่าจะติดเทอมของ $\frac{\partial y(k-i)}{\partial\alpha_1(k)}$ (โดย i เป็นจำนวนเต็ม) ซึ่งจริงๆ จะไม่สามารถหาค่าได้เพราะต้องทำการแตก $y(k-1), y(k-2)$..เป็นการตอบสนองของส่วนที่ผ่านมามากมาย เพราะฉะนั้นจึงต้องมีการประมาณค่าส่วนที่จะพิจารณาคือ ถ้าค่าของ μ มีค่าน้อยพอจะทำให้การเปลี่ยนแปลงของ $y(k)$ แต่ละครั้งมีค่าน้อยทำให้สามารถประมาณได้ว่า

$$(k-i)=\frac{\partial y(k-i)}{\partial \alpha_1(k-i)}\approx\frac{\partial y(k-i)}{\partial \alpha_1(k)} \qquad (2.13)$$

จะได้

$$(k)=(1+\alpha_0).\alpha_1(k).\psi(k-1)+(1+\alpha_0).y(k-1)-\alpha_0\psi(k-2) \qquad (2.14)$$

จากสมการของ $\alpha_1(k+1)$ และ $\psi(k)$ ข้างต้น จะเป็นสมการที่นำไปทำการ Update ค่าของ α_1 เพื่อการปรับค่าความถี่ทาง Input ที่ทำให้ค่าของ $y(k)$ มีมากที่สุด ซึ่งก็คือความถี่กลางนั่นเอง (เพราะ gain ที่ความถี่กลางมีค่ามากที่สุด)

จากนั้นจึงนำเอาสมการข้างต้นมาเขียนโปรแกรม Simulation เพื่อดูผลการตอบสนองต่อสัญญาณ Input ในลักษณะต่างๆ รวมถึงการเปลี่ยนแปลงสัมประสิทธิ์ที่มีการปรับค่าตามสัญญาณ Input ว่ามีการเปลี่ยนแปลงลักษณะไหนอย่างไรเพื่อที่จะนำเอาสมการเหล่านี้ไป เขียนโปรแกรมใช้ประมวลผลสัญญาณจริงๆบน Chip DSP

โดยมีเงื่อนไขตอนเริ่มต้นดังต่อไปนี้

-สัญญาณ Input เป็นสัญญาณ sine ความถี่ 1000 Hz กับ Noise แบบสุ่ม (มี SNR=10dB) เหมือนกับหัวข้อที่ผ่านมา

-ความถี่ในการ sampling เท่ากับ 10 kH_z

-โดยกำหนดเงื่อนไขเริ่มต้นของ $\alpha_1 = 0$; $\alpha_0 = 0.5$ โดยสามารถคำนวณความถี่เริ่มต้นใน Time Domain ได้เท่ากับ 2.5 kHz

-ค่า Step Size Parameter (μ)= 0.0006

โดยโปรแกรมสามารถแสดงได้ดังนี้

```
clear
alp0=.5;                          %Q-factor
fi=3.5e3;                         %frequency input
sf=10e3;                          %sampling  frequency
fa=.2236;                         %standard derivation factor of noise
k=20000;                                %number of sampling point
w=0:(1/sf):(k*(1/sf));
xi=sin(w.*2*pi*fi)+fa*randn(size(w));       %calculate input signal+noise
xs=[0,0,(xi)];                            %delay for calculate output
mue=0.00015;                              %mue0/2
```

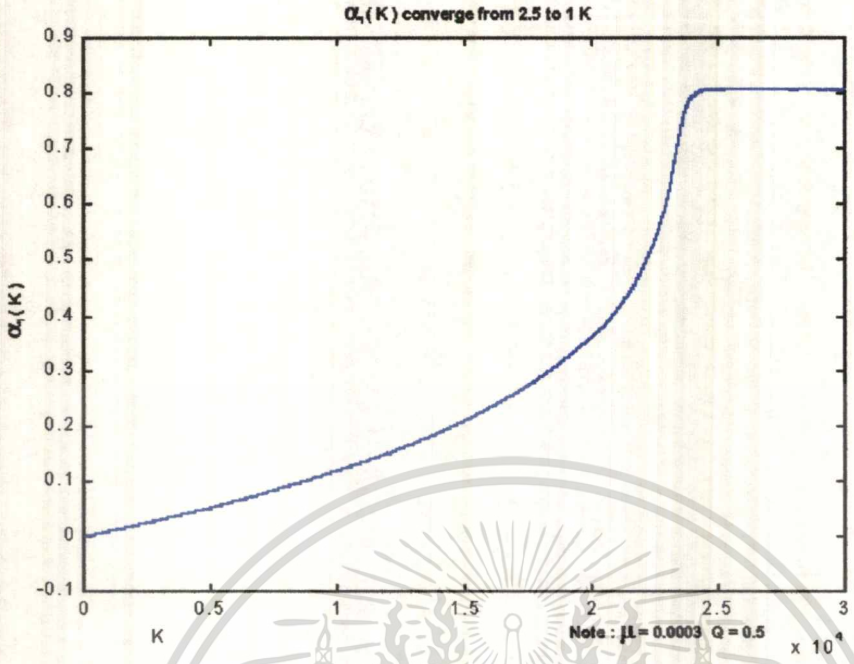
```
for r=1:k
  n=r;
  t=(r-1);
  x=xs(:,(n+2));
  x1=xs(:,(n+1));
  x2=xs(:,n);
  if t==0;
    y1=0;
    y2=0;
    alp1=0;                                  %initial frequency in Z domain
    phi1_1=0;
    phi1_2=0;
  else
    y2=y1;
    y1=y;
  end
  y=(((1-alp0)/2)*((x-x2))+((alp1)*(1+alp0)*y1)-(alp0*y2));      %calculate output
  z(n)=y;                                                        % store output  to z   (to plot)
  phi1=(alp1*(1+alp0)*phi1_1)-(alp0*phi1_2)+((1+alp0)*y1);       % calculate phi
  up=(mue*y*phi1);
  alp1=alp1+up;                                                  % update alp1 for next time
  phi1_2=phi1_1;
  phi1_1=phi1;
  alp1dif(n)=alp1;                                               % store alp1  alp1dif  (to plot)
  end
  Z=[(z),0]
```

(จะเป็นส่วนหลักในการคำนวณ โดยไม่ได้แสดงรายละเอียดในการ plot)

Note: ค่าที่ สามารถ Plot ดูได้คือ alp1diff , z และ xi

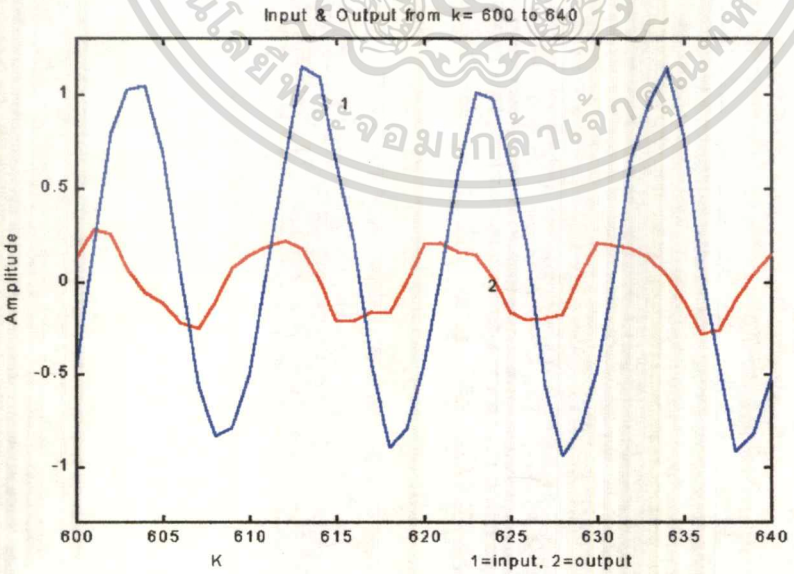
: ยิ่งคำนวณที่ค่า K มาจะใช้เวลาในการคำนวณมาก ควรใช้ computer ที่มีความเร็วสูงจะสะดวก

2.4.1 ผลตอบสนองที่เวลาต่าง ๆ ในการติดตามสัญญาณ Input

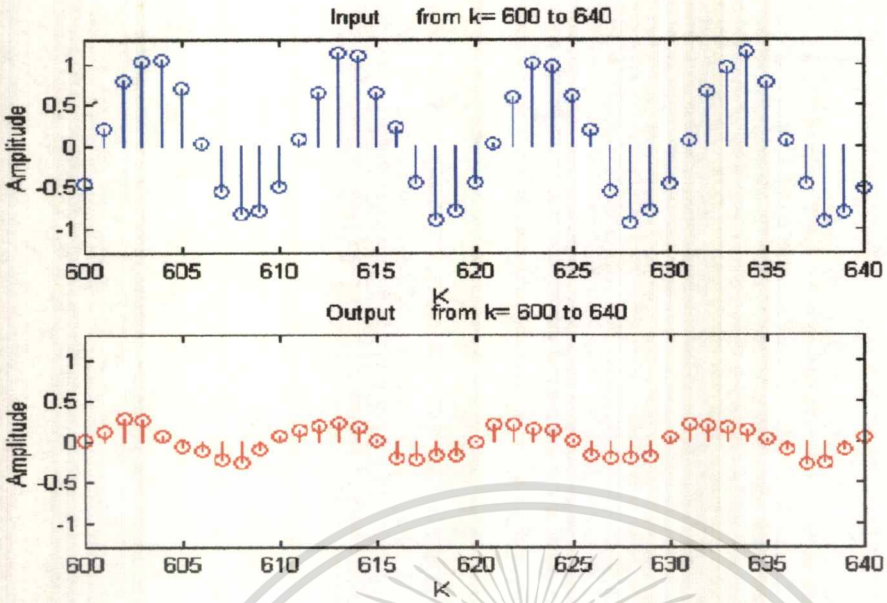


รูปที่ 2.17 แสดงการปรับค่า $\alpha_1(k)$ เข้าหาค่าตามความถี่สัญญาณ Input

ในกราฟจะทำการ Normalize คาบเวลาการ Sampling ให้เท่ากับ 1 เพื่อง่ายต่อการพิจารณา จากรูปแสดงการปรับค่าความถี่กลางของ Filter (ในที่นี้ได้กำหนดตอนเริ่มต้นไว้ที่ 0 (2.5KHz)) เข้าหาค่าของความถี่ Input (1 KHz) จะเห็นว่าเมื่อมีการปรับเข้าสู่ค่าที่ถูกต้องค่าจะไม่มีการเปลี่ยนแปลง

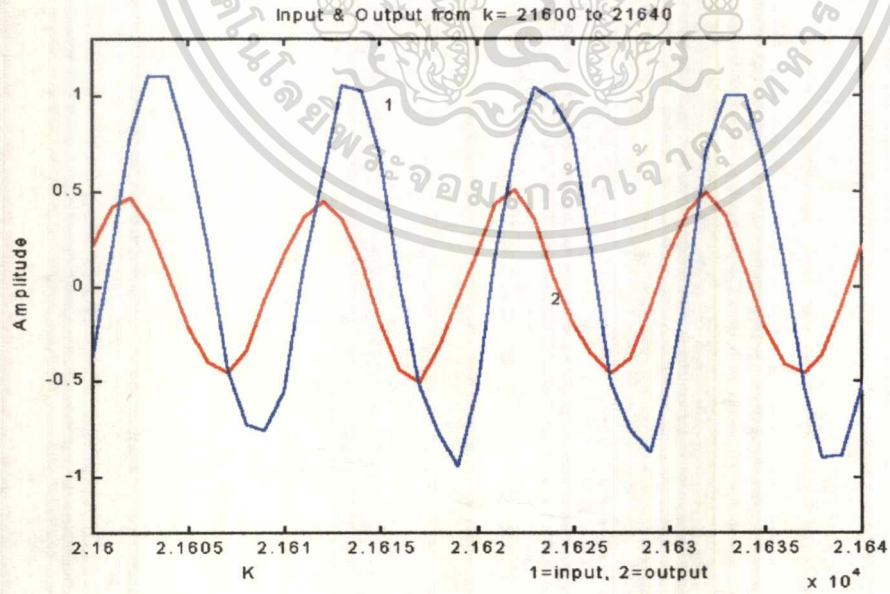


รูปที่ 2.18 แสดงสัญญาณ Input และ Output ขณะที่ระบบเพิ่งเริ่มทำงาน

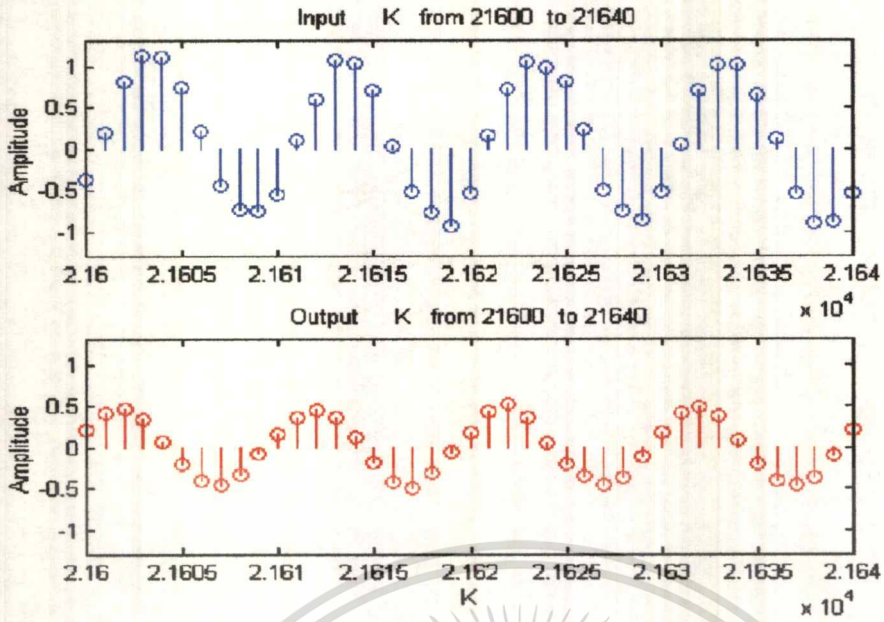


รูปที่ 2.19 แสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

จากรูปทั้งสองแสดงรูปคลื่นที่เวลาแรกๆ ขณะที่ความถี่ Input และความถี่กลางของระบบยังห่างกันอยู่มากจะเห็นว่า รูปคลื่นที่ Output มีขนาดเล็กกว่า Input มากรวมถึงมีการ Shift Phase ด้วยโดยในขณะนี้ระบบกำลังปรับค่าความถี่กลางของมันอยู่ซึ่งจะค่อยๆ ปรับไปจนถึงค่าที่ถูกต้องดังจะเห็นในรูปถัดไป (ดูที่กราฟในแนวนอนคือค่า K ซึ่งก็คือจำนวนครั้งของการ Sampling)

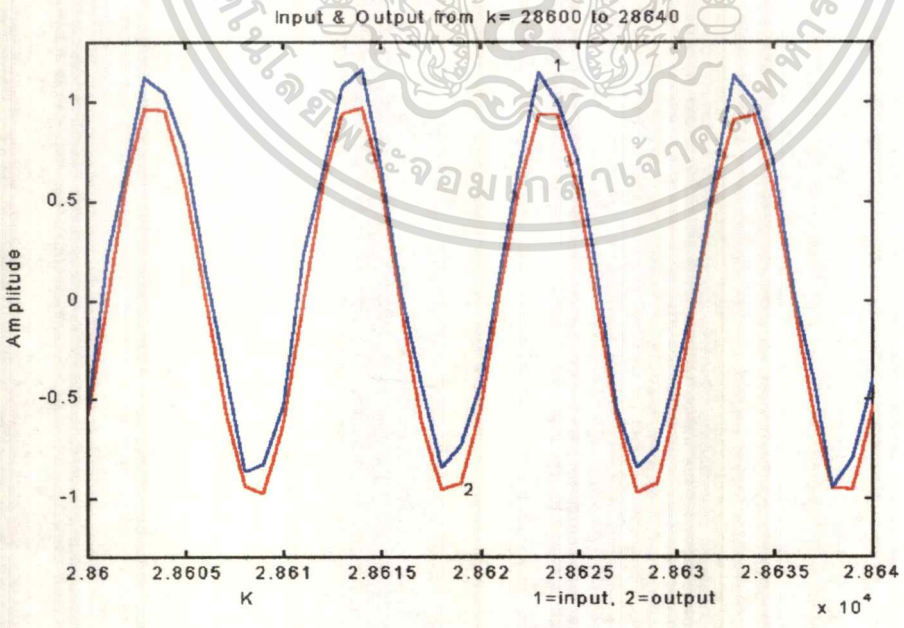


รูปที่ 2.20 แสดงรูปคลื่นที่ปรับค่า $\alpha_1(k)$ เข้าใกล้ค่า Input มากขึ้น

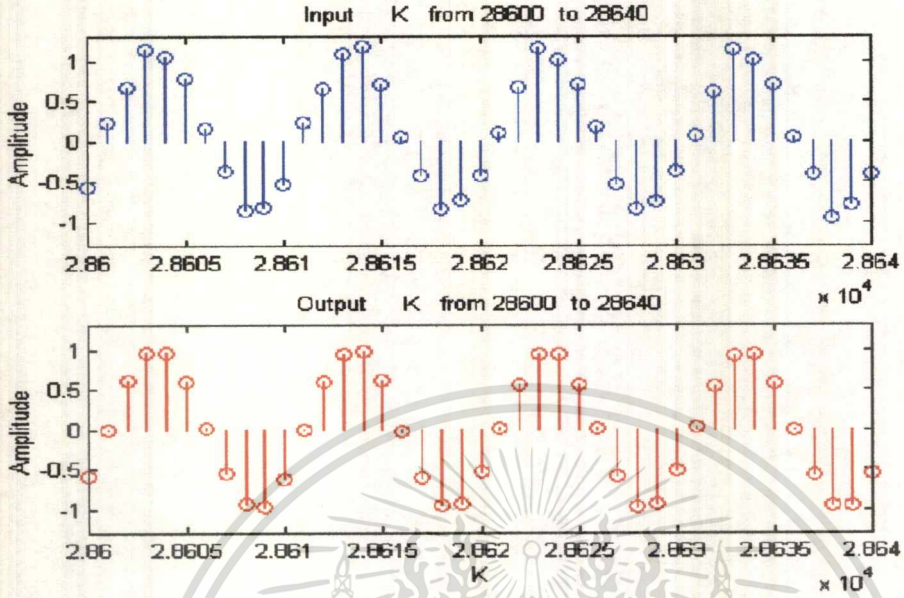


รูปที่ 2.21 แสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

จากรูปแสดงให้เห็นว่ารูปคลื่นจะมีลักษณะเข้าใกล้สัญญาณ Input มากขึ้นกว่ารูปที่ผ่านมารวมทั้ง Phase จะมีการเพี้ยนน้อยลง รวมทั้งมีขนาดเข้าใกล้ Input มากขึ้นเพราะในขณะนี้ค่าความถี่กลางของระบบกำลังเข้าใกล้ความถี่ของรูปคลื่นทาง Input มากขึ้นโดยดูรูปค่า $\alpha_1(k)$ ที่ผ่านมาประกอบ



รูปที่ 2.22 แสดงให้เห็นเมื่อค่าของ $\alpha_1(k)$ ถูกปรับจนเท่ากับค่าของสัญญาณ Input

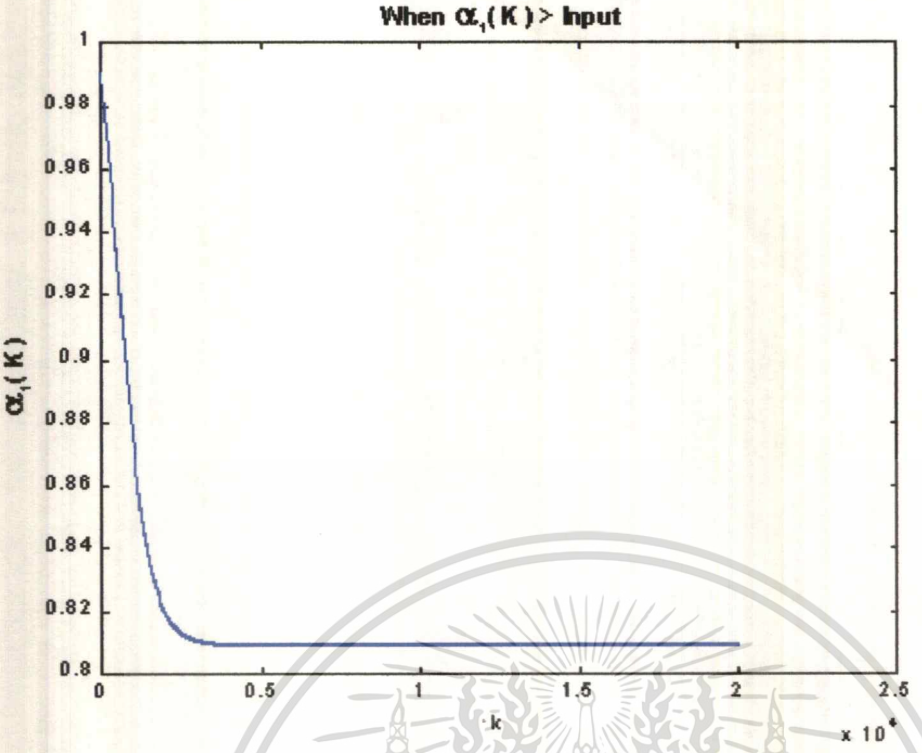


รูปที่ 2.23 แสดงสัญญาณ Input และ Output ในรูปของ Sampling Point

จะเห็นว่าลักษณะของสัญญาณ Output จะมีลักษณะเกือบเหมือนกับสัญญาณ Input การตอบสนองทาง Phase จะ Inphase กับ Output ทุกประการแต่ในที่นี้ที่อินพุทมีสัญญาณรบกวนอยู่ด้วยซึ่งลักษณะการตอบสนองขณะนี้จะเหมือนกับการตอบสนองของ Filter ธรรมดาที่กล่าวผ่านมาในหัวข้อที่ 2.2.2 ในขณะที่ความถี่ทาง Input ตรงกับความถี่กลางของ Filter ซึ่งการแยกสัญญาณรบกวนออกจากสัญญาณ sine นั้นไม่สามารถเห็นได้ชัดเพราะสัญญาณที่นำมาแสดงเป็นสัญญาณที่เหมือนสัญญาณที่ทำการ Sampling มาแล้วซึ่งได้กล่าวมาแล้วในหัวข้อ 2.3

2.4.2 การติดตามสัญญาณ Input ที่เงื่อนไขต่าง ๆ

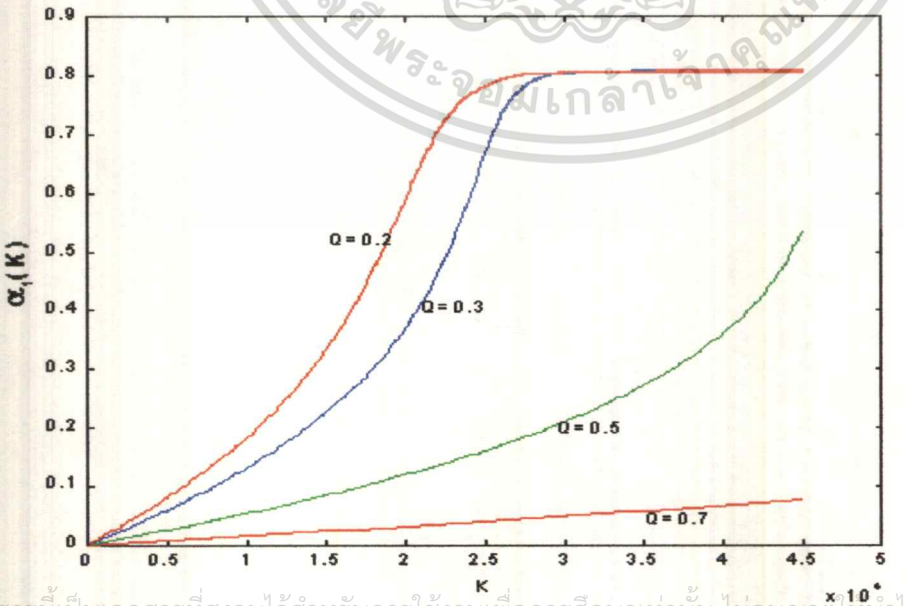
ในส่วนนี้จะทำการวิเคราะห์หาการตอบสนองค่าของการติดตามสัญญาณ Input เมื่อทำการเปลี่ยนตัวแปรต่างๆ ตลอดจนการเปลี่ยนค่าของ α_1 (k) ในลักษณะต่างๆ



รูปที่ 2.24 แสดงการปรับค่าลดลงของ $\alpha_1(k)$

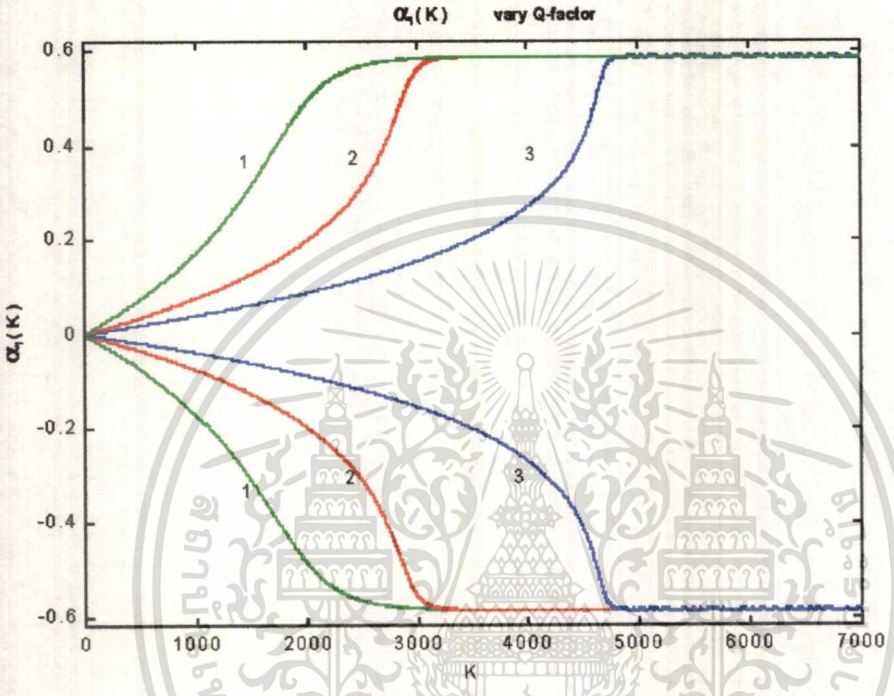
จากรูปแสดงการตอบสนองของค่า $\alpha_1(k)$ เมื่อมีการกำหนดเงื่อนไขเริ่มต้นให้ $\alpha_1(k)$ มีค่ามากกว่าค่าของสัญญาณ Input(ค่าของความถี่กลางน้อยกว่าค่าความถี่ของ input) เพื่อแสดงถึงการลดลงของค่า $\alpha_1(k)$

การเปลี่ยนค่า α_0



รูปที่ 2.25 แสดงการตอบสนองเมื่อเปลี่ยนค่า α_0 ค่าต่างๆ

จากรูปแสดงให้เห็นว่าการเปลี่ยนแปลงค่าของ α_0 มีผลต่อการลู่เข้าหา $\alpha_1(k)$ ช้าหรือเร็วต่างๆ กัน ทั้งนี้เพราะการเปลี่ยนแปลงค่า Q มากขึ้นมีผลทำให้ Amplitude ของสัญญาณ Output ที่ไม่ใช่ความถี่กลางมีขนาดเล็กลง ทำให้เทอมของสัญญาณ $\psi(k)$ ที่ใช้ใน Update มีขนาดเล็กลงทำให้การเปลี่ยนแปลงมีค่าน้อย ในขณะที่ความถี่กลางมีขนาดมากหรือน้อยกว่าความถี่ Input มาก ทำให้การลู่เข้าของค่า $\alpha_1(k)$ ช้าแต่พอลู่เข้าแล้วสัญญาณความถี่ข้างเคียงที่ไม่ต้องการมีค่าน้อยมาก



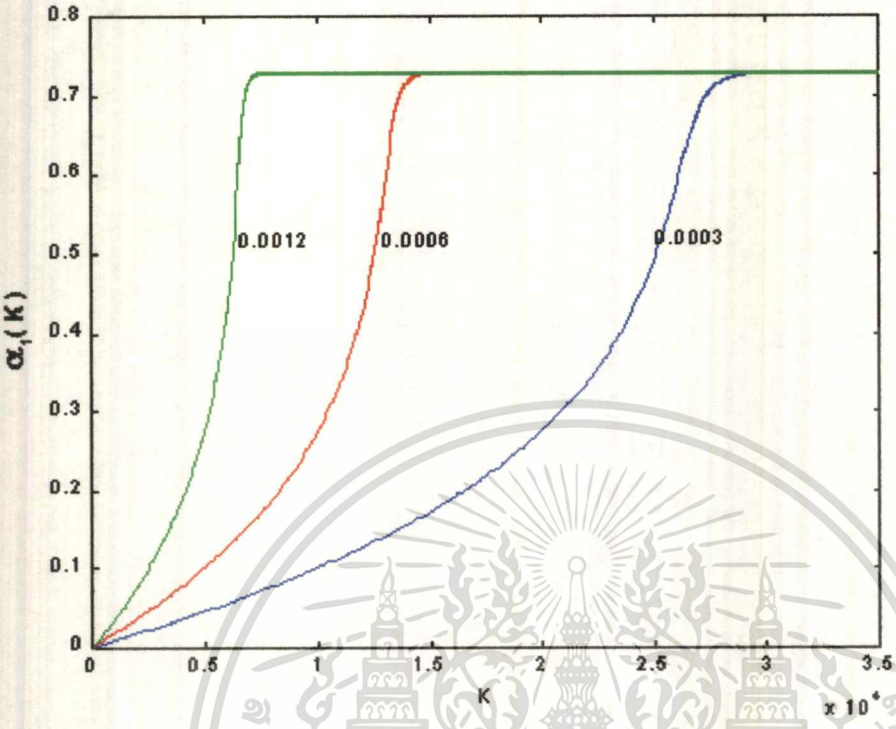
1) α_0 =0.4, 2) α_0 =0.6, 3) α_0 =0.7

รูปที่ 2.26 การเปลี่ยนค่าของ $\alpha_1(k)$ ในการเปลี่ยนค่า α_0 ค่าต่างๆ

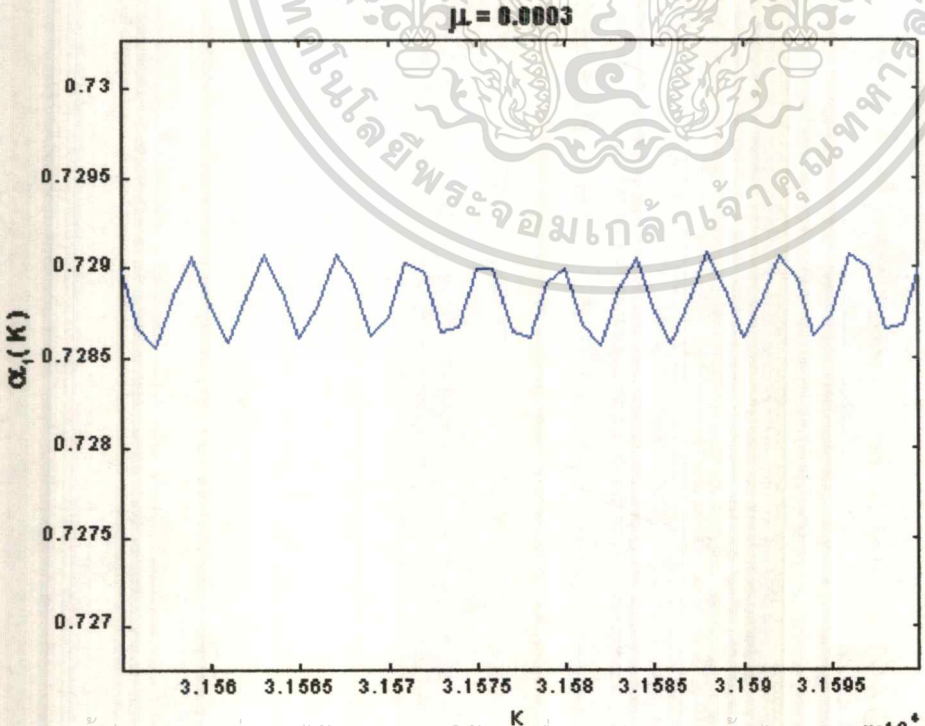
จากรูปเป็นการตอบสนองต่อการติดตามสัญญาณ Input ที่ค่าความถี่ 3.5KHz (ซึ่งค่า $\alpha_1(k)$ จะน้อยกว่า 0) และที่ สัญญาณ Input 1.5KHz (ซึ่ง α_1 จะมากกว่า 0) โดยจะติดตามความถี่ Input เดียวกันและจะมีการปรับค่า ต่างๆ กัน

การเปลี่ยนค่า Step Size Parameter (μ)

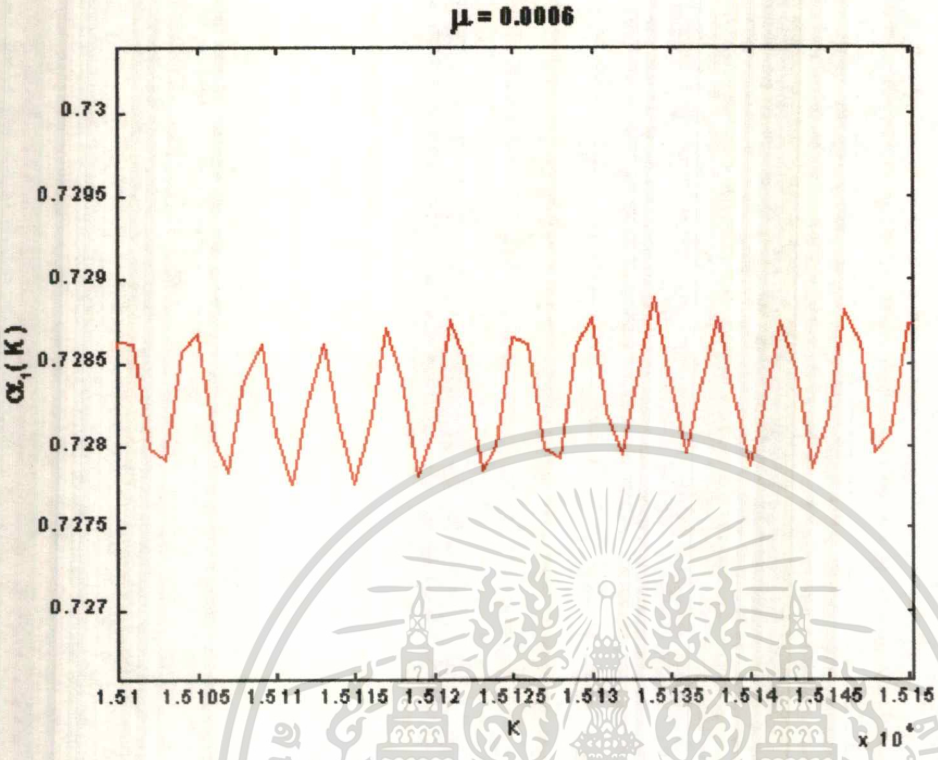
การลู่เข้าของค่า $\alpha_1(k)$ เมื่อมีการเพิ่มหรือลด Step Size Parameter จากการทดลองจะเห็นว่าเมื่อเพิ่มค่า จะทำให้การเปลี่ยนแปลงค่า $\alpha_1(k)$ เร็วขึ้นแต่ถ้าเพิ่มมากไปเมื่อระบบปรับ $\alpha_1(k)$ เข้าสู่ค่าที่ตรงแล้วจะเกิดการแกว่งของค่า $\alpha_1(k)$ มากและยิ่งมีการเพิ่มค่า μ มากขึ้นเท่าใดจะทำให้เกิดการแกว่งมากขึ้นเท่านั้นในการออกแบบจะต้องเลือกขนาดที่เหมาะสมกับช่วงของสัญญาณที่ต้องการใช้งาน



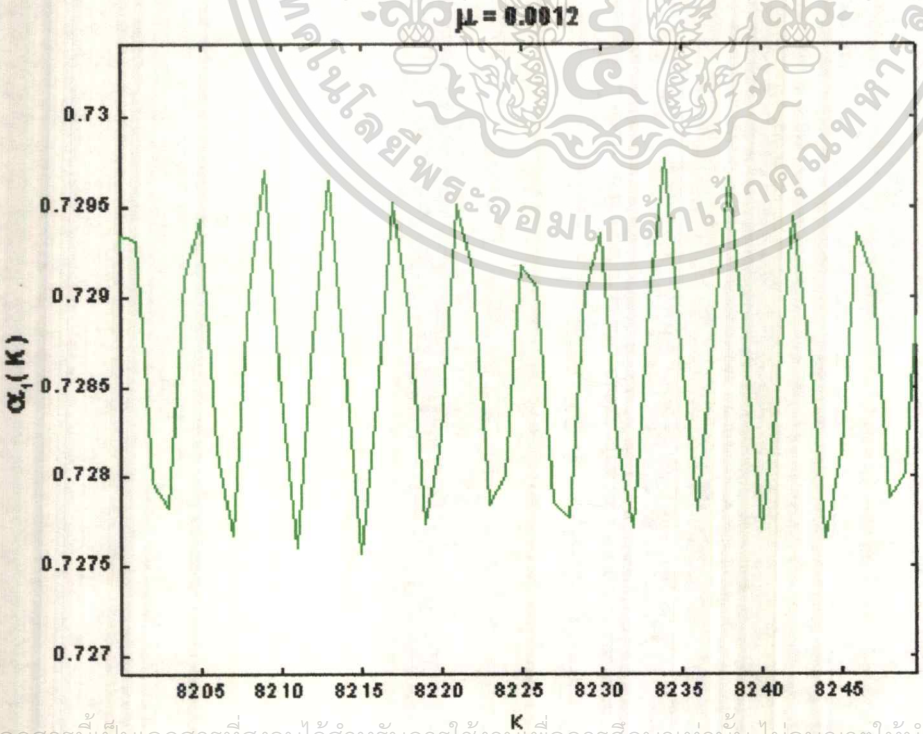
รูปที่ 2.27 แสดงการเกิดการเปลี่ยนค่า $\alpha_1(k)$ เมื่อมีการเปลี่ยนค่า μ



รูปที่ 2.28 แสดงการแกว่งที่ค่า μ = 0.0003



รูปที่ 2.29 แสดงการแกว่งที่ค่า μ = 0.0006



รูปที่ 2.30 แสดงการแกว่งที่ค่า μ = 0.0012

จากการทำการ Simulation โดยการเปลี่ยนค่า μ จะเห็นว่าจะทำให้การลู่เข้าสู่ค่าที่ถูกต้องเร็วขึ้นกว่าเดิมมาก โดยจะเห็นได้จากรูป แต่จะมีผลกระทบเมื่อค่า $\alpha_1(k)$ เข้าสู่ค่าที่ถูกต้องจะยังมีการแกว่งอยู่มาก ยิ่งค่า μ มีค่ามากจะทำให้การแกว่งของส่วนนี้มีมาก

2.5 สรุปผลการ Simulation

จากการศึกษาทฤษฎีและการทดลองทำการ Simulation ในเงื่อนไขต่างๆ ที่ผ่านมาในบทนี้ทั้งหมดทำให้สามารถมีความเข้าใจถึงทฤษฎีและการใช้งานการประมวลแบบดิจิตอลได้เป็นอย่างดีตั้งแต่การทำงานของ IIR Digital Filter เงื่อนไขของตัวแปรต่างๆ ที่มีผลต่อการตอบสนองลักษณะต่างๆ เช่นการเปลี่ยนความถี่กลาง การเปลี่ยนค่า Q-factor จะมีผลต่อการตอบสนองของระบบอย่างไร

การเปลี่ยนแปลงความถี่ของการ Sampling จะมีผลต่อระบบและการตอบสนองความถี่ของดิจิตอลฟิลเตอร์คือความถี่ของการ Sampling จะเป็นตัว Scaling ความถี่ของ Z Domain เป็นความถี่ใน Time Domain ตามสมการที่ (2.8) ซึ่งทำให้เรามีความเข้าใจในขั้นตอนที่จะทำการเขียนโปรแกรมในการประมวลผลสัญญาณจริงๆ

ในส่วนของ Adaptive IIR Filter หลังจากศึกษาทฤษฎีแล้วได้ทำการทดลองในรูปแบบต่างๆ ตั้งแต่การดูรูปคลื่นของสัญญาณ Output ที่เงื่อนไขต่างๆ ตั้งแต่ระบบเริ่มทำงานจนถึงขณะที่ระบบปรับค่า $\alpha_1(k)$ จนถึงค่าที่ถูกต้อง ตลอดจนการดูค่าการเปลี่ยนแปลงของค่า $\alpha_1(k)$ ในเงื่อนไขต่างๆ กัน เช่นการลดค่า α_0 (Q-factor) ทำให้การปรับค่า $\alpha_1(k)$ ไปสู่ค่าที่ถูกต้องจะทำได้เร็วขึ้น หรือการเพิ่มค่าของ μ (Step size parameter) จะทำให้การติดตามความถี่สัญญาณ Input ของระบบสามารถทำได้รวดเร็วขึ้น แต่จะมีการแกว่งของค่า $\alpha_1(k)$ ขณะที่อยู่ในค่าที่ถูกต้องมากขึ้น

ซึ่งทั้งหมดนี้จะทำให้เรามีความเข้าใจในการทำงานระบบการประมวลผลแบบดิจิตอลได้ดีขึ้น และสามารถที่จะทำนำความรู้ที่ได้ไปทำการเขียนโปรแกรมในการประมวลผลสัญญาณจริงได้เป็นอย่างดี อีกทั้งเป็นแนวทางในการที่จะวิเคราะห์ผลตอบสนองที่จะเกิดขึ้นในการทำการประมวลผลสัญญาณจริงๆ ได้ดีขึ้น

บทที่ 3

ทฤษฎี Chip DSP TMS320C5x

Chip DSP ในตระกูล Fixed-Point ของบริษัท Texas Instrument มีการพัฒนามาหลายรุ่นซึ่งทั้งหมดจะมีชุดคำสั่งซึ่งสามารถใช้ร่วมกันได้โดยคำสั่งของรุ่นเก่าจะยังคงใช้ได้กับ Chip ตัวใหม่ซึ่งสามารถทำให้ผู้ใช้ศึกษาเพิ่มเติม ไม่มากก็จะทำให้สามารถใช้ Chip รุ่นใหม่ๆได้ไม่ยาก และหนึ่งในนั้นคือ Chip ในตระกูล C5x จะมีหลายรุ่นซึ่งจะแตกต่างกันในส่วนของ Memory ที่บรรจุมาภายในเพื่อให้ผู้ใช้เลือกนำไปใช้งานในลักษณะต่างๆ กันแต่ในส่วนหลักๆ ของการทำงานภายในจะเหมือนกันโดยในโครงงานนี้จะนำ Chip รุ่น C50 มาใช้งานแต่ในส่วนของทฤษฎีในบทนี้จะเรียกรวมๆ ว่า C5x

3.1 การใช้งานทั่วไป

Automotive	Consumer	Control
Adaptive Ride Control Antiskid Brake Cellular Telephone Digital Radio Engine Control Global Positioning Navigation Vibration Analysis Voice Commands	Digital Radio/TV Educational Toys Music Synthesizer Power Tools Radar Detector Solid-State Answering Machines	Disk Drive Control Engine Control Laser Printer Control Motor Control Robotics Control Servo Control
General-Purpose	**Graphics/Imaging**	**Industrial**
Adaptive Filtering Convolution Correlation Digital Filtering Fast Fourier Transforms Hilbert Transforms Waveform Generation Windowing	3-D Rotation Animation/Digital Map Homomorphic Processing Pattern Recognition Image Enhancement Image Compression/ Transmission Robot Vision Workstations	Numeric Control Power-Line Monitoring Robotics Security Access
Instrumentation	**Medical**	**Military**
Digital Filtering Function Generation Pattern Matching Phase-Locked Loops Seismic Processing Spectrum Analysis Transient Analysis	Diagnostic Equipment Fetal Monitoring Hearing Aids Patient Monitoring Prosthetics Ultrasound Equipment	Image Processing Missile Guidance Navigation Radar Processing Radio Frequency Modems Secure Communications Sonar Processing
Telecommunications		**Voice/Speech**
1200- to 19200-bps Modems Adaptive Equalizer ADPCM Transcoder Cellular Telephone Channel Multiplexing Data Encryption Digital PBXs Digital Speech Interpolation (DSI)	DTMF Encoding/Decoding Echo Cancellation FAX Line Repeater Speaker Phone Spread Spectrum Communications Video Conferencing X.25 Packet Switching	Speech Enhancement Speech Recognition Speech Synthesis Speaker Verification Speech Vocoding Voice Mail Text-to-Speech

ตารางที่ 3.1 แสดงรายการของการใช้งาน C5x ในด้านต่างๆ

ด้วยลักษณะการทำงานแบบ Real Time และความสามารถในการใช้งานที่หลากหลายทำให้ Chip รุ่น C5x ใช้งานได้อย่างกว้างขวางในการแก้ปัญหาต่างๆ ที่มีการทำงานแบบ Signal-Processing เช่น การถอดรหัส หรือการกรองสัญญาณ ไม่เพียงเท่านั้นในรุ่น C5x ยังออกแบบมาเพื่อรองรับการทำงานแบบที่มีการทำงานหลายอย่างในเวลาเดียวกันด้วย

3.2 ลักษณะทั่วไป

ในรุ่น C5x ยังประกอบไปด้วย C50 , C51 และ C53 ซึ่งทั้งสามตัวเป็นสิ่งประดิษฐ์ประเภท Static CMOS ที่มีสถาปัตยกรรมภายในอยู่บนพื้นฐานของรุ่น C25 ด้วยการผสมผสานกันของสถาปัตยกรรมชั้นสูงที่มีการแยกระหว่าง Program-BUS และ Memory-BUS ,การเพิ่มอุปกรณ์สนับสนุนลงบน Chip ,การสร้าง On-Chip Memory ที่มีขนาดใหญ่ขึ้นและคำสั่งพิเศษที่มีความสามารถสูงขึ้น ซึ่งเป็นพื้นฐานให้การทำงานมีความยืดหยุ่นและความเร็วสูงขึ้น

TMS32051 ถูกออกแบบให้มีความเร็วในการทำงานมากกว่า 28 MIPS ในอนาคตคาดว่าจะมีการพัฒนาขึ้นอีก และสามารถตอบสนองความต้องการของตลาดได้มากขึ้น ในรุ่น C5x ออกแบบให้มีประโยชน์ใช้สอยได้ดังนี้

- ใช้หลักในการออกแบบที่มุ่งความเร็วสูงสุดและใช้งานได้ง่าย
- ใช้เทคโนโลยีชั้นสูงเพื่อเพิ่มความเร็วในการทำงาน
- มีความ Compatible กับรุ่นก่อนๆ ไม่ว่าจะเป็นรุ่น C1x , C2x ทำให้สามารถทำการ Upgrade ได้ง่าย
- มีการปรับปรุงชุดคำสั่งเพื่อความเร็วในการทำงานแบบ Algorithms ที่มีประสิทธิภาพมากขึ้นและเพิ่มความสามารถในการทำงานกับภาษาชั้นสูง
- ใช้เทคนิคแบบใหม่ในการออกแบบช่วยให้ประหยัดกำลังงานในการทำงานและทำให้การกระจายของคลื่นวิทยุไปสู่ภายนอกน้อยลง
- ใช้สถาปัตยกรรมชั้นสูงเพื่อเพิ่มความเร็วในการทำงานและสามารถทำงานได้หลายๆ อย่างในเวลาเดียวกัน

ตารางที่ 3.2 แสดงภาพรวมของความสามารถในรุ่น C5x โดยแสดงถึงความจุของ On-Chip RAM และ ROM จำนวนของ Serial และ Parallel I/O port , Execution Time ของแต่ละ Machine Cycle และชนิดของ Package รวมทั้งจำนวนขาทั้งหมดของ Chip

TMS320 Device	On-Chip Memory			I/O Ports		Cycle Time (ns)	Package Type QFP*
	RAM		ROM				
	Data	Data+Prog	Prog	Serial	Parallel†		
TMS320C50	1K	9K	2K	2	64K	50/35	132-pin ceramic
TMS320C51	1K	1K	8K	2	64K	50/35	132-pin plastic
TMS320C53	1K	3K	16K	2	64K	50/35	132-pin plastic

ตารางที่ 3.2 แสดงคุณลักษณะของ Chip ในรุ่น C5x

3.3 รายละเอียดเกี่ยวกับอุปกรณ์

รายละเอียดของรุ่น C5x แสดงไว้ข้างล่างนี้โดยแสดงลักษณะที่สำคัญของแต่ละเบอร์ , โดยชื่อของ Chip จะแสดงอยู่ในวงเล็บ

- ใช้เวลาทำงานตามคำสั่งแบบ Fixed-Point ด้วยความเร็ว 35-50 ns
- มีคำสั่งที่ Compatible กับรุ่น C1x , C2x
- RAM-Based Memory Operation (C50)
- ROM-Based Memory Operation (C51)
- 9K x 16 bit on-chip program/data RAM (C50)
- 1K x 16 bit on-chip program/data RAM (C51)
- 3K x 16 bit on-chip program/data RAM (C53)
- 2K x 16 bit on-chip boot ROM (C50)
- 8K x 16 bit on-chip program ROM (C51)
- 16K x 16 bit on-chip program ROM (C53)
- 1056 x 16 bit dual-access on-chip data RAM
- 32 bit ALU , 32 bit ACC และ 32 bit ACCB (Accumulator Buffer)
- 16 bit PLU (Parallel Logic Unit)
- 16 x 16 bit parallel multiplier พร้อมทั้ง 32 bit product capability
- มีความสามารถในการคูณด้วยรอบคำสั่งเดียว
- 8 Auxiliary Register
- 11 Context-switch register (shadow register) เพื่อใช้ในการ Interrupt
- 0-16 bit shifter และ 64 bit incremental
- 2 circular buffer สำหรับการอ้าง address แบบ circular มีคำสั่ง single-instruction repeat และ block repeat

- มีคำสั่ง move แบบ block เพื่อการจัดการข้อมูลที่ดีขึ้น
- Full-Duplex Synchronous serial port เพื่อการสื่อสารระหว่าง C5x กับอุปกรณ์ภายนอก
- มี TMD (Time Division Multiple Access) Serial port
- 64K parallel I/O port และมี 16 port ที่สามารถ mapped ไปยัง memory ได้
- 16 wait state generator ที่ควบคุมการทำงานได้ด้วย software
- เชื่อมต่อกับอุปกรณ์ภายนอกแบบ DMA ได้สะดวก
- ภายในประกอบด้วย 4-deep pipeline สำหรับ delayed branch
- index-addressing mode
- Bit-Reverse index-addressing mode สำหรับ radix-2 FFT
- มีความสามารถหารได้ภายใน clock ลูกเดียว
- มี clock generator ภายใน chip
- JTAG boundary scan logic (IEEE standard, 1149.1)
- 5V. static CMOS พร้อมด้วย power-down mode 2 mode

3.3.1 ส่วนสำคัญของ CPU

การพัฒนาขึ้นมาเป็นรุ่น C5x ยังใช้ Source Code ที่ Compatible กับรุ่น C1x และ C2x เมื่อมีการพัฒนา Program จึงทำได้ง่าย ในการพัฒนารุ่นต่างๆ ของ C5x นี้ยังมีการสร้าง 32 bit accumulator buffer (ACCB) เพิ่มเติมขึ้นมาด้วย เพื่อสนับสนุนการทำงานของ ACC

ใน control function แบบใหม่มี PLU ซึ่งเป็นอิสระในการทำงานดังนั้นจึงสามารถทำงานในแบบสมการ boolean ได้ดีขึ้น และมีการบริการการ interrupt ด้วย IRS จึงทำงานได้เร็วขึ้น การจัดการข้อมูลได้มีการปรับปรุงโดยมีการใช้คำสั่ง block move และมีคำสั่ง memory-mapped ในรุ่น C5x มี register สำหรับทำการ mapped memory ทั้งหมด 28 ตัว และมี register สำหรับทำการ mapped I/O ทั้งหมด 16 ตัว

3.3.2 On-Chip ROM

C50 มี 2K x 16 bit maskable programmable ROM หน่วยความจำนี้ใช้สำหรับการ Boot เครื่องในตอนเริ่มต้น ในกรณีที่ ROM ภายนอกมีความเร็วต่ำ โดยการเลือก ROM สามารถเลือกได้ในขณะ Reset ด้วยขา MP/$\overline{MC}$ ในครั้งแรกที่ทำการ Boot เครื่อง โดย Boot ROM สามารถที่จะทำการยกเลิกการใช้งานได้ด้วยการควบคุม bit ใน PMST Register

3.3.3 On-Chip data RAM

อุปกรณ์ทั้งหมดในรุ่น C5x จะมี RAM ขนาด 1056 x 16 bit โดย RAM ในส่วนนี้จะมีความสามารถในการทำงานได้ถึง 2 ครั้งใน 1 machine cycle (dual-access RAM) โดยใช้สำหรับเก็บข้อมูลโดยเฉพาะแต่ก็สามารถเก็บ program ในพื้นที่ส่วนนี้ได้ถ้าต้องการ

3.3.4 On-Chip Program/Data RAM

C50 มี 9K x 16 bit On-Chip RAM ซึ่งพื้นที่หน่วยความจำส่วนนี้สามารถจัดโครงสร้างด้วย software ให้เป็นพื้นที่ program หรือ data หรือเป็นทั้งพื้นที่ program และ data ก็ได้

3.3.5 On-Chip Memory Security

C5x มีระบบป้องกันข้อมูลที่บรรจุอยู่ภายใน On-Chip Memory เมื่อมีการ Set ค่าของ Register ต่างๆ ที่เกี่ยวกับการแบ่งหน่วยความจำแล้วจะไม่มีคำสั่งจากภายนอกเข้าไปใช้งานในพื้นที่หน่วยความจำได้

3.3.6 Address-Mapped Software Wait-State Generator

การกำหนด Wait-State เมื่อติดต่อกับหน่วยความจำหรืออุปกรณ์ I/O สามารถกระทำได้โดยการ Set ค่า Register ที่เกี่ยวข้องด้วย Software โดยไม่จำเป็นที่จะต้องต่อ Hardware เพิ่มเติม ในส่วนนี้ประกอบไปด้วยวงจร Wait-State Generators จำนวน 16 ชุดด้วยกัน เพื่อทำการสร้าง Wait-State 0,1,2,3,หรือ 7 Wait-State ในการติดต่อกับหน่วยความจำภายนอกและอุปกรณ์ I/O จะสามารถกำหนดขอบเขต เพื่อที่จะทำการ Map กับ Wait-State ได้ด้วยขนาด 16K-word ต่อ 1 วงจร

3.3.7 Parallel I/O Ports

Chip ในรุ่น C5x จะมี I/O port ทั้งหมด 64K-port โดยที่ 16 port ใดๆใน 64K-port สามารถ Map ลงไปยัง Data Memory ได้ port เหล่านี้สามารถกำหนด Address ได้ด้วยคำสั่ง IN และ OUT และ I/O port ที่ถูก Map ไปยัง Data Memory สามารถทำงานได้ด้วยคำสั่งอ่านและเขียนหน่วยความจำแบบปกติได้

3.3.8 I/O Ports

Chip ในรุ่น C5x ประกอบไปด้วย Serial port ความเร็วสูงจำนวน 2 port ซึ่งสามารถทำงานได้ด้วยความเร็วถึง ¼ ของ Machine Cycle (CLKOUT1) โดยที่ Serial port ตัวแรกทำงานแบบ Synchronous Full-Duplex และมี Buffer ภายในสำหรับการรับและการส่งข้อมูล ส่วน Serial port อีกตัวหนึ่งเป็นแบบ Full-Duplex ที่สามารถกำหนดการทำงานให้เป็นแบบ Synchronous หรือ TDM (Time Division Multiple Access) ก็ได้ ซึ่ง TDM จะใช้งานทั่วไปในระบบ Multi-Processor

Registers	Description
SPC	Serial port control register
DXR	Data transmit register
DRR	Data receive register
XSR	Transmit shift register
RSR	Receive shift register

ตารางที่ 3.3 Serial Port Register

3.3.9 Hardware Timer

Chip ในรุ่น C5x จะมีวงจร Timing ขนาด 16 bit พร้อมด้วย 4 bit Pre-scalar ความถี่ Clock จะมีอัตราตั้งแต่ 1/2 ถึง 1/32 ของ Machine Cycle ขึ้นอยู่กับการ Program Divide-Down Ratio วงจร Timer สามารถควบคุมให้หยุด , เริ่มต้นใหม่ , Reset หรือ Disable ด้วย Status bit พิเศษที่เป็นของวงจร Timer

3.3.10 User-Maskable Interrupts

Chip ในรุ่น C5x มีสาย External-Interrupt จำนวน 4 เส้นดังนั้นอุปกรณ์ภายนอกแต่ละตัวสามารถควบคุม TMS320 ได้ และอุปกรณ์ภายในจะสามารถควบคุมการ Interrupt ได้ 5 ช่องจาก Timer 1 ช่องและ Serial อีก 4 ช่องด้วย

3.3.11 JTAG Scanning Logic

Chip ในรุ่น C5x มีความสามารถในการตรวจสอบการทำงานอย่างต่อเนื่องเกี่ยวกับการทำงานของอุปกรณ์ภายนอกที่เชื่อมต่ออยู่และการทำงานภายใน CPU เองได้อย่างดี โดยที่ JTAG จะเชื่อมต่ออยู่กับวงจร Scanning Logic อื่นๆ ภายใน CPU Chip ในตระกูล C5x จึงสามารถทำงานเป็นวงจรตรวจสอบโดยอ่านค่าจากขาสัญญาณ JTAG Serial Scan และขาสัญญาณ Emulation-Dedicated

3.3.12 Packages

Chip ในรุ่น C5x มีลักษณะ Package เป็น QFP (Quad Flat Pack Package) 132 pin โดยมีจุดประสงค์ให้กินพื้นที่ในบอร์ดให้น้อยที่สุด โดยใช้พื้นฐานเดียวกันกับการออกแบบ Chip C25

3.4. สถาปัตยกรรมภายใน

โครงสร้างสถาปัตยกรรมภายในของ TMS320 DSP ประกอบไปด้วยส่วนพื้นฐาน 3 ส่วนด้วยกัน คือ

-Central Processing Unit (CPU)

-Memory

-Peripheral-Interfacing Circuit

ซึ่งในส่วนนี้กล่าวถึง สถาปัตยกรรมและการทำงานหลักของ CPU C51 ที่มีความสามารถในการทำงานทางคณิตศาสตร์ด้วยความเร็วสูง ด้วยสถาปัตยกรรมการทำงานแบบขนานของมัน

3.4.1 โครงสร้างสถาปัตยกรรมโดยรวม

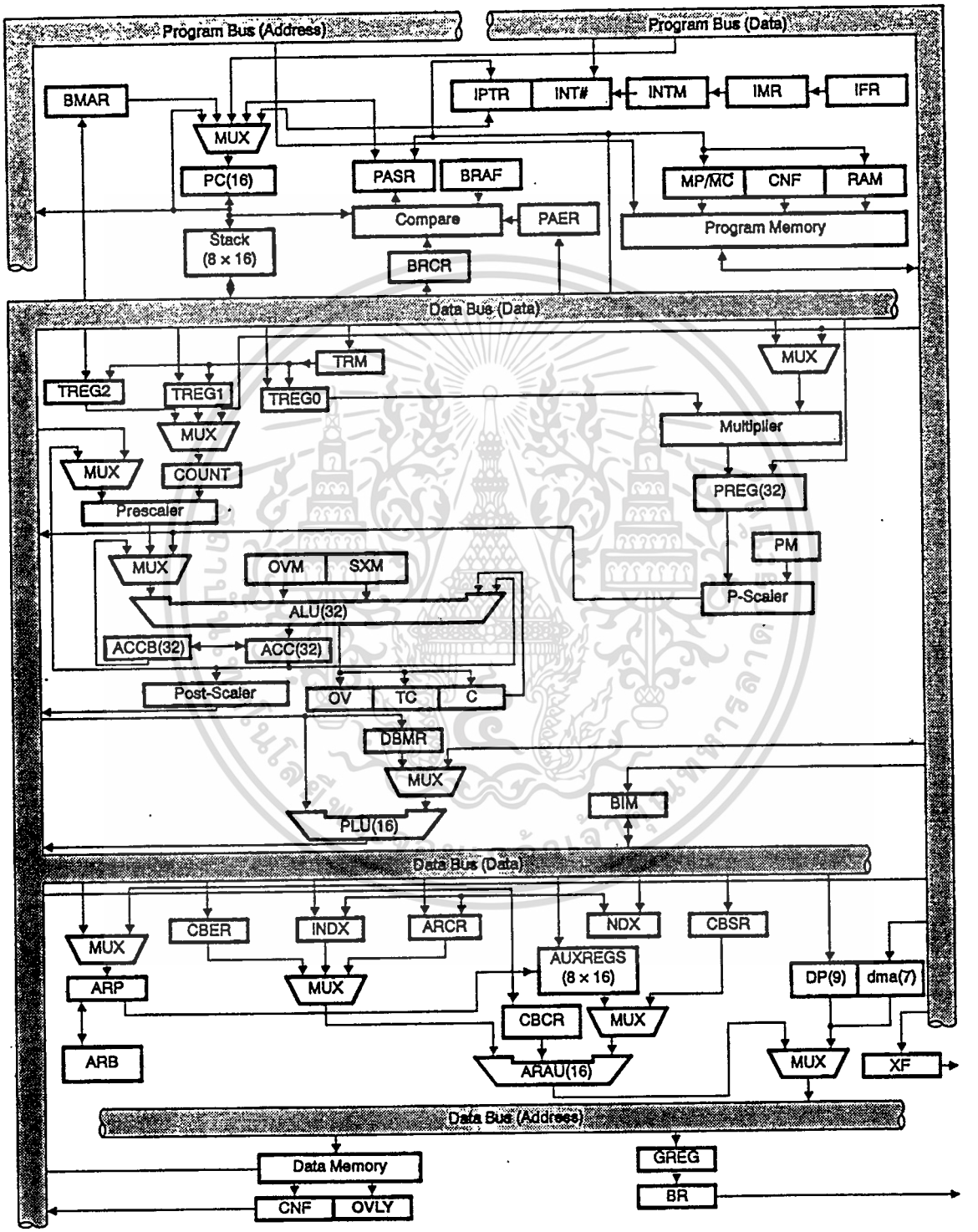
Chip ในรุ่น C5x เป็น Chip ที่มีความสามารถสูงในการทำงานด้าน Digital Signal Processing โดยออกแบบให้มีความคล้ายคลึงกับ Chip ในรุ่น C25 ใช้สถาปัตยกรรมการทำงานแบบแยก Memory Bus ระหว่าง Program Bus และ Data Bus ให้มีการทำงานแยกกันต่างหาก เพื่อความเร็วสูงสุดในการทำงาน โดยมีคำสั่ง Transfer รองรับการทำงานทั้งสองส่วน

C5x สามารถใช้งานในการทำ 2-Complement โดยการใช้ 32 bit ALU และ Accumulator ALU มีลักษณะเป็น General Purpose ที่ใช้ 16 bit word จาก Data memory หรืออ้างแบบ Immediate จากคำสั่งหรือเก็บผลลัพธ์จากการคูณในขนาด 32 bit ด้วยการเพิ่มเติมการทำงานของ ALU ทำให้ CPU สามารถทำคำสั่ง Boolean ได้ด้วย

Accumulator จะมีหน้าที่เป็น Output ของ ALU และในขณะเดียวกันมันก็เป็นทั้ง Input ของ ALU ด้วย Accumulator มีขนาด 32 bit โดยแบ่งเป็น 16 bit บน (bit 31 ถึง bit 16) และ 16 bit ล่าง (bit 15 ถึง bit 0) ในชุดคำสั่งของ C5x ก็ยังมีที่เก็บข้อมูลชั่วคราวสำหรับ Accumulator ด้วยซึ่งมีขนาด 32 bit เท่ากัน มีชื่อเรียกว่า ACCB ในการเพิ่มความสามารถของ ALU มีการเพิ่มหน่วย PLU ขึ้นมาซึ่งสามารถทำคำสั่งเกี่ยวกับ Logic ได้ โดยการทำงานเกี่ยวกับ bit จะไม่ส่งผลกระทบไปยัง Accumulator ALU สามารถใช้ในการโอนย้าย bit ทำให้สามารถควบคุม bit และแก้ไข bit ได้อย่างรวดเร็ว เพื่อใช้ในการ set bit , clear bit และ test bit สำหรับ status register ต่างๆ

การคูณเลข 16 bit 2 จำนวนแบบ 2-Complement โดยให้ผลลัพธ์ออกมาเป็นเลข 32 bit สามารถกระทำได้ภายใน 1 รอบคำสั่ง ส่วนของการคูณประกอบไปด้วย 3 ส่วนสำคัญคือ Multiplier Array , PREG (Product Register) และ TREG0 (Temporary Register) โดยที่ TREG0 มีขนาด 16 bit

และ PREG มีขนาด 32 bit ในการคูณนั้นค่าของตัวคูณอาจจะนำมาจาก Data Memory , Program Memory เมื่อใช้คำสั่ง MAC/MACD/MADS/MADD หรืออาจเป็นค่า Immediate ภายในคำสั่ง MPY # ก็ได้ ด้วยการคูณที่มีความเร็วสูงนี้ ทำให้ DSP มีการทำงานที่มีประสิทธิภาพสูงในการทำงาน เช่นการ Filter หรือ Correlation



รูปที่ 3.1 Block Diagram แสดงโครงสร้างภายในทั้งหมดของ C5x

ส่วนของ Scaling Shifter ภายใน C5x มีลักษณะที่เชื่อมต่อระหว่าง Data Bus และ ALU โดยมี Input ขนาด 16 bit ทางด้าน Data Bus และ Output ขนาด 32 bit ทางด้าน ALU Scaling Shifter มีความสามารถในการเลื่อน bit ไปทางซ้ายได้ตั้งแต่ 0-16 bit จาก Input Data ที่เข้ามา ตามค่าที่กำหนดภายใน TREG1 โดยค่าทางขวาหลังจากทำการเลื่อน bit แล้วจะถูกแทนที่ด้วยค่าศูนย์ ส่วนค่า MSB สามารถให้เป็น 0 หรือ Sign ก็ได้ ขึ้นอยู่กับสถานะของ Sign-Extension Mode bit (SXM) ของ Status Register ST1 การเพิ่มความสามารถในการ Shift นี้ทำให้ Processor มีความสามารถเพิ่มขึ้นในด้าน Numerical-Scaling , Bit-Extraction , Extended-Arithmetic และ Overflow-Prevention

C5x มี Hardware Stack 8 ระดับ เพื่อป้องกันข้อมูลขณะ Interrupt และขณะทำคำสั่ง Call โดยขณะทำการ Interrupt Register หลักอันได้แก่ ACC , ACCB , ARCR , INDX , PMSTPREG , ST0 และ ST1 TREGS จะถูก Push ลงบน Stack ระดับแรกสุด และจะถูก Pop ขึ้นมาเมื่อมีการ Interrupt Return หรือกลับจากการ Call

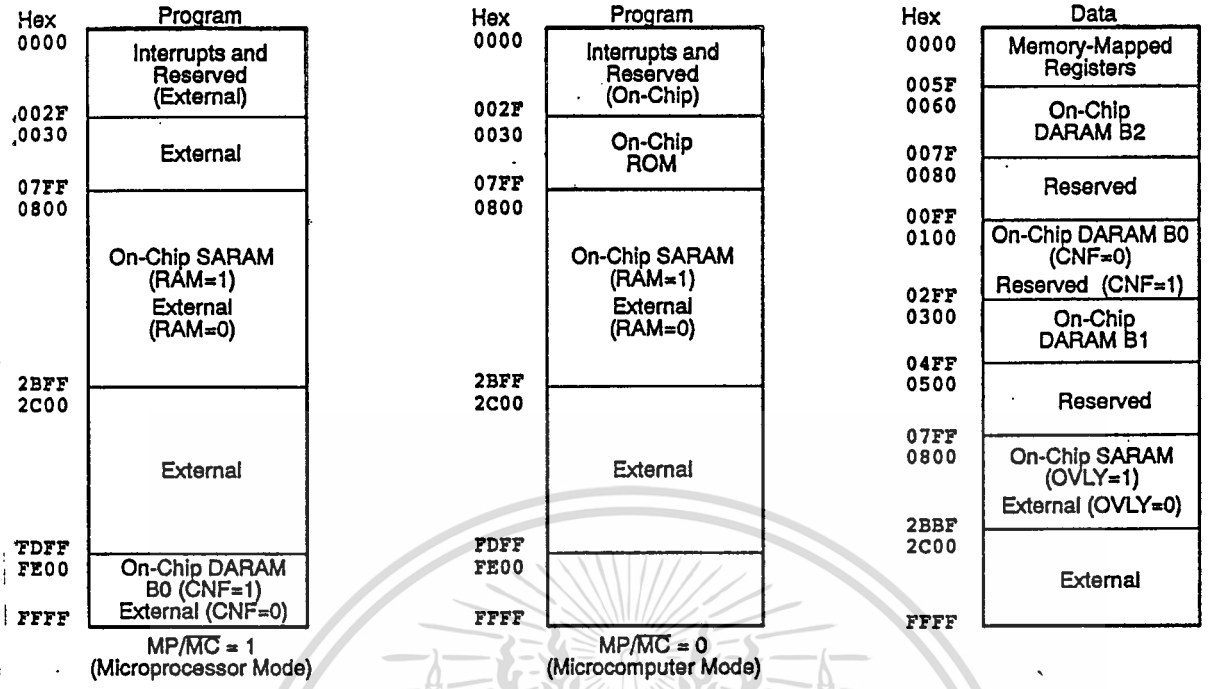
3.5 Memory

หน่วยความจำทั้งหมดของ Chip ในรุ่น C5x มีขนาดสูงสุด 224K 16-bit word พื้นที่ในหน่วยความจำถูกแบ่งเป็น 4 ส่วนดังนี้คือ พื้นที่ Program ขนาด 64K-words , Local Data ขนาด 64k-words , Global Data ขนาด 32k-words และ I/O Port อีก 64K ในสถาปัตยกรรมแบบขนานจะทำให้อุปกรณ์แยกการทำงานได้ 3ส่วน ได้แก่ Fetch , Read และ Write ซึ่งโครงสร้างและการทำงานอธิบายได้ดังนี้

3.5.1 Memory Space Chip

ใน C5x มีสถาปัตยกรรมแบบหลาย Memory Space ซึ่งสามารถทำงานใน Bus แบบขนาน ทั้งหมด 3 Bus สิ่งนี้เองที่ทำให้ CPU สามารถทำงานเกี่ยวกับ Program และ Data ได้ในเวลาเดียวกัน Parallel Bus ทั้งสามได้แก่ PAB (Program Read/Write Bus) , DAB1 (Data Read Bus) และ DAB2 (Data Write Bus) แต่ละ Bus ทำงานในพื้นที่หน่วยความจำที่แตกต่างกัน หน่วยความจำใน C5x แบ่งออกเป็นสี่ส่วนที่แยกเป็นอิสระจากกัน คือส่วน Program , Local Data , Global data และ input/output Port ซึ่งสามารถ Mapped ให้อยู่ภายใน Chip หรืออยู่ข้างนอก Chip ก็ได้

Program Space จะเก็บค่าของคำสั่งที่จะทำการ Executed , Local Data Space เก็บค่าของ Data ที่ถูกใช้โดยคำสั่ง , Global Data Space มีความสามารถในการ Share ข้อมูลระหว่าง Processor หรือสามารถจองไว้ใช้เพื่อเป็น Data Space ก็ได้ ส่วน I/O Space จะเชื่อมต่อไปยังอุปกรณ์ภายนอกในรูปแบบของ Memory Map ภายนอกและยังสามารถจองพื้นที่นี้ไว้ใช้เป็นพื้นที่เก็บ Data แบบ Extra ได้ด้วย



รูปที่ 3.2 การใช้พื้นที่หน่วยความจำของ C50

3.5.2 Program memory

Chip ในรุ่น C5x สามารถต่อหน่วยความจำเพิ่มเติมได้ถึง 64K 16 bit word เมื่อต่อหน่วยความจำดังกล่าวเพิ่มขึ้นแล้ว จะทำให้ C5x มีหน่วยความจำทั้งหมดคือ On-Chip ROM , Single-Access Program/Data RAM และ Dual-Access RAM ด้วยการใช้ Software จะทำให้สามารถจัดโครงสร้างของหน่วยความจำแต่ละส่วนให้อยู่ภายในหรือภายนอก Address Map ก็ได้เมื่อ Mapไปยัง Program Space Chip จะประมวลผลข้อมูลจำกัดอยู่ภายในขอบเขตของมันโดยอัตโนมัติเมื่อ CPU สร้าง Address นอกเหนือไปจากขอบเขต Chip จะเปลี่ยนการทำงานไปประมวลผลข้อมูลจากอุปกรณ์ภายนอกโดยอัตโนมัติ

CNF	RAM	MP/MC	ROM	SARAM	DARAM B0	Off-Chip
0	0	0	0000–07FF			0800–FFFF
0	0	1				0000–FFFF
0	1	0	0000–07FF	0800–2BFF		2C00–FFFF
0	1	1		0800–2BFF		0000–07FF
						2C00–FFFF
1	0	0	0000–07FF		FE00–FFFF	0800–FDFF
1	0	1			FE00–FFFF	0000–FDFF
1	1	0	0000–07FF	0800–2BFF	FE00–FFFF	2C00–FDFF
1	1	1		0800–2BFF	FE00–FFFF	0000–07FF
						2C00–FDFF

ตารางที่ 3.4 Program Memory Configuration Control

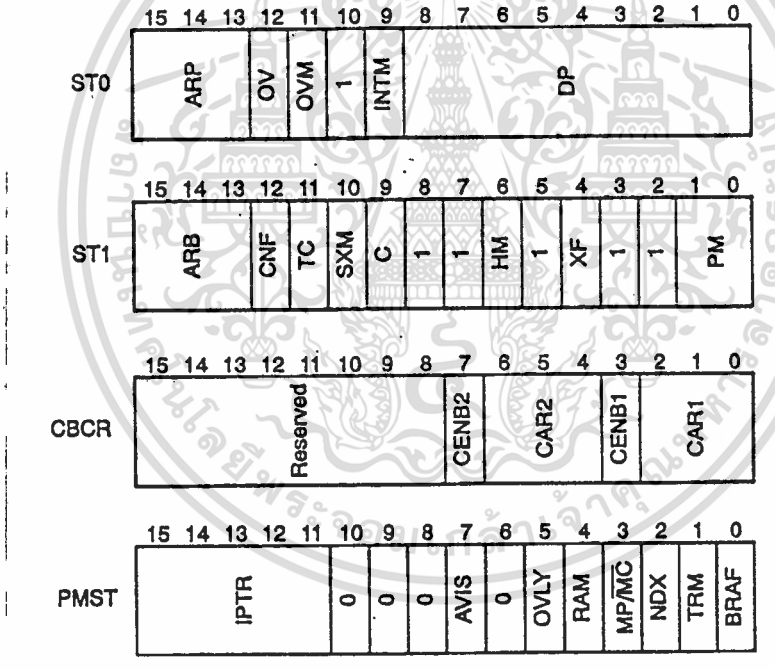
3.5.2.1 การจัดโครงสร้างของ Program space

หลังจากที่ทำการ Reset การจัดโครงสร้างของข้อมูล จะถูกกำหนดด้วยขา MC/MP ถ้าแรงดันที่ขานี้มีค่า High C51 จะจัดโครงสร้างข้อมูลเป็น Microprocessor และ on-chip ROM จะไม่มีการนำมาใช้งานถ้าขานี้มีแรงดันเป็น Low C51 จะจัดโครงสร้างข้อมูลเป็น Microcomputer คือ On-Chip ROM จะถูกนำมาใช้งาน ดังนั้นถ้าต้องการนำ C51 ไปใช้งานเป็น Microcomputer มันจะเริ่มทำงานด้วยคำสั่งบน On-Chip ROM ไม่เช่นนั้นมันจะเริ่มการทำงานที่หน่วยความจำภายนอกในขณะที่ Program เริ่มทำงาน เราสามารถทำการเปลี่ยนแปลงค่าของ MP/MC ได้ใน PMST Register โดยการทำตามคำสั่งดังเช่นแสดงไว้ในบรรทัดข้างล่างนี้

```
OPL # 8,PMST                ;remove boot ROM from program space
```

3.5.2.2 Program Memory Address Mapped

ค่าของ Reset , Interrupt และ Trap Vector จะถูกเก็บไว้ในพื้นที่ Program ซึ่งพื้นที่ในส่วนนี้จะอยู่ในตอนต้นของของเนื้อที่ในหน่วยความจำ ในขณะที่ Reset ข้อมูลใน Vector Mapped จะมีค่าชี้ไปยัง Address 0h



ตารางที่ 3.5 Status and Control Register Organization

3.5.3 Local Data Memory

ขนาดของ Local Data Memory สามารถขยายได้ถึง 64K 16 bit-word โดยใน Chip C51 จะสร้างเนื้อที่ขนาด 1K ไว้บน Chip เป็นแบบ SARAM Chip ทั้งหมดในรุ่น C5x มีขนาดของหน่วยความจำ Dual-Access Space RAM (DARAM) ขนาด 1056 words โดยสามารถนำมาจัดโครงสร้างใหม่ด้วย Software ให้อยู่ภายในหรือภายนอก Local Data Address Map ก็ได้ เมื่อหน่วยความจำส่วนนี้ถูก Map

ลงใน Data Space Chip จะทำการประมวลผลข้อมูลโดยอยู่ภายในขอบเขตที่กำหนด ถ้ามีการกำหนดพื้นที่ใช้งานเกินขอบเขต Processor จะเปลี่ยนไปทำงานในพื้นที่ Address ภายนอกโดยอัตโนมัติ

CNF	OVLY	DARAM B0	DARAM B1	DARAM B2	SARAM	Off-Chip
0	0	100h–2FFh	300h–4FFh	60h–7Fh		800h–FFFFh
0	1	100h–2FFh	300h–4FFh	60h–7Fh	800h–BFFh	C00h–FFFFh
1	0	–	300h–4FFh	60h–7Fh		800h–FFFFh
1	1	–	300h–4FFh	60h–7Fh	800h–BFFh	C00h–FFFFh

ตารางที่ 3.6 Local Data Memory Configure Control

3.5.4 Global Memory

สำหรับการทำงานแบบ Multi-Processing Chip ในรุ่น C5x มีความสามารถในการกำหนด Memory Space เพื่อใช้เป็นทางผ่านสำหรับสื่อสารข้อมูลระหว่าง Processor โดยอาศัยขาสัญญาณ BR และ Ready Global Memory เป็น Memory ที่จัดให้มีการใช้งานร่วมกันระหว่าง Processor มากกว่าหนึ่งตัว เพื่อที่ว่าการใช้งานร่วมกันจะทำได้โดยสะดวก Global Memory Address Space ถูกแบ่งออกเป็นส่วนๆ คือส่วน Local และ Global ในส่วน Local จะถูกใช้งานในหน้าที่ที่ต้องแยกอิสระจากกัน และในส่วน Global จะใช้เพื่อติดต่อสื่อสารกันกับ Processor ตัวอื่นๆ

3.5.5 Input/Output Space

C5x มีพื้นที่รองรับอุปกรณ์ I/O ขนาด 64K 16 bit เช่น Digital-to-Analog Converter (D/A)

Memory-Mapped Core Processor Registers			
Name	Address		Description
	Dec	Hex	
——	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
BRCR	9	9	Block Repeat Counter Register
PASR	10	A	Block Repeat Program Address Start Register
PAER	11	B	Block Repeat Program Address End Register
TREG0	12	C	Temporary Register Used for Multiplicand
TREG1	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

ตารางที่ 3.7 Memory-Mapped Register and I/O Ports

Memory-Mapped Core Processor Registers (Concluded)			
Name	Address		Description
	Dec	Hex	
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
CBER1	27	1B	Circular Buffer 1 End Register
CBSR2	28	1C	Circular Buffer 2 Start Register
CBER2	29	1D	Circular Buffer 2 End Register
CBCR	30	1E	Circular Buffer Control Register
BMAR	31	1F	Block Move Address Register
Memory-Mapped Peripheral Registers			
DRR	32	20	Data Receive Register
DXR	33	21	Data Transmit Register
SPC	34	22	Serial Port Control Register
—	35	23	Reserved
TIM	36	24	Timer Register
PRD	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 S/W Wait-State Register
CWSR	42	2A	S/W Wait-State Control Register
—	43–47	2B–2F	Reserved
TRCV	48	30	TDM Data Receive Register
TDXR	49	31	TDM Transmit Data Register
TSPC	50	32	TDM Serial Port Control Register
TCSR	51	33	TDM Channel Select Register
TRTA	52	34	TDM Receive/Transmit Address Register
TRAD	53	35	TDM Received Address Register

ตารางที่ 3.7 Memory-Mapped Register and I/O Ports (ต่อจากส่วนที่แล้ว)

Name	Address		Description
	Dec	Hex	
—	54–79	36–4F	Reserved
Memory-Mapped I/O Ports†			
PA0	80	50	I/O Port 50h
PA1	81	51	I/O Port 51h
PA2	82	52	I/O Port 52h
PA3	83	53	I/O Port 53h
PA4	84	54	I/O Port 54h
PA5	85	55	I/O Port 55h
PA6	86	56	I/O Port 56h
PA7	87	57	I/O Port 57h
PA8	88	58	I/O Port 58h
PA9	89	59	I/O Port 59h
PA10	90	5A	I/O Port 5Ah
PA11	91	5B	I/O Port 5Bh
PA12	92	5C	I/O Port 5Ch
PA13	93	5D	I/O Port 5Dh
PA14	94	5E	I/O Port 5Eh
PA15	95	5F	I/O Port 5Fh

† See Section 6.2 for memory-mapped I/O ports.

ตารางที่ 3.7 Memory-Mapped Register and I/O Ports (ส่วนจบ)

3.5.5.1 Addressing Space Input/Output Port

การทำงานติดต่อกับ Parallel I/O Port จะติดต่อแบบ Multiplex ด้วย Address Bus และ Data Bus เดียวกันกับการติดต่อกับ Program/Data Memory แต่พื้นที่ใช้งานของ I/O จะถูกแยกออกจาก Program/Data Memory ด้วยสัญญาณ IS Port ทั้งหมดจำนวน 65536 Ports สามารถทำงานได้ด้วยคำสั่ง IN และ OUT ดังตัวอย่างข้างล่างนี้

```
IN    DAT7,FFFEh
OUT   DAT7,FFFFh
```

16 Address จาก 64K I/O Port จะถูก Mapped ลงบนพื้นที่ของ Data Memory ทำให้ I/O Port สามารถเขียนหรืออ่านได้ด้วยคำสั่ง IN และ OUT เช่นเดียวกันกับการประมวลผลคำสั่งอื่นที่อยู่ภายใน Data Space ดังตัวอย่างต่อไปนี้ ซึ่งอธิบายการอ้างตำแหน่งแบบ Direct Address เพื่อทำงานเกี่ยวกับอุปกรณ์ I/O ในตำแหน่ง 51h

```
SACL  51h
```

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
BRCR	9	9	Block Repeat Counter Register
PASR	10	A	Block Repeat Program Address Start Register
PAER	11	B	Block Repeat Program Address End Register
TREG0	12	C	Temporary Register Used for Multiplicand
TREG1	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
CBER1	27	1B	Circular Buffer 1 End Register
CBSR2	28	1C	Circular Buffer 2 Start Register
CBER2	29	1D	Circular Buffer 2 End Register
CBCR	30	1E	Circular Buffer Control Register
BMAR	31	1F	Block Move Address Register
Peripheral Memory-Mapped Registers			
DRR	32	20	Data Receive Register
DXR	33	21	Data Transmit Register
SPC	34	22	Serial Port Control Register
——	35	23	Reserved

ตารางที่ 3.8 Data Page 0 Address Map

Name	Address		Description
	Dec	Hex	
Peripheral Memory-Mapped Registers (Continued)			
TIM	36	24	Timer Register
PRD	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
TDXR	49	31	TDM Data Transmit Register
TSPC	50	32	TDM Serial Port Control Register
TCSR	51	33	TDM Channel Select Register
TRTA	52	34	Receive/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
PA7	87	57	I/O Port 87
PA8	88	58	I/O Port 88
PA9	89	59	I/O Port 89
PA10	90	5A	I/O Port 90
PA11	91	5B	I/O Port 91
PA12	92	5C	I/O Port 92
PA13	93	5D	I/O Port 93
PA14	94	5E	I/O Port 94
PA15	95	5F	I/O Port 95
B2	96–127	60–7F	Scratch Pad RAM

ตารางที่ 3.8 Data Page 0 Address Map (ต่อจากส่วนที่แล้ว)

3.6. Software Application

C5x ยึดถือรูปแบบ Source Code ให้ Compatible กับ C1x และ C2x และมีโครงสร้างที่พัฒนาขึ้นเพื่อปรับปรุงประสิทธิภาพในการทำงานและการใช้งานได้กว้างมากขึ้น ชุดคำสั่งใน C5x จะมีเพิ่มขึ้นเพื่อรองรับ Hardware รวมทั้งเพิ่มการทำ Data Movement และ Memory Mapped Register รวมไปถึง

รองรับ Parallel Logic Unit (PLU) เพื่อการทำงานเกี่ยวกับคณิตศาสตร์ Boolean , 32 bit Accumulator Buffer และ Register ที่ใช้สำหรับบริการการ Interrupt เนื่องจาก On-Chip DARAM

3.6.1 Processor Initialization

ก่อนทำการประมวลผล Algorithm มีความจำเป็นที่จะต้องกำหนดค่าเริ่มต้นต่างๆ ให้แก่ Processor โดยทั่วไปการ Initialization จะกระทำทุกครั้งหลังจากที่มีการ Reset โดยหลังทำการ Reset แล้ว IPTR bit ของ PMST จะถูก Clear ดังนั้น Vector จากการ Mapping จะชี้ไปที่ Page 0 ของพื้นที่ Program Memory ซึ่งพื้นที่ดังกล่าวโดยปกติแล้วบรรจุคำสั่ง Branch เพื่อชี้ไปยังตำแหน่งของ Initialization Routine ส่วนใน Hardware จะมีการ Disable Interrupt โดยอัตโนมัติ โครงสร้างของ Processor หลังจากทำการ Reset ที่จะต้องทำการ Initialization ใหม่ได้แก่ส่วนต่างๆ ดังต่อไปนี้

Memory-Mapped และ Peripheral Control Register

Interrupt Structure (INTM)

Mode Control (OVM,SXM,PM,AVIS,NDX,TRM)

Memory control (RAM,OVLY,CNF)

Auxiliary Register และ Auxiliary Pointer (ARP)

Data Memory Page Pointer (DP)

โดยที่ OVM (Overflow Mode) , TC (Test/Control Flax) , IMR (Interrupt Mask Register) , Auxiliary Register Pointer (ARP) , Auxiliary Register Pointer Buffer (ARB) และ Data Memory Page Pointer (DP) ไม่ต้องทำการ Initial ใหม่หลังทำการ Reset

3.6.2 Interrupt

C5x ประกอบไปด้วย External Maskable Interrupt จำนวน 4 ช่องและ Non-Maskable Interrupt อีกหนึ่งช่องไว้ให้กับอุปกรณ์ภายนอก และยังมี Interrupt ภายในซึ่งได้แก่ Interrupt ที่เกิดจาก Serial Port , Timer และ Interrupt ที่เกิดจาก Software ในคำสั่ง Interrupt อันได้แก่ INTR,TRAPและ NMI

C5x มีความสามารถในการสร้าง Software Interrupt โดยใช้คำสั่ง INTR ซึ่งจะทำให้บางส่วนจาก 32 Interrupt Service Routine ทำการบริการการ Interrupt โดยที่ 20 ISR ถูกจองไว้เพื่อบริการแก่ External Interrupt , Peripheral Interrupt และสงวนไว้ใช้ในอนาคต ส่วน Interrupt Vector ที่เหลืออีก 12 Location สามารถนำไปใช้ได้โดยผู้เขียน Program ในคำสั่ง INTR สามารถอ้าง Interrupt ใดๆ ก็ได้ภายใน 32 Location

Context Saving และ Restoring Function จะทำงานเมื่อ Interrupt Trap ถูก Execute และ 8-Deep Hardware Stack จะถูกใช้เพื่อเก็บค่า Address ที่จะต้องกลับมาทำงานต่อหลังจากการบริการการ Interrupt เสร็จสิ้นเรียบร้อยแล้ว ดังนั้นจะมี One Deep Stack (หรือ Shadow Register) เพื่อเก็บค่าของแต่ละ Register ดังต่อไปนี้

ACC	Accumulator
ACCB	Accumulator Buffer
PREG	Product Register
ST0	Status Register 0 (INTM not restored)
ST1	Status Register 1 (XF not restored)
PMST	Processor Mode Status Register
TREG0	Temporary Register for Multiplier
TREG1	Temporary Register for Shift count
TREG2	Temporary Register for Bit test
INDX	Indirect Address Index Register
ARCR	Auxiliary Register Compare Register

เมื่อได้รับสัญญาณ Interrupt Trap Register ทั้งหมดจะถูก Push ลงบน One-Deep Stack และ Shadow Register เหล่านี้จะถูก Pop เมื่อมีการกลับจาก Interrupt ด้วยคำสั่ง RET1 หรือ RETE

3.6.3 Software Stack

C5x มี Hardware Stack ภายในจำนวน 8 Deep ด้วยกันเพื่อใช้ในการเก็บค่า Address ในขณะทำการ Subroutine และ Interrupt

คำสั่ง PUSH และ POP มีการเพิ่มความสามารถของคำสั่งขึ้นมาเป็นคำสั่งใหม่อีก 2 คำสั่งได้แก่ PSHD และ POPD ทำให้ Stack สามารถ Store และ Recover ได้จาก Data Memory เมื่อมี Stack ถูกใช้ไปทั้งหมด 7 ค่าและมีการเขียนค่า Stack ทับลงไปอีกมากกว่า 2 ค่า โดยที่ค่าเก่ายังไม่ถูก POP ขึ้นมาจำเป็นที่จะต้องมีการขยาย Stack

เนื่องจากคำสั่ง Call ใช้ Stack ในการเก็บค่า Program Counter คำสั่ง BACC จะมีประโยชน์ในการกลับเข้าสู่ Main Program โดยไม่จำเป็นต้องเก็บค่าของ Program Counter ไว้ใน Memory

3.6.4 Logical and Arithmetic Operations

PLU จะช่วยทำให้การส่งผ่านข้อมูลข้ามไปยัง Data Memory ทำได้โดยไม่ส่งผลกระทบต่อค่าภายใน Accumulator หรือ Product Register ทำให้สามารถเคลื่อนย้ายค่าไปยังตำแหน่งใดๆ ภายใน Data Memory ได้โดยตรงกับ Source Operand

C5x ประกอบไปด้วยเงื่อนไขต่างๆ เพื่อทำการตรวจสอบก่อนที่จะกระโดดไปทำโปรแกรมอื่นๆ บางส่วนจาก 13 เงื่อนไข โดยสามารถทดสอบแต่ละเงื่อนไขหรือทดสอบเงื่อนไขร่วมกันได้โดยใช้คำสั่ง CC , RETC , XC และ BCND

Search Algorithm ด้วยคำสั่ง CRGT ด้วยการใช้คำสั่ง CRGT และ RPTB เราสามารถค้นหาค่าสูงสุดภายใน Data Block และตำแหน่งของมันได้ การหาค่าต่ำสุดสามารถใช้คำสั่ง CRLT ค้นหาได้ในลักษณะเดียวกัน

3.6.5 Matrix Multiplication Using Nested Loops

C5x มีคำสั่ง 3 คำสั่ง คำสั่งแรกคือ RPT (Single-Instruction Repeat) สามารถทำคำสั่งซ้ำได้ n ครั้ง คำสั่ง RPTB (Repeat Block) ทำงานด้วยจำนวน Loop ที่ตั้งไว้ภายใน BRCR Register ส่วนคำสั่ง BANZ (Branch if AR Not Zero) เป็นอีกคำสั่งที่สามารถกำหนดจำนวน loop ได้โดยใช้ Auxiliary Register

3.6.6 Circular Buffers

Circular Addressing เป็นหัวข้อสำคัญในชุดคำสั่งของ C5x การทำงานแบบ Convolution , Correlation และ FIR Filter สามารถทำงานได้โดยการใช้ Circular Buffer ภายใน Memory C5x มีการรองรับการทำงานแบบต่อเนื่องกันระหว่าง 2 Buffer โดยใช้ Auxiliary Register โดยมี CBSR1 , CBSR2 , CBER1 , CBER2 และ CBCR ทำหน้าที่เป็น Memory-Mapped Register ที่ควบคุมการทำงานของ Circular buffer

โดยก่อนที่จะเริ่มทำงานจะต้องมีการ load ค่า Address เริ่มต้นและสุดท้าย ลงใน Buffer Register และ Auxiliary Register ทำงานเป็น Pointer จะต้องทำการ Initial ค่าด้วย

3.6.7 Extended-Precision Arithmetic

การวิเคราะห์ตัวเลข , การคำนวณ Floating-Point หรือการทำงานอื่นๆ อาจมีความจำเป็นที่จะต้องใช้ตัวเลขที่มีค่ามากกว่า 32 bit เนื่องจาก C5x เป็น 16/32 bit Fixed-Point Processor ดังนั้นในการคำนวณจึงต้องอาศัยการขยาย bit ด้วย Software เข้าช่วยในการทำงาน

3.6.7.1 การคูณ

ประเด็นสำคัญที่คอยช่วยเหลือเมื่อทำการคำนวณแบบ Extended-Precision คือการใช้คำสั่ง MPYU (Unsigned Multiply) ซึ่งมีหน้าที่ในการคูณเลขแบบ 16 bit แบบไม่มีเครื่องหมาย 2 จำนวน และ

เก็บผลลัพธ์ที่ได้จากการคำนวณ ซึ่งมีขนาด 32 bit ไว้ภายใน Product Register ด้วยการทำงานภายในรอบคำสั่งเดียว

3.6.7.2 การหาร

การหารจำนวนจริงและทศนิยมสามารถทำได้ด้วยคำสั่ง SUBC ซึ่งเป็นคำสั่งพิเศษที่ทำการลบหลายๆ ครั้งติดต่อกันโดยกำหนดให้ตัวตั้งและตัวหารเป็นจำนวนบวกมีค่า 16 bit เมื่อทำคำสั่ง SUBC Processor จะทำการลบทั้งหมด 16 ครั้งแล้วจึงเก็บค่าผลหารที่มีขนาด 16 bit ไว้ในครึ่งล่างของ Accumulator และเก็บเศษอีก 16 bit ไว้ในส่วนบนของ Accumulator

การใช้คำสั่ง SUBC สามารถทำได้ด้วยวิธีการเช่นเดียวกับการหารยาว ตัวตั้งจะถูก Shift จนกระทั่งค่าในตัวหารถูกลบจนหมด แล้วเปลี่ยนเป็นค่าลบ ในการลบแต่ละครั้งที่ไม่ทำให้คำตอบมีค่าเป็นลบ ค่าหนึ่งจะถูกเลื่อนเข้าไปแทนที่ใน LSB

การหารทั้งสองแบบ ขั้นต้นทั้งตัวตั้งและตัวหารจะต้องมีค่าเป็นบวก ดังนั้นผลหารจะถูกกำหนดและการคำนวณค่าของผลหารจะใช้ค่า Absolute ของตัวตั้งและตัวหารเท่านั้น

3.6.8 Floating-Point Arithmetic

ในการทำงานทางคณิตศาสตร์แบบ Floating-Point ของ C5x ค่าของ Operand จะต้องถูกเปลี่ยนไปเป็นแบบ Fixed-Point และจะต้องเปลี่ยนกลับมาเป็นแบบ Floating-Point การเปลี่ยนเลขให้เป็น Floating-Point จะทำได้โดยการ Normalize Input Data ในการคูณจำนวน Floating-Point 2 จำนวน Mantissa จะต้องถูก Renormalized

คำสั่งของ C5x สำหรับการทำงานแบบ Floating-Point คือ NORM , SATL , SATH และ XC โดยคำสั่ง NORM จะถูกใช้เพื่อเปลี่ยนตัวเลขในแบบ Fixed-Point ไปเป็นตัวเลขในแบบ Floating-Point คำสั่ง SATL ทำงานร่วมกันกับคำสั่ง SATH โดยการใช้คาบเวลาเพียง 2 Cycle เพื่อทำการเลื่อน bit ไปทางขวาได้ตั้งแต่ 0-31 bit ส่วนคำสั่ง XC ช่วยป้องกันการเกิด Extra Cycle ที่เกิดจากคำสั่ง Branch

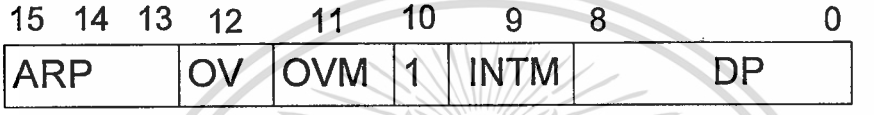
3.7 Register กำหนดสถานะที่สำคัญใน C5x

ในส่วนของ CPU จะมี Register สำหรับกำหนดสถานะการทำงานของ C5x ซึ่งมีรายละเอียดในการทำงานแต่ละส่วนได้หลายแบบจะถูกกำหนดโดย Register ทั้งหมด 4 ตัว คือ ST0,ST1,CBCR และ PMST ซึ่งสามารถที่จะ Store และ Load ค่าเหมือนกับ Data Memory และยอมให้มีการเปลี่ยนค่าโดยการเขียนและเก็บของเดิมไว้เมื่อมีการเรียกใช้ Subroutine หรือเมื่อเกิดการ Interrupt จะมีการเก็บค่าของมันไว้ 1 ระดับโดยจะมีการคืนค่าเมื่อมีการกลับจาก Interrupt (ดูรายละเอียดจาก คำสั่ง RETI RETE) ยกเว้น bit XF ใน ST1

โดย Register PMST และ CBCR จะอยู่ใน Page 0 ของ Data Memory ดังนั้นมันจึงถูกกระทำโดยตรง โดย CALU และ PLU โดยสามารถอ้างถึงและเก็บค่าเหมือนกับ Data Memory ธรรมดาโดยเมื่อมันมีการเปลี่ยนแปลงค่าจะมีผลกับการกระทำใน Pipeline ใน 2 Stage ถัดมา

ส่วนค่าใน ST0 และ ST1 จะถูกเขียนและอ่านโดยคำสั่ง LST และ SST ตามลำดับโดยไม่สามารถเขียนและอ่าน เหมือน PMST กับ CBCR ได้แต่คำสั่ง LST ไม่สามารถกำหนด bit INTM ได้แต่ทั้งหมดสามารถใช้คำสั่ง SETC,CLRC เพื่อทำให้เป็น 1 และ 0 ได้ตามลำดับ

ST0 (Status Register 0) มีขนาด 16 bit มีรายละเอียดดังนี้



ARP Auxiliary Register Pointer 3 bit นี้จะเอาไว้ชี้ Register AR ต่างๆเพื่อใช้ในการอ้าง Address แบบ Indirect เมื่อทำการเก็บค่าของ ARP ค่าของ ARP อันเก่าจะถูกเก็บใน ARB โดยค่าใน ARP สามารถเปลี่ยนได้หลายทางโดยการเปลี่ยนค่า เกี่ยวกับการอ้าง Memory ของหลายคำสั่งหรือคำสั่ง MAR จะกระทำกับ ARP โดยตรง

OV Overflow Flag bit จะเป็น bit แสดงการ Overflow จะเป็น 1 เมื่อเกิดการ Overflow ขึ้นใน ALU โดยเมื่อทำการ Reset bit นี้จะเป็น 1 สามารถ Clear ได้โดยการใช้คำสั่ง BCND(D) ที่มีเงื่อนไขบน bit นี้และคำสั่งอื่นๆ ลักษณะนี้หรือคำสั่ง LST

OVM Overflow Mode bit เมื่อ bit นี้เป็น 0 จะมีการ Check การเกิด Overflow จาก ALU เท่านั้นแต่ถ้า bit นี้เป็น 1 จะมีการ Check การเกิด Overflow จาก Timer และ Counter ด้วย สามารถใช้คำสั่ง SETC CLRC และ LST ทำการเปลี่ยนแปลงค่า bit นี้ได้

INTM Interrupt Bit Mode เมื่อ bit นี้เป็น 0 จะมีเฉพาะ Interrupt แบบ Non-Maskable เท่านั้นที่สามารถทำงานได้ แต่เมื่อเป็น 1 ทุกๆ Interrupt จะสามารถทำงานได้ โดยเมื่อทำการ Reset bit นี้จะเป็น 1 สามารถทำการเปลี่ยนแปลงค่าด้วยคำสั่ง SETC,CLRC เหมือน bit อื่นๆ แต่ไม่สามารถใช้คำสั่ง LST กับ bit นี้ได้โดยการเกิด Interrupt Maskable จะทำให้ bit นี้เป็น 0 และเมื่อมีการกลับจาก Interrupt Service Routine ด้วยคำสั่ง RETE จะทำให้ bit กลายเป็น 1

DP Data Memory Page Pointer ค่า 9 bit ในนี้จะเป็นค่าที่ จะไปรวมกับค่า 7 bit ในคำสั่งในการอ้าง Memory แบบ Direct โดย 9 bit นี้จะเป็น MSB ในการอ้าง Memory ซึ่งมี 16 bit โดยค่านี้สามารถเปลี่ยนแปลงโดยคำสั่ง LDP หรืออ่านโดยคำสั่ง LST

ST1 (Status Register 1) มีขนาด 16 bit มีรายละเอียดดังนี้

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB			CNF	TC	SXM	C	1	1	HM	1	XF	1	1	PM	

ARB Auxiliary Register Pointer Buffer เป็นตัวเก็บค่าสำรองของ ARP

CNF On-Chip RAM Configuration Control Bit ถ้า bit นี้เป็น 0 แสดงว่า RAM ส่วนนี้สามารถเป็นได้ทั้ง Program และ Data แต่ถ้าเป็น 1 จะเป็นส่วนของการเก็บโปรแกรมอย่างเดียว โดย bit นี้สามารถกระทำได้โดยคำสั่ง SETC,CLRC,LST#1 โดยเมื่อ Reset จะเป็น 0

TC Test Control/Flag Bit โดย bit นี้จะมีผลกับคำสั่ง BITT,BITT,CMPR,LST#1,NORM, CPL,XPL,OPL และ APL โดย bit นี้จะเป็น 1 เมื่อ

1.Bit ที่ทดสอบด้วยคำสั่ง BIT,BITT ได้ 1

2.เมื่อมีการเปรียบเทียบด้วยคำสั่ง CMDR คือเปรียบเทียบค่าใน RCR กับ AR ที่ ชี้โดย ARP

3.เมื่อมีการเปรียบเทียบด้วยคำสั่ง NORM คือทำการ XOR bit 30 กับ bit 31

4.เมื่อมีการใช้คำสั่ง CPL เปรียบเทียบ Data กับ DBMR

5.เป็นผลของ Function ทาง Logic ต่างๆ XPL,OPL,APL

Bit นี้สามารถใช้เป็นเงื่อนไขของ คำสั่ง BCND,RETC,CC และ XC

SXM Sign Extension Mode Bit ถ้ามีค่าเป็น 1 จะมีการ Check เครื่องหมายของการกระทำทางคณิตศาสตร์ต่างๆถ้าเป็น 0 จะไม่มีการคำนึงถึงเครื่องหมาย โดย สามารใช้คำสั่ง CLRC,SETC ทำการเปลี่ยนค่า bit นี้ได้โดย bit นี้จะเป็น 1 เมื่อทำการ Reset

C Carry Bit จะเป็น 1 เมื่อเกิดการเกินจากการบวกและจะเป็น 0 เมื่อเกิดการยืมจากการลบโดยหลังการบวกปกติจะเป็น 0 และหลังการลบปกติจะเป็น 1 แต่จะไม่มีการ Check จากการบวกลบที่มีการ Shift 16 bit ด้วย โดยคำสั่ง Shift และ Rotate ACC จะมีผลกับ bit นี้เช่นกัน โดยสามารถอ่านและแก้ไขโดยคำสั่ง SETC,CLRC และ LST#1 โดยเมื่อทำการ Reset จะเป็น 1

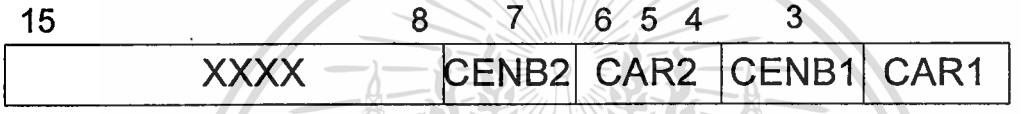
HM HOLD Mode Bit เมื่อเป็น 1 จะทำการหยุดการทำงานทุกอย่างแล้วทำการ Active ขา HOLD เมื่อเป็น 0 มันจะทำในโปรแกรม Memory ภายในจนหมดแล้วทำการ Set ขาที่ทำการติดต่อกับสัญญาณภายนอกให้เป็น High impedance

XF XF Pin Status Bit เป็นขาแสดงสถานะของขา XF ซึ่งสามารถนำไปควบคุมอุปกรณ์ภายนอก โดยสามารถใช้คำสั่ง CLRC,SETC ทำการเปลี่ยนแปลงค่าของ bit (ขา) นี้ได้แต่

ในการทำ Interrupt จะไม่มีการเก็บค่า bit นี้ไว้ ถ้าใน Interrupt Routine มีการแก้ไขค่า bit นี้จะไม่มีการคืนค่าเดิมหลังการ Return จาก Interrupt

PM Product Shift Mode จะเป็นตัวกำหนดเงื่อนไขการเก็บผลการคูณว่ามีจะมีการ Shift หรือไม่ โดยเมื่อเป็น 00 ผลคูณจะไม่มีการ Shift เมื่อส่งค่าผ่านไปยัง ALU , เป็น 01 จะมีการ Shift Left ไป 1 bit แล้วแทนค่า LSB ด้วย 1 , เป็น 10 จะ Shift Left 4 bit แล้วทำการแทน LSB ด้วย 0 , เป็น 11 จะ Shift Right 6 bit โดยจะแทนด้วย Sign Extension การ Shift จะมีผลกรณีที่มีการเก็บค่าไปยัง Data Memory ด้วยโดย 2 bit นี้จะเป็น 00 สามารถเขียนและอ่านโดย คำสั่ง SPM และ LST#1

CBCR Circular Buffer Control Register จะใช้งาน 8 bit



CAR1 เป็นตัวกำหนด Auxiliary register ที่จะทำการควบคุมโดย circular buffer1

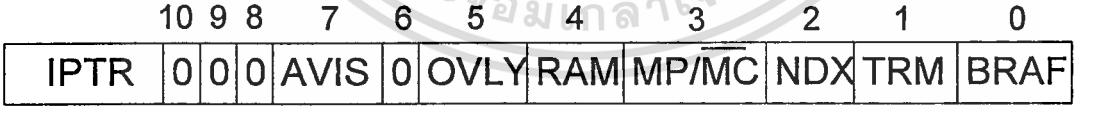
CAR2 เป็นตัวกำหนด Auxiliary register ที่จะทำการควบคุมโดย circular buffer2

CENB1 Circular bufer1 enable เมื่อเป็น 1 circular buffer1 จะทำงาน เป็น 0 ไม่ทำงาน

CENB2 Circular bufer1 enable เมื่อเป็น 2 circular buffer2 จะทำงาน เป็น 0 ไม่ทำงาน

โดยทั้งหมดจะเป็น 0 เมื่อ reset

PMST Processor Mode Status Register



IPTR Interrupt Pointer Vector เป็นค่ากำหนดการ Map Address ของการ Interrupt คือจะนำค่า Address ของ Interrupt Service Routine ที่กำหนดในตอนแรกสุดมาบวกกับค่าของ IPTR จะเป็น Address ใหม่ที่จะทำการ Service ซึ่งสามารถแก้ไขได้โดยค่าหลังจากการ Reset จะเป็น 0

AVIS Address Visibility Mode เมื่อขานี้เป็น 0 การเปลี่ยนแปลงของการ Access Address ภายในจะมีการนำมาแสดงที่ขา Address ด้วย แต่จะไม่มีผลกับสาย ฉ Control Data Bus โดยการเปลี่ยนแปลงนี้จะสามารถนำไปใช้ในการตรวจสร้าง

Interrupt Vector ร่วมกับขา IACK เมื่อ bit นี้เป็น 1 จะไม่มีการเปลี่ยนแปลงของขา Address เมื่อมีการอ้าง Address ภายในโดย bit นี้จะเป็น 0 เมื่อทำการ Reset

OVLY Ram Overlay Bit bit นี้จะทำการกำหนดให้ RAM ส่วนที่สำหรับเก็บโปรแกรมสามารถนำมาใช้เก็บข้อมูลได้ ถ้า bit นี้เป็น 1 ถ้าเป็น 0 ไม่สามารถทำได้ โดย bit นี้จะเป็น 0 เมื่อทำการ Reset

CNF	OVLY	DARAM B0	DARAM B1	DARAM B2	SARAM	Off-Chip
0	0	100h–2FFh	300h–4FFh	60h–7Fh		800h–FFFFh
0	1	100h–2FFh	300h–4FFh	60h–7Fh	800h–13FFh	1400h–FFFFh
1	0	–	300h–4FFh	60h–7Fh		800h–FFFFh
1	1	–	300h–4FFh	60h–7Fh	800h–13FFh	1400h–FFFFh

ตาราง 3.9 Local Data Memory Configuration Control

RAM เมื่อ bit นี้เป็น 1 จะกำหนดให้ RAM ภายในเป็นส่วนของการเก็บโปรแกรม ถ้าเป็น 0 จะไม่ใช้ สามารถแสดงการใช้ RAM ได้ดังตาราง

OVLY	RAM	On-Chip SARAM Configuration
0	0	Disabled
0	1	Mapped into program space
1	0	Mapped into data space
1	1	Mapped into both program and data spaces

ตารางที่ 3.10 On-Chip Single-Access RAM Configuration Control

MP / $\overline{\text{MC}}$ Microprocessor/Microcomputer Bit เมื่อเป็น 0 จะมีการใช้ ROM ภายในเมื่อเป็น 1 จะไม่มีการใช้ ROM ภายในโดยจะอ่านสถานะจากขานี้ในการ Reset ครั้งแรก หลังจากนั้นการ Reset ครั้งต่อไปจะไม่มีผล

NDX Enable Extra Index Register ถ้าเป็น 0 จะทำงานในโหมดการอ้าง Address แบบ Indirect ที่จะเหมือนกับตระกูล C2x ซึ่งก็คือสามารถใช้การอ้าง Address แบบ Indirect แบบ Indexing หรือการเปรียบเทียบค่า AR ได้เฉพาะ AR0 เท่านั้น แต่ในการทำงานของ C5x จะสามารถทำได้หมดเมื่อกำหนดให้ bit นี้เท่ากับ 1 โดยจะเป็น 0 เมื่อทำการ Reset

TRM Enable Multiple TREGs เป็น bit กำหนดการทำงานของ $TREG_0$ $TRGE_1$ $TREG_2$ ให้ทำงานในโหมดต่างๆ คือถ้าเป็น 0 จะทำงานในโหมดที่เหมือนกับ C2x ซึ่งทั้ง 3 Register จะมีผลเหมือนกันกับคำสั่งของ C2x แต่ถ้าเป็น 1 จะสามารถแยกใช้

T-Register แต่ละตัวเป็น Pre-Scaling Shifter หรือใช้กับคำสั่ง BITT ได้โดย bit นี้จะเป็น 0 เมื่อทำการ Reset

BRAF Block Repeat Active Flag เป็น bit แสดงการทำงานของ Block Repeat ซึ่งจะเป็น 1 เมื่อมีการทำงาน โดยเมื่อทำการ Reset bit นี้จะเป็น 0

3.8 Register อื่น ๆ ที่สำคัญ

Software wait-state Register ในการใช้งาน C5x ต่อกับอุปกรณ์ภายนอกถ้าอุปกรณ์ภายนอกมีการทำงานช้ากว่าการอ้างถึงของ C5x มันจะอนุญาตให้มีการรอการทำงานของอุปกรณ์ภายนอกซึ่งสามารถกำหนดได้ด้วย Software โดยมี Register ต่างๆ มากำหนดค่าพวกนี้ซึ่งจะมีทั้งหมด 3 ตัว คือ PDWSR(16 bit) , IOWSR(16 bit) และ CWSR(5 bit) ซึ่ง PDWSR จะเป็นตัวกำหนดการแบ่ง Memory ออกเป็น 4 Block อย่างละ 16k ทั้งของ Program Memory และ Data Memory , ส่วน IOWSR จะควบคุมทั้ง 16 ส่วนของ I/O โดยแบ่ง I/O ออกเป็น 16 ส่วนคือสามารถมี I/O ที่มี Wait State เท่ากันได้ 16 ชุด ส่วน CWSR จะเป็นตัวกำหนด Range ของ Wait State โดย bit Big ใน CWSR จะเป็นตัวกำหนดโดยถ้าเป็น 0 จะกำหนด Wait State แยกเป็น port เลย แต่ถ้าเป็น 1 จะกำหนดเป็น block อย่างละ 8k

โดยที่ Range ของ Wait State สามารถกำหนดได้ถึง 7 Machine Cycle เพื่อให้การติดต่อกับอุปกรณ์ภายนอกสามารถทำได้อย่างรวดเร็วที่สุดสำหรับแต่ละอุปกรณ์ ส่วนอุปกรณ์ที่มี Wait State มากกว่านี้สามารถควบคุมได้โดยผ่านทางขา Ready

Register	Bits	Space	Address Range	
PDWSR	0–1	Program	0000h–3FFFh	
	2–3		4000h–7FFFh	
	4–5		8000h–0BFFFh	
	6–7		0C000h–0FFFFh	
	8–9	Data	0000h–3FFFh	
	10–11		4000h–7FFFh	
	12–13		8000h–0BFFFh	
	14–15		0C000h–0FFFFh	
IOWSR		I/O	BIG = 0	BIG = 1
	0–1		Port 0/1, Port 10/11, etc.	0000h–1FFFh
	2–3		Port 2/3, Port 12/13, etc.	2000h–3FFFh
	4–5		Port 4/5, Port 14/15, etc.	4000h–5FFFh
	6–7		Port 6/7, Port 16/17, etc.	6000h–7FFFh
	8–9		Port 8/9, Port 18/19, etc.	8000h–9FFFh
	10–11		Port 0A/0B, Port 1A/1B, etc.	0A000h–0BFFFh
	12–13		Port 0C/0D, Port 1C/1D, etc.	0C000h–0DFFFh
	14–15		Port 0E/0F, Port 1E/1F, etc.	0E000h–0FFFFh

ตารางที่ 3.11 การใช้งาน bit ต่างๆใน PDWSR และ IOWSR

จากตารางจะเห็นว่ามีการแบ่ง bit ใน Register ทั้งสองออกเป็นคู่ๆ เพื่อกำหนด Wait State ให้ทั้งส่วนของ Program และ Data (ใน PDWSR) ซึ่งสามารถมีได้ 4 ค่าของ Wait State โดยจะเป็นค่าอะไรต้องดูจากค่าใน CRSR ประกอบเช่นใน bit 0 กับ 1 จะเป็นตัวกำหนด Wait State ของ Program Memory จากช่วง Address 0000h ถึง 3FFFh และส่วนของ I/O สามารถกำหนดได้ 2 แบบ ขึ้นอยู่กับ bit Big ใน CWSR โดยที่ถ้า bit Big เป็น 0 จะเป็นตัวกำหนดให้ I/O port ที่ Map เป็น Register เป็นคู่ๆ (ดูจากตาราง 5.12) แต่ถ้า bit Big เป็น 1 จะเป็นการกำหนด Wait State ให้ กับ I/O ที่แบ่งเป็นช่วงๆ เหมือน Memory โดยค่าของ Wait State แต่ละค่าจะดูจากค่าใน CWSR อีกครั้ง การกำหนดค่าของ Wait State นั้นจะเป็นการทำกับอุปกรณ์ภายนอกเท่านั้นจะไม่มีผลกับ Memory ภายในแต่อย่างใด

โดยที่ การกำหนดค่าใน CWSR จะทำการกำหนดค่าของ Wait State ให้กับส่วนต่างๆ ดังตาราง

n (Bit Position in CWSR)	Space
0	Program
1	Data
2	I/O (lower-half: Port 0–Port 7 if BIG=0, 0000h–7FFFh if BIG=1)
3	I/O (upper-half: Port 8–Port F if BIG=0, 8000h–0FFFFh if BIG=1)
4	BIG mode bit

ตารางที่ 3.12 การใช้งาน bit ต่างๆ ของ CWSR

โดยที่ค่าของ bit ต่างๆ จะมีผลกับค่าของ Wait State ดังตาราง (ต้องดูค่าที่กำหนดใน PDWSR และ IOWSR ประกอบด้วย)

Wait-State Field† of PDWSR or IOWSR (Binary Value)	No. of Wait States (CWSR Bit n = 0)	No. of Wait States (CWSR Bit n = 1)
00	0	0
01	1	1
10	2	3
11	3	7

ตาราง 3.13 ความสัมพันธ์ของค่า Wait State กับค่าใน PDWSR และ IOWSR

โดยหลังการ Reset Wait State ทุกๆ ค่าจะกลายเป็น 7 Machine Cycle โดยค่าใน bit Big จะเป็น 0

3.9 Register เกี่ยวกับ Interrupt

ในการใช้งานอุปกรณ์ประเภท Microprocessor ลักษณะการทำงานที่สำคัญอย่างหนึ่งคือการ Interrupt เราต้องทราบการทำงานของการ Interrupt เพราะมีความจำเป็นที่จะต้องกำหนดตั้งแต่ขั้นตอนการออกแบบ Hardware และการเขียนโปรแกรม โดย C5x มี Register ที่ทำการจัดการเกี่ยวกับการ Interrupt อยู่หลายตัวซึ่งอยู่ใน Register PMST คือ IPTR ซึ่งได้กล่าวถึงมาแล้วที่เหลืออีก 2 ตัวคือ IFR และ IMR ซึ่งมีรายละเอียดแต่ละตัวดังนี้ (ดูรายละเอียดเพิ่มเติมในส่วนของการจัดการ Interrupt)

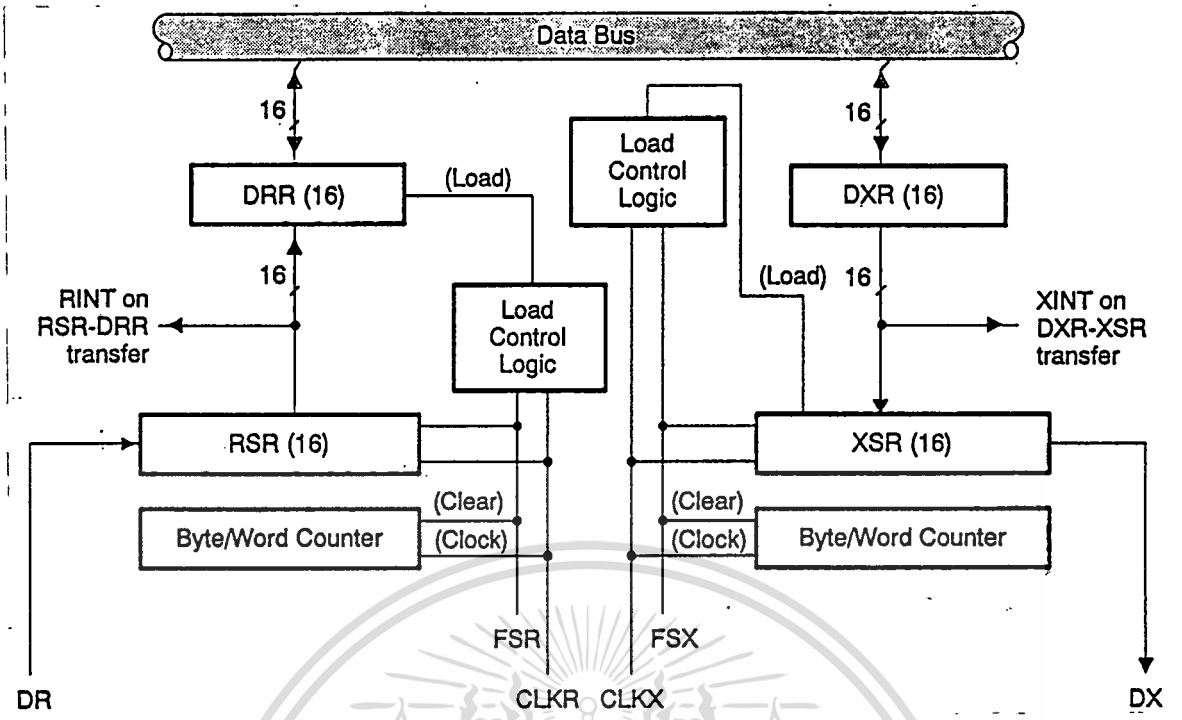
IFR Register (Interrupt Flag Register) ขนาด 9 bit ใน Data Memory ถือเป็น Memory Map Register ใช้ในการแสดงสถานะของ Maskable Interrupt ทั้ง 9 ตัว โดยจะเป็น 1 เมื่อ Interrupt นั้นทำงานโดยสามารถสั่ง Clear Interrupt ได้โดยการสั่งเขียน bit นั้นๆ เป็นส่วนการทำงานโดยปกติจะเป็นอัตโนมัติเมื่อเกิดการ Interrupt จะทำให้มันเป็น 1 อัตโนมัติและจะ Clear อัตโนมัติเมื่อมีการตอบรับนั้นโดยจะทำการ Clear พร้อมๆ กับ CPU สร้างสัญญาณ Interrupt Acknowledge โดยทั้งหมดจะเป็น 0 เมื่อทำการ Reset ค่า bit ต่างๆ ที่จะควบคุมการ Interrupt ใน Register นี้จะสามารถแสดงดังนี้

8	7	6	5	4	3	2	1	0
INT4	TXINT	TRINT	XINT	RINT	TINT	INT3	INT2	INT1

IMR (Interrupt Mark Register) ขนาด 9 bit ใน Data Memory ถือเป็น Memory Map Register เหมือนกับ IFR เป็น Register ที่จะเป็นตัวกำหนดการใช้งาน Interrupt ใดๆ โดยที่ bit ใดเป็น 1 จะแสดงว่าอนุญาตให้มีการใช้งาน Interrupt นั้นโดยสามารถเขียนและอ่านได้เหมือน Data Memory หรือคำสั่งเกี่ยวกับ Memory Map Register เช่น SAMM

3.10 Register ควบคุมการทำงานของอุปกรณ์ร่วม (Peripheral Interfacing Circuit)

SPC Register (Serial Port Control Register) เป็น Register ขนาด 16 bit ที่ใช้ในการควบคุมการทำงานของ Serial port ที่จะทำการติดต่อกับอุปกรณ์ภายนอกแบบอนุกรมโดย อยู่ในส่วนของ Memory Map Register โดยมีการกำหนดการทำงานต่างๆ เหมือนกับ Data Memory และ Memory Map Register อื่นๆ โดยมีรายละเอียดแต่ละ bit ดังนี้



รูปที่ 3.3 การทำงานของ Serial port ใน C5x

15	14	13	12	11	10
FREE	SOFT	RSRFULL	XSR EMPTY	XRDY	RRDY

9	8	7	6	5	4	3	1	2	0
IN1	INO	RRST	XRST	TXM	MCM	FSM	FO	DLB	RES

Bit 0 จะเป็น 0 ตลอดไม่มีการกำหนดค่าอะไร

Bit 1 DBL Digital Loop Back Mode เอาไว้ทำการตรวจสอบการทำงานของ Serial port โดยในเงื่อนไขนี้ ขา DR และ FSR จะถูกต่อกับ DX และ FSX โดย Clock ของการทำงานจะถูกกำหนดด้วยเงื่อนไขของ bit MCM

Bit 2 FO Format bit จะเป็นตัวกำหนด Mode การส่ง โดยถ้าเป็น 0 จะส่งใน Mode 16 bit (Word) ถ้าเป็น 1 ส่งใน Mode 8 bit (Byte) โดยจะส่งแบบ MSB First ทั้งสอง Mode

Bit 3 FSM Frame Sync Mode bit จะเป็นตัวกำหนด Mode การ Synchronous ของการส่ง ซึ่งถ้า bit นี้เป็น 1 ในการส่งแต่ละ Frame จะต้องการสัญญาณ FSX/FSR (สัญญาณ Sync ตอนจบ Frame ขณะรับและส่ง) สำหรับการส่งแต่ละ Word สำหรับการทำงานใน Mode ต่อเนื่อง bit นี้จะเท่ากับ 0

Bit 4 MCM Clock Mode bit เป็นการกำหนดค่า CLKX ถ้าเท่ากับ 0 จะมาจากขา CLKX ถ้าเท่ากับ 1 จะมาจากขา CLKOUT1 โดยถ้า bit DLB =1 และ MCM =1 Clock จะถูกกำหนดจากแหล่งกำเนิดสัญญาณภายใน

Bit 5 TXM Transmit Mode bit Configures จะเป็นตัวกำหนด Mode ของสัญญาณ Sync คือเมื่อ bit นี้เท่ากับ 0 จะเป็นการกำหนดขา FSX ให้แสดงเป็น Input แต่ถ้าเป็น 1 จะทำการกำเนิดสัญญาณนี้โดยอ้างอิงกับสัญญาณ CLKX และเป็นตัวกำหนดสัญญาณ Framing Signal

Bit 6,7 XRST RRST โดย 2 bit นี้จำเป็นต้องทำการเขียนให้เป็น 00 แล้วทำการเปลี่ยนแปลงค่าของ bit ที่ 1-5 เมื่อเสร็จแล้วจะต้องทำให้ 2 bit นี้เป็น 1 แล้วจะทำให้ไม่สามารถทำการเปลี่ยนแปลงค่าใดๆ ใน SPCได้ และ 2 bit นี้จะเป็นตัวแสดงการทำงานของ Serial port เช่นเมื่อ XRDY=0 จะทำให้ XRST เท่ากับ 0 ด้วยซึ่งหมายถึงเกิดการหยุดการทำงานของการส่งข้อมูลพร้อมๆ กับการเกิด Transmit Interrupt

Bit 8,9 2 bit นี้จะเป็น bit แสดงสถานะของ CLKR และ CLKX สามารถ Test ด้วยคำสั่ง BIT,BITT

Bit 10,11 RRDY XRDY Receive Ready and Transmit Ready bit เป็น bit แสดงสถานะของการรับส่งข้อมูลเช่นถ้า RRDY เปลี่ยนจาก 0 ไป 1 จะหมายถึงมีการ Copy ข้อมูลจาก RSR ไป DRR แล้วสามารถอ่านข้อมูลที่ DRR ได้แล้วโดยจะเกิดพร้อมๆ กับ Receive Interrupt ส่วนการทำงานของ XRDY ก็เหมือนกันแต่จะเป็นของด้านส่ง

Bit 12 XSEMPTY Transmit Shift Register Empty จะ Active Low เมื่อข้อมูลใน XSR ว่างโดยที่ข้อมูลใน DXR ยังไม่ได้รับข้อมูลใหม่หลังจากทำการ Load ข้อมูลจาก DXR ไป XSR ครั้งสุดท้าย ซึ่งจะทำให้ bit นี้เป็น Low ซึ่งแสดงถึงการเกิด Underflow ของการส่งข้อมูลจะไม่มีผลใน Burst Mode

Bit 13 RSRFULL เหมือนกับ bit ที่ 12 แต่จะเป็นการแสดงการ Overflow ของการรับข้อมูลคือเมื่อมีการรับข้อมูลจาก RSR มาไว้ใน DRR แต่ยังไม่มีการอ่านค่าจาก DRR แล้วมีข้อมูลชุดใหม่จาก FSR มาทำให้เกิดการเปลี่ยนจาก 0 ไปเป็น 1 ใน bit นี้

Bit 14 SOFT ถ้า bit นี้เป็น 0 จะสามารถหยุดการทำงานของ Serial port ได้ทุกเวลาในระหว่างการส่งถ้าเป็น 1 จะทำการหยุดได้เฉพาะเมื่อส่งข้อมูลหมดทั้ง Frame แล้วเท่านั้นโดยสามารถกำหนดการ Enable การใช้งานใน Mode การหยุดได้ที่ bit 15

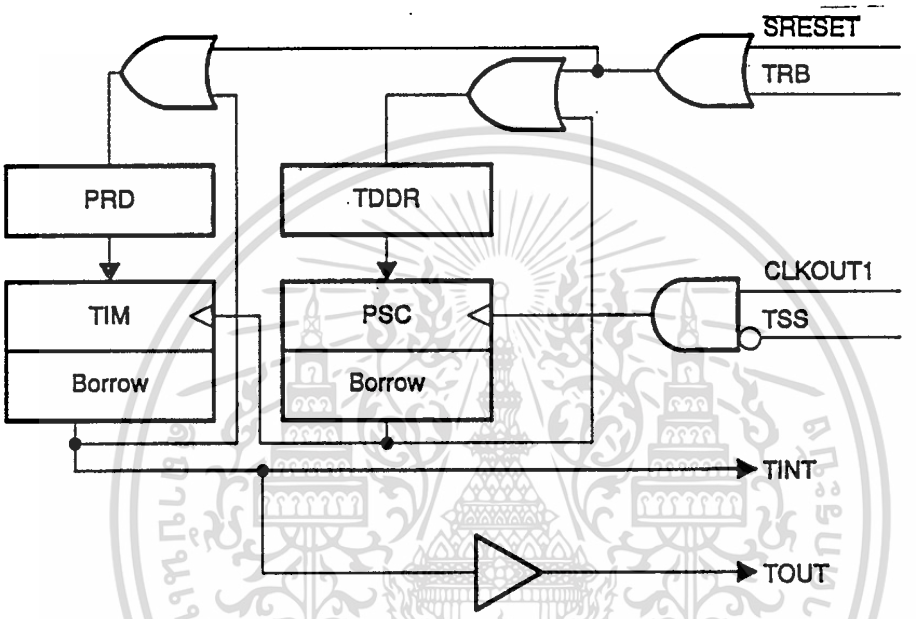
Bit 15 FREE เป็น bit กำหนดการ Enable การทำงานของ bit 14

TCR Register (Timer Control Register)

ใน C5x มี Hardware Timer ขนาด 16 bit โดยมี 4 bit Pre-Scalar สามารถกำหนดการ Control ผ่านทาง Register TCR ซึ่งถือเป็น Memory Map Register สามารถทำการอ่านและเขียนเหมือนกับ Memory ทั่วไปโดยที่อัตราการนับจะสามารถกำหนดได้ใน PRD และ PSC ซึ่ง Timer จะทำการนับโดย

นับลงทุกๆ Machine Cycle โดยเมื่อนับถึง 0 แล้วจะทำการสร้างสัญญาณ Interrupt ของ Timer ขึ้นมาพร้อมๆ กับการเปลี่ยนค่าที่ขา TOUT โดยอัตราการหารหรือการ Interrupt จะเป็น

$$Tout = \frac{1}{t_c \times (TDDR + 1) \times (PRD + 1)}$$



รูปที่ 3.4 แสดง Block Diagram การทำงานของ Timer ใน C5x

โดยที่ค่า PRD จะสามารถเขียนใน Register นี้โดยตรงแต่ค่าของ TDDR จะทำการเขียนผ่าน Register TCR :ซึ่งรายละเอียดของ Register TCR จะสามารถแสดงได้ดังนี้

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

โดยที่

Bit 0-3 TDDR (Timer division down ratio) จะเป็นตัวกำหนดอัตราการ หาร โดยจะโหลด ค่าจาก TDDR ไปเก็บใน PSC แล้วทำการนับลงโดยขณะนับจะไม่เกี่ยวกับค่าใน TDDR

Bit 4 TSS จะเป็นตัวกำหนดการ Start stop ของ timer โดยเป็น 1 จะ Stop timer เป็น 0 จะ Start timer

Bit 5 TRB (timer reset) จะเป็น bit ที่ทำการสั่ง Reset timer ซึ่งจะทำการ load ค่าจาก TDDR ไปเก็บใน PSC แล้วจะทำการนับใหม่ โดยขณะอ่าน bit จะเป็น 0 ตลอด

bit 6-9 PSC จะทำการนับลงมาเรื่อยๆตามการทำงานของ CPU โดยไม่สามรถอ่านได้ต้องทำให้มันหยุดการทำงานก่อนโดยการให้ TSS=1

ส่วน bit FREE และ SOFT จะทำงานเสมือนกับของ Serial port อีก 4 bit ที่เหลือไม่มีการใช้งาน

3.11 การจัดการกับ Interrupt ใน C5x

ในส่วนของ CPU C5x จะมีระดับความสำคัญของการใช้งาน Interrupt อยู่ 3 ระดับคือ Non-maskable Interrupt, Maskable Interrupt และ Software Interrupt โดยมีระดับความสำคัญของการทำงานตามลำดับ โดยการ Interrupt ที่มีความสำคัญสูงสุดคือการ Reset โดยจะเป็นการกำหนดค่าต่างๆ ให้กับ CPU ใหม่เกือบทั้งหมดโดยเมื่อการ Reset จะมีการกำหนดค่าต่างๆ ดังนี้

1. Bit CNF(On-Chip RAM Configuration Control bit)ใน Register ST1 จะถูกจะเป็น 0 ทำให้การทำให้การใช้งาน Dual Access RAM Block ถูกนำไปใช้ในการเก็บ Data
2. โปรแกรม Counter จะถูกทำให้เป็น 0 โดยขณะที่ขา RS ยังคงเป็น 0 ค่าที่ขา Address จะไม่ถูกกำหนดค่าแต่ถ้าขา HOLD ถูกทำให้เป็น 0 พร้อมกับ RS จะทำให้เกิดการสร้างสัญญาณ HOLDA คือการทำให้สาย Address เป็น High Impedance
3. Bit INTM จะถูก Set เป็น 1 ทำให้ทุกๆ Maskable Interrupt ไม่สามรถทำงานได้และทุกๆ bit ใน IFR จะเป็น 0
4. Status bit ต่างๆ ใน Status Register จะถูก load ค่าต่างๆ ดังนี้ OV=0,XF=1,SXM=1,PM=0,HM=1,BRAF=0,TRM=0,NDX=0,CENB1=0,CENB2=0, IPTR=0,OVLY=0,AVIS=0,RAM=0,BIG=0,CNF=0,INTM=1,C=1 และค่า MP/MC จะอ่านจากสถานะของขานี้ (ในการ Reset ครั้งแรก)
5. Register GREG (Global Memory Allocation Register) จะ Clear เพื่อทำการชี้ที่ Memory ภายใน
6. Clear Repeat Counter
7. สัญญาณ Interrupt Acknowledge จะถูกสร้างเหมือนกับเกิดการ Maskable Interrupt
8. สัญญาณ Synchronous Reset จะถูกสร้างขึ้นเพื่อทำการ Reset อุปกรณ์ต่างๆ ภายใน

โดย Interrupt ที่มีความสำคัญรองลงมาคือ Interrupt จากขา NMI โดยลำดับความสำคัญในการ Interrupt และ Address ที่ CPU จะกระโดดไปทำหลังเจอการ Interrupt ได้แสดงดังตารางที่ 3.14

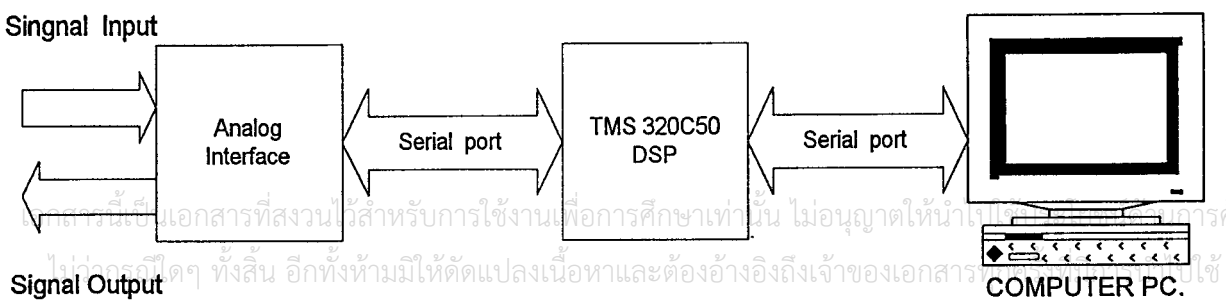
Name	Location		Priority	Function
	Dec	Hex		
RS	0	0	1 (highest)	External reset signal
NMI	36	24	2	Nonmaskable interrupt
INT1	2	2	3	External user interrupt #1
INT2	4	4	4	External user interrupt #2
INT3	6	6	5	External user interrupt #3
TINT	8	8	6	Internal timer interrupt
RINT	10	A	7	Serial port receive interrupt
XINT	12	C	8	Serial port transmit interrupt
TRNT	14	E	9	TDM port receive interrupt
TXNT	16	10	10	TDM port transmit interrupt
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	Reserved
——	40–63	28–3F	N/A	Software interrupts

ตารางที่ 3.14 ลำดับความสำคัญและ Addressของการบริการ Interrupt ต่างๆ

โดยระดับความสำคัญของ Software Interrupt จะไม่ได้กำหนดไว้เพราะจะไปกำหนดในโปรแกรม อีกที จะเห็นว่ามีการเว้นที่ว่างไว้สอง Word เพื่อบรรจุคำสั่ง Branch ต่างๆ ซึ่งจะมีความยาว 2 Word ส่วนของตำแหน่งของโปรแกรมที่ทำการ Service จะนำไปรวมกับค่า ใน IPTR ซึ่งตอนแรกจะเป็น 0 ก็จะเป็น Address ที่กำหนดในตาราง โดยถ้าค่าของ IPTR จะสามารถ Map ค่าได้ภายใน 32 Block อย่างละ 2K

3.12 TMS320C5x DSK Starter Kit

ส่วนสำคัญในโครงงานนี้ เป็นชุดพัฒนาโปรแกรม สำหรับ Chip DSP ตระกูล TMS320C5x ซึ่งเป็น CHIP Digital Signal Processing ตระกูล Fixed-Point ซึ่งนำมารวมกับ Chip Analog Interface เป็นบอร์ดที่จะใช้ในการทดลองเกี่ยวกับ DSP ในชุดทดลองนี้ประกอบด้วย



- ส่วนของ Debugger ที่ใช้พัฒนาโปรแกรม โดยทำการแก้ไขผ่านทาง Computer PC (โดยติดต่อกันผ่าน Serial Port ของ PC) ซึ่งสามารถ แก้ไข ดูค่า Register ,Run program ที่ละ step เพื่อดูผลการแปลงของค่า ต่างๆ
- ส่วนที่เป็น ส่วนรับ,ส่ง, สัญญาณจะผ่าน RCA Jack โดยสัญญาณ Analog ทั้งสองขั้วจะถูกจัดการโดย Chip Analog Interface เบอร์ TLC 32040 ซึ่งภายในประกอบด้วย Analog to Digital และ Digital to Analog ขนาด 14 bit ซึ่ง Chip สามารถโปรแกรม การทำงานของมันผ่านทาง Register ภายในโดยที่มันติดต่อกับ TMS320C5x แบบ Serial ทั้งรับและส่งข้อมูล ทั้งการโปรแกรมค่าต่างๆใน Register
- ส่วนที่สำคัญที่สุดคือ Chip DSP TMS320C50 เป็นส่วนที่นำมาทำการรับค่าเข้ามาเพื่อกระทำการคำนวณแบบต่างๆ ตามโปรแกรมโดยภายในจะมี ส่วนของ ROM และ RAM ในการเก็บ Program และ Data เพื่อการเข้าถึงที่รวดเร็วเพราะในการประมวลผลสัญญาณ แบบ Real Time ต้องใช้ความเร็วของอุปกรณ์สูง ส่วนของการเก็บ Program และ Data (10K RAM)ก็มีมากพอสำหรับ Application ขนาดใหญ่ทางด้าน Signal Processing

นอกจากนี้ยังมีความสะดวกในการพัฒนาโปรแกรม เพราะเป็นชุดทดลองสำหรับการเริ่มต้นการเรียนรู้การทำงานของ DSP ตัวนี้ที่บริษัทผลิต Chip มาสำหรับผู้เริ่มต้นศึกษาและยังมี Program ตัวอย่าง ซึ่งเป็น Program เกี่ยวกับการนำไปใช้งานทางด้านต่างๆ (สามารถ Download ได้เพิ่มเติมที่www.ti.com) รวมทั้งมีส่วนของการรองรับการขยาย Hardware ซึ่งสามารถทำได้สะดวก

หมายเหตุ:ในบทนี้ไม่ได้กล่าวถึงรายละเอียดของวงจรต่างๆ ของบอร์ดและการทำงานของ Chip Analog Interface โดยสามารถศึกษาข้อมูลเพิ่มเติมได้จาก User's Guide ของ TMS320C5x Starter kit ส่วนรายละเอียดเพิ่มเติมที่ไม่ได้กล่าวถึงของ C50 สามารถศึกษาได้จาก TMS320C5x User's Guide

บทที่ 4
การเขียนโปรแกรมบน Chip DSP

เนื่องจากการ Simulation ดูผลตอบสนองจากระบบการประมวลผลแบบดิจิตอลนั้นมีการศึกษากันอย่างกว้างขวาง แต่ในการนำมาทำการทดลองเพื่อประมวลผลกับสัญญาณจริงๆ มีน้อยมาก ซึ่งในวัตถุประสงค์หลักของโครงงานนี้จะทำในส่วนนี้ โดยจะทำการศึกษาในส่วนของการใช้งานและการโปรแกรม Chip DSP ซึ่งได้ทำการเสนอในบทที่ผ่านมาแล้ว จากสมการที่ทำการ Simulation ในโปรแกรม Matlab ในขั้นตอนนี้เราจะทำการเขียนโปรแกรมลงใน Chip DSP เพื่อดูผลตอบสนองต่อสัญญาณ จริงๆ

4.1 Frequency Response of Analog Interface Chip

ก่อนที่จะทำการโปรแกรม Chip เพื่อให้ทำงานเป็น Filter และทำการวิเคราะห์การตอบสนองในรูปแบบต่างๆ เราต้องทราบว่าที่ภาคสุดท้ายในส่วนของ Digital to Analogใน IC Analog Interface (TLC 32040CFN) จะมี Low pass Filter (Reconstruction Filter) อยู่ทำให้การวิเคราะห์การตอบสนองความถี่ของ Filter ที่เราจะโปรแกรมลงบน Chip DSP เกิดการผิดพลาดได้ฉะนั้นก่อนอื่นเราจะต้องทำการวิเคราะห์การตอบสนองความถี่ของส่วนนี้เสียก่อน โดยทำการเขียนโปรแกรมรับค่าเข้ามาแล้วทำการส่งออกไปเลยโดยไม่ต้องทำการคำนวณใดๆ เลยโดยโปรแกรมสามารถแสดงได้ดังนี้

```
;          SAMPLING FREQUENCY =19.53 KHz to test response of  low pass Filter in digital to
analog output  in analog interface

      .MMREGS
      .ds    0f00h
TA    .word   16
RA    .word   16          ; This set up of AIC registers give
                          ; a sampling freq of 19.53 KHz
TB    .word   16            ;
RB    .word   16            ;
AIC_CTR .word   08h

      .ps     0080ah
rint:  B      RECEIVE
xint:  B      TRANSMIT
      .ps     0a00h
```

```
        .entry          ;
        ;-----------------------------------------------
        SETC   INTM
        LDP    #0
        OPL    #0834h,PMST
        LACC   #0
        SAMM   CWSR
        SAMM   PDWSR
        SETC   SXM          ; SXM MUST BE SET
        SPLK   #022h,IMR    ; This turns on receive interrupt only

        CALL   AICINIT
        SPLK   #12h,IMR     ; This turns on receive interrupt only
        CLRC   OVM
        SPM    0
        CLRC   INTM

WAIT:                       ; a main program would go here
        B      WAIT

RECEIVE:
        LDP    #TEMP
        CLRC   INTM         ; ENABLE INTERRUPTS FOR DEBUGGING DSK PURPOSES
        LAMM   DRR          ; load accumulator with word received from AIC
        SAMM   DXR          ; write output word to transmit register
        RETE

TRANSMIT:
        RETE

;  DESCRIPTION: This routine initializes the 'C50 serial port   *
;           and the TLC320C40's (AIC) TA,RA,TB,RB and        *
;           control registers                                *

AICINIT: SPLK   #20h,TCR        ; To generate 10 MHz from Tout
```

```
        SPLK    #01h,PRD        ; for AIC master clock
        MAR     *,AR0
        LACC    #0008h          ; Non continuous mode
        SACL    SPC             ; FSX as input
        LACC    #00c8h          ; 16 bit words
        SACL    SPC
        LACC    #080h           ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; and taking it high after 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG
        ;----------------
        LDP     #TA             ;
        SETC    SXM             ;
        LACC    TA,9            ; Initialize TA and RA register
        ADD     RA,2            ;
        CALL    AIC_2ND         ;
        ;----------------
        LDP     #TB
        LACC    TB,9            ; Initialize TB and RB register
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ;
        ;----------------
        LDP     #AIC_CTR
        LACC    AIC_CTR,2       ; Initialize control register
        ADD     #03h            ;
        CALL    AIC_2ND         ;
        RET                     ;

AIC_2ND:
        LDP     #0              ; Data page point is 0  (MM regs)
        SACH    DXR             ; send ACChi 00
```

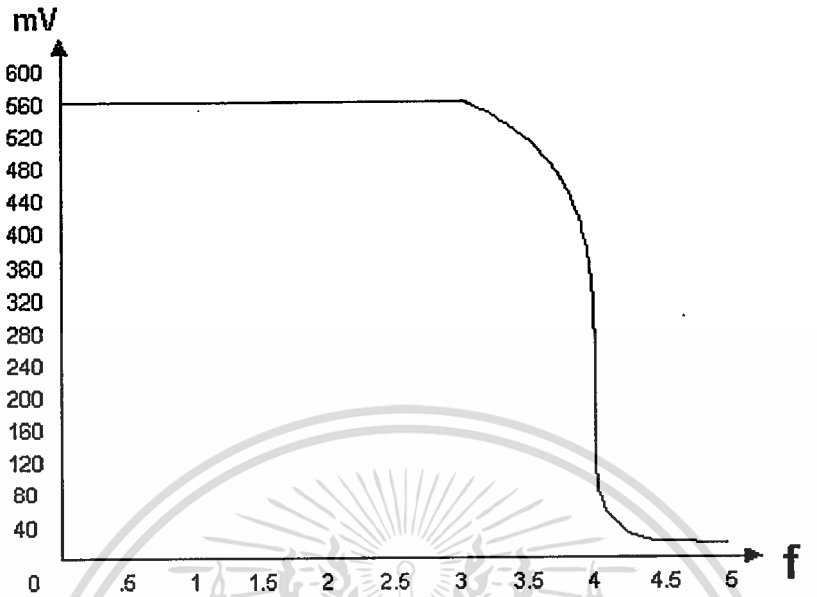
```
CLRC   INTM        ; enable interrupts
IDLE               ; wait for interrupt
ADD    #6h,15      ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
SACH   DXR         ; send ACC Hi to initiate secondary protocol
IDLE               ; wait for interrupt
SACL   DXR         ; send the T register data
IDLE               ; wait for interrupt
LACL   #0          ; clear ACC Lo
SACL   DXR         ; send another to make sure 1st word got sent
IDLE               ; wait for interrupt
SETC   INTM
RET
.END
```

โดยความถี่ Sampling ที่ระบุและเปลี่ยนแปลงในการทดลองนี้จะไม่มีผลกับ Low pass Filter เพราะเราจะทำการเปลี่ยนแปลงเฉพาะค่าใน Register TB,RB ซึ่งจะไม่มีผลกับการตอบสนองความถี่ของ Low pass Filter(Switch Capacitor Filter) เพราะ Clock ที่ใช้ในส่วนนี้จะมาจากการหาร Master Clock กับค่าใน Register TA,RA (ดูรายละเอียดการทำงานของ Chip Analog Interface จาก DSK Starter Kit User Guide)
จากนั้นทำการรันโปรแกรมข้างต้นสามารถเก็บค่าการตอบสนองที่ความถี่ต่างๆ ได้ดังนี้

Frequency	Volt Output
500 Hz	560 mV
1.00 kHz	560 mV
2.00 kHz	560 mV
3.00 kHz	560 mV
3.40 kHz	544 mV
3.60 kHz	440 mV
3.82 kHz	395 mV
3.85 kHz	320 mV
3.90 kHz	232 mV
4.00 kHz	120 mV
4.50 kHz	16 mV

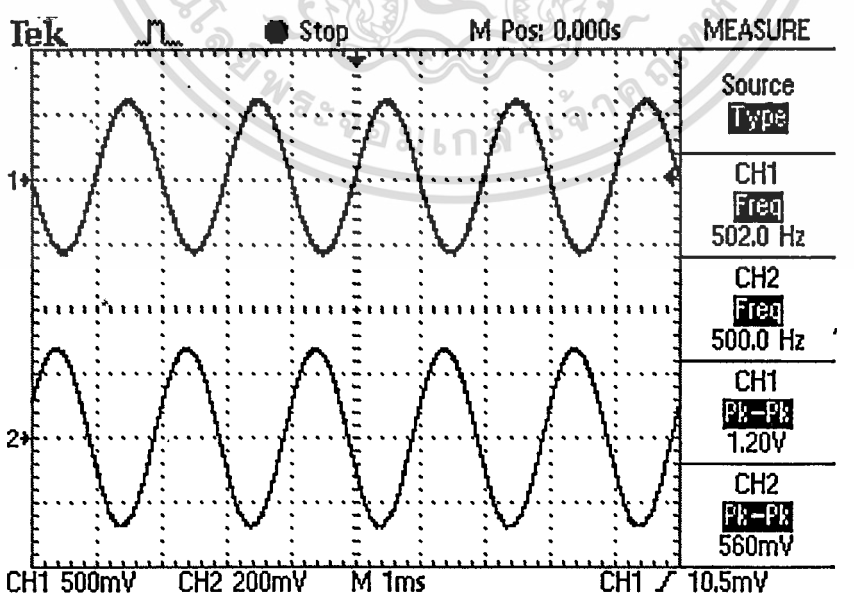
ตารางที่ 4.1 การตอบสนองความถี่

ซึ่งสามารถ Plot เป็น กราฟได้ดังนี้

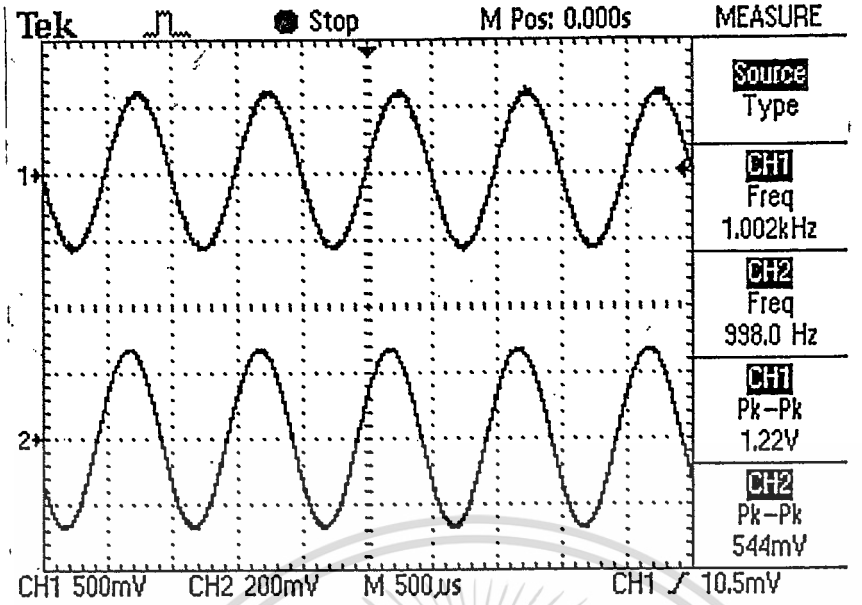


รูปที่ 4.1 กราฟแสดงการตอบสนองความถี่

จากกราฟจะเห็นว่าความถี่ Cutoff ของ Low pass Filter จะอยู่ที่ 3.82 KHz ซึ่งเวลาทดลองจะต้องอยู่ในช่วงไม่เกินความถี่นี้ อีกอย่างคือการตอบสนองเชิง Phase ของมัน ซึ่งสามารถแสดงโดยรูปของการตอบสนองที่ความถี่ต่างๆ



รูปที่ 4.2 แสดงสัญญาณ Input และ Output ที่ความถี่ 500Hz



รูปที่ 4.3 แสดงสัญญาณ Input และ Output ที่ความถี่ 1 KHz

จากรูปแสดงถึงการตอบสนองทาง Amplitude และ Phase จะเห็นว่ามีการ Shift Phase ไปของแรงดัน Output โดยจากการทดลองเปลี่ยนค่าความถี่พบว่าจะมีการ Shift Phase ไปประมาณ 90 องศาที่ความถี่ประมาณทุกๆ 600 Hz

จากข้อมูลข้างต้นจะทำให้สามารถวิเคราะห์การตอบสนองแบบต่างๆของ Filter ที่เราจะทำการทดลองได้ถูกต้องขึ้นทั้งทาง Amplitude และ Phase

4.2 IIR Band pass Filter

ในขั้นตอนนี้จะนำเอาสมการ ของ Band pass IIR Filter มาทำการเขียนโปรแกรมเพื่อดูผลตอบสนองของ Transfer Function ที่เราทำการ Simulate ตอนแรก โดยดูผลการตอบสนองว่าจะเหมือนที่ทำการ Simulate หรือไม่

จากสมการที่ (2.9) Time Domain ของ IIR Band pass Filter คือ

$$y(k)=\left(\frac{1-\alpha_0}{2}\right)[x(k)-x(k-2)]+\alpha_1(1+\alpha_0).y(k-1)-\alpha_0 y(k-2)$$

โดยกำหนดความถี่การ Sampling เท่ากับ 15.625 KHz (กำหนดจาก ค่า TA,RA, TB,RB ในโปรแกรม)

โดยกำหนดความถี่กลางของ Filter = $\pi/20$

ทำให้ได้ ความถี่กลางจริงๆ ใน Time Domain ได้จากสมการที่ 2.8

$$f = \frac{\pi/20}{2\pi(1/19.51325^3)}$$

$$= 390.625 \text{ KHz}$$

โดย $\alpha_1 = \cos(\pi/20) = .9876$

ค่า $\alpha_0 = 0.92$ (Q-Factor)

จากสมการด้านบนเราทำการแทนค่าคงที่ต่างๆได้ดังนี้

$$y(k) = 0.04x(k) - 0.04x(k-2) + 1.896y(k-1) - 0.92y(k-2)$$

4.2.1 การแปลงค่าจำนวนทศนิยมมาทำการคำนวณในรูปจำนวนเต็ม

ค่าคงที่ต่างๆ ในแต่ละเทอมต้องทำการแปลงก่อนที่จะทำการเขียนลง Chip เพราะค่านี้จะเป็นค่า ที่อยู่ในรูปของจำนวนทศนิยม (Floating-Point) ส่วนใน Chip ตระกูล C50 จะสามารถคำนวณในรูปจำนวนเต็ม (Fixed-Point) ได้เท่านั้นโดยในการคำนวณนั้นจะต้องทำการนำค่าเหล่านี้มาทำการคูณกับค่าที่ A/D แปลงมาซึ่งจะอยู่ในรูปของจำนวนเต็มเช่นกันแต่ระดับแรงดันใน Domain ของ Analog นั้นจะเป็นค่าทศนิยมเช่นกัน ซึ่งในการคูณกันนั้นผลออกมาจะไม่ถูกต้องตามความจริง เพราะการคูณค่าทศนิยมกับจำนวนใดๆ นั้นจะทำให้คำตอบที่มีค่าน้อยลง เพราะฉะนั้นการทำการเก็บค่าคำตอบของการคูณนั้นต้องทำการ Scale ค่าให้อยู่ในค่าที่ถูกต้องก่อน ซึ่งการ Scale ค่าของการคำนวณในรูป Binary คือการทำการ Shift นั่นเองโดยการ Shift ขวาไปหนึ่ง bit หมายถึงการหารสองในฐาน 10 ส่วนการ Shift ซ้ายไปหนึ่ง bit คือการคูณด้วยสองในฐาน 10 โดยในการ Shift ซ้าย หรือการหารนั้นจะเป็นค่าที่ bit LSB ไม่นำมาคิดเพราะการ Shift ซ้ายนั้นจะตัด LSB ทิ้ง หรือถ้าจะมองในอีกแง่หนึ่งคือการหารด้วยการ Shift นั้นจะได้ค่าที่ถูกต้องก็ต่อเมื่อ จำนวน bit ทางด้าน LSB นั้นจะต้องเป็น 0 เท่ากับจำนวน bit ที่ทำการ Shift ซึ่งใน Ship DSP โดยทั่วๆ ไปนั้นจะมี Shifter ที่ทำหน้าที่ในส่วนนี้อยู่แล้ว ซึ่งส่วนมากจะทำในขั้นตอนของการเก็บและโหลดค่า

โดยในการแปลงค่าสัมประสิทธิ์ของ Filter ในที่นี้จะทำการแปลงโดยอ้างอิงกับการแปลงแรงดันของ A/D มาอยู่ในรูป Binary ที่จะทำการคำนวณเพราะจะเป็นการง่ายในการ Scale ค่าหลังการคำนวณโดยจะอ้างอิงกับแรงดันอ้างอิงในการแปลงค่าของ A/D ซึ่งก็คือ 2 Volt แต่ในการแปลงนั้นจะมีทั้งส่วนของบวกและลบโดยการแปลงในส่วนของแต่ละค่าจะเป็น 14 bit ซึ่งทำให้ได้รายละเอียดเท่ากับ 2^{14} =16384 ระดับ ซึ่งค่านี้จะเท่ากับ 1 ในโดเมนของแรงดัน

ซึ่งทำให้สามารถคำนวณค่าคงที่ต่างๆได้ดังนี้

$$0.04 \times 16384 = 655.36$$

$$1.896 \times 16384 = 31064$$

$$0.92 \times 16384 = 15073$$

สามารถเขียนสมการกับค่าคงที่ในหน่วย ของค่าที่ทำการ Scale แล้วได้ดังนี้

$$y(k) = 655x(k) - 655x(k-2) + 31064y(k-1) - 15073y(k-2) \quad (4.1)$$

โดยสามารถเขียนโปรแกรมเพื่อให้ Chip DSP ทำการคำนวณหาแรงดัน Output แต่ละขณะเวลาเพื่อส่งออกไปที่ Digital to Analog เพื่อทำการแปลงกลับเป็น Analog

โปรแกรมสามารถแสดงดังได้ดังนี้

```
;                    BAND-PASS   IIR   FILTER
;
; -> IIR FILTER
;   This program uses an order2 IIR filter to implement a BANDPASS  filter.
;   The center frequency is 390 Hz
; ---------------------------------------------------------------------

        .MMREGS
        .ds     0f00h
TA      .word   16
RA      .word   16              ; This set up of AIC registers give
                                ; a sampling freq of 15.625 Hz

TB      .word   20
RB      .word   20
AIC_CTR .word   08h
OUTPUT  .word   0

; Filter Coefficient Generator  bandpass @ fcut=390.625Hz
a2      .word   -655    ; -0.04         X(n-2)   coefficient
a1      .word    0      ;  0            X(n-1)
a0      .word    655    ;  .04          X(n)
b2      .word   -15073  ; -0.92         Y(n-2)
b1      .word    31067  ;  1.896        Y(n-1)

;Storage deley line(y(n-1)...Y(n-N),X(n).....X(n-N)
```

```
YN      .word   0,0
XN      .word   0,0
XNL     .word   0

        .ps     0080ah          ; Program initial  Address for interrupt
rint:   B       RECEIVE
xint:   B       TRANSMIT

        .ps     0a00h           ; Program initial  Address
        .entry          ;
        ;-----------------------------------------

        SETC    INTM            ; Disable All interrupt  To modify  internal register
        LDP     #0              ; DATA Page Pointer  point to memory  map register
        OPL     #0834h,PMST     ; Write to PMST register
        LACC    #0              ;Assign to  value   to software wait state  genarate
        SAMM    CWSR
        SAMM    PDWSR
        SETC    SXM             ; Enable sign extension mode ( SXM bit=1 )
        SPLK    #022h,IMR       ; This turns on transmit interrupt only (IMR =interrupt mark register )
                                ; send data to set  AIC value
        CALL    AICINIT     ;
        SPLK    #12h,IMR        ; This turns on receive interrupt only
        CLRC    OVM
        SPM     0       ;
        CLRC    INTM        ;

WAIT:                           ; a main program would go here
        B       WAIT        ;

RECEIVE:
        LDP     #XN
        CLRC    INTM            ; ENABLE INTERRUPTS FOR DEBUGGING DSK PURPOSES

        LAMM    DRR             ; load accumulator with word received from AIC
```

```
                        ; Next line should be commented out if the
    SACL   XN           ; store the value of received word to a variable
    LAR    AR0,#XNL     ; load AR0 with address of last delay element
    ZAP                 ; ZERO ACC AND PRODUCT REGISTERS
    MAR    *,AR0        ; AR0 is the current AR register.

    RPT    #4           ; Repeat the next instruction 4 times through
    MACD   #a2,*-       ; the complete coeff table.
    APAC                ; Accumulate last product

    SACH   OUTPUT,1     ; ACC -> OUTPUT.  Get rid of xtra sign bit.
    LACC   OUTPUT       ;
    SFL                 ; left shift
    SACL   YN
    AND    #0fffch      ; Two LSB's must be zero for AIC
    SAMM   DXR          ; write output word to transmit register
    RETE

TRANSMIT:
    RETE

; DESCRIPTION: This routine initializes the 'C50 serial port   *
;          and the TLC320C40's (AIC) TA,RA,TB,RB and        *

;          control registers                    *
;AICINIT: SPLK   #20h,TCR     ; To generate 10 MHz from Tout(TCR =timer control register)
      SPLK   #01h,PRD        ; for AIC master clock
      MAR    *,AR0
      LACC   #0008h          ; Non continuous mode
      SACL   SPC             ; FSX as input   (SPC= serial port control register)
      LACC   #00c8h          ;
      SACL   SPC
      LACC   #080h           ; Pulse AIC reset by setting it low
```

```
        SACH   DXR
        SACL   GREG              ;generate reset signal with access to  external memory
        LAR    AR0,#0FFFFh
        RPT    #10000            ;
        LACC   *,0,AR0           ; (.5ms at 50ns)
        SACH   GREG
        ;-----------------
        LDP    #TA         ;
        SETC   SXM         ;
        LACC   TA,9                ; Initialize TA and RA register
        ADD    RA,2        ;
        CALL   AIC_2ND     ;
        ;-----------------
        LDP    #TB
        LACC   TB,9                ; Initialize TB and RB register
        ADD    RB,2        ;
        ADD    #02h        ;
        CALL   AIC_2ND     ;
        ;-----------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2           ; Initialize control register
        ADD    #03h        ;
        CALL   AIC_2ND     ;
        RET                ;

AIC_2ND:
        LDP    #0                  ; Data page point is 0  (MM regs)
        SACH   DXR                 ; send ACC Hi 00
        CLRC   INTM                ; enable interrupts
        IDLE                       ; wait for interrupt
        ADD    #6h,15              ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH   DXR                 ; send ACC Hi to initiate secondary protocol
        IDLE                       ; wait for interrupt
        SACL   DXR                 ; send the T register data
        IDLE                       ; wait for interrupt
```

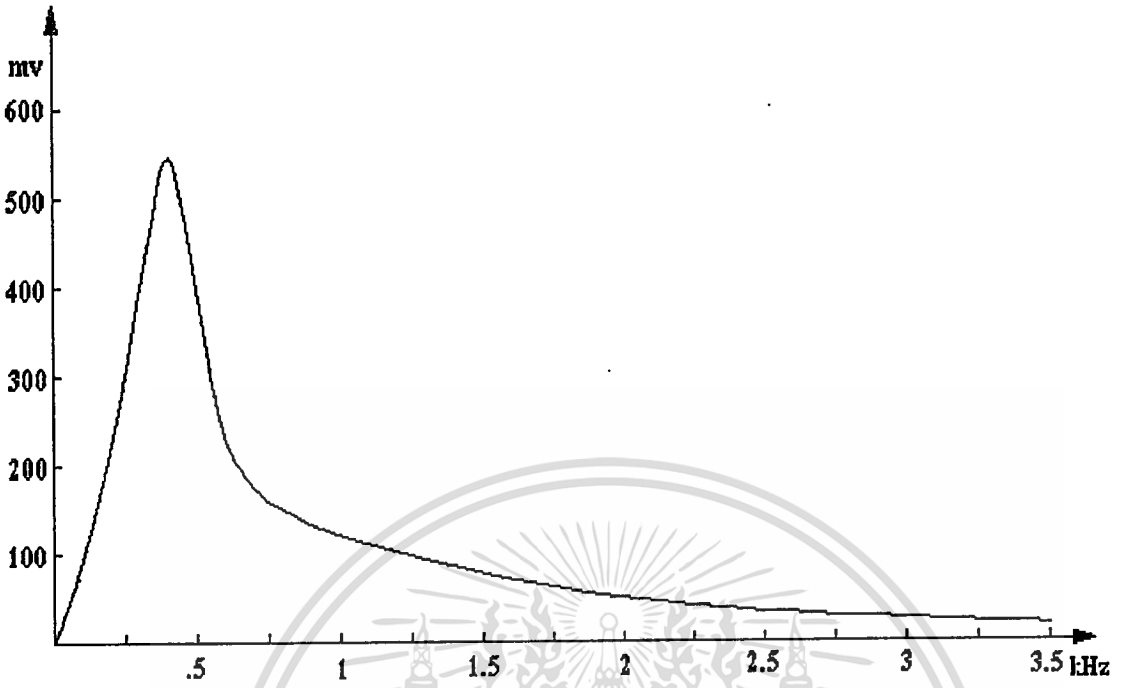
```
LACL   #0                    ; clear ACCUMULATOR LOW
SACL   DXR                   ; send another to make sure 1st word got sent
IDLE                         ; wait for interrupt
SETC   INTM
RET                  ;
.END
```

จากการรันโปรแกรมข้างต้นแล้วทำการป้อนสัญญาณ Input ความถี่ต่างๆ แล้วทำการวัดค่าการตอบสนองทาง Amplitude ที่ความถี่ต่างๆ ดังตาราง

Frequency	Volt Output
50 Hz	26 mV
100 Hz	86 mV
200 Hz	158 mV
250 Hz	283 mV
300 Hz	395 mV
330 Hz	472 mV
360 Hz	544 mV
390 Hz	560 mV
420 Hz	544 mV
450 Hz	496 mV
515 Hz	395 mV
600 Hz	296 mV
650 Hz	256 mV
700 Hz	224 mV
800 Hz	180 mV
900 Hz	156 mV
1 kHz	132 mV
2 kHz	60 mV
3 kHz	36 mV
4 kHz	8 mV
5 kHz	4 mV

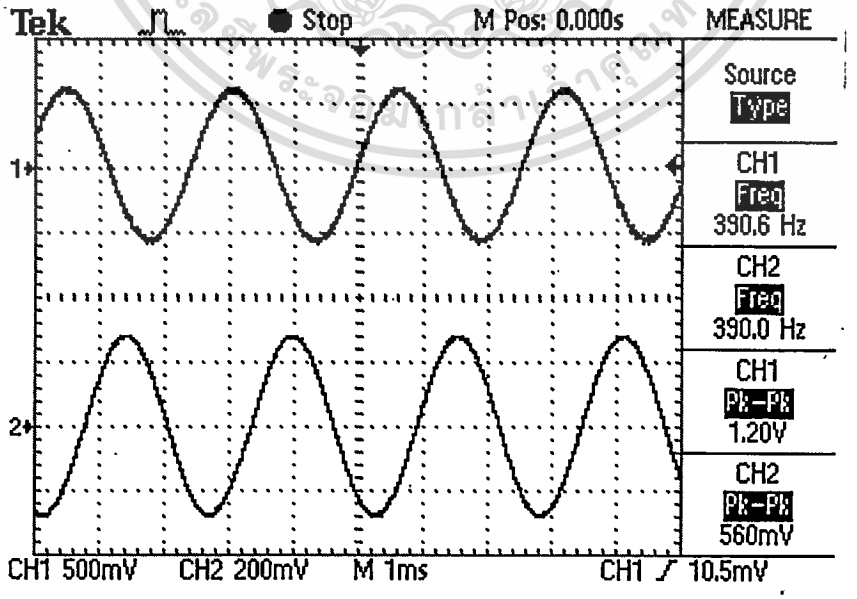
ตารางที่ 4.2 การตอบสนองความถี่ของ IIR Band pass Filter

ซึ่งสามารถ Plot เป็นกราฟได้ดังนี้

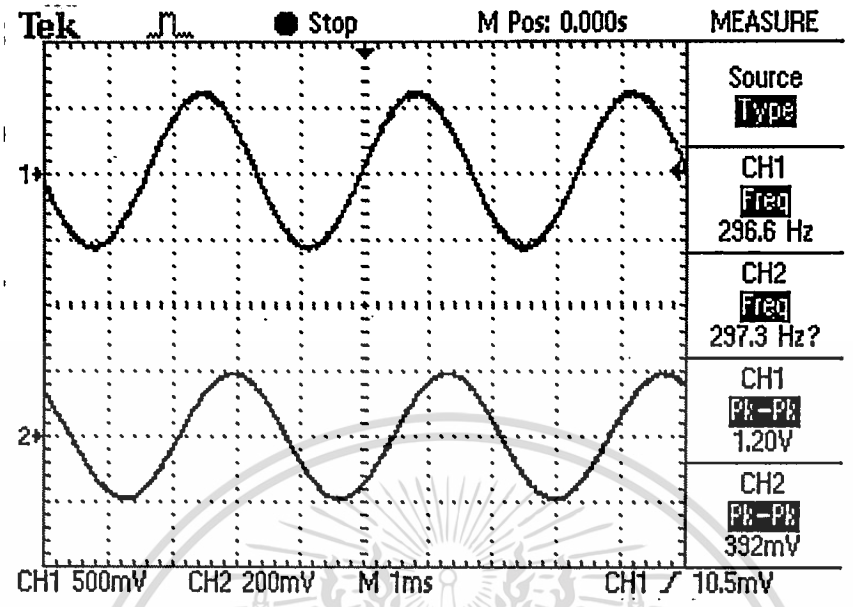


รูปที่ 4.4 กราฟแสดงการตอบสนองความถี่ของ IIR Band pass Filter

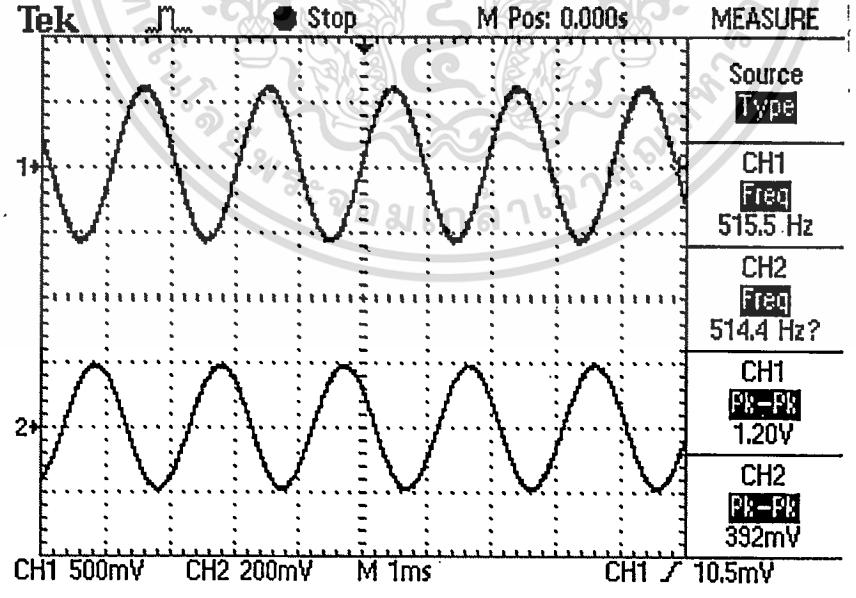
จากกราฟจะเห็นว่าการตอบสนองความถี่เป็นไปตามที่คำนวณไว้คือค่า Amplitude ที่ความถี่ 390 Hz มีค่ามากที่สุด โดยรูปคลื่นที่ความกลางและความถี่ Cutoff ทั้งสองด้านสามารถแสดงดังนี้



รูปที่ 4.5 การตอบสนองที่ความถี่กลาง ของ Band pass IIR filter



รูปที่ 4.6 การตอบสนองที่ความถี่ Cutoff ด้านต่ำ



รูปที่ 4.7 การตอบสนองที่ความถี่ Cutoff ด้านสูง

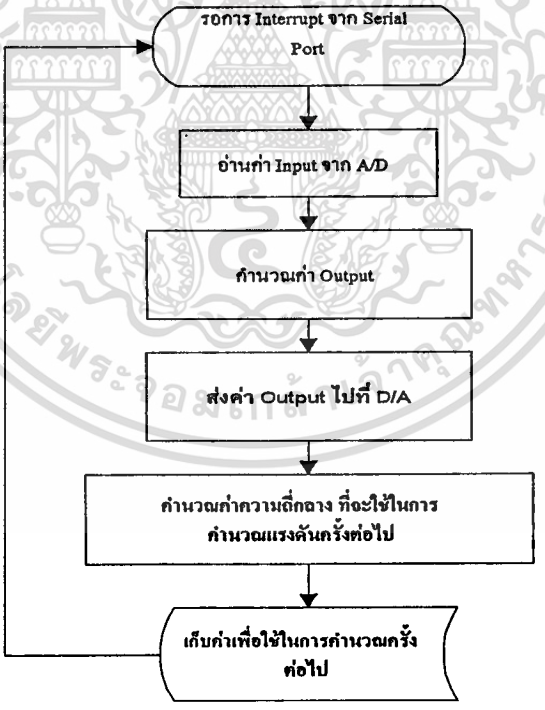
จะเห็นว่าที่ความถี่ความถี่กลางการตอบสนองทาง Phase ของ Filter นี้จะไม่ตรงกับทางทฤษฎี เพราะว่ามีผลตอบสนอง ของ Low pass Filter ที่ภาคสุดท้ายของ D/A มาเกี่ยวข้องด้วย ซึ่งจริงๆ ที่ความถี่กลาง จะไม่มีความเพี้ยนทาง Phase เกิดขึ้น

4.3 Adaptive Digital IIR Filter

ส่วน ของ Adaptive Digital IIR Filter จะทำการเขียนโปรแกรมเพื่อดูผลการตอบสนองต่อการเปลี่ยนแปลงความถี่ของสัญญาณ Input (การเปลี่ยนค่า $\alpha_1(k)$) ว่าจะตามสัญญาณ Input อย่างไร รวมทั้งการแยกสัญญาณ Sine wave ที่เป็นความถี่หลักออกจากสัญญาณรบกวน

4.3.1 การโปรแกรม

ส่วนของการโปรแกรมจะเหมือนกับของ IIR Filter ในส่วนแรก แต่จะมีส่วนของการ Update ค่าความถี่กลางของ Filter ในกรณีที่สัญญาณ Input มีการเปลี่ยนแปลงความถี่เพิ่มขึ้นมาคือการคำนวณค่าในส่วนของการ $\alpha_1(k+1)$ และ $\psi(k)$ โดยสามารถเขียน Flowchart ขั้นตอนการคำนวณของโปรแกรมได้ดังนี้



โดยจากสมการทาง Time Domain ของ Digital IIR Filter และสมการการ Update ค่าความถี่กลางที่ได้ทำการ Simulation ดูการตอบสนองค่าต่างๆ กับ Matlab มาแล้วในตอนแรกคือ

$$y(k)=\left[\frac{1-\alpha_0}{2}\right].[x(k)-x(k-2)]+y(k-1)\alpha_1(k).(1+\alpha_0)-\alpha_0 y(k-2)$$

$$\alpha_1(k+1)=\alpha_1(k)+\mu.y(k).\psi(k)$$

$$\psi(k)=(1+\alpha_0).\alpha_1(k).\psi(k-1)+(1+\alpha_0).y(k-1)-\alpha_0\psi(k-2) \qquad (4.2)$$

ในขั้นนี้เราจะทำการเขียนโปรแกรมลงบน Chip DSP เพื่อให้มันทำการแยกเอาสัญญาณ Sine Wave ที่มี Noise โดยมี SNR เท่ากับ 10 dB เหมือนกับที่ทำการ Simulate มารวมทั้งดูการเปลี่ยนแปลงความถี่กลางตามความถี่ Input ว่ามีการติดตามสัญญาณที่ความถี่ต่างๆ อย่างไรบ้าง

การกำหนดค่าต่างๆ

ในขั้นตอนนี้เราจะกำหนดค่าความถี่ของสัญญาณ Sampling เท่ากับ 10.08 KHz ค่า Q-Factor (α_0) เท่ากับ 0.7 (เหตุที่กำหนดมากไม่ได้เพราะจะทำให้การติดตามการเปลี่ยนแปลงความถี่ของสัญญาณ Input ทำได้ช้าแต่ถ้ากำหนดค่า Q น้อยเกินไปก็จะทำให้ความถี่ที่เราไม่ต้องการออกมามาก) ส่วนค่าของความถี่กลาง $\alpha_1(k)$ จะเริ่มต้นที่ค่า 0 คือค่าความถี่ที่ครึ่งหนึ่งของ Nyqust Rate (ทำให้ได้ความถี่ใน Time Domain ประมาณเท่ากับ 2.5 KHz จากสมการที่ 2.8) ส่วนค่า Step Size Parameter (μ) กำหนดไว้ที่ 0.0003 ดังนั้นสามารถเขียนสมการที่ (4.2) ใหม่ได้เป็น

$$y(k)=0.15[x(k)-x(k-2)]+0.y(k-1)-0.7y(k-2)$$

$$\alpha_1(k+1)=\alpha_1(k)+.0003.y(k).\psi\ (k)$$

$$(k)=0.\psi(k-1)+1.7y(k-1)-.7\psi(k-2) \qquad (4.3)$$

จะเห็นว่ามีเทอมที่เป็นศูนย์อยู่คือเทอมที่มีการคูณกับ α_1 แต่พอมันทำการ Update ค่า α_1 เพื่อติดตามสัญญาณ Input มันก็จะมีค่าขึ้นมาเอง

ซึ่งสามารถคำนวณค่าคงที่ต่างๆ โดยอ้างอิงกับ A/D และ D/A Converter ขนาด 14 bit ซึ่งได้อธิบายไปแล้วในหัวข้อ 4.2 ที่ผ่านมา

หลังจากนั้นเราจะทำการเขียนโปรแกรมโดยสามารถแสดงได้ดังนี้

```
;                    ADAPTIVE IIR  FILTER
;-----------------------------------------------------------------

        .MMREGS
        .ds     0f00h
TA      .word   16      ;
```

```
RA      .word   16      ; This set up of AIC registers give
                        ; a sampling freq of 10.08 KHz
                        ;
TB      .word   31      ;
RB      .word   31      ;
AIC_CTR .word   08h
OUTPUT  .word   0
temp1   .word   0
temp2   .word   0

;Filter Coefficient Generator  bandpass

mue     .word    5  ;  .0003
alp1k   .word    0  ;        (start only)
a2      .word   -2458 ;  -0.15  -(1-alp0)/2
a1      .word    0  ;   0
a0      .word    2458 ;  0.15   (1-alp0)/2
b2      .word   -11469 ;  -.7
b1      .word    0  ;   0     (start only)
d       .word    27853 ;   1.7    (1+alp0)

;Storage deley line

YN1     .word   0
YN2     .word   0
XN      .word   0,0
XNL     .word   0
BUF     .word   0
YNx     .word   0
PHI1    .word   0
PHI2    .word   0

        .ps     0080ah
```

```
rint:  B     RECEIVE
xint:  B     TRANSMIT

       .ps    0a00h
       .entry          ;
       ;-----------------------------------------
       SETC   INTM
       LDP    #0
       OPL    #0834h,PMST
       LACC   #0
       SAMM   CWSR
       SAMM   PDWSR
       SETC   SXM        ; SXM MUST BE SET
       SPLK   #022h,IMR  ; This turns on receive interrupt only

       CALL   AICINIT    ;
       SPLK   #12h,IMR
       CLRC   OVM
       SPM    0          ;
       CLRC   INTM       ;

       LAR    AR5,#alp1k    ; calculate new cof for next time (b1)
       LAR    AR3,#mue
       LAR    AR4,#temp2    ; AR4 point to address off temp2
       LAR    AR7,#alp1k

WAIT:                ; a main program would go here
       B     WAIT    ;

RECEIVE:
       LDP    #XN
       CLRC   INTM      ; ENABLE INTERRUPTS FOR DEBUGGING DSK PURPOSES
       LAMM   DRR       ; load accumulator with word received from AIC
```

```
        SACL    XN              ; store the value of received word to a variable
        LAR     AR0,#XNL        ; load AR0 with address of last delay element
        ZAP                     ; ZERO ACC AND PRODUCT REGISTERS
        MAR     *,AR0           ; AR0 is the current AR register.

        RPT     #4              ; Repeat the next instruction 4 times through
        MACD    #a2,*-          ; the complete coeff table.
        APAC                    ; Accumulate last product
        SACH    OUTPUT,2        ; ACC -> OUTPUT.  Get rid of extra sign bit.
        LACC    OUTPUT
;       SACL    YN1             ;Comment out this line and two line next
;       MAR     *,AR7           ; if  want  to  see  output signal
;       LACC    *
        AND     #0fffch         ; Two LSB's must be zero for AIC
        SAMM    DXR             ; write output word to transmit register

        LACC    YN2             ; Adaptation section
        SACL    YNx
        LAR     AR1,#PHI2       ; AR1 point to add off PHI2
        MAR     *,AR1           ; AR1 is current auxiliary register
        ZAP                     ; Zero Acc&Product register
        RPT     #2              ; Repeat next instruction 2 time
        MACD    #b2,*-          ; Calculate PHI1
        APAC
        SACH    PHI1,2          ; store PHI1
        LACC    PHI1
        AND     #0f000h         ;  Zero 4 bit low  off PHI17
        SACL    PHI1
        LAR     AR2,#YN1        ; calculate alp1(k+1)
        MAR     *,AR2           ; AR2 is current AR register
        ZAP
        MAC     #PHI1,*,AR3     ; yn*PHI1
        APAC
        SACH    temp1,2
```

```
        ZAP
        MAC     #temp1,*        ; Calculate Alp1(k)
        APAC                    ; (yn*PHI1)*mue
        SACH    temp2,2
        LACC    alp1k
        MAR     *,AR4
        ADD     *,0
        SACL    alp1k
        MAR     *,AR5           ;
        ZAP
        MAC     #d,*            ; alp1(1+alp0)
        APAC
        SACH    b1,2
        RETE

TRANSMIT:
        RETE
;
AICINIT: SPLK   #20h,TCR        ; To generate 10 MHz from Tout
        SPLK    #01h,PRD        ; for AIC master clock
        MAR     *,AR0
        LACC    #0008h          ; Non continuous mode
        SACL    SPC             ; FSX as input
        LACC    #00c8h          ; 16 bit words
        SACL    SPC
        LACC    #080h           ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; and taking it high after 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG
        ;---------------
        LDP     #TA             ;
```

```
        SETC   SXM          ;
        LACC   TA,9         ; Initialize TA and RA register
        ADD    RA,2         ;
        CALL   AIC_2ND      ;
        ;--------------------
        LDP    #TB
        LACC   TB,9         ; Initialize TB and RB register
        ADD    RB,2         ;
        ADD    #02h         ;
        CALL   AIC_2ND      ;
        ;--------------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2    ; Initialize control register
        ADD    #03h         ;
        CALL   AIC_2ND      ;
        RET                 ;

AIC_2ND:
        LDP    #0           ; Data page point is 0  (MM regs)
        SACH   DXR          ; send ACChi 00
        CLRC   INTM         ; enable interrupts
        IDLE                ; wait for interrupt
        ADD    #6h,15       ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH   DXR          ; send ACChi to initiate secondary protocol
        IDLE                ; wait for interrupt
        SACL   DXR          ; send the T register data
        IDLE                ; wait for interrupt
        LACL   #0           ; clear ACClo
        SACL   DXR          ; send another to make sure 1st word got sent
        IDLE                ; wait for interrupt
        SETC   INTM
        RET                 ;
        .END
```

4.3.2 ผลการตอบสนองสัญญาณในลักษณะต่าง ๆ ของ Adaptive IIR Filter

ในขั้นนี้เราจะทดลองการเปลี่ยนแปลงของสัญญาณ Input 2 แบบคือ สัญญาณ Sine Wave (ความถี่หลัก) บวกกับสัญญาณรบกวนที่เป็น Sine Wave ความถี่สูงและอีกอันหนึ่งจะบวกกับสัญญาณ Noise แบบสุ่ม

ส่วนการพิจารณาทาง Output จะดูลักษณะการขจัดสัญญาณรบกวนแบบต่างๆ และที่สำคัญคือการดูการติดตามการเปลี่ยนแปลงความถี่ของสัญญาณ Input การรักษาเสถียรภาพของการเปลี่ยนแปลงติดตามความถี่กลาง

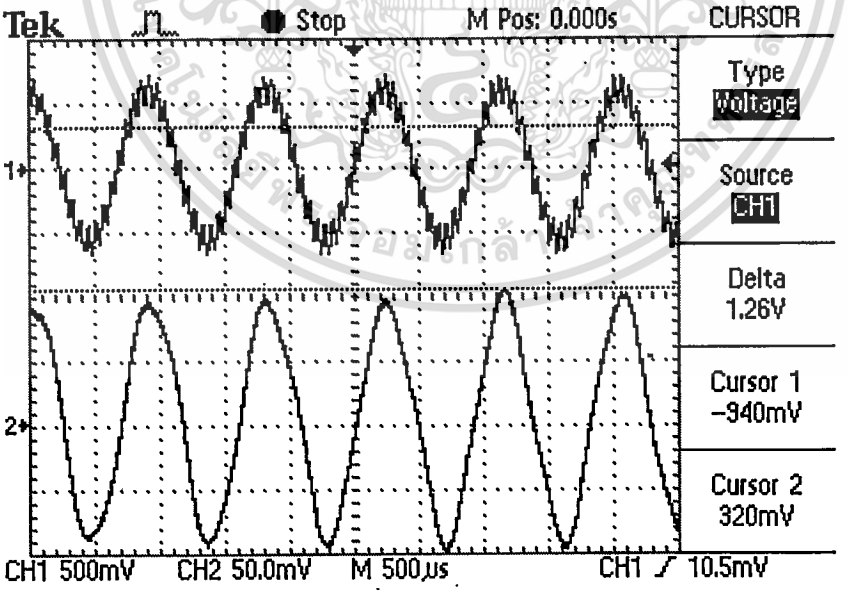
Note ; เครื่องมือที่ใช้ในการบันทึกผลการทดลองคราวนี้ คือ

- อุปกรณ์การแสดงบันทึกผลคือ Digital Oscilloscope Tektronix TDS 220 สามารถ Print รูปคลื่นที่จอออกมาได้เลย

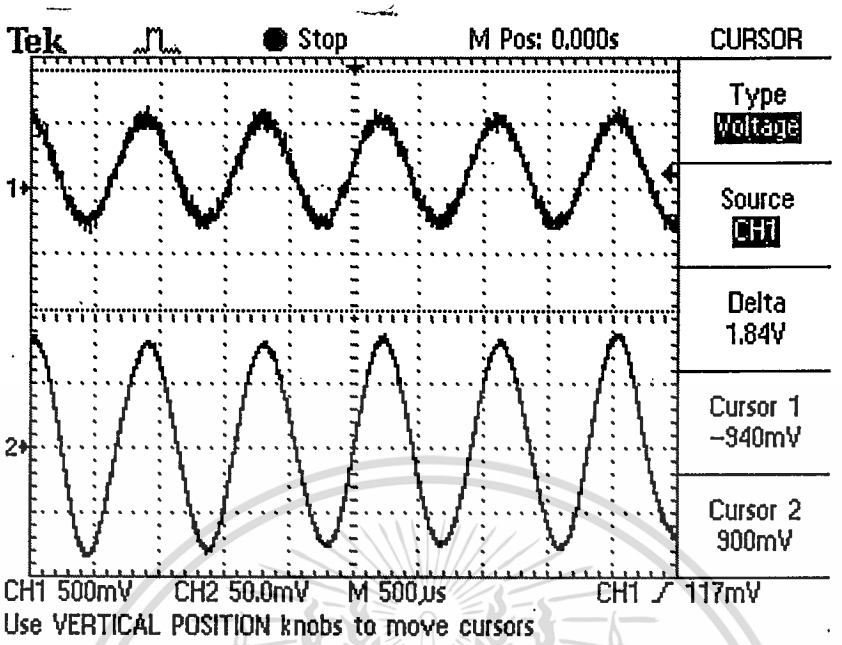
-อุปกรณ์กำเนิดสัญญาณทดสอบและสัญญาณรบกวนทั้งสองแบบคือ Function Generator Hewlett Packard รุ่น 33120A

ผลการทดลอง

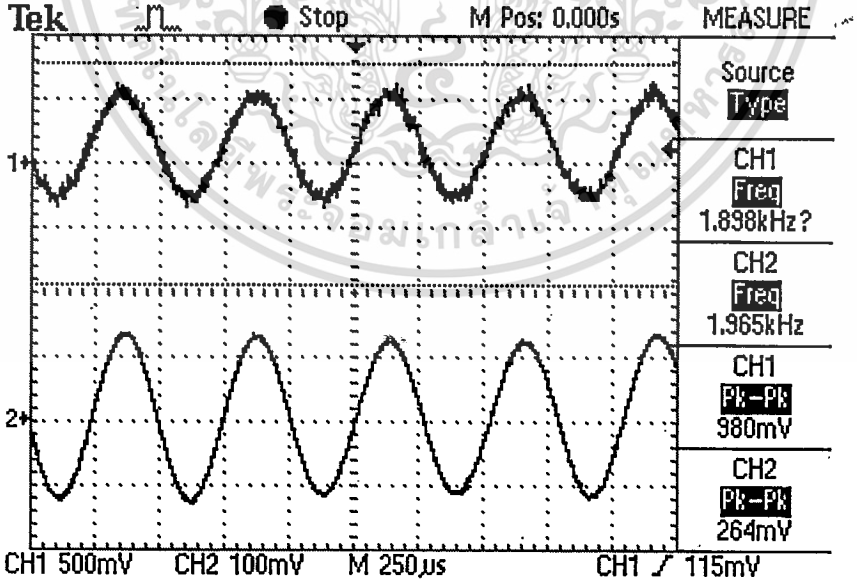
สามารถบันทึกค่าต่างๆ ออกมาดังรูปต่อไปนี้



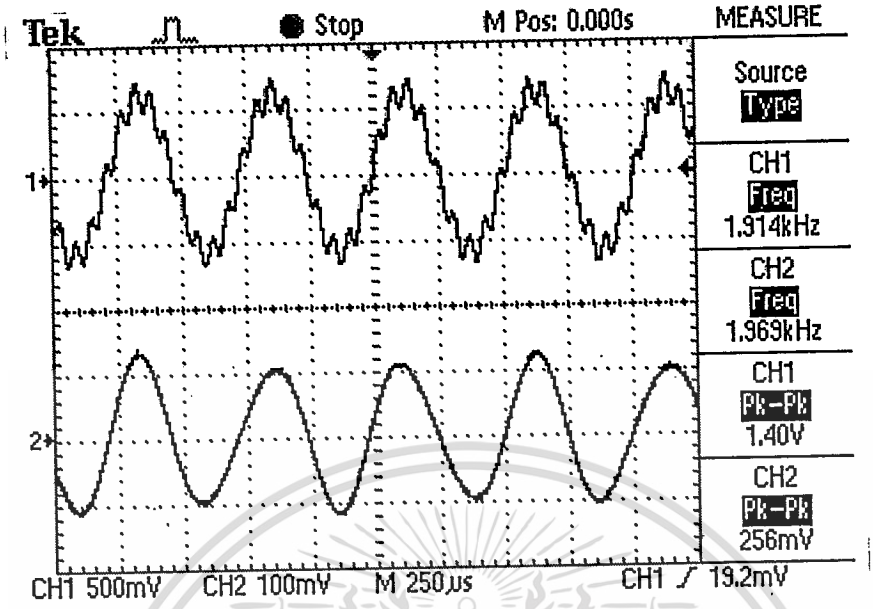
รูปที่ 4.8 เมื่อสัญญาณ Input เป็น sin 1.1k+.22sin (17k)



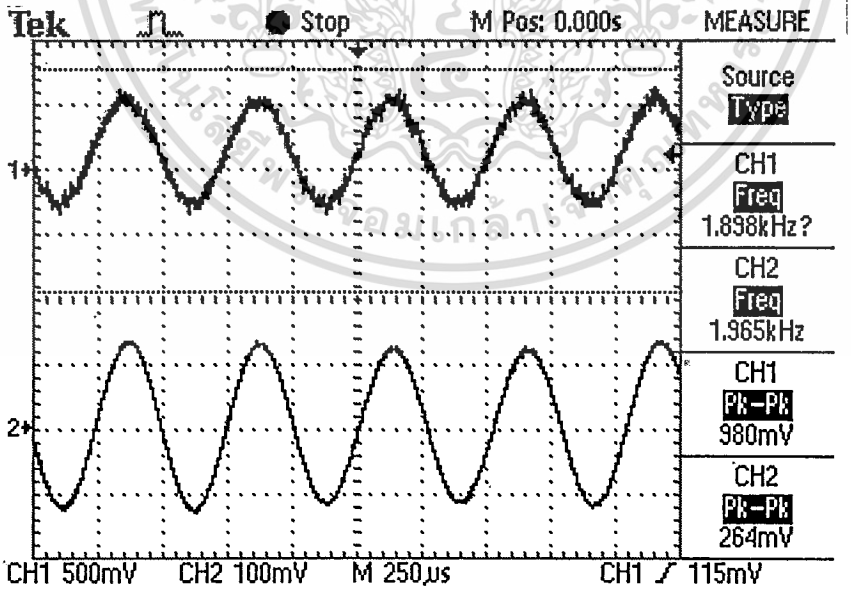
รูปที่ 4.9 เมื่อสัญญาณ Input เป็น sin 1.1k+.22random noise



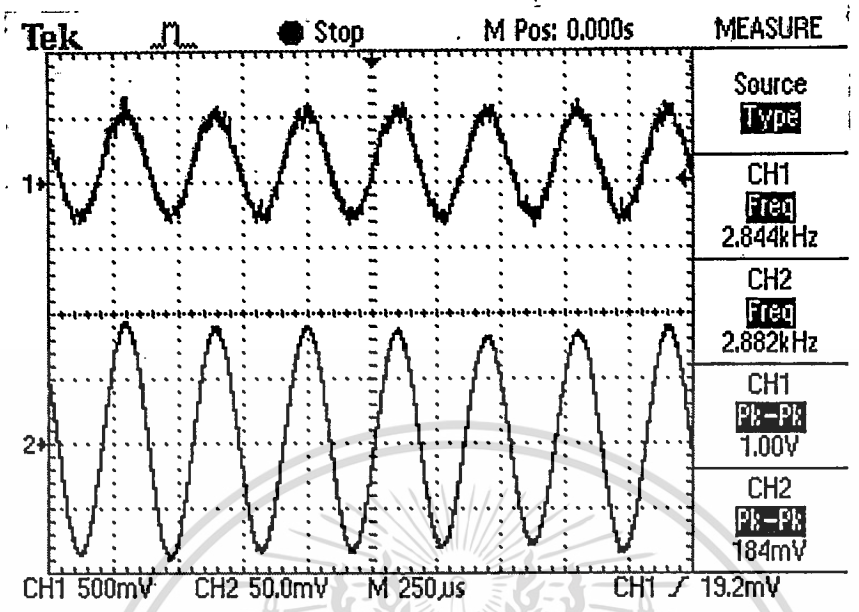
รูปที่ 4.10 เมื่อสัญญาณ Input เป็น sin 1.95k+.22random noise



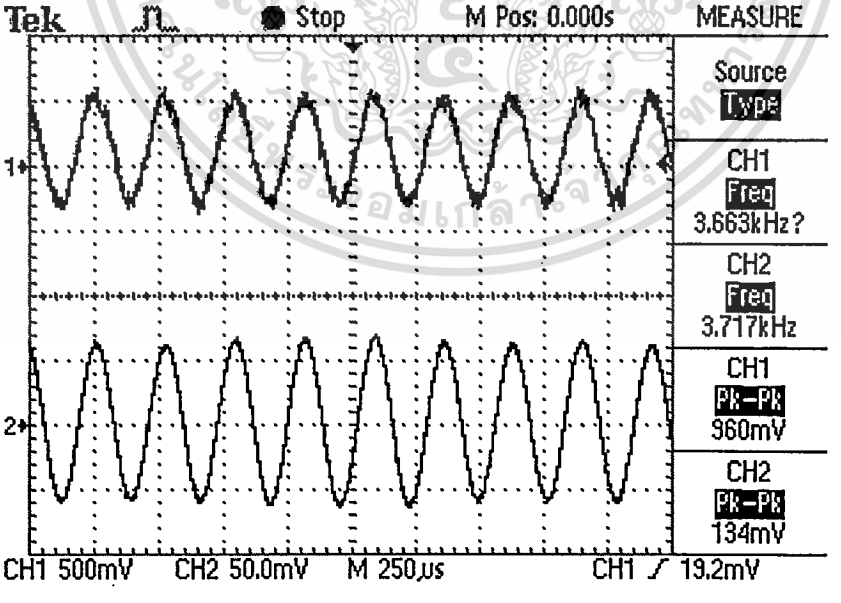
รูปที่ 4.11 เมื่อสัญญาณ Input เป็น sin 1.95k+.22sin (17k)



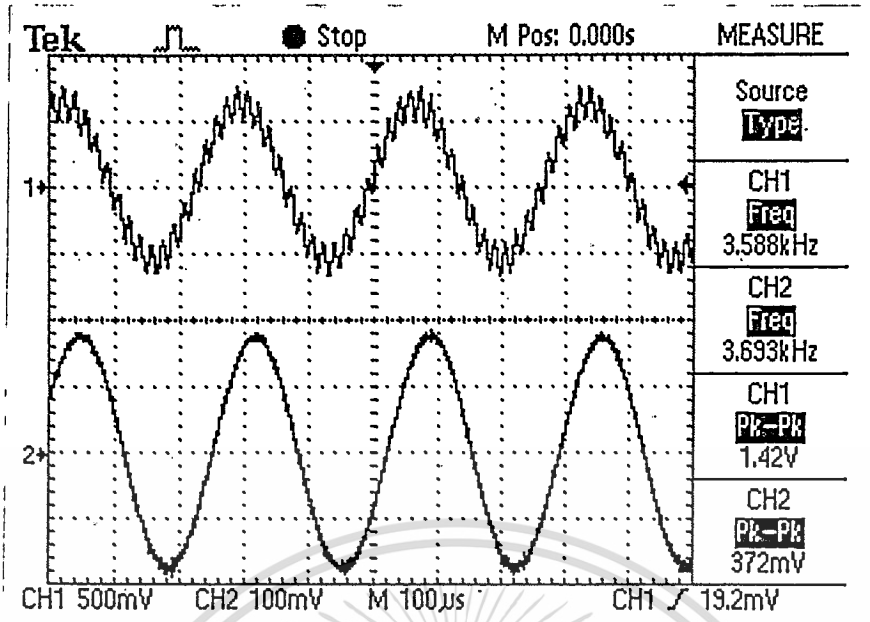
รูปที่ 4.12 เมื่อสัญญาณ Input เป็น sin 2.8k+.22sin(25k)



รูปที่ 4.13 เมื่อสัญญาณ Input เป็น sin 2.8k+.22random noise



รูปที่ 4.14 เมื่อสัญญาณ Input เป็น sin 3.7k+.22random noise



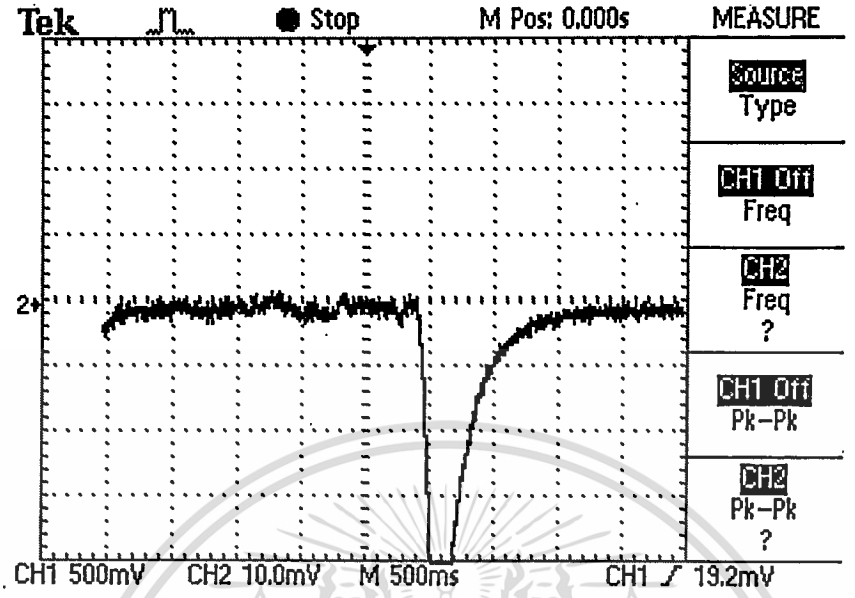
รูปที่ 4.15 เมื่อสัญญาณ Input เป็น sin 3.7k+.22sin(25k)

จากรูปผลการแยก Noise ออกจากสัญญาณ Sine จะเห็นได้ว่าสามารถกำจัด Noise ได้ในระดับที่ดีมากไม่ว่าจะเป็น Noise แบบ Uniform หรือ Random Noise ตั้งแต่ความถี่ในช่วง 600 Hz จนถึงประมาณ 3.8 KHz.

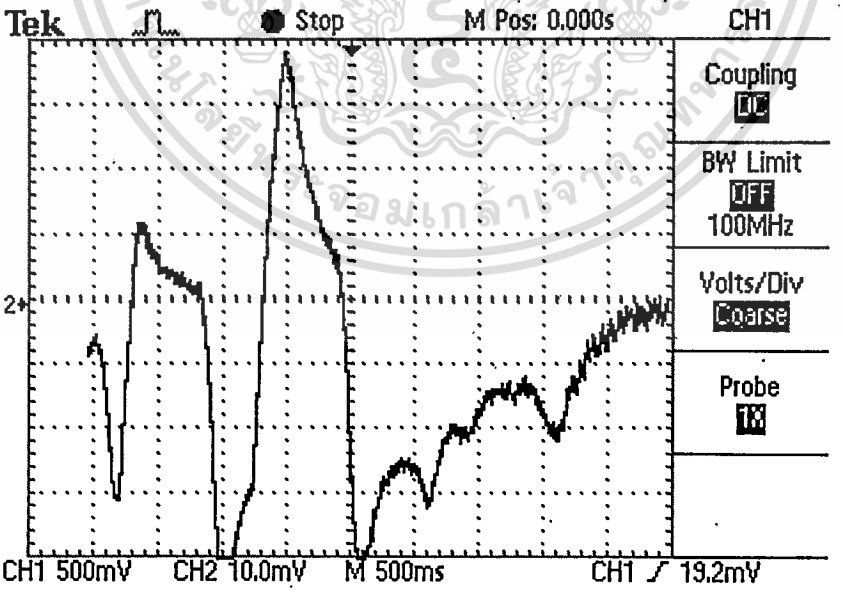
4.3.3 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ตามสัญญาณ Input ลักษณะต่างๆ

ในขั้นตอนนี้จะดูการเปลี่ยนแปลงของค่า $\alpha_1(k)$ (ค่าความถี่กลางของ Filter) ตามสัญญาณ Input โดยในส่วนของโปรแกรมแทนที่จะส่งสัญญาณรูปคลื่น Output ออกมาแสดงที่ Scope เราจะส่งสัญญาณ $\alpha_1(k)$ ออกมาทาง Output เพื่อดูการตอบสนองที่ Scope โดยการปรับ Time ที่ Scope แบบช้าๆ (500ms) โดยในการทดลองเราทำการเลื่อนความถี่แบบ manual ด้วยมือที่ Function Generator เพื่อดูการเปลี่ยนแปลง $\alpha_1(k)$ โดยสามารถแสดงดังนี้

Note: การดูการเปลี่ยนแปลง ค่า $\alpha_1(k)$ จะทำโดยการเอาเครื่องหมาย ; (Semi-colon) หน้าบรรทัดที่ 13,14,15 ในส่วน Routine Receive ออก



รูปที่ 4.16 การลดลงของค่า $\alpha_1(k)$ เมื่อเพิ่มความถี่ Input

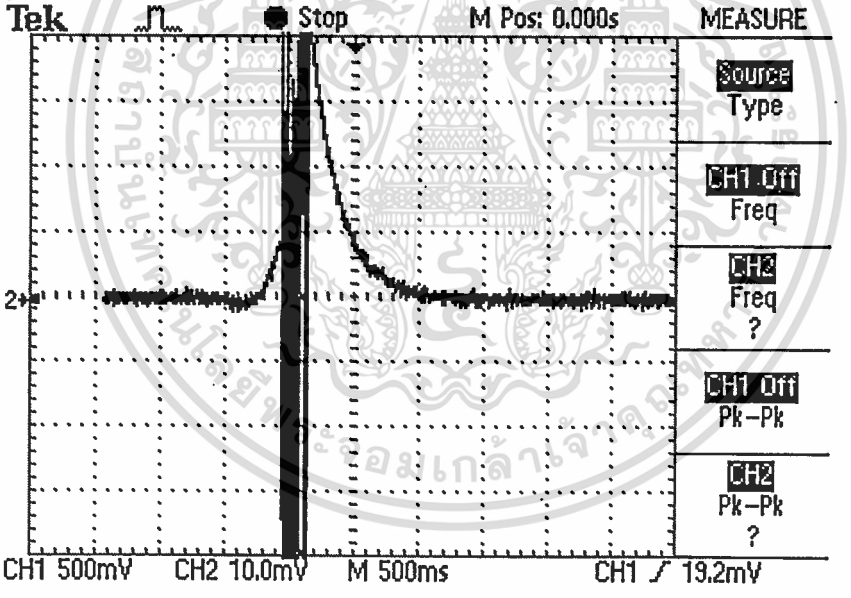


รูปที่ 4.17 การเพิ่มและลดลงของค่า $\alpha_1(k)$ เมื่อลดและเพิ่มความถี่ Input

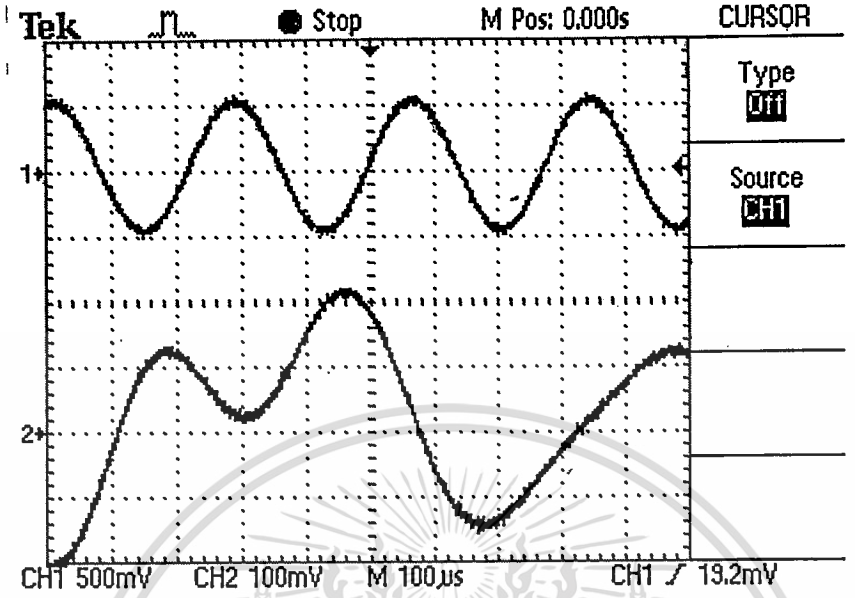
จากรูปแรกจะเห็นการลดลงของค่า $\alpha_1(k)$ เมื่อถึงจุดที่ไม่มีการเปลี่ยนแปลง (จริงๆ ยังมีการเปลี่ยนแปลงอยู่เล็กน้อย) จะลดลงสู่ค่า 0 เพราะใน D/A ไม่สามารถจับสัญญาณการเปลี่ยนแปลงของระดับแรงดัน DC เล็กน้อยได้

4.3.4 ความไม่เสถียรภาพ

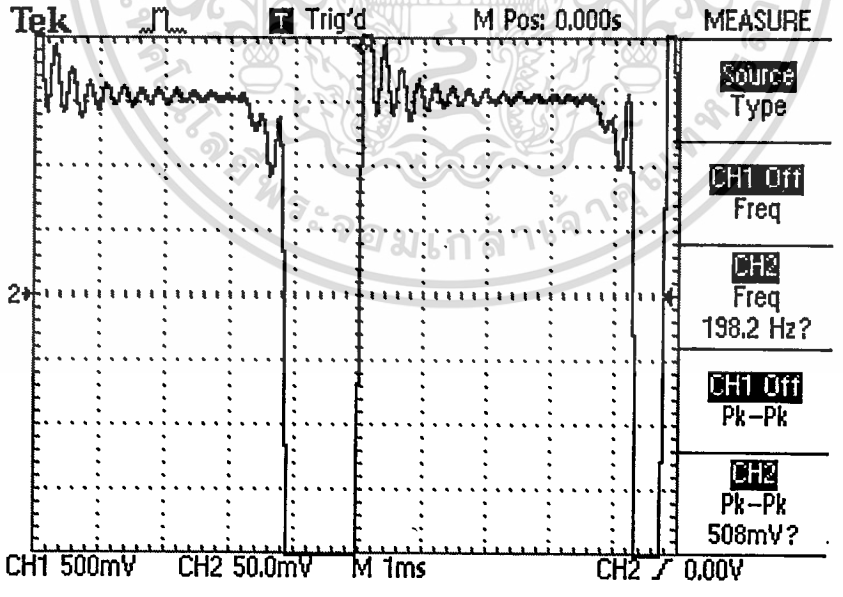
ในขั้นตอนแรกที่ทำการทดลองดูค่า $\alpha_1(k)$ เมื่อเราทำการเปลี่ยนแปลงค่า $\alpha_1(k)$ และเมื่อเราทำการเปลี่ยนค่าความถี่สัญญาณ Input อย่างรวดเร็วจะเกิดการไม่เสถียรภาพขึ้นเพราะเทอมของ $\psi(k)$ ที่เป็นอัตราส่วนการเปลี่ยนแปลงค่า $y(k)$ เทียบกับ $\alpha_1(k)$ มีค่ามากเกินไปทำให้ค่า $\alpha_1(k+1)$ มีค่ามากเกินไป ซึ่งค่าใน memory จะเพิ่มจนค่าใน memory เพิ่มขึ้นจนเป็น 0FFFFh จะทำให้กลายเป็นค่าลบ (เพราะการเก็บค่าลบในที่นี้จะเก็บเป็นรูป Two-Complement คือ bit แรกๆ จะเก็บเป็น 1 ซึ่งทำให้ค่าสายอยู่ระหว่างค่าบวกและค่าลบจนมีการเปลี่ยนแปลงค่าความถี่ Input อีกครั้งจึงสามารถปรับสู่ค่าที่ถูกต้องได้อีกครั้งดังสามารถแสดงดังรูปที่ 4.18



รูปที่ 4.18 แสดงการไม่เสถียรภาพแล้วกลับสู่เสถียรภาพของ $\alpha_1(k)$



รูปที่ 4.19 แสดงรูปคลื่น Output ขณะที่ไม่เสถียรภาพ



รูปที่ 4.20 การแสดงค่า $\alpha_1(k)$ ขณะที่ไม่เสถียรภาพ (ที่ Scale Time ค่าน้อยๆ)

กรณีนี้สามารถแก้ไขได้โดยการตัด bit 4 bit ล่างของเทอม $\psi(k)$ทิ้ง (ในโปรแกรมบรรทัดที่ 28 ใน Routine Receive) ทำให้การสามารถรักษาเสถียรภาพของระบบดีขึ้นแต่การติดตามสัญญาณ Input จะช้าลงนิดหน่อยรวมทั้งการแกว่งของสัญญาณ Output (ความถี่) จะลดลง

4.3.5 การดูการเปลี่ยนแปลงค่าของ $\alpha_1(k)$ ที่เงื่อนไขเริ่มแรก (เหมือนตอน Simulation)

เนื่องจากการดูการติดตามความถี่ของสัญญาณ Input ที่มีการเปลี่ยนแปลงค่า $\alpha_1(k)$ ในช่วงแรกที่ระบบเริ่มทำงานนั้นไม่สามารถที่จะทำได้เพราะค่าจะมีการเปลี่ยนแปลงเร็วมาก (ความจริงจะดูเห็นแต่หาค่าเป็นรูปคลื่นออกมาไม่ได้ซึ่งจะมีการเปลี่ยนแปลงช่วง DSP เริ่มรันโปรแกรมครั้งเดียวจึงไม่สามารถบันทึกผลและนำมาคำนวณค่าเวลาในการติดตามสัญญาณได้ การที่จะสามารถดูได้ต้องทำการเขียนโปรแกรมในการเก็บค่า $\alpha_1(k)$ ไว้ใน Data Memory เพื่อนำมาทำการแสดงออกที่ Scope เป็นรูปคลื่นเพื่อเป็นการสะดวกที่จะนำมาทำการอ้างหรือทำการคำนวณค่าต่างๆ แต่เนื่องจากการที่จะเก็บค่าของ $\alpha_1(k)$ ทุกๆ ค่าที่ทำการคำนวณนั้นจะเป็นการสิ้นเปลือง Memory มากคือจริงๆ แล้ว Memory ไม่พอหรือถ้า Memory พอก็จะเป็นการยากที่จะแสดงออกมาที่ Scope เพราะความถี่ของมันจะออกมาต่ำมากจน Scope ที่มีไม่สามารถจับรูปคลื่นของมันได้ดังนั้นจึงต้องมีการข้ามการเก็บค่าของ $\alpha_1(k)$

โดยค่าความถี่ที่เหมาะสมในการแสดงที่ Scope คือความถี่ 20 Hz (จากการทดลอง) ซึ่งจากอัตราการ Sampling ที่ 10 KHz จะได้ค่า Time = .1ms และที่ความถี่ 20 Hz จะได้ค่า Time = 50ms ดังนั้นจะทำให้ความถี่ Output ได้ 20 Hz ที่ความถี่การ Sampling 10 KHz ต้องการค่าที่จะส่งให้ D/A เท่ากับ 50/0.1 = 500 ค่า เพราะฉะนั้นในการเก็บค่า $\alpha_1(k)$ เอาไว้ 500 ค่า แต่ในโปรแกรมจะมี 5 ค่าแรกที่ได้ทำการกำหนดไว้ให้เป็น 0 เพื่อให้รูปคลื่นสามารถพิจารณาง่ายจะเป็นค่าของ $\alpha_1(k)$ จริงๆ 495 ค่า ส่วนการที่จะข้ามทีละกี่ค่านั่นจะแล้วแต่กรณีที่เราทำการพิจารณา สามารถกำหนดและเปลี่ยนค่าได้ในโปรแกรม โดยในโปรแกรมจะแสดงรูปคลื่น Output ของสัญญาณก่อนในช่วงแรกของการทำงานคือขณะที่กำลังเก็บค่า $\alpha_1(k)$ นั้นจะสามารถแสดงรูปสัญญาณ Output ไปด้วยเพื่อดูรูปคลื่นว่ามีความเสถียรภาพหรือไม่อย่างไรโดยค่าเวลาของการแสดงสามารถกำหนดได้ในโปรแกรมไม่เกี่ยวกับส่วนของการเก็บค่า $\alpha_1(k)$

โดยโปรแกรมสามารถแสดงดังนี้

```
;                    ADAPTIVE IIR  FILTER
;    To generate  alp1(k)  signal  to shows in analog scope  use
;    memory  1024 byte to store data alp1(k)
;-------------------------------------------------------

      .MMREGS
      .ds   1200h
TA    .word  16      ;
```

```
RA      .word   16      ; This set up of AIC registers give
                        ; a sampling freq of 10.08 KHz
TB      .word   31      ;
RB      .word   31      ;
AIC_CTR .word   08h
OUTPUT  .word   0
temp1   .word   0
temp2   .word   0
temp3   .word   0

mb      .word   10      ; Mark if calculate  output = (1+rp)*del
mb2     .word   10      ; Mark if calculate   alp1k = dif1*alp1dx
rp      .word   2
del     .word   07fffh
dif     .word   0
dif1    .word   45      ;   dif1*alp1dx
alp1dx  .word   495     ;   BYTE  off data memory to use store alp1(k)

; Filter Coefficient Generator  bandpass  IIR  filter

mue     .word   5               ; .0003
alp1k   .word   0               ;  (start only)
a2      .word   -2458           ; -0.15  -(1-alp0)/2
a1      .word   0               ; 0
a0      .word   2458            ;  0.15   (1-alp0)/2
b2      .word   -11469          ; -.7
b1      .word   0               ; 0     (start only)
d       .word   27853           ; 1.7   (1+alp0)

;Storage deley line
YN1     .word   0
YN2     .word   0
XN      .word   0,0
```

```
XNL     .word   0
BUF     .word   0
YNx     .word   0
PHI1    .word   0
PHI2    .word   0
BUF2    .word   0
alp100  .word   0,0,0,0,0          ; 0 of alp1 5 start value(display)
alp1d   .word   0                  ; alp1k data initial address
;       .include "ar.asm"          ;Clear Data memory to store Alp1(k)

        .ps     0080ah
rint:   B       RECEIVE
xint:   B       TRANSMIT

        .ps     0990h
        .entry
        ;-------------------------------------------------
        SETC    INTM
        LDP     #0
        OPL     #0834h,PMST
        LACC    #0
        SAMM    CWSR            ; Software wait state generator register
        SAMM    PDWSR
        SETC    SXM             ; SXM MUST BE SET
        SPLK    #022h,IMR       ; This turns on receive interrupt only

        CALL    AICINIT
        SPLK    #12h,IMR
        CLRC    OVM
        SPM     0
        CLRC    INTM
        LDP     #alp1d
```

```
        LACC  #00eh          ; AR6 use to specify  by circular buffer1
        SAMM  CBCR           ; Write to circular buffer control register
        LAR   AR7,#temp3
        LAR   AR6,#alp100
        LAR   AR4,#mue
        LAR   AR5,#temp2     ; AR4 point to address off temp2
        LAR   AR3,#YN1
        LAR   AR1,#alp1k
        SAR   AR6,temp1
        LACC  temp1
        SAMM  CBSR1          ; Start  address off  circular buffer1
        LDP   #alp1dx
        ADD   alp1dx
        SAMM  CBER1          ; End   address of  circular buffer1
        SUB   #01h
        SACL  temp3
        LAR   AR6,#alp1d

WAIT:                        ; a main program would go here

        B     WAIT           ;

RECEIVE:
        CLRC  INTM           ; ENABLE INTERRUPTS FOR DEBUGGING DSK PURPOSES
        LDP   #mb2
        LACC  mb2
        SUB   #02h
        BCND  OUTB,GEQ,C
        MAR   *,AR6
        LACC  *+             ; Load alp1k with  increate AR6 for next time
        AND   #0fff0h        ; Two LSB must zero for AIC
        SAMM  DXR
```

```
BR:
      NOP
      RETE
OUTB:                    ; Calculate output
      LDP    #XN
      LAMM   DRR         ; Load accumulator with word received from AIC
      SACL   XN          ; Store the value of received word to a variable
      LAR    AR0,#XNL    ; Load AR0 with address of last delay element
      ZAP                ; ZERO ACC AND PRODUCT REGISTERS
      MAR    *,AR0       ; AR0 is the current AR register.

      RPT    #4          ; Repeat the next instruction 4 times through
      MACD   #a2,*-      ; the complete coeff table.
      APAC               ; Accumulate last product

      SACH   OUTPUT,2    ; ACC -> OUTPUT.  Get rid of extra sign bit.
      LACC   OUTPUT
      SACL   YN1
      AND    #0fffch     ; Two LSB's must be zero for AIC
      SAMM   DXR         ; write output word to transmit register

;*****************************************
; Adaptive signal calculation   section
;*****************************************

      LACC   YN2
      SACL   YNx
      LAR    AR2,#PHI2   ; AR1 point to add off PHI2
      MAR    *,AR2       ; AR1 is current auxiliary register
      ZAP                ; Zero Acc&Product register
      RPT    #2          ; Repeat next instruction 2 time
      MACD   #b2,*-      ; calculate PHI1
      APAC
```

```
        APAC
        SACH    PHI1,2          ; store PHI1
        LACC    PHI1
        AND     #0fff0h         ; Use 4 MSB off PHI1
        SACL    PHI1
        MAR     *,AR3           ; AR3  is current AR register
        ZAP
        MAC     #PHI1,*,AR4     ; yn*PHI1
        APAC                    ;  AR4= address off mue
        SACH    temp1,2
        ZAP
        MAC     #temp1,*,AR5            ; Calculate Alp1(k)
        APAC                    ; (yn*PHI1)*mue=temp2
        SACH    temp2,2
        LACC    alp1k           ; AR5=address off temp2
        ADD     *,0             ; alp1(k+1)
        SACL    alp1k
        MAR     *,AR1
        ZAP
        MAC     d,*             ; alp1*(1+alp0)
        APAC
        SACH    b1,2

        LDP     #del
        LACC    del
        SUB     #01h
        SACL    del
        BCND    RS,NC,LT

BO:
        LDP     #mb
        LACC    mb
        SUB     #02h
```

```
      BCND  BR,NC,LT
;***********************************************
;This routine to stor alp1k every (dif1) value
;***********************************************
      LDP   #dif
      LACC  dif
      SUB   #01h
      SACL  dif
      BCND  STORA,NC,LT      ; Store  if 20 value next
      B     BR

RS:                          ; Check (del*(rp+1))
      LDP   #rp
      LACC  rp
      SUB   #01h
      SACL  rp
      BCND  MMB2,NC,LT       ; Mark if =0
      LACC  #07fffh          ; Re store del
      SACL  del
      B     BO

STORA:                       ;Store every diff1 value
      LDP   #alp1k
      LACC  alp1k
      MAR   *,AR6
      SACL  *+               ; Store alp1k with shift  AR6
      LDP   #dif1
      LACC  dif1
      SUB   #01h
      SACL  dif              ; Restore  dif

      LDP   #temp2
      MAR   *,AR6
```

```
        SAR     AR6,temp2
        LACC    temp2
        MAR     *,AR7
        SUB     *                       ; Check address off AR6 with CBER
        BCND    MMB,C,GT                ; Branch  to  MMB if AR6 equal  CBER
        B       BR

MMB2:                                   ; Mark if store alp1(K) = Alp1dx*dif1 value

        LDP     #mb2
        LACC    #01h
        SACL    mb2
        B       BR

MMB:                                    ; Mark  if  show output =del*(1+rp)*(1/sampling rate) sec
        LDP     #mb
        LACC    #01h                    ; carry bit is ONE
        SACL    mb                      ; Mark  if  AR6=ACCB=CBER
        B       BR
;*****************************************
TRANSMIT:
      RETE
;***********************************************************************
;  DESCRIPTION: This routine initializes the 'C50 serial port    *
;           and the TLC320C40's (AIC) TA,RA,TB,RB and       *
;           control registers                       *
;***********************************************************************
AICINIT: SPLK   #20h,TCR        ; To generate 10 MHz from Tout
        SPLK    #01h,PRD        ; for AIC master clock
        MAR     *,AR0
        LACC    #0008h          ; Non continuous mode
        SACL    SPC             ; FSX as input
        LACC    #00c8h          ; 16 bit words
```

```
        SACL   SPC
        LACC   #080h          ; Pulse AIC reset by setting it low
        SACH   DXR
        SACL   GREG
        LAR    AR0,#0FFFFh
        RPT    #10000         ; and taking it high after 10000 cycles
        LACC   *,0,AR0        ; (.5ms at 50ns)
        SACH   GREG
        ;---------------
        LDP    #TA            ;
        SETC   SXM            ;
        LACC   TA,9           ; Initialize TA and RA register
        ADD    RA,2           ;
        CALL   AIC_2ND        ;
        ;---------------
        LDP    #TB
        LACC   TB,9           ; Initialize TB and RB register
        ADD    RB,2           ;
        ADD    #02h           ;
        CALL   AIC_2ND        ;
        ;---------------
        LDP    #AIC_CTR
        LACC   AIC_CTR,2      ; Initialize control register
        ADD    #03h           ;
        CALL   AIC_2ND        ;
        RET                   ;
AIC_2ND:
        LDP    #0             ; Data page point is 0  (MM regs)
        SACH   DXR            ; send ACChi 00
        CLRC   INTM           ; enable interrupts
        IDLE                  ; wait for interrupt
        ADD    #6h,15         ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH   DXR            ; send ACChi to initiate secondary protocol
```

```
SACL   DXR          ; send the T register data
IDLE                ; wait for interrupt
LACL   #0           ; clear ACClo
SACL   DXR          ; send another to make sure 1st word got sent
IDLE                ; wait for interrupt
SETC   INTM
RET
.END
```

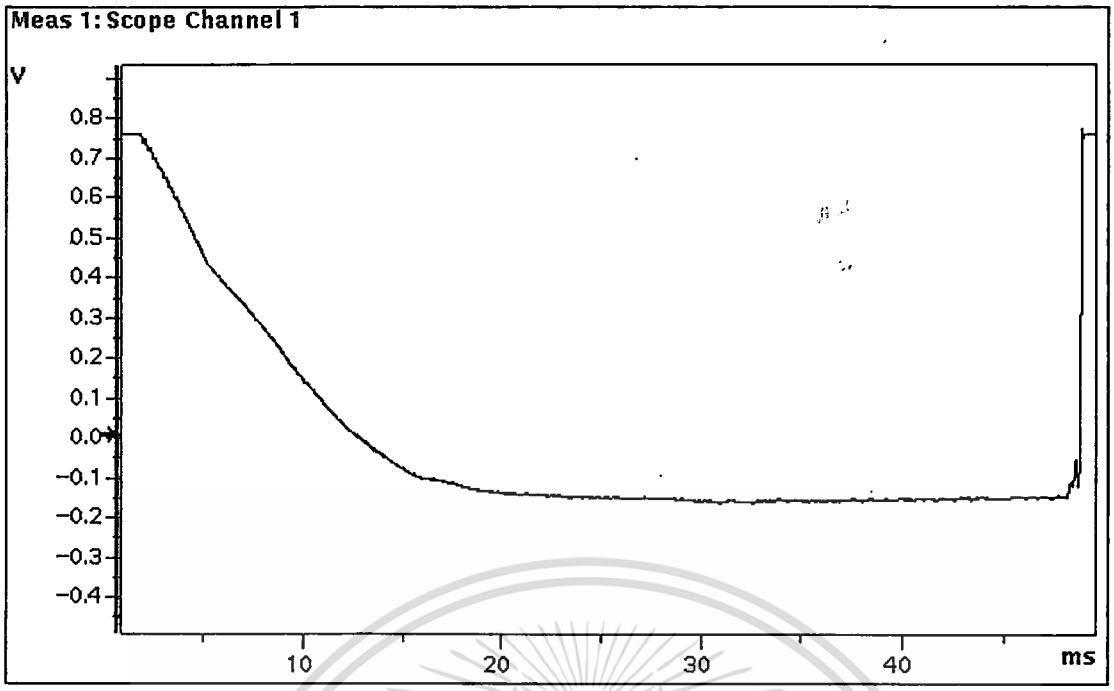
ในการดูการเปลี่ยนแปลงค่าของ $\alpha_1(k)$ นั้นในเงื่อนไขเริ่มแรกจะทำการกำหนดค่าเริ่มต้นของมันที่ 0 ซึ่งจะเท่ากับความถี่ใน Time Domain ที่ 2.5 KHz หลังจากนั้นก็ปรับความถี่ Input ตามต้องการแล้วสั่ง Run Program จะเห็นการปรับค่าของระดับแรงดัน ในช่วงแรกหลังจากนั้นจะเห็นรูปคลื่นมีความถี่ 20 Hz ที่ Scope ซึ่งก็คือรูปการเปลี่ยนแปลงค่าของ $\alpha_1(k)$ ซึ่งเมื่อเรารู้ค่าในการข้ามค่าและในการเก็บจะทำให้สามารถคำนวณค่าเวลาการติดตามความถี่ที่เงื่อนไขนั้นได้ ส่วนการเปลี่ยนค่าตัวแปรต่างๆ จะทำการแปลงค่าเหมือนในหัวข้อที่ 4.2.1 แล้วเปลี่ยนค่าในโปรแกรม

Note: ในการบันทึกผลการทดลองคราวนี้จะทำการบันทึกผลกับ Tektronix AM 700 Audio Measurement Set ซึ่งเป็นอุปกรณ์วัดทดสอบค่าต่างๆ สำหรับความถี่ในย่าน Audio ซึ่งมีทั้ง Scope , Function Generator , FFT Analyzer ซึ่งสามารถบันทึกค่าออกมาเป็น File รูปภาพตามมาตรฐานของคอมพิวเตอร์ได้

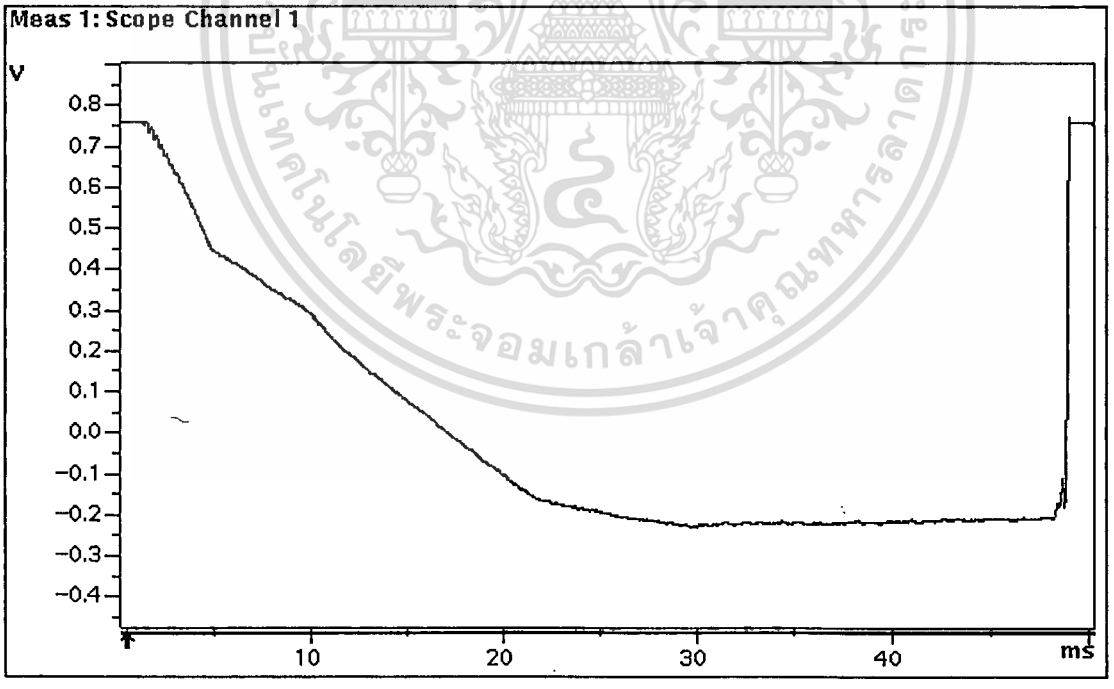
4.3.5.1 การติดตามสัญญาณความถี่ Input เมื่อมีการเปลี่ยนแปลงค่า α_0 (ค่า Q)

ในหัวข้อนี้จะกำหนดเงื่อนไขความถี่เริ่มต้นของ $\alpha_1(k)$ เท่ากับ 0 (ซึ่งเท่ากับ 2.5 KHz ใน Time Domain) เหมือนกับในตอนแรก ซึ่งจะให้ทำการติดตามความถี่สัญญาณ Input ที่ 3.3 KHz โดยจะทำการเปลี่ยนค่า α_0 ค่าต่างๆ แล้วดูเวลาในการติดตามความถี่สัญญาณ Input โดยกำหนดค่า μ ไว้ที่ 0.0003 โดยในเงื่อนไขนี้จะไม่มีสัญญาณรบกวนซึ่งสามารถแสดงรูปคลื่นของ $\alpha_1(k)$ ได้ดังนี้

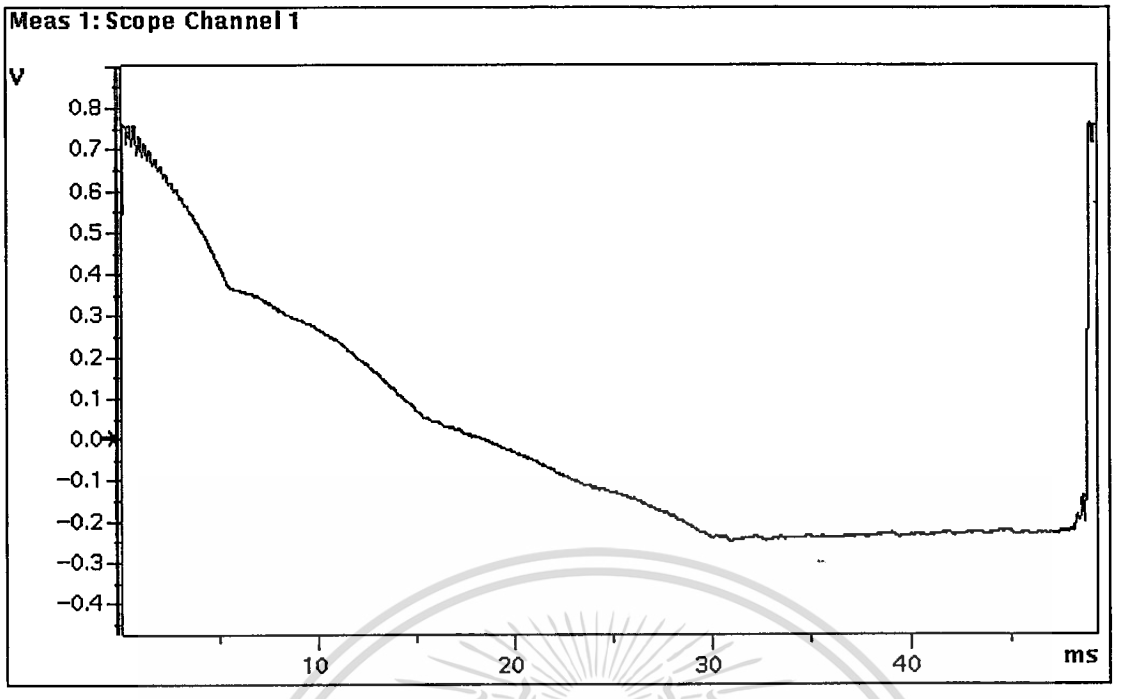
Note: โดยในส่วนนี้จะเก็บค่า $\alpha_1(k)$ โดยข้ามทีละ 40 ค่า



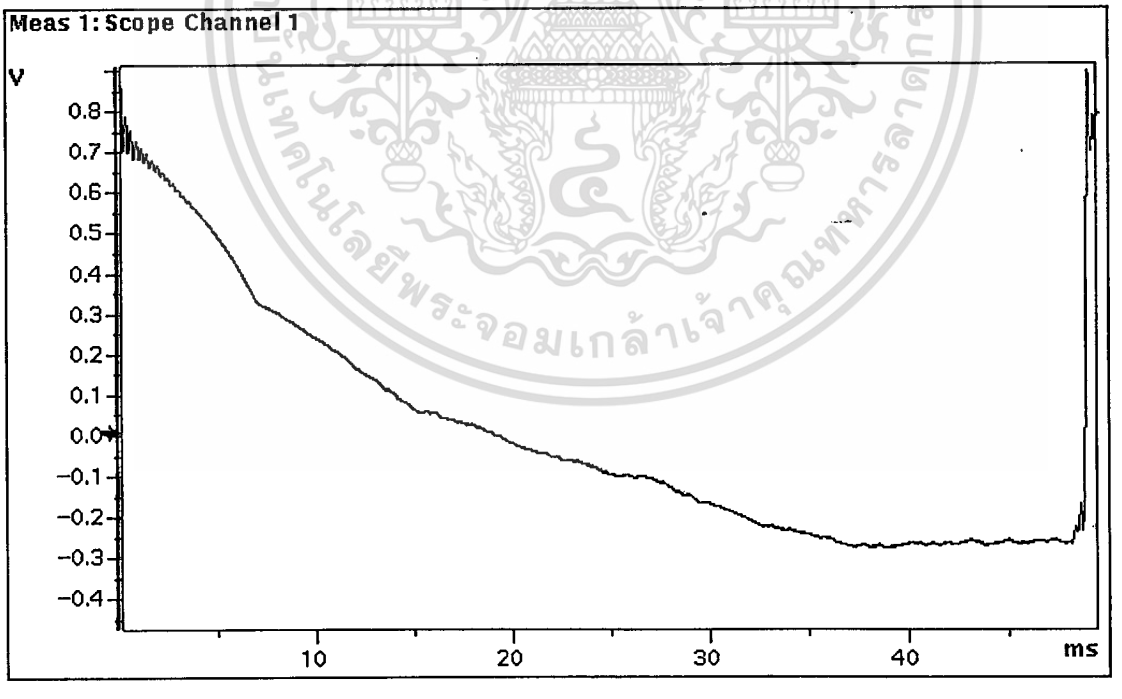
รูปที่ 4.21 การเปลี่ยนค่าของ $\alpha_1(k)$ ที่ ค่า $\alpha_0 = 0.4$



รูปที่ 4.22 การเปลี่ยนค่าของ $\alpha_1(k)$ ที่ ค่า $\alpha_0 = 0.5$



รูปที่ 4.23 การเปลี่ยนค่าของ $\alpha_1(k)$ ที่ ค่า $\alpha_0 = 0.6$



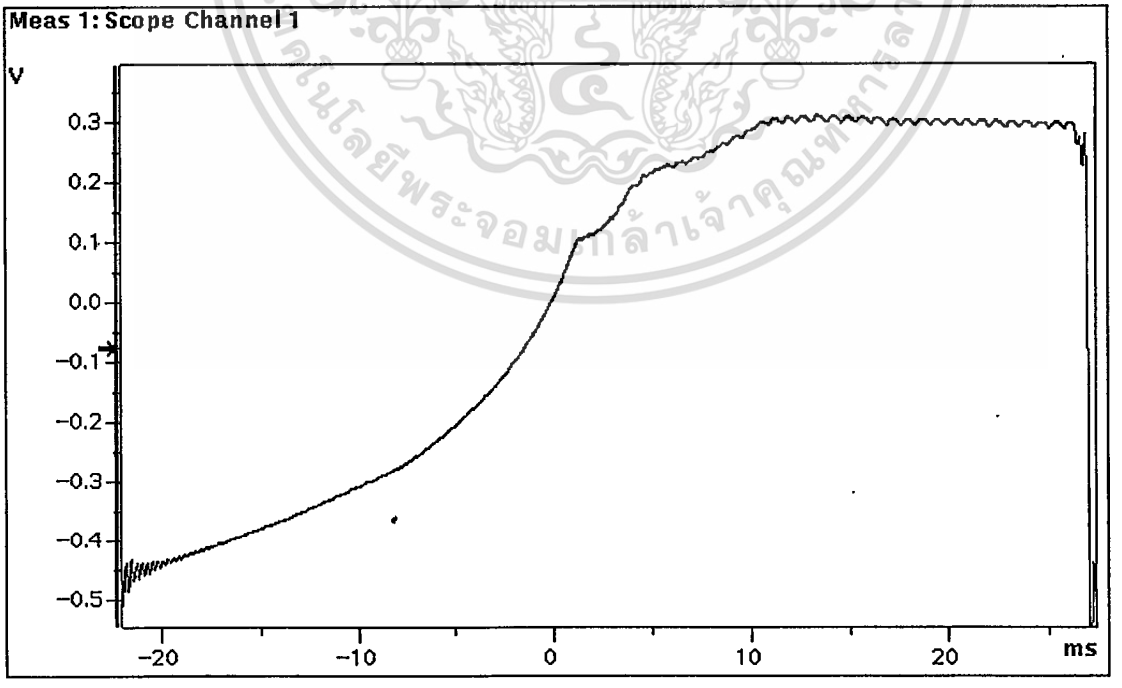
รูปที่ 4.24 การเปลี่ยนค่าของ $\alpha_1(k)$ ที่ ค่า $\alpha_0 = 0.7$

จากรูปจะเห็นว่าความถี่ของรูปคลื่นที่แสดงออกมาที่ Scope จะมีความถี่ประมาณ 20 Hz เท่ากับที่คำนวณมาตอนเขียนโปรแกรม จากรูปการเปลี่ยนแปลงค่าของ $\alpha_1(k)$ จะเปลี่ยนค่าลดลงเพราะที่ความถี่สูงกว่า 2.5 KHz ค่าของมันจะเป็นลบแต่ Scope ที่ทำการวัดไม่สามารถวัดค่าเป็น DC ได้เพราะถูกออกแบบมาสำหรับวัดค่าในย่านความถี่ Audio (ทั้งนี้ได้ทำการวัดใน Analog Scope จะเห็นว่าค่าอยู่ในย่านลบและจะเริ่มต้นที่ 0 แต่ไม่สามารถบันทึกผลมาแสดงได้) โดยจากรูปจะเห็นว่าเมื่อเราเพิ่มค่า α_0 จะทำให้การติดตามความถี่สัญญาณ Input ช้าลงเหมือนกับที่ทำการ Simulate ไว้

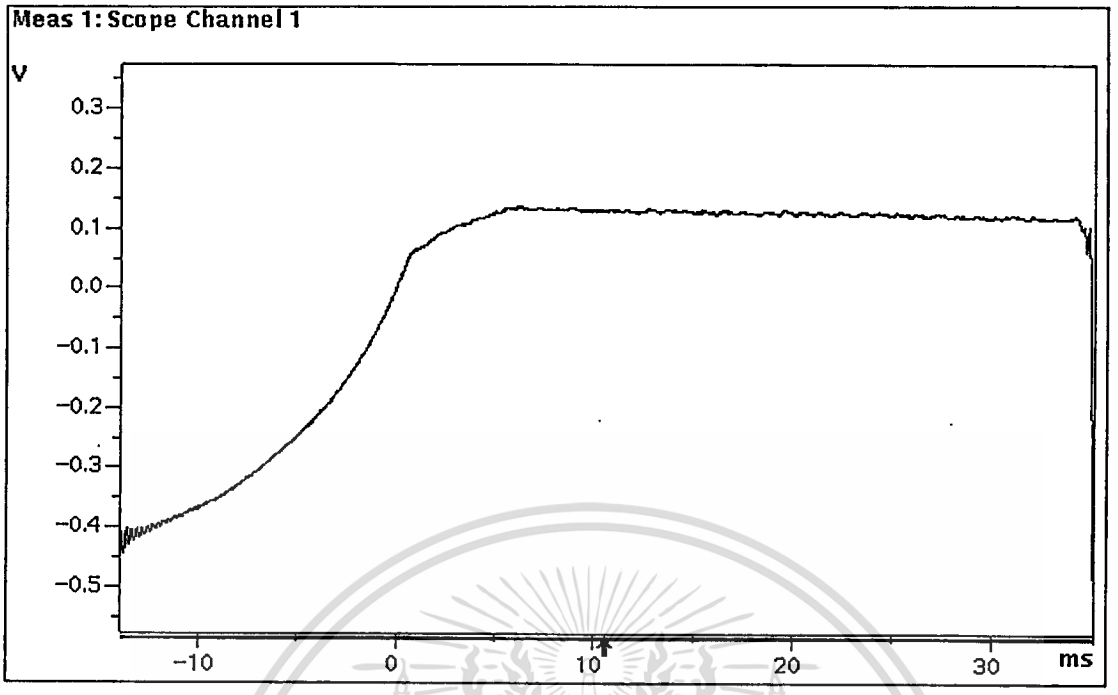
4.5.3.2 การติดตามความถี่ที่ความถี่ต่างๆ

ในขั้นตอนนี้จะทำการทดลองดูการเปลี่ยนแปลงความถี่ที่จะเริ่มต้นที่ 2.5K โดยจะดูค่าที่มันทำการติดตามสัญญาณความถี่ต่างๆ โดยค่าที่มันเปลี่ยนไปนั่นจะไม่เท่ากัน ซึ่งสามารถดูเห็นความเตกต่างได้ที่ขนาด Amplitude ของรูปคลื่นได้โดยในขั้นตอนนี้จะกำหนดค่าของ α_0 ไว้ที่ 0.5 ค่า μ ไว้ที่ 0.0003 โดยการเก็บค่า $\alpha_1(k)$ ทำการข้ามทีละ 35 ค่า ทำการเปลี่ยนสัญญาณ Input ความถี่ต่างแล้ว Run DSP แล้วทำการบันทึกผล

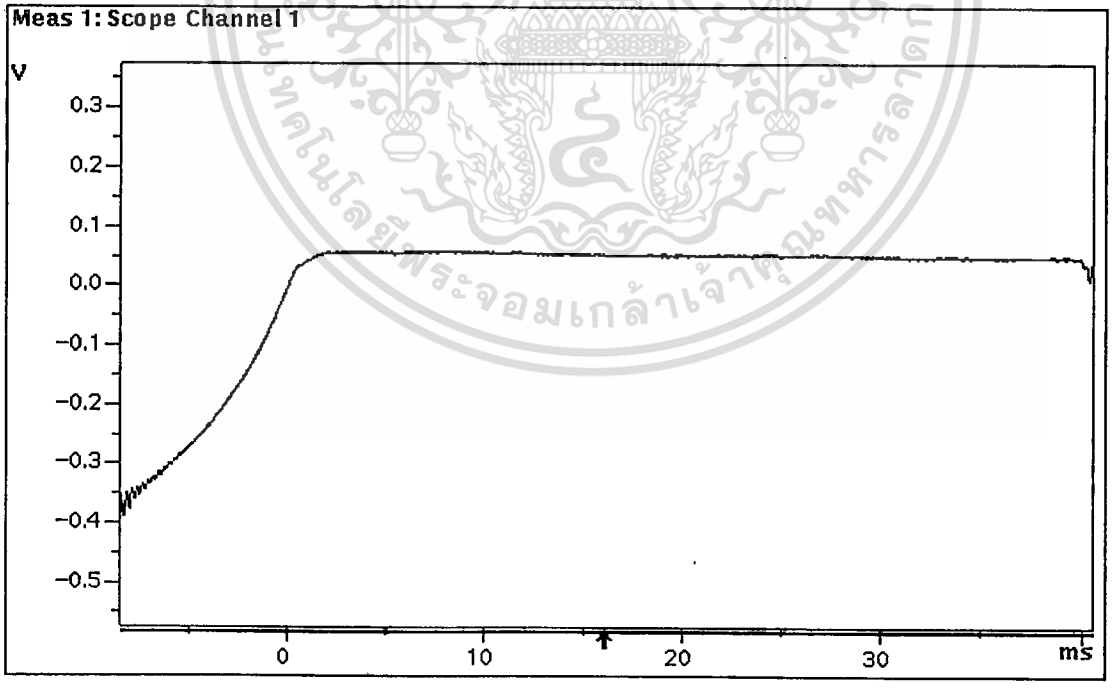
โดยรูปคลื่นที่บันทึกผลจากการทดลองมีดังนี้



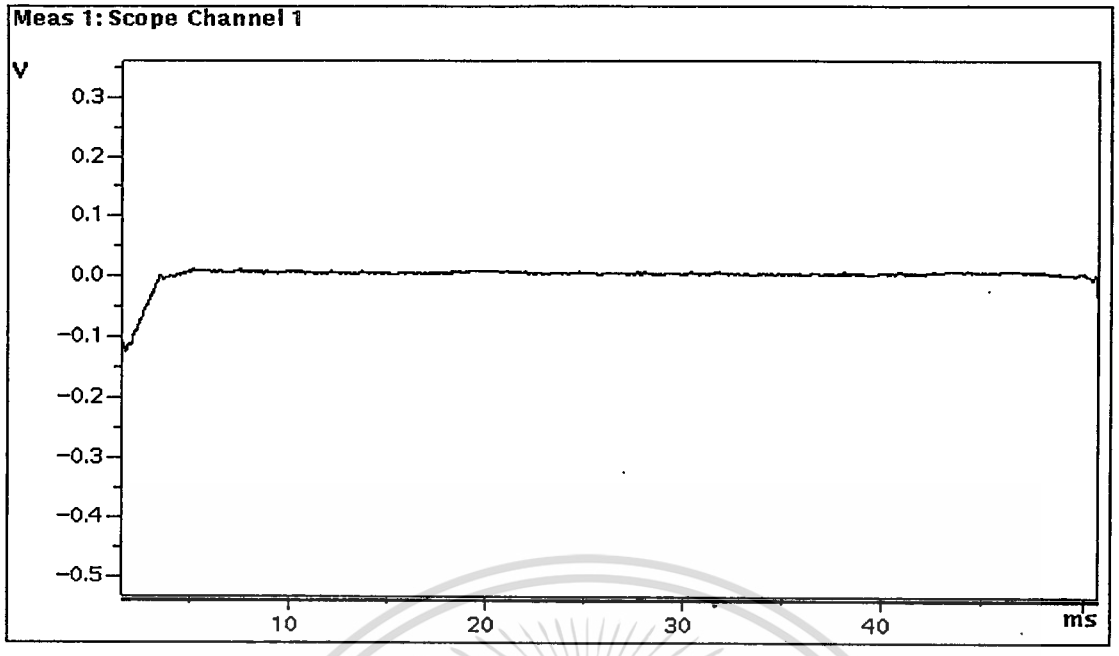
รูปที่ 4.25 ค่าของ $\alpha_1(k)$ ที่เปลี่ยนติดตามความถี่ Input ที่ความถี่ 1.5 kHz



รูปที่ 4.26 ค่าของ $\alpha_1(k)$ ที่เปลี่ยนติดตามความถี่ Input ที่ความถี่ 1.6 kHz

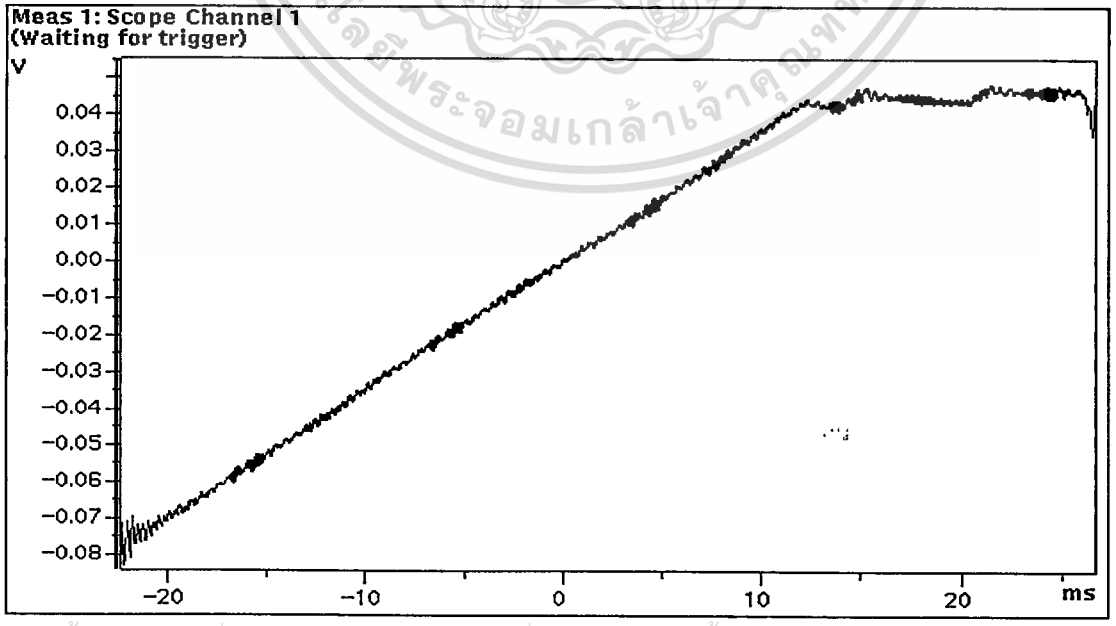


รูปที่ 4.27 ค่าของ $\alpha_1(k)$ ที่เปลี่ยนติดตามความถี่ Input ที่ความถี่ 1.7 kHz

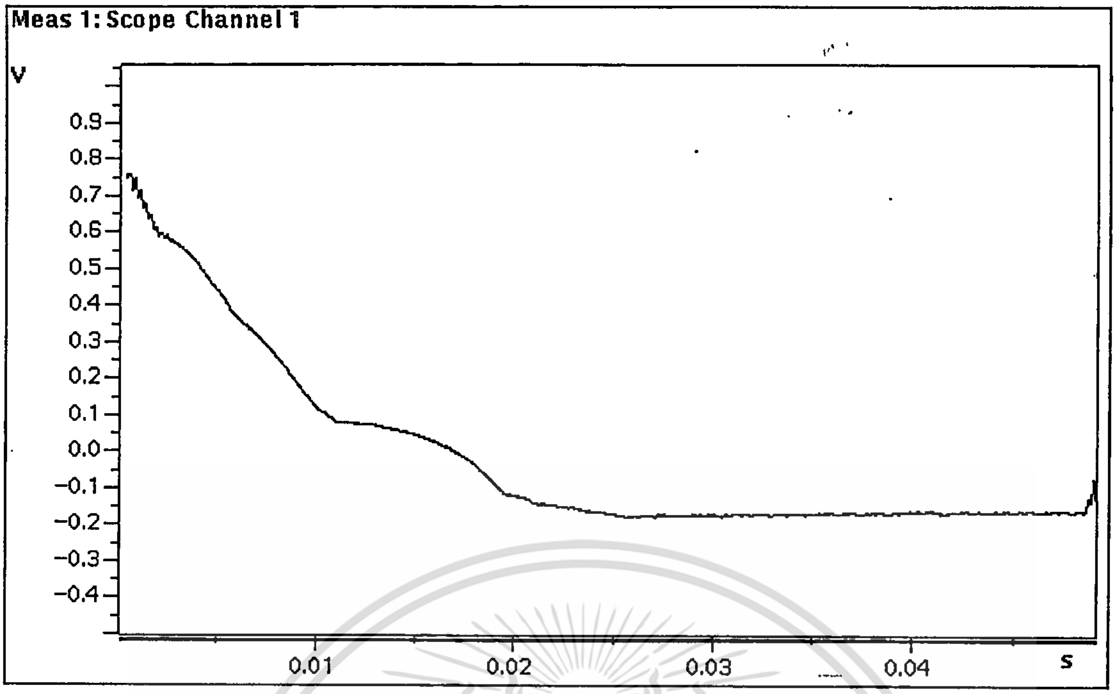


รูปที่ 4.28 ค่าของ $\alpha_1(k)$ ที่เปลี่ยนติดตามความถี่ Input ที่ความถี่ 2 kHz

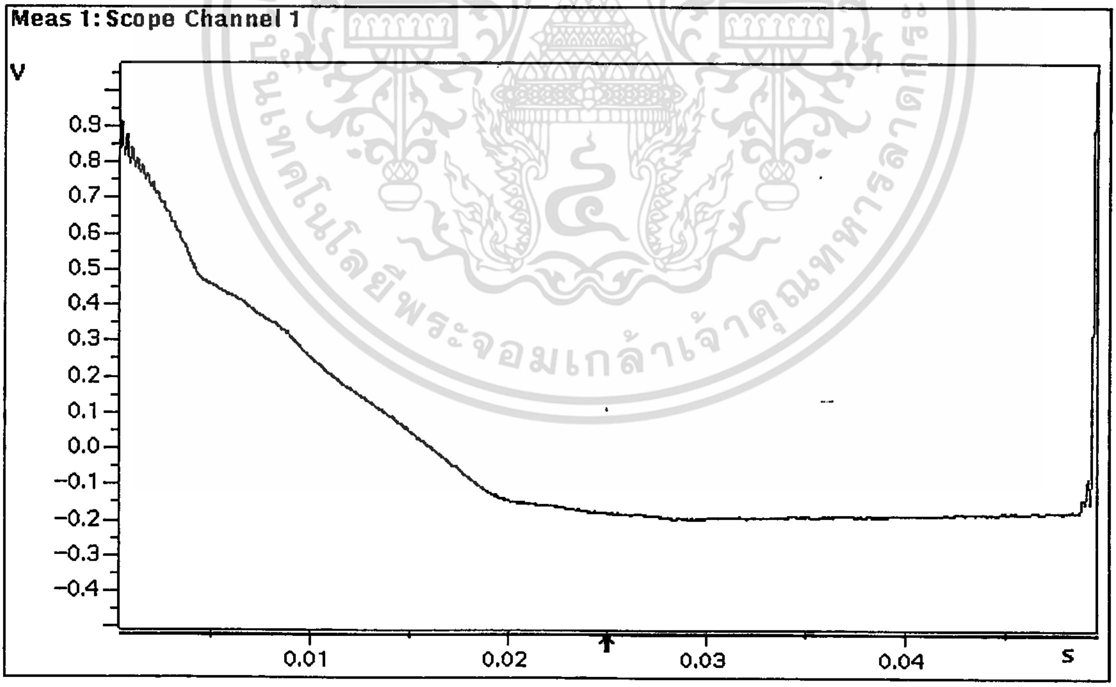
โดยจากรูปค่า $\alpha_1(k)$ ทั้งหมดที่ทำการทดลองแล้วบันทึกมาทั้งหมดนั้นจะแสดงให้เห็นใน Scale เดียวกันทั้งหมด เพื่อจะเห็นค่าการเปลี่ยนแปลงที่ชัดเจน จะเห็นว่าการเปลี่ยนแปลงค่าของ $\alpha_1(k)$ ที่ความถี่ใกล้กับ 2.5 kHz จะมีการเปลี่ยนแปลงน้อยกว่าความถี่ที่ห่างกับ 2.5 kHz โดยในการเปลี่ยนแปลงในช่วงของความถี่ที่น้อยกว่า 2.5 kHz จะมีค่ามากว่า 0 ดังที่อธิบายมาแล้วในหัวข้อที่ผ่านมาโดยรูปถัดไปจะเป็นค่าของ $\alpha_1(k)$ จาก 2.5 kHz ที่เปลี่ยนไปเป็น 2 kHz แต่จะทำการเก็บค่าโดยการข้ามทีละ 2 ค่าซึ่งก็คือการแสดงรูปที่ผ่านมาแต่เป็นการแสดงให้เห็นในช่วงเวลาที่มีการเปลี่ยนแปลงของค่า $\alpha_1(k)$ มากขึ้น



รูปที่ 4.29 การเปลี่ยนค่า $\alpha_1(k)$ เมื่อ Input เป็น 2 kHz



รูปที่ 4.30 การเปลี่ยนค่า $\alpha_1(k)$ เมื่อ Input เป็น 3.1 kHz

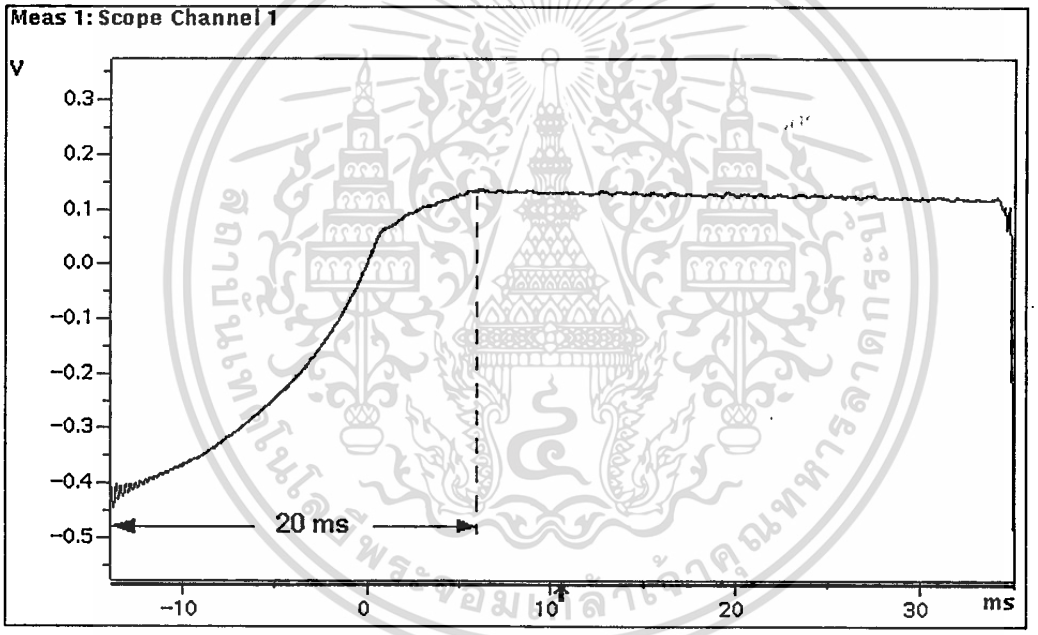


รูปที่ 4.31 การเปลี่ยนค่า $\alpha_1(k)$ เมื่อ Input เป็น 3.3 kHz

จากสองรูปที่ผ่านมาจะเห็นว่าเมื่อความถี่ Input สูงกว่า 2.5 kHz การเปลี่ยนแปลงค่าของ $\alpha_1(k)$ จะเปลี่ยนไปเป็นค่าลบดังที่ได้อธิบายมาแล้วจะเห็นว่าการเปลี่ยนค่าจะเป็นไปตามทฤษฎี

4.5.3.3 การคำนวณเวลาในการปรับค่า $\alpha_1(k)$ จากรูปที่ Scope

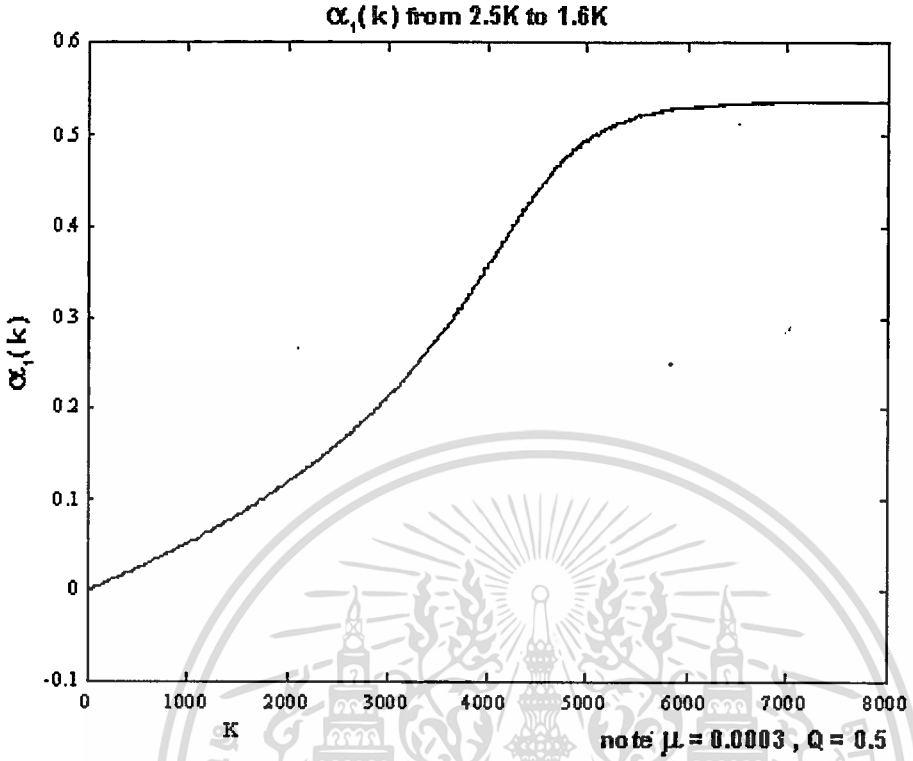
ในขั้นตอนนี้จะทำการคำนวณค่า $\alpha_1(k)$ โดยจากรูปที่ 4.26 จะนำเอามาคำนวณหาค่าเวลาการติดตามความถี่สัญญาณ Input โดยค่าความถี่ในการ Sampling 10 kHz และการเก็บค่าของ $\alpha_1(k)$ จะข้ามทีละ 35 ค่า จะสามารถนำมาทำการคำนวณได้โดย จากรูปสามารถคำนวณค่า Time จากรูปก่อน ซึ่งจะได้ค่า Time ในการเปลี่ยนแปลงค่าไปสู่ค่าที่ถูกต้องที่แสดงในรูปที่ 4.32 ประมาณ 20 ms และในขั้นตอนของการเขียนโปรแกรมที่ทำการ เก็บค่า โดยการข้ามที่ละ 35 ค่า ทำการเก็บค่าทั้งหมด 495 จะได้ค่า K ทั้ง หมดที่ทำการเก็บ 17325 ค่า ซึ่งก็คือค่าของ ค่า K ทั้งหมด 1 รูปคลื่นของ $\alpha_1(k)$ ซึ่งมีค่าTime ทั้งหมด 50 ms (จากที่กล่าวถึงมาแล้ว) แต่ค่าของ $\alpha_1(k)$ ที่ติดตามสัญญาณ Input จะใช้เวลา 20 ms จะสามารถคำนวณค่า K จริงๆ จากการเทียบบัญญัติไตรยางค์



รูปที่ 4.32 การอ่านค่าการติดตามสัญญาณ Input ของค่า $\alpha_1(k)$

โดยค่า $K = 17325 \times \frac{20}{50} = 6930$

ซึ่งสามารถคำนวณค่าเวลาที่ใช้ในการทำการติดตามสัญญาณ Input จะเท่ากับ $6930 \times .1ms = 693ms$ โดยที่จะทำการเปรียบเทียบกับผลของการ Simulation ที่เงื่อนไขเดียวกัน ดังรูปที่ 4.33 จะแสดงต่อไปนี้



รูปที่ 4.33 การเปลี่ยนแปลงค่า $\alpha_1(k)$ ติดตามค่าความถี่สัญญาณ Input (Simulate)

จากรูปการ Simulation ที่ได้ทำการ กำหนดเงื่อนไขเดียวกันกับ เขียนโปรแกรมบน Chip เพื่อทำการเปรียบเทียบ กับการทำงานจริงๆ โดยที่จากรูปจะเห็นว่ามีการติดตามความถี่ที่ Input ทันที่ประมาณค่า K ประมาณ 6500 ค่า ส่วนการ คำนวณค่าจากการ เขียนโปรแกรมจะได้ประมาณ 6930 ค่า

บทที่ 5
สรุปผลและวิจารณ์ผลการทดลอง

จากการศึกษาและทำการทดลองตั้งแต่ขั้นตอนทำการ Simulation ในรูปแบบต่างๆ และนำมาทำการเขียนโปรแกรมจริงๆ ทำให้เราสามารถทำความเข้าใจเกี่ยวกับการทำงาน ของการประมวลผลสัญญาณแบบดิจิตอลทั้งในทฤษฎีและการนำไปทำการประมวลผลสัญญาณจริงๆ รวมทั้งเรียนรู้การใช้งาน Chip Digital Signal Processing ทั้งในขั้นตอนของการเขียนโปรแกรมและแนวทางในการนำไปใช้งานด้านต่างๆ รวมทั้งสามารถเข้าใจเงื่อนไขต่างๆ ที่เป็นข้อดีและข้อจำกัด ของการประมวลผลสัญญาณแบบดิจิตอลได้ชัดเจนขึ้นกว่าการศึกษาทางทฤษฎีหรือทำการ Simulation เพียงอย่างเดียว

5.1 การแยกสัญญาณ Sine ออกจากสัญญาณรบกวน

ในการพิจารณาผลการทำงานในส่วนนี้จะดูรูปคลื่นที่สัญญาณ Output ซึ่งจะเห็นว่าสามารถกำจัด Noise ได้ในระดับที่ดีมากต่อไปคือการดูการติดตามความถี่ของสัญญาณ Input ที่ความถี่ต่างๆ โดยการ Track เปลี่ยนความถี่สัญญาณ Input ด้วยมือซึ่งพบว่าระบบสามารถทำการติดตามการเปลี่ยนความถี่ได้ดีโดยสามารถติดตามความถี่ ที่ด้านความถี่ต่ำถึงประมาณ 600 Hzในด้านความถี่สูงจะได้ประมาณ 3.8 kHz โดยมีเงื่อนไขคือการเปลี่ยนแปลงความถี่ จะต้องไม่มากเกินไปเช่นเปลี่ยนจาก 600 Hz ไปเป็น 3 kHz จะไม่สามารถทำได้เพราะ Amplitude ที่ความถี่ที่เปลี่ยนไปเป็นความถี่ใหม่ โดยที่ความถี่กลางของ ระบบยังคงอยู่ที่ค่าเดิมนั้นจะมีค่าน้อยเกินไปที่จะนำมาทำการ Update ค่าของ $\alpha_1(k)$ โดยจะสรุปสาเหตุเป็นข้อในตอนท้ายอีกครั้ง

5.2 การติดตามความถี่ของสัญญาณ Input

จากการเปรียบเทียบค่าเวลาในการทำการติดตามความถี่ของสัญญาณ Input ของการ Simulation และการคำนวณค่าจากขั้นตอนของการเขียนโปรแกรมจริงๆ จะเห็นว่าค่าที่ได้จากการทดลองมีค่ามากกว่าค่าที่ได้จากการ Simulation ที่เงื่อนไขเดียวกันไม่มาก (จากการคำนวณในหัวข้อที่ 4.5.3.3) โดยจากการคำนวณรูปอื่นๆ พบว่าที่ความถี่กลางเริ่มแรกห่างจากความถี่ Input มากจะมีค่าความแตกต่างมากขึ้นเพราะยิ่งการคำนวณหลายๆ ค่าก็จะมีค่าผิดพลาดที่ค่าสุดท้ายมากขึ้น

ส่วนเงื่อนไขของค่าความถี่ที่ทำการติดตามความถี่ของสัญญาณ Input ในเงื่อนไขเริ่มแรกนั้น เราได้พบข้อจำกัดของมันคือจะมีขอบเขตในการติดตามค่าความถี่ที่ Input คือยิ่งค่า α_0 มีค่ามากเท่าใดขอบเขตของความถี่ในการติดตามที่เงื่อนไขเริ่มแรก (เงื่อนไขเดียวกับการ Simulation) จะมีขอบเขตในการติดตามความถี่จำกัดมากขึ้นโดยที่ค่า $\alpha_0 = 0.5$ นั้นจะทำการติดตามสัญญาณ ที่ด้านความถี่ต่ำได้ต่ำสุดประมาณ 1.4 kHz และทางด้านความถี่สูงประมาณ 3.6 kHz หรือ ประมาณ 1.1 kHz รอบๆ ความถี่กลาง (2.5 kHz) ซึ่งถ้าค่า α_0 มากกว่านี้จะมีขอบเขตแคบกว่านี้อีก โดยที่การ Simulation นั้นจะ

สามารถติดตามความถี่ได้กว้างกว่าแต่จะใช้เวลานานมากที่เงื่อนไขเหล่านี้โดยสามารถสรุปสาเหตุที่ทำให้เกิดข้อจำกัดต่างๆ ที่ทำให้เกิดข้อจำกัดเหล่านี้เป็นข้อๆ ในหัวข้อถัดไป

5.3 สรุปผลการทดลอง

จากการเขียนโปรแกรมในการ Simulation และเขียนโปรแกรมใน Chip DSP จริงๆ ทำให้สามารถสรุปข้อแตกต่างที่ทำให้ได้ค่าการติดตามความถี่ในเงื่อนไขต่างๆ ที่เหมือนกัน ระหว่างการ Simulation กับการเขียนโปรแกรมบน Chip DSP ซึ่งมีการผลการทำงานที่แตกต่างกันได้เป็นข้อๆ ดังนี้

1. ในการคำนวณที่ใช้ Software ในการ Simulation จะมีเงื่อนไขในการทำการเก็บค่าที่ไม่จำกัด เช่นค่าทศนิยม ขนาด 0.00008 ในการ Simulation จะยังคงมีค่าอยู่ (เพราะ Software ที่ทำการคำนวณค่าจะจัดการบันทึกเก็บและคำนวณค่าในคอมพิวเตอร์ ซึ่งสามารถทำได้มากคือสามารถเก็บคำนวณค่าได้ถึง 2^{1023} (ใน Matlab Version 5) โดยจริงๆ ในการทำการคำนวณของคอมพิวเตอร์ปัจจุบัน มีการคำนวณในรูป 32 bit แต่ค่าส่วนนี้จะถูกจัดการ โดย Software) แต่ในการเขียนโปรแกรมใน Chip DSP ซึ่งมีการเก็บคำตอบในการ คำนวณใน รูป 16 bit ซึ่งค่าดังกล่าวจะกลายเป็น 0 ดังนั้นใน เทอมที่ทำการ Update ค่า $\alpha_1(k)$ นั้นค่าจะไม่มีการเปลี่ยนแปลงเลยในกรณีของการเขียนโปรแกรมใน Chip DSP โดยที่ไม่สามารถใช้ Software ในการแก้ไขตรงนี้ได้เพราะจะเป็นการสิ้นเปลืองเวลาในการทำงานของ DSP มากจนไม่สามารถที่จะทำการประมวลผลแบบ Real Time ได้ ซึ่งค่าเวลาการติดตามความถี่หรือขอบเขตของการติดตามความถี่ คิดว่าน่าจะเกิดจากเงื่อนไขนี้เป็นเงื่อนไขหลักที่ทำให้ไม่เหมือนกับการ Simulation
2. เงื่อนไขการแปลงค่าของ D/A เนื่องจากในการแปลงค่าจะทำการอ้างอิงกับการแปลงค่าของ 2^{14} bit ซึ่งจะได้ค่าที่ Full Scale ก็ต่อเมื่อค่าที่แรงดัน Input มีค่าเท่ากับ 6 $V_{p\text{-}p}$ หรือ $\pm 3V$ ซึ่งในการกำหนดระดับแรงดันที่ใช้ในการทดลองนั้น เราจะไม่กำหนดไว้ที่ 6 $V_{p\text{-}p}$ เพราะจะทำให้การทำงานไม่ค่อยเสถียรภาพ เราจะทำการกำหนดในการทดลองประมาณ 5.5 Volt ซึ่งจะทำให้การ Update ค่าที่จะทำการคำนวณหาค่า $\alpha_1(k)$ ในเทอมของ $x(n)$ และ $y(n)$ ต่างๆ มีค่ามากที่สุดไม่เต็ม Scale จึงทำให้เทอม (k) ที่จะทำการ Update $\alpha_1(k)$ นั้นมีค่าไม่เท่ากับการคำนวณโดยการ Simulation ซึ่งจะทำให้การ Update ค่านั้นมีการเปลี่ยนแปลงค่าไม่เหมือนกับการ Simulation รวมถึงเงื่อนไขของการเปลี่ยนค่า α_0 ที่มีผลต่อขอบเขตของการติดตามความถี่ทางด้าน Input ด้วย
3. เงื่อนไขของการแปลงค่าที่จะนำมาทำการคำนวณในการหาค่าในการ Update ซึ่งจะต้องมีการปัดค่าเพราะในการแปลงค่า (Scaling) จากค่าจำนวนทศนิยมเป็นจำนวนเต็มนั้นจะมีการปัดค่าซึ่งจะมีค่าที่ต่างจากค่าจริงๆ พอสมควรจะทำให้เกิดการแตกต่างกันกับค่าของการ Simulate เพราะที่การคำนวณหลายๆ ค่าจะเห็นความแตกต่างมากขึ้น
4. การอ่านค่าจากกราฟที่ขั้นตอนของการทำการคำนวณหาค่า K จะเกิดจากการประมาณ

5. มีขั้นตอนของการแปลงค่าที่จะทำการคูณในรูปของจำนวนทศนิยม (Floating-Point) มาเป็นทำการคูณในรูปของจำนวนเต็ม (Fixed Point) แทน ซึ่งใน Chip DSP จะมีการแปลงค่าทำให้เกิดการประมาณและปัดค่าที่ได้จากการแปลง

5.4 แนวทางในการทำการพัฒนา

1. อาจจะทำการทดลองทำการสร้างการ์ด Interface กับคอมพิวเตอร์ PC ที่จะมีรูปแบบการคำนวณในรูปของ 32 bit และใช้การเขียนโปรแกรมด้วยภาษาชั้นสูงในคอมพิวเตอร์ โดยเลือกใช้ D/A และ A/D ที่มี Sampling Rate สูงๆ จะสามารถที่จะทำการทดลองในเงื่อนไขต่างๆ ได้กว้างขึ้น เช่นสามารถทดลองในช่วงความถี่กว้างขึ้นมีการคำนวณในรูปที่ทำการประมาณค่าน้อยลงแต่จะสามารถที่จะนำไปใช้งานจริงได้ยาก
2. อาจจะทดลองเขียนโปรแกรมใน Chip DSP ที่มีการคำนวณในรูปทศนิยมได้เช่น Chip TMS320 ตระกูล C3x ซึ่งจะมีความเหมาะสมในการทำการคำนวณในงานด้านนี้มากกว่า
3. หา Algorithm ในการติดตามความถี่ที่มีประสิทธิภาพมากขึ้น เช่น Algorithm ที่ทำการ Update ค่า Q-Factor ด้วยขณะ Update ค่าความถี่กลางซึ่งจะสามารถทำการติดตามสัญญาณได้ดีขึ้นแต่จะมีการเปลืองเวลาของ CPU มากขึ้นในการคำนวณซึ่งต้องทำการทดลองดู
4. ทำการทดลองเขียนโปรแกรมที่ทำการ Cascade ระบบซึ่งจะเพิ่ม order ในการคำนวณ โดยในการคำนวณของ Stage หลังๆ ไม่ต้องคำนวณค่าสัมประสิทธิ์ในการ Update ค่าความถี่แต่จะนำสัมประสิทธิ์ค่าความถี่กลางของการ Update ครั้งล่าสุดมาทำการคำนวณเหมือน Band pass Filter ธรรมดาแต่สามารถที่จะไม่ต้องใช้ค่า Q ที่สูงมากใน Stage แรกซึ่งจะสามารถทำการติดตามสัญญาณ Input ได้เร็วขึ้นและใน Stage หลังจะทำการคำนวณรูปคลื่นที่จะทำการส่งออกที่ Output โดยการนำเอาค่าความถี่กลางที่ Update มาใน Stage แรกมาเป็นค่าที่ทำการคำนวณแต่จะนำค่า Q ที่มากๆ มาใช้ใน Stage นี้ซึ่งจะสามารถกำจัดสัญญาณรบกวนได้ดีกว่า
5. ทดลองเขียนโปรแกรมที่สามารถเก็บค่าของการ Update สัมประสิทธิ์ต่างๆ ในรูป 32 bit โดยโปรแกรมจะมีความซับซ้อนมากขึ้นพอสมควร แต่จะสามารถทำการคำนวณทันหรือไม่นั้น จะต้องทดลองดู โดยทำเท่าที่จะสามารถคำนวณทัน

เอกสารอ้างอิง

[1] กิตติพัฒน์ ต้นตระรุ่งโรจน์, "สัญญาณและระบบเวลาเต็มหน่วย" , พิมพ์ดี , กรุงเทพฯ , พฤษภาคม , 2540

[2] ชวลิต เบญจางคประเสริฐ, อุทัย ศรีธีระวิโรจน์ และ กนก เจนจิระพงศ์เวช, "อะแดปทีฟอัลกอริทึมสำหรับการดีเทคสัญญาณซายน์คลื่นเดี่ยว โดยใช้ฟิลเตอร์แบบปรับค่าคิวแฟกเตอร์", การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 19 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น , วันที่ 7-8 พฤศจิกายน 2539 , หน้า cm-49 ~ cm-55

[3] วิวัฒน์ กิรานนท์, "วิศวกรรมการสื่อสาร", อักษรสยามการพิมพ์, กรุงเทพฯ , มกราคม , 2540

[4] สุธรรม ศรีเกษม, "MATLAB เพื่อการแก้ปัญหาทางวิศวกรรม" , ศรีอนันต์การพิมพ์ , กรุงเทพฯ

[5] เอก ไชยสวัสดิ์ , "สัญญาณและระบบ" , วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์ , กรุงเทพฯ , 2538

[6] Alan, V.Oppenheim and Ronald, W.Schafer, "DISCREAT-TIME SIGNAL PROCESSING" ,Prentice-Hall , Englewood Cliffs , NJ , 1989

[7] D.M.Etter, "ENGINEERING PROBLEM SOLVING WITH MATLAB", Prentice-Hall , Simon & Schuster (Asia) , 1996

[8] Digital Signal processing Products , "TMS320C5x User's Guide" , Texas Instrument : 1993

[9] Microprocessor Development Systems , "TMS320C5x DSP Starter Kit User's Guide" , Texas Instruments : 1994.

[10] Sanjit,K. Mitra , "Handbook for Digital Signal Processing" , John Wiley & Sons , Inc ,California , 1993