การประยุกต์ใช้งานไมโครโปรเซสเซอร์ DSP ในการทำตัวกรองเชิงเลข

MICROPROCESSOR APPLICATION (DSP) FOR DIGITAL FILTER

โดย

นายสันติภาพ อุโคตร

นางสาววิรุณรยา สุดสน

นางสาวอลงกต ใจวังโลก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

การประยุกต์ใช้งานไมโครโปรเซสเซอร์ DSP ในการทำตัวกรองเชิงเลข

MICROPROCESSOR APPLICATION (DSP) FOR DIGITAL FILTER

โดย

นายสันติภาพ อุโคตร เลขประจำตัว 35104464

นางสาววิรุณรยา สุดสน เลขประจำตัว 35104546

นางสาวอลงกต ใจวังโลก เลขประจำตัว 35104547

อาจารย์ที่ปรึกษา

อ. เทอดศักดิ์ ลิ่วหาทอง

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งานไมโครโปรเซสเซอร์ DSP ในการทำตัวกรองเชิงเลข

(MICROPROCESSOR APPLICATION (DSP) FOR DIGITAL FILTER)

ผู้จัดทำ

1. นายสันติภาพ อุโคตร รหัสประจำตัว 35104464
2. นางสาววิรุณรยา สุดสน รหัสประจำตัว 35104546
3. นางสาวอลงกต ใจวังโลก รหัสประจำตัว 35104547

.....อาจารย์ที่ปรึกษา

(อ. เทอดศักดิ์ ลิ่วหาทอง)

การประยุกต์ใช้งานไมโครโปรเซสเซอร์ DSP ในการทำตัวกรองเชิงเลข

MICROPROCESSOR APPLICATION (DSP) FOR DIGITAL FILTER

ผู้จัดทำ

1. นายสันติภาพ อุโคตร รหัสประจำตัว 35104464
2. นางสาววิรุณรยา สุดสน รหัสประจำตัว 35104546
3. นางสาวอลงกต ใจวังโลก รหัสประจำตัว 35104547

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการทดสอบได้

.....

(อ. เทอดศักดิ์ ลิ่วหาทอง)

อาจารย์ที่ปรึกษา

การประยุกต์ใช้งานไมโครโปรเซสเซอร์ DSP ในการทำตัวกรองเชิงเลข

สันติภาพ อุโคตร
วิรุณรยา สุดสน
อลงกต ใจวังโลก
อ. เทอดศักดิ์ ลิ่วหาทอง อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

ระบบประมวลผลสัญญาณเชิงเลข หรือ ดิจิตอลซิกแนลโปรเซสซิง (Digital Signal Processing : DSP) ได้ถูกนำไปประยุกต์ใช้ในงานด้านต่าง ๆ เช่น ระบบควบคุม, ระบบสื่อสารข้อมูล, ระบบประมวลผลสัญญาณภาพ, เครื่องมือวัดต่าง ๆ เป็นต้น

สำหรับปริญญานิพนธ์นี้ เป็นการศึกษาถึงทฤษฎีและการออกแบบตัวกรองสัญญาณเชิงเลขในระบบดิจิตอลซิกแนลโปรเซสซิง โดยนำเอาบอร์ดสำเร็จรูป DSP Starter Kit ซึ่งเป็นบอร์ดที่ใช้สำหรับประมวลผลสัญญาณดิจิตอลโดยเฉพาะมาใช้งาน มี TMS320C50 เป็นซีพียูหลักในการประมวลผล ในการติดต่อกับสัญญาณภายนอกที่เป็นสัญญาณอนาลอกใช้ไอซีเบอร์ TLC32040 และติดต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรมโดยผ่านพอร์ต RS232

ตัวกรองเชิงเลขที่ได้ออกแบบในโครงงานนี้จะออกแบบ โดย มีคอมพิวเตอร์เป็นตัวควบคุม ทางกลุ่มได้พัฒนาโปรแกรมภาษาแอสเซมบลีของ TMS320C50 เพื่อนำไปประยุกต์ใช้งานในการ ออกแบบตัวกรองเชิงเลขเพื่อให้ผู้ใช้สามารถออกแบบวงจรกรองเชิงเลขได้โดยสะดวกตามคุณสมบัติที่ต้องการ

MICROPROCESSOR APPLICATION (DSP) FOR DIGITAL FILTER

Santipap Ukot
Wiroonraya Sudson
Alongkot Jaiwangloke
Therdsak Leewhathong Advisor
1996

ABSTRACT

Digital Signal Processing or DSP is another type of the processing system. By this present time , DSP system has been developed for modern theories and many applications in control system , telecommunication , instrument , image processing , etc.

For this thesis , our project group had studied the theories of DSP system and designed of digital filter , we use the DSP Starter Kit Board that has core cpu TMS320C50 , analog data communication are via the TLC32040 (Analog Interface Circuit) and PC communication are via the RS232 port.

Our group use the assembly language of TMS320C50 for controling the type of digital filter . By this software the user will design the digital filter to have the require specification easily and quickly.

สารบัญ

สารบัญรูป

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

ชุดทดลองการประมวลผลสัญญาณเชิงเลข หรือ ดิจิตอลซิกแนลโปรเซสซิง (Digital Signal Processing : DSP) นี้เป็นการนำเอาความรู้ทางด้านการประมวลผลเชิงเลขมาใช้ในการทำตัวกรอง เชิงเลขแบบ FIR (Finite Impulse Response Digital Filter) หรืออาจจะเรียกได้ว่าเป็นตัวกรองชนิด ไม่มีการป้อนกลับ (Nonrecursive Filter) การออกแบบในโครงงานนี้เลือกใช้แบบ FIR เพราะมีข้อดีอยู่ 3 ประการคือ

1. เป็นระบบที่เสถียร (Stable)
2. เป็นระบบที่เป็นจริง (Realizable)
3. สามารถออกแบบให้เป็นเฟสเชิงเส้นได้ง่าย (Linear Phase)

ชุดทดลองการประมวลผลสัญญาณเชิงเลขนี้ใช้บอร์ด DSK (DSP Starter Kit) มีซีพียูเบอร์ TMS320C50 เป็นตัวประมวลผล ทำการอินเตอร์เฟสกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม และใช้ โปรแกรมบน PC ในการเลือกคุณสมบัติต่างๆ ของตัวกรอง เช่น ตัวกรองความถี่ต่ำผ่าน, ตัวกรอง ความถี่สูงผ่าน, ตัวกรองความถี่ช่วงผ่าน หรือ ตัวกรองความถี่กำจัดช่วง เพื่อให้ DSK กรองสัญญาณ ตามรูปแบบที่ผู้ใช้ต้องการโดยไม่ต้องเขียนโปรแกรมภาษาแอสเซมบลีบน DSK ให้ยุ่งยาก

ชุดทดลองการประมวลผลสัญญาณเชิงเลขนี้ สามารถทำเป็นตัวกรองความถี่ได้ในช่วงของความถี่เสียง (Voice) เพราะมีข้อจำกัดของการแปลงดิจิตอลเป็นอนาลอก (D/A)และอนาลอกเป็นดิจิตอล (A/D) ของไอซีเบอร์ TLC32040 ที่เป็นวงจรอินเตอร์เฟสสัญญาณอนาลอก ที่มีความถี่สุ่ม (Sampling Frequency) ในช่วงความถี่ของเสียง

เมื่อทำการออกแบบตัวกรองสามารถดูผลการออกแบบ เช่น กราฟของผลตอบสนองอิมพัลส์ ฯลฯ ได้บนจอคอมพิวเตอร์ และนำเอาข้อมูลที่ออกแบบไปให้ DSK ประมวลผลตามที่ต้องการ

บทที่ 2

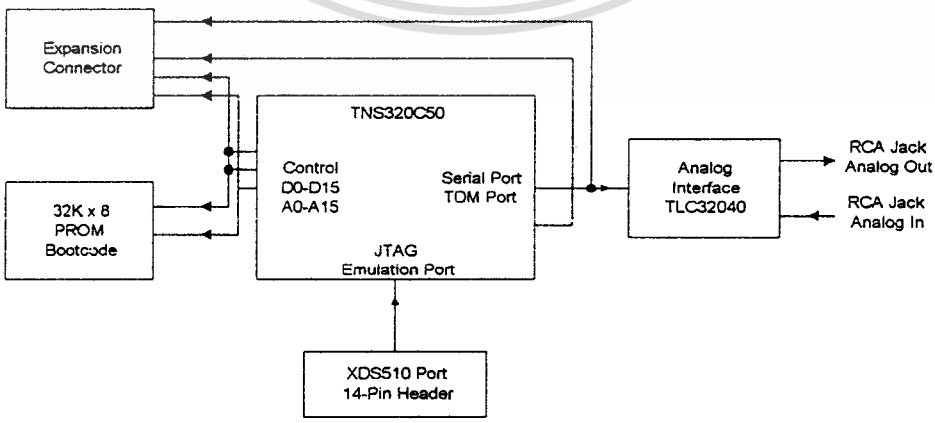
โครงสร้างของ TMS320C50 DSP Starter Kit (50'DSK)

2.1 ลักษณะทั่วไปของบอร์ด

- ใช้โปรเซสเซอร์เบอร์ TMS320C50
- คำสั่งใช้เวลาประมาณ 50 ns (instruction cycle time)
- 32K-byte PROM
- สามารถใช้กับสัญญาณในช่วงความถี่เสียง โดยผ่านทาง TLC32040
- ใช้มาตรฐาน RCA คอนเนคเตอร์ สำหรับ อนาลอกอินพุท และ เอาท์พุท เพื่อให้สามารถต่อเข้ากับไมโครโฟน และ ลำโพงได้โดยตรง
- ในการติดต่อกับการควบคุม มีคอนเนคเตอร์ XDS510 ติดต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม
- สามารถขยาย I/O บัสได้เพื่อใช้สำหรับการออกแบบภายนอก

รูปที่ 2.1 แสดงบล็อกไดอะแกรมของบอร์ด ซึ่งประกอบไปด้วย โฮสอินเทอร์เฟส, อนาลอกอินเทอร์เฟส และ อีมูเลชั่นพอร์ท ทำให้สามารถติดต่อกับ พีซีได้โดยผ่านทาง RS232 นอกจากนี้ยังมี PROM ขนาด 32 Kbytes ที่ใช้เก็บเคอร์แนลโปรแกรมไว้สำหรับการบูต

ส่วนของอนาลอกอินเตอร์เฟส (Analog Interface) ใช้ TLC32040 ซึ่งเป็นวงจรอินเทอร์เฟส สัญญาณอนาลอก (analog interface circuit :AIC) ที่มี RCA คอนเน็กเตอร์ 2 ตัวสำหรับอินพุท และ เอาท์พุท



รูปที่ 2.1 บล็อกไดอะแกรมของ TMS320C5x DSK

จากรูปที่ 2.1 การทำงานเริ่มต้นโดยการบูตโลดเดอร์ (Bootloader) ซึ่งบรรจุบนพีรอมบูตโลดเดอร์(PROM Bootloader) ขนาด 32K เบอร์ 27PC256FM จะทำการบูต(Boot) บอร์ด เป็นการเซ็ตค่าที่จำเป็นสำหรับ TMS ซึ่งใช้เป็นดีบักเกอร์(debugger) ของ DSK

เมื่อทำการรีเซต เริ่มต้นที่ตำแหน่ง 000H โดยขารีเซต หรือที่ $\overline{BR}$ จะเป็นสถานะต่ำ (low) ทำให้เรียกเคอร์เนลโปรแกรม (Kernel program) โดยใช้ขา $\overline{BIO}$ และ XF ของ TMS ในการติดต่อ กับ PC ทาง RS232

ในการติดต่อเมื่อ $\overline{BIO}$ เป็นสถานะต่ำ (low) จะเป็นการแสดงว่าเริ่มติดต่อกับ RS232 ซึ่งจะ เป็นการกำหนดบิตเริ่มต้น สำหรับการคำนวณ โดยเริ่มที่ 1 บิตเริ่มต้น + 7 บิตข้อมูล แล้วหารด้วย 8 จะได้อัตราบอด (Baud Rate) ที่คำนวณจากคำสั่ง NOP ในโครงงานนี้กำหนดอัตราบอดอยู่ที่ 9600 bit/sec

2.1.1 วงจรส่วน CPU TMS320C50

ขา$MP/\overline{MC}$ ของ TMS320C50 ต่อลงกราวด์เพื่อให้อยู่ในโหมดไมโครคอมพิวเตอร์ ในส่วนของอินเตอร์รัพต์ $\overline{INT1}$ - $\overline{INT4}$ และ *NMI* จะแอคทีฟในสถานะต่ำ (low) ต้องมีตัวต้านทาน ต่อเข้ากับไฟเลี้ยงไว้ ส่วนของการติดต่อกับ RS232 จะต่อที่ $\overline{BIO}$ และ *XF* ซึ่งจะกล่าวอีกทีในการ อธิบายส่วนของการติดต่อกับ PC ส่วนสัญญาณนาฬิกาของระบบจะใช้คริสตอลขนาด 40 MHz ต่อเข้าที่ *CLKIN* ในส่วนของ *CLKOUT1* จะต่อไปยัง TLC32040

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

ในการติดต่อกับ TLC32040 เป็นการติดต่อแบบอนุกรมโดยมีขา *DR* สำหรับรับข้อมูล และ ขา *DX* สำหรับส่งข้อมูลโดยกำหนดสัญญาณที่ *CLKR* และ *CLX* ในการติดต่อจะติดต่อเป็นเฟรม โดยมีการกำหนดเฟรมที่ *FSR* และ *FSX* กำหนดเฟรมในการรับและเฟรมในการส่งตามลำดับ

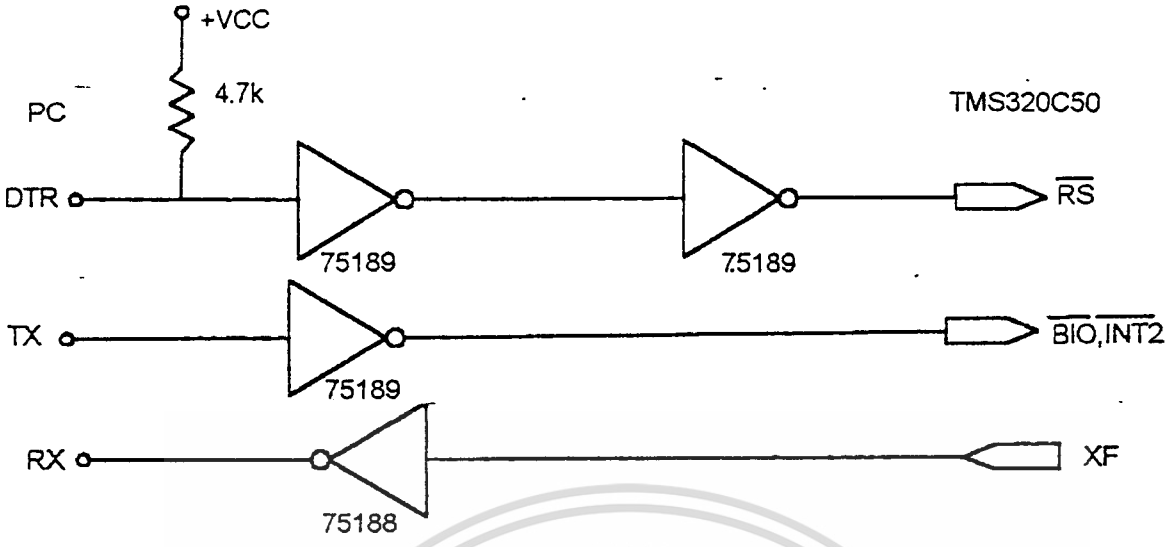
2.1.2 วงจรส่วนการติดต่อกับ RS232

การติดต่อกับ PC ผ่านทางพอร์ตอนุกรม ในโครงงานนี้จะใช้ COM2 มีขั้นตอนดังนี้ ข้อมูล จาก พีซี จะเข้าที่ขา *TX* ผ่าน UTA ซึ่งเป็นเบอร์ 75189 เพื่อเปลี่ยนสัญญาณ RS232 มาเป็นสัญญาณ TTL มาต่อกับ $\overline{INT2}$ และ $\overline{BIO}$ เข้าที่ TMS320C50 อีกที

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

ส่วนขา*DTR* จะใช้เป็นขาสำหรับรีเซตบอร์ด ซึ่งมี 75189X2 สำหรับเปลี่ยนระดับสัญญาณ

การต่อเพิ่มเติมในส่วนของการติดต่อทาง RS232 จะมีการต่อดังรูปที่ 2.2



รูปที่ 2.2 การติดต่อกับ RS232

โดยที่

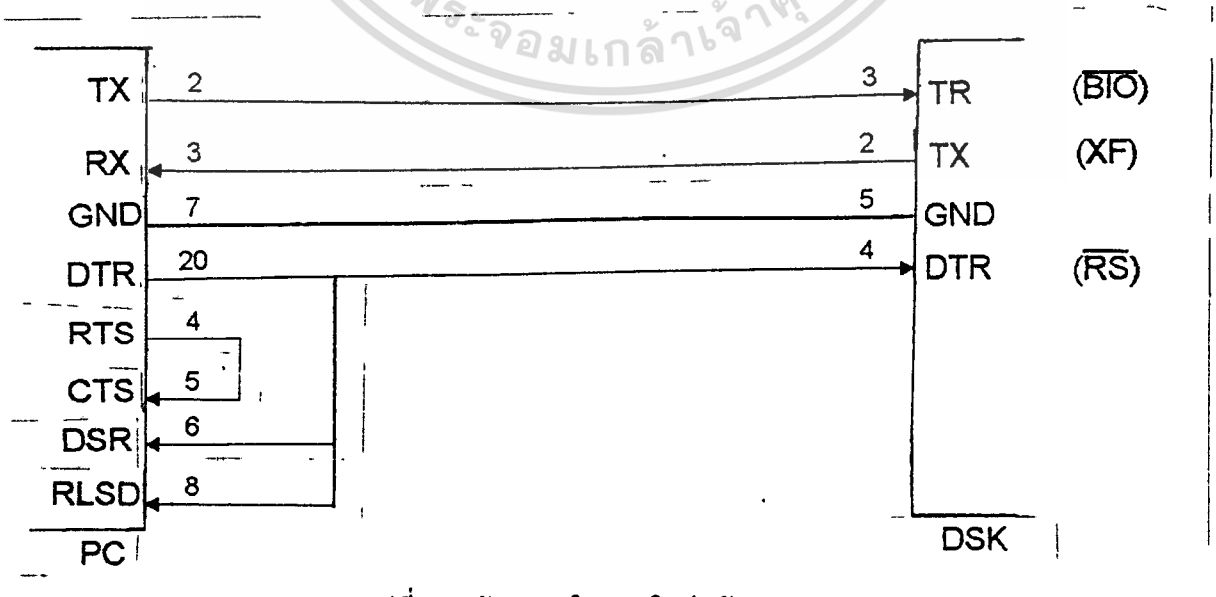
$\overline{CTS}$: clear to send

$\overline{DSR}$: data set ready

$\overline{DTS}$: request to send

DTR : data terminal ready

การรับสัญญาณจาก RS232 มาเป็น TTL และ จาก TTL มาเป็น RS232 จะใช้ IC 75189 และ 75188 ตามลำดับ โดยที่ปกติแล้ว ระดับ RS232 +12V จะเป็น logic 0 และ -12V จะเป็น logic 1 ซึ่งจะต่อดังรูปที่ 2.3



รูปที่ 2.3 สัญญาณในการติดต่อกับ RS232

จากผลการทดลองเมื่อไม่มีการรับส่งจะได้

TX เป็น (-10V) logic 1

TR เป็น (-10V) logic 1

$\overline{DTR}$ เป็น (+10V) logic 0

$\overline{DSR}$ เป็น (+10V) logic 0

2.1.3 วงจรส่วนที่ติดต่อกับ TLC32040

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

ในการติดต่อกับสัญญาณอนาลอกภายนอกจะเข้ามาทาง *AIN+* หรือ *AUXIN±* เข้ามายัง TLC32040 ในการส่งข้อมูลอนาลอก จะส่งออกทาง *OUT+, OUT-*

2.1.4 วงจรส่วนที่ติดต่อกับหน่วยความจำ

เนื่องจากบอร์ดมี PROM เบอร์ 27PC256 ซึ่งเก็บบูตโหลดเดอร์ไว้ และส่วนของการติดต่อจะใช้บัสแอดเดรส A0-A14 และบัสข้อมูล D0-D7 ในการอ่านใช้ขา $\overline{RD}$ และ $\overline{OE}$ โดยการเลือก ชิปที่ $\overline{CE}$ ทาง $\overline{BR}$

ในส่วนของ RAM ที่ใช้ จะใช้ภายใน TMS320C50 เอง ซึ่งมีขนาด 9K สำหรับหน่วยความจำโปรแกรม และ 9Kสำหรับหน่วยความจำข้อมูลซึ่งจะอธิบายอีกครั้งในหัวข้อหน่วยความจำ

2.1.5 วงจรส่วนไฟเลี้ยงและหัวต่อขยายบอร์ด

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

2.2 ข้อกำหนดต่างๆที่ใช้ในบอร์ด

2.2.1 RAM พื้นที่ B2 เก็บ ซีพียูรีจิสเตอร์(CPU register)

2.2.2 SARAM ขนาด 384 เวิร์ด เก็บค่าที่ใช้ติดต่อทาง RS232

2.2.3 ซีพียู จะต้องอยู่ในโหมดไมโครคอมพิวเตอร์ เพราะจะใช้ติดต่อหรือบุตโหลดเดอร์ที่ โปรแกรมภายใน

2.2.4 สงวน $\overline{INT2}$ สำหรับติดต่อกับ RS232 ที่ขา $\overline{BIO}$

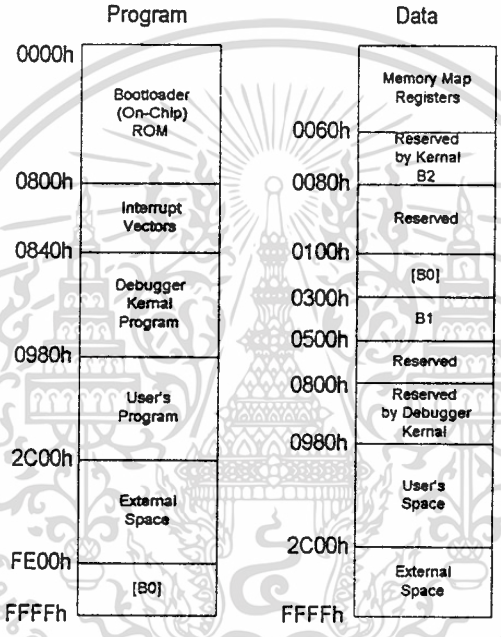
2.2.5 ซอฟแวร์อินเตอร์รัพต์(TRAP)จะใช้สำหรับซิงเกิ้ลสเตป(single step) ของดีบักเกอร์

หน่วยความจำใน C50' DSK นั้น จะแบ่งออกเป็นหน่วยความจำสำหรับโปรแกรม และหน่วยความจำสำหรับข้อมูล ซึ่งในแต่ละแอดเดรสก็จะถูกใช้งานต่างๆ ตามรูปที่ 2.4 และผู้ใช้สามารถใช้หน่วยความจำภายนอกขนาด 10K ได้โดยไม่จำเป็นต้องต่อภายนอก

TLC32040 บนบอร์ด จะมีลักษณะดังนี้

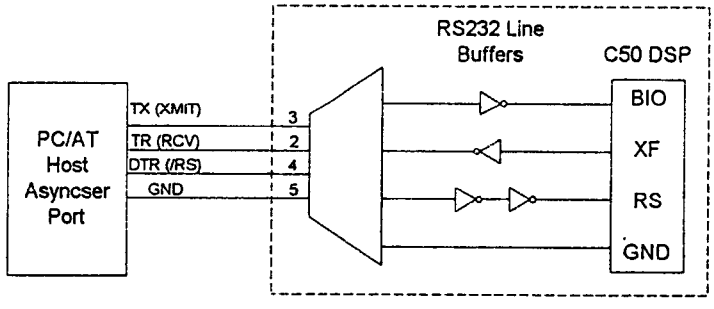
- ใช้ในการแปลง A/D และ D/A ขนาด 14 บิต
- สามารถเปลี่ยนแปลงการแซมปลิ้งทั้งหมดของ A/D,D/A และความถี่ฟิลเตอร์ได้

การติดต่อระหว่าง AIC และ TMS320C50 จะผ่านทางพอร์ตอนุกรมภายในบอร์ด



รูปที่ 2.4 หน่วยความจำภายใน C'50 DSK

การต่อใช้งาน C50 DSK กับ PC จะใช้ขา XF และ BIO ที่ต่อผ่านพอร์ตอนุกรม RS232 ดังแสดงในรูปที่ 2.5



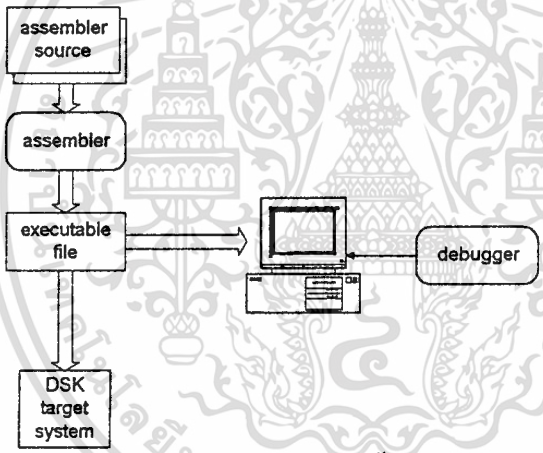
รูปที่ 2.5 การต่อระหว่าง DSK และ PC โดยผ่านทาง RS232

นอกจากนี้ DSK ยังมีแอสเซมเบอร์ และ ดีบักเกอร์ ของมันเอง ซึ่งจะมีประโยชน์มาก เนื่องจากภาษาแอสเซมบลีที่ใช้ใน DSK นั้นจะสนับสนุนการทำงานด้านประมวลผลสัญญาณ (Signal Processing) และ ดีบักเกอร์ก็มีความสามารถที่จะทำงานในแบบทีละขั้น (single-step) และเบรคพอยท์เพื่อช่วยในการแก้ไขโปรแกรม

2.3 การสร้างโปรแกรมเพื่อใช้กับ DSK

1. ทำการสร้างไฟล์หลัก (source file) สำหรับโปรแกรม เช่น exam.asm
2. แปลงไฟล์หลัก โดยใช้ DSK แอสเซมเบอร์ ซึ่งมีคำสั่งดังนี้

   dsk5a exam.asm
3. หากต้องการตรวจแก้ไขโปรแกรมโดยใช้ดีบักเกอร์ ก็ทำได้โดยใช้โปรแกรม dsk5d.exe ซึ่งได้ให้มากับบอร์ดแล้ว



รูปที่ 2.6 แสดงขั้นตอนการสร้างโปรแกรม

บทที่ 3

TMS320C50

TMS320C50 เป็นโปรเซสเซอร์ที่ทำงานด้าน DSP (Digital Signal Processing) ซึ่งตระกูล TMS320 จะมีอยู่หลายเบอร์ เช่น TMS320C50,TMS320C51 และ TMS320C53 ซึ่งเป็นการนำเอาสถาปัตยกรรมของ 'C25 มาปรับปรุงให้ดีขึ้น ทั้งในด้านความเร็วและความสะดวกอื่น ๆ

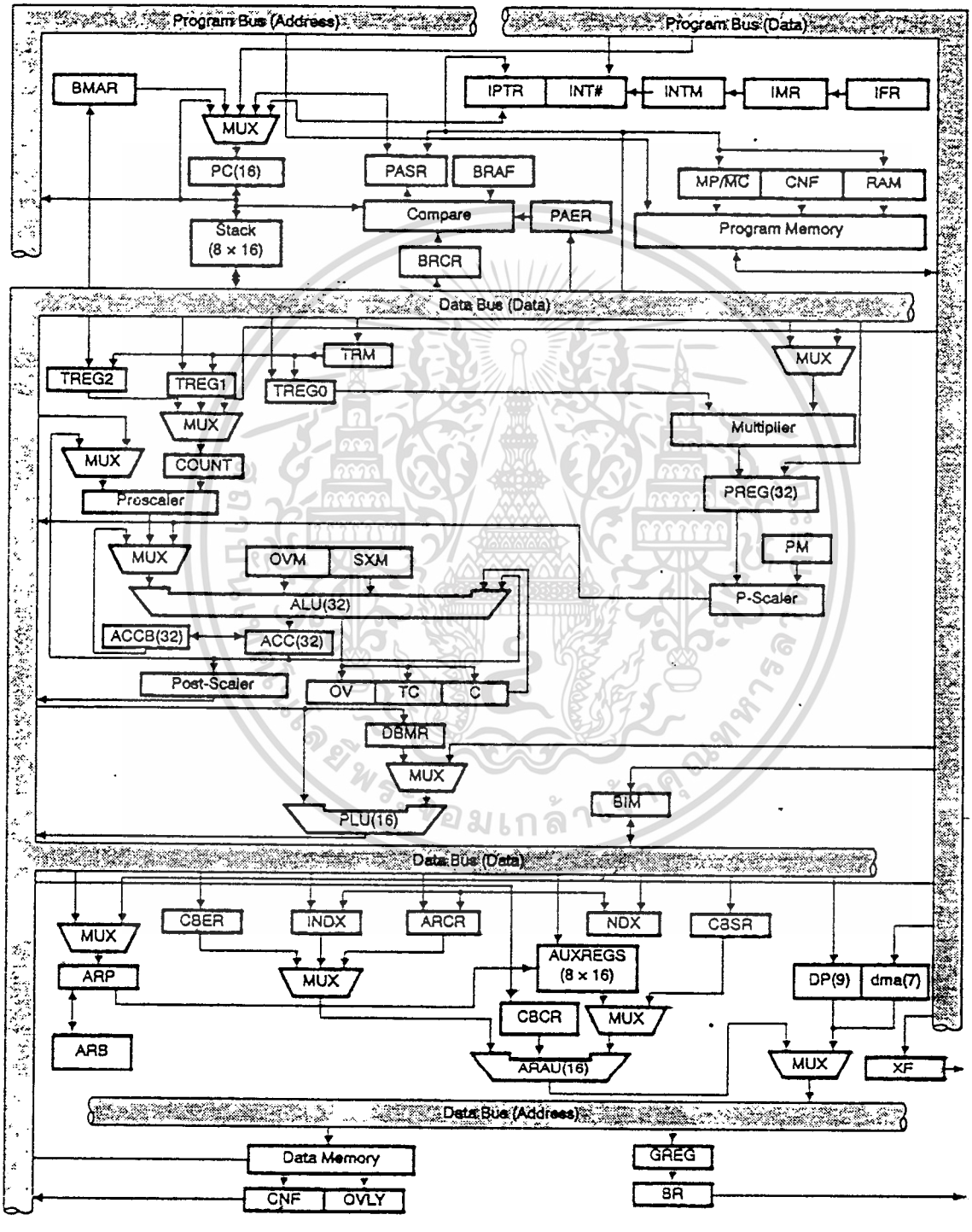
3.1 จุดเด่นของTMS320C50

1. มีแรมบนบอร์ด 10K เวิร์ด
2. ทำงานได้รวดเร็ว คือ 30-50 ns ต่อหนึ่งคำสั่ง
3. ใช้งานแทน 'C1X และ 'C2X ได้
4. มีแรม สำหรับ โปรแกรม/ข้อมูล ขนาด 9K x 16 บิต
5. มีรอมขนาด 2K x 16 บิต สำหรับการบูต
6. สามารถต่อหน่วยความจำข้างนอกได้ถึง 224K x 16 บิต ซึ่งประกอบด้วย หน่วยความจำสำหรับเก็บโปรแกรม 64K หน่วยความจำเก็บข้อมูล 64K สำหรับ I/O และอื่น ๆ อีก 64K
7. มี ALU (Arithmetic Logic Unit) ACC (accumulator) และ ACCB (accumulator buffer) ขนาด 32 บิต
8. มี PLU (Parallel Logic Unit) ขนาด 16 บิต
9. มีคำสั่งในการคูณ 16 บิต ที่ทำงานใน 1 ไซเกิล
10. มีรีจีสเตอร์ถึง 8 ตัว ในการคำนวนและเก็บค่า
11. มีสแต็ก (STACK)
12. มีคำสั่งสำหรับการเลื่อนบิต (shift) ตั้งแต่ 0-16 บิต
13. อ้างแอดเดรสแบบเซอร์คูลาร์(circular)โดยการใช้เซอร์คูลาร์บัฟเฟอร์(circular buffer)
14. มีคำสั่งสำหรับการทำซ้ำโดยเฉพาะสำหรับการเคลื่อนย้ายบล็อก ภายในคำสั่งเดียว
15. มีคำสั่งเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำข้อมูลและหน่วยความจำ โปรแกรม
16. มีพอร์ตอนุกรมที่รับส่งแบบ ฟูลดูเพล็ก (Full duplex) สำหรับ 'C50 กับอุปกรณ์อื่นๆ
17. มี TDM (Time-division multiple) ของพอร์ตอนุกรม
18. สามารถกำหนดสัญญาณนาฬิกา โดยใช้ไทม์เมอร์(Timer), เคาน์เตอร์ (Counter)
19. มีซอฟแวร์ได้ทั้ง หยุด (Stop), เริ่ม (Start) และ รีเซ็ต (Reset)

20. มี พอร์ต I/O ได้ถึง 64K และ มี 16 ตำแหน่ง สำหรับการเข้าถึงหน่วยความจำ

21. ทำงานแบบไปป์ไลน์ (pipeline)

ไดอะแกรมของ TMS320C50 แสดงดังรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมของ TMS320C50

22. สามารถผลิตสัญญาณนาฬิกา โดยการหารสัญญาณนาฬิกาที่จ่ายให้ซีพียู

23. ใช้เทคโนโลยีของ ซีมอส (CMOS) และใช้ไฟเลี้ยงเพียง 5V

สถาปัตยกรรมของ TMS320C50 สร้างขึ้นมาเพื่อความเร็วในการทำงานด้าน DSP และเพื่อให้การทำงานของบัสไม่ขึ้นต่อกัน จึงแยกเป็น บัสโปรแกรม (program bus) และ บัสข้อมูล (data bus) ออกจากกัน โดยบัสโปรแกรมจะเป็นทางเข้าของรหัสคำสั่ง และโอเปอร์แรนด์ของคำสั่ง ส่วนบัสข้อมูลจะเชื่อมต่อโดยตรงระหว่างหน่วยความจำที่ใช้เก็บข้อมูลกับวงจรการทำงานประมวลผล เช่น ALU และ AR0-AR7 ซึ่งโครงสร้างการคำนวนทางคณิตศาสตร์นี้ ยึดหลักการให้ทำงาน ด้วยประสิทธิภาพ เช่น การเลื่อนบิต (shift) การคูณ และ คำสั่งทางลอจิก

3.2 ตำแหน่งขา และหน้าที่การทำงานของ TMS320C50

ตำแหน่งขา และหน้าที่การทำงานของ TMS320C50 แสดงในตารางที่ 3.1

ตารางที่ 3.1 ตำแหน่งขา และหน้าที่การทำงานของ TMS320C50

สัญญาณ	ขา	สถานะ	การทำงาน
			กลุ่มบัสแอดเดรสและบัสข้อมูล(Adress and Data bus)
A15 (MSB)	77	I/O/Z	เป็นบัสแบบขนาน (parallel Address Bus) ใช้สำหรับชี้
A14	76		ตำแหน่งของ หน่วยความจำข้อมูล และ หน่วยความจำ
A13	75		โปรแกรม หรือ I/O ภายนอก เมื่ออยู่ใน โฮลด์โหมด (hold
A12	74		mode) จะเป็นอิมพีแดนซ์สูง (high impedance) สัญญาณ
A11	73		เหล่านี้ใช้เป็นอินพุต สำหรับ DMA ภายนอกของแรมภายใน
A10	72		(On-chip single access RAM) ซึ่งจะเป็นอินพุต เมื่อ
A9	64		$\overline{HOLDA}$ และ $\overline{BR}$ ถูกขับ (drive) ให้เป็นสถานะต่ำ (low)
A8	63		
A7	62		
A6	61		
A5	60		
A4	59		
A3	58		
A2	57		
A1	56		
A0 (LSB)	55		

สัญญาณ	ขา	สถานะ	การทำงาน
D15 (MSB) D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0	6 7 8 9 10 11 12 13 23 24 25 26 27 28 29 30	I/O/Z	เป็นบัสข้อมูลแบบขนาน (parallel data bus) ใช้ส่งผ่านข้อมูลระหว่างซีพียูหลัก (core cpu) กับหน่วยความจำข้อมูล/โปรแกรมภายนอก หรือ อุปกรณ์ I/O เมื่อไม่มีเอาท์พุต สัญญาณเหล่านี้จะเป็นอิมพีแดนซ์สูง หรือเมื่อขา $\overline{RS}$ หรือ $\overline{HOLD}$ อยู่ในสถานะต่ำ (active low) และขา $\overline{OFF}$ เป็นสถานะต่ำ นอกจากนี้ยังใช้สำหรับ DMA ภายนอกของแรม(single access RAM)
			กลุ่มสัญญาณควบคุมหน่วยความจำ
$\overline{DS}$ $\overline{PS}$ $\overline{IS}$	89 91 90	O/Z	เลือกหน่วยความจำข้อมูล/โปรแกรมและ I/O ปกติมีสถานะสูงแต่เมื่อเป็นสถานะต่ำจะเป็นการติดต่อกับภายนอกเมื่อขา $\overline{OFF}$ อยู่ในสถานะต่ำจะอยู่ในสถานะอิมพีแดนซ์สูง
READY	128	I	สัญญาณข้อมูลพร้อม (Data ready input) ใช้แสดงเมื่ออุปกรณ์ภายนอกส่งข้อมูลเรียบร้อยแล้ว และเมื่อยังทำงานไม่เสร็จ (READY=0) จะต้องมีการรอ 1 ไซเคิล (wait 1 cycle) และเช็คขา READY อีกครั้ง ในสภาวะปกติขา READY จะทำงานหลังจากที่มีสัญญาณ $\overline{BR}$
$R/\overline{W}$	92	I/O/Z	สัญญาณอ่าน/เขียน (read/write signal) เป็นสัญญาณควบคุมการอ่านและเขียนข้อมูล จะเป็นสถานะอิมพีแดนซ์สูง เมื่ออยู่ในโฮลด์โหมด ถูกใช้ใน DMA ของแรมภายนอก เมื่อ $\overline{HOLDA}$ และ $\overline{IAQ}$ อยู่ในสถานะต่ำ ใช้แสดงทิศทางของ

สัญญาณ	ขา	สถานะ	การทำงาน
			บัสข้อมูล สำหรับ DMA อ่าน (read อยู่ในสถานะสูง) และเขียน (write อยู่ในสถานะต่ำ)
$\overline{STRB}$	93	I/O/Z	สัญญาณสโตรบ (strob signal) ปกติมักอยู่ในสถานะสูง จะเป็นสถานะต่ำ เมื่อค่าของบัสภายนอกเป็นอิมพีแดนซ์สูง ในโฮล์ดโหมดเมื่อ $\overline{HOLDA}$ และ $\overline{IAQ}$ แอคทีฟสัญญาณนี้จะใช้เลือกการเข้าถึงหน่วยความจำ
$\overline{RD}$	82	O/Z	สัญญาณเลือกอ่าน (read select) ขานี้จะทำงานเมื่อมีการอ่าน จะต่อโดยตรงกับ OE ของอุปกรณ์ภายนอก สัญญาณนี้จะใช้ในการอ่านค่าหน่วยความจำโปรแกรม/ข้อมูลและI/O ภายนอก เป็นอิมพีแดนซ์สูงเมื่ออยู่ในโฮล์ดโหมด
$\overline{WE}$	83	O/Z	สัญญาณเขียน (write enable) จะใช้สำหรับการเขียนค่าในหน่วยความจำโปรแกรม/ข้อมูล และ I/O ภายนอกทั้งหมด เป็นอิมพีแดนซ์สูงเมื่ออยู่ในโฮล์ดโหมด
			กลุ่มสัญญาณมัลติโปรเซสซิง(Multiprocessing)
$\overline{HOLD}$	129	I	สัญญาณโฮล์ด (hold input) เป็นสัญญาณที่ใช้เพื่อแสดงว่ากำลังมีการติดต่อกับบัสตำแหน่ง, บัสข้อมูล และ บัสควบคุม เมื่อถูกตอบรับ (acknowledge) โดย 'C5X จะอยู่ในสภาวะอิมพีแดนซ์สูง
$\overline{HOLDA}$	108	O/Z	สัญญาณตอบรับสัญญาณโฮล์ด(hole acknowledge signal) ใช้แสดงว่าวงจรอยู่ในสถานะโฮล์ด (hold state) ทั้งบัสตำแหน่ง,บัสข้อมูลและบัสควบคุมอยู่ในสภาวะอิมพีแดนซ์สูง
$\overline{BR}$	94	I/O/Z	สัญญาณการขอใช้บัส (bus request signal) แสดงเมื่อมีการติดต่อกับหน่วยความจำข้อมูล สัญญาณจากขานี้ใช้กับหน่วยความจำข้อมูลที่ว่างได้ 32k เวิร์ดเมื่อขา $\overline{HOLDA}$ อยู่ในสถานะต่ำสัญญาณจากขานี้ใช้กับ DMA ของแรมภายนอก $\overline{BR}$ จะเป็นสถานะต่ำเมื่อติดต่อกับแรมภายนอก
$\overline{IAQ}$	1	O/Z	สัญญาณรับคำสั่ง (instruction acquisition signal) จะแสดงค่าสถานะต่ำ เมื่อมีเข้าถึงคำสั่งที่อยู่บนแอดเดรสบัส ใช้กับ DMA ของแรมภายนอกเมื่อ $\overline{HOLDA}$ อยู่ในสถานะต่ำ

สัญญาณ	ขา	สถานะ	การทำงาน
$\overline{BIO}$	130	I	สัญญาณควบคุมบรานซ์ (brance control input) ถ้าเป็นสถานะต่ำจะเป็นการให้ทำคำสั่งที่เป็นเงื่อนไขสัญญาณนี้จะทำงานเมื่อมีการนำคำสั่ง (fetch) ที่เป็นเงื่อนไข
XF	109	O/Z	สัญญาณติดต่อภายนอก (external flag output) ถูกเซตให้เป็นสถานะสูงหรือสถานะต่ำ . โดยคำสั่งพิเศษหรือโดยโหลดค่าใน สเตตรีจิสเตอร์ (state register (ST1)) เมื่อ รีเซต ขานี้จะเป็นสถานะสูง
$\overline{IACK}$	112	O/Z	สัญญาณตอบรับอินเทอร์รัพต์(interrupt acknowledge signal) แสดงค่าเมื่อรับค่าอินเทอร์รัพต์และโปรแกรมเคาท์เตอร์ จะนำค่า อินเทอร์รัพท์เวกเตอร์ (interrupt vector) ซึ่งกำหนด โดย A15-A0
			การอินนิเชียลไลซ์(Initialization),อินเตอร์รัพต์,คำสั่งรีเซต
INT4 INT3 INT2 INT1	41 40 39 38	I	สัญญาณ อินเทอร์รัพต์ จากผู้ใช้ภายนอก (external user interrupt input) กำหนดโดยรีจิสเตอร์ควบคุมอินเทอร์รัพต์ (interrupt mask register) และ บิตอินเทอร์รัพต์โหมด (interrupt mode bit) สามารถรีเซตผ่าน รีจิสเตอร์บอกอินเทอร์รัพต์ (interrupt flag register)
$\overline{NMI}$	42	I	สัญญาณนอน-มาร์กอะเบิลอินเทอร์รัพต์ (non maskable interrupt) เป็นอินเทอร์รัพต์ภายนอก ไม่สามารถควบคุมโดย INTM หรือ MR เมื่อ $\overline{MNI}$ ทำงานจะมีการนอินเทอร์รัพต์
MC / $\overline{MP}$	5	I	ขาเลือกโหมดไมโครโปรเซสเซอร์/ไมโครคอมพิวเตอร์ (microprocessor/microcomputer mode select pin)ถ้าเป็นสถานะต่ำ (microcomputer mode) จะทำให้โปรแกรมรอมภายในส่งไปยังหน่วยความจำโปรแกรม (program memory space)ในโหมดไมโครโปรเซสเซอร์
			สัญญาณออสซิเลเตอร์/ไทม์เมอร์ และ CLKIN1/2
CLKOUT1	110	O/Z	สัญญาณนาฬิกาส่งออก (master clock output signal หรือ CLKIN2 frequency) มีค่าไซเคิลเท่ากับอัตราแมชชีนไซเคิล (machine-cycle) ของซีพียู

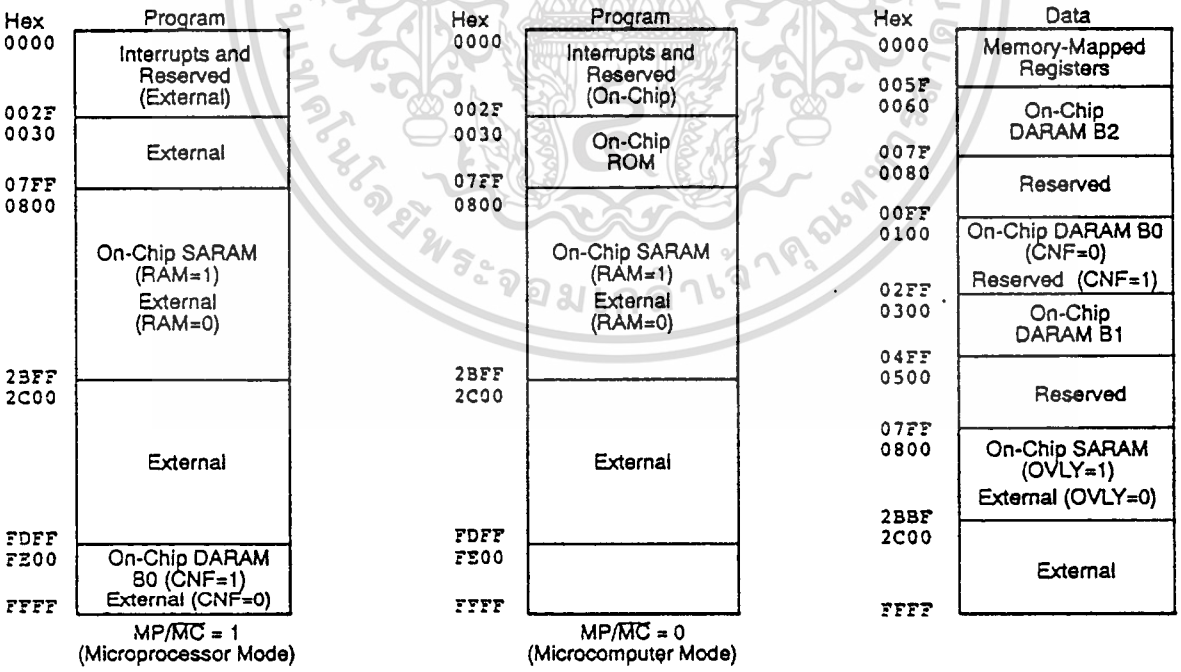
สัญญาณ	ขา	สถานะ	การทำงาน
CLKMD1 CLKMD2	71 103	I	CLKMD1 CLKMD2 0 0 สัญญาณนาฬิกาภายนอก เป็นสัญญาณนาฬิกาเข้าจากขา X2/CLKIN ทำให้ออสซิลเลเตอร์ (oscillator) ภายใน และ PLL disable 0 1 สำหรับตรวจสอบ 1 0 เป็นสัญญาณนาฬิกาอินพุต (input clock)สำหรับ CLKIN2 ทำให้ออสซิลเลเตอร์ภายในหยุดทำงาน และ PLL ทำงานแทน 1 1 เป็นสัญญาณนาฬิกาอินพุตสำหรับขา X2/CLKIN1ทำให้ออสซิลเลเตอร์ภายในทำงานและ PLL ภายในไม่ทำงาน
X2/CLKIN	122	O	ขาอินพุตสำหรับออสซิลเลเตอร์ภายใน (input pin to internal oscillator from crystal) ถ้าออสซิลเลเตอร์ภายในไม่ถูกใช้ สัญญาณนาฬิกาจะเป็นอินพุตสำหรับอุปกรณ์บนขานี้ แมชชีนไซเคิลภายในเป็นครึ่งหนึ่งของอัตรา CLK
X1			เป็นขาเอาท์พุตของออสซิลเลเตอร์ภายในสำหรับคริสตัล ถ้าไม่ใช้ออสซิลเลเตอร์ภายในจะไม่มีการต่อกับขานี้
CLKIN2			เป็นอินพุตสำหรับสัญญาณนาฬิกาสำหรับขับอัตราแมชชีน (machine rate)
TOUT			เอาท์พุตไทม์เมอร์ (timer output) ขานี้ให้สัญญาณพัลส์เมื่อไทม์เมอร์ภายใน (on-chip timer) นับถึง 0 ความกว้างพัลส์เท่ากับ CLKOUT 1 ไซเคิล
			สัญญาณพอร์ตอนุกรม
CLKR CLKRT	46 126	I I	เป็นขาที่รับสัญญาณนาฬิกาจากข้างนอกเพื่อกำหนดให้การรับข้อมูล (DR/TDR) เข้าไปเก็บไว้ที่ RSR (serial port reccive shift register) แต่ถ้าขานี้ไม่ใช้สามารถที่จะใช้เป็นขาอินพุต ของ IN0 ของ SPC/TSPC รีจิสเตอร์ได้

สัญญาณ	ขา	สถานะ	การทำงาน
CLKX TCLKX	124 123	I/O/Z I/O/Z	เป็นขาที่แสดงสัญญาณนาฬิกาจากข้างนอกเพื่อกำหนดให้ DR/TDR ส่งข้อมูลไปที่ DX/TDX CLKX จะเป็นอินพุต ถ้า MCM บิต ที่การควบคุมพอร์ตอนุกรม (serial port control) มีค่าเป็น 0 และอาจจะขับความถี่เป็น $\frac{1}{4}$CLKOUT1 เมื่อ MCM เป็น 1 ถ้าขานี้ไม่ใช้สามารถที่จะทำเป็นอินพุตของบิต IN1 ของ SPC/TSPC รีจิสเตอร์
DR TDR	43 44	I I	เป็นขาเพื่อรับสัญญาณข้อมูล ซึ่งเมื่อรับมาแล้วจะเก็บไว้ที่ RSR (serial port receive shift register)
DX TDX	106 107	O/Z	เป็นขาเพื่อส่งสัญญาณข้อมูล ซึ่งข้อมูลจะส่งจาก XSR (serial port transmit shift register)
FSR TFSR/TADD	45 125	I I/O/Z	แสดงสัญญาณการพร้อมของเฟรม (Frame synchronization) สำหรับรับสัญญาณอินพุต TFSR จะเป็นได้ทั้งอินพุต/เอาท์พุต(TADD)เมื่อพอร์ตอนุกรมอยู่ในโหมด TDM
FSX TFSX/TFRM	104 105	I/O/Z I/O/Z	แสดง สัญญาณ การพร้อม ของเฟรม (Frame Synchronization)สำหรับส่งสัญญาณขานี้จะเลือกได้โดยทางซอฟแวร์ จะเป็น เอาท์พุตเมื่อ TSM ถูกเซตให้เป็น 1
TCK	34	I	สัญญาณนาฬิกาตรวจสอบ JTAG (JTAG test clock) เป็นสัญญาณนาฬิกาแบบฟรี-รันนิ่ง(free-running)ซึ่งมีค่าดิวตี้ไซเคิล (duty cycle 50%)การเปลี่ยนTAP (test access port)ซึ่งเป็นอินพุตสัญญาณนาฬิกาจะควบคุมTAP รีจิสเตอร์คำสั่ง(Instruction registe) หรือเลือกการทดลองรีจิสเตอร์ข้อมูล(data register)ที่ขอบขาขึ้นของTCKในการเปลี่ยนTAPจะเป็นสัญญาณเอาท์พุตจะปรากฏที่ขอบขาลงของ TCK
TDI	67	I	เป็นสัญญาณนาฬิกาเพื่อเลือกรีจิสเตอร์ในขอบขาขึ้นของ TCK

สัญญาณ	ขา	สถานะ	การทำงาน
TDO	100	O/Z	เป็นการทดสอบเอาท์พุทข้อมูลในขอบขาลงของ TCK
TMS	31	I	เป็นการเลือกโหมดทดสอบ JTAG และเป็นสัญญาณนาฬิกาอินพุตที่ทดสอบพอร์ตตรวจสอบการเข้าถึง (test access port (TAP)) จะทำงานที่ขอบขาขึ้นของ TCK
$\overline{TRST}$	2	I	ทดสอบการรีเซตจะเป็นสถานะสูง
EMU0 $EMU1/\overline{OFF}$	118 119	I/O/Z	ขาอิมูเลเตอร์ 1 / ขาหยุดการทำงานของเอาท์พุต(emulator pin 1 / disable all output) จะทำงานที่สถานะต่ำ

3.3 การจัดหน่วยความจำของ TMS320C50

TMS320C50 จะมีการแบ่งหน่วยความจำเป็นส่วนของหน่วยความจำโปรแกรม และหน่วยความจำข้อมูล ตามรูป 3.2



รูปที่ 3.2 การเข้าถึงหน่วยความจำ ของ TMS320C50

3.3.1 หน่วยความจำข้อมูล (Data memory)

ในการออกแบบของ 'C5X จะใช้สถาปัตยกรรมแบบฮาร์วาร์ด (Harward) กล่าวคือ มีการแยก หน่วยความจำข้อมูล และ หน่วยความจำโปรแกรมออกจากกัน ซึ่งทำให้แยกส่วนอย่างชัดเจนซึ่งมีผลดีต่อความเร็ว

หน่วยความจำข้อมูล ของ 'C5X สามารถขยายได้ถึง 64K x 16 บิต ซึ่ง 'C51 มีแรมบนชิพถึง 9K ซึ่งแรมจะเป็นแบบ SARAM (single-access RAM) และ DARAM (dual-access RAM) ขนาด 1056 เวิร์ด บนชิพ ซึ่งมีข้อดี คือ

1. มีการปฏิบัติการได้เร็วไม่มีการรอ (wait state)
2. ทำงานโดยวิธีการไปป์ไลน์ ทำให้ทำงานได้รวดเร็ว
3. ช่วยลดค่าใช้จ่ายเรื่องการต่อหน่วยความจำภายนอก
4. ช่วยประหยัดไฟ ซึ่งถ้าให้การต่อหน่วยความจำภายนอกจะเปลืองกว่า

ในการกำหนดการใช้หน่วยความจำข้อมูลจะต้องมีการเซ็ตค่าของบิตของรีจิสเตอร์ควบคุมหน่วยความจำ (Control state Register) ดังนี้

ตารางที่ 3.2 การเซ็ตบิตรีจิสเตอร์ควบคุมหน่วยความจำเพื่อใช้หน่วยความจำข้อมูล

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

เนื่องจากการติดต่อกับรีจีสเตอร์ต่าง ๆ และการส่งค่าต่าง ๆ จะทำบน เพจ 0 ซึ่งจะเป็นการใช้แบบส่งแบบรีจิสเตอร์ (Register Memory Map) ดังตารางที่ 3.3

ตารางที่ 3.3 ตำแหน่งแอดเดรสข้อมูล เพจ 0

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

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

3.3.2 หน่วยความจำโปรแกรม (Program memory)

ในการใช้งานของหน่วยความจำโปรแกรม ซึ่งสามารถขยายได้ถึง 64K ซึ่ง 'C50 มีรอม, SARAM และ DARAM ซึ่งมีความเร็วสูง โดยไม่มีสภาวะการรอ (wait state)

ในการทำงานของซีพียู 'C50 สามารถใช้ร่วมกับรอมภายในขนาด 4K ซึ่งสามารถ โปรแกรม จากโรงงาน มีความเร็วในการทำงานเต็มที่ ในการเลือกใช้จะต้องกำหนดที่ขาของ

MP/$\overline{MC}$ หากขาดังกล่าว เป็นสถานะสูง ตำแหน่ง 4K แรกจะเป็นหน่วยความจำภายนอกชิพ แต่ถ้าเป็นสถานะต่ำ ตำแหน่ง 4K เวิร์ดแรกจะเป็นรอมภายในชิพ ดังรูปที่ 3.2

การกำหนดสถานะ (status) ของค่าต่าง ๆ จะกำหนดตามตารางที่ 3.4

ตารางที่ 3.4 การกำหนดค่าสำหรับใช้หน่วยความจำโปรแกรม

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

การใช้งานของการอินเตอร์รัพท์ จะต้องกำหนด เวกเตอร์ของแอดเดรสในหน่วยความจำโปรแกรม ซึ่ง อินเตอร์รัพท์เวกเตอร์ จะแสดงดังตารางที่ 3.5

ตารางที่ 3.5 ตำแหน่งของอินเตอร์รัพท์เวคเตอร์

Name	Location		Priority	Function
	Dec	Hex		
$\overline{RS}$	0	0	1 (highest)	External reset signal
$\overline{NMI}$	36	24	2	Nonmaskable interrupt
$\overline{INT1}$	2	2	3	External user interrupt #1
$\overline{INT2}$	4	4	4	External user interrupt #2
$\overline{INT3}$	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
$\overline{INT4}$	18	12	11	External user interrupt #4
——	20–33	14–21	N/A	Reserved
TRAP	34	22	N/A	Trap instruction vector
——	38–39	26–27	N/A	Reserved
——	40–63	28–3F	N/A	Software interrupts

3.4 โหมดการอ้างถึงหน่วยความจำ (Memory Addressing Mode)

การอ้างแอดเดรสสามารถอ้างได้ 64k สำหรับหน่วยความจำโปรแกรม และ 96k สำหรับหน่วยความจำข้อมูล 'C50 มีวิธีการอ้างแอดเดรส ได้ถึง 8 วิธีดังนี้

1. โดยการเข้าถึงโดยตรง (Direct address bus)โดยการใช้แอดเดรสบัสโดยตรง ซึ่งต้องมีรีจิสเตอร์ชี้เพจของข้อมูล (data page pointer (DP)) บอกด้วยว่าเป็นเพจใด DP สามารถอ้างได้ 512 เพจ แต่ละ เพจ = 128 เวิร์ด ตัวอย่างเช่น ADD 01 0h หมายถึง เอาข้อมูลที่ชี้ที่ตำแหน่ง 010h (ซึ่งมี DP ชี้บอกว่าอยู่ page ใด) นำมาบวกกับ ACC

2. โดยใช้การเข้าถึงหน่วยความจำ(Memory Map) จะเหมือนกับวิธีแรกแต่จะใช้เฉพาะเพจ 0 เท่านั้น เช่น

LAMM PMST โดยที่ PMST เป็น รีจิสเตอร์

3. โดยใช้ ออกซิลลารี รีจิสเตอร์ (Auxiliary register) โดยการอ้างแอดเดรสที่บรรจุใน AR0-AR7 เช่น ADD* ซึ่ง ARP จะเป็นตัวบอกว่าเป็น AR ตัวที่เท่าใด ตัวอย่างเช่น

ARP	4	4
AR4	0302h	0302h

หน่วยความจำข้อมูล

302h	2h	2h
ACC	2h	4h

จะเห็นว่าแอดเดรสที่ได้อยู่ใน AR4 (0302h)

4. โดยการใช้รีจิสเตอร์คำสั่ง (Instruction Register) ซึ่งเป็นการเอาแอดเดรสโดยตรงมาจากคำสั่งเลย เช่น

ADD # 0FFh เอาข้อมูลที่แอดเดรส FFh มาบวกกับ ACC

โดยวิธีนี้จะอ้างได้แบบสั้น (short immediate) คือ ค่าแอดเดรส จะเป็น 00-FF

5. โดยการใช้ PC (program counter) จะเหมือนการใช้รีจิสเตอร์คำสั่ง แต่สามารถอ้างแบบยาว (Long Immediate (0000-FFFF)) เช่น

ADD # 0FFFh

6. โดยเข้าถึงรีจิสเตอร์โดยตรง (Register Access) เป็นการใช้รีจิสเตอร์ที่ทำงานโดยเฉพาะ เช่น TREG0, TREG1,TREG2, ARCR, DBMR ตัวอย่างคำสั่ง

APL * ,AR7

7. โดยใช้คำสั่งตัวที่ 2 เป็นตัวชี้แอดเดรส เช่น คำสั่ง BLDD # 02345h, 012h หมายถึงเอาข้อมูลที่ชี้ที่แอดเดรส 02345h ไปเก็บที่แอดเดรส 012h

8. โดยอ้างถึงหน่วยความจำเป็นบล็อก (Block Memory Address Register) ซึ่ง BMAR จะเป็นตัวชี้ข้อมูล ตัวอย่างเช่น BLDD BMAR, 012h

3.5 อุปกรณ์เสริม (Peripherals)

TMS320C50 มีการอินเตอร์เฟสกับภายนอกได้ถึง 8 วิธี ซึ่งประกอบด้วย

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

ซึ่งทั้งหมดจะควบคุมผ่านรีจิสเตอร์ควบคุม โดยการใช้การวิธีการอ้างหน่วยความจำ ในการ ใช้งานจะเป็นการใช้โดยการส่งลงที่แอดเดรสของหน่วยความจำโปรแกรม และ ข้อมูล

ในที่นี้จะขอกล่าวเฉพาะหัวข้อที่เกี่ยวข้องกับโครงงานที่ทำดังต่อไปนี้

3.5.1 อินเทอร์รัพต์ (Interrupt)

TMS320C50 มีอินเตอร์รัพท์ทั้งหมด 16 อินเตอร์รัพท์ (INT16-INT1) แต่จะไม่ใช้พร้อมกันทั้งหมด (ปกติใช้ 9)

การรีเซต (Reset)จะเป็นอินเทอร์รัพต์แบบนอน-มาร์กอะเบิล จากภายนอก (nonmaskable external interrupt) ซึ่งจะเกิดขึ้นเมื่อมีการรีเซต ดังนี้

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

0 → OV	1 → XF	1 → SXM	0 → PM	1 → HM
0 → BRAI	0 → TRM	0 → NDX	0 → CENM1	0 → CENM2
0 → IPTR	0 → OVLY	0 → AVIS	0 → RAM	0 → BIG
0 → CNF	1 → INTM	MP/MC (pin) → PMST (MP/MC)		1 → C

5. GREG = 000000

6. RPTC = 00

7. IACK จะแสดงสภาวะการเก็บการรีเซต

ในการควบคุมการอินเตอร์รัพท์จะมี IFR (Interrupt Flag Register) แสดงสภาวะการอินเตอร์รัพท์ และ IMR (Interrupt Mask Register) แสดงการ mask การอินเตอร์รัพท์ ดังรูปที่ 3.3

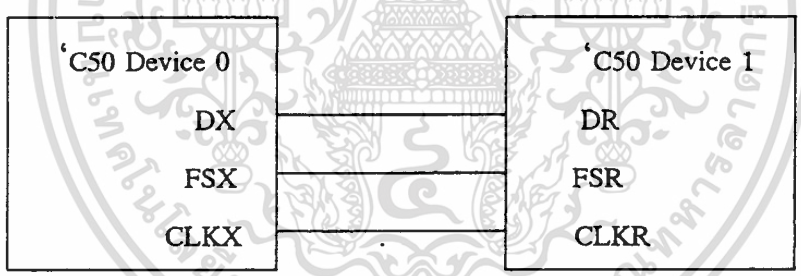
15 9	8	7	6	5	4	3	2	1	0
Reserved	$\overline{INT4}$	*TXNT*	*TRNT*	*XINT*	*RINT*	*TINT*	$\overline{INT3}$	$\overline{INT2}$	$\overline{INT1}$

รูปที่ 3.3 รีจิสเตอร์อินเตอร์รัพท์แฟลก

ในการกำหนดให้ใช้อินเตอร์รัพท์จะต้องทำการเคลียร์ที่ INTM บิต ด้วย

3.5.2 พอร์ตอนุกรม (Serial Port)

Serial port ของ TMS320C50 จะเป็นแบบฟลู-ดูเพล็ก คือ สามารถรับและส่งภายในเวลาเดียวกัน ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกอื่น ๆ ได้ ไม่ว่าจะเป็น CODEC A/D หรือ ระบบอื่น ๆ ในการต่อขาต่าง ๆ จะเป็นตามรูปที่ 3.4



รูปที่ 3.4 แสดงการต่อพอร์ทอนุกรมกับพอร์ทภายนอก

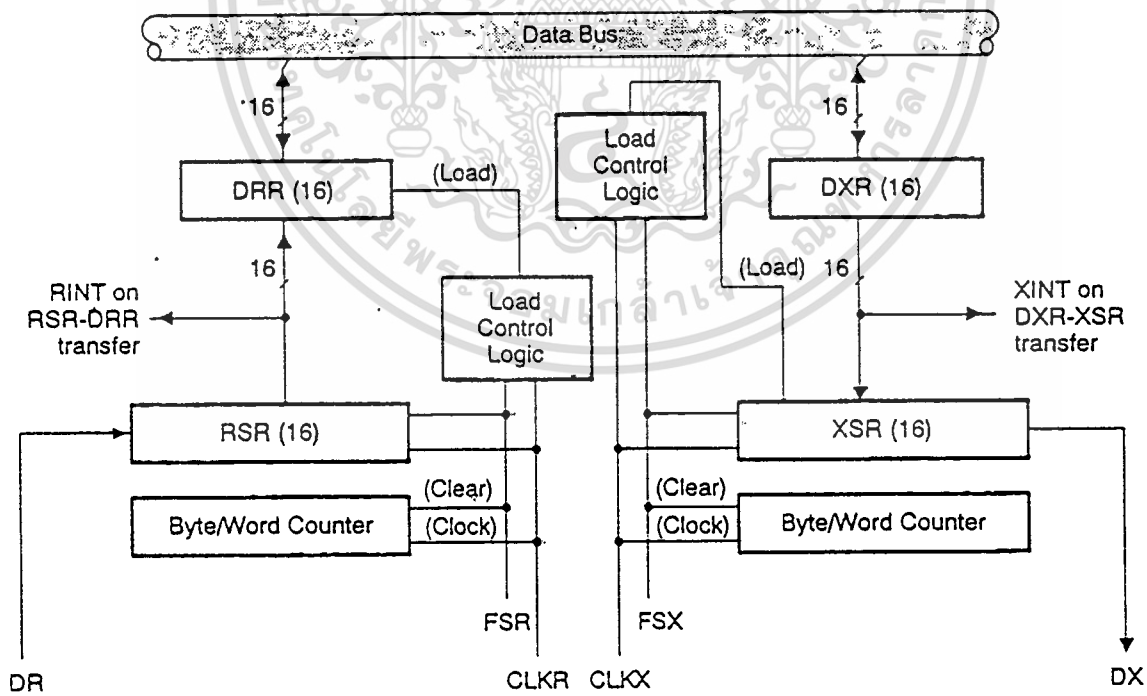
<u>ตารางที่ 3.6</u> ขาสัญญาณของพอร์ทอนุกรม

ขาสัญญาณ	ความหมาย
CLKX	สัญญาณคล็อกที่ใช้ในการส่ง
CLKR	สัญญาณคล็อกที่ใช้ในการรับ
DX	สัญญาณข้อมูลอนุกรมในการส่ง
DR	สัญญาณข้อมูลอนุกรมในการรับ
FSX	สัญญาณการซิงโครไนซ์เฟรมการส่ง
FSR	สัญญาณการซิงโครไนซ์เฟรมการรับ

ในการควบคุมพอร์ตอนุกรมจะมีรีจิสเตอร์ที่จำเป็นตามตารางที่ 3.7

ตารางที่ 3.7 รีจิสเตอร์พอร์ทอนุกรม

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

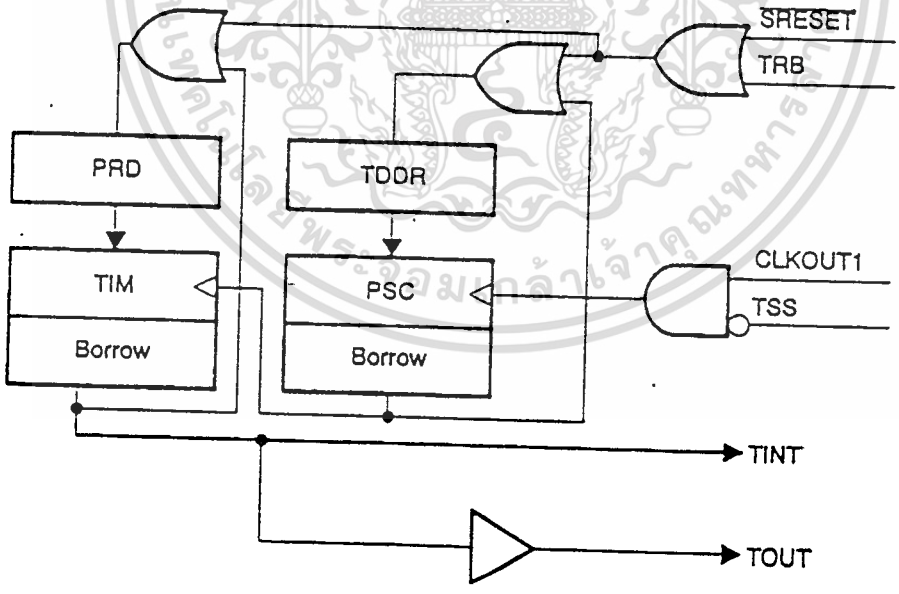


รูปที่ 3.5 บล็อกไดอะแกรมการทำงานของพอร์ทอนุกรม

ในการส่งข้อมูลทำได้โดยการเขียนข้อมูลลงใน DXR (Data transmit register) โดย XSR (Transmit Shift Register) จะเป็นตัวเลื่อนบิตตามสัญญาณนาฬิกาจนครบไปตามขา DX ส่วนการรับข้อมูลจะรับมาทางขา DR เข้ามาที่ RSR (Receive Shift Register) เมื่อครบแล้วจะทำการอินเตอร์รัพท์บอกให้ทราบ แล้วสามารถรับข้อมูลโดยการอ่านข้อมูลจาก DRR (Data Receive Register)

3.5.3 ไทม์เมอร์ (Timer)

ในการใช้ไทม์เมอร์ของ TMS320C50 ซึ่งจะสามารถตั้งคาบเวลาของสัญญาณนาฬิกาได้โดยสัญญาณจะออกที่ขา Tout ของ TMS320C50



รูปที่ 3.6 บล็อกไดอะแกรมของไทม์เมอร์

ซึ่งการคำนวนคาบเวลา จะเป็นตามสูตร

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

ซึ่ง t_c เป็นคาบเวลาของสัญญาณนาฬิกาที่จ่ายให้ที่ขา CLKOUT1

TDDR (Timer Divide Down Ratio)

PRD (Period Register) เป็นรีจีสเตอร์ที่สามารถใส่ค่าคาบเวลาได้

ในการทำงานจะต้องมีการใส่ค่าต่าง ๆ ลงบน TCR (Timer Control Register) ดังตารางที่ 3.8

ตารางที่ 3.8 รีจิสเตอร์ควบคุมไทม์เมอร์ (TCR)

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

3.6 การเขียนโปรแกรมและคำสั่ง

3.6.1 การกำหนดสถานะเริ่มต้นสำหรับ TMS320C50

ทุกครั้งที่มีการใช้งาน TMS320C50 จำเป็นต้องมีการกำหนดสถานะเริ่มต้น(initialize)ตัวซีพียู ก่อนเสมอ ซึ่งต้องทำการเตรียมสภาวะของ โปรเซสเซอร์ให้พร้อม โดยปกติจะทำไว้เมื่อเกิดการรีเซต ซึ่งการรีเซตของ TMS320C50 จะเกิดขึ้นเมื่อขา $\overline{RS}$ อยู่ในสถานะต่ำ ซึ่งจะทำให้ *IPTR* (interrupt pointer) ซึ่งอยู่ใน *PMST* (processor mode status register) เกิดการเคลียร์ ทำให้เกิดการ map เวกเตอร์ไปที่ page 0 ในการเกิด interrupt นั้นเมื่อรีเซตจะทำให้ disable

ในการกำหนดสถานะเริ่มต้นสำหรับโปรเซสเซอร์นั้นจะประกอบด้วย

- การ Map หน่วยความจำและ รีจิสเตอร์ควบคุม
- การเซตโครงสร้างของ อินเตอร์รัพต์
- โหมดควบคุมต่างๆ (OVM, SXM, PM, AVIS, NDX, TRM)
- การควบคุมหน่วยความจำ (RAM, OVLY, CNF)
- กำหนด รีจิสเตอร์ช่วย(Auxiliary register) และ ตัวชี้รีจิสเตอร์ช่วย
- กำหนดตัวชี้เพจข้อมูล (Data page pointer: DP)

การกำหนดค่าใน PMST (Processor mode status register)

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

รูปที่ 3.7 การกำหนดค่าใน PMST

ค่า IPTR เราจะกำหนดที่ IPTR = 10000B หรือเป็นเพจที่ 1 ซึ่งตรงกับโปรแกรมตำแหน่ง 0800h ส่วนการกำหนดการใช้ RAM ภายใน จะต้องใช้ RAM 9K ซึ่งอยู่ในชิป กำหนด RAM, OVLY เป็น "1" RAM ที่ใช้จะเป็น SARAM โดยที่หน่วยความจำโปรแกรมอยู่ที่ 0800h-2BFFh และหน่วยความจำข้อมูลอยู่ที่ 0800h-2BFFh

ในส่วนของการกำหนดอินเตอร์รัพต์เวกเตอร์(interrupt vector) ในโครงงานนี้ มีการกำหนดดังนี้

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

ในโครงงานนี้เรากำหนด interrupt จาก IMR (interrupt mask register) ซึ่งจะกำหนดดังนี้

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

รูปที่ 3.8 การกำหนดค่าใน IMR

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

ตัวอย่าง

```
setc    INTM            ; กำหนด INT1 , XINT
SPLK  #031h, IMR        ; RINT แอคทีฟ
clrc    INTM
```

ในโครงงานนี้ใช้ $\overline{INT2}$, *RINT* , *XINT* แอคทีฟ

3.6.2 การเขียนโปรแกรมทางด้านคณิตศาสตร์และลอจิก (Logic and Arithmetic Operations)

TMS320C50 มี PLU (Parallel Logic Unit) ซึ่งจะเป็นการกระทำทางลอจิก (logic) ที่มีความเร็วมาก ในส่วนนี้มีรีจิสเตอร์ซึ่งใช้ในการประมวลผล ประกอบไปด้วย

1.) สเกลลิงชิฟเตอร์(scaling shifter) 16 บิต

2.) ตัวคูณแบบขนาน16×16 บิต

3.) อริธเมติก ลอจิก ยูนิต (arithmetic logic unit: ALU) 32บิต

4.) แอคคิวมูเลเตอร์ (accumulator :ACC) 32 บิต

5.) แอคคิวมูเลเตอร์บัฟเฟอร์ (accumulator buffer : ACCB) 32 บิต

ซึ่งมีประสิทธิภาพในการประมวลผล ทางด้าน คณิตศาสตร์ และ ลอจิก ที่จำเป็นในการใช้งานทางด้านการประมวลผล ทางดิจิตอล

การใช้สเกลลิงชิฟเตอร์ (scaling shifter) โปรเซสเซอร์ตัวนี้สามารถประมวลผลแล้วทำการเลื่อนบิตได้ ทำให้สะดวกสำหรับการคูณในรูป 2^n ตัวอย่างเช่น

LACC #01111h, 8

ซึ่งเป็นการโหลดค่าเข้า ACC ด้วยค่า 01111h แต่จะเลื่อนไป 8 bit ผลคือ ACC=011110000000B

3.6.3 คำสั่งของ TMS320C50 ที่สำคัญ

ในการใช้งานของโปรเซสเซอร์ TMS320C50 จะต้องเขียนโปรแกรมเป็นภาษาแอสเซมบลี จึงจำเป็นต้องทราบคำสั่งและวิธีใช้ซึ่งมีอยู่มากมาย สามารถสรุปได้ดังหัวข้อต่อไปนี้

1.) คำสั่งที่เกี่ยวกับแอคคิวมูเลเตอร์ และ หน่วยความจำ ซึ่งเป็นคำสั่งที่ใช้บ่อย และสำคัญ

LACC โหลด ACC โดยการชิฟต์เข้ามา

LACL โหลดค่าเข้ามาในเวิร์ดต่ำของ ACC

ซึ่งคำสั่งเป็นการนำค่ามาเก็บไว้ที่แอคคิวมูเลเตอร์ซึ่งมีขนาด 32 bit จะมีทั้งนำมาเก็บทั้ง 32 bit หรือ ขนาด 16 bit คำสั่งจะอำนวยความสะดวกในการใช้งานเพราะมีการเลื่อน ข้อมูลเข้ามาด้วย

SACH เก็บค่าในเวิร์ดสูงของ ACC โดยการชิฟต์เข้ามา

SACL เก็บค่าในเวิร์ดต่ำของ ACC โดยการชิฟต์เข้ามา

จะตรงข้ามกับคำสั่งที่ผ่านมา ซึ่งจะมีทั้ง 32 บิต และ 16 บิต ส่วนอีกคำสั่งหนึ่งเป็นการติดต่อระหว่าง ACC กับ เมมโมรีรีจิสเตอร์แมป (memory register map) คือ

SAMM ส่งค่า ACCL ไปยังเมมโมรีเมปรีจิสเตอร์

LAMM โหลด ACC ด้วยค่าในเมมโมรีแมปรีจิสเตอร์

2.) คำสั่งเกี่ยวกับ รีจิสเตอร์ช่วย และ เพจข้อมูล

LAR โหลด AR

LDP โหลดเพจข้อมูล

MAR ขยายรีจิสเตอร์ช่วย (Modify Auxiliary register)

ซึ่งมีรีจิสเตอร์ช่วย ถึง 8 ตัวคือ AR0-AR7 ในการใช้งานจะต้องกำหนด ARP(Auxiliary Register Pointer) ด้วย ส่วนในการกำหนดเพจ ของข้อมูลมีทั้งหมด 512เพจ เพจละ 128 เวิร์ด ซึ่งจะกล่าวในเรื่องหน่วยความจำอีกครั้งหนึ่ง

3.) คำสั่งเกี่ยวกับ พาราลเลลลอจิกยูนิต (Parallel Logic Unit)

TMS320C50 มีคำสั่งเกี่ยวกับ logic ซึ่งจำเป็นในการใช้งานถึงระดับบิต ที่สำคัญคือ

SPLK เก็บค่าลงบนหน่วยความจำข้อมูล

ซึ่งเป็นคำสั่งการเก็บค่าลงบนหน่วยความจำข้อมูล ซึ่งกระทำในระดับบิต ส่วนมากจะใช้ในการควบคุมบิตต่างๆ ในรีจิสเตอร์ ที่จำเป็นไม่ว่าจะเป็น PMST (Processor mode status register) , IMR (Interrupt mask register) , BRCR (Block repeat countor register) เป็นต้น

4.) คำสั่งในการคูณ

ในการประมวลผล การคูณมีความจำเป็นอย่างมาก TMS320C50 จึงมีคำสั่งที่ใช้รองรับเกี่ยวกับการประมวลผล ทางดิจิตอลไว้มาก ไม่ว่าจะเป็นการ คอนโวลูชัน(convolution),คอร์รีเลชัน (correlation)

LT โหลด TREGO

MACD การคูณ/แอคคิวมูเลต(accumulate) โดยชิฟต์ข้อมูล

ZPR Zero product register

คำสั่ง MACD เป็นการคูณแล้วเลื่อนข้อมูลไปในตัวซึ่งสะดวกในการทำ คอนโวลูชัน

5.) คำสั่งสำหรับการกระโดด

B กระโดดโดยไม่มีเงื่อนไข

BANZ กระโดด ถ้า ARX=0

BCND กระโดดโดยมีเงื่อนไข

CALL	เรียก โดยไม่มีเงื่อนไข
RETI	รีเทอร์นอินเตอร์รัพต์ (Return interrupt)
RET	รีเทอร์น (Return)

จะเห็นว่าคำสั่ง BCND (Branch condition) จะเป็นคำสั่งที่สำคัญเพราะมีเงื่อนไขมากดังนี้

ACC = 0	EQ
ACC ≠ 0	NEQ
ACC < 0	LT
ACC ≤ 0	LEQ
ACC > 0	GT
ACC ≥ 0	GEQ
C = 0	NC
C = 1	C
OV = 0	NOV
OV = 1	OV
$\overline{BIO}$ low	BIO
TC = 0	NTC
TC = 1	TC
Uncondition	UNC

6.) คำสั่งเกี่ยวกับการควบคุมระบบ

จากที่ได้กล่าวในพาราลเลลลอจิกยูนิต (Parallel Logic Unit) ซึ่งกระทำระดับบิต จำเป็นสำหรับการควบคุมระบบแล้ว ยังมีคำสั่งซึ่งเจาะจงระดับบิต มากขึ้น ดังนี้

SETC	เซตบิต
CLRC	เคลียร์บิต
BIT	ทดสอบบิต

ซึ่งคำสั่ง BIT จะใช้มากเพราะมีเงื่อนไขหลายอย่างเช่น BIT 0h,15 เลข 15 จะเป็นตัวกำหนด บิต ซึ่งมีได้ตั้งแต่ 0 ถึง 15 นั้นหมายถึง เราจะทดสอบบิตได้ทุกบิตทุกตำแหน่งในหน่วยความจำด้วย

บทที่ 4

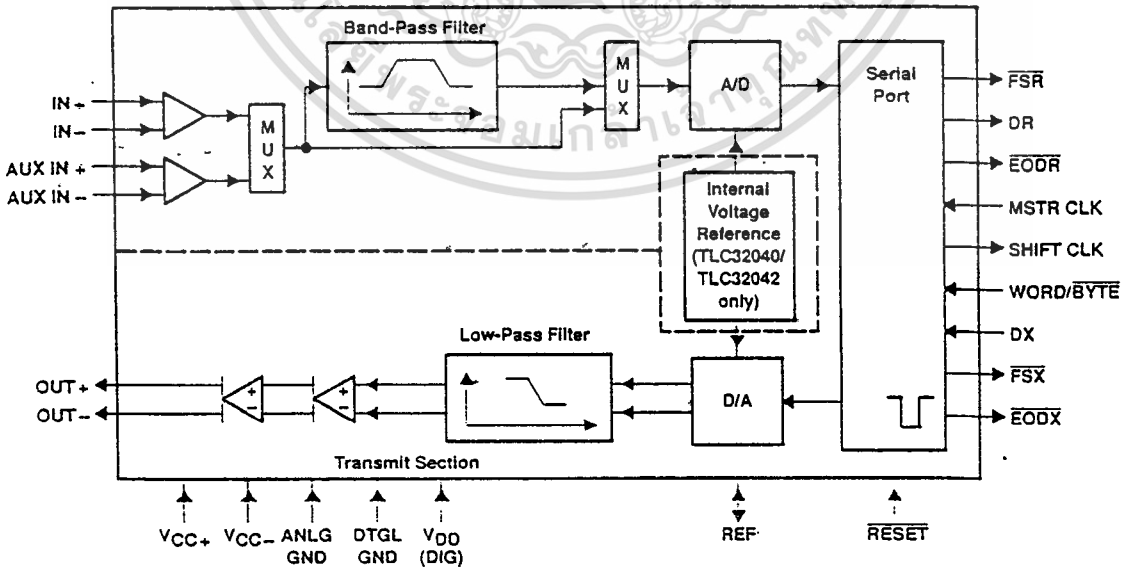
วงจรอินเตอร์เฟสสัญญาณอนาลอก TLC32040

TLC320C40เป็นชิพซีมอส(CMOS Chip)ที่ใช้เชื่อมต่อกับTMS320C50 ในการทำงาน DSP

4.1 ลักษณะสำคัญของ TLC320C40 มีดังนี้

- ใช้เทคโนโลยีการผลิต Advanced LinCMOS™ Silicon-Gate Process
- ความละเอียดของ ADC และ DAC เป็น 14 บิต
- สามารถเปลี่ยนอัตราแซมปลิ้งของ ADC และ DAC ได้ถึง 19,200 ครั้ง/วินาที
- มี Switched-Capacitor Antialiasing Input Filter และ Output-Reconstruction Filter
- มีพอร์ตอนุกรมสำหรับติดต่อโดยตรงกับ TMS3211,TMS320C17,TMS32020 และTMS 320C25 DSP
- สามารถปรับอัตราการแปลงของ ADC และ DAC ทั้งแบบซิงโครนัส หรือ อะซิงโครนัสโดยใช้ซอฟท์แวร์ควบคุม

4.2 ฟังก์ชันบล็อกไดอะแกรม (Function block Diagram)



รูปที่ 4.1 ฟังก์ชันบล็อกไดอะแกรมของ TLC32040

4.3 ตำแหน่งขาและหน้าที่การทำงานของแต่ละขา

ตารางที่ 4.1 แสดงตำแหน่งขาและหน้าที่การทำงานของแต่ละขา

ชื่อ	หมายเลข	I/O	คำอธิบาย
ANLG GND	17,18		กราวด์อนาลอก (แยกกับกราวด์ดิจิตอล)
AUX IN+	24	I	นอน-อินเวอร์ติงออกซิลลารีอินพุท(Noninverting auxilary Input)
AUX IN-	23	I	อินเวอร์ติง ออกซิลลารีอินพุท (Inverting auxilary Input)
DGTL GND	9		กราวด์ดิจิตอล
DR	5	O	ใช้สำหรับส่ง เอาท์พุท ADC จาก AIC (analog interface circuit) ไปยัง 'C50 ผ่านทางพอร์ตอนุกรม(ต้องซิงค์กับ SHIFT CLK)
DX	12	I	ใช้สำหรับรับอินพุท DAC หรือคำสั่งการควบคุมจาก 'C50 ซึ่งการส่งผ่านพอร์ตอนุกรมจะต้องซิงค์กับ SHIFT CLK
$\overline{EODR}$	3	O	สัญญาณหยุดรับข้อมูล (End of data receive) : ในการติดต่อผ่านพอร์ตอนุกรมในโหมดเวิร์ด (word-mode) สัญญาณ$\overline{EODR}$ จะอยู่ในสถานะต่ำ ทันทีเมื่อทั้ง 16 บิตของเอาท์พุท A/D ได้ถูกส่งจาก AIC ไปยัง 'C50 ซึ่งสามารถใช้สัญญาณนี้ในการอินเทอร์รัพท์ไมโครโปรเซสเซอร์ให้ทราบว่าสิ้นสุดการติดต่อแล้ว หรือ ใช้ สโตรบ และ ให้รีจิสเตอร์เลื่อนข้อมูลออก (enable external serial-to-parallel shift register) ก็ได้ แต่ถ้าเป็นโหมดไบต์ (byte-mode) สัญญาณ $\overline{EODR}$ จะอยู่ในสถานะต่ำ หลังจากไบต์แรกได้ส่งไปยัง 'C50 แล้ว และยังคงรักษาสถานะต่ำจนกระทั่งไบต์ที่สองได้ส่งไป ทั้งนี้ก็เพื่อให้รู้ว่าไบต์แรกหรือไบต์ที่สองออกไป
$\overline{EODX}$	11	O	สัญญาณหยุดส่งข้อมูล (End of data transmit) ก็คล้ายกับ $\overline{EODR}$ ซึ่งจะบอกให้ทราบว่าการติดต่อจาก 'C50 ไปยัง AIC นั้นเสร็จ แล้วทั้งในโหมดเวิร์ด และ โหมดไบต์ ก็คล้ายกับ $\overline{EODR}$
$\overline{FSR}$	4	O	สัญญาณซิงค์การรับ (Frame sync receive):ในการติดต่อทางพอร์ตอนุกรม$\overline{FSR}$จะมีสถานะต่ำตลอดการส่งจากAICไปยัง 'C50(โดยส่งผ่านขาDR)ซึ่งบิตแรกที่จะส่งต้องพร้อมอยู่ที่ขาDR ก่อน$\overline{FSR}$จะโลว์
$\overline{FSX}$	14	O	สัญญาณซิงค์การส่ง (Frame sync transmit): เมื่อสัญญาณนี้อยู่ในสถานะต่ำ พอร์ตอนุกรม 'C50 จะส่งบิตไปยัง AIC โดยมาที่ขา DX ในการติดต่ออนุกรมทุกโหมด $\overline{FSR}$ จะมีสถานะต่ำตลอดการส่ง
IN+	26	I	นอน-อินเวอร์ติง อินพุท (Noninverting input)
IN-	25	I	อินเวอร์ติง อินพุท (Inverting input)
MSTR CLK	6	I	สัญญาณนาฬิกามาสเตอร์ (master clock) จะใช้ในการควบคุมทุกส่วนภายใน AIC ไม่ว่าจะเป็นสัญญาณนาฬิกาเลื่อน (shift clock).

ชื่อ	หมายเลข	I/O	คำอธิบาย
.			สัญญาณนาฬิกาควบคุมฟิลเตอร์ (switched-capacitor filter clock), A/D และ D/A ไทม์มิ่ง
OUT+	22	O	นอน-อินเวอร์ติง เอาท์พุท (Noninverting output)
OUT-	21	O	อินเวอร์ติง เอาท์พุท (Inverting output)
REF	8	I/O	สำหรับ TLC32040 และ TLC32042 แรงดันอ้างอิงภายในจะถูกส่งออกมาที่ขานี้ แต่ถ้าเป็น TLC32040,TLC32041 และ TLC32042 แรงดันอ้างอิงจากภายนอกจะถูกต่อเข้าที่ขานี้
$\overline{RESET}$	2	I	รีเซตจะทำการตั้งค่า TA,TA′,TB,RA,RA′,RB และรีจิสเตอร์ควบ คุม (control register) ให้เป็นค่าดีฟอลต์(default) ซึ่งจะบอกภายหลัง
SHIFT CLK	10	O	สัญญาณนาฬิกาเลื่อนเกิดจากการหารความถี่ของสัญญาณนาฬิกามาสเตอร์ด้วย 4 ซึ่งสัญญาณนี้จะใช้ในการติดต่อทางพอร์ตอนุกรม
VDD	7		ไฟเลี้ยงวงจรดิจิตอล (Digital supply voltage) 5V±5%
Vcc+	20		ไฟเลี้ยงวงจรอนาลอกด้านบวก 5V±5%
Vcc -	19		ไฟเลี้ยงวงจรอนาลอกด้านลบ -5V±5%
WORD/$\overline{BYTE}$	13	O	ขานี้จะทำงานร่วมกับรีจิสเตอร์ควบคุมเพื่อใช้ในการเลือกโหมดการติดต่ออนุกรม ซึ่งมี 4 แบบดังนี้ <u>การติดต่อแบบ อะซิงโครนัส</u> <u>ในโหมดไบต์</u> (WORD/$\overline{BYTE}$ = low) พอร์ตอนุกรมจะติดต่อโดยตรงกับ 'C50 และจะติดต่อทีละ 8 บิต 2 ครั้ง ซึ่งมีขั้นตอนการทำงานดังนี้ 1. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะต่ำ 2. 8 บิตแรกถูกส่งออกไป หรือรับเข้ามา 3. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะต่ำ 4. $\overline{FSX}$ หรือ $\overline{FSR}$high ประมาณ 4 สัญญาณนาฬิกาเลื่อน แล้วอยู่ในสถานะต่ำ 5. 8 บิตต่อมา(ไบต์ที่สอง)ถูกส่งหรือรับเข้ามา 6. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะสูง 7. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะสูง <u>ในโหมดเวิร์ด</u> พอร์ตอนุกรมจะต่อตรงกับพอร์ตอนุกรมของ 'C50 และมีการติดต่อครั้งเดียว 16 บิต ซึ่งมีขั้นตอนการทำงานดังนี้ 1. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะต่ำ

			2. 16 บิตถูกส่งหรือรับเข้ามา
ชื่อ	**หมายเลข**	**I/O**	**คำอธิบาย**
			3. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะสูง 4.$\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะต่ำ
WORD/$\overline{BYTE}$	13	O	<u>การติดต่อแบบซิงโครนัส</u> ในกรณีนี้ แบนด์พาส ฟิลเตอร์ (bandpass filter) และ อัตราการแปลง A/D (A/D conversion timing) จะถูกกำหนดจาก TX เคาน์เตอร์ A, TX เคาน์เตอร์ B และ TA, TA′,และ TB แทน ส่วนในการติดต่อทั้งในแบบโหมดเวิร์ดและโหมดไบต์นั้นมีขั้นตอนเหมือนกับการติดต่อแบบอะซิงโครนัส

4.4 การทำงานของ TLC32040

4.4.1 อินพุททางอนาลอก (analog input)

สำหรับ อนาลอกอินพุทจะมี 2 กลุ่มคือ IN+, IN- และ AUX IN+, AUX IN- ซึ่งสามารถเลือกใช้กลุ่มใดกลุ่มหนึ่งโดยจะใช้ในแบบ ดิฟเฟอเรนเชียล (differential) หรือ ซิงเกิล-เอนด์ (single-ended) และค่าเกน สำหรับอินพุท IN+, IN-, AUX IN+ และ AUX IN- สามารถจะใช้โปรแกรมตั้งค่าได้ (มี3ค่าคือ1,2,หรือ4)การเลือกใช้กลุ่มอินพุทใดจะเลือกโดยใช้ซอฟท์แวร์ควบคุม

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

สำหรับ A/D แบนด์พาส ฟิลเตอร์ (A/D bandpass filter) เราสามารถที่จะเลือกใช้หรือไม่ใช้ก็ได้โดยใช้ ซอฟท์แวร์ควบคุม ความถี่ของสัญญาณนาฬิกาควบคุมฟิลเตอร์ (filter clock) จะเป็นตัวกำหนด ทรานเฟอร์ฟังก์ชัน (transfer function) ของฟิลเตอร์โดยจะคิดอัตราส่วนจากความถี่สัญญาณนาฬิกาควบคุมฟิลเตอร์ 28kHzที่ความถี่ต่ำที่เริ่มมีลักษณะเป็นความถี่สูงผ่านจะมีความถี่เป็น 300 Hz อัตราการแปลง D/A ก็หาได้จากความถี่ ที่หาร 228 kHz ด้วย RX เคาน์เตอร์ B

4.4.3 เอาท์พุททางอนาลอก (analog output)

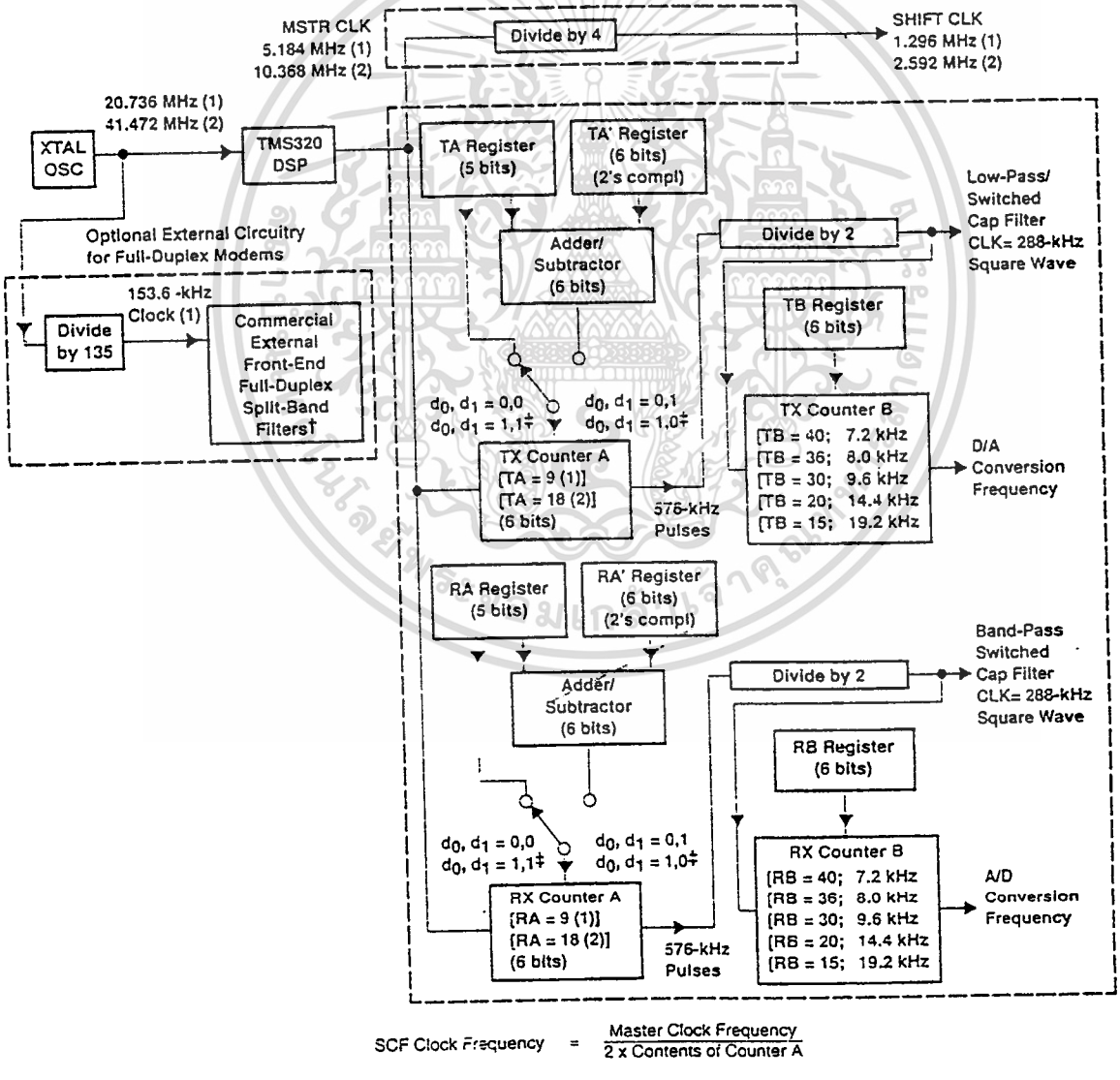
อนาลอกเอาท์พุทจะมีเพาเวอร์แอมปลิไฟล์ (power anplifier) มีเอาท์พุททั้งแบบนอน-อินเวอร์ติง (noninverting) และ แบบอินเวอร์ติง (inverting) เนื่องจากมีแอมปลิไฟล์ทำให้เอาท์พุท สามารถขับ ทรานฟอร์เมอร์ชนิดไฮ-บริด (transformer hybrid) หรือ โหลดอิมพีแดนซ์ต่ำได้ โดยใช้ทั้งแบบดิฟเฟอเรนเชียล หรือ ซิงเกิล-เอนด

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

เช่นเดียวกับ A/D ฟิลเตอร์โดยทรานเฟอร์ฟังก์ชันของฟิลเตอร์ถูกกำหนดจากอัตราส่วนกับความถี่ 228 kHz และอัตราการแปลง D/A ก็หาได้จากความถี่ 228kHz หารด้วย TX เคาน์เตอร์ B

4.4.5 การต่อกลับ (loopback)

การต่อกลับจะให้ผู้ใช้ตรวจสอบวงจรโดย OUT+ และ OUT- จะต่อเข้าภายในกับ IN+ และ IN- ดังนั้นบิต DAC(d15-d2) จะถูกส่งไปยังขา DX และเปรียบเทียบกับบิต ADC ที่รับมาจากขา DR ซึ่งโดยปกติจะต้องมีค่าเท่ากัน (ในทางปฏิบัติอาจไม่เท่ากันก็ได้) ในการตรวจสอบ ถ้าใช้ขา IN+ และ IN- สัญญาณภายนอกที่ต่อกับ IN+ , IN- จะไม่มีผลแต่ถ้าใช้ AUX IN+, AUX IN- สัญญาณภายนอกจะถูกรวมกับ OUT+ และ OUT- สำหรับการควบคุมการต่อกลับ จะทำโดยการตั้งค่าในรีจิสเตอร์ควบคุม



รูปที่ 4.2 แสดงไทม์มิ่งภายใน

รูปแบบของบิตใน AIC DR หรือ DX เวิร์ด

A/D or D/A MSB

1st bit sent ↓ 1st bit sent of 2nd byte ↓ A/D or D/A LSB ↓

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

รูปที่ 4.3 แสดงรูปแบบบิตใน AIC DR หรือ DX word

รูปแบบของบิตใน AIC DX (AIC DX data word format section)

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

รูปที่ 4.4 แสดงรูปแบบการ formatใน AIC DX data word

ในรูปแบบการส่งครั้งแรก (Primary DX serial communication protocal (d0-d15)) นั้น เราจะใช้ d1 และ d0 เป็นตัวเลือก ส่วน d2-d15 จะถูกส่งไปยังรีจิสเตอร์ควบคุมการแปลง D/A (D/A converter register) ซึ่งจะแบ่งการทำงานออกเป็นกรณีต่าง ๆ ได้ดังนี้คือ

ตารางที่ 4.2 แสดงการกำหนดค่าใน d1 และ d0 ในการทำงานในโหมดต่างๆของ AIC

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

รูปแบบของบิตในการส่งครั้งที่สอง (Secondary DX serial communication protocol)

รูปแบบของบิตในการส่งครั้งที่สอง (Secondary DX serial communication protocol)

ตารางที่ 4.3 แสดง Secondary DX serial communication protocol

XX ←to TA register→ X X ←to RA register→ 0 0	d13 และ d6 เป็น MSR (เป็นบวก)
X ←to TA′register→ X ← to RA′register → 0 1	d14 และ d7 เป็นบิตแบบ 2'S
X ←to TB register→ X ← to RB register → 1 0	d14 และ d7 เป็น MSB (เป็นบวก)
X X X X X X X X d7 d6 d5 d4 d3 d2 1 1 Control ← Register →	d2 = 0/1 ลด/เพิ่ม bandpass filter d3 = 0/1 disable/enable ฟังก์ชัน loopback d4 = 0/1 disable/enable ขา AUX IN+ และ ขา AUX IN- d5 = 0/1 บิตกำหนดการรับส่งแบบ asynchronous / synchronous d6, d7 เป็นบิตควบคุม Gain

ฟังก์ชัน Reset

ฟังก์ชันรีเซตจะเป็นการตั้งค่าอัตราการเปลี่ยนแปลง A/D และ D/A ให้เป็น 8 kHz ถ้าใช้สัญญาณนาฬิกามาสเตอร์ 5.184 Mhz โดยค่าในรีจิสเตอร์จะถูกตั้งค่าเริ่มต้นเป็น

TA = RA = 9 , TA′= RA′= 1 , TB = RB = 24 และบิตในรีจิสเตอร์ควบคุมเป็น d7=1, d6=1, d5=1, d4=0, d3=0, d2=1

ข้อบังคับในการกำหนดค่าในรีจิสเตอร์ของ AIC

1. รีจิสเตอร์ TA ในโหมดเวิร์ด ต้องมีค่า ≥ 4
2. รีจิสเตอร์ TA ในโหมดไบต์ ต้องมีค่า ≥ 5
3. รีจิสเตอร์ TA′สามารถเป็นได้ทั้ง ค่าบวก, ลบ หรือ ศูนย์
4. รีจิสเตอร์ RA ในโหมดเวิร์ด ต้องมีค่า ≥ 4
5. รีจิสเตอร์ RA ในโหมดไบต์ ต้องมีค่า ≥ 5
6. รีจิสเตอร์ RA′สามารถมีค่าเป็นได้ทั้ง ค่าบวก, ลบ หรือ ศูนย์
7. (TA±TA′) จะต้องมีค่า > 1
8. (RA±RA′) จะต้องมีค่า > 1
9. TB จะต้องมีค่ามากกว่า 1

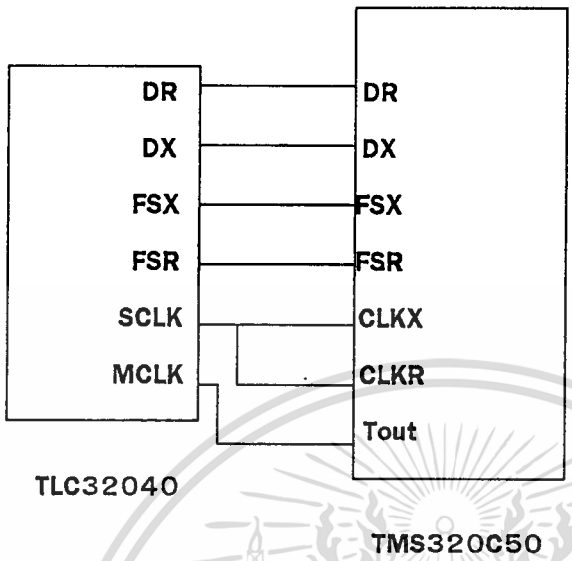
ถ้าไม่เป็นไปตามเงื่อนไขทั้ง 9 ข้อที่กล่าวมา AIC จะทำงานตามเงื่อนไขดังตารางที่ 4.4

ตารางที่ 4.4

เงื่อนไข	การตอบสนองของ AIC
1. ถ้า TA+TA′= 0 หรือ 1 หรือ TA-TA′= 0 หรือ 1	จะทำการโหลดค่าในรีจิสเตอร์ TA ลงใน TX เคาน์เตอร์ A แทน
2. ถ้า TA+TA′ < 0	MODULO 64 จะถูกใช้เพื่อให้แน่ใจว่า ค่าที่เป็นบวกถูกโหลดไปไว้ที่ TX เคาน์เตอร์ A แทน เช่น TA+TA′+ 40 HEX โหลดไปไว้ที่ TX เคาน์เตอร์ A
3. ถ้า RA+RA′ = 0 หรือ 1 RA-RA′ = 0 หรือ 1	จะทำการโหลดค่าในรีจิสเตอร์ RA ลงใน RX เคาน์เตอร์ A
4. ถ้า RA+RA′ = 0 หรือ 1	MODULO 64 ถูกใช้เหมือนข้อ 2 เพียงแต่โหลดไปที่ RX เคาน์เตอร์ A
5. ถ้า TA = 0 หรือ 1 หรือ RA = 0 หรือ 1	AIC หยุดทำงาน
6. ถ้า TA < 4 ในโหมดเวิร์ด TA < 5 ในโหมดไบต์ RA < 4 ในโหมดเวิร์ด RA < 5 ในโหมดไบต์	การติดต่อทางพอร์ตอนุกรมของ AIC ไม่ทำงาน
7. ถ้า TB = 0 หรือ 1	โหลดค่า TB ใหม่ ด้วยค่า 24 HEX
8. ถ้า RB = 0 หรือ 1	โหลดค่า RB ใหม่ด้วยค่า 24 HEX
9. ถ้า AIC และ DSP ไม่สามารถติดต่อกันได้	จะเก็บค่า DAC Output ไว้

4.5 การอินเตอร์เฟสระหว่าง TMS320C50 กับ TLC32040

การใช้อินเทอร์รัพต์ของพอร์ตอนุกรม



รูปที่ 4.5 แสดงการเชื่อมต่อระหว่าง TLC32040 กับ TMS320C50

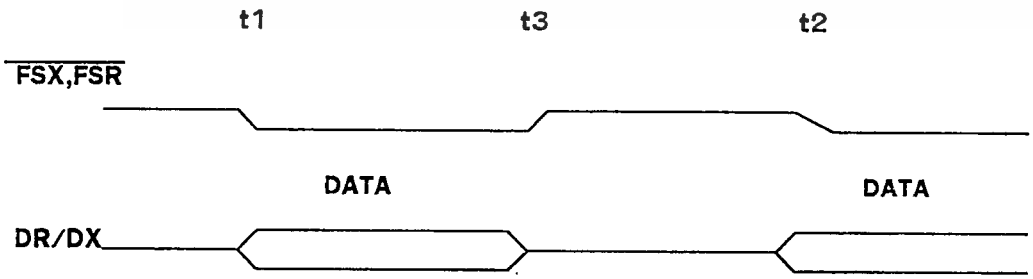
TMS320C50

การส่ง โดยการเขียน DXR จะเป็นการขออินเทอร์รัพต์ของ XINT (โดยไม่มีการมาร์ค)

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

TLC32040

เนื่องจากการส่งระหว่าง TLC กับ TMS เป็นแบบซิงโครนัส TLC32040 จะส่งและรับทุก ๆ เฟรมซิงโครนัส (Frame Synchonous)



รูปที่ 4.6 แสดงช่วงเวลาในการส่งและรับข้อมูลระหว่าง TLC32040 กับ TMS320C50

เนื่องจากความถี่แซมปลิ้งสูงสุด= 19.2 Khz เพราะฉะนั้นข้อมูลต้องมาก่อน (ภายใน t2 - t1) t2-t1 ต้องไม่น้อยกว่า 52.08ไมโครวินาที ข้อมูลที่มาจาก TLC32040 แล้วทำการประมวลผลทางคณิตศาสตร์แล้วส่งกลับมาที่ TLC32040 อีกครั้ง ต้องใช้เวลาน้อยกว่า 1/fs จึงจะสามารถได้สัญญาณที่มีความถี่เข้าเท่ากับความถี่ออก เนื่องจาก t3-t1 ช่องของข้อมูล(16 bit)ใช้เวลาประมาณ (1/sclk)x16 ประมาณเท่ากับ 6.17 ไมโครวินาที เพราะฉะนั้น 6.17 ไมโครวินาที เป็นเวลาที่ใช้ในการส่ง 1 ข้อมูล จะต้องใช้เวลาเหลือประมวลผล(52.08-6.17) = 45.91 ไมโครวินาที

หรือประมาณ (45.91ไมโครวินาที)/(50นาโนวินาทีต่อ 1 คำสั่ง) = 918 คำสั่ง

<u>สรุป</u> สามารถเขียนโปรแกรมถึงกว่า 900 คำสั่ง เพื่อประมวลผลข้อมูล 1 ข้อมูลก่อนที่จะเกิดการอินเทอร์รัพต์ อีกครั้ง แต่เราใช้ได้ถึงเพราะ fs มักจะน้อยกว่า 19.2 Khz

บทที่5

วงจรกรองสัญญาณดิจิตอลประเภท เอ็ฟไออาร์

(Finite Impulse Response(FIR) Filter Design Techniques)

วงจรกรองสัญญาณดิจิตอลประเภท FIR มีคุณสมบัติที่สำคัญ 2 ประการ คือเราสามารถจะออกแบบให้มีการตอบสนองความถี่ของเฟสมีความเป็นเชิงเส้นได้โดยแท้จริง และ วงจรกรองสัญญาณดิจิตอลประเภท FIR จะเสถียรเสมอ เนื่องจากโครงสร้างปราศจากส่วนป้อนกลับ

การออกแบบวงจรกรองสัญญาณดิจิตอลประเภท FIR มีหลายวิธี แต่ในที่นี้จะพิจารณาเพียง 2 วิธี คือวิธีวิธีหน้าต่าง(window method) และวิธี DFT (Discrete Fourier Transform)ผสมหน้าต่าง

5.1 การออกแบบวงจรกรองสัญญาณดิจิตอลประเภท FIR ด้วยวิธีหน้าต่าง

ฟังก์ชันระบบของวงจรกรองสัญญาณดิจิตอลประเภท FIR เขียนเป็นคณิตศาสตร์ได้ว่า

$$H_{FIR}(z) = \sum_{n=-N_P}^{N_P} b_k z^{-k} \qquad (5.1)$$

หากเราทราบค่าสัมประสิทธิ์ b_k , $-N_F \leq k \leq N_P$ เราก็สามารถนำไปสร้างเป็นวงจรโดยใช้โครงสร้างแบบไม่ป้อนกลับได้ จากคุณสมบัติของวงจรกรองสัญญาณดิจิตอลประเภท FIR เราทราบว่าสัมประสิทธิ์ดังกล่าวและะค่าตอบสนองค่าหนึ่งนั้นมีค่าเท่ากัน หรือ

$$h(n) = b_n \quad ;-N_F \leq n \leq N_P \qquad (5.2)$$

ซึ่งก็คือเราสามารถสร้างวงจรกรองสัญญาณดิจิตอล FIR ได้โดยตรงจากค่าตอบสนองค่าหนึ่ง

โดยปกติแล้วเราจะออกแบบวงจรกรองสัญญาณดิจิตอลประเภท FIR ให้มีการตอบสนองความถี่ของเฟสมีความเป็นเชิงเส้น ซึ่งกระทำได้โดยมีเงื่อนไขว่าค่าตอบสนองค่าหนึ่งที่ต้องการ $\{h_D(n)\}$ จะต้องมีลักษณะสมมาตร กล่าวคือ$\{h_D(n)\}$ เป็นจำนวนที่สิ้นสุดและสมมาตรในช่วง $-N_F \leq n \leq N_P$ โดยที่ $N_F = N_P$ (อันเนื่องมาจากความสมมาตร) หาก $\{h_D(n)\}$ มีลักษณะดังกล่าวมานี้ เราจะสามารถสร้างวงจรกรองสัญญาณดิจิตอลประเภท FIR ที่มีค่าตอบสนองค่าหนึ่ง ตรงตาม $\{h_D(n)\}$ ที่กำหนดมาให้ได้อย่างแม่นยำ โดยทำให้สัมประสิทธิ์ b_n เท่ากับ $h_D(n)$ ดังสมการ (5.3)

$$b_n = h_D(n) \quad ;-N_F \leq n \leq N_P \qquad (5.3)$$

แต่ในทางปฏิบัติเรามักจะพบปัญหาคือ$\{h_D(n)\}$ มีจำนวนอนันต์

เราสามารถสรุปการออกแบบด้วยวิธีหน้าต่างได้ว่าเมื่อมีผลตอบสนองค่าหนึ่งจำนวนอนันต์ที่ต้องการ$\{h_D(n)\}$เราต้องเลือกค่า$\{w(n)\}$ ที่ทำให้ค่าตอบสนองดังกล่าวมีค่าตอบสนองของแอมปลิจูดตรงตามเงื่อนไขที่กำหนด เนื่องจากโครงสร้างของวงจร FIRจะใช้จำนวนหน่วยตัวคูณมากขึ้น เมื่อ N มีค่ามากขึ้น ดังนั้นเราจึงต้องกำหนด $\{w(n)\}$ ที่มีจำนวนน้อยเพื่อลดจำนวนหน่วยตัวคูณลงซึ่งจะประหยัดค่าใช้จ่ายในการสร้างวงจร

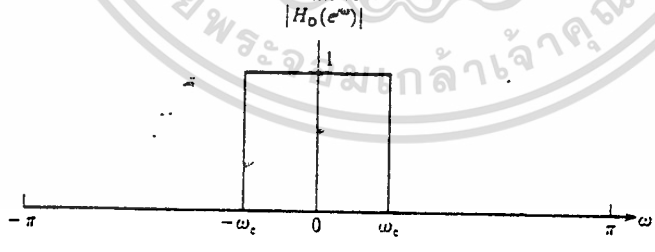
การคูณสัญญาณที่ไม่ต่อเนื่องสองสัญญาณในแกนเวลาจะหมายถึงการคอนโวลูชัน(convolution) ของสเป็กตรัม หรือฟูเรียร์ทรานฟอร์มของสัญญาณทั้งสองในแกนความถี่ กล่าวคือถ้าให้ $H_D(e^{j\omega})$ เป็นสเปคตรัมของ $\{h_D(n)\}$ และ $W(e^{j\omega})$ เป็นสเปคตรัมของ $\{w(n)\}$ แล้วจะได้ทรานส์เฟอร์ฟังชั่น $H_N(e^{j\omega})$ ของวงจรกรองสัญญาณประเภท FIR คือ

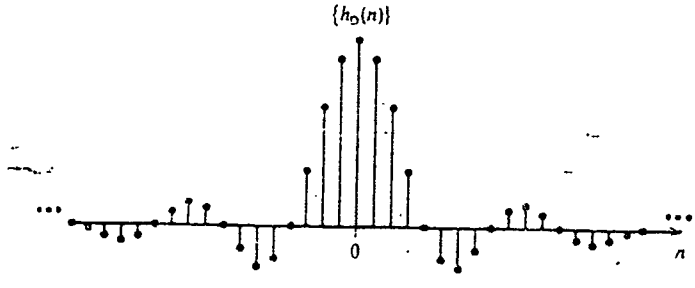
$$H_N(e^{j\omega}) = H_D(e^{j\omega}) * W(e^{j\omega})$$
$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_D(e^{j\theta})\, W(e^{j[\omega-\theta]})d\theta \qquad (5.4)$$

เนื่องจากฟังก์ชั่นทั้งสองซึ่งอยู่ภายในเครื่องหมายอินทิกรัลเป็นฟังก์ชั่นคาบ การคอนโวลูชันจึงเป็นแบบเซอร์คูลาร์คอนโวลูชัน ดังนั้นทำการอินทิเกรตเพียงคาบเดียวก็เพียงพอ การหาผลลัพธ์ของสมการ5-4 แบบสูตรสำเร็จนั้นหาได้ยากเนื่องจากโดยทั่วไปแล้วสเปคตรัมจะเป็นฟังก์ชั่นเชิงซ้อนและมีรูปแบบที่ซับซ้อนของไซน์และโคไซน์อย่างไรก็ตามถ้าสัญญาณไม่ต่อเนื่องในแกนเวลามีลักษณะสมมาตรรอบจุด n=0 แล้วสเปคตรัมจะเป็นฟังก์ชั่นของค่าจริงในโดเมนความถี่ จากคุณสมบัติดังกล่าวนี้แสดงให้เห็นผลกระทบอันเนื่องจากการ คอนโวลูชัน ซึ่งอธิบายได้ดังนี้

เพื่อแสดงผลกระทบอันเนื่องจากการคอนโวลูชัน พิจารณาการออกแบบด้วยวิธีหน้าต่างเพื่อให้ได้วงจรกรองสัญญาณที่มีค่าตอบสนองความถี่ของขนาด ใกล้เคียงค่าตอบสนองของ ตัวกรองความถี่ต่ำผ่าน ในอุดมคติมากที่สุด ซึ่ง โลว์พาส(lowpass) อุดมคติที่มีความถี่คัทออฟที่ ω_C สามารถ เขียนเป็นทรานส์เฟอร์ฟังก์ชั่นที่ต้องการได้ว่า

$$H_D(e^{j\omega}) = \begin{cases} 1 & ;-\omega_c \le \omega \le \omega_c \\ 0 & ;-\pi < \omega < -\omega_c \text{ and } \omega_c < \omega < \pi \end{cases} \qquad (5.5)$$





รูปที่ 5.1 การตอบสนองความถี่ของโลว์พาสอุดมคติและค่าตอบสนองค่าหนึ่งจำนวนอนันต์ที่ต้องการ $\{h_D(n)\}$ ของโลว์พาสอุดมคตินี้ รูปนี้แสดงที่ $\omega c = \pi/4$

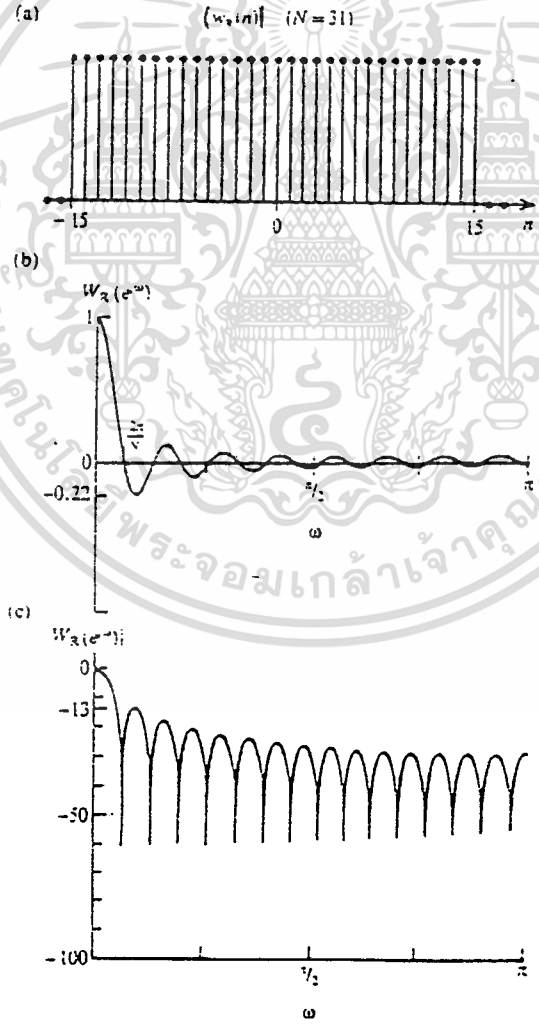
ดังแสดงในรูปที่4.1 กรณี $\Omega c=\pi/4$ ค่าตอบสนองค่าหนึ่งของโลว์พาสอุดมคติดังกล่าวก็คืออินเวิสฟูเรียร์ทรานส์ฟอร์ม(Inverse Fourier Transform)ของสมการ (5.5)ซึ่งหาได้ดังนี้

$$h_D(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_D(e^{j\omega})e^{j\omega n}d\omega$$

$$= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}d\omega = \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{j2\pi n} \qquad (5.6)$$

$$= \frac{\sin(\omega_c n)}{\pi n} \qquad ;\textit{for all } n$$

จะเห็นว่ามีลักษณะสมมาตรโดยจุดสมมาตรอยู่ที่ n=0 และมีจำนวนที่ไม่สิ้นสุด กล่าวคือมีค่าตามแกนเวลาทั้งบวกและลบไปจนถึงอนันต์ดังแสดงในรูปที่4.1ต่อไปจะพิจารณาการหา{$h_D(n)$} โดยใช้หน้าต่างแบบต่างๆอันได้แก่แบบสี่เหลี่ยม(rectangular)แบบสามเหลี่ยม(triangular) และแบบโคไซน์



รูปที่ 5.2 หน้าต่างสี่เหลี่ยม (ซึ่งมี N =31)

a) ค่าของหน้าต่างในแกนเวลา b) สเปคตรัม c) ขนาดของสเปคตรัมเป็น dB

หน้าต่างสี่เหลี่ยม ค่าของหน้าต่างสี่เหลี่ยมจะมีนิยามว่า

$$w_R(n) = \begin{cases} 1 & ; \frac{-(N-1)}{2} \le n \le \frac{(N-1)}{2} \\ 0 & ; Otherwise \end{cases} \qquad (5.7)$$

ซึ่งแสดงให้เห็นเป็นตัวอย่าง ในรูปที่ 5.2 กรณี N=31 เราจะตัดทอนค่าตอบสนองค่าหนึ่ง จำนวนอนันต์{$h_D(n)$} ใน(5.6) เพื่อให้ได้มาซึ่งค่าตอบสนองค่าหนึ่งจำนวนสิ้นสุด {$(h_N(n)$} โดยการคูณแต่ละค่าของ{$h_D(n)$} ใน (5.6) กับแต่ละค่าของหน้าต่างสี่เหลี่ยม {$w_R(n)$} ใน (5.7) ดังนี้

$$h_N(n) = w_R(n)h_D(n) \qquad for\ every\ value\ of\ n$$
$$= \frac{\sin(\omega_c n)}{\pi n} \qquad for\ \frac{-(N-1)}{2} \le n \le \frac{(N-1)}{2} \qquad (5.8)$$

ค่าจริงของทรานส์เฟอร์ฟังก์ชั่น $H_N(e^{j\omega})$ ของวงจรกรองสัญญาณดิจิตอลแบบ FIR ซึ่งหาได้อยู่ในรูปที่ 5.3 ลักษณะของ $H_N(e^{j\omega})$ ที่ได้ต่างกับในอุดมคตินั้นเป็นผลจากการ convolution ระหว่าง $W_R(e^{j\omega})$ และ $H_D(e^{j\omega})$ เราจะเริ่มจากการพิจารณาลักษณะของ $W_R(e^{j\omega})$ ซึ่งเป็นตัวที่ทำให้ เกิดการเบี่ยงเบนไปจากค่าตอบสนองอุดมคติ สเปคตรัมของหน้าต่างเขียนได้ว่า

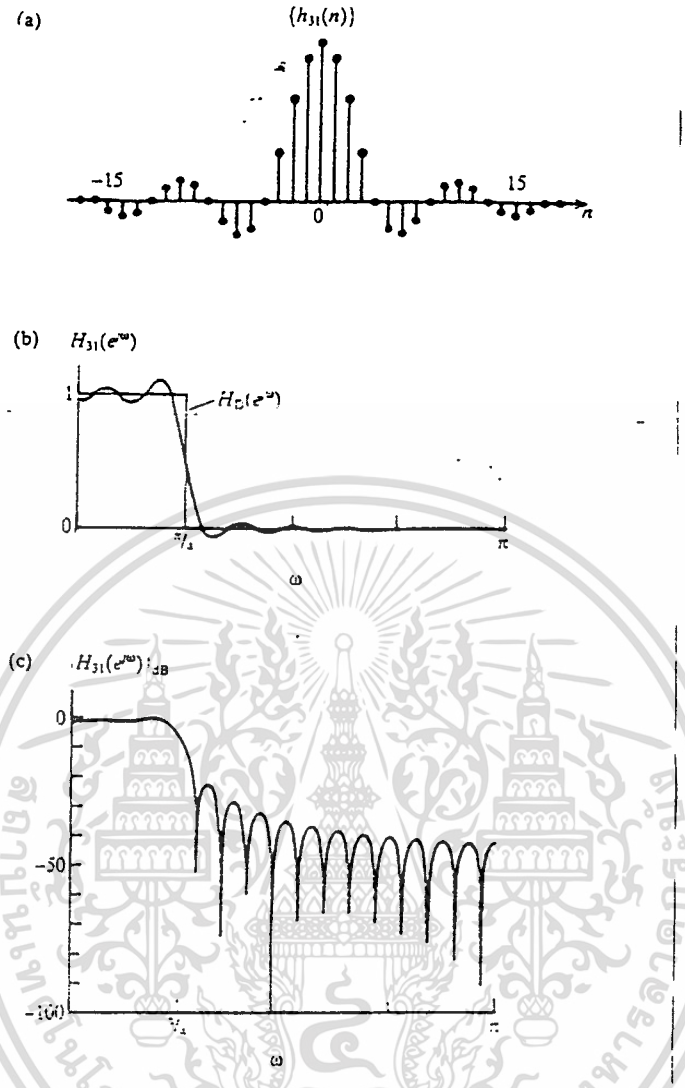
$$W_R(e^{j\omega}) = \sum_{n=-(N-1)/2}^{(N-1)/2} e^{-j\omega n} = e^{j\omega(N-1)/2} \sum_{n=0}^{N-1} e^{-j\omega n} \qquad (5.9)$$

ซึ่งเมื่อใช้สูตรผลบวกของอนุกรมเรขาคณิตที่สิ้นสุดจะได้ว่า

$$W_R(e^{j\omega}) = e^{j\omega(N-1)/2} \frac{1-e^{-j\omega N}}{1-e^{-j\omega}} \qquad (5.10)$$

$$W_R(e^{j\omega}) = e^{j\omega(N-1)/2} \frac{e^{-j\omega N/2} e^{j\omega N/2} - e^{-j\omega N/2}}{e^{-j\omega/2} e^{j\omega/2} - e^{-j\omega/2}}$$
$$= \frac{\sin(N\omega/2)}{\sin(\omega/2)} \qquad (5.11)$$

สเปคตรัม $W_R(e^{j\omega})$ กรณี N=31 แสดงอยู่ในรูปที่ 5.2 สเปคตรัม $W_R(e^{j\omega})$ มีลักษณะที่สำคัญสองประการสำหรับการใช้งานคือ ความกว้างของโหลบหลัก (main lobe) และความสูงต่ำ (amplitude) ของโหลบข้างเคียง (side lobe) ความกว้างของโหลบหลัก หมายถึงระยะระหว่างจุดสองจุดที่อยู่ใกล้ $\omega = 0$ มากที่สุด ที่ค่า $W_R(e^{j\omega})$ เป็นศูนย์ สำหรับหน้าต่างสี่เหลี่ยมความกว้างของ main lobe เท่ากับ $4\pi/N$ และความสูงสุดของโหลบข้างเคียงจะอยู่ที่ลอนข้างเคียงแรกซึ่งมีค่าประมาณ 22% ของความสูงของโหลบหลัก หรือ -13 dB เมื่อเรากำหนดให้ความสูงของโหลบหลักที่ $\omega=0$ อยู่ที่ 0 dB



รูปที่ 5.3 ผลจากการประยุกต์หน้าต่างสี่เหลี่ยม (ซึ่งมี N=31)
กับค่าตอบสนองค่าหนึ่งของโลว์พาสอุดมคติ

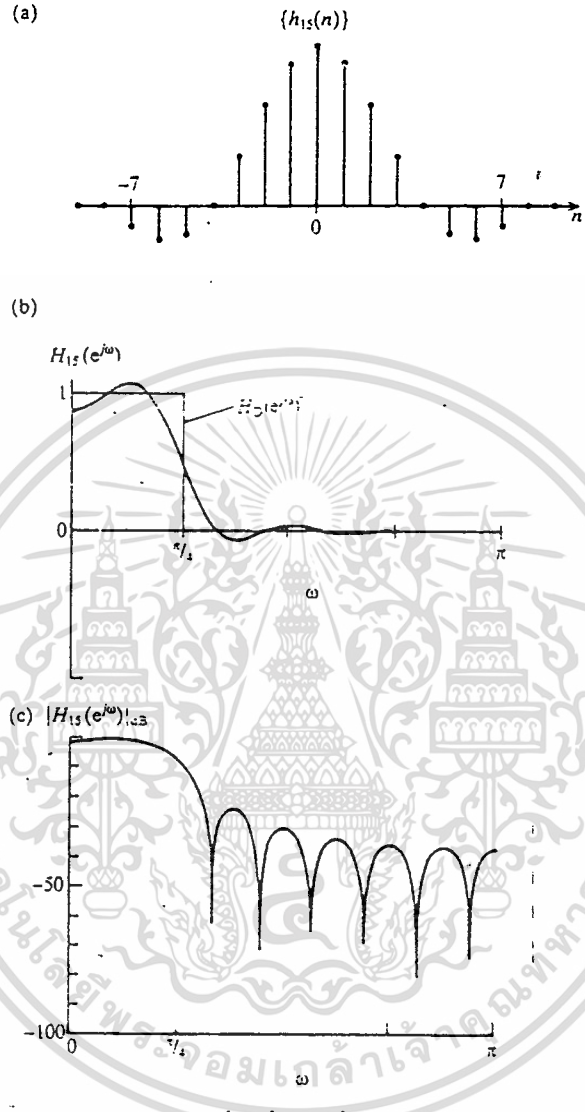
a) ค่าตอบค่าหนึ่งซึ่งมีจำนวนสิ้นสุดกล่าวคือจำนวน 31

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

c) การตอบสนองความถี่ขนาดเป็น dB

ต่อไปเราจะพิจารณาถึงผลกระทบของลักษณะดังกล่าวที่มีต่อ $H_N(n)$ ที่ได้จากการคอนโวลูชั่น รูปที่5-4 แสดงทรานส์เฟอร์ฟังก์ชั่น $H_N(e^{j\omega})$ และการตอบสนองของขนาด $|H_N(e^{j\omega})|$ dB กรณีหน้าต่างสี่เหลี่ยมที่มี N= 31 ซึ่งจะเห็นว่ามีความแตกต่างจาก $H_D(e^{j\omega})$ และ$|H_D(e^{j\omega})|$ dB ที่เราต้องการ กล่าวคือมีการเปลี่ยนแปลงที่ $\omega = \omega_C$ ซึ่งในทางอุดมคติจะเปลี่ยนแปลงอย่างฉับพลันจากค่า 1ไปสู่ค่า 0แต่วงจรกรองสัญญาณที่ได้จะมีการเปลี่ยนแปลงแบบค่อยลาดลงจากค่า 1ไปสู่ค่า 0 ในย่านผ่านความถี่วงจรกรองสัญญาณที่ได้จะมีค่า โอเวอร์ชูต(over shoots) และค่า อันเดอร์ชูต(under

shoots) ในย่านหยุดความถี่นั้นทางอุดมคติเราต้องการให้เป็น 0 แต่วงจรกรองสัญญาณจะมีค่าสูงเกินและต่ำกว่าค่า 0 บางครั้งเรียกว่าลีกเกจ leakage



รูปที่ 5.4 ผลการประยุกต์หน้าต่างสี่เหลี่ยม (ซึ่งมี N=15) ของโลว์พาสอุดมคติ

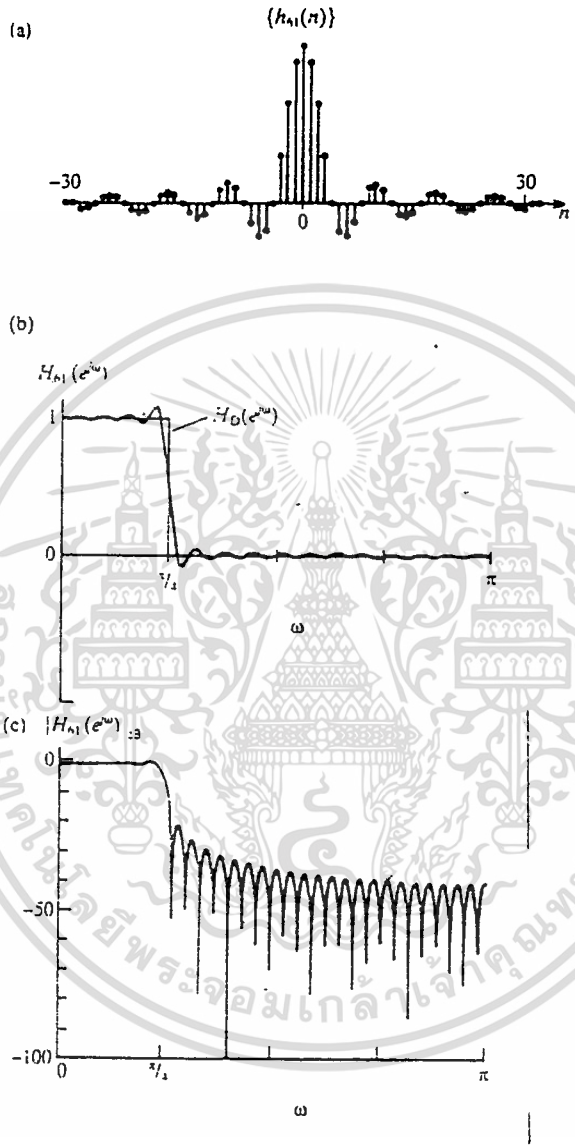
a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนที่สิ้นสุดกล่าวคือจำนวน 15

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

c) การตอบสนองความถี่ขนาดเป็น dB

โหลบหลักของ$W_R(e^{j\omega})$ทำให้เกิดความลาดชันในทรานส์เฟอร์ฟังก์ชั่นที่ได้ซึ่งความลาดชันนี้จะสัมพันธ์กับความกว้างของ โหลบหลัก ของ $W_R(e^{j\omega})$ เนื่องจากเราทราบว่าความกว้างของโหลบหลักของ $W_R(e^{j\omega})$ มีค่าเท่ากับ $4\pi/N$ เพราะฉะนั้นเราสามารถจะลดความลาดลงได้ตามแต่จะต้องการโดยการเพิ่มความกว้างของหน้าต่างหรือนั่นคือเพิ่มค่า N ให้มากขึ้น รูปที่ 5.4 และ 5.5 แสดงลักษณะของ $H_N(e^{j\omega})$ ที่ค่า N ซึ่งต่างกัน อย่างไรก็ดีการเพิ่มจำนวน N แม้ว่าจะมีประโยชน์

คือช่วยลดความลาดลงแต่มีข้อเสียคือเมื่อสร้างวงจรด้วยจำนวน N ที่สูง เราต้องใช้หน่วยในการคูณมากขึ้น ทำให้ค่าใช้จ่ายสูงขึ้น



รูปที่5.5 ผลจากการประยุกต์หน้าต่างสี่เหลี่ยม (ซึ่งมี N =61) ของโลว์พาสอุดมคติ

a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนที่สิ้นสุดกล่าวคือ จำนวน 61

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

c) การตอบสนองความถี่ของขนาดเป็น dB

โหลบข้างเคียง(side lobe)ของ$W_R(e^{j\omega})$ทำให้วงจรกรองสัญญาณที่ได้มีค่าตอบสนองความถี่ของขนาดมีโอเวอร์ชูตและอันเดอร์ชูตหรือ มีริปเปิล (ripple) ทั้งในย่านผ่านความถี่และย่านหยุดความถี่ ในช่วง จาก 0 ถึง π ปรากฏการณ์นี้เราเรียกว่า ปรากฏการณ์กิบส์ (Gibbs phenomenon)

ซึ่งเป็นข้อเสียของ หน้าต่างสี่เหลี่ยม แม้ว่าจะเพิ่ม ความกว้างของหน้าต่างหรือเพิ่มค่าของ N ดังแสดงใน รูปที่ 5.3 ,5.4 และ 5.5 ก็ยังไม่สามารถกำจัดริปเปิล ดังกล่าวได้

เพื่อลดผลของ โหลบข้างเคียง เราต้องพิจารณาใช้หน้าต่างแบบอื่นที่มี ริปเปิ้ล ของโหลบข้างเคียง น้อยกว่า หน้าต่างสี่เหลี่ยม จาก โหลบข้างเคียง ของสเปคตรัมของ $W_R(e^{j\omega})$ ใดๆ จะเป็นตัวที่ชี้ว่าหน้าต่างนั้นๆ มีส่วนประกอบความถี่สูงมากน้อยเพียงใด สำหรับหน้าต่างสี่เหลี่ยม ส่วนประกอบ ความถี่สูงนี้จะเนื่องมาจากการเปลี่ยนแปลงฉับพลันจาก 0ไป 1 มีลักษณะลาดแทนที่จะมีลักษณะฉับพลัน ด้วยเหตุผลดังกล่าวมานี้เราจึงจะลองพิจารณาหน้าต่างสามเหลี่ยม ซึ่งบางทีก็เรียกว่า หน้าต่างบาร์เลทท์ (Barlett) ดังต่อไปนี้

หน้าต่างสามเหลี่ยมหรือหน้าต่างบาร์เลทท์

หากเราทำหน้าต่างสี่เหลี่ยมให้เรียวลงโดยให้ค่าสูงสุดอยู่ตรงกลางจากนั้นให้ลดลงแบบเชิงเส้นไปสู่ค่าศูนย์ที่ขอบหน้าต่าง เราก็จะได้หน้าต่างสามเหลี่ยมจำนวน N ค่า ซึ่งเขียนได้ว่า

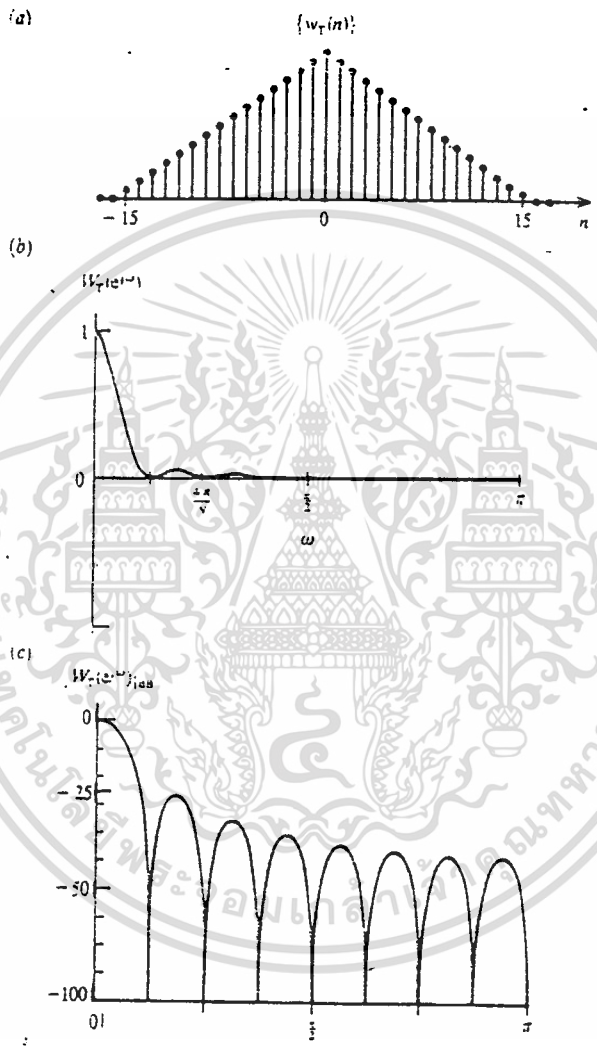
$$w_T(n) = 1 - \frac{2|n|}{N-1} \qquad ;-(N-1)/2 \le n \le (N-1)/2 \tag{5.12}$$

หน้าต่างสามเหลี่ยมและสเปคตรัม ของ $W_T(e^{j\omega})$ แสดงในรูปที่ 4.6 ตามที่คาดคะเนไว้ จะเห็นว่าขนาดของโหลบข้างเคียง เล็กลงกว่าของกรณีหน้าต่างสี่เหลี่ยม กล่าวคือลดจาก -13 dB เป็น -25 dB แต่ความกว้างของ โหลบหลัก จะมีค่าเป็น $8\pi/N$ หรือนั่นคือความกว้างเป็นสองเท่าของหน้าต่างสี่เหลี่ยมที่จำนวน N เท่ากัน

รูปที่ 5.7 แสดงค่าตอบสนองความถี่ของวงจรกรองสัญญาณประเภท FIR ซึ่งได้รับจากการประยุกต์หน้าต่างสี่เหลี่ยมเข้ากับค่าตอบสนองค่าหนึ่งของโลว์พาสอุดมคติ เมื่อเปรียบเทียบกับกรณีที่ใช้หน้าต่างสี่เหลี่ยม ก็จะพบว่าหน้าต่างสามเหลี่ยมจะให้การตอบสนองความถี่ที่เรียบกว่า การเปลี่ยนแปลงจากย่านผ่านความถี่ไปยังย่านหยุดความถี่จะไม่ชันเหมือนกรณีที่ใช้หน้าต่างสี่เหลี่ยม ในย่านหยุดความถี่การตอบสนองความถี่ก็เรียบกว่าเช่นกันแต่ความสามารถในการลดทอน (attenuation) สัญญาณจะต่ำกว่า ด้วยลักษณะดังกล่าวมานี้ จึงทำให้หน้าต่างสามเหลี่ยมมิใช่ตัวเลือกที่ดีเสมอไป

ต่อไปเราจะพิจารณาหน้าต่างจำพวกหนึ่งที่เรียกว่า หน้าต่างเรสโคไซน์(raised-cosine window) ซึ่งเมื่อเทียบกับหน้าต่างสามเหลี่ยมแล้วจะมีลักษณะเรียบที่บริเวณขอบหน้าต่างแต่จะเข้าใกล้ 1 มากกว่าในบริเวณกลางหน้าต่าง ความเรียบที่บริเวณขอบหน้าต่างจะช่วยลดระดับของโหลบข้างเคียง และส่วนกลางของหน้าต่างที่กว้างขึ้นจะช่วยลดอาการที่เพี้ยนไปจาก $\{h_N(n)\}$ บริเวณ

n=0

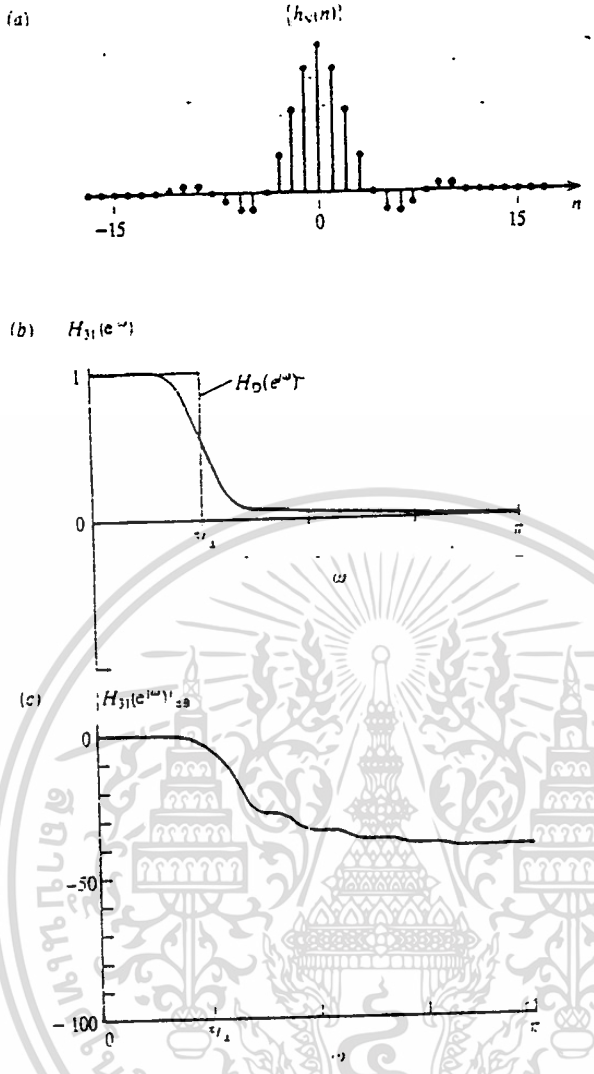


รูปที่ 5.6 **หน้าต่างสี่เหลี่ยม (ซึ่งมี** N =31)

a) ค่าของหน้าต่างในแกนเวลา

b) สเปคตรัม

c) ขนาดของสเปคตรัมเป็น dB



รูปที่ 5.7 ผลจากการประยุกต์หน้าต่างสี่เหลี่ยม (ซึ่งมี N= 31) กับค่าตอบสนองค่าหนึ่งของโลว์พาสอุดมคติ

a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนสิ้นสุดกล่าวคือ N=31

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

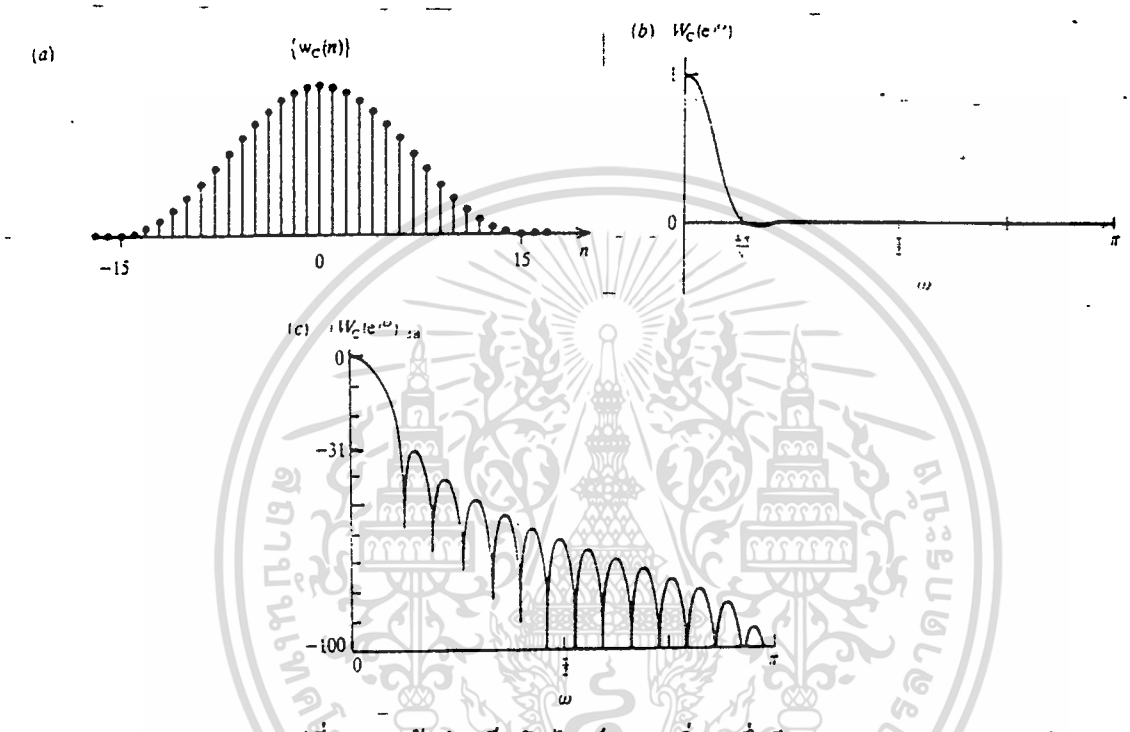
c) การตอบสนองความถี่ของขนาดเป็น dB

หน้าต่างเรสโคไซน์ เพื่อจะลดระดับของ โหลบข้างเคียงให้ต่ำลงไปอีก เราจะพิจารณาหน้าต่างที่บริเวณขอบมีลักษณะลาดมากขึ้น โดยการใช้ฟังก์ชั่นที่เรียกว่า เร็สโคไซน์ ซึ่งแบ่งออกเป็น 3 ประเภท คือหน้าต่างแฮนนิ่ง (Hanning Window) หรือบางทีก็เรียกตรงๆว่า หน้าต่างเร็สโคไซน์ หน้าต่างแฮมมิ่ง(Hamming Window) และหน้าต่างแบล็คแมน (Blakman Window)

เราจะเริ่มพิจารณาหน้าต่างแฮนนิ่ง ซึ่งมีสัญญลักษณ์ $\{w_C(n)\}$ และนิยามว่า

$$w_R(n) = \begin{cases} 0.5 + 0.5\cos(2\pi n/(N-1)) & ;-(N-1)/2 \le n \le (N-1)/2 \\ 0 & ;another\ N \end{cases} \qquad (5.13)$$

รูปที่ 5.8 แสดงค่าของหน้าต่างนี้ และสเปคตรัม $w_C(e^{j\omega})$ ของมัน เมื่อ N = 31 ซึ่งโหลบข้างเคียงแรก (ซึ่งมีระดับสูงกว่าลอนข้างเคียงอื่นๆ) จะอยู่ที่ -31 dB นั่นคือ ต่ำกว่ากรณีหน้าต่างสามเหลี่ยม 6 dB และความกว้างของ โหลบหลัก จะมีค่า $8\pi/N$ ซึ่งเท่ากับกรณีหน้าต่างสามเหลี่ยม เพราะฉะนั้นหน้าต่างแฮนนิ่งจึงเป็นที่นิยมมากกว่าหน้าต่างสามเหลี่ยม



รูปที่ 5.8 หน้าต่างเร็สโคไซน์ (แฮนนิ่ง) (ซึ่งมี N=31)

a) ค่าของหน้าต่างในแกนเวลา b) สเปคตรัม c) ขนาดของสเปคตรัมเป็น dB

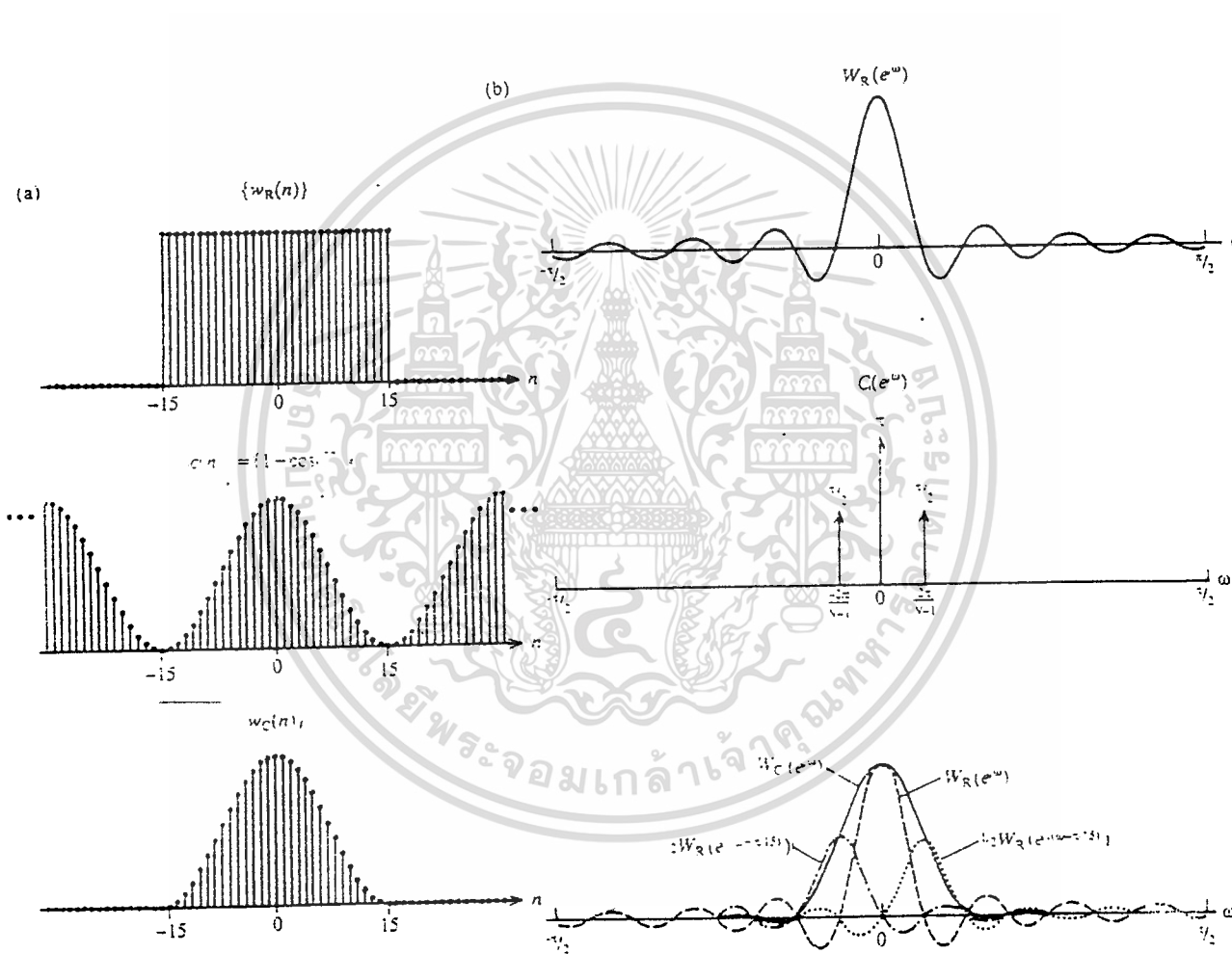
โหลบข้างเคียง มีระดับลดลง เมื่อ $\{w_C(n)\}$ เป็นผลคูณระหว่าง ค่าเร็สโคไซน์ ซึ่งมีจำนวนอนันต์กับค่าหน้าต่างสี่เหลี่ยมคือ

$$w_R(n) = \left[0.5 + 0.5\cos\left(\frac{2\pi n}{(N-1)}\right)\right] w_R(n) \qquad ;for\ all\ n$$
$$= c(n) w_R(n) \qquad ;for\ all\ n \tag{5.14}$$

โดยที่ $\{ c(n) \}$ เป็นค่าเร็สโคไซน์จำนวนอนันต์ ดังแสดงในรูปที่ 4.9 (a) การคูณในแกนเวลาตามความสัมพันธ์ (5.14) นี้ ก็คือ การคอนโวลูชั่นวงกลมของฟูเรียร์ทรานส์ฟอร์มของฟังก์ชั่นทั้งสองในแกนความถี่นั่นเอง เนื่องจาก$\{c(n)\}$ประกอบด้วยค่าคงที่และค่าโคไซน์ที่ความถี่ $(N-1)$ดังนั้น ฟูเรียร์ทรานส์ฟอร์มจึงสามารถเขียนให้อยู่ในรูปของฟังก์ชั่นเดล้าดิเร็ค (Dirac delta function) 3 ฟังก์ชั่นดังนี้

$$C(e^{j\omega}) = \pi\delta(\omega) + 0.5\pi\delta\left(\omega - \frac{2\pi}{N-1}\right) + 0.5\pi\delta\left(\omega + \frac{2\pi}{N-1}\right) \quad ;-\pi \le \omega \le \pi \tag{5.15}$$

เมื่อเราทำการคอนโวลูชั่น $W_R(e^{j\omega})$ และ $C(e^{j\omega})$ สเปคตรัม $W_R(e^{j\omega})$ จะไปปรากฏที่ตำแหน่งของอิมพัลส์ทั้งสามและค่าของแต่ละสเปคตรัม จะถูกปรับไปตามน้ำหนัก ของแต่ละอิมพัลส์ ผลรวมของสเปคตรัมทั้งสามนี้จะให้สเปคตรัมของหน้าต่างแฮนนิ่ง$W_c(e^{j\omega})$ ตามต้องการดังแสดงใน รูปที่ 5.10 (b) การที่สเปคตรัม $W_R(e^{j\omega})$ เคลื่อนไปปรากฏที่ตำแหน่งของอิมพัลส์ทั้งสามนั้นจะส่งผลให้เกิดการหักล้างระหว่างโหลบข้างเคียง ของสเปคตรัมทั้งสาม ทำให้ผลลัพธ์สุดท้ายคือ $W_C(e^{j\omega})$ โหลบข้างเคียงมีขนาดลดลง

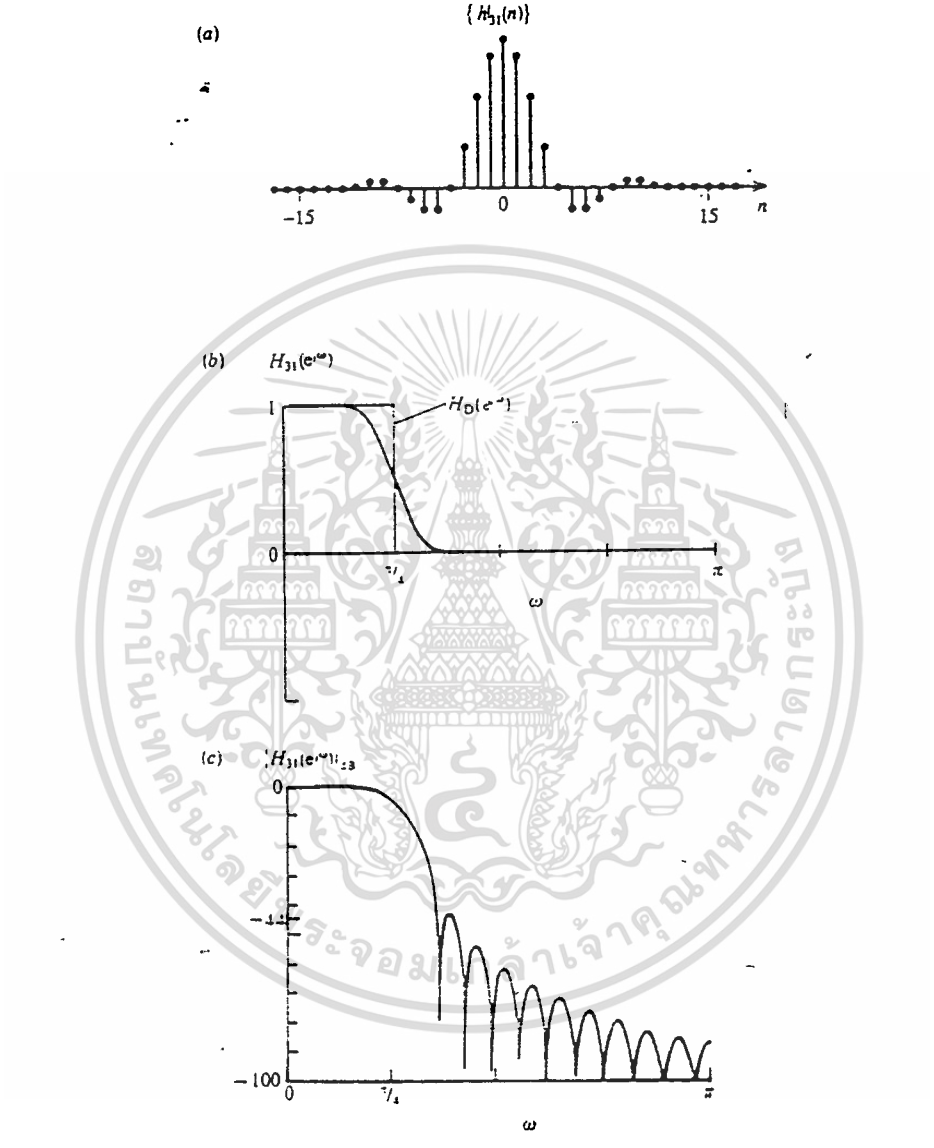


รูปที่ 5.9 การตีความหน้าต่างเร็สโคไซน์หรือหน้าต่างแฮนนิ่ง

(a) แกนเวลา (b) แกนความถี่

รูปที่ 5.10 แสดงผลจากการประยุกต์หน้าต่างแฮนนิ่งเพื่อออกแบบวงจรกรองสัญญาณดิจิตอลประเภท FIRให้มีค่าตอบสนองความถี่ของขนาดใกล้เคียงอุดมคติมากที่สุด ข้อที่เราสังเกตได้ชัดเจนคือ ที่ย่าน

หยุดความถี่มีความสามารถลดทอนสัญญาณสูงขึ้น กล่าวคือค่าสูงสุดของขนาดในย่านหยุดความถี่จะประมาณ -44 dB ที่ความถี่สูงขึ้นไปความสามารถลดทอนสัญญาณก็จะยิ่งสูงขึ้นไปอีก



รูปที่ 5.10 การประยุกต์หน้าต่างเร็สโคไซน์(N=31)กับค่าตอบสนองค่าหนึ่งของโลว์พาสอุดมคติ

a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนสิ้นสุดกล่าวคือ N=31

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

c) การตอบสนองความถี่ของขนาดเป็น dB

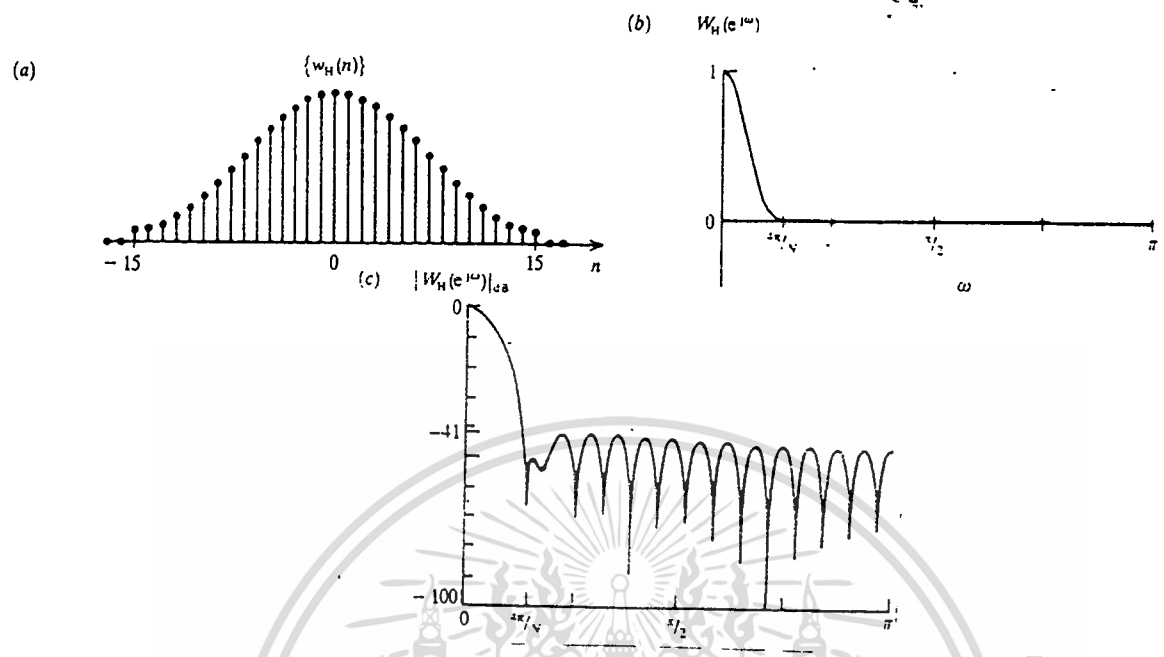
หน้าต่างต่อไปคือหน้าต่างแฮมมิ่งซึ่งสามารถลดระดับของโหลบข้างเคียง ลงไปได้อีก โดยการเพิ่มค่าคงตัวให้แก่ หน้าต่างโคไซน์หน้าต่างแฮมมิ่งมีสัญญลักษณ์ว่า $\{W_H(n)\}$

$$W_R(n) = \begin{cases} 0.54 + 0.46\cos(2\pi n(N-1)) & ;-(N-1)/2 \le n \le (N-1)/2 \\ 0 & ;another\ n \end{cases} \quad (5.16)$$

การที่ระดับของโหลบข้างเคียงแรกลดลงไปมากกว่ากรณีหน้าต่างแฮนนิ่งนั้นเราสามารถจะอธิบายได้จากการที่สเปคตรัมของหน้าต่างสี่เหลี่ยม ($W_R e^{j\omega}$) เคลื่อนและถูกปรับค่าตามที่แสดงในรูปที่ 5.9 (b) ในกรณีของแฮนนิ่งนั้น ค่าฟังก์ชั่นเดลต้าดิแร็คที่ ω=0 และ $2\pi/(N-1)$ มีอัตราส่วน 2:1 (กล่าวคือ 0.5 :0.5/2) ซึ่งส่งผลให้โหลบข้างเคียงแรกของ $W_c(e^{j\omega})$ มีระดับอยู่ที่ประมาณ -31 dB โดยการเปลี่ยนอัตราส่วนนี้เป็น 2.35:1 (0.54:0.46/2) จึงทำให้ระดับของโหลบข้างเคียงแรกของ $W_C(e^{j\omega})$ สามารถลดลงไปได้อีก และกลายเป็นหน้าต่างแฮมมิ่งนั่นเอง ค่าหน้าต่างแฮมมิ่งและสเปคตรัม $W_H(e^{j\omega})$ ของมันกรณี N =31 แสดงอยู่ในรูปที่ 4.11 ซึ่งขนาดหรือระดับของลอนข้างเคียงแรก จะลดลงไปอยู่ที่ -41 dB หรือนั่นคือต่ำกว่าของหน้าต่างแฮนนิ่งอยู่ 10 dB อย่างไรก็ดีก็มีข้อเสียเปรียบอื่น เมื่อเทียบกับ หน้าต่างแฮนนิ่งกล่าวคือ โหลบข้างเคียงที่ความถี่สูงของหน้าต่างแฮนนิ่งมีระดับที่ลดลงตามลำดับ ขณะที่โหลบข้างเคียงที่ความถี่สูงของหน้าต่างแฮมมิ่งมีระดับที่เกือบคงที่

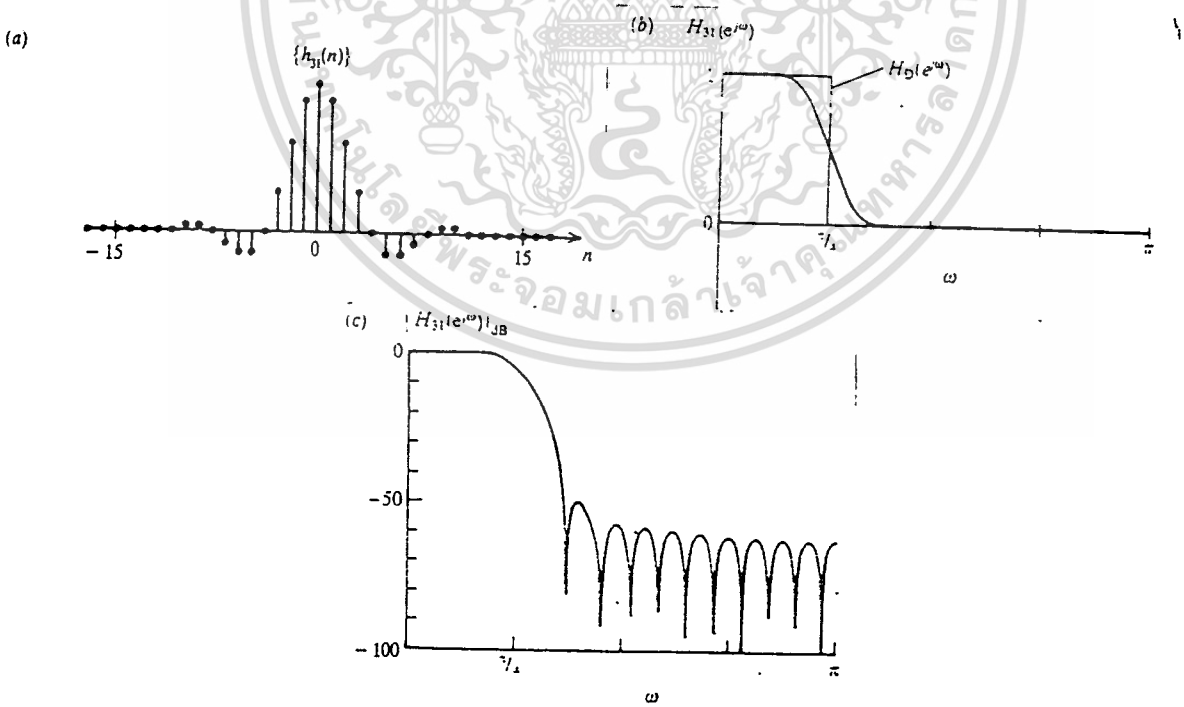
รูปที่ 5.12 แสดงผลการประยุกต์ หน้าต่างแฮมมิ่งในการออกแบบวงจรกรองสัญญาณดิจิตอลประเภท FIR ให้มีค่าตอบสนองความถี่ของขนาดใกล้เคียงโลว์พาสอุดมคติมากที่สุด ข้อที่สังเกตได้คือ โหลบข้างเคียงแรกมียอดอยู่ที่ -51 dB หรือนั่นคือต่ำกว่ากรณีที่ใช้หน้าต่างแฮนนิ่งอยู่ -7 dB อย่างไรก็ดีเมื่อความถี่สูงขึ้น ความสามารถที่จะกำจัดสัญญาณจะไม่เพิ่มขึ้นเท่ากับในกรณีวงจรกรองสัญญาณที่ออกแบบโดยหน้าต่างแฮนนิ่ง

จะเห็นว่าความสามารถกำจัดสัญญาณในย่านหยุดความถี่ของวงจรกรองสัญญาณนั้นจะขึ้นอยู่กับระดับของโหลบข้างเคียง ของฟังก์ชั่นหน้าต่าง แม้นว่าหน้าต่างจะสามารถให้การลดทอนถึง -51 dB ดังแสดงในรูปที่ 5.12 ก็ตาม แต่ก็อาจจะไม่เพียงพอสำหรับงานบางประเภท ฉะนั้นหน้าต่างที่มีระดับโหลบข้างเคียง ต่ำกว่าของหน้าต่างแฮมมิ่งจึงเป็นสิ่งจำเป็น หน้าต่างแฮมมิ่งสามารถลดระดับลอนข้างเคียงได้ โดยยังรักษาระดับความกว้างของลอนข้างเคียงหลักอยู่ ต่อไปเราจะศึกษาหน้าต่าง ซึ่งระดับของโหลบข้างเคียง สามารถลดลงได้อีก แต่จำเป็นต้องขยายความกว้างของโหลบหลัก ตัวอย่างเช่น หน้าต่างแบล็กแมนที่จะกล่าวดังต่อไปนี้



รูปที่ 5.11 หน้าต่างแฮมมิ่ง (N = 31)

a) ค่าของหน้าต่างในแกนเวลา b) แกนความถี่ c) ขนาดของสเปกตรัมเป็น dB



รูปที่ 5.12 ผลจากการประยุกต์หน้าต่างแฮมมิ่งซึ่ง N = 31

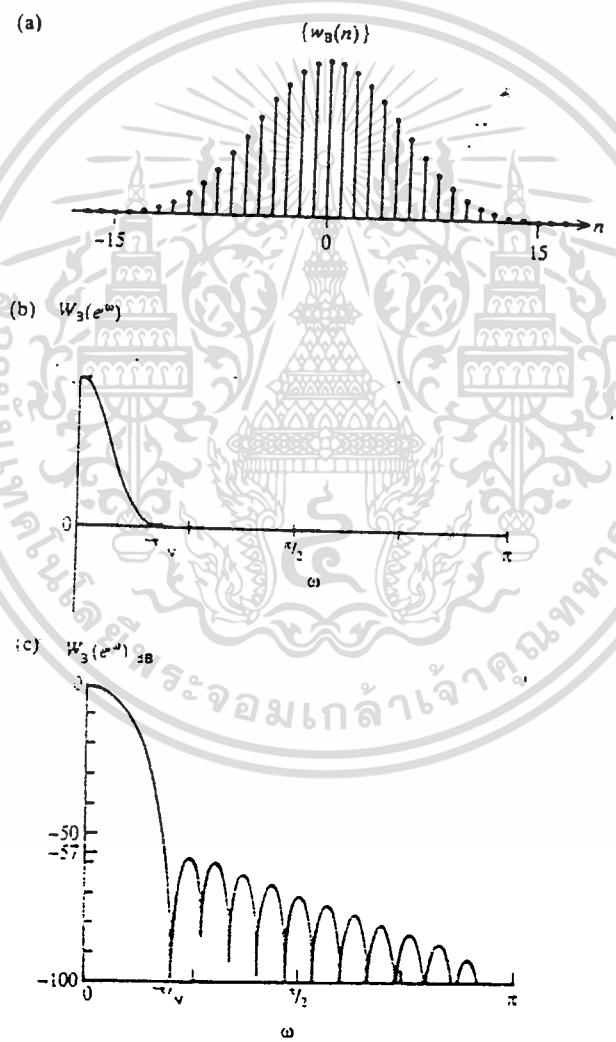
a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนสิ้นสุดกล่าวคือ N = 31

b) การตอบสนองความถี่ c) การตอบสนองความถี่ขนาดเป็น dB

หน้าต่างแบล็คแมนสามารถลดระดับของโหลบข้างเคียงต่อไปได้อีกโดยเพิ่มจำนวนโคไซน์มากขึ้น ดังนี้

$$w_B(n) = 0.42 + 0.5\cos(2\pi n/(N-1)) + 0.80\cos(2\pi n/(N-1))$$
$$;-(N-1)/2 \le n \le (N-1)/2$$
$$= 0 \qquad ;all\ of\ n \tag{5.17}$$

ดังแสดงในรูปที่ 5.13 โดยที่ขนาดของโหลบข้างเคียงแรกลดลงไปอยู่ที่ -57 dB แต่ความกว้างของโหลบหลัก เพิ่มไปเป็น $12\pi/N$



รูปที่ 5-13 หน้าต่างแบล็คแมน (N=31)

a) ค่าของหน้าต่าง

b) แกนความถี่

c) ขนาดของสเปคตรัมเป็น dB

การลดลงของระดับของ โหลบข้างเคียง สามารถอธิบายได้ในทำนองเดียวกับในกรณีที่เคยอธิบายมาแล้วในกรณีของหน้าต่างแฮนนิ่งกล่าวคือ เราสามารถเขียนหน้าต่างแบล็คแมนในรูปผลคูณระหว่างค่า จำนวนอนันต์ของแบล็คแมนและค่าของหน้าต่างสี่เหลี่ยมดังนี้

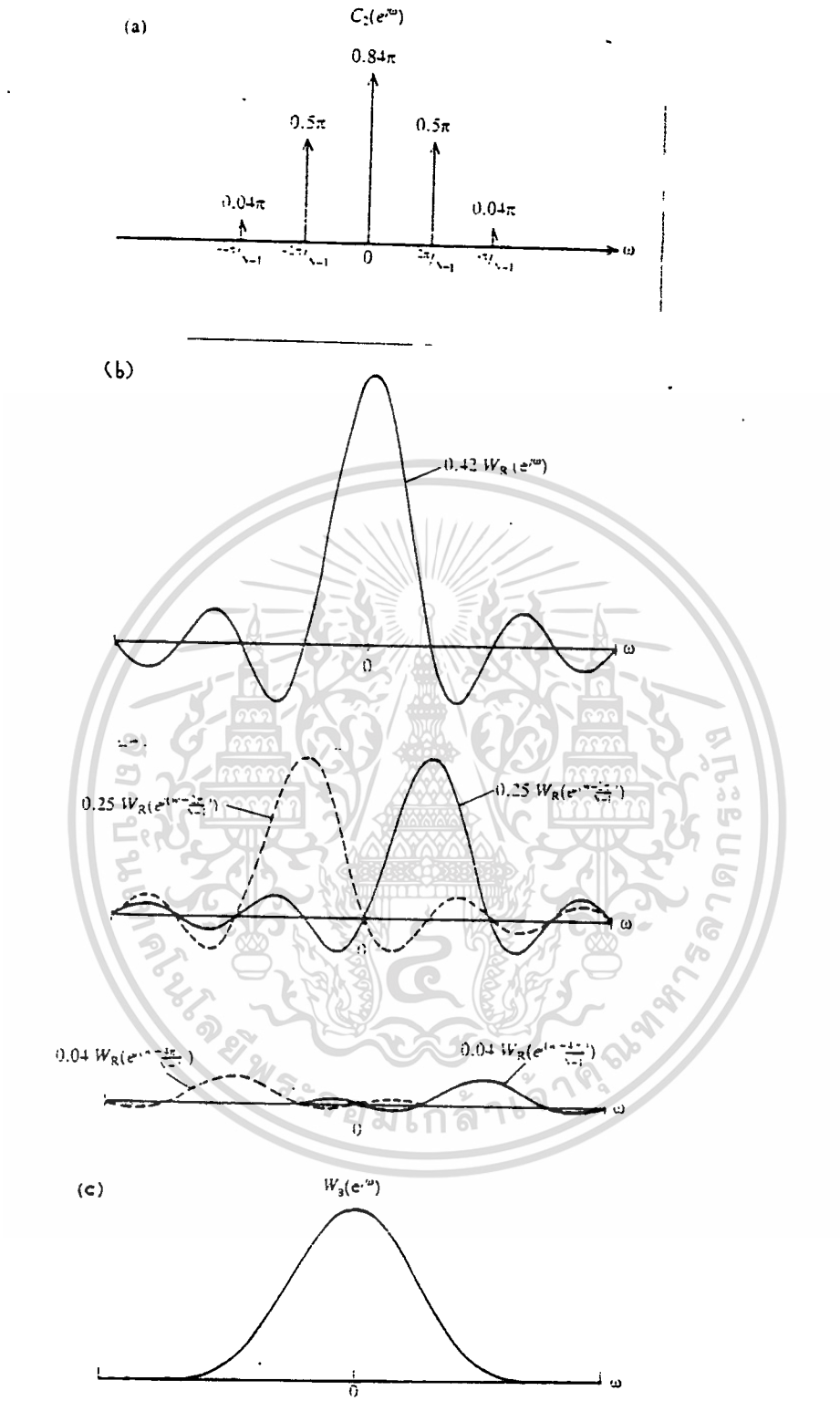
$$\begin{aligned} w_B(n) &= \left[0.42 + 0.45\cos(2\pi n/(N-1)) + 0.08\cos(4\pi n/(N-1))\right] w_R(n) \\ &= C_2(n) w_r(n) \qquad ;all\ of\ n \end{aligned} \tag{5.18}$$

โดยที่ $\{W_R(n)\}$ เป็นค่า หน้าต่างสี่เหลี่ยมจำนวน N ค่า และ $\{C_2(n)\}$ เป็นค่าจำนวนอนันต์ของแบล็คแมนจากที่เคยทราบแล้วว่าการคูณระหว่างสัญญาณสองสัญญาณในแกนเวลาก็คือการคอนโวลูชั่นของสเปคตรัม ของสัญญาณทั้งสองในแกนความถี่ เนื่องจาก $\{C_2(n)\}$ ประกอบด้วยค่าคงที่ค่าหนึ่ง และฟังก์ชั่นโคไซน์สองตัวที่ความถี่ $w = 2/(N-1)$ และ $w = 4/(N-1)$ ดังนั้นจึงเขียนฟูเรียร์ทรานส์ฟอร์มของ (5.18) ได้ว่า

$$\begin{aligned} C_2(e^{j\omega}) = {} & 0.84\pi\delta(\omega) + 0.5\pi\,\delta(\omega - \frac{2\pi}{N-1}) + 0.5\pi\,\delta(\omega + \frac{2\pi}{N-1}) \\ & + 0.04\pi\delta(\omega - \frac{4\pi}{N-1}) + 0.04\pi\delta(\omega + \frac{4\pi}{N-1}) \qquad ;\pi \le \omega \le \pi \end{aligned} \tag{5.19}$$

ซึ่งแสดงอยู่มนที่ 5.14 a) ส่วนรูปที่ 5.14 b) แสดงการที่ $W_R(e^{j\omega})$ เคลื่อนไปอยู่ที่ตำแหน่งของอิมพัลส์ต่างๆ และถูกปรับค่าตามน้ำหนักของค่าอิมพัลส์นั้นและรูปที่ 5.14 c) แสดงสเปคตรัมของหน้าต่างแบล็คแมน $W_B(e^{j\omega})$ ซึ่งเป็นผลรวมของบรรดา $W_R(e^{j\omega})$ ที่ถูกปรับค่าและปรากฏที่ตำแหน่งอิมพัลส์ต่างๆ ในรูปที่ 5.15 b) เราจะสังเกตว่าความกว้างของโหลบหลัก ที่กว้างขึ้นเนื่องมาจาก $W_R(e^{j\omega})$ ที่ถูกปรับค่าและปรากฏที่ $\omega = \pm 4\pi/(N-1)$ แต่ในขณะเดียวกันก็จะช่วยลดระดับของ โหลบข้างเคียง ด้วย

รูปที่ 5.15 แสดงผลจากการประยุกต์หน้าต่างแบล็คแมนในการออกแบบวงจรกรองสัญญาณFIR ให้มีค่าตอบสนองความถี่ใกล้เคียงโลว์พาสอุดมคติมากที่สุด ระดับของลอนข้างเคียงจะลดลงไปอยู่ที่ -74 dB แต่ว่าเกิดขึ้นเมื่อ $\omega > \pi/2$ การที่ย่านเปลี่ยนแปลงมีขนาดกว้างกว่าที่พบในตัวอย่างที่ใช้หน้าต่างอื่นที่ผ่านมา เนื่องจากหน้าต่างแบล็คแมน มี main lobe ที่กว้างกว่าหน้าต่างอื่นๆ

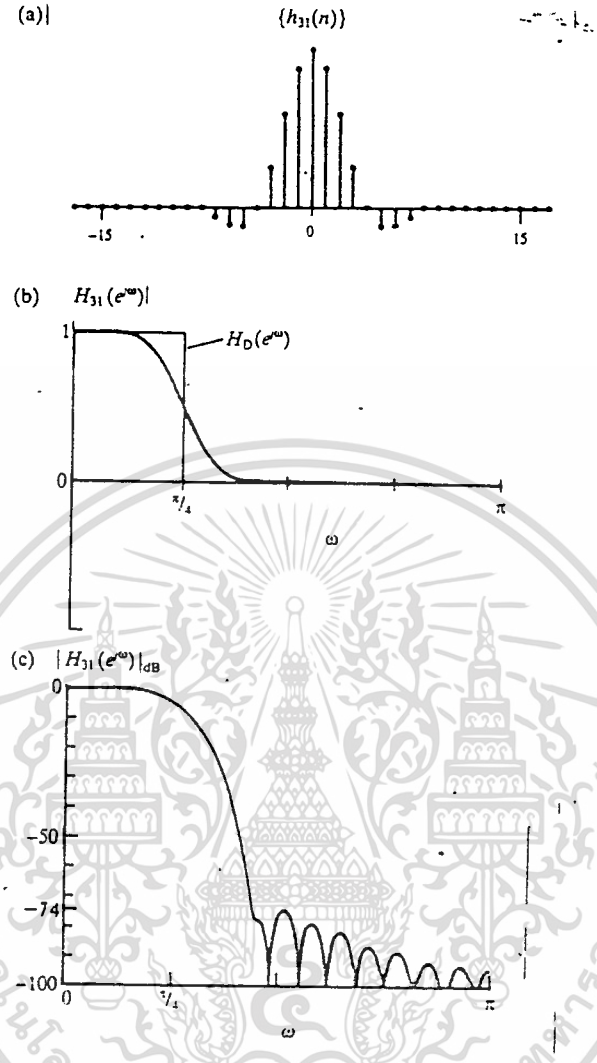


รูปที่ 5.14 การตีความหน้าต่างแบล็คแมนในแกนความถี่

a) สเปคตรัมของฟังก์ชั่นแบล็คแมนซึ่งมีค่าถึงอนันต์

b) สเปคตรัมของหน้าต่างสี่เหลี่ยมที่ถูกปรับค่า และ เคลื่อนไปอันเนื่องจากการคอนโวลูชั่น

c) สเปคตรัมของหน้าต่างแบล็คแมนซึ่งเป็นผลบวกของแต่ละสเปคตรัม



รูปที่ 5.15 ผลจากการประยุกต์หน้าต่างแบล็คแมน ซึ่ง N=31

a) ค่าตอบสนองค่าหนึ่งซึ่งมีจำนวนสิ้นสุด

b) การตอบสนองความถี่ของทรานส์เฟอร์ฟังก์ชั่นเทียบกับอุดมคติ

c) การตอบสนองความถี่ขนาดเป็น dB

ตารางที่ 5.1 คุณสมบัติสเปคตรัมของหน้าต่างจำนวน N ค่า

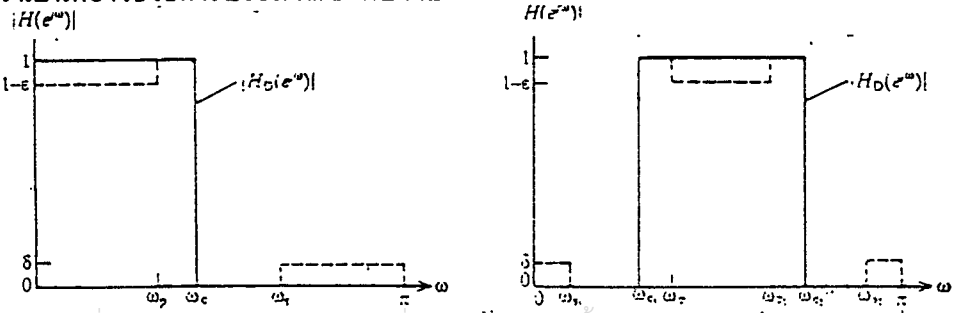
window	ความกว้างของ main lobe	ระดับสูงสุดของ side lobe (dB)
Rectancular	$4\pi/N$	-13
Barlett	$8\pi/N$	-25
Hanning	$8\pi/N$	-31
Hamming	$8\pi/N$	-41
Blackman	$12\pi/N$	-57

5.2 การออกแบบวงจรกรองสัญญาณดิจิตอลแบบFIR โดยการผสมวิธี DFT และวิธีหน้าต่าง

เราจะรวมวิธีหน้าต่างและวิธี DFT เข้าด้วยกัน เพื่อออกแบบวงจรกรองสัญญาณดิจิตอลแบบ FIR การรวมวิธีทั้งสองเข้าด้วยกันช่วยกำจัดข้อเสียเปรียบของแต่ละวิธีออกไปได้ ในวิธีหน้าต่างเราจำเป็นต้องทราบค่าตอบสนองความถี่ที่ต้องการ$\{h_D(n)\}$ซึ่งพบว่าบ่อยครั้งเมื่อเรากำหนดขนาด $|H_D(e^{j\omega})|$ และเฟสArg $|H_D(e^{j\omega})|$ ที่ต้องการมาให้ เราไม่สามารถหา $\{h_D(n)\}$ จากสูตรสำเร็จได้ แต่วิธี DFT จะช่วยเราได้โดยให้$\{h_D(n)\}$ ซึ่งเป็นค่าจำนวนสิ้นสุดที่ประมาณใกล้เคียงกับ$\{h_D(n)\}$ ที่ต้องการ โดยต้องทำให้จำนวนจุด N_d มีค่าสูงพอ อย่างไรก็ดีในวิธี DFT โดยลำพังนั้นจะพบว่าวงจรกรองสัญญาณที่ได้จะมีทรานส์เฟอร์ฟังก์ชั่น $H_d(e^{j\omega})$ที่ต้องการในลักษณะมีโอเวอร์ชูตและ อันเดอร์ชูต ซึ่งลักษณะนี้จะทำให้ลดลงได้โดยใช้วิธีหน้าต่างเข้าช่วยนอกจากนี้วิธีหน้าต่างยังอนุญาตให้เราสามารถลดจำนวนของ$\{h_d(n)\}$ลงได้ด้วยซึ่งทำได้โดยที่ไม่ฝืนเงื่อนไข ของข้อกำหนดขนาดและเฟสที่ต้องการการลดจำนวนของ $\{h_d(n)\}$ ไม่สามารถทำได้ด้วยวิธีDFT เพียงลำพัง

พิจารณาการหาค่าขนาดที่ต้องการ $\{|H_D(k)|\}$ จากเงื่อนไขข้อกำหนดของขนาดที่ให้มา จากนั้น จะพิจารณากำหนดค่าเฟสที่เหมาะสม จากค่าขนาดและ เฟสที่ได้ เราจะทำการหา อินเวิสฟูเรียร์ทรานส์ฟอร์มซึ่งจะให้ค่าตอบสนองค่าหนึ่ง $\{h_d(n)\}$ จำนวนจำนวน N_dค่าต่อไปเราใช้ค่าหน้าต่างจำนวน N ค่า โดยที่ $N < N_d$ เพื่อหาค่า $\{h_N(n)\}$ จาก$\{h_d(n)\}$

การกำหนดค่าขนาดที่เหมาะสม เมื่อใช้วิธีหน้าต่างออกแบบวงจรสัญญาณดิจิตอลแบบFIR เราจะพบว่าผลที่ได้รับข้อหนึ่งก็คือ การตอบสนองความถี่ของขนาดของวงจรจะมีย่านผ่านความถี่ที่แคบกว่าย่านผ่านความถี่ที่ต้องการซึ่งเป็นผลจากการคอนโวลูชั่นระหว่าง โหลบหลักของสเปคตรัมหน้าต่างกับทรานส์เฟอร์ฟังก์ชั่น $H_D(e^{j\omega})$ ที่ต้องการ ดังนั้น เราควรกำหนดให้ค่าขนาดที่ต้องการ $|H_D(e^{j\omega})|$ มีความถี่คัทออฟเลยเข้าไปในย่านเปลี่ยนแปลง ดังแสดงในรูปที่ 5.16 การที่จะให้เลยเข้าไปเท่าไหร่นั้นจะขึ้นอยู่กับความกว้างของ โหลบหลัก ของสเปคตรัมหน้าต่างซึ่งจะสัมพันธ์จำนวนของค่าตอบสนองค่าหนึ่งสุดท้ายที่ใช้และประเภทของหน้าต่างที่ใช้อย่างไรก็ดีเนื่องจากในตอนเริ่มต้นเรายังไม่มีโอกาสจะทราบจำนวนของค่าตอบสนองค่าหนึ่งสุดท้ายดังกล่าวได้ เราจึงจำเป็นต้องใช้วิธีการประมาณซึ่งพบว่าประมาณ30 % ของความกว้างของย่านเปลี่ยนแปลง



รูปที่ 5.16 ขอบเขตข้อกำหนดของโลว์พาสส์ และแบนด์พาสที่ต้องการ

วิธีการดังกล่าวข้างต้นนี้ ทำให้เรากำหนดค่าขนาดที่ต้องการ $|H_D(e^{j\omega})|$ ที่เหมาะสมได้จากนั้นจึงทำการสุ่ม$H_D(k)$จำนวน N_d จุดระหว่าง $0<\omega<2\pi$ เพื่อให้ได้ค่า $\{H_D(k)\}$ เราจะใช้ค่า N_d ที่ค่อนข้างสูงมากกว่าจำนวนของค่าตอบสนองค่าหนึ่งของวงจรกรองสัญญาณที่จะได้รับจากการออกแบบนี้ โดยปกติแล้ว N_d = 128 จะเพียงพอสำหรับวงจรทั่วไป

เพื่อป้องกันความพลาดพลั้งที่อาจเกิดจากการหาค่า $|H_D(k)|$ นั้นเราจะสุ่ม $|H_D(e^{j\omega})|$ ที่ $(N_d/2)+1$ จุดระหว่าง $0<\omega<\pi$ ก่อน จากนั้นจึงใช้คุณสมบัติสมมาตรคู่ (even symmetry) ของการตอบสนองความถี่ของขนาดหาค่าที่เหลือ ซึ่งกระบวนการดังกล่าวนี้ทำให้ได้ว่า

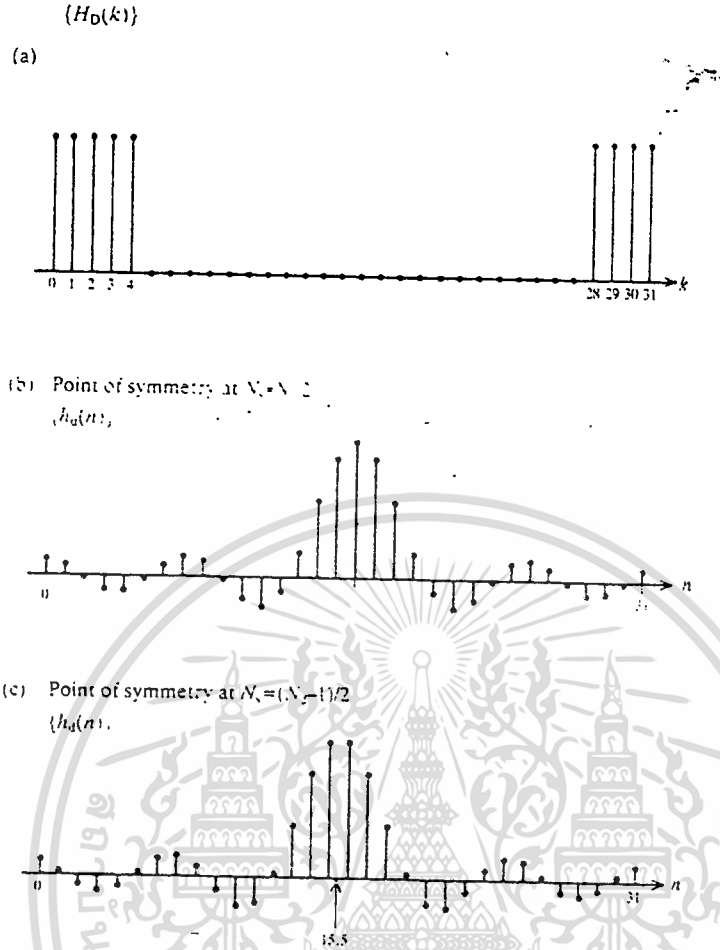
$$|H_D(k)| = |H_D(e^{j\omega})|_{\omega = e^{j2\pi k/Nd}} \quad \text{สำหรับ} 0 \le k \le N_d/2$$

$$|H_D(k)| = |H_D(N_d - k)| \quad \text{สำหรับ} \quad (N_d/2)+1 \le k \le N_d-1 \qquad (5.20)$$

การกำหนดค่าเฟสเชิงเส้นที่เหมาะสม

เราเริ่มจากคุณสมบัติที่สำคัญ ของวงจรกรองสัญญาณดิจิตอลแบบ FIR ที่ว่าหากเราต้องการให้ค่าเฟสของวงจรดังกล่าวมีความเป็นเชิงเส้นแล้วค่าตอบสนองค่าหนึ่งที่เป็นเชิงเหตุ (causal) จำนวน N_d ค่าจะต้องมีความ สมมาตร (ทั้งสมมาตรคู่และสมมาตรคี่) รอบจุดๆหนึ่ง ซึ่งอยู่ประมาณกึ่งกลาง หรือที่ $N_s = (N_d-1)/2$ ความสมมาตรมีความเป็นไปได้เพียง 4 กรณีเท่านั้น กล่าวคือ กรณีสมมาตรคู่และสมมาตรคี่ และในแต่ละกรณีจะแล้วแต่ว่าจุดสมมาตรอาจอยู่ตรงจุดสุ่มพอดีหรืออยู่ระหว่างจุดสุ่มถ้าเราใช้วิธีฟาสท์ฟูเรียร์ทรานส์ฟอร์มในการคำนวณหาอินเวสฟูเรียร์ทรานส์ฟอร์มแล้ว N_d จะต้องเป็นเลขคู่ ดังนั้นจุดสมมาตรจะไม่ตรงกับจุดสุ่มพอดีแต่จะอยู่ระหว่างจุดสุ่ม ในตัวอย่างของโลว์พาสส์อุดมคติ ซึ่งค่าเฟสเป็นศูนย์ซึ่งอธิบายในหัวข้อที่ 5.2 ค่าตอบสนองค่าหนึ่งจำนวน N_d ค่าเกือบจะสมมาตรรอบจุด n =0 หากไม่ใช่เพราะว่ามีค่า $n = N_d/2$ ที่ปราศจากคู่ของมัน อย่างก็ดีกรณีดังกล่าวนี้เราสามารถใช้ค่าหน้าต่างจำนวน N ค่า โดยที่ $N < N_d$ ทำการกำจัดค่าที่ $n = N_d/2$ ได้ ทำให้ $\{h_N(n)\}$ ที่ได้รับมีความสมมาตรรอบจุดสุ่มพอดีค่าเฟสเชิงเส้นที่เหมาะสมและสอดคล้องกับค่าขนาดจะถูกกำหนดโดยคุณสมบัติความสมมาตรของค่าตอบสนองค่าหนึ่งดังกล่าวมาข้างต้น

ค่าเฟสเชิงเส้นในแต่ละกรณีของสมมาตรแสดงอยู่ในตารางที่ 5.2 ในการคำนวณหาค่าจริงและค่าจินตภาพ จากค่าขนาดและเฟส เราจะใช้ค่าเฟสจากตารางที่ 5.2 นี้



รูปที่ 5.17 ผลที่ได้รับจากการเลือกความชันของเฟสเชิงเส้นที่มีค่าตอบสนองค่าหนึ่ง
a) ค่าขนาดโลว์พาสส์อุดมคติที่ถูกสุ่มด้วย 32 จุด และใช้รวมกับค่าเฟสเชิงเส้นในการคำนวณหาอินเวสฟูเรียร์ทรานส์ฟอร์ม
b) ค่าตอบสนองค่าหนึ่งเมื่อเราให้ความชันของเฟสที่มีค่า -16
c) ค่าตอบสนองค่าหนึ่งเมื่อเราให้ความชันมีค่า -15.5

ตารางที่ 5. 2 ค่าตอบสนองความถี่ของเฟสเชิงเส้นในแต่ละกรณีของสมมาตรของค่าตอบสนองค่าหนึ่ง$\{h_d(n)\}$ จำนวน N_d ค่า (N_d เป็นจำนวนคู่)

สมมาตรคู่โดยจุดสมมาตรตรงจุดสุ่ม $N_s = N_d/2$ $Arg[H_D(e^{j\omega})] = -\omega N_d/2$ **สำหรับ** $0 \le \omega \le \pi$
สมมาตรคู่โดยจุดสมมาตรอยู่ระหว่างจุดสุ่ม $N_s = (N_d-1)/2$ $Arg[H_D(e^{j\omega})] = -\omega(N_d-1)/2$ **สำหรับ** $0 \le \omega \le \pi$
สมมาตรคี่โดยจุดสมมาตรอยู่ตรงจุดสุ่ม $N_s = N_d/2$ $Arg[H_D(e^{j\omega})] = -\omega N_d/2 + \pi/2$ **สำหรับ** $0 \le \omega \le \pi$
สมมาตรคี่โดยจุดสมมาตรอยู่ระหว่างจุดสุ่ม $N_s = (N_d-1)/2$ $Arg[H_D(e^{j\omega})] = -\omega(N_d-1)/2 + \pi/2$ **สำหรับ** $0 \le \omega \le \pi$

ที่ความถี่ซึ่งค่าขนาดเป็นศูนย์นั้นเราจะให้ค่าเฟสเป็นอะไรก็ได้ โดยปกติเพื่อความสะดวกเราก็มักจะให้เป็นค่าเชิงเส้นตามตารางที่ 5.2 ดังนั้นที่ค่าเฟสของกรณีสมมาตรคี่จะเท่ากับ $\pi/2$ จากความรู้เรื่องการแปลง Z เราพบว่าค่าตอบสนองค่าหนึ่งซึ่งมีลักษณะสมมาตรคู่และจำนวนค่าเป็นจำนวนคู่ด้วยนั้นการแปลงZ ของมันจะมีซีโร่ที่ z =-1 เสมอ หรือนั่นก็คือ ค่าตอบสนองความถี่ของขนาดจะเป็นศูนย์ที่ $\omega=\pi$ กรณีเช่นนี้จะเกิดเมื่อจุดสมมาตรอยู่ระหว่างจุดที่สุ่มที่ $N_s=(N_d-1)/2$ ทำให้ที่ $\omega=\pi$ ค่าเฟสของมันเท่ากับ $-(N_d-1)\pi/2$

การหาค่าเฟสที่ความผิดพลาดที่จะเกิดขึ้นทำได้โดยการสุ่ม $\mathrm{Arg}[H_D(e^{j\omega})]$ ที่ $N_d/2+1$ จุดระหว่าง $0\le\omega\le\pi$ แล้วใช้คุณสมบัติที่ว่าเฟสมีความเป็นสมมาตรคี่เสมอ หาค่าส่วนที่เหลือ ทำให้ได้ว่า

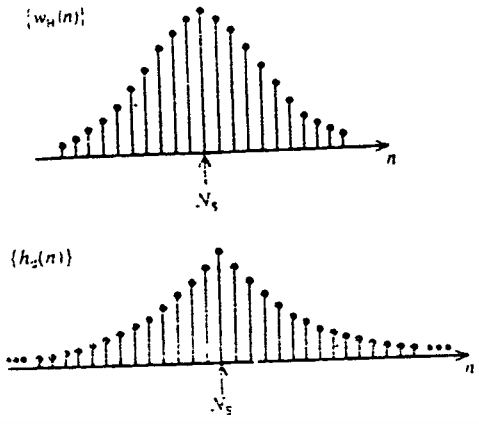
$$\mathrm{Arg}[H_D(k)]=\mathrm{Arg}[H_D(e^{j\omega})]_{\omega}=e^{j2\pi k/Nd} \quad \text{สำหรับ } 0\le k\le N_d/2$$

$$\mathrm{Arg}[H_D(k)]=-\mathrm{Arg}[H_D(N_d-k)] \quad \text{สำหรับ } (N_d/2)+1\le k\le N_d-1 \qquad (5.21)$$

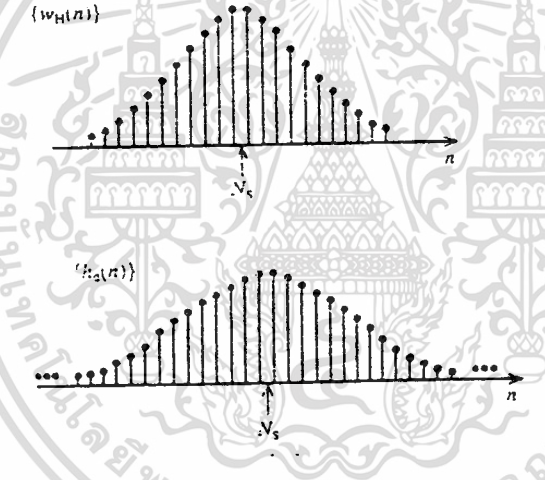
ลองพิจารณากรณีค่าตอบสนองความถี่ของขนาดของโลว์พาสส์อุดมคติ ซึ่งมีความถี่คัทออฟ $\omega_c=\pi/4$ และได้รับการสุ่มระหว่าง $0\le\omega\le2\pi$ ด้วย จำนวนจุด $N_d=32$ จากที่เรามีความรู้ว่าเมื่อค่าตอบสนองค่าหนึ่งมีลักษณะสมมาตรคี่แล้วทรานส์เฟอร์ฟังชั่นของมันจะมีซีโร่ที่ $\omega=0$ ซึ่งจะไม่เหมาะสำหรับกรณีโลว์พาสมีขนาดที่ไม่เป็นศูนย์ที่ $\omega=0$ เพราะฉะนั้นเราจะเลือกค่าตอบสนองค่าหนึ่งที่มีลักษณะสมมาตรคู่ซึ่งจากตารางที่ 5.2 จะพบว่าทำให้เฟสเชิงเส้นเป็นไปได้เพียง 2 ลักษณะเท่านั้น คือ ลักษณะแรกคือมีเฟส -16ω โดย $0\le\omega\le\pi$ และจุดสมมาตรอยู่ที่ $N_s=16$ ซึ่งตรงกับจุดสุ่มพอดีส่วนลักษณะที่สองคือมีเฟส $-15.5\ \omega$ โดย $0\le\omega\le\pi$ และจุดสมมาตรอยู่ที่ $N_s=15.5$ ซึ่งอยู่ระหว่างจุดสุ่ม จากนั้นทำให้เราคำนวณหาค่าตอบสนอง ค่าหนึ่ง $\{h_d(n)\}$ ของแต่ละลักษณะของเฟสได้ดังแสดงในรูปที่ 5.17

การประยุกต์หน้าต่าง ตอนนี้เราสามารถหา $\{h_d(n)\}$ จำนวน N_d ค่าซึ่งจะประมาณใกล้เคียง $\{h_D(n)\}$ ที่ต้องการ แต่ $\{h_d(n)\}$ นี้อาจให้ค่าขนาดที่อยู่ภายในเงื่อนไขข้อกำหนดของขนาดมากเกินไปฉะนั้นหากเราลดจำนวนของ$\{h_d(n)\}$ จาก N_d ให้เป็น N โดยการใช้ค่าหน้าต่างจำนวน N เราก็จะได้วงจรกรองสัญญาณดิจิตอลแบบ FIR ที่ตรงตามเงื่อนไขข้อกำหนดได้พอดี

เพื่อให้ได้เฟสของวงจรกรองสัญญาณดิจิตอลแบบFIR มีความเป็นเชิงเส้น หน้าต่างที่ใช้จะต้องมีค่าตอบสนองค่าหนึ่ง $\{h_N n)\}$ที่สมมาตรด้วย และเพื่อรักษาลักษณะสมมาตรของ$\{h_d(n)\}$ เราต้องเลือก N ของหน้าต่างให้คล้องจองกันด้วย กล่าวคือ หากจุดสมมาตรของ $\{h_d(n)\}$ ถูกเลือกให้ตกลงที่จุดสุ่มพอดี จำนวน N ของหน้าต่างจะต้องเป็นเลขคี่ แต่หากจุดสมมาตรตกลงระหว่างจุดสุ่มแล้ว เราต้องเลือกจำนวน N ของหน้าต่างให้เป็นเลขคู่ดังแสดงในรูปที่ 5.18



จุดสมมาตร N_s อยู่ระหว่างจุดสุ่ม



รูปที่ 5.18 การเลือกจำนวน N_s และกำหนดตำแหน่ง N_s ของหน้าต่างแฮมมิ่ง เพื่อรักษาความสมมาตรที่เดิมมีอยู่ใน$\{h_d(n)\}$

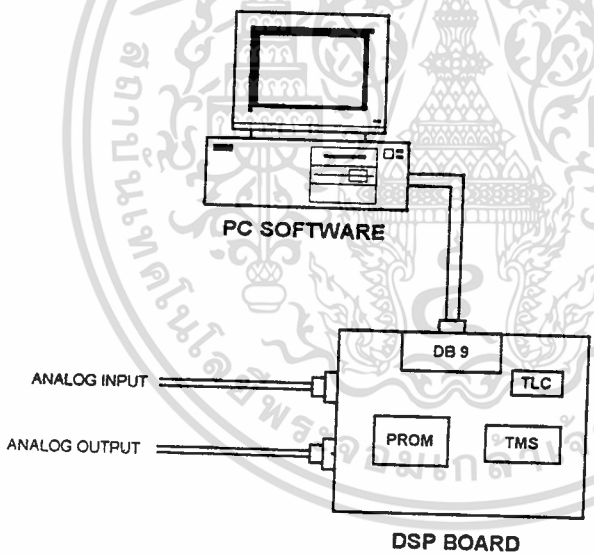
สรุป จากวิธีการออกแบบวงจรกรองสัญญาณดิจิตอลแบบFIR สองวิธี วิธีแรกเรียกว่าหน้าต่าง ซึ่งจะมีประโยชน์ในการออกแบบจากค่าตอบสนองค่าหนึ่งที่ทราบค่า หรือเป็นสูตรสำเร็จซึ่งได้จากการคำนวณอินเวสฟูเรียร์ทรานส์ฟอร์มจากค่าขนาดและเฟสที่กำหนดมาให้วิธีที่สองเรียกว่าวิธีDFT /หน้าต่าง ซึ่งค่าตอบสนองค่าหนึ่งจะคำนวณโดยการหาอินฟูเรียร์ทรานส์ฟอร์มไม่ต่อเนื่องจากค่าขนาดที่ต้องการและค่าเฟสเชิงเส้นที่เหมาะสม จากนั้นจะใช้ค่าหน้าต่างช่วยลดจำนวนค่าตอบสนองค่าหนึ่งลงและขณะเดียวกันช่วยกำจัด ripple ในย่านผ่านความถี่และการรั่วในย่านหยุดความถี่ด้วย

บทที่ 6
รายละเอียดเกี่ยวกับโครงงาน

ในโครงงานนี้เป็นการใช้บอร์ดDSP (Digital Signal Processing) มาประยุกต์ใช้งานด้านดิจิตอลฟิลเตอร์ ซึ่งดิจิตอลฟิลเตอร์นั้นมีด้วยกันหลายชนิด ไม่ว่าจะเป็นแบบ FIR (Finite Impulse Response Filter) หรือแบบ IIR (Infinite Impulse Response Filter) และอาจแบ่งเป็นตัวกรองแบบป้อนกลับและแบบไม่ป้อนกลับก็ได้ ในการประยุกต์ใช้บอร์ด DSP นี้ จะใช้การออกแบบวงจรกรองแบบ FIR ซึ่งวงจรกรองแบบ FIRนี้ มีข้อดีอยู่ 3 ประการคือ

1. เป็นระบบที่เสถียร (stable)
2. เป็นระบบที่เป็นจริง(realizable)
3. สามารถออกแบบให้เป็นเฟสเชิงเส้นได้ง่าย(linear phase)

6.1 โครงสร้างทางฮาร์ดแวร์



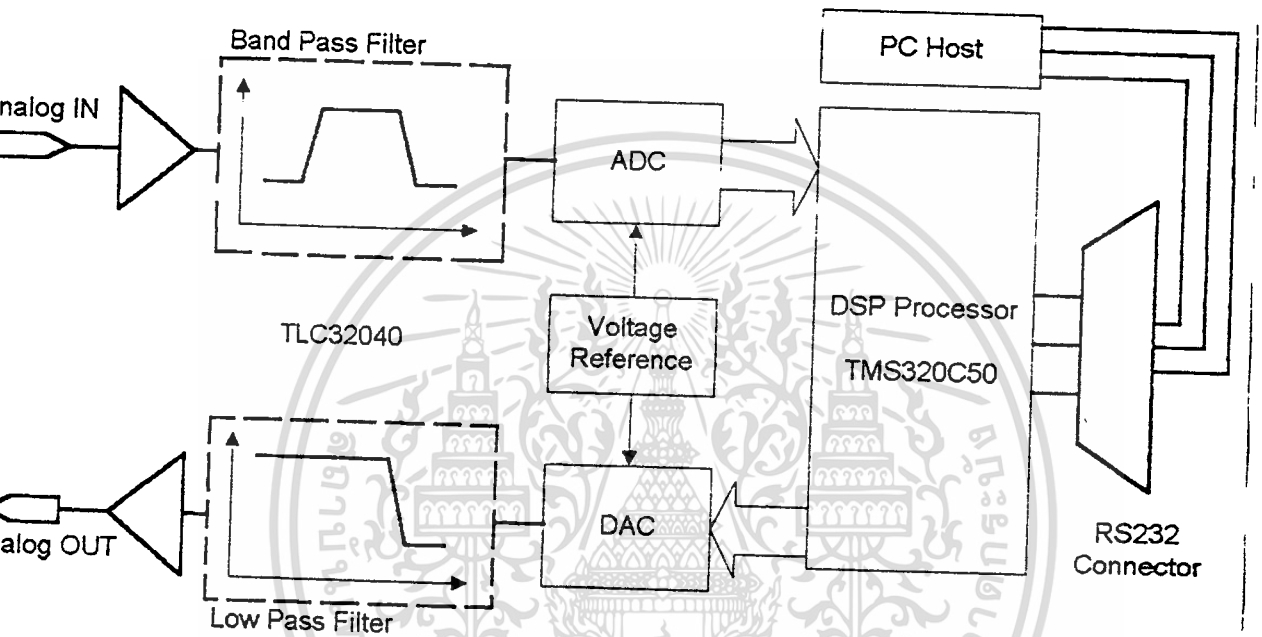
รูปที่ 6.1 แสดงโครงสร้างทางฮาร์ดแวร์ ของโครงงาน

จากโครงสร้างของบอร์ดเราสามารถเขียนเป็นบล็อกไดอะแกรมของระบบ DSP ที่ใช้ในโครงงานดังรูปที่ 6.2

จากระบบ DSP ตามบล็อกไดอะแกรมจะประกอบด้วย สัญญาณอนาลอกเข้ามาจะผ่านวงจรกรองแบบแบนด์พาส เพื่อที่จะกำจัดช่วงความถี่ของสัญญาณอินพุท และยังช่วยโปรเซสเซอร์ไม่ให้ทำงานนอกเหนือความถี่ทำงาน จากนั้นจะส่งต่อไปยังวงจรแปลงสัญญานอนาลอกเป็นดิจิตอล (ADC) แล้วส่งไปให้ระบบประมวลผลสัญญาณดิจิตอลต่อไป ระบบประมวลผลสัญญาณดิจิตอล(DSP Processor : TMS320C50) ก็จะนำเอาอัลกอริทึมที่มีอยู่มากระทำกับข้อมูลสัญญาณ จากนั้นก็จะส่งข้อมูลสัญญาณที่ถูกประมวลผลแล้ว ไปให้ส่วนของวงจรแปลงสัญญาณดิจิตอลเป็น

อนาลอก (DAC) เพื่อเปลี่ยนสัญญาณข้อมูลกลับไปเป็นสัญญาณอนาลอกที่ต้องการ และเพื่อให้เอาท์พุทเป็นสัญญาณอนาลอกมากขึ้นจะต้องผ่าน วงจรกรองสัญญานให้เรียบ (Low Pass Filter) และจะขจัดสัญญาณความถี่สูงที่ไม่ต้องการออกไป

จากระบบข้างต้นนั้นถูกควบคุมด้วยพีซีโฮสต์ (PC Host) โดยจะควบคุมทางด้านการเปลี่ยนแปลง และคำสั่งให้ทางโปรแกรม กำหนดค่าที่จำเป็นต่อโปรเซสเซอร์



รูปที่ 6.2 บล็อกไดอะแกรมของระบบ DSP ที่ใช้ในโครงงาน

6.2 รายละเอียดทางด้านซอร์ฟแวร์

เนื่องด้วยโครงงานนี้เป็นการออกแบบวงจรกรองชนิดต่างๆ ซึ่งสามารถเปลี่ยนค่าพารามิเตอร์ ของ วงจรกรองได้ เช่นการเปลี่ยนชนิดของวงจรกรอง , การเปลี่ยนความถี่คัทออฟ และอื่นๆ จึงจำเป็นต้องมีโปรแกรมสำหรับการทำงานต่าง ๆ ดังที่กล่าวมา ซึ่งจะแบ่งออกเป็น 2 ส่วนคือ

1. ส่วนโปรแกร์มคำนวณฟิลเตอร์ บน PC
2. ส่วนโปรแกรมคำนวณฟิลเตอร์ บน TMS

6.2.1 โปรแกรมคำนวณฟิลเตอร์ บน PC

การออกแบบ FIR ที่นำมาใช้นี้จะใช้สมการของคอนโวลูชั่น ซัม(convolution sum)

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k) \qquad (6.1)$$

โดยที่ h(n) , n=0,1, . . . ,N-1 จะเป็นการตอบสนองอิมพัลส์ (Impulse response)

ของวงจรกรองที่เราจะออกแบบ ก่อนอื่นเราต้องกำหนดการตอบสนองความถี่ (Frequency response) ของวงจรกรอง ที่เราจะออกแบบก่อน โดย

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h(n) e^{-j\omega n} \qquad (6.2)$$

จากสมการ (6.2) หาแมกนิจูดของการตอบสนองความถี่ $|H(e^{j\omega})|$ และหาเฟส ของ $H(e^{j\omega})$ ได้เป็น

$$H(e^{j\omega}) = \beta - \alpha\omega \qquad ; \omega \geq 0 \qquad (6.3)$$

ในการหา h(n) จะสามารถแบ่งออกได้โดย

$$h(n) = h(N-1-n) \qquad ; 0 \leq n \leq N-1$$

$\beta = 0$, $\alpha = (N-1)/2$ ซึ่งจะเป็นแบบสมมาตร

และ $h(n) = -h(N-1-n)$; $0 \leq n \leq N-1$

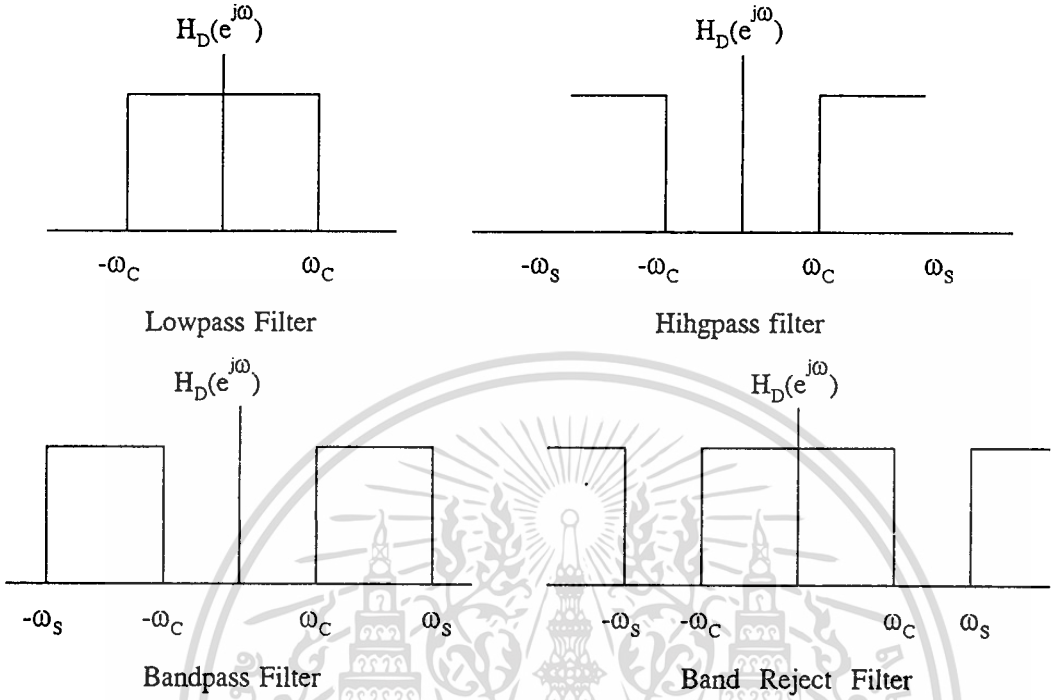
$\beta = \pm\pi/2$ และ $\alpha = (N-1)/2$ จะเป็นแบบไม่สมมาตร

จากการพิจารณาพบว่าการออกแบบหาh(n)โดยใช้แบบสมมาตรจะลดการคำนวณลงถึง50% ต่อมาในการออกแบบของ FIR จำเป็นต้องมีการใช้ วิธีที่เรียกว่า วิธีหน้าต่าง (window method) เพื่อทำให้ตัวกรอง ที่ได้เป็นตัวกรองที่ใกล้เคียงกับตัวกรอง ในอุดมคติมากที่สุด ซึ่งการใช้หน้าต่าง มีหลายวิธี ดังแสดงในตารางที่ (6.1)

<u>ตารางที่ 6.1</u> แสดง หน้าต่างแบบต่าง ๆ

Window Type	Window Function $w(n)$, $0 \leq n \leq N-1$
Rectangular	1
Bartlett	$\frac{2n}{N-1}, \quad 0 \leq n \leq \frac{N-1}{2}$ $2 - \frac{2n}{N-1}, \quad \frac{N-1}{2} \leq n \leq N-1$
Hanning	$0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$
Hamming	$0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$
Blackman	$0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$
Kaiser	$\frac{I_0\left[\beta\sqrt{1-\left(1-\frac{2n}{N-1}\right)^2}\right]}{I_0[\beta]}$ I_0 เป็นค่าจากฟังก์ชั่นเบสเซล (Bessel function) อันดับที่ 0

การออกแบบต้องกำหนดตัวกรองแบบอุดมคติ $H_D(e^{j\omega})$ ก่อน ดังรูปที่ 6.3



รูปที่ 6.3 แสดงคุณสมบัติของตัวกรองอุดมคติแบบต่าง ๆ

โดยกำหนดให้เป็น ยูนิตี้แมกนิจูด (unity magnitude) (=1)

จะได้

$$H_D(e^{j\omega}) = \begin{cases} e^{-j\alpha\omega} & ;|\omega| \le \omega_c \\ 0 & ;\omega_c < |\omega| \le \pi \end{cases} \tag{6.4}$$

โดยที่ ω_C = Corner Frequency เราจะหา h(n) ได้จาก

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})\, e^{j\omega n}\, d\omega \tag{6.5}$$

การหา h(n) จำเป็นต้องกำหนดค่า N เพื่อทำให้ครอบคลุมการคำนวณ

$$h(n) = \begin{cases} h_d(n) & ;0 \le n \le N-1 \\ 0 & ;Otherwise \end{cases} \tag{6.6}$$

และ $$\alpha = \frac{N-1}{2}$$

วิธีนี้จะเรียกว่า การใช้วิธีวินโดว์ ถ้ากำหนดให้ฟังก์ชั่นวินโดว์ เป็น w(n) จะเขียนใหม่ได้เป็น

$$h(n) = h_d(n).w(n) \tag{6.7}$$

โดยที่

$$w(n) = \begin{cases} Some\ symmetric\ function\ over & ;0 \le n \le N-1 \\ 0 & ;Otherwise \end{cases}$$

ในการออกแบบมักจะเลือก w(n) ตามการลดทอนสัญญาณที่ช่วงหยุด (stopband)ว่ามากน้อยเพียงใด

ตัวอย่างที่ 6.1

ออกแบบวงจรกรองโลว์พาสฟิลเตอร์ ที่มีอัตราแซมปลิ้ง(sampling rate) = 8kHz

ความถี่แซมปลิ้ง: 8kHz

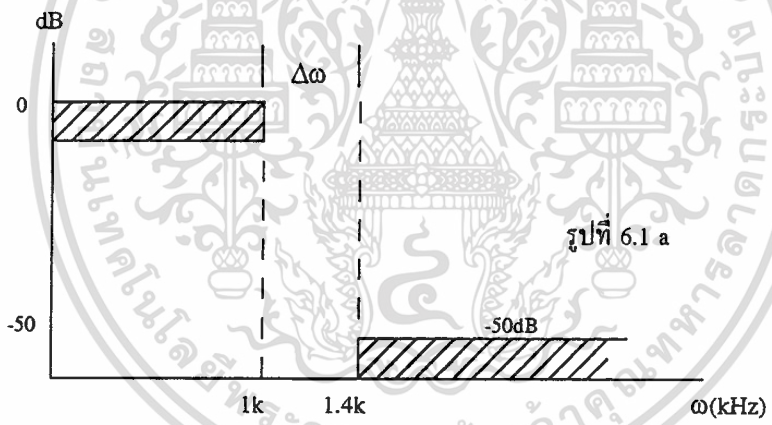
ช่วงผ่านความถี่: 0-1 kHz

ช่วงหยุด : 1.4-4kHz

การลดทอนสัญญาณในช่วงหยุด: 50 dB

จากตารางที่ (6.2) จะเห็นว่าการที่ทำให้การลดทอนสัญญาณในช่วงหยุด มีค่า 50 dB จะใช้ w(n) ได้เมื่อเป็นแฮมมิ่ง หรือ แบล็คแมน ในที่นี้เลือกใช้หน้าต่างแฮมมิ่ง

หาช่วงความถี่เปลี่ยนแปลง จากรูปที่ 6.1 a



รูปที่ 6.1 a

$$\Delta\omega = \frac{2\pi\left(\ 1.4kHz - 1kHz\ \right)}{8kHz} = 0.1\pi$$

หาขนาดของ N ที่ใช้ จากตาราง

$$N = \frac{6.6\pi}{\Delta\omega} = \frac{6.6\pi}{0.1\pi} = 66$$

ซึ่งจะเลือกใช้ N = 67 (สมมาตร) และจะได้

$$\alpha = \frac{N-1}{2} = 33$$

คำนวณหา

$$\omega_c = \frac{(1kHz + 1.4kHz)\pi}{8kHz}$$

$$= 0.3\pi$$

ซึ่งจะได้

$$H_D(e^{j\omega}) = \begin{cases} e^{-j33\omega} & ;|\omega| \le 0.3\omega \\ 0 & ;0.3\pi \le \omega < \pi \end{cases}$$

แก้สมการหาค่า $$h_d(n) = \frac{\sin[0.3(n-33)]}{\pi(n-33)}$$

กำหนดให้ $w(n) = 0.54-0.46\cos(2\pi n/66) \quad 0 \le n \le 66$

จะได้ $$h(n) = \frac{\sin[0.3\pi(n-33)]}{\pi(n-33)}\left[0.54-0.46\cos(2\pi n/60)\right] \quad ; 0 \le n \le 66$$

นำ h(n) มาใช้ในการคำนวณบน PC เพื่อหาค่า h(n) ที่ $0 \le n \le 66$ ดังตาราง

ตารางแสดง ค่าสัมประสิทธิ์ของตัวกรอง ในตัวอย่างที่ 6.1

Filter Length = 67

h(0) = -2.384527E-04 = h(66)	h(17) = 6.570477E-03 = h(49)
h(1) = -7.765305E-04 = h(65)	h((18) = 1.284836E-02 =h(48)
h(2) = -7.336193E-04 = h(64)	h(19) = 8.665973E-03 = h(47)
h(3) = -3.645996E-09 = h(63)	h(20) = -5.224195E-03 = h(46)
h(4) = 1.002993E-03 = h(62)	h(21 = -1.844362E-02 = h(45)
h(5) = 1.417815E-03 = h(61)	h(22) = -1.802628E-02 = h(44)
h(6) = 5.574838E-04 = h(60)	h(23) = -3.472296E-08 = h(43)
h(7) = -1.283883E-03 = h(59)	h(24) = 2.407036E-02 = h(42)
h(8) = -2.636652E-03 = h(58)	h(25) = 3.303238E-02 = h(41)
h(9) = -1.861351E-03 = h(57)	h(26) = 1.266898E-02 = h(40)
h(10) = 1.168261E-03 = h(56)	h(27) = -2.890585E-02 = h(39)
h(11) = 4.265746E-03 = h(55)	h(28) = -6.040657E-02 = h(38)
h(12) = 4.278595E-03 = h(54)	h(29) = -4.523324E-02 = h(37)
h(13) = 1.676480E-08 = h(53)	h(30) = 3.217674E-02 = h(36)
h(14) = -5.849049E-03 = h(52)	h(31) = 1.501070E-01 = h(35)
h(15) = -7.980914E-03 = h(51)	h(32) = 2.569817E-01 = h(34)
h(16) = -2.997856E-03 = h(50)	h(33) = 3.000000E-01 = h(33)

โปรแกรมบนPCจะจัดการเก็บตัวกรอง h(n) เพื่อส่งให้บอร์ดเก็บเป็นหน่วยความจำข้อมูลของTMSอีกที สำหรับโปรแกรมการคำนวณหา ค่าสัมประสิทธิ์ ของ h(n) (โลว์พาส)แสดงไว้ในภาคผนวก ข

เนื่องจากจะต้องมีการแสดงผลของตัวกรอง ที่ออกแบบมา จึงต้องทำการพล็อตกราฟของผลตอบสนองความถี่โดยพล็อต จากการหาขนาดของ $H_D(e^{j\omega})$

โปรแกรมที่ใช้ในการพล็อตผลตอบสนองความถี่ แสดงไว้ในภาคผนวก ข

เพื่อให้การคำนวณและออกแบบตัวกรอง ให้ได้ผลตอบสนองที่กำหนดไว้แน่นอนขึ้น เราเลือกใช้วิธี วินโดว์ (window methods) โดยเลือกใช้ ไคเซอร์วินโดว์ จากข้อจำกัดที่ว่า

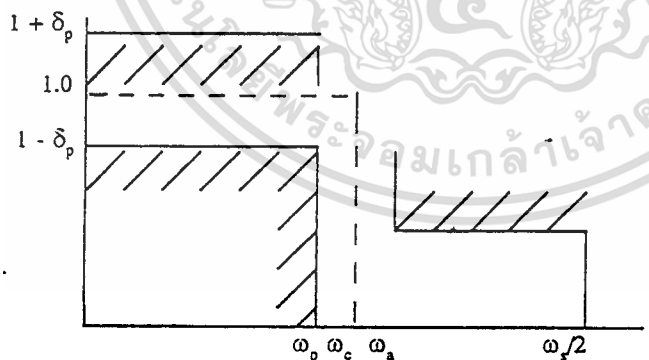
- ความกว้างของโหลบหลัก (main lobe) แปรตามค่า 1/N หมายถึงว่าถ้าต้องการออกแบบตัวกรองให้มีแถบเปลี่ยนความถี่ค่าแคบ ต้องใช้ N ค่ามากๆ

- ค่าขนาดของโหลบข้าง (side lobe) ทำให้เกิดลูกคลื่นบนผลตอบสนองความถี่ไม่แปรตามค่า N แต่ขึ้นอยู่กับชนิดของวินโดว์ที่ใช้

ดังนั้นการออกแบบตัวกรองให้มีค่าลดทอนในแถบหยุดน้อยที่สุด (minimum stopband attenuation) ตามต้องการอาจทำได้โดยต้องเลือกชนิดของวินโดว์ให้เหมาะสม

ไคเซอร์ (J.F. Kaiser) ใช้ฟังก์ชันทรงคล้ายทรงกลมแบนข้าง (prolate spheroidal function) มาทำการประเมินค่า ผลที่ได้คือ ได้วินโดว์ที่มีคุณสมบัติแลกเปลี่ยน (trade off) ของโหลบหลักกับค่าขนาดของโหลบข้าง โดยมีพารามิเตอร์ เป็นตัวควบคุมเพื่อปรับค่าขนาดโหลบข้างเมื่อเทียบกับค่าขนาดของโหลบหลักได้

การออกแบบโดยทั่วไปพารามิเตอร์ δ_p และ δ_s เป็นตัวกำหนดคุณสมบัติของค่าขนาดลูกคลื่นในแถบผ่าน (pass band) และแถบหยุด (stop band) ตามลำดับ ถ้ากำหนดให้ A_p เป็นค่าขนาดของลูกคลื่นในแถบผ่าน A_s เป็นค่าลดทอนในแถบหยุดน้อยที่สุด และ B_t เป็นค่าความกว้างของแถบเปลี่ยนสถานะจะเขียนได้ดังนี้



$$A_p = 20 \log\left[\frac{(1+\delta_p)}{1-\delta_p}\right]$$

$$A_a = -20 \log\delta_s$$

$$B_t = \omega_a - \omega_p$$

ถ้าสมมติ ให้ A_p' และ A_s' เป็นค่าขนาดของลูกคลื่นในแถบผ่าน และ ให้ค่าลดทอนในแถบหยุดน้อยที่สุดขั้นตอนการออกแบบมีดังนี้

1.) หาผลตอบสนองอิมพัลส์ h(n) โดยประยุกต์ใช้อนุกรมฟูริเยอร์ กับผลตอบสนองอุดมคติคือ

$$h_d(\omega) = 1 \quad ;|\omega| < \omega_c$$
$$= 0 \quad ;\omega_c < |\omega| < \frac{\omega_s}{2} \qquad \text{โดย} \qquad \omega_c = \frac{\omega_p + \omega_a}{2}$$

2.) เลือกพารามิเตอร์จาก $A_p < A_p'$ และ $A_a > A_a'$ โดยเลือก

$$\delta = \min(\delta_p, \delta_s)$$

3.) คำนวณหา A_a ตามค่า $\delta = \delta_p = \delta_s$

4.) เลือกพารามิเตอร์ α

$$\alpha = 0 \qquad \text{เมื่อ } A_a < 21$$
$$= 0.5842(A_a\text{-}21)^{0.4} + 0.07886(A_a\text{-}21) \qquad \text{เมื่อ } 21 < A_a < 50$$
$$= 0.1102(Aa\text{-}8.7) \qquad \text{เมื่อ } A_a > 50$$

5.) เลือกพารามิเตอร์ D ให้สอดคล้องกับ

$$D = 0.9222 \qquad \text{เมื่อ } A_a < 21$$
$$= \frac{A_a - 7.95}{14.36} \qquad \text{เมื่อ } A_a > 21$$

6.) คำนวณค่า N ที่เป็นเลขคี่ที่ต่ำสุด

$$N \geq \frac{\omega_s D}{B_t} + 1$$

7.) คำนวณ $h(n) = h_d(n)\, w_k(n)$ สำหรับ $n \leq \frac{(N-1)}{2}$

8.) ฟังก์ชันถ่ายโอนของตัวกรองเขียนเป็น $H(\omega) = h(0) + 2\sum_{n=1}^{(N-1)/2} h(n)\cos 2nft$

6.2.2 <u>การคำนวณฟิลเตอร์บน TMS</u>

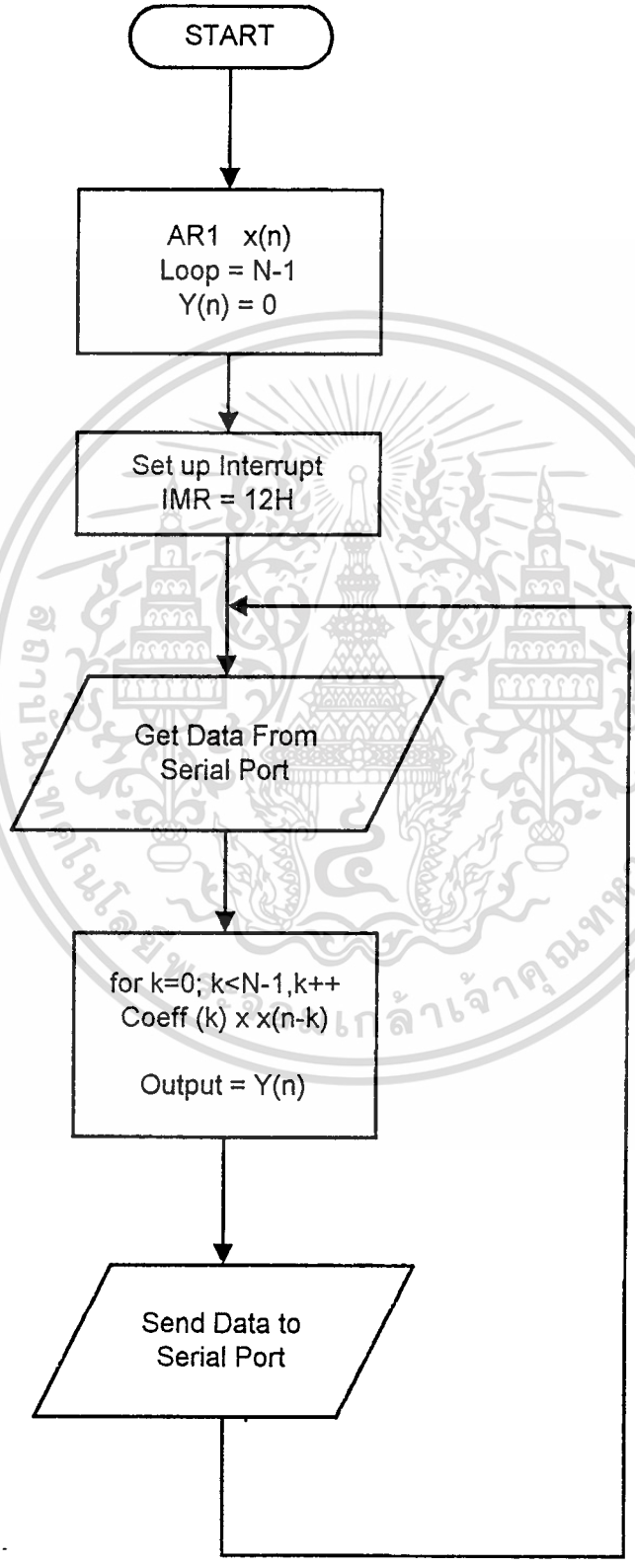
เนื่องจากโปรแกรมบน TMS ต้องเป็นเอสเซมบลี้ ของตัว TMS 320C50 ซึ่งโปรแกรมจะกำหนดไว้เลยโดยเป็นการคำนวณหาคอนโวลูชั่น จากสมการ

$$y(i) = \sum_{k=0}^{N-1} b_x x(i-k) \qquad \text{หรือ}$$

$$y(i) = b_0x(i) + b_1x(i\text{-}1) + b_2x(i\text{-}2) + b_3x(i\text{-}3) + \ldots + b_{N-1}x(i\text{-}(N\text{-}1))$$

โดยที่ $b_0, b_1, \ldots, b_{N-1}$ เป็นสัมประสิทธิ์ของวงจรกรอง ซึ่งคำนวณมาจาก PC ในการเปลี่ยนแปลงรูปแบบของตัวกรองก็เปลี่ยนแต่ค่าสัมปะสิทธิ์เท่านั้น ซึ่งจะทำการเปลี่ยนได้บน PC เป็นการง่ายต่อการใช้งาน ส่วนสัญญาณ อินพุท x(i-k) จะรับมาจาก TLC32040 ซึ่งเป็นวงจรการเชื่อมต่อแบบอนาลอก โดยกำหนดความถี่แซมปิ้ง ได้ ($\leq$19.2 kHz)

สำหรับโปรแกรมที่ใช้คำนวณแสดงเป็นโฟลว์ชาร์ต ได้ดัง รูปที่ 6.4



รูปที่ 6.4 แสดงโฟลว์ชาร์ตของโปรแกรมที่ใช้คำนวณตัวกรอง บน TMS

ในส่วนของโปรแกรมบน TMS320C50 สามารถแบ่งได้ดังนี้

1) โปรแกรมการกำหนดค่าเริ่มต้น (Initialization Processor) TMS320C50

2) โปรแกรมการกำหนดค่าเริ่มต้น (Initialization) TLC32040

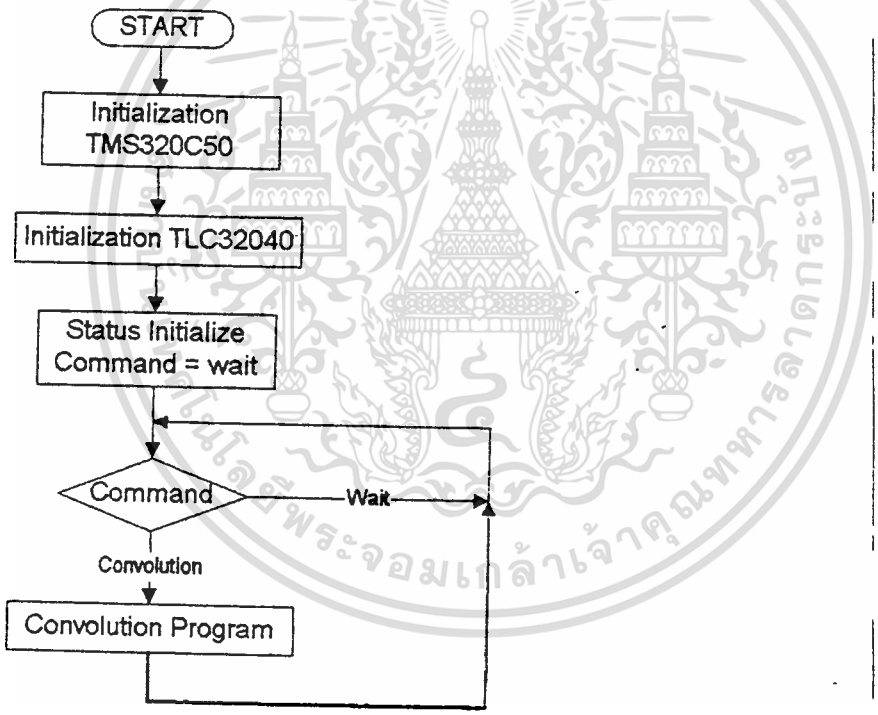
3) โปรแกรมคอนโวลูชั่น

4) โปรแกรม รับ/ส่ง ข้อมูล RS232

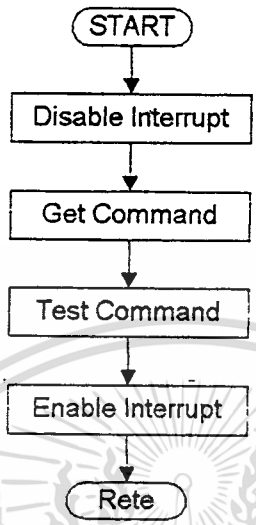
5) โปรแกรมควบคุมทดสอบคำสั่ง (Test Command)

จากโปรแกรมทั้งหมด สามารถนำมาเขียนเป็นโปรแกรมหลักและอธิบายได้ดังนี้

โฟลว์ชาร์ต(flow chart) โปรแกรมหลัก

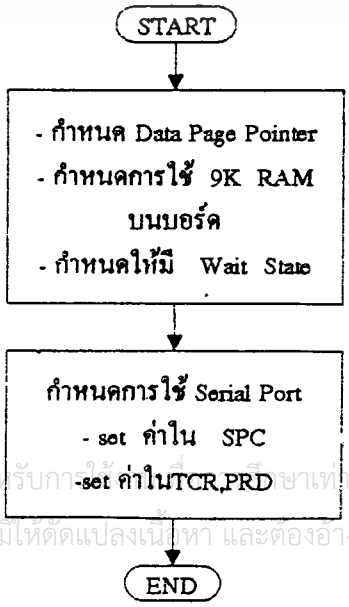


โปรแกรมบริการอินเตอร์รัพต์



เนื่องจากใช้อินเทอร์รัพต์ ในการติดต่อกับพอร์ตอนุกรม ของคอมพิวเตอร์ผ่านทาง RS232 ซึ่งเราจะใช้ อินเทอร์รัพ หมายเลข1 ($\overline{INT1}$) จะเป็นตัวคอยรับข้อมูลมา ทั้งโปรแกรมหลักและโปรแกรมบริการ อินเทอร์รัพต์ จะอธิบายดังนี้คือ เมื่อเริ่มโปรแกรม จะทำการกำหนดค่าเริ่มต้นข้อมูลบนTMS320C50 ที่จำเป็นในการใช้ แล้วจะเซตค่าที่กำหนดการใช้งานให้กับ TLC32040 ซึ่งเป็น IC ที่ทำหน้าที่แปลงสัญญาณอนาลอก เป็น ดิจิตอล (A/D) และ แปลงสัญญาณ ดิจิตอล เป็น อนาลอก (D/A) และกำหนดอัตราแซมปิ้ง เนื่องจากต่อมาเราต้องการทำคอนโวลูชั่นเพื่อทำตัวกรองแก่สัญญาณที่เข้า และส่งกลับไปอีกที ข้อมูลหรือสัมประสิทธิ์ ต่างๆ นั้นจะรับมาจาก PC ดังนั้นจึงจำเป็นต้องรอเพื่อทดสอบคำสั่งว่าครบหรือพร้อมหรือยัง ในส่วนของ โปรแกรมบริการอินเทอร์รัพต์ จะเป็นการรับข้อมูลจาก PC มาเก็บใช้เป็นค่าสัมประสิทธิ์ ซึ่งจะอธิบายละเอียดตามหัวข้อต่อไปนี้

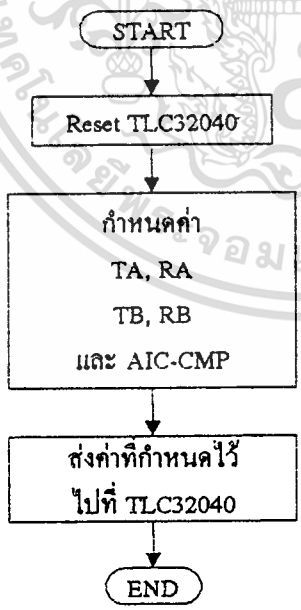
1) <u>โปรแกรมการกำหนดค่าเริ่มต้น(Initialization)TMS320C50</u>



ซึ่งเราสามารถเขียนโปรแกรมหลักได้ดังนี้

```
;-------------------------------------------------------------------------;
; TMS32C05X INITIALIZATION                                                ;
; This routine initializes the C5x registers, internal RAM and            ;
; external RAM from xxxx to FFFF                                          ;
;-------------------------------------------------------------------------;
                .ps     0a00h
                .entry
start           ldp     #0              ; Set data page pointer
                splk    #830h,PMST      ; 9K on-chip RAM as Data, No ROM
                lacl    #0              ; Set Wait State Control Regester
                samm    CWSR            ; for 0 waits in pgm & data memory
                samm    PDWSR           ;
```

2) โปรแกรมการกำหนดค่าเริ่มต้น(Initialization)TLC32040



ซึ่งเราสามารถเขียนโปรแกรมหลักได้ดังนี้

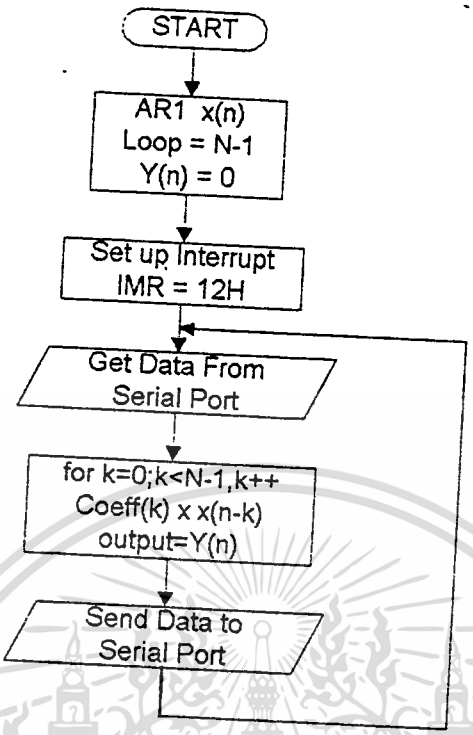
```
;-------------------------------------------------------------------------;
;       Set TLC32040                                                      ;
;-------------------------------------------------------------------------;
```

```
AIC_SET:    lacl  #020h
            samm  IMR           ;XMIT interrupt
            lacc  #AIC_CMD,2
            add   #03h
            call  AIC_2nd
            ;----------------------
            lacc  #TB,9
            add   #RB,2
            add   #02h
            call  AIC_2nd
            ;----------------------
            lacc  #TA,9
            add   #RA,2
            call  AIC_2nd
            ;----------------------
            ret

AIC_2nd:    sach  DXR
            clrc  INTM
            idle
            add   #6,15
            sach  DXR
            idle              ;ACCU_hi requests 2nd XMIT
            samm  DXR
            idle              ;ACCU_lo sets up registers
            lacl  #0
            samm  DXR         ;make sure the word got sent
            idle
            setc  INTM
            ret
;-------------------------------------------------------------------;
```

3) <u>โปรแกรมการคอนโวลูชั่น</u>

สำหรับโปรแกรมที่ใช้คำนวณแสดงเป็นโฟลว์ชาร์ต ได้ดังนี้



4) <u>โปรแกรมรับส่งผ่าน RS232</u>

ในการส่งข้อมูลทาง PC จะกำหนดรูปแบบดังนี้

- ใช้ พอร์ต 1 (COM2)
- อัตรา Baud rate = 9600 baud
- ไม่มีพ าริตี้บิต
- 2 บิตสิ้นสุด
- 8 บิต ข้อมูล

<u>รูปแบบของข้อมูล</u>

START BIT	D0	D1	D2	D3	D4	D5	D6	D7	STOP	STOP

จะใช้บิตเริ่มต้นเป็นสถานะต่ำ เป็นการเปิดอินเทอร์รัพ เบอร์ INT1 ของ TMS320C50 การกำหนด baud rate จะคำนวณทาง TMS โดยการเก็บค่าลงใน AR5 Baud rate = 9600 bit/sec ซึ่ง 1 bit = 104.1666 μsec จะได้ 8 bit เป็น 833.333 μsec เราจะใช้คำสั่ง NOP เพื่อกำหนดค่าคำนวณ AR5 คำสั่ง NOP 1 คำสั่งใช้เวลาทำงาน 50 nsec เพราะฉะนั้น 833.333 μsec ใช้ทำงานคำสั่ง NOP เป็น 16,666 คำสั่ง AR5 จะเก็บค่า 1 bit ดังนั้นจะได้ AR5 = 823H

ในการติดต่อ RS232 TMS320C50 จะใช้ขา BIO เป็นตัวรับ และขา XF เป็นตัวส่ง ในการใช้งานถ้าส่ง "1" จะทำการ setc แต่ถ้าส่ง "0" จะทำการ clrc บิตนั้นๆ ซึ่งขา BIO และ XF จะใช้ควบคุมระดับบิตได้

ในการอ่านข้อมูลจาก PC จะเขียนได้ดังนี้

```
; READS: read serial mode
;-------------------------------------------------------------------;
reads    lar   ar5,#7
         lacl  #0
reads1   Bcndd STOK,bio    ;wait for start bit
         lar   AR5,#7
         lacl  #0
         B     reads1      ;
STOK     rpt   BITLEN2     ;BITLEN is scaled and
         nop
         mar   *,AR5       ;number of bits - 1
WTBIT    sfr
         rpt   BITLEN      ;decremented by 8/3 for
         nop               ;BITLEN/2 wait
         bcnd  ZEROBT,bio
         add   #80h
ZEROBT   BANZ  WTBIT,*-    ;last bit ?
         RET               ;ACC = read value
```

ส่วนโปรแกรมส่งข้อมูลเขียนได้ดังนี้

```
;-------------------------------------------------------------
; xmtbyte to PC
;-------------------------------------------------------------
xmtbyte  clrc c            ; startbit=0
         lar  ar5,#8       ; counter: 1 startbit+ 8 databits (+ 2 stopbits)
nextbit1 bcnd snd0,nc      ; if c=1 send 1 else send 0
snd1     setc xf           ; send one
         b    snd
snd0     clrc xf
snd      rpt  BITLEN       ; send one bit
         mar  *,ar5
         ror               ; lsb(accu) -> carrybit
         banz nextbit1,*-  ; repeat for entire word (10 bits);
```

```
setc xf
rpt  BITLEN
nop
rpt  BITLEN
nop
ret
```

ทั้งโปรแกรม reads และ xmtbyte จะส่งเพียง 8 bit ในการใช้งานจำเป็นต้องส่งถึงสองครั้ง เพื่อให้ได้ 16 bit

5) โปรแกรมควบคุมทดสอบคำสั่ง(Test Command)

เนื่องจากจะติดต่อกับ TMS320C50 เพื่อทำงานในขั้นตอนต่างๆ โดยจะกำหนดดังนี้

- Command 01h

เป็นการส่งค่า TA, TB, RA, RB โดย PC จะส่งค่าดังกล่าวให้แก่ TMS320C50 เมื่อมีการเปลี่ยน sampling rate

- Command 02h

เป็นการส่งค่าสัมประสิทธิ์ ให้กับ TMS โดยจะส่งค่า N และ Coeff ที่คำนวณจาก PC โดยจะเป็นการ down load ส่งไปที่ TMS ตามแอสเดรส ที่กำหนด

- Command 05h

เพื่อเป็นการเริ่มทำการคอนโวลูชั่น เมื่อรับค่า N, Coeff และ TA, TB, RA, RB เรียบร้อยแล้ว

- Command 07h

เมื่อการรับส่ง TA, TB, RA, RB และ N, Coeff ยังไม่สมบูรณ์จะต้องรอจนกว่าเสร็จสิ้นการบอกให้รอ(wait)

- Command 08h หรืออื่นๆ นอกจากที่กำหนดมา

เมื่อการทดสอบคำสั่ง(Test Command) ไม่ตรงกับค่าก่อนหน้านี้แล้ว แสดงว่า TMS320C50 จะผิดพลาด คือ โปรแกรมที่ได้กำหนดนั้นได้ออกจากโปรแกรม จำเป็นต้องมีการรีเซตใหม่ โดยการกระโดดไปที่แอสเดรสเริ่มต้น 0800h

ในการทดสอบคำสั่ง จะตรวจสอบเมื่อมีการรับส่งครั้งแรกโดยรูปแบบ(Format)ของการรับส่ง ผลของโปรแกรม
ต่างๆ จะขึ้นต้นด้วย Command แล้วตามด้วยข้อมูลที่ Command นั้นระบุ

6.3 การออกแบบ FIR Filter ที่ใช้ในโครงงาน

ในการออกแบบจะกำหนดฟังก์ชั่นวินโดว์ (window function) เพื่อให้มีประสิทธิภาพในการกำจัดช่วงผ่านความถี่ (Bandpass) ให้ต่ำลง มีวินโดว์ที่ใช้มีดังนี้

1. หน้าต่างสี่เหลี่ยม (Rectangular window)
2. หน้าต่างบาร์ตเลท(Bartlett window)
3. หน้าต่างแฮมมิ่ง(Hamming window)
4. หน้าต่างแฮนนิ่ง(Hanning window)
5. หน้าต่างแบล็คแมน(Blackman window)
6. หน้าต่างไคเซอร์(Kaiser window)

ซึ่งแต่ละฟังก์ชั่นวินโดว์ (window function) จะมีขนาดสำหรับการลดทอนสัญญาณช่วงหยุด(Bandstop) ได้ต่างกัน

ในการออกแบบจะเลือกใช้สมการของคอนโวลูชั่น(Convolution)

$$y(n) = \sum_{k=0}^{N-1} h(n)x(n-k)$$

เมื่อ h(k) เป็นผลตอบสนองอิมพัลส์ (Impulse Response) ที่ได้จากการกำหนด $H(e^{j\omega})$ ที่ออกแบบมา จะเก็บไว้ที่หน่วยความจำข้อมูล(data memory)ของ TMS320C50 ซึ่งมี N จำนวน

x(n) เป็นอินพุทที่รับมา

y(n) เป็นเอาท์พุทที่ได้

เนื่องจากขั้นตอนการหา h(k) จะต้องหาจาก การกำหนดผลตอบสนองความถี่ ของฟิลเตอร์จากสูตร

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n}$$

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} h_d(e^{j\omega})e^{j\omega n}dw$$

และนำมาคูณกับฟังก์ชั่นวินโดว์ (w(n))

$$h(n) = h_d(n).w(n)$$

ซึ่งจะได้ h(n) ทั้งหมด N จำนวน

โปรแกรมสำหรับการคำนวณ

1.) โปรแกรมบน PC คำนวณหา h(n)

ดังที่ได้กล่าวมาการคำนวณหา h(n) จะต้องกำหนดผลตอบสนองความถี่ก่อน ($H(e^{j\omega})$) ซึ่งจะกำหนดจากโปรแกรมด้วย

-f_c เป็นความถี่คัทออฟของฟิลเตอร์

-f_s เป็นความถี่แซมปลิ้ง

-วินโดว์ กำหนดฟังก์ชั่นวินโดว์ที่ใช้

การคำนวณหา h(n)

1. ฟิลเตอร์ FIR ความถี่ต่ำ(lowpass)

$$h_d(n) = \left(\frac{2f_c}{F}\right)\frac{\sin 2\pi nf_c/F}{2\pi nf_c/F} \qquad ; \quad n>0$$

$$= \frac{2f_c}{F} \qquad ; \quad n=0$$

ที่

$$f_c = \frac{1}{2}\left(f_p + f_s\right) \qquad \Delta F = f_s - f_p$$

2. ฟิลเตอร์ FIR ความถี่สูง(highpass)

$$h_d(n) = -\left(\frac{2f_c}{F}\right)\frac{\sin 2\pi nf_c/F}{2\pi nf_c/F} \qquad ; \quad n>0$$

$$= 1-\frac{2f_c}{F} \qquad ; \quad n=0$$

3.ฟิลเตอร์ FIR ช่วงความถี่ผ่าน (pass band)

$$h_d(n) = \frac{1}{n\pi}\left[\sin\left(\frac{2\pi nf_{c2}}{F}\right) - \sin\left(\frac{2\pi nf_{c1}}{F}\right)\right] \qquad ; \quad n>0$$

$$= \frac{2}{F}\left(f_{c2} - f_{c1}\right) \qquad ; \quad n=0$$

4. ฟิลเตอร์ FIR ช่วงหยุด(stopband)

$$h_d(n) = \frac{1}{n\pi}\left[\sin\left(\frac{2\pi nf_{c1}}{F}\right) - \sin\left(\frac{2\pi nf_{c2}}{F}\right)\right] \qquad ; \quad n>0$$

$$= \frac{2}{F}\left(f_{c1} - f_{c2}\right) + 1 \qquad ; \quad n=0$$

2.)<u>โปรแกรมบน TMS320C50</u>

ซึ่งโปรแกรมบน TMS320C50 เป็นภาษาเอสเซมบลี้ ของ CPU นี้ จะเป็นการทำการคอนโวลูชั่น สัญญาณระหว่าง h(n) กับ x(n) ดังสูตร

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k)$$

จากสมการ

$$y(n) = h(0)\, x(n) + h(1)\, x(n-1) + h(2)\, x(n-2) + \ldots + h(N-1)\, x(n-(N-1))$$

1. จะต้องเก็บ h(0) ถึง h(N-1) ที่หน่วยความจำข้อมูล (Data memory) ใช้กำหนดเป็นที่ Coeff

2. อินพุทข้อมูล(Data input) x(n) เมื่อเริ่มโปรแกรมจะรับมา 1 ค่า

$$y(0) = h(0)\, x(0) + h(1)\, x(-1) + \ldots + h(N-1)\, x(-(N-1))$$

ซึ่ง

x(0) เป็นค่าที่รับมา

x(-1) เป็นค่าที่รับก่อน x(0)

x(-2) เป็นค่าที่รับมาก่อน x(-1)

.

.

.

x(-(N-1)) เป็นค่าที่รับมาก่อน x(-(N-2))

3. เมื่อรับมาเรื่อยๆ x(0) จะต้องเลื่อนมาที่ x(-1) และ x(-1) จะต้องเลื่อนมาที่ x(-2) ไปจนถึง x(-(N-2)) เลื่อนมาที่ x(-(N-1)) และต่อ x(-(N-1)) สุดท้ายปล่อยออกไป

4. แต่ละอินพุท เข้ามาจะมีการคูณกันถึง N ครั้ง และบวกกัน N-1 ครั้ง เพราะฉะนั้นจะต้องใช้ ความเร็วมาก ในการเขียนโปรแกรม TMS320C50 มีคำสั่งสำหรับ DSP โดยเฉพาะ

บทที่ 7

ผลการทดลองและสรุปผลการทดลอง

7.1 ผลการทดลอง

7.1.1 โปรแกรม FIR Filter

การใช้บอร์ด DSK

เนื่องจากอุปกรณ์สำคัญที่ทำงานร่วมกัน บนบอร์ดที่ใช้ทดลอง มี IC TMS320C50 และ TLC32040 โดยที่ TMS320C50 เป็นตัวประมวลผลซึ่งได้เขียนโปรแกรมต่าง ๆ เก็บไว้ใน RAM ส่วนของผู้ใช้ โดยรับข้อมูลเข้ามาทาง TLC32040 และนำมาประมวลผล แล้วจะส่งออกไปให้ TLC32040 อีกทีหนึ่ง ซึ่งสามารถแบ่งการทดลองสำหรับใช้งานดังต่อไปนี้

1) การควบคุม TMS320C50

2) การควบคุม TLC32040

ในการควบคุม TMS320C50 ได้ผลการทดลองดังนี้

การกำหนดค่าเริ่มต้นของ TMS320C50

PMST (Precessor Mode Status register) มีค่าเป็น 830H

กำหนดได้ดังนี้

1) อินเทอร์รัพเวกเตอร์พอยเตอร์ (Interrupt Vector Pointer)กำหนดให้อยู่ที่เพจ1

2) กำหนดให้ใช้ RAM 9K บนชิพ

3) ไม่มีสภาวะคอย (wait state)

ซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
start   ldp    #0
        splk   #830H,PMST
        lacl   #0
        samm   CWSR
        samm   PDWSR
```

การกำหนดพอร์ตอนุกรม(serial port)

1) การรับส่งมีเฟรมซิงค์ (Frame Sync) ทุกๆการอินเทอร์รัพ ซึ่งต้องกำหนดตาม TLC32040

2) กำหนดการส่งแบบ 16 บิต

3) ใช้สัญญาณนาฬิกา (Clock) จากภายนอกซึ่งผลิตจาก TMS320C50

4) มีการอินเทอร์รัพ ทุกๆเฟรมซิงค์(Frame Sync) ตามความถี่แซมปลิ้ง

การเขียนโปรแกรมควบคุม TLC32040

กำหนดค่าต่าง ๆได้ดังนี้

1) แปลงความถี่ (Conversion Frequency) 31.25kHz โดยกำหนดจากรีจิสเตอร์

TA = 40

TB = 4

RA = 40

RB = 4

2). กำหนดอัตราขยาย

Gain = 4 โดยการกำหนด บิตที่ 6 และ 7 ของรีจิสเตอร์(Control Register) d6 = 1,d7 = 1

3) กำหนดใช้แบนด์พาสฟิลเตอร์ (bandpass filter) ทางอินพุท โดยไม่ให้อินพุท ที่เข้ามามีความถี่เกินกว่า ความถี่แซมปลิ้ง

ผลของการใช้โปรแกรมร่วมกับบอร์ด DSK

1) อินพุทที่รับเข้ามาจาก TLC32040 จะมีค่าลดลงเป็นครึ่งหนึ่งของสัญญาณอินพุทจริง มีผลการทดลองดังนี้

ตารางที่ 7-1 ผลการทดลอง

สัญญาณอินพุท	ค่าที่ TLC32040 ส่งให้ TMS320C50
0V	0000H,FFFCH
1V	2AAAH
2V	5554H
3V	7FFFH
-1V	D552H
-2V	AAA8H
-3V	8000H

ในการนำไปใช้งาน เพื่อที่จะส่งค่าออกไป จะต้องเลื่อน ไปอีก 2 บิต เพื่อให้ได้ค่าเท่ากับอินพุทที่เข้ามา

2) วัดสัญญาณต่าง ๆ ที่ขาสัญญาณระหว่าง TLC32040 และ TMS320C50 ได้ดังนี้

CLKIN 41.472 MHz (ความถี่ crystal)

TOUT 10.368 MHz (ความถี่ clock สำหรับ TLC32040

ที่รับมาจาก TMS320C50)

FSK/FSR ขึ้นอยู่กับความถี่แซมปลิ้ง จากการทดลองใช้ f_s =31.25 KHz ได้ FSX/FSR ประมาณ 16-18 KHz และ duty cycle ประมาณ 60%

DR/DX สัญญาณรับข้อมูลจะเป็นข้อมูล 16 บิต

3) ผลของการทำฟิลเตอร์โดยเขียนโปรแกรมเป็น FIR

โปรแกรมที่ใช้สามารถกำหนดค่า จำนวนของจุดฟิลเตอร์ ได้ (N) ผลของการรันโปรแกรมโดยใช้โลว์พาสฟิลเตอร์(Lowpass filter) ได้สัมประสิทธิ์ดังนี้ โดยที่ความถี่แซมปลิ้ง (Sampling) 31.25 kHz

lowpass filter
Ap : 0.900 dB
As : 40.000 dB
bartlett window
fs2 : 1000.000 Hz
fp2 : 900.000 Hz
F : 31250.00 Hz

---

frequency response

Frequency : 100.000000 Hz output : 0.355150 V: 0.000000 dB
Frequency : 105.565455 Hz output : 0.354820 V: -0.008075 dB
Frequency : 111.440654 Hz output : 0.354850 V: -0.007341 dB
Frequency : 117.642833 Hz output : 0.355260 V: 0.000000 dB
Frequency : 124.190193 Hz output : 0.355990 V: 0.000000 dB
Frequency : 131.101942 Hz output : 0.357070 V: 0.000000 dB
Frequency : 138.398362 Hz output : 0.358490 V: 0.000000 dB
Frequency : 146.100861 Hz output : 0.360230 V: 0.000000 dB
Frequency : 154.232039 Hz output : 0.362300 V: 0.000000 dB
Frequency : 162.815755 Hz output : 0.364470 V: 0.000000 dB
Frequency : 171.877193 Hz output : 0.366720 V: 0.000000 dB
Frequency : 181.442941 Hz output : 0.368790 V: 0.000000 dB

Frequency : 191.541067 Hz output : 0.370490 V: 0.000000 dB

Frequency : 202.201199 Hz output : 0.371550 V: 0.000000 dB

Frequency : 213.454617 Hz output : 0.371730 V: 0.000000 dB

Frequency : 225.334338 Hz output : 0.370810 V: 0.000000 dB

Frequency : 237.875220 Hz output : 0.368810 V: 0.000000 dB

Frequency : 251.114059 Hz output : 0.365810 V: 0.000000 dB

Frequency : 265.089700 Hz output : 0.362140 V: 0.000000 dB

Frequency : 279.843149 Hz output : 0.358440 V: 0.000000 dB

Frequency : 295.417694 Hz output : 0.355500 V: 0.000000 dB

Frequency : 311.859034 Hz output : 0.354110 V: -0.025473 dB

Frequency : 329.215409 Hz output : 0.355130 V: -0.000489 dB

Frequency : 347.537745 Hz output : 0.358660 V: 0.000000 dB

Frequency : 366.879803 Hz output : 0.364030 V: 0.000000 dB

Frequency : 387.298335 Hz output : 0.369510 V: 0.000000 dB

Frequency : 408.853250 Hz output : 0.373100 V: 0.000000 dB

Frequency : 665.677505 Hz output : 0.367440 V: 0.000000 dB

Frequency : 702.725489 Hz output : 0.349290 V: -0.144514 dB

Frequency : 741.835362 Hz output : 0.342800 V: -0.307421 dB

Frequency : 783.121878 Hz output : 0.363700 V: 0.000000 dB

Frequency : 826.706176 Hz output : 0.391880 V: 0.000000 dB

Frequency : 872.716139 Hz output : 0.368260 V: 0.000000 dB

Frequency : 921.286765 Hz output : 0.250380 V: -3.036245 dB

Frequency : 972.560569 Hz output : 0.087242 V: -12.193724 dB

Frequency : 1026.687992 Hz output : 0.036421 V: -19.781199 dB

Frequency : 1083.827854 Hz output : 0.034456 V: -20.262940 dB

Frequency : 1144.147809 Hz output : 0.034076 V: -20.359265 dB

Frequency : 1207.824844 Hz output : 0.028514 V: -21.907074 dB

Frequency : 1275.045796 Hz output : 0.029987 V: -21.469576 dB

Frequency : 1346.007899 Hz output : 0.029977 V: -21.472474 dB

Frequency : 1420.919368 Hz output : 0.028447 V: -21.927507 dB

Frequency : 1500.000000 Hz output : 0.028805 V: -21.818879 dB

---

highpass filter

Ap : 0.900 dB

As : 40.000 dB

bartlett window

fs2 : 700.000 Hz

fp2 : 900.000 Hz

F : 31250.00 Hz

---

frequency response

Frequency : 100.000000 Hz output : 0.017843 V: -26.663372 dB

Frequency : 105.565455 Hz output : 0.017752 V: -26.707785 dB

Frequency : 111.440654 Hz output : 0.017617 V: -26.774090 dB

Frequency : 117.642833 Hz output : 0.017454 V: -26.854830 dB

Frequency : 124.190193 Hz output : 0.017257 V: -26.953424 dB

Frequency : 131.101942 Hz output : 0.017046 V: -27.060280 dB

Frequency : 138.398362 Hz output : 0.016903 V: -27.133453 dB

Frequency : 146.100861 Hz output : 0.016811 V: -27.180859 dB

Frequency : 154.232039 Hz output : 0.016841 V: -27.165373 dB

Frequency : 162.815755 Hz output : 0.017013 V: -27.077112 dB

Frequency : 171.877193 Hz output : 0.017272 V: -26.945877 dB

Frequency : 181.442941 Hz output : 0.017530 V: -26.817091 dB

Frequency : 191.541067 Hz output : 0.017672 V: -26.747015 dB

Frequency : 202.201199 Hz output : 0.017600 V: -26.782476 dB

Frequency : 213.454617 Hz output : 0.017341 V: -26.911247 dB

Frequency : 225.334338 Hz output : 0.016973 V: -27.097557 dB

Frequency : 237.875220 Hz output : 0.016808 V: -27.182409 dB

Frequency : 251.114059 Hz output : 0.017070 V: -27.048059 dB

Frequency : 265.089700 Hz output : 0.017819 V: -26.675062 dB

Frequency : 279.843149 Hz output : 0.018614 V: -26.295937 dB

Frequency : 295.417694 Hz output : 0.018880 V: -26.172689 dB

Frequency : 311.859034 Hz output : 0.018275 V: -26.455582 dB

Frequency : 329.215409 Hz output : 0:017279 V: -26.942358 dB

Frequency : 347.537745 Hz output : 0.017073 V: -27.046534 dB

Frequency : 366.879803 Hz output : 0.018396 V: -26.398262 dB

Frequency : 387.298335 Hz output : 0.019743 V: -25.784466 dB

Frequency : 408.853250 Hz output : 0.019130 V: -26.058432 dB

Frequency : 431.607795 Hz output : 0.017361 V: -26.901236 dB

Frequency : 455.628734 Hz output : 0.019075 V: -26.083439 dB

Frequency : 480.986548 Hz output : 0.023484 V: -24.277288 dB

Frequency : 507.755639 Hz output : 0.023179 V: -24.390837 dB

Frequency : 536.014552 Hz output : 0.018649 V: -26.279619 dB

Frequency : 565.846203 Hz output : 0.027119 V: -23.027256 dB

Frequency : 597.338120 Hz output : 0.034225 V: -21.005861 dB

Frequency : 630.582706 Hz output : 0.019028 V: -26.104866 dB

Frequency : 665.677505 Hz output : 0.084631 V: -13.142140 dB

Frequency : 702.725489 Hz output : 0.207070 V: -5.370387 dB

Frequency : 741.835362 Hz output : 0.324660 V: -1.464154 dB

Frequency : 783.121878 Hz output : 0.384270 V: 0.000000 dB

Frequency : 826.706176 Hz output : 0.375540 V: -0.199606 dB

Frequency : 872.716139 Hz output : 0.344110 V: -0.958784 dB

Frequency : 921.286765 Hz output : 0.342340 V: -1.003577 dB

Frequency : 972.560569 Hz output : 0.363910 V: -0.472850 dB

Frequency : 1026.687992 Hz output : 0.364820 V: -0.451157 dB

Frequency : 1083.827854 Hz output : 0.347850 V: -0.864890 dB

Frequency : 1144.147809 Hz output : 0.352590 V: -0.747330 dB

Frequency : 1207.824844 Hz output : 0.363930 V: -0.472372 dB

Frequency : 1275.045796 Hz output : 0.352120 V: -0.758916 dB

Frequency : 1346.007899 Hz output : 0.352390 V: -0.752259 dB

Frequency : 1420.919368 Hz output : 0.360720 V: -0.549326 dB

Frequency : 1500.000000 Hz output : 0.349310 V: -0.828510 dB

bandpass filter

Ap : 0.900 dB

As : 40.000 dB

hamming window

fs1 : 300.000 Hz

fp1 : 500.000 Hz

fp2 : 700.000 Hz

fs2 : 900.000 Hz

F : 31250.000 Hz

---

frequency response

Frequency : 20.000000 Hz output : 0.018817 V: -26.535631 dB

Frequency : 21.627653 Hz output : 0.018631 V: -26.621916 dB

Frequency : 23.387769 Hz output : 0.018437 V: -26.712833 dB

Frequency : 25.291128 Hz output : 0.018233 V: -26.809477 dB

Frequency : 27.349387 Hz output : 0.017984 V: -26.928913 dB

Frequency : 29.575153 Hz output : 0.017717 V: -27.058834 dB

Frequency : 31.982057 Hz output : 0.017444 V: -27.193718 dB

Frequency : 34.584842 Hz output : 0.017144 V: -27.344395 dB

Frequency : 37.399448 Hz output : 0.016861 V: -27.488972 dB

Frequency : 40.443115 Hz output : 0.016638 V: -27.604616 dB

Frequency : 43.734483 Hz output : 0.016447 V: -27.704905 dB

Frequency : 47.293711 Hz output : 0.016325 V: -27.769575 dB

Frequency : 51.142599 Hz output : 0.016345 V: -27.758940 dB

Frequency : 55.304720 Hz output : 0.016509 V: -27.672222 dB

Frequency : 59.805565 Hz output : 0.016836 V: -27.501862 dB

Frequency : 64.672701 Hz output : 0.017333 V: -27.249165 dB

Frequency : 69.935937 Hz output : 0.018025 V: -26.909134 dB

Frequency : 75.627509 Hz output : 0.018785 V: -26.550415 dB

Frequency : 81.782277 Hz output : 0.019542 V: -26.207258 dB

Frequency : 88.437936 Hz output : 0.020221 V: -25.910585 dB

Frequency : 95.635250 Hz output : 0.020611 V: -25.744658 dB

Frequency : 103.418301 Hz output : 0.020595 V: -25.751404 dB

Frequency : 111.834757 Hz output : 0.020080 V: -25.971365 dB

Frequency : 120.936166 Hz output : 0.019054 V: -26.426914 dB

Frequency : 130.778273 Hz output : 0.017608 V: -27.112438 dB

Frequency : 141.421356 Hz output : 0.016459 V: -27.698570 dB

Frequency : 152.930602 Hz output : 0.016538 V: -27.656979 dB

Frequency : 165.376501 Hz output : 0.018322 V: -26.767181 dB

Frequency : 178.835280 Hz output : 0.021050 V: -25.561596 dB

Frequency : 193.389370 Hz output : 0.023089 V: -24.758535 dB

Frequency : 209.127911 Hz output : 0.022725 V: -24.896561 dB

Frequency : 226.147295 Hz output : 0.019383 V: -26.278219 dB

Frequency : 244.551763 Hz output : 0.016609 V: -27.619768 dB

Frequency : 264.454035 Hz output : 0.022323 V: -25.051586 dB

Frequency : 285.976007 Hz output : 0.031592 V: -22.035095 dB

Frequency : 309.249495 Hz output : 0.031530 V: -22.052158 dB

Frequency : 334.417040 Hz output : 0.016717 V: -27.563471 dB

Frequency : 361.632787 Hz output : 0.060719 V: -16.360147 dB

Frequency : 391.063424 Hz output : 0.155816 V: -8.174398 dB

Frequency : 422.889205 Hz output : 0.269010 V: -3.431270 dB

Frequency : 457.305052 Hz output : 0.362860 V: -0.831857 dB

Frequency : 494.521752 Hz output : 0.399330 V: 0.000000 dB

Frequency : 534.767246 Hz output : 0.373580 V: -0.578966 dB

Frequency : 578.288025 Hz output : 0.333610 V: -1.561857 dB

Frequency : 625.350641 Hz output : 0.344320 V: -1.287394 dB

Frequency : 676.243338 Hz output : 0.393800 V: -0.121125 dB

Frequency : 731.277817 Hz output : 0.364010 V: -0.804372 dB

Frequency : 790.791149 Hz output : 0.186100 V: -6.631711 dB

Frequency : 855.147834 Hz output : 0.019149 V: -26.383717 dB

Frequency : 924.742036 Hz output : 0.026835 V: -23.452606 dB

Frequency : 1000.00000 Hz output : 0.024470 V: -24.253960 dB

---

bandstop filter

Ap : 0.900 dB

As : 40.000 dB

blackman window

fp1 : 300.000 Hz

fs1 : 500.000 Hz

fs2 : 700.000 Hz

fp2 : 900.000 Hz

F :31250.000 Hz

---

frequency response

Frequency : 100.000000 Hz output : 0.366150 V: 0.000000 dB

Frequency : 104.712855 Hz output : 0.365640 V: -0.012107 dB

Frequency : 109.647820 Hz output : 0.364650 V: -0.035656 dB

Frequency : 114.815362 Hz output : 0.362980 V: -0.075526 dB

Frequency : 120.226443 Hz output : 0.360690 V: -0.130499 dB

Frequency : 125.892541 Hz output : 0.357580 V: -0.205716 dB

Frequency : 131.825674 Hz output : 0.353750 V: -0.299252 dB

Frequency : 138.038426 Hz output : 0.349290 V: -0.409458 dB

Frequency : 144.543977 Hz output : 0.344100 V: -0.539487 dB

Frequency : 151.356125 Hz output : 0.338560 V: -0.680467 dB

Frequency : 158.489319 Hz output : 0.332770 V: -0.830298 dB

Frequency : 165.958691 Hz output : 0.327100 V: -0.979570 dB

Frequency : 173.780083 Hz output : 0.321940 V: -1.117682 dB

Frequency : 181.970086 Hz output : 0.317710 V: -1.232563 dB

Frequency : 190.546072 Hz output : 0.315050 V: -1.305591 dB

Frequency : 199.526231 Hz output : 0.314350 V: -1.324911 dB

Frequency : 208.929613 Hz output : 0.316230 V: -1.273119 dB

Frequency : 218.776162 Hz output : 0.320890 V: -1.146057 dB

Frequency : 229.086765 Hz output : 0.328590 V: -0.940094 dB

Frequency : 239.883292 Hz output : 0.339030 V: -0.668418 dB

Frequency : 251.188643 Hz output : 0.351450 V: -0.355910 dB

Frequency : 263.026799 Hz output : 0.364550 V: -0.038039 dB

Frequency : 275.422870 Hz output : 0.376360 V: 0.000000 dB

Frequency : 288.403150 Hz output : 0.384280 V: 0.000000 dB

Frequency : 301.995172 Hz output : 0.385260 V: 0.000000 dB

Frequency : 316.227766 Hz output : 0.376380 V: 0.000000 dB

Frequency : 331.131121 Hz output : 0.354940 V: -0.270082 dB

Frequency : 346.736850 Hz output : 0.319640 V: -1.179959 dB

Frequency : 363.078055 Hz output : 0.271740 V: -2.590109 dB

Frequency : 380.189396 Hz output : 0.211260 V: -4.776835 dB

Frequency : 398.107171 Hz output : 0.146434 V: -7.960342 dB

Frequency : 416.869383 Hz output : 0.083238 V: -12.866748 dB

Frequency : 436.515832 Hz output : 0.030499 V: -21.587469 dB

Frequency : 457.088190 Hz output : 0.015277 V: -27.592419 dB

Frequency : 478.630092 Hz output : 0.032959 V: -20.913700 dB

Frequency : 501.187234 Hz output : 0.037712 V: -19.743589 dB

Frequency : 524.807460 Hz output : 0.033738 V: -20.710793 dB

Frequency : 549.540874 Hz output : 0.028343 V: -22.224264 dB

Frequency : 575.439937 Hz output : 0.025910 V: -23.003832 dB

Frequency : 602.559586 Hz output : 0.025012 V: -23.310212 dB

Frequency : 630.957344 Hz output : 0.021504 V: -24.622796 dB

Frequency : 660.693448 Hz output : 0.015657 V: -27.379009 dB

Frequency : 691.830971 Hz output : 0.014932 V: -27.790821 dB

Frequency : 724.435960 Hz output : 0.020560 V: -25.012718 dB

Frequency : 758.577575 Hz output : 0.081136 V: -13.088909 dB

Frequency : 794.328235 Hz output : 0.194990 V: -5.472933 dB

Frequency : 831.763771 Hz output : 0.321010 V: -1.142810 dB

Frequency : 870.963590 Hz output : 0.402720 V: 0.000000 dB

Frequency : 912.010839 Hz output : 0.410350 V: 0.000000 dB

Frequency : 954.992586 Hz output : 0.374760 V: 0.000000 dB

Frequency : 1000.000000 Hz output : 0.356810 V: -0.224440 dB

จากการทดลองโดยนำสัมประสิทธิ์ที่ได้มาทำการ Convolution Sum บน TMS320C50 ซึ่งจะได้ผลตอบสนองตามความถี่ที่ตั้งไว้

7.2 สรุปผลการทดลอง

จากการทดลองการใช้บอร์ดทำตัวกรองเชิงเลข (Digital Filter) ซึ่งประกอบด้วย กรองความถี่ต่ำผ่าน (Lowpass filter) กรองความถี่สูงผ่าน (Highpass Filter) กรองช่วงความถี่ผ่าน (Bandpass Filter) และกำจัดความถี่ (Bandstop Filter) โดยพอใช้โปรแกรมที่เขียนขึ้นให้รันบนซีพียูเบอร์ TMS320C50 ซึ่งเป็นฟิลเตอร์แบบ FIR (Finite Impulse Response) มีวินโดว์ครอบอีกที สามารถตอบสนองความถี่ตามที่กำหนดไว้ได้ ทั้งนี้ขึ้นอยู่กับการออกแบบตัวกรองความถี่ด้วย ซึ่งผลจากการออกแบบบางครั้งเมื่อป้อนพารามิเตอร์ที่จำกัดอาจทำให้โปรแกรมไม่ตอบสนองได้ ซึ่งพอจะสรุปข้อจำกัดของการใช้โปรแกรมได้ดังนี้

1.ความถี่สุ่ม (Sampling) จะต้องอยู่ในช่วงที่กำหนด

2.ค่า N จะต้องมีค่ามากพอ และไม่เกินค่าที่กำหนด

3.ค่า A_p ,A_s จะต้องป้อนให้เหมาะสมเพื่อไม่ให้การคำนวณค่า N มีค่ามากจนโปรแกรมทำงาน ไม่ได้

ในการใช้บอร์ดและโปรแกรมควบคุมสามารถสรุปได้ดังนี้

1.บอร์ดที่ใช้เป็น TMS320C50 DSP starter kit ซึ่งมี TLC32040 เป็นตัวแปลงสัญญาณอนาลอกดิจิตอล บอร์ดสามารถเชื่อมต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม (RS232) ในการใช้ต้องทำการโหลดโปรแกรมซึ่งถูกแปลงโดยเอสเซมเบอร์ของบอร์ดเอง

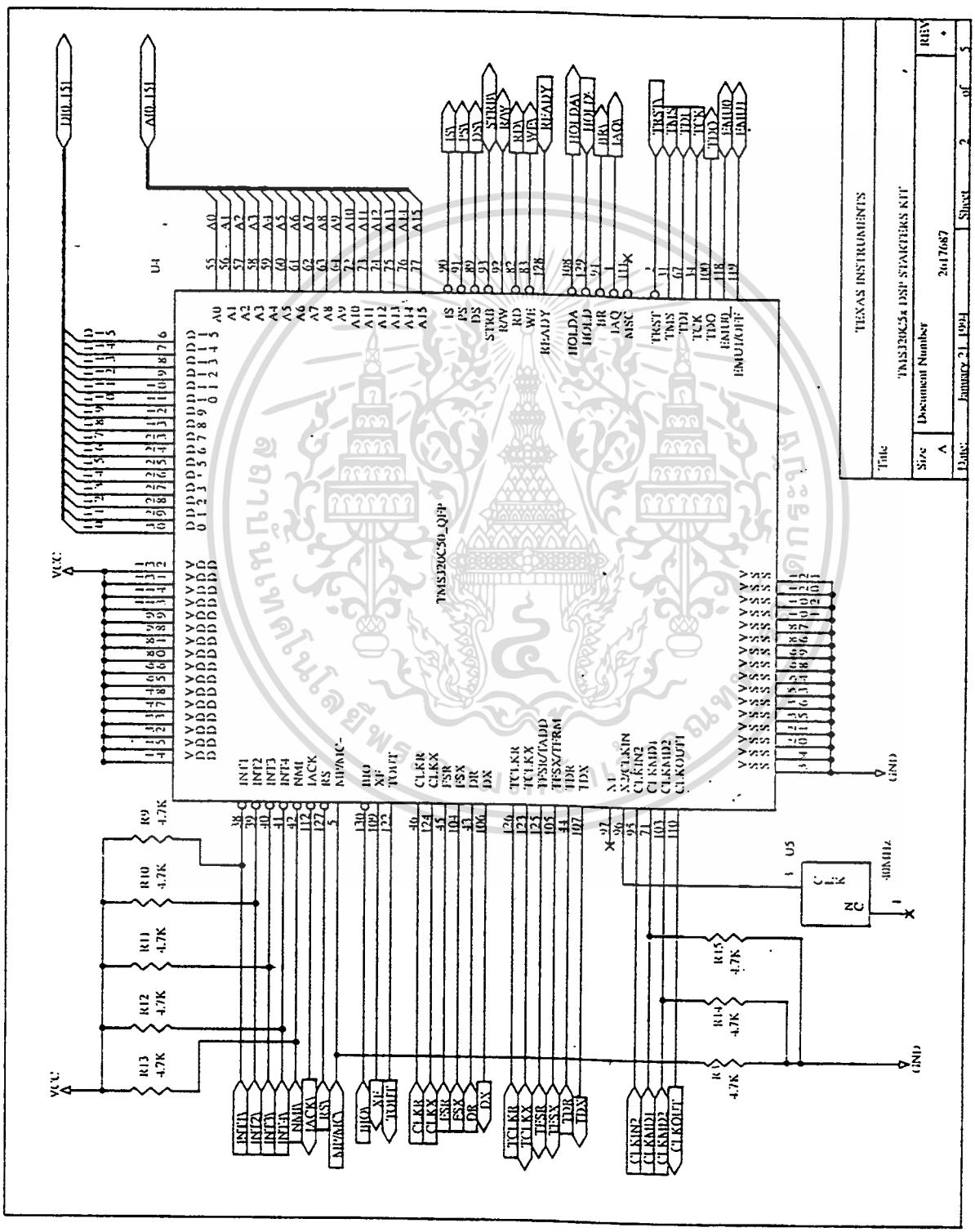
2.การใช้โปรแกรมควบคุม จะต้องโหลดโปรแกรม AS.dsk แล้วจึงการรันโปรแกรม DSP51.exe แล้วจึงสามารถออกแบบฟิลส์เตอร์และเขียนกราฟได้

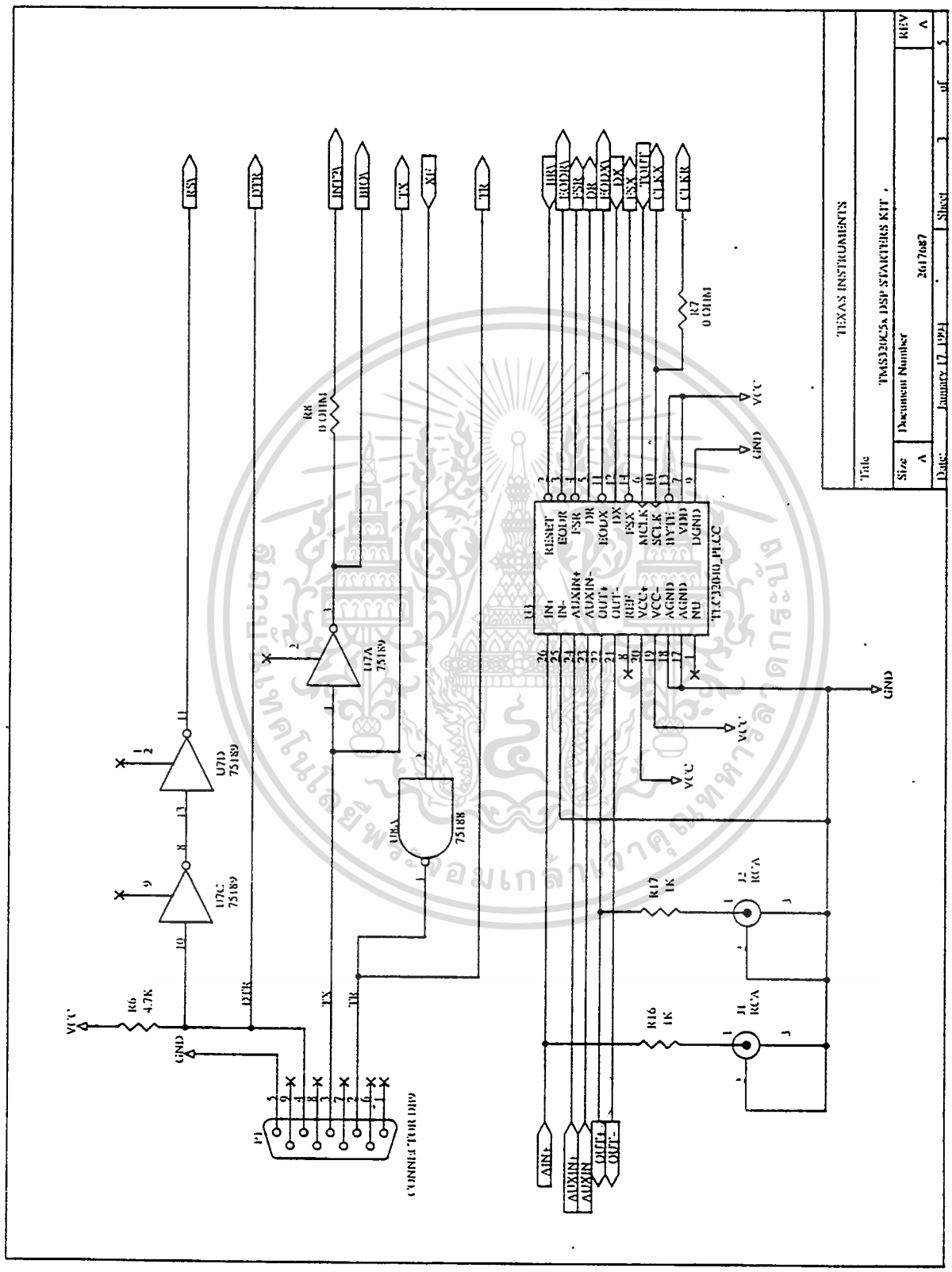
3.การรันโปรแกรมของบอร์ดสามารถดูได้จากการต่อฟังก์ชั่นเจนเนอร์เรเตอร์ป้อนเข้าไปที่ขาอินพุทแล้วต่อขาเอาท์พุทไปยังออสซิลโลสโคปดูผลได้

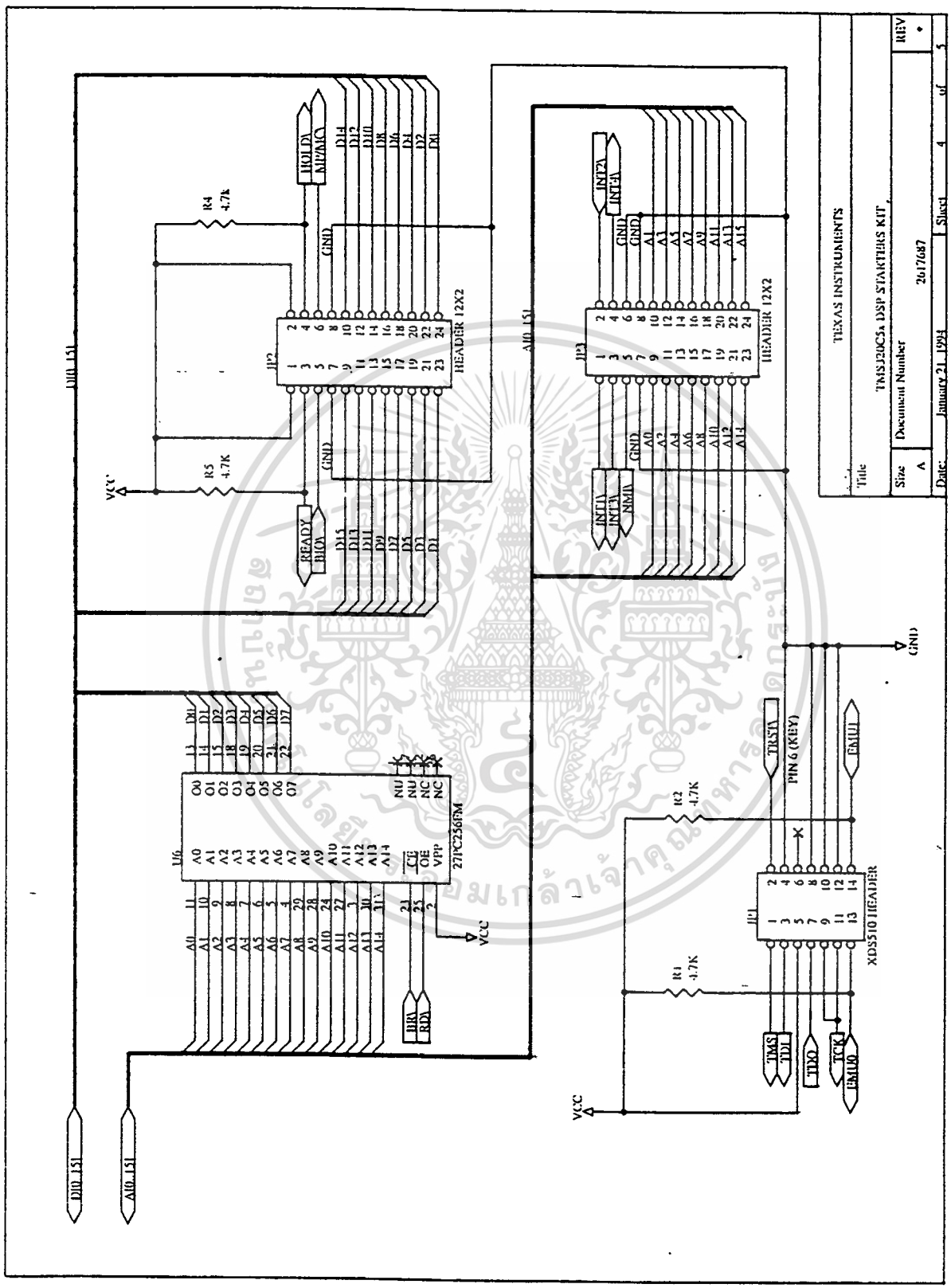
4.การออกแบบต้องดูความเหมาะสมของการใช้โปรแกรมด้วย

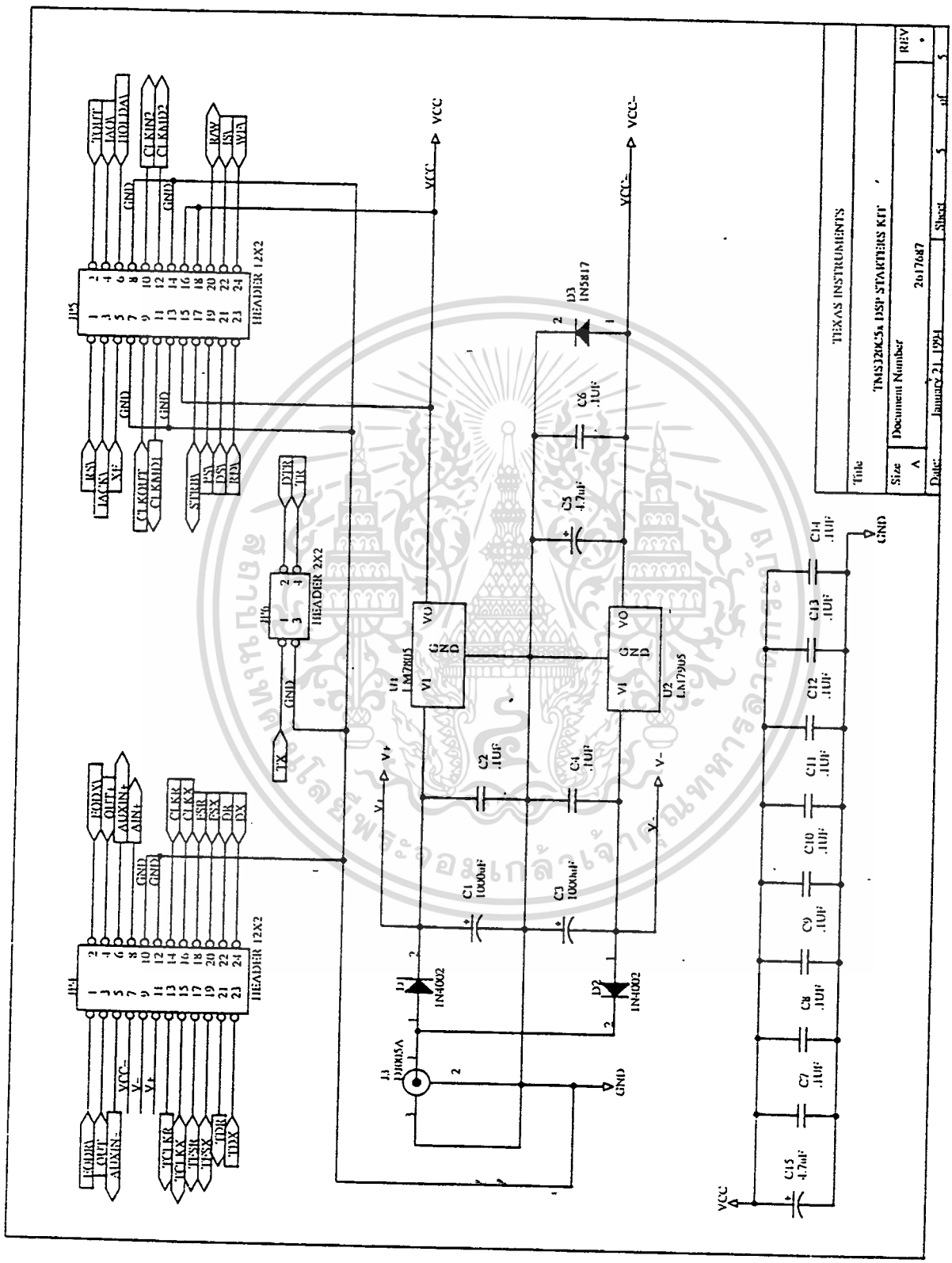
ภาคผนวก ก

รูปวงจรแสดงส่วนต่าง ๆ บนบอร์ด DSK(DSP STARTER KIT)









ภาคผนวก ข

โปรแกรมในการออกแบบฟิลเตอร์

```
int N;coeff[500],n;
#include<c:\dsp50\senddata.c>
#include<c:\dsp50\afz.c>
#include<c:\dsp50\idealf.c>
void main(void)
{
     int a,n,i;
     char
savefile1[30]="savehex",savefile2[30]="savefloat";
     system("dsk51 as.dsk");
     port_intit(PORT,231);
     sport(PORT,0x09);
     a = rport(PORT);
     if(a !=0x01b)
       {
        printf("begin program error: %x\n",a);
        getch();
        exit(0);
       }
 do{
    clrscr();
    printf("\n\n\nNOW! send command to tms\n");
    printf("     01: send LastXn\n");
    printf("     02: download \n");
    printf("     05: convolution\n");
    printf("     07: stop tms \n");
    printf("     08: send length(N)\n");
    printf("  command:");
      scanf("%x",&a);
      sport(PORT,0x80);
      sport(PORT,a);
      n = a;
      a = rport(PORT);
      if(a != n+2)
      {
printf("command error:%x",a);
       getch();
       }
      switch(n)
      {
       case 1:printf(" send LastXn = 1000h +N  ");
              a = N + 0x1000;
              printf(":%x\n",a);
              send_16(a);
              break;
       case 2:printf(" download \n");
ideallowpass(900.00,1000.00,31250.00,0.9,10.00);
              printf(" wait\n");
              writefile(savefile1,savefile2);
              send_16(0xe00);
              readfile(savefile1,savefile2);
              break;
       case 5:printf(" convolution\n");break;
       case 7:printf(" stop tms \n");break;
       case 8:printf(" send length (N)");
              a = N;
              printf( ":%x\n",a);
              send_16(a);
              break;
      default :printf(" comand error \n");break;
      }
       printf(" press any key to continue (ESC to
exit)\n");
       i =bioskey(0);
    }while(i != 0x011b);
}
int N;coeff[500],n;
```

```
#include<c:\dsp50\senddata.c>
#include<c:\dsp50\afz.c>
#include<c:\dsp50\idealf.c>
void main(void)
{
     int a,n,i;
     char
savefile1[30]="savehex",savefile2[30]="savefloat";
     system("dsk51 as.dsk");
     port_intit(PORT,231);
     sport(PORT,0x09);
     a = rport(PORT);
     if(a !=0x01b)
      {
       printf("begin program error: %x\n",a);
       getch();
       exit(0);
      }
   clrscr();
   printf("\n\n\nNOW! send command to tms\n");
   printf("     01: send LastXn\n");
   printf("     02: download \n");
   printf("     05: convolution\n");
   printf("     07: stop tms \n");
   printf("     08: send length(N)\n");
    sport(PORT,0x80);
    sport(PORT,0x02);
    a = rport(PORT);
    if(a != 0x04)
    {
    printf("command error:%x",a);
    getch();
    }
          printf(" download \n");
ideallowpass(900.00,1000.00,31250.00,0.9,10.00);
          printf(" wait\n");
          writefile(savefile1,savefile2);
          send_16(0xe00);
          readfile(savefile1,savefile2);
    sport(PORT,0x80);
    sport(PORT,0x01);
    a = rport(PORT);
    if(a != 3)
    {
    printf("command error:%x",a);
    getch();
    }
          printf(" send LastXn = 1000h +N  ");
          a = N + 0x1000;
          printf(":%x\n",a);
          send_16(a);
    sport(PORT,0x80);
    sport(PORT,0x08);
    a = rport(PORT);
    if(a != 0x0a)
    {
    printf("command error:%x",a);
    getch();
    }
          printf(" send length (N)");
          a = N;
          printf( ":%x\n",a);
          send_16(a);
    sport(PORT,0x80);
    sport(PORT,0x05);
    a = rport(PORT);
    if(a != 0x07)
    {
    printf("command error:%x",a);
    getch();
    }
    printf("convolution\n");
```

```
}
#if !defined(__AF_C)
#define __AF_C
void readfile(char fileread[30],char file1read[30]);
void writefile(char filewrite[30],char file1write[30]);
int readfile_st(char fileread[30]);
void writefile_st(char filewrite[30]);
void readfile(char fileread[30],char file1read[30])
{
  int n,datasend;
  int buf;
  FILE *coefffile,*coeff1file;
   if((coefffile=fopen(fileread,"rb"))==NULL)
    {
     printf("connot open file \n");
     exit(0);
    }
   if((coeff1file=fopen(file1read,"rb"))==NULL)
    {
     printf("connot open file \n");
     exit(0);
    }
   fread(&N,sizeof(int),1,coefffile);
   for(n=0;n<N;n++)
    {
     fread(&buf,sizeof(int),1,coefffile);
     fread(&hh[n],sizeof(float),1,coeff1file);
    }
   fclose(coefffile);
   fclose(coeff1file);
}
void writefile(char filewrite[30],char file1write[30])
{
  int n;
  int coeff;
  int buf;
  FILE *coefffile,*coeff1file;
   if((coefffile=fopen(filewrite,"wb"))==NULL)
    {
     printf("cannot open file \n");
     exit(0);
    }
   if((coeff1file=fopen(file1write,"wb"))==NULL)
    {
     printf("cannot open file \n");
     exit(0);
    }
   fwrite(&N,sizeof(int),1,coefffile);
   for(n=0;n<N;n++)
    {
     buf =hh[n]*0x2aaa;
     fwrite(&buf,sizeof(int),1,coefffile);
     fwrite(&hh[n],sizeof(float),1,coeff1file);
    }
   fclose(coefffile);
   fclose(coeff1file);
}
int readfile_st(char fileread[30])
{
  int n;
  int coeff;
  FILE *stfile;
   if((stfile=fopen(fileread,"rb"))==NULL)
    {
     setcolor(LIGHTRED);
     setfillstyle(1,WHITE);
     do{
     outtextxy(160,190,"        file not found ");
     outtextxy(160,220,"    escape to continue");
     delay(1000);
     bar(150,180,380,230);
     delay(1000);
```

```
        }while(!kbhit());
       return 1;
       }
      else
      {
      rewind(stfile);
      while(!feof(stfile))
      {
       fread(&file_filter,sizeof(struct filegrop),1,stfile);
      }
       fclose(stfile);
      }
     return 0;
 }
 void writefile_st(char filewrite[30])
 {
    int n;
    int coeff;
    FILE *stfile;
     if((stfile=fopen(filewrite,"wb"))==NULL)
      {
       printf("cannot open file \n");
       exit(0);
      }
     fwrite(&file_filter,sizeof(struct filegrop),1,stfile);
     fclose(stfile);
 }
#endif
#include<stdio.h>
#include<dos.h>
#include<string.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
void readfile_st(char fileread[30]);
void writefile_st(char fileread[30]);
```

```
#include<c:\dsk\term2\af2.c>
struct filegrop{
      int windowtype;
      float Apf;
      float Asf;
      float fp1f;
      float fp2f;
      float fs1f;
      float fs2f;
      float fsf;
      int   nf;
  }file_filter;
main()
{
 char rest[40];
 char s1[20],s2[10]=".dat";
 printf("first string:");
 gets(s1);
 strcat(s1,s2);
 printf("file name:%s\n",s1);
 printf("wintype: ");
 scanf("%d",&file_filter.windowtype);
 printf("Ap: ");
 scanf("%f",&file_filter.Apf);
 printf("As: ");
 scanf("%f",&file_filter.Asf);
 printf("fp1: ");
 scanf("%f",&file_filter.fp1f);
 printf("fp2: ");
 scanf("%f",&file_filter.fp2f);
 printf("fs1: ");
 scanf("%f",&file_filter.fs1f);
 printf("fs2: ");
 scanf("%f",&file_filter.fs2f);
 printf("F: ");
 scanf("%f",&file_filter.fsf);
```

```
  printf("N: ");
  scanf("%d",&file_filter.nf);
  writefile_st(s1);
  file_filter.windowtype =0;
  file_filter.Apf=0;
  file_filter.Asf=0;
  file_filter.fp1f=0;
  file_filter.fp2f=0;
  file_filter.fs1f=0;
  file_filter.fs2f=0;
  file_filter.fsf=0;
  file_filter.nf=0;
  readfile_st(s1);
  printf("wintype: ");
  printf("%d\n",file_filter.windowtype);
  printf("Ap: ");
  printf("%f\n",file_filter.Apf);
  printf("As: ");
  printf("%f\n",file_filter.Asf);
  printf("fp1: ");
  printf("%f\n",file_filter.fp1f);
  printf("fp2: ");
  printf("%f\n",file_filter.fp2f);
  printf("fs1: ");
  printf("%f\n",file_filter.fs1f);
  printf("fs2: ");
  printf("%f\n",file_filter.fs2f);
  printf("F: ");
  printf("%f\n",file_filter.fsf);
  printf("N: ");
  printf("%d\n",file_filter.nf);
  getch();
 }
void readfile_st(char fileread[30])
{
  int n;
  int coeff;
  FILE *stfile;
   if((stfile=fopen(fileread,"rb"))==NULL)
    {
     printf("connot open file \n");
     exit(0);
    }
   rewind(stfile);
   while(!feof(stfile))
   {
    fread(&file_filter,sizeof(struct filegrop),1,stfile);
   }
    fclose(stfile);
 }
void writefile_st(char filewrite[30])
{
  int n;
  int coeff;
  FILE *stfile;
   if((stfile=fopen(filewrite,"wb"))==NULL)
    {
     printf("cannot open file \n");
     exit(0);
    }
    fwrite(&file_filter,sizeof(struct filegrop),1,stfile);
    fclose(stfile);
}
#if !defined(__AF_C)
#define __AF_C
FILE *coefffile;
void readfile(char fileread[30]);
void writefile(char filewrite[30]);
void readfile(char fileread[30])
{
  int n;
  int coeff;
```

```
    if((coefffile=fopen(fileread,"rb"))==NULL)
     {
      printf("connot open file \n");
      exit(0);
     }
    fread(&coeff,sizeof(int),1,coefffile);
    printf("first:%x\n",coeff);
    for(n=0;n<10;n++)
     {
      fread(&coeff,sizeof(int),1,coefffile);
      printf("%d:%x\n",n,coeff);
     }
    fclose(coefffile);
}
void writefile(char filewrite[30])
{
  int n;
  int coeff;
   if((coefffile=fopen(filewrite,"wb"))==NULL)
    {
     printf("cannot open file \n");
     exit(0);
    }
   printf("first data(hex):");
   scanf("%x\n",&coeff);
   fwrite(&coeff,sizeof(int),1,coefffile);
   for(n=0;n<10;n++)
    {
     printf("%d data(hex):",n);
     scanf("%x\n",&coeff);
     fwrite(&coeff,sizeof(int),1,coefffile);
    }
    fclose(coefffile);
}
#endif
#if !defined(__AFZ_C)
#define __AFZ_C
void readfile(char fileread[30],char file1read[30]);
void writefile(char filewrite[30],char file1write[30]);
/*int readfile_st(char fileread[30]);
void writefile_st(char filewrite[30]);*/
void readfile(char fileread[30],char file1read[30])
{
  int n;
  int buf;
  FILE *coefffile,*coeff1file;
  if((coefffile=fopen(fileread,"rb"))==NULL)
   {
    printf("connot open file \n");
    exit(0);
   }
  if((coeff1file=fopen(file1read,"rb"))==NULL)
   {
    printf("connot open file \n");
    exit(0);
   }
  fread(&N,sizeof(int),1,coefffile);
  send_16(N);
  for(n=0;n<N;n++)
   {
    fread(&buf,sizeof(int),1,coefffile);
    fread(&coeff[n],sizeof(float),1,coeff1file);
    send_16(buf);
   }
   fclose(coefffile);
   fclose(coeff1file);
}
void writefile(char filewrite[30],char file1write[30])
{
  int n;
  int buf;
  FILE *coefffile,*coeff1file;
```

```
    if((coefffile=fopen(filewrite,"wb"))==NULL)
     {
      printf("cannot open file \n");
      exit(0);
     }
    if((coeff1file=fopen(file1write,"wb"))==NULL)
     {
      printf("cannot open file \n");
      exit(0);
     }
    fwrite(&N,sizeof(int),1,coefffile);
    for(n=0;n<N;n++)
     {
      buf =coeff[n];
      fwrite(&buf,sizeof(int),1,coefffile);
      fwrite(&coeff[n],sizeof(float),1,coeff1file);
     }
     fclose(coefffile);
     fclose(coeff1file);
}
/*int readfile_st(char fileread[30])
{
   int n;
   int coeff;
   FILE *stfile;
     if((stfile=fopen(fileread,"rb"))==NULL)
      {
       setcolor(LIGHTRED);
       setfillstyle(1,WHITE);
       do{
       outtextxy(160,190,"          file not found ");
       outtextxy(160,220,"     escape  to  continue");
       delay(1000);
       bar(150,180,380,230);
       delay(1000);
       }while(!kbhit());
       return 1;
      }
     else
     {
     rewind(stfile);
     while(!feof(stfile))
     {
      fread(&file_filter,sizeof(struct filegrop),1,stfile);
     }
       fclose(stfile);
     }
     return 0;
   }
  void writefile_st(char filewrite[30])
  {
    int n;
    int coeff;
    FILE *stfile;
     if((stfile=fopen(filewrite,"wb"))==NULL)
      {
       printf("cannot open file \n");
       exit(0);
      }
      fwrite(&file_filter,sizeof(struct filegrop),1,stfile);
      fclose(stfile);
  } */
  #endif
  #if !defined(__BLOCK_C)
  #define __BLOCK_C
  void drawblock(int xs,int ys,int xl,int yl,int dimen)
  {
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(BLACK);/*LIGHTGRAY);*/
    setfillstyle(1,LIGHTGRAY);
    bar(xs+10,ys+10,xl+10,yl+10);
    setfillstyle(1,WHITE);
```

```
    bar(xs,ys,xl,yl);
    setcolor(BLACK);
    rectangle(xs+2,ys+2,xl-2,yl-2);
    line(xs+3,ys+3+dimen,xl-3,ys+3+dimen);
    setcolor(LIGHTGRAY);
    rectangle(xs+3,ys+3,xl-3,yl-3);
    line(xs+2,ys+2+dimen,xl-2,ys+2+dimen);
  }

#if !defined (__DESIGN_C)
#define __DESIGN_C
void graphlp(void);
void graphhp(void);
void graphbp(void);
void graphbs(void);
void ok(void);
void notok(void);
void design(void)
{    int n,h;
     menudesign();
    do  {
            switch(selec)
              {
                    case 0: fil(); break;
                    case 1: winselec(); break;
                    case 2: selecname();break;
                    case 3: selecap();break;
                    case 4: selecas();break;
                    case 5: selecfp1();break;
                    case 6: selecfs1();break;
                    case 7: selecfp2();break;
                    case 8: selecfs2();break;
                    case 9: selecf();break;
                    case 10:ok();break;
                    case 11:notok();break;
              }
    }while(selec1 !=ESC);
                    drawmain();
              putimage(50,100,saveveiw[0],4);
                    setcolor(WHITE);
                    setfillstyle(1,BLACK);
                    bar(360,75,getmaxx()-60,90);
              setfillstyle(1,WHITE);
            bar(370,100,getmaxx()-60,getmaxy()-80);
             outtextxy(380,80,"    Design FIR fiter");
                    helpdesign();
}
void selecfs2(void)
{
  float s2;
  selec =8;
  setfillstyle(1,BLACK);
   bar(getmaxx()-100,270,getmaxx()-20,289);
  bar(getmaxx()-200,270,getmaxx()-135,285);
  setcolor(WHITE);
  outtextxy(getmaxx()-200,275,"     fs2");
  setcolor(BLUE);
  gotoxy(69,18);
  scanf("%f",&fs2);
  setfillstyle(1,WHITE);
  bar(getmaxx()-200,270,getmaxx()-135,285);
  setcolor(BLUE);
  outtextxy(getmaxx()-200,275,"     fs2");
  selec = 9;
}
 void initialize_graphics_mode(void)
 {
   int gdriver =DETECT,gmode,errorcode;
   initgraph(&gdriver,&gmode,"c:\\turboc\\graphic");
   errorcode = graphresult();
   if (errorcode != grOk)
     {
```

```
        printf(" Graphics error :
%s\n",grapherrormsg(errorcode));
        printf(" Press any key halt : ");
        getch();
        exit(1);
       }
 }
 void close_graphics_mode(void)
 {
  closegraph();
 }
 void drawmain(void)
 {   int x,y;
     setviewport(0,0,getmaxx(),getmaxy(),1);
     setbkcolor(LIGHTGRAY);
     setcolor(LIGHTGRAY);
     drawblock1(10,10,getmaxx()-10,getmaxy()-40,30);
     drawblock1(10,getmaxy()-40,getmaxx()-
10,getmaxy()-10,0);
     setfillstyle(1,1);
     bar(40,14,getmaxx()-14,41);
     bar(15+30/4,18,10+30*3/4,36);
     bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
     settextstyle(0,0,1);
     setcolor(WHITE);
     outtextxy(150,24,"  Digital Filter Control  DSP
card");
     outtextxy(45,getmaxy()-30,"  F1 Help  F2 Plot
F3 Run  F4 Design  F10 Menu  Alt-x exit");
     drawblock(50,100,275,125,0);
     drawblock(50,150,275,175,0);
     drawblock(50,200,275,225,0);
     drawblock(50,250,275,275,0);
     drawblock(50,300,275,325,0);
     setcolor(BLACK);
     outtextxy(70,110,"  Design FIR filter");
     outtextxy(70,160,"  Execute program ");
     outtextxy(70,210,"  Plot graph filter");
     outtextxy(70,260,"  TMS320C50 data");
     outtextxy(70,310,"  Exit to dos ");
     drawblock(350,70,getmaxx()-50,getmaxy()-70,20);
     setcolor(BLUE);
     setfillstyle(1,BLACK);
     bar(360,75,getmaxx()-60,90);
     outtextxy(380,80,"  Design FIR fiter");
     outtextxy(430,200,"help design");
 }
 void drawmain1(void)
 {   int x,y;
     setcolor(BLUE);
     setfillstyle(1,BLACK);
     bar(360,75,getmaxx()-60,90);
     outtextxy(380,80,"  Design FIR fiter");
     outtextxy(430,200,"help design");
     setcolor(BLACK);
     drawblock(80,80,getmaxx()-80,getmaxy()-80,0);
     settextstyle(0,0,8);
     outtextxy(170+5,140+5,"DSP50");
     setcolor(BLUE);
     outtextxy(170,140,"DSP50");
     setfextstyle(0,0,1);
     outtextxy(170,230," Program for control DSP card
TMS320C50");
     outtextxy(200,250,"  Version 1.00");
     outtextxy(250,280,"  By");
     outtextxy(200,320,"santipap  ukot");
     outtextxy(200,330,"alongkot  jaivanglok");
     outtextxy(200,340,"alun  sudton");
     settextstyle(0,0,1.5);
     outtextxy(180,360,"Engineer Electronics  KMITL");
     settextstyle(0,0,1);
     getch();
```

```
}
void filtertype(void)
{
  int i,num,cal,xs,ys,xl,yl,ynew1,ynew2;
  int cal1;
  num = 0;
  cal1= (getmaxx())/6;
  drawblock(130,210,260,338,0);
  setfillstyle(1,LIGHTGRAY);
  bar(133,214,257,239);
  setcolor(BLUE);
  outtextxy(140,220," lowpass ");
  outtextxy(140,250," highpass");
  outtextxy(140,280," bandpass");
  outtextxy(140,310," bandstop");
  xs = 133; ys = 214;
  xl = 257; yl = 337;
  cal =(yl-ys)/4;
   for(;;) {
                  i =bioskey(0);
                  switch(i)
                  {
          case UP     : if(num == 0)
          {
            setfillstyle(1,WHITE);
            bar(xs,ys,xl,ys+cal);
            outtextxy(140,220," lowpass");
            num=3;
           setfillstyle(1,LIGHTGRAY);
            bar(xs,ys+cal*3,xl,ys+cal*4);
            outtextxy(140,310," bandstop");
          }
          else
          {
            switch(num)
            {  case 1:ynew1=ys+cal;ynew2=ys+cal*2;
                 setfillstyle(1,WHITE);
                 bar(xs,ynew1,xl,ynew2);
                 outtextxy(140,250," highpass");
                 num=num-1;
                 setfillstyle(1,LIGHTGRAY);
                 bar(xs,ynew1-cal,xl,ynew2-cal);
               outtextxy(140,220," lowpass");
               break;
            case 2:ynew1=ys+cal*2;ynew2=ys+cal*3;
               setfillstyle(1,WHITE);
               bar(xs,ynew1,xl,ynew2);
               outtextxy(140,280," bandpass");
               num=num-1;
               setfillstyle(1,LIGHTGRAY);
               bar(xs,ynew1-cal,xl,ynew2-cal);
               outtextxy(140,250," highpass");
               break;
            case 3:ynew1=ys+cal*3;ynew2=ys+cal*4;
               setfillstyle(1,WHITE);
               bar(xs,ynew1,xl,ynew2);
               outtextxy(140,310," bandstop");
               num=num-1;
               setfillstyle(1,LIGHTGRAY);
               bar(xs,ynew1-cal,xl,ynew2-cal);
               outtextxy(140,280," bandpass");
               break;
            }
          }break;
          case DOWN     : if(num == 3)
          {
            setfillstyle(1,WHITE);
            bar(xs,ys+cal*3,xl,ys+cal*4);
            outtextxy(140,310," bandstop");
            num=0;
            setfillstyle(1,LIGHTGRAY);
            bar(xs,ys,xl,ys+cal);
```

```
 outtextxy(140,220," lowpass");
}
else
{
 switch(num)
 { case 0:ynew1=ys;ynew2=ys+cal;
    setfillstyle(1,WHITE);
                    bar(xs,ynew1,xl,ynew2);
                    outtextxy(140,220," lowpass");
                    num=num+1;
                    setfillstyle(1,LIGHTGRAY);
                   bar(xs,ynew1+cal,xl,ynew2+cal);
                    outtextxy(140,250," highpass");
                    break;
     case 1:ynew1=ys+cal;ynew2=ys+cal*2;
                  setfillstyle(1,WHITE);
                    bar(xs,ynew1,xl,ynew2);
                    outtextxy(140,250," highpass");
                    num=num+1;
                    setfillstyle(1,LIGHTGRAY);
          bar(xs,ynew1+cal,xl,ynew2+cal);
                    outtextxy(140,280," bandpass");
                    break;
    case 2:ynew1=ys+cal*2;ynew2=ys+cal*3;
                  setfillstyle(1,WHITE);
                    bar(xs,ynew1,xl,ynew2);
                    outtextxy(140,280," bandpass");
                    num=num+1;
                 setfillstyle(1,LIGHTGRAY);
                    bar(xs,ynew1+cal,xl,ynew2+cal)
         outtextxy(140,310," bandstop");
         break;
                  }
      }break;
case RETURN :  setfillstyle(1,WHITE);
        settextstyle(0,0,2);
                    bar(176+cal1,114,185+cal1*2,125);
                  switch(num)
                    {
                 case 0: filtype=0;
                      outtextxy(178+cal1,117,"
lowpass");break;
                      case 1: filtype=1;
                 outtextxy(178+cal1,117,"  highpass");break;
case 2: filtype=2;
                 outtextxy(178+cal1,117,"  bandpass");break;
case 3: filtype=3;
                 outtextxy(178+cal1,117,"  bandstop");break;
                  }
                 bar(100,150,550,420);
                 drawfiltertype();
                 break;
case ESC :setfillstyle(1,WHITE)
                  bar(100,150,550,420);
                  drawfiltertype();
                  break;
                  } /*switch*/
                    if(i==RETURN)break;
                    if(i==ESC)break;
                 }/*for*/
}
void drawfiltertype(void)
{
 outtextxy(120,155," dB");
 line(150,150,150,380);
 line(140,380,550,380);
 line(151,150,151,380);
 line(140,381,550,381);
 switch(filtype)
 {
   case 0:drawlowpass();break;
   case 1:drawhighpass();break;
```

```
    case 2:drawbandpass();break;
    case 3:drawbandstop();break;
  }
 }
void drawlowpass(void)
{
  setfillstyle(1,LIGHTBLUE);
  bar(152,160,300,180);
  bar(380,360,550,378);
  outtextxy(300,390,"fc");
  outtextxy(380,390,"ft");
}
void initialize_graphics_mode(void)
{
   int gdriver =DETECT,gmode,errorcode;
   initgraph(&gdriver,&gmode,"z:\\lang\\tc\\graphic");
   errorcode = graphresult();
   if (errorcode != grOk)
    {
     printf(" Graphics error :
%s\n",grapherrormsg(errorcode));
     printf(" Press any key halt : ");
     getch();
     exit(1);
    }
 }
void close_graphics_mode(void)
{
 closegraph();
}
void drawmain(void)
{
   int x,y;
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(BLACK);
    setcolor(LIGHTGRAY);
     drawblock1(10,10,getmaxx()-10,getmaxy()-40,30);
 drawblock1(10,getmaxy()-40,getmaxx()-10,getmaxy()-
10,0);
     setfillstyle(1,1);
     bar(40,14,getmaxx()-14,41);
     bar(15+30/4,18,10+30*3/4,36);
     bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
     settextstyle(0,0,1);
     setcolor(WHITE);
   outtextxy(150,24,"Digitall Filter Control  DSP card");
     outtextxy(45,getmaxy()-30,"  F1 Help  F2 Plot
F3 Run  F4 Design  F10 Menu  Alt-x exit");
     drawblock(50,100,275,125,0);
     drawblock(50,150,275,175,0);
     drawblock(50,200,275,225,0);
     drawblock(50,250,275,275,0);
     drawblock(50,300,275,325,0);
     setcolor(BLACK);
     outtextxy(70,110,"   Design FIR filter");
     outtextxy(70,160,"   Execute program ");
     outtextxy(70,210,"   Plot graph filter");
     outtextxy(70,260,"   About project");
     outtextxy(70,310,"   Exit to dos ");
     drawblock(350,70,getmaxx()-50,getmaxy()-70,20);
     setcolor(WHITE);
     setfillstyle(1,BLACK);
     bar(360,75,getmaxx()-60,90);
     outtextxy(380,80,"   Design FIR fiter");
     helpdesign();
}
void drawmain1(void)
 {
     setcolor(BLACK);
    drawblock4(80,80,getmaxx()-80,getmaxy()-80,0);
    settextstyle(0,0,8);
    outtextxy(170+5,140+5,"DSP50");
```

```
    setcolor(BLUE);
    outtextxy(170,140,"DSP50");
    settextstyle(0,0,1);
    outtextxy(170,230," Program for control DSP card
TMS320C50");
    outtextxy(200,250,"      Version 1.00");
    outtextxy(250,280,"   By");
    outtextxy(200,320,"santipap     ukot");
    outtextxy(200,330,"alongkot     jaivanglok");
    outtextxy(200,340,"wirunraya    sudson");
    settextstyle(0,0,1.5);
    outtextxy(180,360,"Engineer Electronics  KMIT'L");
    settextstyle(0,0,1);
    setfillstyle(1,WHITE);
    setcolor(LIGHTRED);
    do{
outtextxy(170,375,"press any key to continue");
          delay(1000);
          bar(165,370,450,390);
          delay(1000);
          }while(!kbhit());
}
void drawline(int sx,int sy,int lx,int ly,int color)
{
  int x,y;
    for(x=sx;x<=lx;x+=1)
      for(y=sy;y<=ly;y+=1)
          putpixel(x,y,color);
}
void selecmain(void)
{
    int i;
    int xs=66;
   for(;;) {
          i =bioskey(0);
          switch(i)
{
 case UP    : if(choice == 0)
 {
  putimage(50,100,saveveiw[choice],0);
  choice =4;
  putimage(50,300,saveveiw[choice],4);
  menuhelp(choice);
 }
 else
 {
  switch(choice)
  {  case 1:xs=150;break;
       case 2:xs=200;break;
       case 3:xs=250;break;
       case 4:xs=300;break;
  }
   putimage(50,xs,saveveiw[choice],0);
   choice =choice-1;
   xs = xs-50;
   putimage(50,xs,saveveiw[choice],4);
   menuhelp(choice);
 }break;
case DOWN: if(choice == 4)
 {
  putimage(50,300,saveveiw[choice],0);
  choice =0;
  putimage(50,100,saveveiw[choice],4);
  menuhelp(choice);
 }
 else
 {
          switch(choice)
          {  case 0:xs=100;break;
               case 1:xs=150;break;
               case 2:xs=200;break;
               case 3:xs=250;break;
```

```
                    }
                putimage(50,xs,saveveiw[choice],0);
                    choice =choice+1;
                    xs = xs+50;
                putimage(50,xs,saveveiw[choice],4);
                menuhelp(choice);
            } break;
case RETURN :switch(choice)
  {
            case 0:design();break;
            case 1:run();break;
            case 2:plot();break;
            case 3:datatms();break;
            case 4:exitmain();break;
           }break;
    case F1    :choice=3;datatms();break;
            case F2    :choice=2;plot();break;
              case F3    :choice=1;run();break;
              case F4    :choice=0;design();break;
              case F10   :break;
              case Altx  :choice=4;exitmain();break;
          } /*switch*/
    }/*for*/
} /*void*/
void saveblock(int xs,int ys,int xl,int yl,int num )
  {
   unsigned int size;
            size =imagesize(xs,ys,xl,yl);
            saveveiw[num] = malloc(size);
            getimage(xs,ys,xl,yl,saveveiw[num]);
  }
void saveblockmain()
{
   saveblock(50,100,275,125,0);
   saveblock(50,150,275,175,1);
   saveblock(50,200,275,225,2);
   saveblock(50,250,275,275,3);
   saveblock(50,300,275,325,4);
   putimage(50,100,saveveiw[0],4);
   setcolor(WHITE);
   setfillstyle(1,BLACK);
   bar(360,75,getmaxx()-60,90);
   outtextxy(380,80,"   Design FIR fiter");
   outtextxy(430,200,"help design");
}
void filtername(void)
{
 char name[80];
 int cal;
 cal= (getmaxx())/6;
 saveblock(100,105,410,145,5);
 drawblock2(100,105,400,135,0);
 gotoxy(20,8);
 gets(name);
 putimage(100,105,saveveiw[5],0);
 setfillstyle(1,WHITE);
 bar(126,114,135+cal,125);
 settextstyle(0,0,0.2);
 setcolor(BLUE);
 outtextxy(128,117,name);
}
void filternameplot(void)
{
 char name[80];
 saveblock(100,105,410,145,5);
 drawblock2(100,105,400,135,0);
 gotoxy(20,8);
 gets(name);
 putimage(100,105,saveveiw[5],0);
 setfillstyle(1,WHITE);
 bar(300,80,400,90);
 settextstyle(0,0,0.2);
```

```
  setcolor(BLUE);
  outtextxy(300,83,name);
 }
 #if !defined(__ERRORD_C)
 #define __ERRORD_C
 void errord(void);
 void errord(void)
 { int i,k;
  k=1;
  drawblock3(170,130,450,250,0);
  setcolor(WHITE);
  outtextxy(200,150,"        Error    ");
  outtextxy(200,175,"parameter mismatch in filter");
  setfillstyle(1,LIGHTRED);
  do{
  outtextxy(200,200,"  press any key continue");
  delay(1000);
  bar(200,185,440,220);
  delay(1000);
  }while(!kbhit());
  getch();
}
#endif
#if !defined(__ERRORN_C)
#define __ERRORN_C
void errorn(void);
void errorn(void)
{ int i,k;
 k=1;
 drawblock3(170,130,450,250,0);
 setcolor(WHITE);
 outtextxy(200,150,"        Error    ");
 outtextxy(200,175,"parameter N < 0 in filter");
 setfillstyle(1,LIGHTRED);
 do{
outtextxy(200,200,"  press any key continue");
  delay(1000);
  bar(200,185,440,220);
  delay(1000);
  }while(!kbhit());
  getch();
}
#endif
#if !defined(__EXIT_C)
#define __EXIT_C
void exitmain(void)
{ int i,k;
  k=1;
  drawblock3(170,130,450,250,0);
  drawblock2(200,190,300,215,0);
  drawblock2(330,190,430,215,0);
  setcolor(WHITE);
  outtextxy(200,150,"        Exit to Dos    ");
  outtextxy(195,170,"Are you sure you want to quit?");
  setcolor(BLACK);
  outtextxy(210,200,"   Quit");
  outtextxy(340,200,"   Cancel");
  saveblock(200,190,300,215,5);
  saveblock(330,190,430,215,6);
  putimage(330,190,saveveiw[6],4);
    for(;;) {
                i =bioskey(0);
                switch(i)
                {
          case LEFT     : if(k == 0)
                  {
                    k=1;
                    putimage(330,190,saveveiw[6],4);
                    putimage(200,190,saveveiw[5],0);
                  }
                  else
                  {
```

```
                k=0;
                putimage(200,190,saveveiw[5],4);
                putimage(330,190,saveveiw[6],0);
               }break;
        case RIGHT    : if(k==1)
                {
                 k=0;
                 putimage(200,190,saveveiw[5],4);
          putimage(330,190,saveveiw[6],0);
        }
        else
        {
          k=1;
          putimage(200,190,saveveiw[5],0);
          putimage(330,190,saveveiw[6],4);
        } break;
case RETURN :switch(k)
        {
                case 0:closegraph();exit(0);
                case 1:break;
        }
case ESC : break;
  } /*switch*/
        if(i==ESC)
        {
         drawmain();
         putimage(50,300,saveveiw[4],4);
         setcolor(WHITE);
         setfillstyle(1,BLACK);
         bar(360,75,getmaxx()-60,90);
        setfillstyle(1,WHITE);
        bar(370,100,getmaxx()-60,getmaxy()-80);
        outtextxy(380,80,"    Exit to dos");
        helpexit();
        break;
        }
        if(i==RETURN)
         if(k==1)
         {
          drawmain();
          putimage(50,300,saveveiw[4],4);
          setcolor(WHITE);
          setfillstyle(1,BLACK);
          bar(360,75,getmaxx()-60,90);
         setfillstyle(1,WHITE);
          bar(370,100,getmaxx()-60,getmaxy()-80);
          outtextxy(380,80,"    Exit to dos");
          helpexit();
          break;
          }
            }/*for*/
          }
          #endif
          #if !defined(__FILFILE_C)
          #define __FILFILE_C
          #include<stdio.h>
          #include<conio.h>
          #include<conio.h>
          FILE *coefffile;
          void readfile(void);
          void writefile(void);
          void readfile(void)
          {
            int n;
             if((coefffile=fopen("testfile","r"))==NULL)
              {
               printf("connot open file \n");
               exit(0);
              }
             fread(&N,2,1,coefffile);
             for(n=0;n<N;n++)
             {
```

```
      fread(&coeff[n],2,1,coefffile);
      }
      fclose(coefffile);
}
void writefile(void)
{
   int n;
     if((coefffile=fopen("testfile","w"))==NULL)
      {
       printf("cannot open file \n");
       exit(0);
      }
     fwrite(&N,2,1,coefffile);
     for(n=0;n<N;n++)
      {
       fwrite(&coeff[n],2,1,coefffile);
      }
     fclose(coefffile);
}
#endif
#include <bios.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
int stringin(int x,int y,int length,char *s);
long integerin(int x,int y,long startvalue);
void main(void)
{
  char  stringdata[20] = "";
  long intin=0;
  int s;
  for(s=0;s<3;s++)
  {
  clrscr(); /* Clear screen */
  gotoxy(1,1);
  cprintf("Old string is %ld",intin);
  gotoxy(1,2);
  cprintf("Please edit string :");
  intin= integerin(21,2,intin);
/*   stringin(21,2,9,stringdata);*/
  gotoxy(1,3);
/* cprintf("New string is %s",stringdata);*/
  cprintf("New string is %ld",intin);
  getch();
  }
}
int stringin(int x,int y,int length,char *s)
/*  stringin() :Read character from keyboard */
/*  (x,y)      :Start location*/
/*  length     :Maximum input     th */
/*  s          :Input and output string */
/*  return     = 0  Cancel editing*/
/*             = 1  Accept editing*/
/*             = -1 Error in function*/
{
  int xx,c;
  unsigned char  oldchar[20],ch,chex;
  if (length>9)
  {
    length =9;
  }
  sprintf(oldchar,"%s",s); /* Store old string  */
  for(xx=0;oldchar[xx]!=0;xx++);/* Set cursor to end of
text */
  for(c=xx+1;c<length;c++)
    s[c] = 0;     /* Clear string area */
  do{
     gotoxy(x,y);
     cputs(s); /* Put string to screen*/
     putch(0x20); /* Put space follow string*/
     gotoxy(x+xx,y); /* Move cursor to edit location*/
     ch = getch(); /* Read first character */
```

```
    if(ch<0x20) /* Control code */
    {
          switch(ch)
          {
    case 0x00 :chex = getch(); /* Read scan code*/
          if((chex==75)&&(xx>0))xx--; /* Move left */
if((chex==77)&&(xx<length))xx++;/*Move right*/
                    break;
    case 0x1b :sprintf(s,"%s",oldchar);  /* restore old
string*/
          return 0;  /* cancel editing*/
case 0x07 :for(c=xx;c<(length-1);c++) /* del
character<^G> */
           s[c] = s[c+1]; /* Move charactor leftside */
          s[length-1] = 0; /* Clear most right
character*/
          break;
case 0x08 :if (xx>0)       /* del character <DEL> */
          {
            xx--;   /* Like case 0x07 */
            for (c=xx;c<(length-1);c++)
                  s[c] = s[c+1];
            s[length-1] = 0;
          }
  }
    }
    else
    {
  if(xx < length)
  {
        for(c=length-2;c>=xx;c--) /* insert character */
           s[c+1] = s[c];
        s[xx] = ch;
        xx++; /* Move cursor location*/
           }
    }
  }while(ch!=0x0d);
  return 1;
}
/* These two functions use stringin() for
  editing value */
long integerin(int x,int y,long startvalue)
/* read integer value */
{
 char rest[80];
 long longout;
 sprintf(rest,"%ld",startvalue);
 stringin(x,y,15,rest);
 longout = atol(rest);
 return(longout);
}
double floatin(int x,int y,double startvalue)
/* read float value */
{
 char rest[50];
 double doubleout;
 sprintf(rest,"%lf",startvalue);
 stringin(x,y,9,rest);
 doubleout = atof(rest);
 return(doubleout);
}
void helpdata(void);
                                    tem = PI*(n-N/2);
                                    hh[n] = (sin(wc*(n-
N/2)))/tem;
                              }
                        }
    }
  }

#if !defined(__MENU_C)
```

```
#define __MENU_C
void menuhelp(int choice)
{   setfillstyle(1,WHITE);
    bar(370,100,getmaxx()-60,getmaxy()-80);
    setfillstyle(1,BLACK);
    bar(360,75,getmaxx()-60,90);
    setcolor(WHITE);

    switch(choice)
    { case 0:outtextxy(380,80,"   Design FIR fiter");
             helpdesign();
             break;
     case 1:outtextxy(380,80,"   Execute program");
             helprun();
             break;
     case 2:outtextxy(380,80,"   Plot graph filter");
             helpplot();
             break;
     case 3:outtextxy(380,80,"   TMS320c50 data");
             helpdata();
             break;
     case 4:outtextxy(380,80,"   Exit to dos");
             helpexit();
             break;
    }

}
void menudesign(void)
{
    int x,y;
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(BLACK);
    setcolor(LIGHTGRAY);
    drawblock1(10,10,getmaxx()-10,getmaxy()-40,30);
    drawblock1(10,getmaxy()-40,getmaxx()-
10,getmaxy()-10,0);
    setfillstyle(1,1);
    bar(40,14,getmaxx()-14,41);
    bar(15+30/4,18,10+30*3/4,36);
    bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
    settextstyle(0,0,1);
    setcolor(WHITE);
    outtextxy(150,24,"   DESIGN FOR FIR FILTER");
    outtextxy(45,getmaxy()-30,"      up,down selec and
enter to choice  (Cancel to exit)");
    setcolor(BLACK);
    setfillstyle(1,BLACK);
    bar(getmaxx()-100,78,getmaxx()-20,97);
    bar(getmaxx()-100,110,getmaxx()-20,129);
    bar(getmaxx()-100,142,getmaxx()-20,161);
    bar(getmaxx()-100,174,getmaxx()-20,193);
    bar(getmaxx()-100,206,getmaxx()-20,225);
    bar(getmaxx()-100,238,getmaxx()-20,257);
    bar(getmaxx()-100,270,getmaxx()-20,289);
    bar(getmaxx()-100,302,getmaxx()-20,321);

    rectangle(80,getmaxy()-180,90,getmaxy()-170);
    rectangle(80,getmaxy()-160,90,getmaxy()-150);
    rectangle(210,getmaxy()-180,220,getmaxy()-170);
    rectangle(210,getmaxy()-160,220,getmaxy()-150);

    rectangle(80,getmaxy()-120,90,getmaxy()-110);
    rectangle(80,getmaxy()-100,90,getmaxy()-90);
    rectangle(80,getmaxy()-80,90,getmaxy()-70);
    rectangle(210,getmaxy()-120,220,getmaxy()-110);
    rectangle(210,getmaxy()-100,220,getmaxy()-90);
    rectangle(210,getmaxy()-80,220,getmaxy()-70);

    drawblock2(getmaxx()-200,getmaxy()-80,getmaxx()-
250,getmaxy()-100,0);
    rectangle(getmaxx()-200,getmaxy()-80,getmaxx()-
250,getmaxy()-100);
```

```
drawblock2(getmaxx()-100,getmaxy()-80,getmaxx()-
150,getmaxy()-100,0);
    rectangle(getmaxx()-100,getmaxy()-80,getmaxx()-
150,getmaxy()-100);
    setcolor(BLUE);
    outtextxy(getmaxx()-148,getmaxy()-95,"cancel");
    outtextxy(getmaxx()-230,getmaxy()-95,"OK");

    outtextxy(getmaxx()-200,83,"filename");
    outtextxy(getmaxx()-200,115,"      Ap");
    outtextxy(getmaxx()-200,147,"      As");
    outtextxy(getmaxx()-200,179,"     fp1");
    outtextxy(getmaxx()-200,211,"     fs1");
    outtextxy(getmaxx()-200,243,"     fp2");
    outtextxy(getmaxx()-200,275,"     fs2");
    outtextxy(getmaxx()-200,307,"      F ");

    outtextxy(110,getmaxy()-180,"lowpass");
    outtextxy(110,getmaxy()-160,"highpass");
    outtextxy(110,getmaxy()-120,"rectangle");
    outtextxy(110,getmaxy()-100,"bartlett");
    outtextxy(110,getmaxy()-80,"hanning");

    outtextxy(240,getmaxy()-180,"bandpass");
    outtextxy(240,getmaxy()-160,"bandstop");
    outtextxy(240,getmaxy()-120,"hamming");
    outtextxy(240,getmaxy()-100,"backman");
    outtextxy(240,getmaxy()-80,"kaiser");

    outtextxy(25,getmaxy()-180,"Filter");
    outtextxy(25,getmaxy()-120,"Window");
     setfillstyle(1,LIGHTGRAY);

    selec = 0;
    selec1= 0;
    switch(ck)
    {
      case 0:bar(80,getmaxy()-180,90,getmaxy()-
170);graphlp();break;
      case 1:bar(80,getmaxy()-160,90,getmaxy()-
150);graphhp();break;
      case 2:bar(210,getmaxy()-180,220,getmaxy()-
170);graphbp();break;
      case 3:bar(210,getmaxy()-160,220,getmaxy()-
150);graphbs();break;
    }
    switch(ck1)
    {
    case 0: bar(80,getmaxy()-120,90,getmaxy()-
110);break;
    case 1: bar(80,getmaxy()-100,90,getmaxy()-
90);break;
    case 2: bar(80,getmaxy()-80,90,getmaxy()-70);break;
    case 3: bar(210,getmaxy()-120,220,getmaxy()-
110);break;
    case 4: bar(210,getmaxy()-100,220,getmaxy()-
90);break;
    case 5: bar(210,getmaxy()-80,220,getmaxy()-
70);break;
    }

}
void menurun(void)
{
   int x,y;
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(BLACK);
    setcolor(LIGHTGRAY);
    drawblock1(10,10,getmaxx()-10,getmaxy()-40,30);
    drawblock1(10,getmaxy()-40,getmaxx()-
10,getmaxy()-10,0);
```

```
    setfillstyle(1,1);
    bar(40,14,getmaxx()-14,41);
    bar(15+30/4,18,10+30*3/4,36);
    bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
    settextstyle(0,0,1);
    setcolor(WHITE);
    outtextxy(150,24,"   EXECUTE PROGRAM &
RUN DSP CARD");
    outtextxy(45,getmaxy()-30,"     enter filter name
and enter to choice  (Cancel to exit)");
}
void menuplot(void)
{
   int x,y;
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(BLACK);
    setcolor(LIGHTGRAY);
    drawblock1(10,10,getmaxx()-10,getmaxy()-40,30);
    drawblock1(10,getmaxy()-40,getmaxx()-
10,getmaxy()-10,0);
    setfillstyle(1,1);
    bar(40,14,getmaxx()-14,41);
    bar(15+30/4,18,10+30*3/4,36);
    bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
    settextstyle(0,0,1);
    setcolor(WHITE);
    outtextxy(150,24,"   PLOT GRAPH FREQUENCY
RESPONSE");
    outtextxy(45,getmaxy()-30,"     enter filter name
and enter to choince  (Cancel to exit)");
   drawblock2(80,120,500,300,0);
   rectangle(80,120,500,300);
   setfillstyle(1,BLACK);
   bar(getmaxx()-500,78,getmaxx()-420,97);
   drawblock2(getmaxx()-400,getmaxy()-80,getmaxx()-
450,getmaxy()-100,0);
   rectangle(getmaxx()-400,getmaxy()-80,getmaxx()-
450,getmaxy()-100);
   drawblock2(getmaxx()-200,getmaxy()-80,getmaxx()-
250,getmaxy()-100,0);
   rectangle(getmaxx()-200,getmaxy()-80,getmaxx()-
250,getmaxy()-100);
   setcolor(BLUE);
   outtextxy(getmaxx()-248,getmaxy()-95,"cancel");
   outtextxy(getmaxx()-430,getmaxy()-95,"OK");
   outtextxy(getmaxx()-600,83,"filename");
   bar(85,125,495,295);
   outtextxy(90,135,"      Plot graph by enter
filterfile ");
   outtextxy(90,145,"          and selec plot type");
   outtextxy(90,280,"          Escape to exit");

   setfillstyle(1,WHITE);
   setcolor(BLACK);
   bar(90,255,170,270);
   bar(240,255,320,270);
   bar(390,255,470,270);
   outtextxy(95,260,"magnitude");
   outtextxy(245,260," phase");
   outtextxy(395,260," impulse");
}
```

```
*********************************************
* PROGRAM CONTROL TMS320C50 ON DSK
BOARD                                    *
*   by santipap ukot electronics engineer kmit'l    *
*********************************************
          .mmregs
BITLEN2     .set   0          ; use for calculate
bitrate
BITLEN      .set   1          ;
TA          .set   4          ; fs = 10.368
Mhz/2*TA*TB
RA          .set   4          ;    = 32.4 Mhz
TB          .set   40         ;
RB          .set   40         ;
AIC_CMD     .set   00h        ; control register
TLC32040
TEMP        .set   2          ; GAIN = 4 ,Auxin
and Loopback no use
TEMP1       .set   3          ;    Synch mode
Xn          .set   1000h      ; X(n--->data memory
B1 (0300H)
Yn          .set   4          ; store yn
COMMAND     .set   5          ; store command
ESC         .set   01bh       ; escape to pc
coeff       .set   0e00n      ; coeff --
>programmemory address 0e00h
length      .set   10         ; store N
LastXn      .set   12         ; LastXn = Xn +
length
startadd    .set   11
;-------------------------------------------------------------;
;       interrupt vectors                                     ;
;-------------------------------------------------------------;
          .ps 0800h
     b   start          ;reset
     b   int1sub        ;02; vectors for interrupt int1
          .ps 0806h
     rete               ;06; vectors for interrupt int3
     nop
     rete               ;08; timer interrupt vector - TINT
     nop
     rete               ;0A; Serial port receive interrupt
     nop
     rete               ;0C; Serial port transmit interrupt
     nop
;-------------------------------------------------------------;
; TMS32C05X INITIALIZATION                                    ;
; This routine initializes the C5x registers, internal RAM
and ;
; external RAM from xxxx to FFFF                              ;
;-------------------------------------------------------------;
          .ps   0a00h
          .entry
start     setc INTM
          ldp   #0          ; Set data page pointer
          splk  #830h,PMST  ; 9K on-chip RAM
as Data, No ROM
          lacl  #0          ; Set Wait State
Control Regester
          samm  CWSR        ; for 0 waits in
pgm & data memory
          samm  PDWSR       ;
;-------------------- ----------------------------------;
; initalize and reset serial port                             ;
;-------------------------------------------------------------;
          splk  #20h,TCR
          splk  #1,PRD
          mar   *,AR0
     lacl   #08h ; set FSM bit for FSX/FSR per frame
     samm   spc  ; Configure for 16 bit mode with
     lacl   #0C8h  ; external CLKX, reset tx and rx
     samm   spc
```

```
        lamm    DRR     ; clear first int
        lacc    #0080h
        sach    DXR     ; clear first int
      sacl    GREG ; Pulse AIC reset by setting it low
        lar     AR0,#0FFFFFh
        rpt     #10000          ; and taking it high
after 1000 cycles
        lacc    *,0,AR0         ; (.5ms at 50ns)
        sach    GREG
        setc    SXM
        setc    OVM
        lar     AR7,#0          ; Buffer initialy filled
        call    AIC_SET         ; DO NOT
CHANGE DP WITHOUT RESTORING IT!
        lacl    #10h            ; RINT
        samm    IMR             ; INTERRUPT
INT2,XINT,RINT ENABLE
        mar     *,AR5   ; ar5 use in this program
        ldp     #19             ; DATAPAGE = 19
addrass  098h - 09ffh
;-------------------------------------------------------------;
;       set baudrate 9600b/sec                                ;
;-------------------------------------------------------------;
        lacc    #0823h
        sacl BITLEN
        lacc  #022h
        sacl BITLEN2                    ;cal baudrate
        lacl #08
        sacl COMMAND
        setc INTM                       ;disanable interrupt
begin   bcnd fadata,bio
        b   begin
fadata  call reads
        sacl TEMP
        lacl #ESC
        call xmtbyte
        lacl TEMP
        sub #09h
        bcnd main,EQ
        b begin
;-------------------------------------------------------------;
;     main program test command                               ;
;-------------------------------------------------------------;
main    idle
        lamm DRR .
        sacl TEMP1
        bcnd loopget,bio
        lacl COMMAND
        and #00ffh
        sub #05
        bcnd convol,EQ
        lacl COMMAND
        and #00ffh
        sub #07
        bcnd testcl1,EQ
        b main
;-------------------------------------------------------------;
;          convolution sum                                    ;
;                N-1                                          ;
;               -------                                       ;
;                .                                            ;
;        y(n) =          .  h(k)*x(n-k)                       ;
;                .                                            ;
;               -------                                       ;
;                k = 0                                        ;
;-------------------------------------------------------------;
convol:         lacl TEMP1
                mar *,AR1
                lar AR1,#Xn
                sacl *                  ;
                lar AR1,LastXn      ; AR1  --> X(n-(n-1))
                zap                 ; ACC = 0 = P
```

```
        rpt length          ; for k=0, k<N-1 , k++
        macd coeff,*-       ; coeff(k) * X(n-k)
        apac                ;        N
        sach Yn             ;       -----
        lacc Yn,2           ;        .
        samm DXR    ; Y(n) =     . h(k)*X(n-k)
        b main
;-------------------------------------------------------------;
testcl1 lacl TEMP1
        sacl  Yn
        lacc Yn
        bsar 3
        sacl Yn
        lacc Yn,2
        samm DXR
        b main
;-------------------------------------------------------------;
:    getdata from pc to tms                 ;
;-------------------------------------------------------------;
loopget  bcnd S,bio
pget1    bcnd pget,bio
        b pget1
pget     call reads
        sacl COMMAND
        add  #2
        call xmtbyte
        call testcommand
        b main
;-------------------------------------------------------------;
:    int1 interrupt subroutine              ;
;-------------------------------------------------------------;
int1sub       bcnd int1sub1,bio
             b int1sub
int1sub1       call reads
             sacl COMMAND
             lacl #ESC
             call xmtbyte
             rete
;-------------------------------------------------------------;
;   test command subroutine                 ;
;-------------------------------------------------------------;
testcommand    mar *,ar0      ; ar0 use this subprogram
             lacc COMMAND
             lar ar0,COMMAND       ; ar0 = command
             banz test0,*-   ;test command =00 (not uses)

test0         banz test1,*-
             b tatbsub              ;test command =01
(ta tb)
test1         banz test2,*-
             b download              ;test command =02
(N,coeff)
test2         banz test3,*-
             ret
test3         banz test4,*-
             ret
test4         banz test5,*-
             b runfir               ;test command =05
(run convolution)
test5         banz test6,*-
             ret
test6         banz test7,*-
             b stoptms              ;test command =07
(wait)
test7         banz test8,*-
             b resettms             ;test command =08
(reset tms);
test8         b resettms
;-------------------------------------------------------------;
:       TA,TB set to TLC320c40
;-------------------------------------------------------------;
tatbsub   call getdata
```

```
        and #00ffffh
        sacl LastXn
        lacl #07
        sacl COMMAND
        ret
        call getdata
        sacl TA
        sacl RA                 ;store ta=ra
        call getdata
        sacl TB
        sacl RB                 ;store tb=rb
        call AIC_SET            ;set tlc320c40
        lacl #ESC
        call xmtbyte            ;send ESC to syncho
        ret
;-------------------------------------------------------;
;       command execute convolution program             ;
;-------------------------------------------------------;
runfir  lacl #05h
        sacl COMMAND
        ret
;-------------------------------------------------------;
;       command wait                                    ;
;-------------------------------------------------------;
stoptms lacl #07h
        sacl COMMAND
        ret
;-------------------------------------------------------;
;       command reset programm branch to start          ;
;-------------------------------------------------------;
resettms call getdata
        and #00ffffh
        sacl length
        lacl #07
        sacl COMMAND
        ret
;-------------------------------------------------------;
;           getdata : receice data 16bit                ;
;-------------------------------------------------------;
getdata  call reads             ;read first data (8 bit)
        sacl TEMP1              ;store TEMP1
        lacl TEMP1
        call xmtbyte            ;send back to pc
        call reads              ;read second data (8
bit)
        add  TEMP1,8            ;data =first data +
second data shift
        sacl TEMP1              ;store TEMP1
        lacl TEMP1
        call xmtbyte            ;send back to pc
        lacc TEMP1              ;lacc = data
        ret
;-------------------------------------------------------;
;           getdata1 : receice data 16bit               ;
;-------------------------------------------------------;
joyshow1  call reads            ;read first data (8
bit)
        lacl #ESC
        call xmtbyte            ;send back to pc
joy1    bcnd joy2,bio
        b    joy1
joy2    call reads              ;read second data (8
bit)
        lacl #ESC
        call xmtbyte            ;send back to pc
        ret
;-------------------------------------------------------;
;           getdata1 : receice data 16bit               ;
;-------------------------------------------------------;
joyshow2  call reads            ;read first data (8
bit)
        lacl #2
```

```
        call xmtbyte            ;send back to pc
joy12   bcnd joy22,bio
        b   joy12
joy22   call reads              ;read second data (8
bit)
        lacl #3
        call xmtbyte            ;send back to pc
        ret
;-------------------------------------------------------------;
;       getdata1 : receice data 16bit                         ;
;-------------------------------------------------------------;
joyshow3  call reads            ;read first data (8 bit)
        lacl #4
        call xmtbyte            ;send back to pc
joy13   bcnd joy23,bio
        b   joy13
joy23   call reads              ;read second data (8 bit)
        lacl #5
        call xmtbyte            ;send back to pc
        ret
;-------------------------------------------------------------;
;       getdata1 : receice data 16bit                         ;
;-------------------------------------------------------------;
joyshow4  call reads            ;read first data (8 bit)
        lacl #6
        call xmtbyte            ;send back to pc
joy14   bcnd joy24,bio
        b   joy14
joy24   call reads              ;read second data (8 bit)
        lacl #7
        call xmtbyte            ;send back to pc
        ret
;-------------------------------------------------------------;
;       getaddress (start address and length)                 ;
;-------------------------------------------------------------;
address  call getdata           ;read data 16 bit
        lacc #0e00h
        sacl startadd           ;store startaddress
        lar  ar6,startadd       ;ar6 = start address
        call  getdata           ;read data 16 bit
        sub #1
        sacl  length            ;store length
        lar  ar7,length         ;ar7 = length
        mar  *,ar6              ;modified ar6
        ret
;-------------------------------------------------------------;
;     download program memory(pc to tms320c50)                ;
;-------------------------------------------------------------;
download call address           ;get start address
loopd   call getdata            ;get data 16 bit
        lacc startadd           ;lacc =start address
        tblw TEMP1              ;write data from
TEMP1 to (lacc)
        add #1                  ;address++
        sacl startadd           ;startaddress ++
        mar *,ar7               ;arp = ar7
        banz loopd,*-           ;branch if ar7 !=
0(length=0)
        lacc #1001h
        add length
        sacl LastXn
        lacl #05
        sacl COMMAND
        ret
;-------------------------------------------------------------;
; READS: read serial mode
;-------------------------------------------------------------
reads   lar   ar5,#7
        lacl  #0
reads1  Bcndd  STOK,bio   ;wait for start bit
        lar   AR5,#7
        lacl  #0
```

```
        B       reads1          ;
STOK    rpt     BITLEN2         ;BITLEN is scaled and
        nop
        mar     *,AR5           ;number of bits - 1
WTBIT   sfr
        rpt     BITLEN          ;decremented by 8/3 for
        nop                     ;BITLEN/2 wait
        bcnd    ZEROBT,bio
        add     #80h
ZEROBT  BANZ    WTBIT,*-        ;last bit ?
        RET                     ;ACC = read value
;---------------------------------------------------------------
;  xmtbyte to PC
;---------------------------------------------------------------
xmtbyte  clrc c                 ; startbit=0
        lar  ar5,#8             ; counter: 1 startbit+ 8 databits
(+ 2 stopbits)
nextbit1 bcnd snd0,nc           ; if c=1 send 1 else send 0
snd1     setc xf                ; send one
        b    snd
snd0     clrc xf
snd      rpt  BITLEN            ; send one bit
        mar  *,ar5
        ror                     ; lsb(accu) -> carrybit
        banz nextbit1,*-        ; repeat for entire word (10
bits);
        setc xf
        rpt  BITLEN
        nop
        rpt  BITLEN
        nop
        ret
;-------------------------------------------------------------;
;          Set TLC320c40                                      ;
;-------------------------------------------------------------;
AIC_SET:        lacl    #020h
        samm    IMR             ;XMIT interrupt
        lacc    #AIC_CMD,2
        add     #03h
        call    AIC_2nd
        ;----------------------
        lacc    #TB,9
        add     #RB,2
        add     #02h
        call    AIC_2nd
        ;----------------------
        lacc    #TA,9
        add     #RA,2
        call    AIC_2nd
        ;----------------------
        ret
AIC_2nd:        sach    DXR
        clrc    INTM
        idle
        add     #6,15
        sach    DXR
        idle            ;ACCU_hi requests 2nd XMIT
        samm    DXR
        idle            ;ACCU_lo sets up registers
        lacl    #0
        samm    DXR     ;make sure the word got sen
        idle
        setc    INTM
        ret
;-------------------------------------------------------------;
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้เป็นอย่างดี เนื่องด้วยการได้รับความช่วยเหลือเป็นอย่างดียิ่งจาก อ. เทอดศักดิ์ ลิ่วหาทอง ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ที่คอยให้คำแนะนำ ให้คำปรึกษา และเอาใจใส่ในการทำโครงงานตลอดมา ตลอดจนสนับสนุนในด้านเอกสารอ้างอิง และอุปกรณ์ในการทดลอง และขอบคุณสำหรับผู้ร่วมงานและเพื่อนๆทุกคนที่ช่วยทำ และให้กำลังใจทำให้โครงงานนี้สำเร็จลุล่วงได้ด้วยดี

หนังสืออ้างอิง

1. Roman Kuc, "Introduction to DIGITAL SIGNAL PROCESSING", McGRAW-HILL INTERNATIONAL EDITIONS, 1982
2. Texas INSTRUMENTS, "TMS320C5X User's Guide" , Texas INSTRUMENTS, 1993
3. Texas INSTRUMENTS, "TMS320C5X DSP Starter Kit User's Guide", Texas INSTRUMENTS,1994
4. ดร. ไพรัช ธัชยพงษ์, "การประมวลสัญญาณดิจิตอล ตอน การออกแบบวงจรกรองดิจิตอล (Digital Signal Processing : Digital Filter Desing)", บริษัท เอดิสันเพรส โปรดักส์ จำกัด, 2535

รายงานความก้าวหน้าโครงงานครั้งที่ 1

ชื่อโครงงาน

ชุดทดลองการประมวลผลสัญญาณเชิงเลข (DSP)

รายชื่อผู้ร่วมโครงงาน

1.นายธันติภาพ	อุโคตร	รหัสประจำตัว	35.104464
2.นางสาววิรุฬรยา	ชุดธน	รหัสประจำตัว	35.104546
3.นางสาวอลงกต	ใจวังโลก	รหัสประจำตัว	35.104547

อาจารย์ที่ปรึกษา

อ. เทอดศักดิ์ ลิ่วหาทอง

สิ่งที่ได้ดำเนินการไปแล้วและหลักการทำงาน

เนื่องจากโครงงานในตอนที่ 2 สืบเนื่องต่อมาจากโครงงานในเทอมที่1 ซึ่งเป็นเรื่องดิจิตอลซิกแนลโปรเซสซิ่ง (ดิจิตอล ฟิลเตอร์) จากเทอมที่แล้วได้ดำเนินงานมาถึงการศึกษาทั้งชิพ เบอร์ TMS320C50 , TLC320C40 และบอร์ด DSK (DSP Starter Kit) ตลอดจนการเขียนโปรแกรม เบื้องต้นในการประมวลผลซิ่งเน้นทางฟิลเตอร์ (FIR filter)

สิ่งที่คาดว่าจะทำในเทอมที่ 2

1.เขียนโปรแกรม FIR filter ทั้งบน TMS320C50 และโปรแกรมภาษา C คำนวณ

2.เขียนโปรแกรม IIR filter ทั้งบน TMS320C50 และโปรแกรมภาษา C คำนวณ

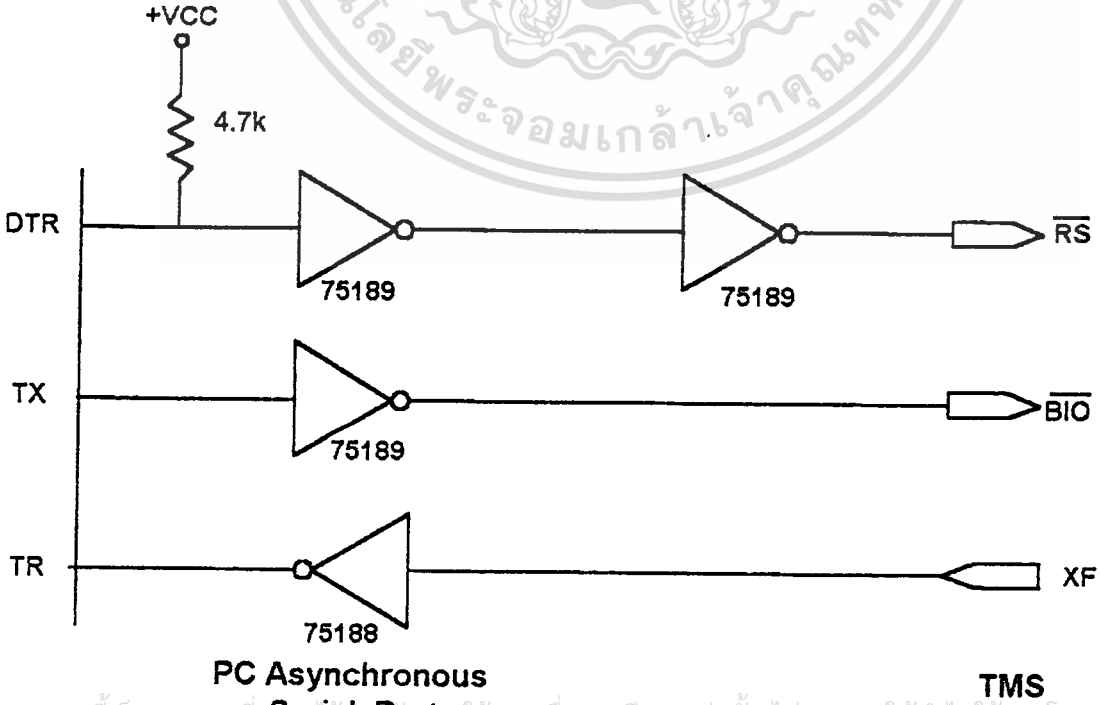
3.เขียนโปรแกรมการติดต่อระหว่าง PC และ DSK

4.เขียนโปรแกรมแสดงผลทาง PC ซึ่งใช้ภาษา C เขียน เพื่อความสะดวกในการใช้งาน

5.เขียนเนื้อหาและรายงานผลทั้งหมด

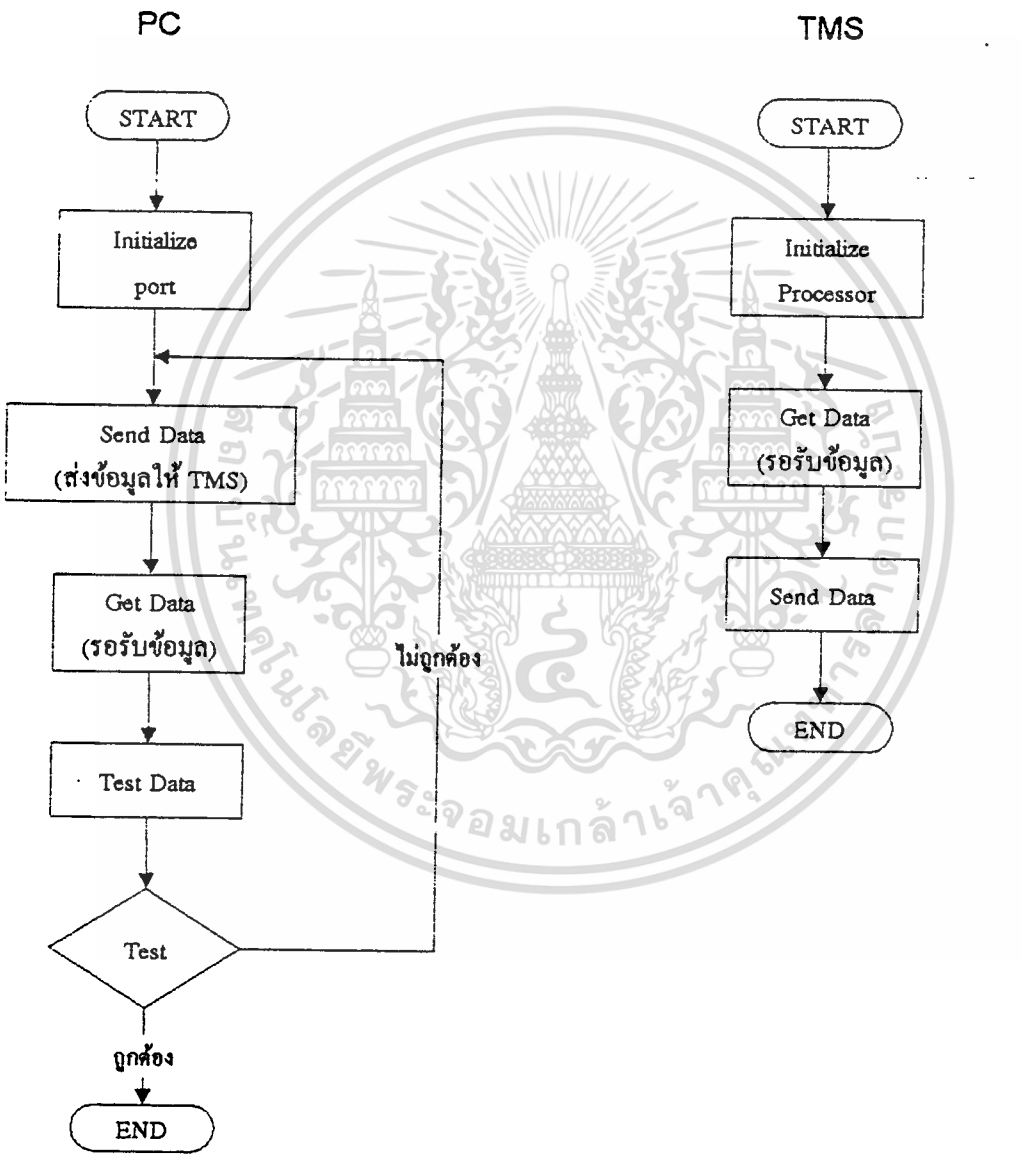
สิ่งที่ได้ดำเนินการไปแล้ว

ขณะนี้ได้เขียนโปรแกรมการติดต่อระหว่าง PC และ DSKซึ่งผลของโปรแกรมจะทำให้การติดต่อ เพื่อส่งข้อมูลระหว่าง PC และ DSK เป็นไปได้สะดวกโดยผ่านทาง Serial port (COM 2) ส่วน TMS จะรับทางขาของ BIO และ XF โดยที่ขาสองขานี้เป็น General Purpose I/O Pins ซึ่งต่อดังนี้



Flowchartการทำงาน

ในการทำอะไรจะหลักการของ Hand shecking กล่าวคือ PC จะส่งข้อมาที่ DSK โดย DSK จะรอรับ ถ้าได้รับแล้ว จะส่งสัญญาณบอก PC ว่าได้รับแล้ว แต่ถ้าข้อมูลนั้นเกิดผิดพลาดจะทำการตรวจสอบ แล้วส่งใหม่อีก จนได้ข้อมูลที่ถูกต้อง ในการทำงานจะต้องให้อัตราบอดเท่ากัน ซึ่งขณะนี้สามารถกำหนดได้ถึง 9600 bit/sec หลักการทำงานสามารถเขียนบน Flowchart ได้เป็น



ในส่วนของ TMS จะทำการคำนวณ Boad rate เองโดยให้ PC ส่งข้อมูลมาเพื่อคำนวณส่วน PC จะกำหนด Boad rate ได้ ในการใช้งานควรจะกำหนด Boad rate ให้คงที่เพื่อให้การส่งจะได้เป็นไปได้อย่างรวดเร็ว

ปัญหาและอุปสรรคในโครงงาน

ในการทำโครงงานขณะนี้เรื่องการติดต่อระหว่าง PC กับ DSK ยิ่งมีปัญหาทางด้าน PC ทำงานช้ากว่า TMS มาก ทำให้บางครั้งเกิดข้อมูลผิดพลาดได้ แต่แก้ปัญหาโดยการทำ Hand shake ส่วนเรื่องการคำนวณของ filter ได้ทดสอบแล้ว บางครั้ง filter ที่ออกแบบมายังใช้งานไม่ได้ โดยได้ทำมาแล้วจากเทอมที่แล้ว โดยดูผลทางด้าน ความถี่ cutoff ต่างๆยังไม่ตรงตามกำหนด โดยที่ผลที่ทำงานได้ดีจะขึ้นอยู่ค่า N ถ้าค่า N มากถึง 200 ขึ้นไปจะ ทำงานได้ผลดี ส่วนต่ำกว่านี้จะไม่ได้ผลที่ดีนัก และค่า N มากกว่า 500 ซึ่งจะทำให้การโปรแกรมช้ามากจนผลที่ออกมามากกว่าความถี่ Sampling ผลที่ได้จึงผิดพลาด ปัญหาเรื่องการใช้ความถี่ Sampling ยังมีเนื่องจาก ชิป TLC320C40 จะมีค่าความถี่ cutoff ของตัวมันเอง การกำหนด Sampling Rate จะต้องขีดจำกัดอยู่ในช่วงหนึ่ง เพราะฉะนั้นในการใช้งานจะต้องทราบและใส่ให้ถูกต้องมิฉะนั้นผลที่ได้จะผิดพลาดไป

ความคิดเห็นของอาจารย์ที่ปรึกษา

.....

(อ.เทอดศักดิ์ สิ่วหาทอง)

รายงานความก้าวหน้าโครงงานครั้งที่ 2

ชื่อโครงงาน

ชุดทดลองการประมวลผลสัญญาณเชิงเลข (DSP)

รายชื่อผู้ร่วมงาน

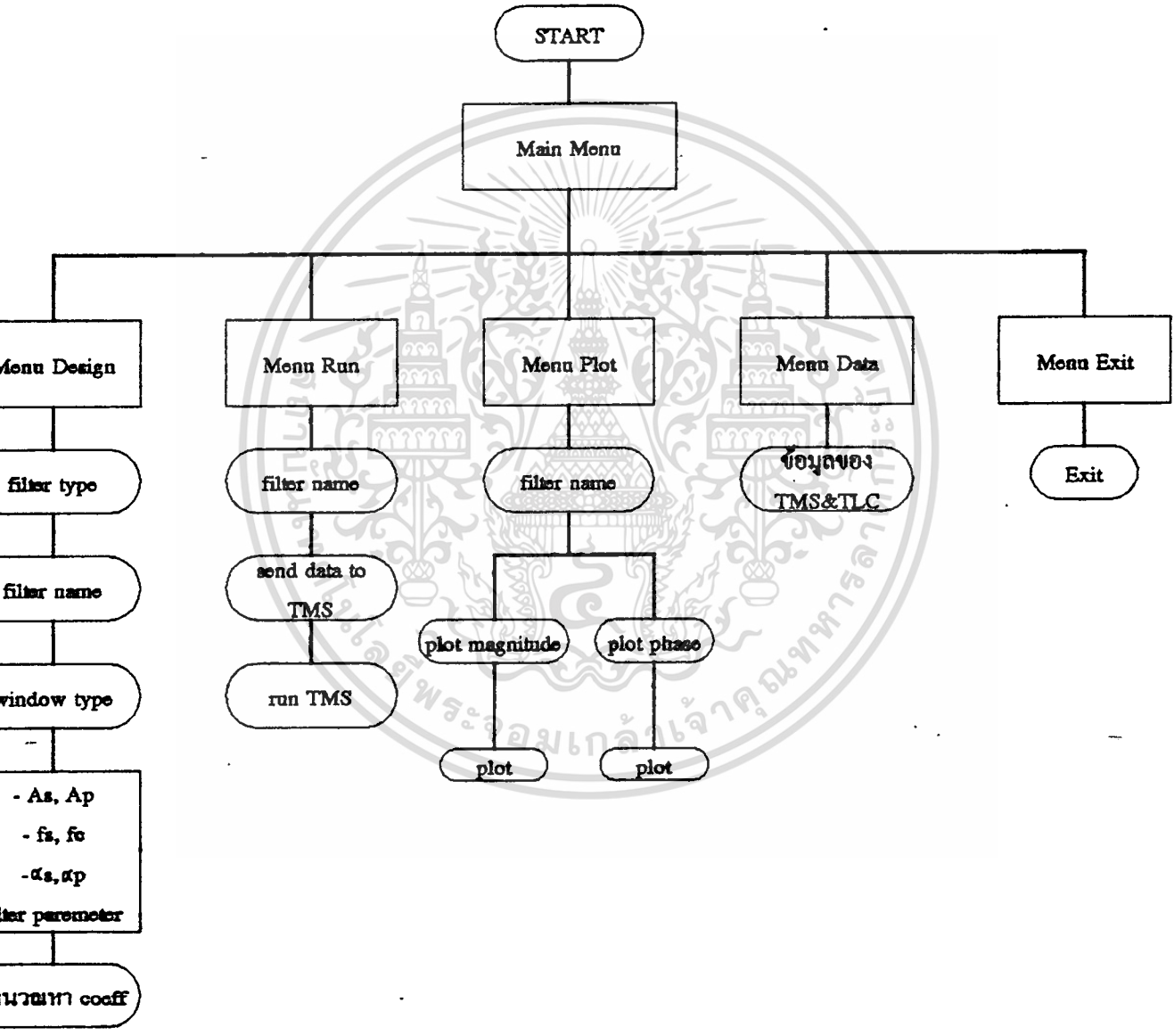
1. นายสันติภาพ อุโคตร รหัสประจำตัว 35104464
2. นางสาววิรุณรยา สุดฉน รหัสประจำตัว 35104546
3. นางสาวอลงกต ใจวังโลก รหัสประจำตัว 35104547

อาจารย์ที่ปรึกษา

อ. เทอดศักดิ์ ลิ่วหาทอง

รายละเอียดเกี่ยวกับโครงงาน และ สิ่งที่ได้ดำเนินการไปแล้ว

ในตอนนี้ได้ศึกษาและเขียนโปรแกรมหลักบน PC โดยจัดโครงสร้างของโปรแกรมตามรูปดังนี้



ในการควบคุมโปรแกรมทั้งหมดจะต้องให้สัมพันธ์กับการเขียนโปรแกรมบน TMS320C50 ซึ่งกำลังดำเนินการอยู่ โปรแกรมหลักบน PC นี้สามารถอธิบายได้เป็นส่วนๆ ดังนี้

-Design จะเป็นส่วนของการออกแบบ filterซึ่งจะมีการป้อนพารามิเตอร์คือAs, Ap, fs, fc, αs, αp ซึ่งจำเป็นสำหรับการหาค่า coefficient ของ filter

-Run เป็นการนำค่า coeff ที่ได้จากการ design ส่งให้ TMS แล้วทำการ run โปรแกรม TMS ด้วย

-Plot จะเป็นการแสดง plot กราฟ ซึ่งจะแบ่งเป็นการ plot magnitude และ plot phase ที่ได้จากการคำนวณ

-Data เพื่อเป็นการใช้งานและทราบข้อมูลของ TMS และรายละเอียดโปรแกรม

-Exit จะเป็นการออกจากโปรแกรม

ปัญหาที่พบและแนวทางการแก้ไข

การใช้โปรแกรมที่สมบูรณ์จะต้องเขียนโปรแกรมบน TMS ให้สอดคล้องก่อนจึงจะสมบูรณ์ แต่โปรแกรมหลักนี้สามารถ run ได้โดยการเว้นส่วนของโปรแกรมที่ต้องติดต่อกับ TMS ไว้ ในการเขียนโปรแกรมหลักเพื่อการใช้งานที่สะดวก และสวยงามจึงใช้เวลาในการเขียนมาก และต้องรอบคอบทุกโปรแกรมที่เขียน

ในส่วนของโปรแกรมที่เขียนบน TMS ได้จัดเป็นส่วนต่างๆไว้ยังไม่ได้รวบรวมเข้าด้วยกันเพื่อให้เสร็จสมบูรณ์

ปัญหาสำคัญที่พบในขณะนี้คือการใช้งานของ filter ซึ่งมีบางกรณีที่ทำให้ filter ไม่สามารถใช้งานได้จริง บางครั้ง filter ที่ออกแบบไม่เป็นไปตามที่เราออกแบบเมื่อใช้งานจริง เพราะฉะนั้น การใช้งานจึงมีขอบเขตที่กำหนดคือ

1. ความถี่ sampling จะต้องอยู่ในช่วงที่กำหนด
2. ค่า N จะต้องมีค่ามากพอ และไม่เกินค่าที่กำหนด
3. ค่า Ap, As จะต้องป้อนให้เหมาะสมเพื่อไม่ให้การคำนวณค่า N มีค่ามากจนโปรแกรมทำงานไม่ได้

ความคิดเห็นของอาจารย์ที่ปรึกษา

(อ. เทอดศักดิ์ ลิ่วหาทอง)

รายงานความก้าวหน้าโครงงานครั้งที่ 3

ชื่อโครงงาน

ชุดทดลองการประมวลผลสัญญาณเชิงเลข (DSP)

รายชื่อผู้ร่วมงาน

1. นายสันติภาพ อุโคตร รหัสประจำตัว 35104464
2. นางสาววิรุณรยา สุดสน รหัส ประจำตัว 35104546
3. นางสาวอลงกต ใจวังโลก รหัสประจำตัว 35104547

อาจารย์ที่ปรึกษา

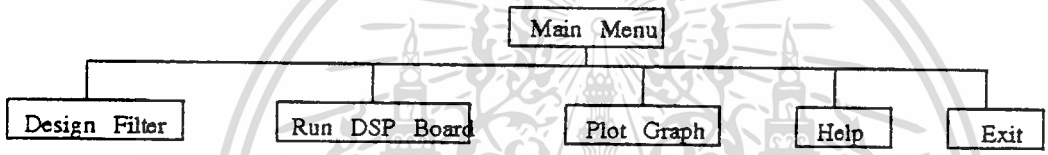
อ. เทอดศักดิ์ ลิ่วหาทอง

รายละเอียดเกี่ยวกับโครงงาน และ สิ่งที่ได้ดำเนินการไปแล้ว

จากรายละเอียดของโครงงาน ซึ่งประกอบด้วยการเขียนโปรแกรมทั้งภาษา C บน PC และภาษาเอสเซมบลีบน TMS320C50 ในส่วนของ TMS320C50 ได้จัดทำขึ้นโดยการเขียนโปรแกรมควบคุมและทำการประมวลผลทางดิจิตอลไปด้วย และที่เชื่อมต่อระหว่าง PC นั้นจะเขียนด้วยภาษา C ซึ่งประกอบด้วย

1. โปรแกรมติดต่อกับ TMS3420C50 ผ่าน RS232
2. โปรแกรมการคำนวณค่า h(n) ของ Filter
3. โปรแกรมควบคุมทั้งหมดซึ่งจะแสดงหน้าจอ

ในส่วนของโปรแกรมติดต่อกับ TMS320C50 และการคำนวณ h(n) ของ Filter ได้จัดทำขึ้นแล้ว โปรแกรมควบคุมทั้งหมดซึ่งจะแสดงหน้าจอกำลังจัดทำโดยมีรายละเอียดดับ Block diagram นี้



ซึ่งแต่ละส่วนสามารถอธิบายโดย

- Design เป็นการออกแบบ Filter โดยการป้อนพารามิเตอร์ที่จำเป็น แล้วทำการออกแบบเก็บไว้บน file
- Run เป็นการทดสอบผลจากการออกแบบ โดยทำการโหลดพารามิเตอร์ต่างๆ ลงไปยังบอร์ด แล้วประมวลผล ซึ่งสามารถดูผลที่ได้ว่าเป็นไปตามที่ออกแบบหรือไม่
- Plot เป็นการ plot กราฟโดยนำค่าที่ออกแบบมา plot
- Help เป็นส่วนที่บอกรายละเอียดของโครงงานและวิธีใช้โปรแกรม
- Exit ออกจากโปรแกรม

ปัญหาที่พบและแนวทางการแก้ไข

1. ปัญหาเรื่องการเขียนโปรแกรมซึ่งส่วนใหญ่จะเป็นการเขียนโปรแกรมโดยภาษา C และต้องใช้รูปแบบต่างๆ มากมาย เช่น การติดต่อกับ TMS320C50 ทาง RS232 , การคำนวณพารามิเตอร์ของ Filter , การแสดงหน้าจอและการควบคุมทั้งหมดต้องรวมกันและประกอบจนเสร็จสมบูรณ์ ทำให้เกิดความล่าช้า และอาจจะต้องตัดบางส่วนออก ทำให้โปรแกรมไม่สมบูรณ์เท่าที่ควร

2. ปัญหาการใช้บอร์ดซึ่งเป็นปัญหาหลักเพราะบางครั้งเวลาที่ออกแบบ Filter บน PC มาแล้วพบว่าสามารถกำหนด cutoff ได้ แต่เมื่อทำการประมวลผลบนบอร์ดแล้วไม่สามารถ cutoff ได้ตามที่ต้องการ ซึ่งปัญหานี้ยังแก้ไขไม่ได้ แต่มีวิธีหลีกเลี่ยงโดย

- จำกัดค่า N ให้มากพอ และไม่เกินที่กำหนด
- การเลือกความถี่ Sampling จะจำกัดหรือกำหนดไว้เลย และการเปลี่ยน Sampling Rate ยังจำกับเฉพาะบางความถี่เท่านั้น

3. การใช้โปรแกรมยังมีบักอยู่มากบางกรณีจะเกิด error ได้ และจะพยายามแก้ไขให้ดีขึ้น

ความคิดเห็นของอาจารย์ที่ปรึกษา

(อ. เทอดศักดิ์ ลิ่วหาทอง)