ดิจิตอลซิกแนลโปรเซสซิ่ง

Digital Signal Processing

ผู้จัดทำ

นาย สันติ แซ่ย่อ 321368

นางสาว สุชาดา ดีชัยยะ 321375

นาย สุทธินันท์ พรศิริกุล 321376

อาจารย์ที่ปรึกษา

รศ. ดร. มนัส สังวรศิลป์

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมศาสตร์อิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา ๒๕๓๕ ภาคเรียนที่ ๒

ปริญญานิพนธ์ปีการศึกษา 2535 (ภาคเรียนที่ 2)

ภาควิชา อิเลคทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การประมวลผลสัญญาณดิจิตอล และการประยุกต์ใช้งาน
(Digital Signal Processing)

ผู้จัดทำ

1 นายสันติ แซ่ย่อ เลขประจำตัว 32.1368

2 นางสาวสุชาดา ดีชัยยะ เลขประจำตัว 32.1375

3 นายสุทธินันท์ พรศิริกุล เลขประจำตัว 32.1376

รศ. ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา

(.....)

ดิจิตอลซิกแนลโปรเซสซิ่ง

(Digital Signal Processing : DSP)

นายสันติ แซ่ย่อ 32.1368

นางสาวสุชาดา ดีชัยยะ 32.1375

นายสุทธินันท์ พรศิริกุล 32.1376

รศ.ดร. มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2535 ภาคเรียนที่ 2

บทคัดย่อ

ระบบประมวลผลสัญญาณเชิงตัวเลขหรือดิจิตอลซิกแนลโปรเซสซิ่งเป็นระบบประมวลสัญญาณอีกแบบหนึ่งที่ได้รับการพัฒนาอย่างมีประสิทธิภาพ และถูกนำไปประยุกต์ใช้งานด้านต่าง ๆ เช่น ระบบควบคุม,ระบบสื่อสารข้อมูล ,เครื่องมือวัดต่าง ๆ เป็นต้น

สำหรับปริญญานิพนธ์ในส่วนของภาคเรียนแรก ทางกลุ่มของผู้จัดทำจึงได้ศึกษาถึงทฤษฎีและได้ออกแบบและพัฒนาฮาร์ดแวร์ของระบบดิจิตอลซิกแนลโปรเซสซิ่ง โดยใช้ไมโครโปรเซสเซอร์ เบอร์ TMS32010 ของ Texas Instrument และติดต่อกับคอมพิวเตอร์ทางบอร์ดอินเตอร์เฟส ซึ่งจะใช้ 8255 เป็นพอร์ทอินพุทและพอร์ทเอาท์พุท ใช้ RAM เบอร์ MT5C2568-20 ขนาด 32Kx8 บิทเป็นหน่วยความจำของระบบ และในส่วนของภาคเรียนที่ 2 ทางกลุ่มได้พัฒนาโปรแกรมภาษาแอสเซมบลี้ของ TMS32010 เพื่อนำไปใช้งานรวมกับระบบฮาร์ดแวร์ที่ได้ออกแบบและพัฒนาขึ้นในภาคเรียนแรก เพื่อนำไปประยุกต์ใช้งานเป็นวงจรตัวกรองเชิงเลข ซึ่งสามารถใช้งานเป็นวงจรกรองเชิงเลขได้ทุกชนิด นอกจากนั้นทางกลุ่มผู้จัดทำยังได้พัฒนาโปรแกรมขึ้น เพื่อให้ผู้ใช้สามารถออกแบบวงจรกรองเชิงเลขได้ทางไมโครซอฟต์วินโดว์ได้ ซึ่งในส่วนนี้จะทำให้ผู้ใช้มีความสะดวกในการออกแบบวงจรกรองให้มีคุณสมบัติตามที่ผู้ใช้ต้องการ.

Digital Signal Processing

SANTI SAEYOR 321368

SUCHADA DEECHAIYA 321375

SUTTINUN PORNSIRIKUL 321376

Dr.MANUS SANGVORASILP Advisor.

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 , Telecommunications , Instrumentations , Image Processing , etc.

In the first semester, our project group had studied the theories of DSP system and designed the circuit which will support our study. In our circuits, we use TMS32010 as microprocessor in mainboard and interface this board with PC by interfacing card used 8255 peripheral. RAM number MT5C2568-20 which has 32Kx8 bits is used as system memory.

In this semester, our group use the assembly language of TMS32010 for controling the hardware in first semester in order to apply for any types of digital filter and we additionally develop the software for the user to design the digital filter by using Microsoft Windows Version 3.0 up.

By this software, the user will design the digital filter to have the required specification easily and quickly.

สารบัญ

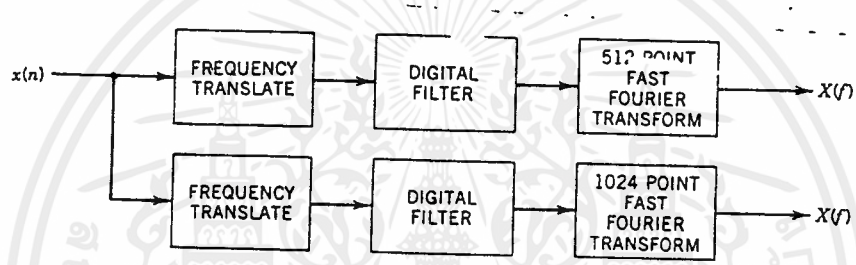
สารบัญ (ต่อ)

บทที่ 1 บทนำ

ดิจิตอลซิกแนลโปรเซสซิง ก็คือการเปลี่ยนรูปแบบของสัญญาณ โดยใช้ การคำนวณทางคณิตศาสตร์ ด้วยดิจิตอลซิกแนลโปรเซสเซอร์ ซึ่งจะเห็นได้ว่าเราสามารถจะนำสัญญาณนั้นใส่ลงไปในสมการคณิตศาสตร์ แล้วได้ผลลัพธ์ออกมาโดยตรงเลยทีเดียว

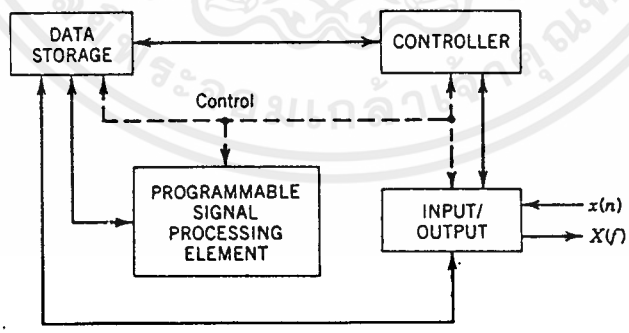
ในช่วง 10 ปีที่ผ่านมา ความก้าวหน้าทางด้านดิจิตอลซิกแนลโปรเซสซิง ได้เกิดขึ้นเป็นอย่างมาก มีการค้นพบอัลกอริทึม (Algorithms) ใหม่ๆเพิ่มขึ้นเรื่อยๆ และการพัฒนาเครื่องมือใหม่ๆ เพื่อช่วยในด้านการออกแบบ ตาม อัลกอริทึมนั้นอีกด้วย ระบบดิจิตอลซิกแนลโปรเซสซิงนี้จะมีอยู่ 2 แบบใหญ่ๆคือ

1.1 <u>ระบบที่ออกแบบมาเพื่อให้ทำงานอย่างใดอย่างหนึ่งโดยเฉพาะ</u> เช่น รูปที่ 1.1



รูปที่ 1.1 แสดงการประมวลผลเฉพาะอย่าง

1.2 <u>ระบบที่ใช้ไมโครโปรเซสเซอร์</u> ซึ่งจะสามารถทำงานเป็นอะไรก็ได้ ขึ้นอยู่กับโปรแกรมที่ใส่ไว้ ดังรูป 1.2



รูปที่ 1.2 แสดงการใช้ไมโครโปรเซสเซอร์ประมวลผล

ความก้าวหน้าทางอิเลคทรอนิส์ครั้งใหญ่ เกิดขึ้นเมื่อได้มีการพัฒนาสร้าง ดิจิตอลซิกแนลโปรเซสเซอร์แบบความเร็วสูงขึ้น (High-Speed Signal Processor) ซึ่งในปัจจุบันนี้ได้มีใช้กันอย่างแพร่หลาย ในราคาที่ไม่แพงนัก

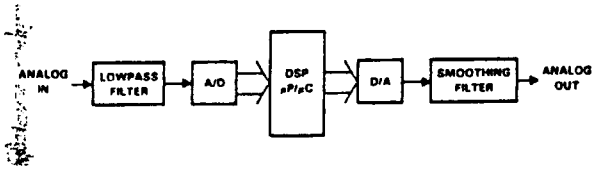
ดิจิตอลซิกแนลโปรเซสเซอร์ จำเป็นจะต้องมีความเร็วที่สูง และถูกออกแบบมาโดยเฉพาะให้สามารถทำงานได้อย่างมีประสิทธิภาพ โดยใช้ข้อดีของการประมวลผลแบบขนาน และการใช้ชุดคำสั่งที่สร้างขึ้นมาเฉพาะ ทั้งนี้ก็เพื่อการประมวลผลแบบเวลาจริง (Real Time) นั่นเอง การประมวลผลแบบเวลาจริงคือ เมื่อเราใส่สัญญาณอินพุทเข้าไป ก็จะได้สัญญาณเอาท์พุทออกมาเลย โดยจะมีเวลาหน่วง (delay) น้อยมาก ซึ่งจะมีประโยชน์ในการใช้งานมากกว่าการประมวลผลแบบไม่ใช่เวลาจริง

ด้วยความสามารถของดิจิตอลซิกแนลโปรเซสซิง จึงได้นำไปใช้งานในแบบต่างๆ ซึ่งให้ผลลัพธ์ที่ดีกว่าใช้วงจรทางอนาลอก เช่น ความน่าเชื่อถือ ความยืดหยุ่น การทดแทนได้ง่าย ขนาดที่เล็กกว่า และเสถียรภาพในระยะยาว และความสามารถของดิจิตอลซิกแนลโปรเซสเซอร์ที่สามารถเปลี่ยนโปรแกรมการทำงานได้ง่าย ทำให้การใช้งานดิจิตอลซิกแนลโปรเซสเซอร์นั้นน่าสนใจ เช่น สามารถเปลี่ยนให้ดีขึ้นเมื่ออัลกอริทึมใหม่ๆถูกค้นพบ หรือความสามารถทำงานได้หลายๆอย่าง โดยเพียงแค่เปลี่ยนโปรแกรมให้กับมันเท่านั้น

ในทางอิเลคทรอนิคส์นั้น โดยส่วนใหญ่มักจะเกี่ยวข้องกับสัญญาณทางไฟฟ้า เช่น การกรองสิ่งที่ไม่ต้องการออกจากสัญญาณอินพุท การแยกเอาข้อมูลออกมา หรือไม่ก็เป็นการสร้างรูปคลื่นที่ต้องการขึ้นมา หรือการเปลี่ยนลักษณะทางแอมปลิจูดของสัญญาณ ซึ่งในอดีตการกระทำเหล่านี้จะทำโดยใช้วงจรอนาลอกที่ออกแบบมาเพื่องานนั้นๆ นอกจากนั้นวงจรทางอนาลอกยังไม่เหมาะที่จะสร้างขึ้นเป็นวงจรรวมขนาดใหญ่ (VLSI) อีกด้วย คุณสมบัติของวงจรอนาลอกนั้นจะขึ้นอยู่กับค่าอุปกรณ์ที่ใช้ เช่น ค่า R,C ที่มีค่าไม่แน่นอน ที่สำคัญจะมีกระแสรั่วไหลที่แปรตามเวลา และยังขึ้นอยู่กับ ไวลท์เตจและอุณหภูมิอีกด้วย การออกแบบวงจรทางอนาลอกนั้นก็ยังน่าเบื่อ ทั้งวงจรหนึ่งๆ จะทำงานได้อย่างเดียวเท่านั้น แต่ถ้าต้องการให้ทำงานอื่นๆ ก็จะต้องเปลี่ยนอุปกรณ์ และก็ต้องเดินสายกันใหม่ นอกจากนี้แล้วการทดสอบวงจรอนาลอกอาจเป็นปัญหาได้ เนื่องมาจาก อิมพีแดนซ์ของเครื่องมือวัดที่ต่อเข้าไปในวงจร ปัญหาพื้นฐานของวงจรอนาลอกเหล่านี้ สามารถแก้ได้โดยการใช้ DSP ข้อได้เปรียบอีกประการของการใช้ DSP แทนวงจรอนาลอกคือ การเปลี่ยนคุณสมบัติของการประมวลผลสามารถทำได้โดยง่าย ซึ่งจะทำให้เราสามารถทดลอง ในทุกทางที่

เป็นไปได้ เพื่อเลือกเอาผลที่ดีที่สุดที่เราพอใจ

โดยทั่วไป ระบบ DSP จะประกอบด้วยส่วนต่างๆดังรูป 1.3

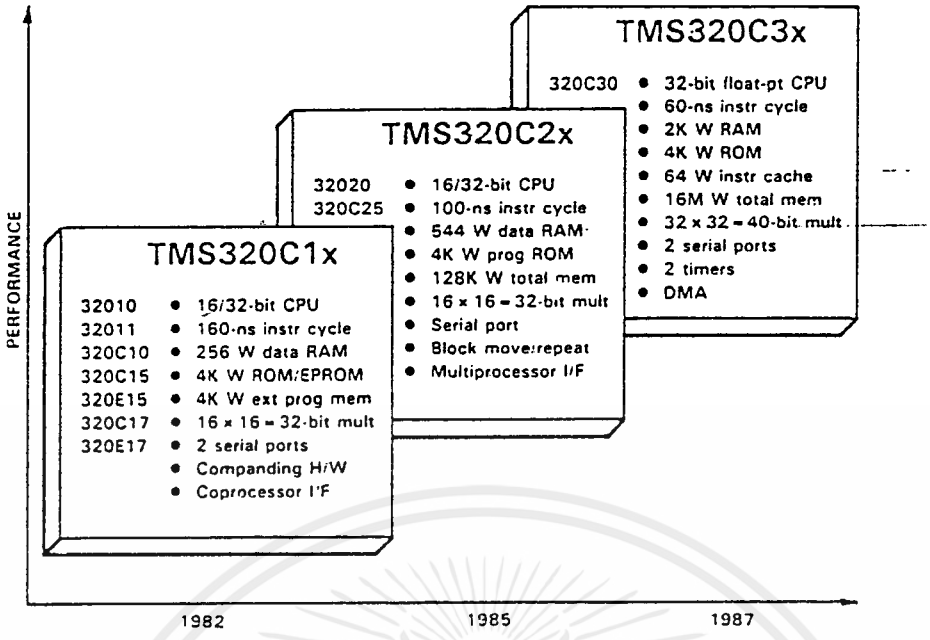


รูปที่ 1.3 แสดงส่วนประกอบของ DSP

สัญญาณอนาลอกอินพุท จะถูกป้อนเข้าสู่ Low Pass Filter เพื่อที่จะจำกัดช่วงความถี่ของสัญญาณอินพุท จากนั้นก็จะส่งต่อไปยัง อนาลอก/ดิจิตอล คอนเวอร์เตอร์(Analog to Digital Converter:A/D) แล้วส่งให้ระบบประมวลผลสัญญาณดิจิตอลต่อไป ระบบประมวลผลสัญญาณดิจิตอลก็จะนำเอาอัลกอริทึมที่มีอยู่มากระทำกับข้อมูลสัญญาณ จากนั้นก็จะส่งข้อมูลสัญญาณที่ถูกแก้ไขแล้ว ไปให้ส่วนดิจิตอล/อนาลอก คอนเวอร์เตอร์(Digital to Analog Converter:D/A) เพื่อเปลี่ยนสัญญาณข้อมูลกลับเป็นสัญญาณอนาลอกที่ต้องการ แต่จากเอาท์พุทของ D/A ซึ่งจะมีลักษณะเป็นขั้นๆแยกกัน ดังนั้นเราจำเป็นต้องใช้วงจรกรองสัญญาณให้เรียบ (Low Pass Filter) เพื่อที่จะขจัดสัญญาณความถี่สูงที่ไม่ต้องการออกไป

ที่พูดมานั้นมีแต่ส่วนดีของ DSP แต่ส่วนเสียของ DSP ก็มี ปัญหาที่เกิดขึ้นจะแตกต่างไปจากปัญหาที่เกิดขึ้นในวงจรอนาลอก เช่น การเกิดควันไตซ์เซชั่น(Quantization) , การเกิดน๊อยส์ (Noise), Idle channal noise , ความไม่ราบเรียบของสัญญาณ , ความละเอียดของสัญญาณ เป็นต้น ตัวแปรที่สำคัญอีกตัวหนึ่งที่จะแยกระบบ DSP ต่างๆออกตามการใช้งานก็คือค่าอัตราการสุ่มข้อมูล(Sampling Rate) ยิ่งอัตราการสุ่มข้อมูลมากยิ่งต้องใช้ระบบประมวลผลที่มีความเร็วสูงยิ่งขึ้นและซับซ้อนยิ่งขึ้นด้วย อัตราการสุ่มข้อมูลนี้จะมีผลต่อการนำไปใช้งาน ถ้ามีอัตราการสุ่มข้อมูลที่สูง ก็จะใช้งานในย่านความถี่ที่สูงขึ้นได้ โดยคุณภาพของสัญญาณข้อมูลยังคงยอมรับได้

บริษัทเทกซัสอินสตูเมนต์ จึงได้ผลิตไอซีที่เป็นตัวประมวลผลออกมาสำหรับใช้งานทางด้านของดิจิตอลซิกแนลโปรเซสซิงโดยเฉพาะ ซึ่งเป็นตระกูลที่มีชื่อว่า TMS320 ซึ่งมีวิวัฒนาการดังรูปที่ 1.4



รูปที่ 1.4 แสดงวิวัฒนาการของไอซีตระกูล TMS320xx

ไอซีตระกูล TMS320 นี้ จะเป็นลักษณะไมโครโปรเซสเซอร์แบบชิพเดียว (Single Chip) สามารถทำงานได้กว้างขวาง และมีความเร็วในการประมวลผลสูง

ไอซีเบอร์ TMS32010 นี้ สามารถนำไปใช้งานได้อย่างกว้างขวาง โดยเฉพาะการประมวลผลแบบเวลาจริง (Real Time) การประยุกต์ใช้งานไอซีเบอร์ TMS32010 แสดงในตารางที่ 1.1

GENERAL-PURPOSE DSP	GRAPHICS/IMAGING	INSTRUMENTATION
Digital Filtering Convolution Correlation Hilbert Transforms Fast Fourier Transforms Adaptive Filtering Windowing Waveform Generation	3-D Rotation Robot Vision Image Transmission/ Compression Pattern Recognition Image Enhancement Homomorphic Processing Workstations Animation/Digital Map	Spectrum Analysis Function Generation Pattern Matching Seismic Processing Transient Analysis Digital Filtering Phase-Locked Loops
VOICE/SPEECH	**CONTROL**	**MILITARY**
Voice Mail Speech Vocoding Speech Recognition Speaker Verification Speech Enhancement Speech Synthesis Text-to-Speech	Disk Control Servo Control Robot Control Laser Printer Control Engine Control Motor Control	Secure Communications Radar Processing Sonar Processing Image Processing Navigation Missile Guidance Radio Frequency Modems
TELECOMMUNICATIONS		**AUTOMOTIVE**
Echo Cancellation ADPCM Transcoders Digital PBXs Line Repeaters Channel Multiplexing 1200-to 19200-bps Modems Adaptive Equalizers DTMF Encoding/Decoding Data Encryption	FAX Cellular Telephones Speaker Phones Digital Speech Interpolation (DSI) X.25 Packet Switching Video Conferencing Spread Spectrum Communications	Engine Control Vibration Analysis Antiskid Brakes Adaptive Ride Control Global Positioning Navigation Voice Commands Digital Radio Cellular Telephones
CONSUMER	**INDUSTRIAL**	**MEDICAL**
Radar Detectors Power Tools Digital Audio TV Music Synthesizer Educational Toys	Robotics Numeric Control Security Access Power Line Monitors	Hearing Aids Patient Monitoring Ultrasound Equipment Diagnostic Tools Prosthetics Fetal Monitors

ตารางที่ 1.1 แสดงการประยุกต์ใช้งานไอซีเบอร์ IMS32010

ในปริญญานิพนธ์นี้ เราจะใช้ IMS 32010 มาสร้างเป็นระบบ DSP ซึ่งเราจะกำหนดคุณสมบัติขั้นต้นดังนี้

- สามารถนำไปประยุกต์ใช้งานได้ในย่าน ความถี่เสียง (0-20KHz)
- สามารถเปลี่ยนแปลงแก้ไขอัลกอริทึมในการประมวลได้โดยง่าย
- ค่าอัตราการสุ่มข้อมูลสามารถเปลี่ยนแปลงได้โดยง่าย
- จำนวนช่องสัญญาณอินพุท, เอาท์พุท สามารถเพิ่มเติมได้

ทั้งนี้ก็เพื่อความยืดหยุ่นของระบบที่จะนำไปใช้งานได้หลายๆแบบ และจะได้นำไปใช้เป็นชุดทดลองและพัฒนาทาง DSP ต่อไป

จากนั้นก็จะเป็นการนำเอาชุดทดลองและพัฒนาทาง DSP ที่ได้สร้างขึ้นนี้ไปทดลองใช้สังเคราะห์รูปคลื่นสัญญาณในแบบต่างๆและใช้เป็น วงจรกรองความถี่ในแบบต่างๆกัน

บทที่ 2 หลักการและทฤษฎี

2.1 สถาปัตยกรรมของ TMS32010

TMS32010 ถูกออกแบบโดยใช้สถาปัตยกรรมแบบฮาร์วาร์ด เป็นดิจิตอลซิกแนลโปรเซสซิงที่มีความสามารถทำงานในฟังก์ชันทางฮาร์แวร์ได้ และมีข้อดีหลายอย่าง เช่น มีความเร็วในการคูณเลข 16 บิท 2 ชุดได้ภายในเวลาเพียง 200 นาโนเซค มีบาร์เรลชิฟเตอร์ 0-16 บิท และฮาร์ดแวร์ที่สำคัญคือ ออกซิลลารีรีจิสเตอร์ (Auxillary Register) สำหรับการเข้าหาหน่วยความจำข้อมูลแบบทางอ้อม สำหรับรายละเอียดต่างๆที่อธิบายในบทนี้มีดังนี้

2.1.1 คุณสมบัติ

2.1.2 สัญญาณขา และรายละเอียดของสัญญาณ

2.1.3 สถาปัตยกรรม

2.1.4 ฟังก์ชันบล็อกไดอะแกรม (Function Block Diagram)

2.1.5 การจัดหน่วยความจำ

2.1.6 หน่วยประมวลผลทางคณิตศาสตร์

2.1.7 การควบคุมระบบ

2.1.8 ฟังก์ชันอินพุทและเอาท์พุท

2.1.9 การขออินเทอร์รัพ (inturrupt)

2.1.1 คุณสมบัติ

ไอซี TMS32010 มีคุณสมบัติดังต่อไปนี้

- วัฏจักรการทำงานต่อ 1 คำสั่ง จะใช้เวลาดังนี้:

  160 ns สำหรับ TMS32010-25/C10-25/C15-25/C17-25

  200 ns สำหรับ TMS32010/C10/11/C11/C15/E15/C17/E17

  280 ns สำหรับ TMS32010-14

- มีหน่วยความจำแบบแรม (RAM) ภายในไอซีขนาด 144/256 เวิร์ด (Words)

- มีหน่วยความจำแบบรอม (ROM) ภายในไอซีขนาด 1.5/4 กิโลเวิร์ด

- สามารถต่อหน่วยความจำภายนอกเพิ่มเติมได้ถึง 4 กิโลเวิร์ด

- มีบัสข้อมูลแบบสองทาง ขนาด 16 บิท และมีความเร็วในการรับ-ส่งข้อมูล 50เมกกะไบท์ต่อวินาที (Megabyte per second:Mbps)

- มีหน่วยประมวลผลทางคณิตศาสตร์ (Arithmatic Logic Unit:ALU) และรีจิสเตอร์แอคคิวมูเลเตอร์ (Accumulator Register) ขนาด 32 บิท

- สามารถทำคำสั่งในการคูณเลขขนาด 16 บิท 2 ชุด ภายใน 1 วัฏจักรการทำงาน และได้ผลลัพธ์เป็นเลขขนาด 32 บิท

- มีตัวเลื่อน (Shifter) ขนาด 0-16 บิท

- มีตัวกำเนิดสัญญาณนาฬิกา (Clock) ภายในตัวเอง

- มีพอร์ท (Port) สำหรับติดต่ออุปกรณ์ภายนอกได้ 8 พอร์ท

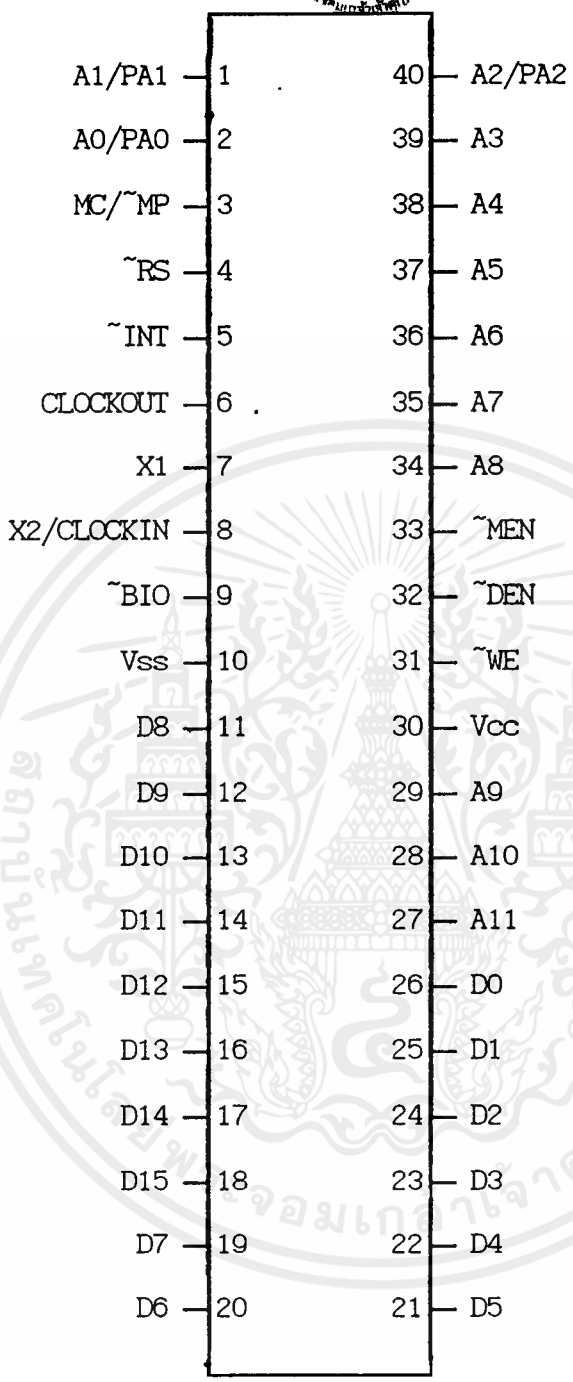
- มีไมโครโปรเซสเซอร์ร่วม สำหรับ TMS320C17/E17

- ใช้ไฟเลี้ยงไอซีขนาด 5 โวลท์

- เป็นไอซีขนาด 40 ขา

- ผลิตโดยเทคโนโลยีแบบ NMOS สำหรับ TMS32010/11 และแบบ CMOS สำหรับ TMS320C10/C15/E15/C17/E17

2.1.2 สัญญาณขา และรายละเอียดคุณสมบัติของ



รูปที่ 2.1 แสดงขาของไอซีเบอร์ IMS32010

สำหรับรายละเอียดของสัญญาณขาของ TMS32010 เป็นดังตารางที่ 2.1

สัญญาณ	สถานะ	รายละเอียด
A11 MSB A10 A9 A8 A7 A6 A5 A4 A3 A2/PA2 A1/PA1 A0/PA0	O	เป็นแอดเดรสบัสขนาด 12 บิท จาก A11(บิทสูง)ถึง A0(บิทต่ำ) และเป็นพอร์ทแอดเดรสขนาด 3 บิท จากบิท PA2 ถึง PA0 ปกติบัสแอดเดรสจะทำงานอยู่เสมอ ยกเว้นขณะกำลังทำงานในคำสั่ง IN และ OUT บิท A2 ถึง A0 จะถูกใช้เป็นพอร์ทแอดเดรส
D15 MSB D14 D13 D12 D11 D10 D9 D8 D7	I/O/Z	เป็นบัสข้อมูลแบบขนาน ขนาด 16 บิท บัสข้อมูลจะอยู่ในสภาวะไฮอิมพีแดนซ์เสมอ ยกเว้นในช่วง WE~ ทำงาน(ที่ลอจิก(logic) "0")

D6 D5 D4 D3 D2 D1 D0 LSB		
~BIO	I	เป็นอินพุทที่ถูกควบคุมโดยคำสั่ง BIOZ ถ้ามีลอจิก"0" จะใช้อุปกรณ์ที่แอดเดรสระบุ
~DEN	O	เป็นตัวกำหนดการรับข้อมูลจากอุปกรณ์อินพุท เมื่อมีลอจิก"0"จะทำการรับข้อมูลจากบัสข้อมูล และจะมีลอจิก"0"เฉพาะในช่วงวัฏจักรแรกของคำสั่ง IN เท่านั้น อนึ่งขณะที่ ~DENมีลอจิก"0"นั้น ~WE และ ~MEN จะไม่ทำงาน (มีลอจิก"1")
~INT	I	เป็นสัญญาณอินเทอร์รัพ (inturrupt signal) ทำงานที่ขอบขาลง (negative-going edge)
MC/~MP	I	เป็นขาที่ใช้เลือกโหมดของหน่วยความจำ ที่ลอจิก"1"จะใช้เป็นโหมดไมโครคอมพิวเตอร์ (microcomputer) ซึ่งจะใช้หน่วยความจำภายใน 1.5 กิโลเวิร์ด และต่อหน่วยความจำภายนอกได้ 2.5 กิโลเวิร์ด ส่วนโหมดไมโครโปรเซสเซอร์(ลอจิก"0") จะใช้เพียงหน่วยความจำภายนอกขนาด 4 กิโลเวิร์ด มีแอดเดรสตั้งแต่ 0 ถึง 4095

~MEN	O	เป็นสัญญาณขอใช้หน่วยความจำ จะทำงาน(ลอจิก"0")ในทุกรอบของแมชชีนไซเคิล (machine cycle) ยกเว้นเมื่อ ~WE และ ~DEN ทำงาน สัญญาณนี้จะถูกควบคุมโดยคำสั่งเฟท (fetch) จากโปรแกรมในหน่วยความจำ
~RS	I	เป็นสัญญาณรีเซ็ท (reset) ทำงานเมื่อลอจิก"0"มีค่าค้างอยู่ 5 สัญญาณนาฬิกา เมื่อทำงานแล้วจะมีผลทำให้ ~DEN,~WE,~MEN ถูกบังคับให้มีลอจิก"1" บัสข้อมูลเป็นไฮอิมพีแดนซ์ แอดเดรสบัสและโปรแกรมเคาเตอร์ (program counter) ถูกเคลียร์
~WE	O	เป็นสัญญาณการเขียนข้อมูลเอาท์พุท ซึ่งจะทำงานที่ลอจิก"0" ข้อมูลจะถูกส่งจากอุปกรณ์เอาท์พุท มายังบัสข้อมูล สัญญาณนี้จะทำงานเฉพาะในคำสั่ง OUT และในคำสั่ง IBLW
CLKOUT	O	สัญญาณนาฬิกาเอาท์พุทของระบบ ซึ่งมีความถี่เท่ากับ 1 ใน 4 ของความถี่ CLKIN
Vcc	I	ไฟเลี้ยง 5 โวลต์
Vss	I	กราวน์
X1	O	ขาคริสตอล(Crystal) เอาท์พุทสำหรับตัวกำเนิดความถี่ภายใน
X2/CLKIN	I	เป็นขาอินพุทให้แก่ตัวกำเนิดความถี่ทั้งภายในและภายนอก

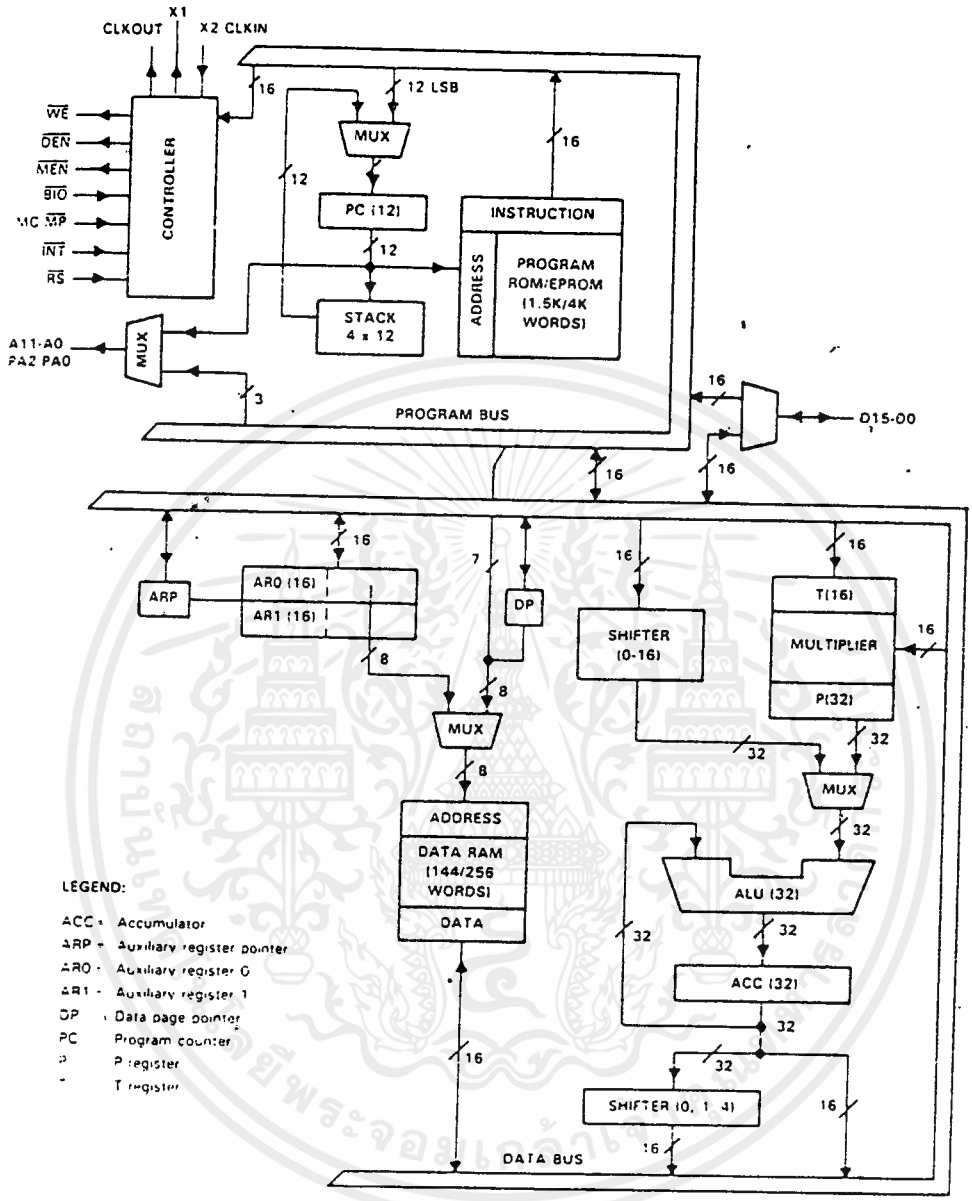
(I:อินพุท, O:เอาท์พุท , Z:ไฮอิมพีแดนซ์(High-impedance)

2.1.3 สถาปัตยกรรม

ดังที่กล่าวไว้แล้วว่า TMS32010 ออกแบบโดยใช้เทคโนโลยีของฮาร์วาร์ด เพราะต้องการความเร็ว และวงจรมีความซับซ้อน เทคโนโลยีแบบนี้ หน่วยความจำที่เก็บโปรแกรมและหน่วยความจำที่เก็บข้อมูลจะแยกกันอยู่คนละที่ จึงสามารถใช้คำสั่งเฟท และคำสั่งเอกซีคิวพร้อมกันได้ รวมทั้งยังสามารถรับส่งระหว่างโปรแกรมและข้อมูลได้ สามารถใช้โปรแกรมที่ถูกเก็บอยู่ในหน่วยความจำแบบแรมได้ ซึ่งจะเป็นความสะดวกมากขึ้น เนื่องจากไม่ต้องใช้หน่วยความจำแบบรอม TMS32010 ประกอบด้วยหน่วยประมวลผลทางคณิตศาสตร์และแอคคิวมูเลเตอร์ขนาด 32 บิท เพื่อใช้ในการคำนวณที่ต้องการความแม่นยำแบบ ดับเบิลเพรสชิชั่น (double-precision), ทู-คอมพลีเมนต์ อะริทเมติก (two's complement arithmatic) หน่วยประมวลผลทางคณิตศาสตร์จะประมวลผลข้อมูลขนาด 16 บิท และสามารถทำงานแบบบูลีน (Boolean) ส่วนแอคคิวมูเลเตอร์จะมีขนาด 32 บิท ทำหน้าที่เก็บข้อมูลจากหน่วยประมวลผลทางคณิตศาสตร์ และยังสามารถส่งข้อมูลกลับไปให้หน่วยประมวลผลทางคณิตศาสตร์ แอคคิวมูเลเตอร์นี้ถูกแบ่งเป็นสองเวิร์ด คือ เวิร์ดสูง (บิท 31 ถึง บิท 16) และ เวิร์ดต่ำ (บิท 15 ถึง บิท 0) มัลติพลายเออร์(multiplier) จะใช้สำหรับการคูณเลข 16 บิท และได้ผลลัพธ์ออกมาเป็นเลขขนาด 32 บิท ภายใน 1 ไซเคิลการทำงาน มัลติพลายจะประกอบด้วย 3 ส่วน ได้แก่ รีจิสเตอร์ T, รีจิสเตอร์ P และ มัลติพลายเออร์อะเรย์ (multiplier array) บาร์เรลชิฟเตอร์ จะทำการชิฟไปทางซ้าย 0 ถึง 16 บิท สำหรับเวิร์ดข้อมูลที่อยู่ในเมมโมรีและเข้าไปเก็บไว้ใน ALU ส่วนในแอคคิวมูเลเตอร์สามารถชิฟไปทางซ้ายได้ 0,1,4 บิท และเก็บผลไว้ในแอคคิวมูเลเตอร์เวิร์ดสูง นอกจากนี้ยังมีสแตค (stack) 4 ระดับ สำหรับเก็บค่าโปรแกรมเคาเตอร์ ระหว่างการขออินเทอร์รัพ และการเรียกใช้ซับรูทีน (sub routine)

2.1.4 ฟังก์ชันบล็อกไดอะแกรม

ฟังก์ชันบล็อกไดอะแกรมที่แสดงในบทนี้จะแสดงถึงการทำงานของ TMS32010 ทางเดินต่างๆใน TMS และ ขาสัญญาณสำหรับอินเทอร์เฟสกับอุปกรณ์ภายนอก



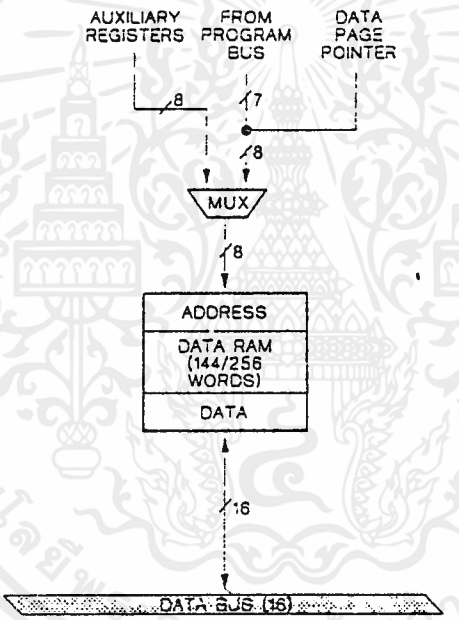
รูป 2.2 แสดงฟังก์ชันบล็อกไดอะแกรมของ TMS32010

2.1.5 การจัดหน่วยความจำ

ในหัวข้อนี้ จะให้รายละเอียดเกี่ยวกับ หน่วยความจำโปรแกรม หน่วยความจำข้อมูล การจัดผังของหน่วยความจำ ออกซิลารีรีจิสเตอร์ และ โหมดการอ้างหน่วยความจำ

หน่วยความจำข้อมูล

หน่วยความจำข้อมูลประกอบด้วย 144/256เวิร์ด 16 บิทบนออนชิพแรม (on-chip RAM) ดังรูป 2.3 ในกรณีที่ใช้หน่วยความจำข้อมูลแบบขยาย ข้อมูลจะถูกดำเนินการบน ออฟชิพแรม (off-chip RAM) และจะถูกอ่านเข้าไปในออนชิพแรมเมื่อต้องการ คำสั่งที่ใช้คือ TBLR/TBLW และ IN/OUT ตัวอย่างเช่น คำสั่ง TBLW จะเป็นการส่งข้อมูลจากหน่วยความจำภายนอก เข้ามายังหน่วยความจำข้อมูล และคำสั่ง IN จะเป็นการอ่านข้อมูลจากพอร์ทภายนอกเข้ามายังหน่วยความจำข้อมูลภายใน



รูป 2.3 แสดงหน่วยความจำข้อมูลแบบออนชิพ

หน่วยความจำโปรแกรม

เป็นหน่วยความจำที่ใช้เก็บชุดของคำสั่งโดยสำหรับใน TMS32010 จะมีหน่วยความจำภายใน หรือที่เรียกว่า "มาสค์รอม" (Mask ROM) มีขนาด 1.5 กิโลเวิร์ด

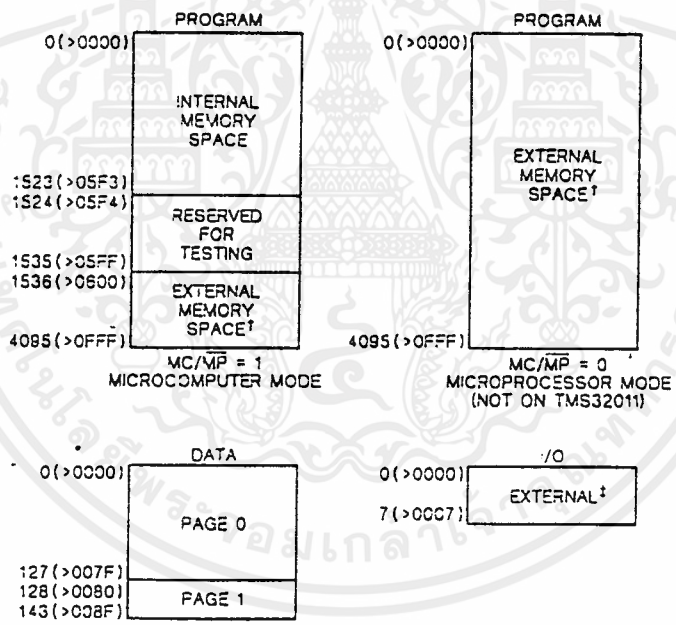
ในการใช้งานหน่วยความจำโปรแกรม เราสามารถเลือกได้ 2 โหมด คือ โหมดไมโคร-คอมพิวเตอร์ และโหมดไมโครโปรเซสเซอร์ โดยในการออกแบบวงจรเราสามารถเลือกได้จากขา MC/$\overline{MP}$

- โหมดไมโครคอมพิวเตอร์ : จะใช้หน่วยความจำรอมภายใน โดยเริ่มจาก แอดเดรสที่ 0 ถึง 1523 แต่แอดเดรสที่ 1524 ถึง แอดเดรสที่ 1535 จะสงวนไว้โดยทางบริษัท และจากแอดเดรสที่ 1536 ถึงแอดเดรสที่ 4095 จะเป็นหน่วยความจำที่เราต้องต่อเพิ่มไว้ภายนอก เป็นขนาด 2.5 กิโลเวิร์ด

- โหมดไมโครโปรเซสเซอร์ : จะใช้หน่วยความจำภายนอกทั้งหมด 4 กิโลเวิร์ด ตั้งแต่แอดเดรสที่ 0 จนถึง แอดเดรสที่ 4095

การจัดหน่วยความจำ

TMS32010 ได้ทำการแบ่งส่วนแอดเดรสไว้ 3 ส่วน สำหรับเก็บหน่วยความจำโปรแกรม หน่วยความจำข้อมูล และอินพุท/เอาท์พุท ดังแสดงในรูป 2.4 ซึ่งแสดงการจัดเก็บทั้ง 2 โหมด



รูปที่ 2.4 แสดงลักษณะการจัดหน่วยความจำภายใน TMS32010

ออกซิลารีรีจิสเตอร์

ใน TMS32010 จะมีรีจิสเตอร์ประเภทหนึ่งที่เรียกว่ารีจิสเตอร์ออกซิลลารี หรือ AR ซึ่ง

แบ่งออกเป็น 2 ตัว คือ AR0 และ AR1 รีจิสเตอร์ AR มักจะใช้ในการอ้างหน่วยความจำแบบทางอ้อม (Idirect Addressing Mode) สำหรับคำสั่งที่เกี่ยวกับการอ้างหน่วยความจำแบบทางอ้อมนั้น จะต้องใส่โอเปอร์แรน (operand) ซึ่งเป็นตำแหน่งของหน่วยความจำของข้อมูลที่ 8 บิทท้ายของ AR ด้วย

เนื่องจากรีจิสเตอร์ AR มีอยู่ 2 ตัว ดังนั้นเราสามารถเลือกใช้ได้โดย ออกซิลลารีรีจิสเตอร์พอยน์เตอร์ (Auxillary Register Poiter:ARP) ซึ่งจะเป็นบิทหนึ่งภายในรีจิสเตอร์บอกสถานะ (Status Register)

โหมดการอ้างหน่วยความจำ

วิธีการอ้างหน่วยความจำของ TMS 32010 สามารถแบ่งออกได้เป็น 3 แบบคือ

1) การอ้างหน่วยความจำโดยตรง (Direct Addressing Mode)

สำหรับการอ้างแอดเดรสของหน่วยความจำโดยตรงนั้น ในคำสั่งจะประกอบไปด้วยแอดเดรสของหน่วยความจำข้อมูล (Data memory Address : dma) ที่ 7 บิทท้ายของตัวคำสั่ง และบิทที่ 7 จะเป็นค่าพอยน์เตอร์ที่ชี้บอกหน้าของหน่วยความจำข้อมูล (Data Memory Page Point : DP) ซึ่งจะเป็นตัวเลือกหน้า ของหน่วยความจำข้อมูล

การอ้างแอดเดรสของหน่วยความจำโดยตรงสามารถใช้ได้สำหรับทุกคำสั่ง ยกเว้นคำสั่งต่อไปนี้

- CALL
- BANZ
- คำสั่งที่ให้ค่าโอเปอร์แรนโดยทันที (Immediate Operand)
- คำสั่งที่ไม่มีโอเปอร์แรน

2) การอ้างแอดเดรสของหน่วยความจำโดยทางอ้อม (Indirect Addressing Mode)

จะเป็นการอ้างถึงตำแหน่งของหน่วยความจำข้อมูลที่ 7 บิทล่างของคำสั่ง และที่บิทที่ 7 จะเป็นค่าของ ARP การอ้างแอดเดรสของหน่วยความจำโดยทางอ้อมจะไม่มีการเลือกหน้าของหน่วยความจำข้อมูล ARP ก็จะเป็นตัวเลือกว่าเป็น AR0 หรือ AR1 AR สามารถเพิ่มหรือลดลงได้โดยอัตโนมัติ เมื่อเราได้มีการใช้คำสั่งที่ใช้อ้างแอดเดรสของหน่วยความจำโดยทางอ้อม

สำหรับการใช้คำสั่งประเภทอ้างแอดเดรสของหน่วยความจำโดยทางอ้อม จะมี AR อยู่ในคำสั่งเป็นจำนวน 9 บิท ซึ่งสามารถถูกโหลดได้ โดย คำสั่ง LAR, LARK และสามารถ

เปลี่ยนค่าได้โดยคำสั่ง MAR

การอ้างแอดเดรสของหน่วยความจำโดยทางอ้อม จะใช้สัญลักษณ์ดังต่อไปนี้ สำหรับการเขียนภาษาแอสเซมบลีของ TMS32010

- "*" : จะใช้ค่าของ AR สำหรับชี้ค่าในหน่วยความจำข้อมูล
- "*-" : จะใช้ค่าของ AR สำหรับชี้ค่าในหน่วยความจำข้อมูล และจะลดค่าของ AR ลงไป 1 หลังจากทำคำสั่งเสร็จ
- "*+" : จะใช้ค่าของ AR สำหรับชี้ค่าในหน่วยความจำข้อมูล และจะเพิ่มค่าของ AR ขึ้นไป 1 หลังจากทำคำสั่งเสร็จ

3) การอ้างแอดเดรสของหน่วยความจำโดยทันที (Imediate Addressing Mode)

ในชุดคำสั่งของ TMS32010 จะมีอยู่ 5 คำสั่ง ที่ใช้ในการแอดเดรสของหน่วยความจำโดยทันที ซึ่งคำสั่งเหล่านี้จะทำงานได้ภายใน 1 วัฏจักรคำสั่ง โดยคำสั่งเหล่านี้ได้แก่

LARK

LACK

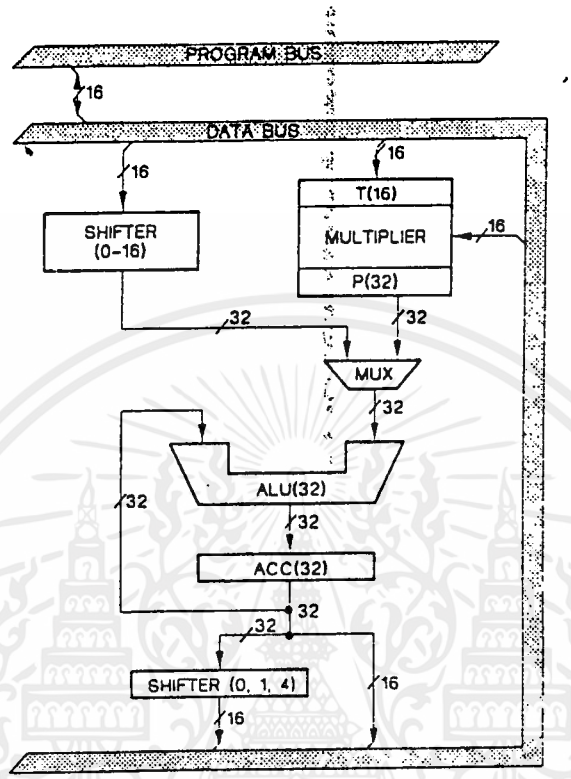
LARP

LDPK

MPYK

2.1.6 หน่วยประมวลผลทางคณิตศาสตร์กลาง

หน่วยประมวลผลทางคณิตศาสตร์กลาง ประกอบด้วยหน่วยการคูณเลข 16 บิทแบบขนาน หน่วยประมวลผลทางคณิตศาสตร์ แอคคิวมูเลเตอร์ขนาด 32 บิท และตัวเลื่อนข้อมูล 2 ชุด ซึ่งบล็อกไดอะแกรมแสดง ในรูปที่ 2.5



รูปที่ 2.5 แสดง บล็อกไดอะแกรม ของหน่วยประมวลผลทางคณิตศาสตร์กลาง (CALU)

2.1.7 การควบคุมระบบ

การควบคุมระบบของ IMS32010 ทำโดยใช้ โปรแกรมเคาท์เตอร์ สแตก สัญญาณรีเซ็ทจากภายนอก สัญญาณอินเทอร์รัพ และ สเตตัสรีจิสเตอร์

- โปรแกรมเคาท์เตอร์และ สแตก จะใช้งานในคำสั่งที่เกี่ยวกับ การเรียกโปรแกรมย่อย การอินเทอร์รัพ และคำสั่งอ่าน/เขียนตาราง โปรแกรมเคาท์เตอร์ซึ่งมี 12 บิท จะทำการเก็บค่าแอดเดรสของหน่วยความจำโปรแกรม ที่กำลังทำงานอยู่ IMS32010 จะทำการอ่านหน่วยความจำโปรแกรมในแอดเดรสที่โปรแกรมเคาเตอร์ระบุ และทำการเพิ่มค่าโปรแกรมเคาท์เตอร์ไปหนึ่ง สำหรับการเฟทคำสั่งต่อไป โปรแกรมเคาท์เตอร์จะมีค่าเป็นศูนย์ เมื่อ

สัญญาณรีเซ็ททำงาน

- **สัญญาณรีเซ็ท** เป็นการอินเทอร์รัพแบบ นอน-มาสเคเบิล (non-mask able) เพื่อเป็นการให้ค่าเริ่มต้นแก่ TMS32010 การรีเซ็ท TMS32010 จะเกิดผลตามมาดังนี้

- สัญญาณ ~DEN, ~WE และ ~MEN จะมีค่าเป็นระดับลอจิก"1"

- บัสข้อมูลจะเป็นไฮอิมพีแดนซ์

แรกเริ่มเคาท์เตอร์จะถูกเซทเป็นศูนย์ และแอดเดรสบัส (A11-A0)จะมีค่าเป็นศูนย์ทั้งหมด หลังจากสัญญาณนาฬิกาลูกถัดไปเมื่อสัญญาณรีเซทเป็นระดับลอจิก"0"

- ไม่สามารถขออินเทอร์รัพได้ และอินเทอร์รัพ แฟลก รีจิสเตอร์ ถูกเซทเป็นศูนย์

- **สเตตัสรีจิสเตอร์** ประกอบด้วยรีจิสเตอร์บอกสถานะ 5 บิท ได้แก่ ออกซิลลารีรีจิสเตอร์ (ARP), พอยเตอร์บอกหน้าหน่วยความจำข้อมูล (DP), อินเทอร์รัพโหมดบิท (INTM), โอเวอร์โฟลว์แฟลก (OV) และ โอเวอร์โฟลว์โหมดบิท (OVM) แสดงดังรูปที่ 2.6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	OVM	INTM	1	1	1	1	ARP	1	1	1	1	1	1	0	DP

รูปที่ 2.6 แสดงรีจิสเตอร์บอกสถานะของ TMS32010

รีจิสเตอร์บอกสถานะสามารถเก็บเข้า และโหลด จากหน่วยความจำได้โดยใช้คำสั่ง SST และ LST ตามลำดับ

ARP ใช้สำหรับการอ้างแอดเดรสของหน่วยความจำโดยทางอ้อม สำหรับบิทนี้จะมีผลเมื่อเกิดการอ้างแอดเดรสของหน่วยความจำโดยทางอ้อม และ คำสั่ง LARP, MAR, LST

DP เป็นพอยน์เตอร์ที่ชี้บอกหน้าของหน่วยความจำข้อมูล มักจะใช้ในคำสั่งที่มีการอ้างแอดเดรสของหน่วยความจำข้อมูลโดยตรง โดยจะไปอยู่บิทที่ 7 ของออปโคดของคำสั่ง ถ้า DP มีค่าเป็น"0" ก็จะหมายถึงหน่วยความจำข้อมูล 128 เวิร์ดแรก หรือเรียกว่าหน้าที่ศูนย์ และถ้า DP มีค่าเป็น"1" ก็จะหมายถึงหน่วยความจำข้อมูลตำแหน่งที่ 128 จนถึงตำแหน่งที่ 143 หรือเรียกว่าเป็นหน้าที่หนึ่ง ค่าของ DP จะเปลี่ยนค่าได้โดยคำสั่ง LST, LDP และ LDPK

INIM ถ้าเราทำการอีนาเบิลอินเทอร์รัพโดยทำให้บิทนี้เป็น "0" เมื่อเกิดการอินเทอร์รัพขึ้น ก็จะทำให้บิทนี้เป็น "1" (ก่อนที่จะเข้าโปรแกรมตอบสนองการอินเทอร์รัพ)

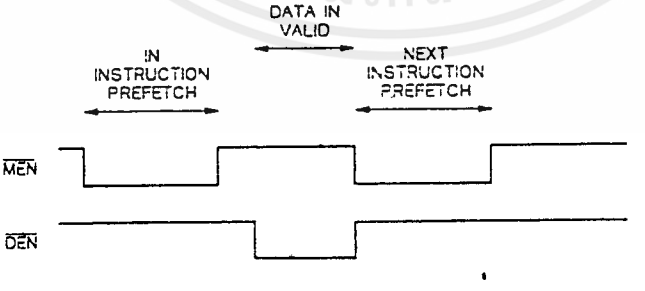
บิทนี้สามารถทำการเซท และรีเซทได้โดยคำสั่ง DINT และ EINT ตามลำดับ ส่วนคำสั่งการรีเซท ที่ขา RS ก็จะมีผลทำให้บิทนี้มีค่าเป็น "1"

OV เป็นบิทบอกสถานะของการเกิดโอเวอร์ไฟลว์ ถ้าบิทนี้เป็น "0" จะแสดงว่าแอคคิวมูเลเตอร์ไม่เกิดโอเวอร์ไฟลว์ แต่ถ้าบิทนี้เป็น "1" จะแสดงว่าแอคคิวมูเลเตอร์เกิดการโอเวอร์ไฟลว์ เราสามารถเคลียร์บิทนี้ได้โดยคำสั่ง BV, LST หรือการรีเซทที่ขา ~RS

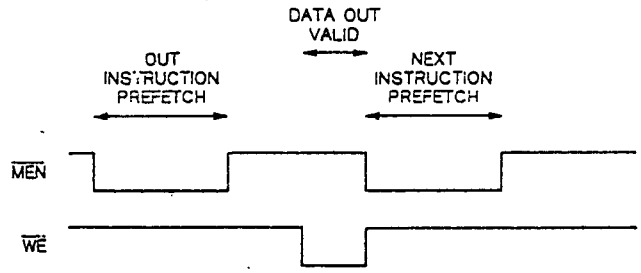
OVM เป็นบิทบอกสถานะของโหมดโอเวอร์ไฟลว์ ถ้าบิทนี้เป็น "0" จะเป็นการดิสเอเบิลการโอเวอร์ไฟลว์ แต่ถ้าบิทนี้เป็น "1" ก็จะเป็นการอีนาเบิลการโอเวอร์ไฟลว์ เราสามารถเซท หรือรีเซทบิทนี้ได้โดยคำสั่ง SOVM และ ROVM ตามลำดับ นอกจากนี้แล้วคำสั่งLST ก็มีผลต่อบิทนี้ด้วย

2.1.8 ฟังก์ชันอินพุทและเอาท์พุท

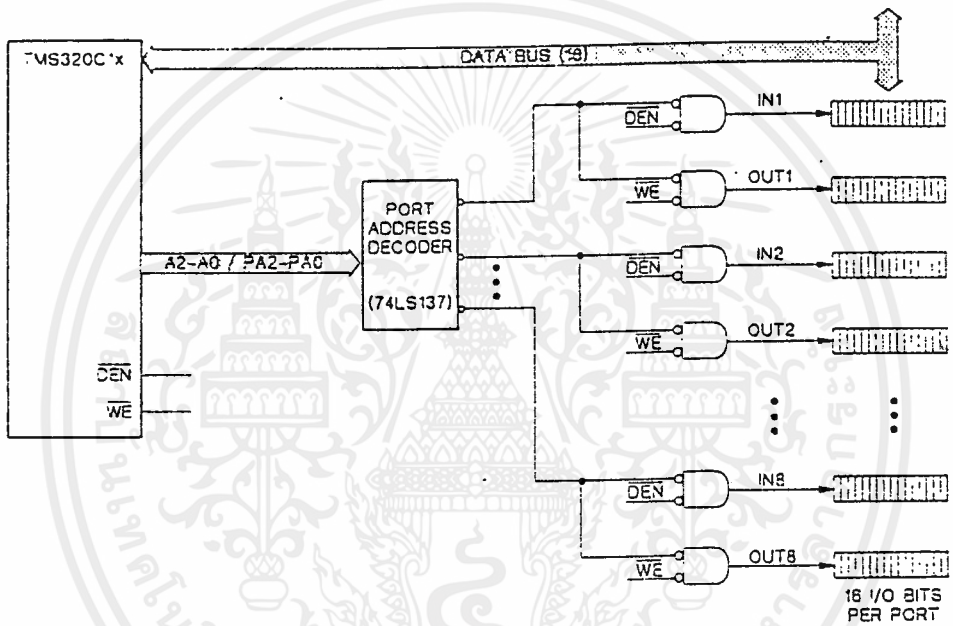
ฟังก์ชันอินพุทและเอาท์พุท ใช้สำหรับติดต่อกับอุปกรณ์ภายนอก ทางบัสข้อมูลขนาด 16 บิท โดยใช้คำสั่ง IN และ OUT พอร์ทอินพุทและเอาท์พุท จะอยู่ที่สามบิทต่ำของแอดเดรสบัส(PA2-PA0) ข้อมูลจะถูกรับ-ส่งระหว่างบัสข้อมูลและหน่วยความจำข้อมูล โดยคำสั่ง ~DEN และ ~WE บัสข้อมูลสองทิศทางภายนอกจะอยู่ในสภาวะไฮอิมพีแดนซ์เสมอ ยกเว้นเมื่อสัญญาณ ~WEสั่งทำงาน (ลอจิก"0") หรือระหว่างคำสั่ง IN ~WE จะทำงานระหว่าง ไซเคิลแรกของคำสั่ง OUT และไซเคิลที่สองของคำสั่ง TBLW และเนื่องจากแอดเดรสของพอร์ทมี 3 บิท จึงสามารถอ้างพอร์ทได้ทั้งหมด 8 พอร์ท และเพื่อความเข้าใจสำหรับการทำงานของอินพุทและเอาท์พุทจะแสดงดังรูป



รูปที่ 2.7 จะแสดงไทม์มิงไดอะแกรม ของคำสั่งอินพุท



รูปที่ 2.8 จะแสดงไทม์มิงไดอะแกรม ของคำสั่งเอาท์พุท



รูปที่ 2.9 แสดงการอินเทอร์เฟส กับอุปกรณ์ภายนอกของ TMS32010

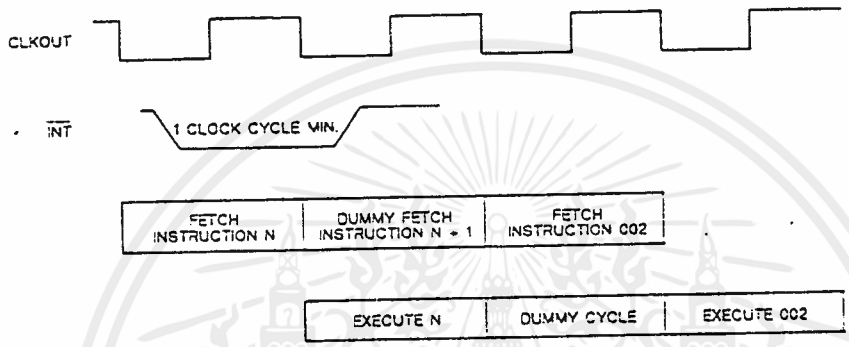
2.1.9 การขออินเทอร์รัพ

สัญญาณอินเทอร์รัพ จะทำงานที่ขอบขาลงของระดับลอจิก เมื่อสัญญาณอินเทอร์รัพอยู่ในสถานะอีนาเบิล (enable) หรือยอมให้มีการอินเทอร์รัพได้ (INTIM เป็นศูนย์) การอินเทอร์รัพจะขึ้นอยู่กับระดับลอจิกของขา ~INT แต่ถ้าอยู่ในสถานะดิสเอเบิล (disable) INTIM จะมีค่าเป็นลอจิก"1" จะไม่ยอมให้มีการอินเทอร์รัพไม่ว่าลอจิกของขา ~INT จะเป็นค่าใด

ก็ตาม

การอินเทอร์รัพจะถูกหน่วงเวลาในกรณีต่อไปนี้

- จนกว่าจะจบคำสั่งสุดท้ายของไซเคิล
- จนกว่าคำสั่งที่ต่อจากคำสั่ง MPY หรือ MPYK จะทำงานเสร็จสมบูรณ์
- จนกว่าคำสั่งที่ต่อจากคำสั่ง EINT ทำงาน



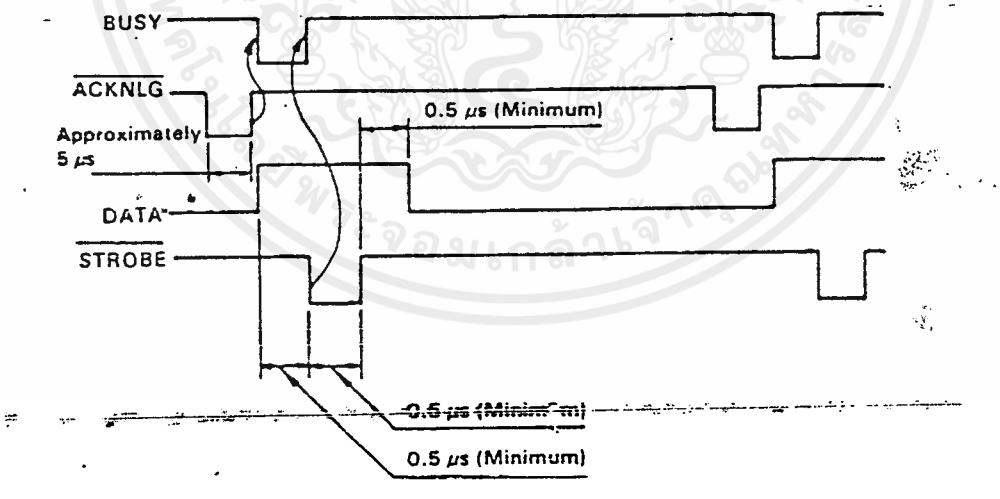
รูปที่ 2.10 แสดงไทม์มิง ของการอินเทอร์รัพ

2.2 <u>การอินเทอร์เฟสกับไอบีเอ็มพีซี</u>

การอินเทอร์เฟสกับไอบีเอ็มพีซีเลือกใช้ผ่านทางพอร์ทแบบขนาน เนื่องจากสามารถส่งข้อมูลเป็นชุด ทีละ 8 บิท ได้ด้วยความเร็วสูง โดยผ่านทางปรินเตอร์อแดปเตอร์ (Printer Adapter) หรือ โมโนโครม/ปรินเตอร์อะแด่ปเตอร์ (MonoChrome/Printer Adapter) ซึ่งค่าแอดเดรสของพอร์ทที่จะส่งข้อมูล จะขึ้นอยู่กับชนิดของการ์ดที่มีในเครื่องนั้น ถ้าเป็นปรินเตอร์อะแดปเตอร์การ์ด จะต้องส่งเอาท์พุทออกไปยังพอร์ท 378 แต่ถ้าเป็น โมโนโครม/ปรินเตอร์อะแดปเตอร์ การ์ด จะต้องใช้พอร์ท 3BC คุณสมบัติโดยทั่วไปของพอร์ทขนานมีดังนี้

- มีอัตราการส่งข้อมูลจาก ไอบีเอ็ม 1000CPS (อักษรต่อวินาที)ซึ่งจะผ่านทางสายเคเบิล 25 ขา (25 pin D-shell connector)
- ซิงโครไนซ์ (Synchronization) โดยใช้สัญญาณ ~STROBE
- แฮนด์เชคกิง (Handshaking) กับสัญญาณ ~ACKNLG และ BUSY
- ใช้ระดับลอจิกอินพุทและสัญญาณควบคุม เช่นเดียวกับระดับลอจิกของไอซี TTL

เนื่อกจากพอร์ทขนานทำหน้าที่ส่งข้อมูลให้กับเครื่องพิมพ์ ดังนั้น สัญญาณ บางขาที่ใช้ควบคุมเครื่องพิมพ์จึงไม่จำเป็นต้องใช้ จะใช้เพียงสัญญาณที่จำเป็นเพียง 3 สัญญาณ คือสัญญาณ BUSY , ~STROBE และ ~ACKNLG ซึ่งจะมีลำดับของการส่งสัญญาณ ดังรูปที่ 2.11



รูปที่ 2.11 แสดงไทม์มิงไดอะแกรมของการอินเทอร์เฟสข้อมูลแบบขนาน

จากรูปจะมีเพียง สัญญาณ ~STROBE และ ข้อมูล 8 บิทที่ส่งมาจากเครื่องไอบีเอ็ม ส่วนสัญญาณ BUSY และ ~ACKNLG จะต้องทำการสร้างขึ้นเองและส่งไปให้ไอบีเอ็ม

สำหรับการทำงานจะเป็นดังนี้

เมื่อมีการส่งข้อมูลผ่านทางพอร์ทขนาน สัญญาณ BUSY จะถูกตรวจสอบ ถ้ามีค่าเป็น "1" แสดงว่าอุปกรณ์ภายนอกไม่พร้อมที่จะรับข้อมูล สัญญาณ BUSY ก็จะถูกตรวจสอบใหม่จนกว่าจะเปลี่ยนค่าเป็น "0"ซึ่งแสดงว่าอุปกรณ์ภายนอกพร้อมที่จะรับข้อมูลแล้ว สัญญาณ~STROBE ซึ่งปกติจะเป็น "1" จะถูกเปลี่ยนค่าเป็น "0" และสัญญาณBUSY จะต้องถูกทำให้เป็น "1" เพื่อให้ข้อมูลถูกจัดเก็บก่อน และไม่ให้ข้อมูลใหม่เข้ามา และสัญญาณ~STIRBE จะถูกหน่วงเวลาไว้อีกอย่างน้อย 0.5 ไมโครเซคก่อนจะเปลี่ยนเป็น "1" ตามเดิม เพื่อให้อุปกรณ์ภายนอกนำสัญญาณนี้ไปทริก และทำการเก็บข้อมูลไว้ เมื่อข้อมูลถูกเก็บเสร็จแล้ว สัญญาณ~ACKNLG จะมีค่าเป็น "0" ประมาณ 5 ไมโครเซก ก่อนที่จะทำให้สัญญาณ BUSY เป็น "0" ใหม่ เป็นการเสร็จการส่งข้อมูล 8 บิท แบบขนาน 1 ชุด ถ้ายังมีข้อมูลที่จะส่งไปยังอุปกรณ์ภายนอกอีก การทำงานดังกล่าวก็จะเริ่มอีก

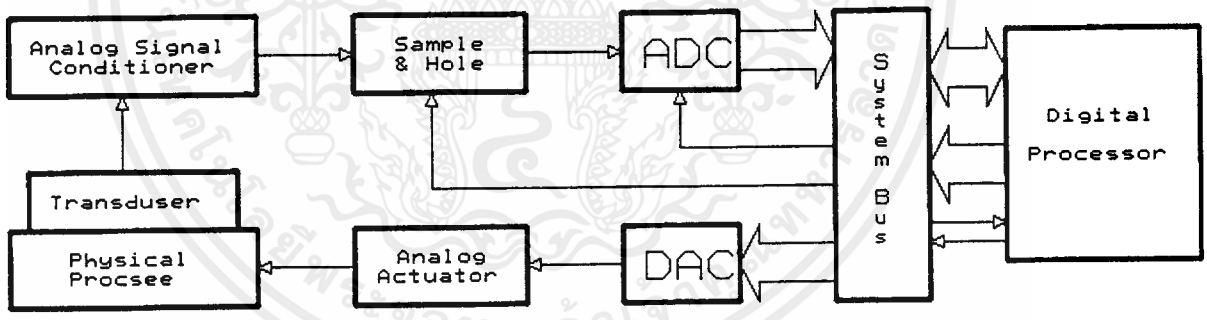
2.3 <u>ทฤษฎีของการแปลงสัญญาณ</u> (Data Acquisition and Conversion)

รูปแบบของสัญญาณทางไฟฟ้า ที่เราคุ้นเคยกันแต่เดิมนั้น โดยมากมักจะอยู่ในรูปของสัญญาณอนาลอก แต่ก่อนนั้นการที่จะนำเอาสัญญาณทางไฟฟ้า มาประมวลผล เพื่อให้เกิดรูปแบบที่ต้องการ นั้นจะกระทำได้โดยใช้อุปกรณ์ทางอนาลอกนั่นเอง

แต่ปัจจุบันนี้เทคโนโลยีทางด้านดิจิตอลได้ก้าวหน้าไปมากทำให้การประมวลผลสัญญาณทางดิจิตอลสามารถทำได้อย่างรวดเร็ว มีประสิทธิภาพและความเชื่อถือได้ ดีกว่าอดีตมากมายนัก

ดังนั้นการเปลี่ยนรูปแบบของสัญญาณ (Conversion) จึงได้มีความจำเป็นขึ้น จากสัญญาณอนาลอกที่มีอยู่แล้วได้ถูกเปลี่ยนให้เป็นสัญญาณทางดิจิตอล โดยอุปกรณ์อนาลอกทูดิจิตอลคอนเวอเตอร์ และจะถูกประมวลผลโดยตัวประมวลผลสัญญาณดิจิตอล เช่น คอมพิวเตอร์ เป็นต้น จากนั้นผลลัพธ์ที่ได้อาจถูกนำมาแสดงผลโดยตรงเลย หรืออาจถูกเปลี่ยนกลับให้อยู่ในรูปของสัญญาณอนาลอกที่ใช้งานได้ การที่จะเปลี่ยนสัญญาณดิจิตอลกลับเป็นสัญญาณอนาลอกนั้นสามารถทำได้โดยใช้อนาลอกทูดิจิตอลคอนเวอเตอร์

สำหรับระบบที่มีการประมวลผลข้อมูลทางดิจิตอลแสดงดังรูปที่ 2.12



รูปที่ 2.12แสดงระบบที่มีการประมวลผลข้อมูลทางดิจิตอล

จากรูปข้างต้นสามารถอธิบายได้คือ การเปลี่ยนแปลงทางกายภาพในลักษณะใดๆ ก็ตาม เช่น อุณหภูมิ ความดัน ความเร็ว จะถูกเปลี่ยนให้มาเป็นสัญญาณไฟฟ้าแบบอนาลอก โดย

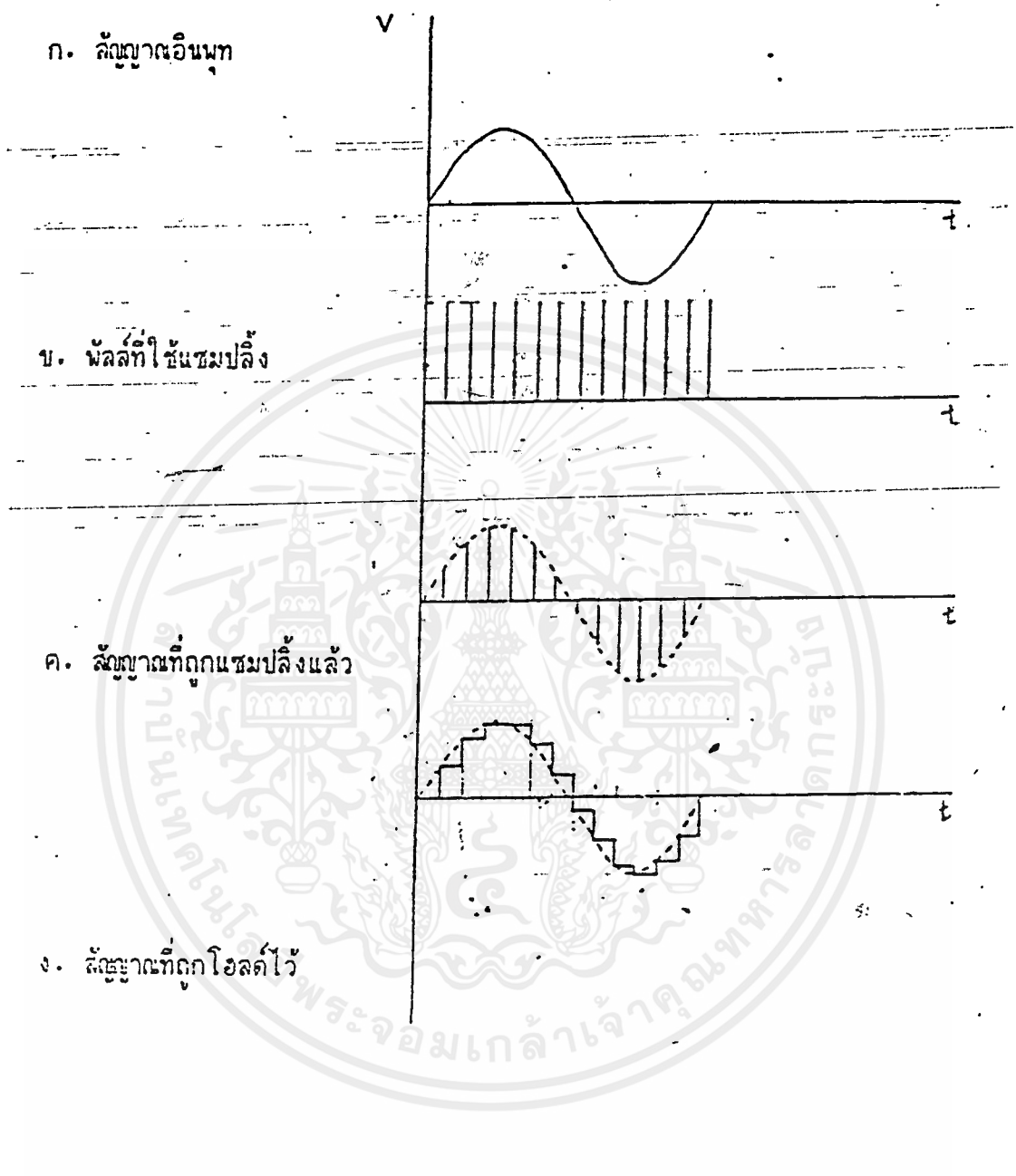
ทรานสดิวเซอร์ (Transducer) ที่มีรูปแบบเหมาะสมกับลักษณะทางกายภาพนั้นๆ จากนั้นสัญญาณทางไฟฟ้าก็จะถูกปรับให้อยู่ในรูปแบบ และขนาดที่เหมาะสมก่อน โดยวงจรต่างๆ เช่น วงจรขยาย หรือ วงจรกรองสัญญาณ เป็นต้น

วงจรแซมเปิลแอนด์โฮลด์ จะสุ่มขนาดของสัญญาณอนาลอกออกมา แล้วก็จะทำการโฮลด์สัญญาณนั้นไว้ชั่วขณะ เพื่อให้ไม่จำเป็นต้องใช้ ADC ที่มีความเร็วมากนัก เมื่อสัญญาณถูกแปลงเป็นดิจิตอลโดย ADC แล้วข้อมูลดิจิตอลจะถูกส่งต่อไปยังบัสของระบบ จากนั้นตัวโปรเซสเซอร์จะทำการประมวลผลข้อมูล แล้วเปลี่ยนข้อมูลผลลัพธ์กลับมา เพื่อควบคุมกิจกรรมทางกายภาพของระบบ โดยผ่านตัวกระทำทางกล (Analog Actuator)

2.3.1 ทฤษฎีของการสุ่มข้อมูล (Sampling)

ในการสุ่มข้อมูลนั้นสัญญาณอนาลอกจะถูกสุ่มเป็นระยะๆคงที่ กลุ่มของสัญญาณที่สุ่ม จะแทนด้วยความสูง ซึ่งเกิดจากการตัดต่อสัญญาณอนาลอกด้วยระยะเวลาอันสั้น ผลของการสุ่มด้วยความเร็ว จะเหมือนกับการคูณขบวนสัญญาณพัลซ์ กับสัญญาณอนาลอก ซึ่งจะได้สัญญาณที่มอดูเลท(Modulate) ระหว่างขบวนพัลส์กับสัญญาณอนาลอก ดังแสดงในรูปที่ 2.13 และถ้าหากเรามีการโฮลด์สัญญาณที่สุ่มได้เอาไว้ เราก็จะได้สัญญาณดังรูปที่ 2.13ง

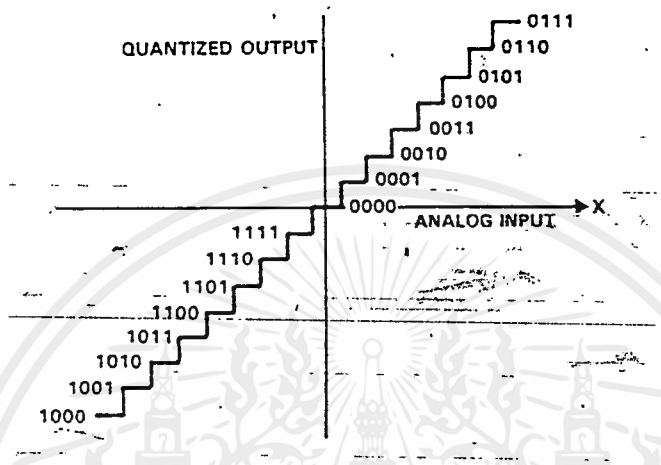
อัตราการสุ่มสัญญาณ หรือความถี่ของการสุ่มสัญญาณควรจะมีค่าเท่าใด ที่ข้อมูลที่สุ่มได้นี้จะเป็นตัวแทนที่ดีของสัญญาณต่อเนื่องนั้น คำตอบคือขึ้นอยู่กับความถี่ของสัญญาณอนาลอก และจากทฤษฎีของการสุ่ม (Nyquist Theorem) กล่าวไว้ว่า "ถ้าสัญญาณต่อเนื่องที่มีความถี่ฮาร์โมนิค (Hamonic Frequency) ไม่เกิน f แล้ว สัญญาณดังกล่าวจะสามารถเปลี่ยนกลับมาเช่นเดิมโดยไม่สูญเสียรายละเอียด หรือผิดเพี้ยนไป ถ้าอัตราการสุ่มมากกว่า 2f "



รูปที่ 2.13 แสดงการสุ่มสัญญาณ

2.3.2 ทฤษฎีการควันไตซ์ (Quantizing Theorem)

การควันไตซ์ เป็นขบวนการที่แปลงสัญญาณอนาลอก ให้เป็นข้อมูลทางดิจิตอลที่เป็นสัดส่วนกับ สัญญาณอนาลอก เช่นอยู่ในรูปของรหัสไบนารี เป็นต้น หากเรานำเอาท์พุทที่ได้จากการควันไตซ์ และขนาดของสัญญาณอนาลอก มาเขียนเป็นกราฟ ก็จะได้กราฟควันไตซ์ทรานสเฟอร์ฟังชัน (Quantize Transfer Function) ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 แสดงทรานสเฟอร์ฟังชันของการควันไตซ์ 4 บิท

จุดสำคัญของการควันไตซ์นั้นได้แก่ รีโซลูชัน (Resolution) ของตัวควันไตซ์ ซึ่งจะสามารถกำหนดได้จากจำนวนบิทของเอาท์พุทรหัสดิจิตอล หรือจากกราฟรูปที่ 2.14 ก็คือส่วนกลับของความกว้างของขั้นบันไดทางแกนนอนนั่นเอง

ค่าความกว้างของขั้นบันไดนี้จะสามารถคำนวนได้จาก

$$Q = FSR / 2^n$$

โดยที่

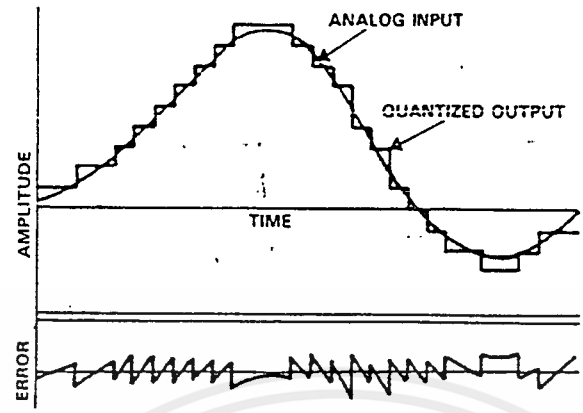
Q : ค่าความกว้างของขั้นบันไดทางแกนนอน

FSR : ช่วงเต็มสเกลของสัญญาณอนาลอก (Full Scale Range)

n : จำนวนบิทของรหัสดิจิตอล

จากสมการข้างต้นจะเห็นว่าเมื่อจำนวนบิทมากขึ้น ขนาดของ Q ก็จะลดลง และถ้าเรานำสัญญาณอนาลอกใดๆ มาทำการควันไตซ์ จะเห็นได้ว่าเมื่อนำเอาผลที่ได้จากการควันไตซ์

มาเปรียบเทียบกับสัญญาณอนาลอกนั้นแล้ว ก็จะพบว่ามีผิดพลาดเกิดขึ้น ซึ่งเราจะเรียกความผิดพลาดนี้ว่า ความผิดพลาดควันไตซ์ (Quantizing Error) ดังแสดงในรูปที่ 2.15

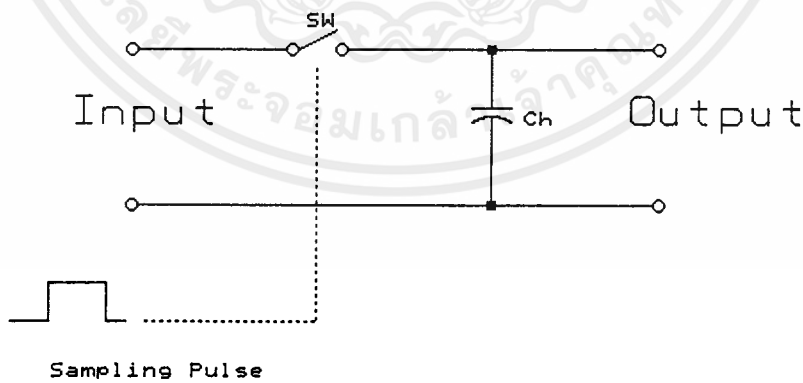


รูปที่ 2.15 แสดงความผิดพลาดควันไตซ์

ความผิดพลาดนี้เป็นธรรมชาติของการควันไตซ์ ซึ่งจะทำการแก้ไขไม่ได้ แต่เราจะลดค่าความผิดพลาดนี้ได้ก็โดยการเพิ่มจำนวนบิทของการควันไตซ์ ค่าความผิดพลาดนี้เองที่จะเป็นตัวกำหนดความแม่นยำในการใช้งาน DSP

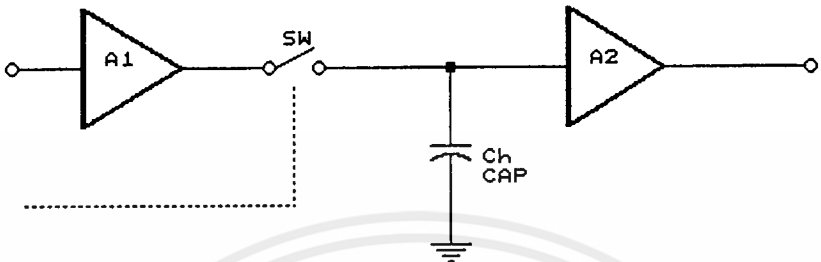
2.3.3 วงจรแซมเปิลแอนด์โฮลด์

วงจรแซมเปิลแอนด์โฮลด์ โดยพื้นฐานแล้วจะเป็นวงจรหรือ อุปกรณ์เก็บแรงดัน (Voltage Memory) ซึ่งใช้อุปกรณ์ที่สำคัญคือ ตัวเก็บประจุ ดังในรูปที่ 2.16



รูปที่ 2.16 แสดงวงจรพื้นฐานของการแซมเปิลแอนด์โฮลด์

อิเลคทรอนิคส์สวิทช์จะต่อสัญญาณแรงดันเข้ากับตัวเก็บประจุ ซึ่งสวิทช์จะถูกควบคุมจากแซมปลิงพัลส์ ช่วงเวลาการตัดสวิทช์ และเวลาในการประจุจนแรงดันถึงค่าที่สุ่มมานั้นจะเรียกว่าอเพอร์เจอร์ไทม์ (Aperture Time) ของแซมปลิงแอนด์โฮลด์



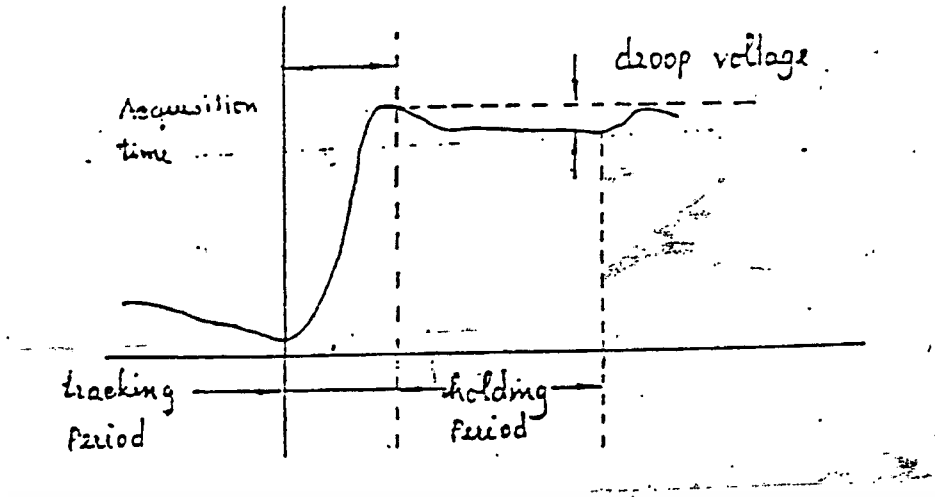
รูปที่ 2.17 แสดงวงจรแซมปลิงแอนด์โฮลด์ที่ใกล้เคียงกับวงจรที่ใช้งานจริง

จากรูปที่ 2.17 โดยการเพิ่มเติมบัฟเฟอร์แอมปลิฟลายเออร์ (Buffer Amplifier) เข้าทางส่วนส่วนอินพุท และเอาท์พุทของวงจรนี้ แอมปลิฟลายเออร์ทางด้านอินพุทจะช่วย

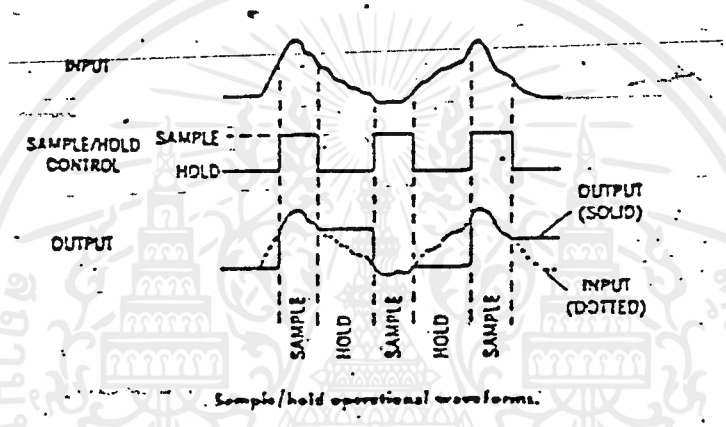
ใช้งาน ให้วงจรมีอินพุทอิมพีแดนซ์สูงและช่วยประจุตัว

เก็บประจุได้เร็วขึ้น ส่วนแอมปลิฟลายเออร์ทางด้านเอาท์พุท จะช่วยทำให้เอาท์พุทมีอิมพีแดนซ์ต่ำสามารถขับเอดีซีได้ง่าย ส่วนสำคัญที่ต้องพิจารณาคือ จะต้องใช้แอมปลิฟลายเออร์ที่กินกระแสต่ำเพื่อให้ดึงกระแสจากตัวเก็บประจุในช่วงโฮลด์สัญญาณน้อยที่สุด มิฉะนั้นจะเกิดการตก (Drop) แก่แรงดันที่โฮลด์ ดังแสดงในรูปที่ 2.18

ส่วนรูปที่ 2.19 แสดงรูปคลื่นสัญญาณที่เกิดจากวงจรแซมเปิลแอนด์ในทางอุดมคติ



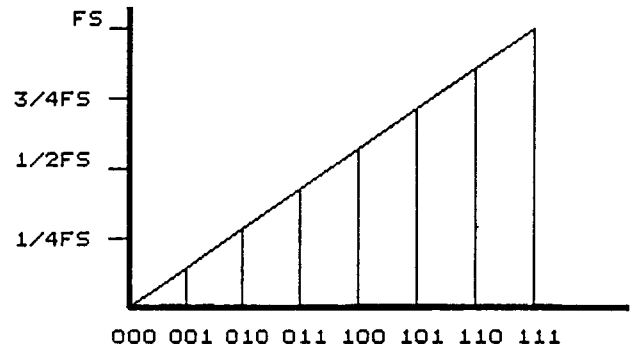
รูปที่ 2.18 แสดงการตกของแรงดันที่โฮลด์ไว้



รูปที่ 2.19 แสดงรูปคลื่นสัญญาณจากวงจรแซมเปิลแอนด์ในทางอุดมคติ

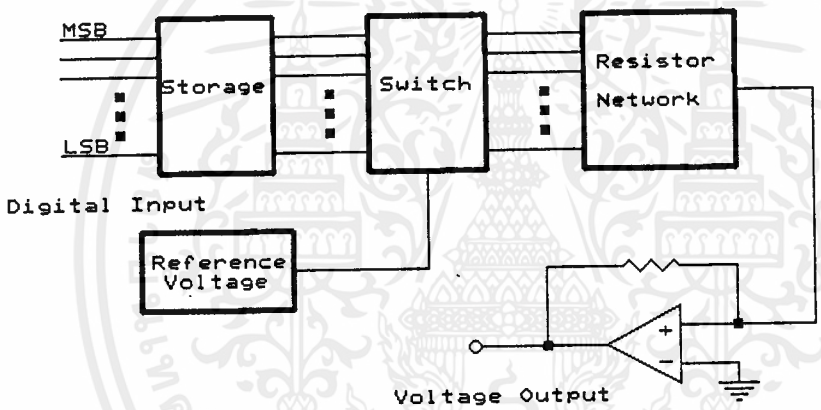
2.3.4 วงจรเปลี่ยนสัญญาณดิจิตอลให้เป็นอนาลอก (DAC)

ดีเอซี เป็นอุปกรณ์สำคัญอย่างหนึ่ง ที่ทำให้นำเอาคอมพิวเตอร์ไปใช้ร่วมกับอุปกรณ์หรือวงจรอนาลอกอื่นๆ รูปที่ 2.20 จะแสดงทรานสเฟอร์ฟังก์ชันของ ดีเอซีขนาด 3 บิท จะเห็นได้ว่ารหัสดิจิตอลอินพุท 1 เวิร์ด จะเปลี่ยนเป็นแรงดันอนาลอก 1 ค่า



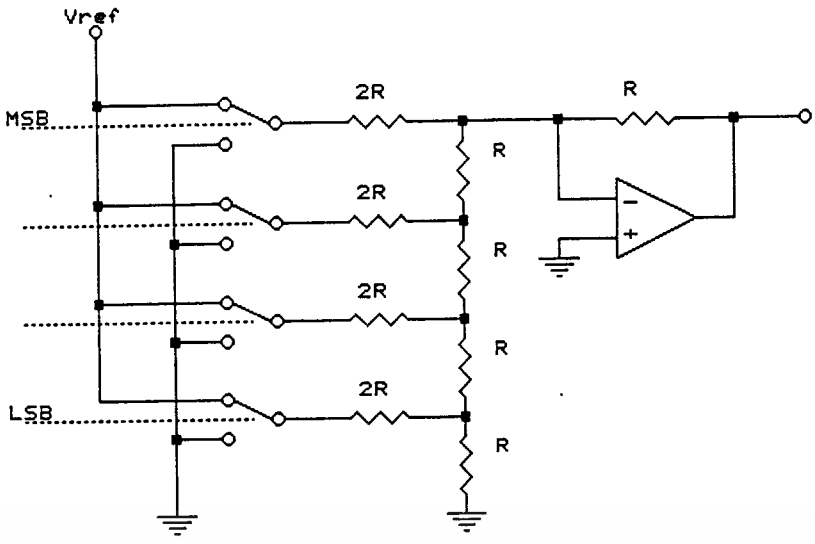
รูปที่ 2.20 แสดงทรานสเฟอร์ฟังก์ชันของดีเอซีขนาด 3 บิท

ลักษณะการจัดวงจร ดีเอซี จะเป็นลักษณะดังรูปที่ 2.21



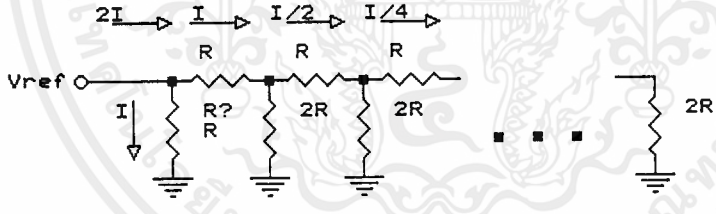
รูปที่ 2.21 แสดงบล็อกไดอะแกรมของ ดีเอซี

วงจรดีเอซี ที่ใช้กันมีหลายแบบแต่ในที่นี้จะขอกล่างถึง วงจร ดีเอซีที่เป็นแบบตัวต้านทาน ขั้นบันได (R-2R ladder) ซึ่งเป็นแบบที่นิยมใช้กันมาก ลักษณะของวงจรแสดงดังรูปที่ 2.22



รูปที่ 2.22 แสดงวงจรดีเอซีขนาด 4 บิท แบบ R-2R ladder

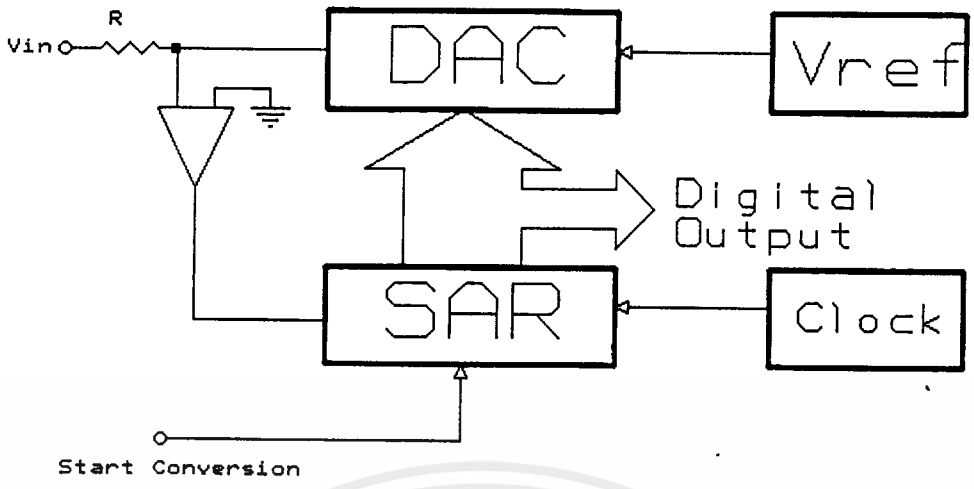
ในวงจรนี้ สวิทช์จะตัดต่อให้แรงดันอ้างอิง ต่อเข้ากับวงจรขั้นบันได จะเห็นได้ว่า สวิทช์อินพุตเป็นp-30Yทรีซิสเตอร์ (2R) เมื่อมองเข้าไปจะเห็นคู่ของรีซิสเตอร์ระหว่างจุดต่อ R-2R ที่ติดกัน กระแสจะถูกบั่นทอนไปในอัตรา 2/1 ซึ่งสอดคล้องกับ รหัสไบนารี ดังรูปที่ 2.23



รูปที่ 2.23 แสดงการแบ่งแรงดันจากแรงดันอ้างอิง

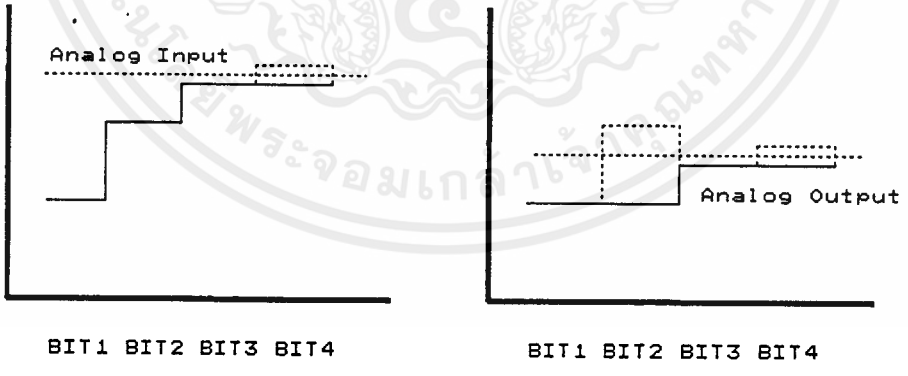
2.3.5 วงจรเปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล

วงจรเอดีซีส่วนใหญ่ จะอยู่ในรูปของไอซี และแบบที่นิยมใช้คือ แบบการประมาณซัคเซสซีพ (Successive Approximatin) ซึ่งนิยมใช้ในงานที่ต้องการความเร็วสูง และปานกลาง วงจรแบบนี้คล้ายกับเคาท์เตอร์ที่ทำงานในลักษณะป้อนกลับ



รูปที่ 2.24 แสดงบล็อกไดอะแกรม ของตัวแปลงแบบซัคเซสซีฟ

รูปที่ 2.24 แสดงฟังก์ชันต่างๆ ในเอดีซี คอมพาราเตอร์จะคอยเปรียบเทียบเอาท์พุทจากเอดีซี กับสัญญาณอนาลอกอินพุท เอาท์พุทที่ได้จะควบคุมรีจิสเตอร์ของตัวประมาณแบบซัคเซสซีฟ การทำงานของ SAR จะเป็นดังนี้ (ดูรูป 2.25)



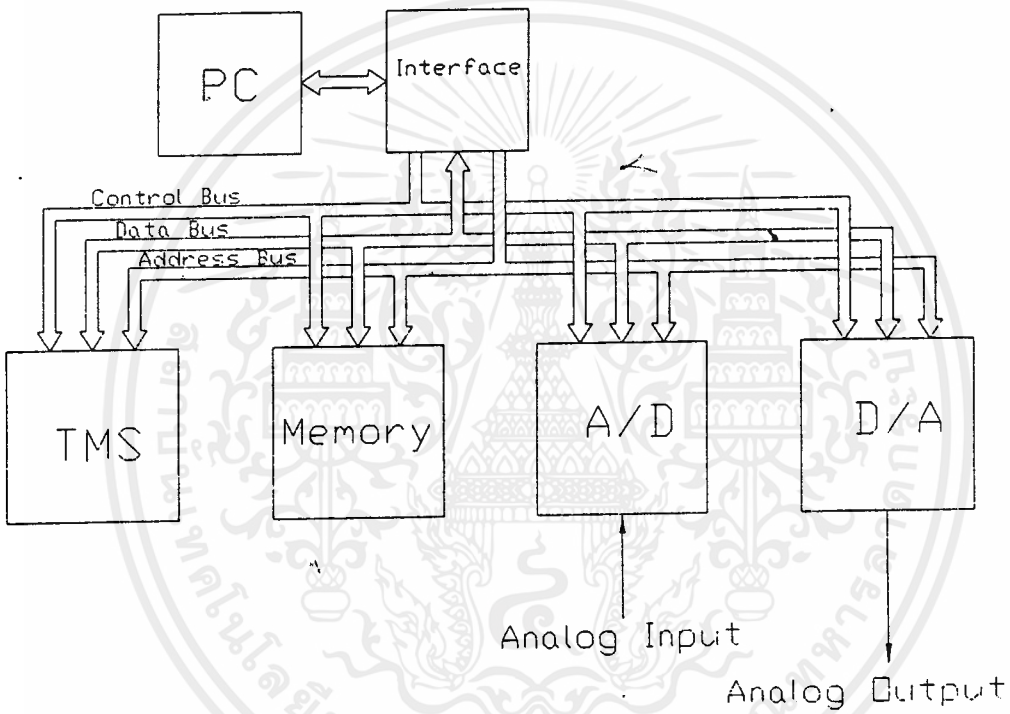
รูปที่ 2.25 แสดงไทม์มิงไดอะแกรมของเอดีซีแบบ SAR

เมื่อ สัญญาณนาฬิกาเข้ามา 1 ลูก ก็จะทำให้บิทสูงส+30V ด เป็น "1" ส่วนบิทอื่นๆยังคง เป็น "0" ดีเอซีจะเปลี่ยนเอาท์พุทของ SAR ให้เป็นสัญญาณอนาลอกเปรียบเทียบกับสัญญาณอินา ลอกอินพุท ถ้าผลการเปรียบเทียบที่ตัวเปรียบเทียบที่ตัวเปรียบเทียบ ว่าน้อยกว่าอินพุทก็คงบิทนั้น เป็น "1" แต่ถ้ามากกว่า ก็จะให้บิทนั้นเป็น "0" จากนั้นจะทำการทดสอบบิทถัดไป โดยทำให้ บิทนั้นเป็น "1" หากผลรวมของ สองบิท มีค่ามากกว่าก็ทำให้บิทนั้นเป็น "0" แต่ถ้าน้อยกว่าก็คง ค่า "1" เอาไว้ แล้วทำการทดสอบบิทถัดไป ด้วยวิธีดังกล่าวจนครบทุกบิท หรือจนกว่าเอาท์พุท จะต่างจาก ค่าแรงดันอินพุท ไม่เกิน "1" LSB

ข้อจำกัดประการหนึ่งสำหรับการทำงาน คืออนาลอกอินพุทจะต้องคงที่ ในช่วงเวลา ที่ทำการเปลี่ยนสัญญาณ โดยเปลี่ยนแปลงได้ไม่เกิน 1/2 LSB ในช่วงสุดท้ายของการเปลี่ยน สัญญาณดิจิตอล เอาท์พุทจะออกมาขนานกันทุกบิท

บทที่ 3 การออกแบบวงจรและการทำงาน

การออกแบบวงจรเราจะเริ่มจาก แผนผังเบื้องต้นของระบบประมวลผลสัญญาณดิจิตอล และเราต้องการให้สามารถเปลี่ยนแปลงแก้ไข อัลกอริทึมการประมวลผลได้ง่ายด้วย IBM/PC ในการจะติดต่อระหว่างตัวประมวลผลสัญญาณดิจิตอล กับ IBM/PC ได้นั้น เราจำเป็นจะต้องมีวงจรในส่วนที่ใช้ติดต่อ (Interface) จากนั้น เมื่อเรามองที่ตัวประมวลผล ซึ่งจะต้องประกอบด้วย หน่วยความจำ และจะมองสัญญาณอินพุทและเอาท์พุทของตัวประมวลผลในรูปของพอร์ท และการที่จะเข้าถึงหน่วยความจำที่เวลาใด ๆ ไม่สามารถทำได้มากกว่า 1 ทาง ดังนั้นจะต้องมีส่วนเลือกการติดต่อ เมื่อมองภาพโดยรวมจะเขียนเป็นบล็อกไดอะแกรมได้เป็น ดังรูปที่ 3.1



รูปที่ 3.1 แสดงบล็อกไดอะแกรมส่วนเลือกในการติดต่อระหว่างส่วนต่างๆ

จากรูปที่ 3.1 เนื่องจากต้องการให้ระบบมีความยืดหยุ่นสูง จึงได้รวมบล็อกไดอะแกรมที่อยู่ในเส้นประเข้าด้วยกันเท่านั้น และจะเรียกส่วนนี้ว่า เมนบอร์ด (Main Board)

จากที่กล่าวมาแล้ว จะเห็นว่าจำนวน หรือคุณสมบัติอินพุท เอาท์พุท ของระบบสามารถเปลี่ยนแปลงได้โดยง่าย ในการออกแบบจะทำการแยกส่วนต่างๆออกจากกันด้วยขั้วต่อ(Header)

เพื่อที่จะทำการถอดเปลี่ยนได้สะดวก และจะขอเรียกจุดต่อสัญญาณระหว่างส่วน Selector และ I/O พอร์ท ว่า Port Header และจุดต่อระหว่างส่วน Interface กับส่วน Selector ว่า Interface Header เพื่อความสะดวกในการกล่าวถึงต่อไป

เมื่อได้ดังนี้แล้ว ก็จะทำการออกแบบวงจรเป็นส่วนๆต่อไป ในการออกแบบนี้ก็จะคำนึงถึงค่าเวลาหน่วง (delay time) ของสัญญาณต่างๆด้วย

3.1 <u>ส่วนเมนบอร์ด</u>

ในส่วนของเมนบอร์ดนี้จะเป็นที่อยู่ของตัวประมวลผล คือตัว TMS 32010 ซึ่งมีดาต้าเมมโมรีและโปรแกรมเมมโมรีอยู่ภายในตัวเอง แต่เนื่องจากการใช้โปรแกรมเมมโมรีภายในตัวเองจะทำให้การเปลี่ยนแปลงแก้ไขโปรแกรมทำได้ไม่สะดวก ดังนั้นจึงจะใช้หน่วยความจำภายนอกแทนโปรแกรมเมมโมรีของ TMS32010 โดยการป้อนลอจิก "0" เข้าที่ขา MC/~MP ของ TMS32010 ซึ่งจะเป็นการบังคับให้ TMS32010 ทำงานในโหมดไมโครโปรเซสเซอร์

3.1.1 ส่วนซีเลคเตอร์

จากบล็อกไดอะแกรมข้างต้น การที่จะเลือกระหว่าง TMS กับ IBM สามารถสร้างได้โดย ใช้ซีเล็คเตอร์สองทางสำหรับระบบบัสต่างๆ โดยเมื่อเราจะเปลี่ยนโปรแกรมที่อยู่ในหน่วยความจำ ก็โดยให้ซีเล็คเตอร์ต่อหน่วยความจำเข้ากับ IBM และเมื่อจะให้ TMS ทำงานตามโปรแกรมนั้นก็โดยให้ซีเล็คเตอร์ต่อหน่วยความจำเข้ากับ TMS

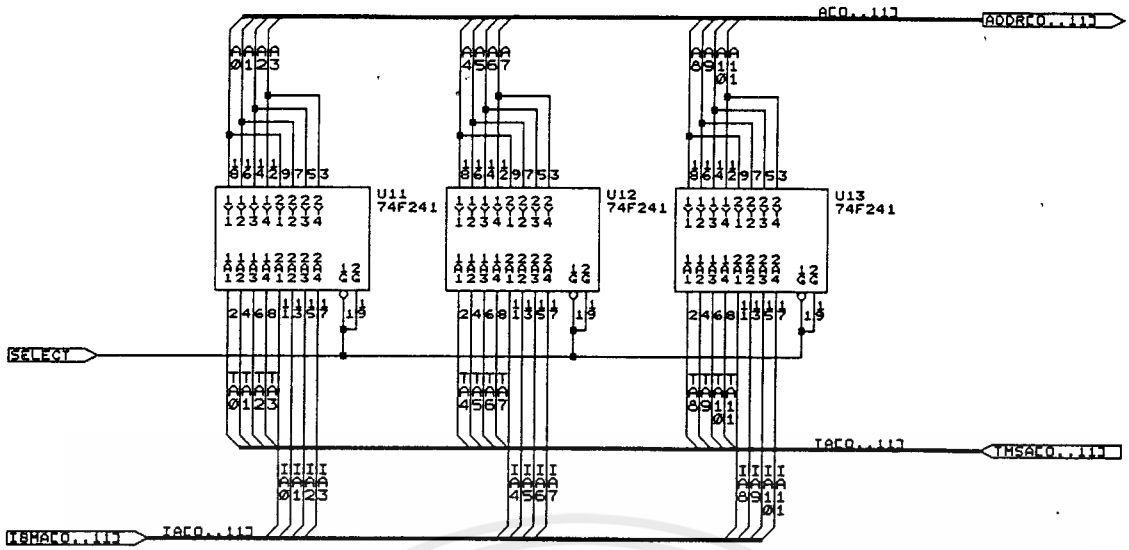
ในส่วนซีเล็คเตอร์จะประกอบด้วย

-แอดเดรสบัสซีเล็คเตอร์

-ดาต้าบัสซีเล็คเตอร์

-ซีเล็คเตอร์บัสสัญญาณควบคุมระบบ

ส่วนแอดเดรสบัสซีเล็คเตอร์จะต้องเป็นแบบทิศทางเดียว12 บิท เนื่องจากแอดเดรสบัสของ TMS32010 นั้นมี 12 บิท เราจะใช้ 74F241 ซึ่งเป็นแบบ 4 บิท 2 ทาง ดังนั้นจึงต้องใช้ 74F241 ทั้งหมด 3 ตัว เพื่อให้ได้ 12 บิท นำมาต่อกันดังรูปที่ 3.2



รูปที่ 3.2 แสดงการต่อ 74F241 ให้เป็นบัสซีเล็คเตอร์ 12 บิท

ส่วนดาต้าบัสซีเล็คเตอร์นั้น จะต้องเป็นแบบสองทิศทาง 16 บิท เนื่องจากดาต้าบัสของ TMS32010 นั้นเป็นแบบสองทิศทางและมี 16 บิท จากความเหมาะสมที่สุด เราจะเลือกใช้ 74F245 ซึ่งเป็นบัฟเฟอร์แบบ 8 บิท 3 สถานะสองทิศทาง นำมาต่อกันเป็นซีเล็คเตอร์สองทางได้ดังรูป 3.3

ส่วนซีเลคเตอร์บัสสัญญาณควบคุมระบบ

บัสสัญญาณควบคุมระบบจะประกอบด้วยสัญญาณต่างๆ ดังนี้

-สัญญาณอ่านอินพุทพอร์ท

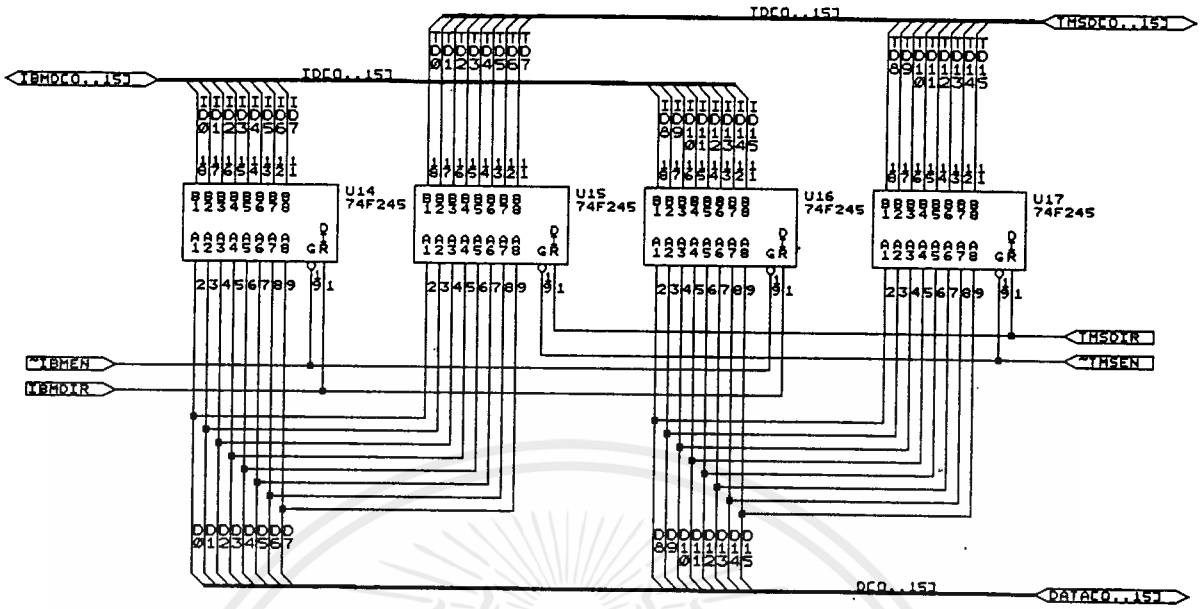
-สัญญาณเขียนเอาท์พุทพอร์ท

-สัญญาณอ่านหน่วยความจำ

-สัญญาณเขียนหน่วยความจำ

ซึ่งสามารถใช้ 72F157 เป็นซีเล็คเตอร์ได้พอดี

ทั้งสามส่วนดังกล่าวนี้จะต้องทำงานสัมพันธ์กัน โดยจะมีสัญญาณควบคุมการเลือกมาจากสัญญาณ IBM/~TMS ซึ่งสัญญาณนี้จะมีลอจิก "1" เมื่อ IBM ต้องการติดต่อกับหน่วยความจำ



รูป 3.3 แสดงการต่อ 74F245 เป็นคาต้าบัสซีเล็คเตอร์

3.1.2 สัญญาณรีเซ็ทระบบ (~RS)

สัญญาณรีเซ็ทระบบจะเป็นสัญญาณที่ใช้รีเซ็ท TMS 32010 และนอกจากนั้นการรีเซ็ทยังมีความจำเป็นเพื่อให้ระบบกลับสู่จุดเริ่มต้น สัญญาณรีเซ็ทนี้จะเกิดขึ้นเมื่อ

- เริ่มจ่ายไฟเลี้ยงให้ระบบ (Power On ~RS)
- กดปุ่มรีเซ็ท (Manual ~RS)
- IBM กำลังติดต่อกับหน่วยความจำและพอร์ท (Soft ~RS)

และเราจะให้ระบบรีเซ็ทที่ลอจิก"0" ดังนั้นสัญญาณรีเซ็ทจะกำหนดได้โดย

~RS = [(Power On ~RS) AND (Manual ~RS) AND NOT (I/~T)]

สัญญาณ [(Power On ~RS) AND (Manual ~RS)] จะถูกสร้างโดยลอจิกเกต และค่า R, C วงจรจะอยู่ในภาคผนวก ก ในแผ่นวงจรชื่อ Power On Reset

เพื่อที่ให้ความกว้างของสัญญาณที่เกิดขึ้นมีค่ามากกว่า 1 ไมโครวินาที

เพราะจากการทำงานของอุปกรณ์ที่ต่ออยู่บน Port Header บางอย่างที่ต้องการสัญญาณ

รีเซ็ทเพื่อกำหนดจุดเริ่มต้นของการทำงานให้สัมพันธ์ กับระบบ เช่น ส่วนแปลงสัญญาณ จากอนาลอกเป็นดิจิตอล ถ้า IBM จะติดต่อกับอุปกรณ์ดังกล่าวก็จะทำไม่ได้เพราะ สัญญาณรีเซ็ทที่ได้จากสมการจะมีอยู่ตลอดเวลา อุปกรณ์นั้นก็จะไม่ทำงาน เราจึงต้องสร้างสัญญาณขึ้นมาใหม่จากสัญญาณรีเซ็ทเพื่อใช้กับอุปกรณ์บน Port Header เราจะกำหนดให้สัญญาณนี้เป็น ~RS' โดยที่เมื่อเกิดสัญญาณ ~RS ก็จะเกิดสัญญาณ ~RS' ขึ้นชั่วขณะเท่านั้น การสร้างสัญญาณนี้ทำโดยใช้ไอซี 74LS123 มาทำเป็นวงจรไมโนสเตเบิล (Monostable)

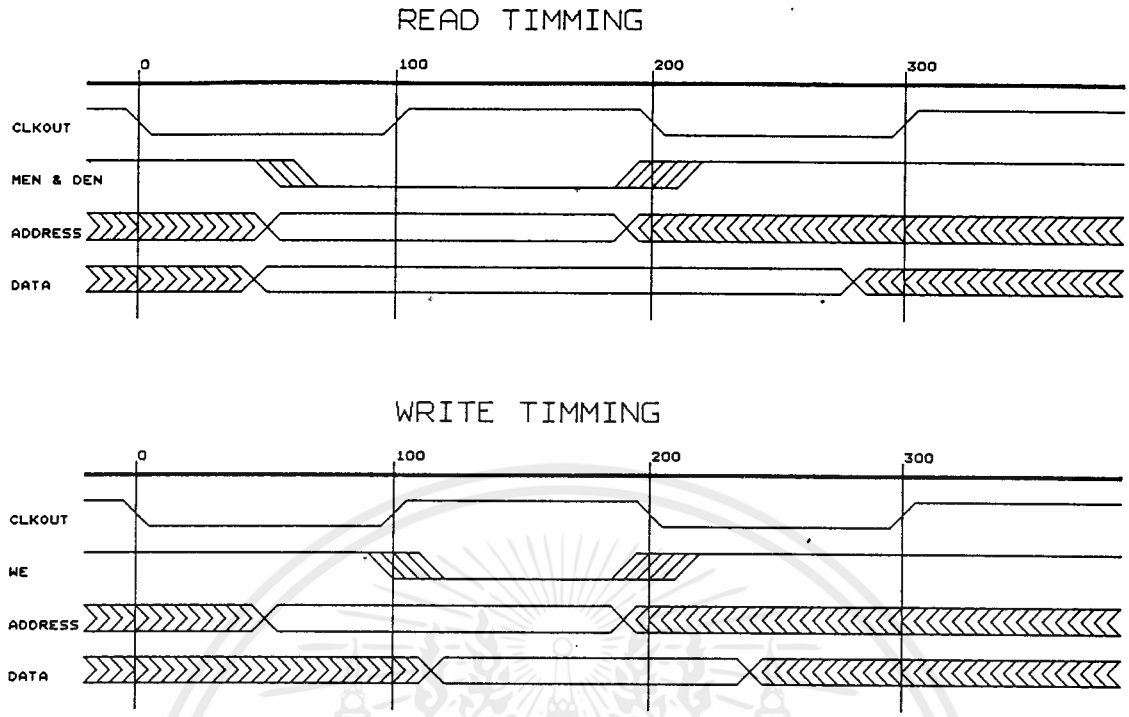
3.1.3 สัญญาณ ~BIO และ ~INT

สัญญาณนี้จะมีประโยชน์มากในการใช้งานโดยเฉพาะ สัญญาณ ~BIO ซึ่งจะเป็นอินพุทของ IMS 32010 ที่ทำให้เราสามารถหยุดคอยได้ เช่นการกำหนดความถี่ของการสุ่มเป็นต้น การรับข้อมูลของ IMS จะรับเพียงช่วงแคบ ดังนั้นการที่จะทำให้สัญญาณที่ถูกส่งเข้าไปที่เวลาใดๆ สามารถถูก IMS รับรู้ได้ จะต้องมีการซิงโครไนซ์ข้อมูลที่เข้ามากับการทำงานของ IMS โดยใช้ D-Flip Flop 74AS74(U3A,U3B) ที่ต่อขา CLK เข้ากับการทำงานของระบบ สัญญาณนาฬิกาของระบบจะได้จากการ ออสซิลเลทของคริสตอล (Crystal) 20 MHz แล้ว IMS จะหารให้เหลือ 5 MHz เพื่อเป็นสัญญาณนาฬิกาของระบบ

3.1.4 ~MEN , ~DEN , ~WE และการเขียนอ่านพอร์ท

จากการทำงานของ IMS32010 ซึ่งมีไทม์มิงไดอะแกรมดังรูปที่ 3.4 จะเห็นได้ว่าเมื่อ IMS ต้องการอ่านหน่วยความจำ สัญญาณ ~MEM จะทำงานและเมื่อ IMS ต้องการอ่านพอร์ท สัญญาณ ~DEN จะทำงาน แต่เมื่อสัญญาณ ~WE ทำงานนั้น จะหมายถึงว่า IMS ต้องการจะเขียนหน่วยความจำหรือพอร์ท อย่างใดอย่างหนึ่ง ดังนั้นเราจำเป็นต้องแยกสัญญาณ ~WE นี้ออกเป็น สัญญาณเขียนพอร์ท และ สัญญาณเขียนหน่วยความจำ โดยอาศัยว่าพอร์ทของ IMS นั้นมีอยู่เพียง 8 พอร์ท จึงออกแบบวงจรให้เมื่อสัญญาณ ~WE ทำงาน และค่าแอดเดรสมีค่ามากกว่า 8 ก็แสดงว่า IMS ต้องการจะเขียนข้อมูลลงหน่วยความจำ แต่ถ้าค่าของแอดเดรสขณะที่สัญญาณ ~WE ทำงาน มีค่าน้อยกว่าหรือเท่ากับ 8 ก็ย่อมแสดงว่า IMS ต้องการจะส่งข้อมูล

ออกไปที่พอร์ท



รูปที่ 3.4 แสดงสัญญาณการอ่านและการเขียน หน่วยความจำและพอร์ท

จากข้างต้นเราจะออกแบบเป็นวงจรได้โดยใช้วิธี คาร์นอร์แมบ(Karnaugh-Map) ทำการลดรูปของสมการ ได้ว่า สัญญาณ ~CS (Chip Select) ของหน่วยความจำคือ

~CS = (~MEN) AND NOT (A3 to A11=0)

ซึ่งสัญญาณ ~CS ก็จะต้องไปผ่านชุด Selector อีกที โดยใช้ไอซี 74F260,74F08 ซึ่งจะอธิบายการทำงานได้ดังตาราง 1

การเขียนอ่านพอร์ท จะกระทำโดย ก่อนอื่นเราต้องทำการแยก (Decode) ตำแหน่งของพอร์ท โดยค่าตำแหน่งของพอร์ทจะอยู่ในแอดเดรสบัส A0-A2 โดยใช้ 74F138 สองเกท เพื่อแยกเป็นพอร์ทสำหรับรับข้อมูลเข้าและพอร์ทสำหรับส่งข้อมูลออก ก็จะได้สัญญาณเลือกพอร์ท(~PS) ต่างๆออกมา ซึ่งสัญญาณเลือกพอร์ทจะต้องไม่เกิดขึ้นเมื่อค่า A3 ถึง A11 ไม่เท่ากับศูนย์ เราจึงต่อ NOT (A3 to A11 = 0) เข้ากับขา G1 ของ 74F138 ทั้งสองตัวและแต่ละตัวก็จะถูกควบคุมแยกกันด้วยสัญญาณ ~WE และ ~DEN เพื่อการส่งและรับข้อมูล

เมื่อรวมส่วนต่างๆเข้าด้วยกันก็จะได้วงจรบนเมนบอร์ดดังภาคผนวก ก

ตารางที่ 1 แสดงการใช้งานหน่วยความจำ

A3-A11	~MEN	~WE	~CS	การทำงาน
0	0	1	0	เป็นการอ่านข้อมูลจาก RAM
0	1	0	1	เป็นการเขียนข้อมูลแต่จะไม่เข้าRAM
1	0	1	0	เป็นการอ่านข้อมูล จาก RAM
1	1	0	0	เป็นการเขียนข้อมูลเข้าสู่ RAM

3.2 ส่วนหน่วยความจำ

จากที่ TMS32010 ทำงานด้วยความเร็วสูงและจะอ่านหน่วยความจำทุกๆ รอบของการทำงาน ทำให้จะต้องใช้หน่วยความจำที่มีความเร็วการทำงานสูงด้วย ในที่นี้เราจะเลือกใช้ไอซีหน่วยความจำเบอร์ MT5C2568 ซึ่งเป็นหน่วยความจำแบบสแตติก (Static RAM) มีขนาด 32K * 8 บิท และมีเวลาในการเข้าถึงข้อมูลไม่เกิน 20 นาโน วินาที ดังนั้นเพื่อที่จะให้ได้หน่วยความจำครบ 32 กิโลเวิร์ด ก็จะต้องใช้ไอซี MT5C2568 จำนวนทั้งหมด 2 ตัว นำมาต่อกันแบบขนาน

3.3 วงจรกำหนดความถี่สุ่ม (Time Base)

วงจรกำหนดความถี่สุ่ม เป็นอุปกรณ์ที่ใช้บน Port Header จุดประสงค์ของการสร้างวงจรนี้ขึ้นมาก็เพื่อเป็นตัวสร้างความถี่ที่ใช้ในการสุ่มข้อมูลให้ TMS โดยจะอาศัยขาสัญญาณ ~BIO ซึ่งสามารถใช้ซอฟแวร์เป็นตัวตรวจสอบได้ โดยเราจะเขียนโปรแกรมให้ TMS รอคอยสัญญาณนี้ก่อนที่จะรับข้อมูลแซมเปิลต่อไปได้ โดยที่ค่าเวลานี้สามารถกำหนดได้ด้วยซอฟแวร์

โดยอาศัยวงจรนับถอยหลังที่สามารถกำหนดค่าเริ่มต้นได้ 4526 ซึ่งเป็นตัวนับแบบ 4 บิท 2 ตัว เพื่อที่จะสามารถกำหนดความถี่ได้ในช่วงที่กว้าง สัญญาณนาฬิกาของระบบซึ่งมีขนาด 5 MHz จะถูกหารด้วย 5 (การหารเป็นการต่อ Count Down Counter แบบหนึ่ง) ด้วยไอซี

74LS193 ได้สัญญาณเอาท์พุท 1 MHz แล้วไปผ่านวงจรหารที่โปรแกรมได้ (4526) ซึ่งค่าที่จะใช้หารจะถูกเก็บอยู่ใน 74LS373 ดังนั้นถ้าเราต้องการความถี่เอาท์พุท 100 KHz เราก็ต้องส่งค่า 10 ไปเก็บไว้ใน 74LS373 ในรูปแบบเลขฐานสองก่อน ดังนั้นจะเห็นได้ว่า เราสามารถจะหารได้มากที่สุดคือ หารด้วย 256 จากความถี่อินพุท ดังนั้นความถี่สุ่มต่ำสุดที่จะทำได้คือ 1M/256 Hz ซึ่งจะเท่ากับ 3.921 KHz แต่โดยธรรมดาแล้ว เราจะกำหนดค่าของความถี่สุ่มเป็นเลขจำนวนเต็ม เอาท์พุทที่ได้จะนำไปเป็นสัญญาณทริกให้ 74LS122 (Monostable) เพื่อสร้างสัญญาณพัลซ์ที่มีความกว้างคงที่ แล้วค่อยส่งเข้าขา ~BIO อีกที การส่งค่ามาเก็บไว้ใน 74LS373 ก็คือการส่งข้อมูลออกไปที่ Port Header นั่นเอง

3.4 <u>ส่วนแปลงสัญญาณอนาลอกเป็นดิจิตอล</u>

ส่วนแปลงสัญญาณอนาลอกเป็นดิจิตอลนี้จะเป็นอุปกรณ์ที่ใช้บน Port Header ซึ่งจะเป็นพอร์ทรับข้อมูลเข้าอย่างหนึ่งของระบบประมวลผล

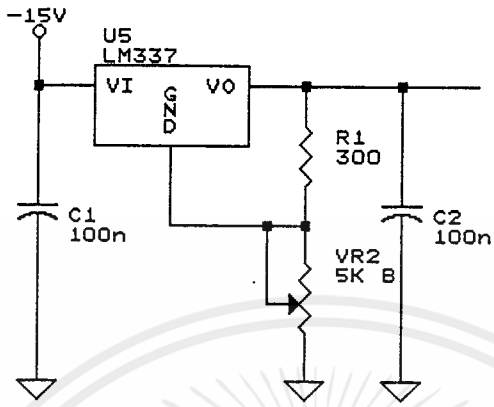
เราจะเลือกใช้ ADC เบอร์ ADC908 ซึ่งมีคอนเวอร์ชั่นไทม์(Convertion Time)เท่ากับ 6 ไมโครวินาที ดาต้า 8 บิท สำหรับเหตุผลที่เลือกใช้ ADC เบอร์นี้ ก็คือการทำงานของ IMS นั้นมีความเร็วสูงมาก ดังนั้นจึงต้องการ ADC ที่ทำงานเร็วทัน และการเลือกใช้ ADCที่มีดาต้าแบบ 16 บิท นั้นจะมีราคาแพง

จากสัญญาณอินพุทซึ่งมีความถี่สูง จึงต้องมีวงจรส่วนแซมเปิล แอนด์ โฮล (Sample and Hold) ซึ่งจะใช้ IC LF398 ซึ่งจากดาต้า บุ๊ค ของ LF398 เราจะสามารถเลือกค่าตัวเก็บประจุที่จะใช้ได้จากกราฟ Acquisition Time ในดาต้า บุค จะได้ค่า ตัวเก็บประจุที่ต้องใช้เท่ากับ 1 นาโน ฟารัด เพื่อความเร็วที่สูงที่สุด

สำหรับสัญญาณที่จะควบคุม LF398 ว่าจะให้ แซมเปิล หรือ โฮล ก็นำมาจากขาสัญญาณ ~BUSY ของ ADC908 ซึ่งจากการทำงานของ ADC908 นี้ เมื่อ ADC กำลังทำการแปลงสัญญาณอยู่ ที่ขา ~BUSY นี้จะมีลอจิก "0" ซึ่งในช่วงนี้ LF398 ก็จะต้องโฮลด์ข้อมูลไว้ จะเห็นได้ว่าสัญญาณที่จะควบคุม LF398 ก็ต้องการลอจิก "0" เช่นเดียวกัน ดังนั้นจึงสามารถต่อขาสัญญาณที่จะควบคุม LF398 เข้ากับขา ~BUSY ของ ADC908 ได้โดยตรง

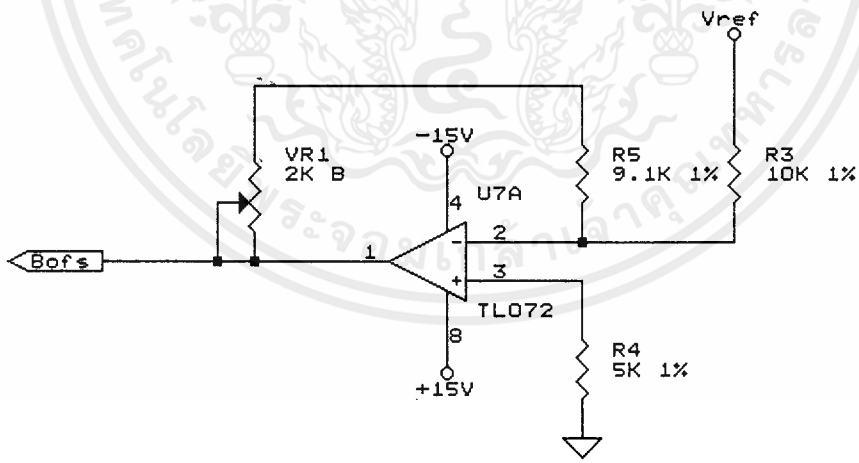
ADC908 นี้จะมีสัญญาณอินพุทอยู่ 2 โหมด คือ โหมดยูนิโพล่าร์ (Unipolar Mode) และโหมดไบโพล่าร์(Bipolar Mode) ในโครงงงานนี้เราจะเลือกการใช้งานในโหมดไบโพล่าร์

เพื่อให้สัญญาณอินพุทอนาลอกสามารถเป็นได้ทั้งบวกและลบ มีขนาดสัญญาณไม่เกิน +10 โวลต์และ -10 โวลต์ ดังนั้นจะต้องใช้แรงดันไฟเปรียบเทียบ (V_{REF}) ขนาด -10 โวลต์ ซึ่งสามารถสร้างได้โดยใช้ไอซี LM337 ซึ่งเป็นไอซีที่สามารถปรับค่าแรงดันได้ นำมาต่อเป็นวงจรดังรูป 3.5



รูปที่ 3.5 แสดงการสร้างสัญญาณอ้างอิงโวลต์เตจ

และสำหรับสัญญาณ B_{OFS} เพื่อทำการปรับออฟเซ็ทของ ADC908 นั้นสามารถสร้างได้โดยใช้วงจรดังรูปที่ 3.6



รูปที่ 3.6 แสดงการต่อวงจรเพื่อสร้างสัญญาณ B_{ofs}

จากการทำงานของ ADC908 ซึ่งจะมีการทำงานอยู่ 3 โหมด คือ

- โหมดการทำงานแบบ RAM
- โหมดการทำงานแบบ ROM
- โหมดการทำงานแบบ Slow-Memory

ตารางการทำงาน ของทั้ง 3 โหมด แสดงดังตารางที่ 2,3 และ 4

ตารางที่ 2 แสดงการทำงานของ ADC908 ในโหมด RAM

อินพุท		เอาท์พุท		การทำงาน
~CS	~RD	~BUSY	บัสข้อมูล	
L	H	H	Z	เริ่มทำการแปลงข้อมูล
L	H-L	H	Z-D	อ่านข้อมูล
L	L-H	H	D-Z	รีเซทตัวแปลงข้อมูล
H	X	X	Z	ไม่ใช้
L	H	L	Z	ไม่ใช้
L	H-L	L	Z	ไม่ใช้
L	L-H	L	Z	ไม่อนุญาตให้เกิดขึ้น

(Z: ไฮอิมพีแดนซ์, H-L: ขอบขาลง, L-H: ขอบขาขึ้น, X: ไม่สนใจ, D: มีข้อมูล)

ตารางที่ 3 แสดงการทำงานของ ADC908 ในโหมด ROM

อินพุท		เอาท์พุท		การทำงาน
~CS	~RD	~BUSY	บัสข้อมูล	
L	H-L	H	Z-D	อ่านข้อมูล
L	L-H	H-L	D-Z	รีเซทและเริ่มแปลงข้อมูล
L	H-L	L	Z	ไม่ใช้
L	L-H	L	Z	ไม่อนุญาตให้เกิดขึ้น

ตารางที่ 4 แสดงการทำงานของ ADC908 ในโหมด Slow-Memory

อินพุท	เอาท์พุท		การทำงาน
~CS & ~RD	~BUSY	บัสข้อมูล	
H	H	Z	ไม่ใช้
H-L	H-L	Z	เริ่มแปลงข้อมูล
L	L	Z	รอการแปลงข้อมูล
L	L-H	Z-D	แปลงข้อมูลเสร็จ, อ่านข้อมูลได้
L-H	H	D-Z	รีเซทและยกเลิกการใช้

จากตารางการทำงานจะเห็นได้ว่า ในโหมด RAM นั้น จะต้องมีสัญญาณในการควบคุม ADC908 มาก ซึ่งจะยุ่งยากในการออกแบบวงจร และในโหมด Slow-Memory ก็จะทำให้TMS เสียเวลาคอยการทำงานของ ADC908 ในช่วงของการแปลงสัญญาณ แต่การใช้งานADC908 ในโหมด ROM นั้นดูจะมีความเหมาะสมมากที่สุด

การที่จะใช้งาน ADC908 ในโหมด ROM นั้นก็เพียงแต่ป้อนลอจิก "0" เข้าที่ขา ~CS และเมื่อเราต้องการจะอ่านข้อมูลจาก ADC908 ก็ป้อนลอจิก "0" เข้าที่ขา ~RD หลังจากที่เราทำการอ่านข้อมูลเสร็จแล้ว ADC908 ก็จะทำการแปลงข้อมูลชุดต่อไป

จาก ADC908 ที่ดาต้าแบบ 8 บิท จะต้องอินเตอร์เฟสกับ ดาต้าบัสของ TMS ที่เป็นแบบ 16 บิท เราจะกำหนดให้ข้อมูล 8 บิทล่างมีค่าเท่ากับข้อมูลที่ได้จาก ADC908 ส่วนข้อมูล 8บิทบนจะมีค่าเท่ากับ "00000000" โดยใช้ 74LS244 ที่ต่อขาอินพุททั้งหมดลงกราวด์

และจากไทม์มิงไดอะแกรมการทำงานของ ADC908 ในโหมด ROM และจากข้อมูลในดาต้าบุ๊คของ ADC908 จะเห็นได้ว่าเมื่อเราทำการอ่านข้อมูลจาก ADC908 แล้วเลิกอ่านข้อมูลนั้นจะยังคงค้างอยู่ที่ขาดาต้าของ ADC908 อีกชั่วระยะเวลาหนึ่ง ซึ่งระยะเวลานี้มีค่านานเกินไปสำหรับดาต้าบัสของ TMS ดังนั้นจึงต้องใช้ 74F373 มาทำการเชื่อมต่อระหว่างขาดาต้าของ ADC908 กับดาต้าบัสของ TMS 8 บิทล่าง

เมื่อรวมส่วนต่างๆ ดังที่กล่าวมาข้างต้น ก็จะได้วงจรของส่วนแปลงสัญญาณ ดังรูปในภาคผนวก ก ซึ่งสามารถอธิบายการทำงานได้ดังนี้

- เมื่อเกิดการรีเซ็ท D Flip Flop จะถูกเคลียร์(Q=0,~Q=1) ซึ่งจะทำให้ ADC อยู่ในสภาวะอ่านข้อมูลสัญญาณ ~BUSY จะเป็น"1" และจาก ~Q ="0" 74LS373 ก็จะรับข้อมูลมาจาก ADC

- เมื่อมีสัญญาณอ่านพอร์ท (~PS) เข้ามา เมื่อ ~PS เป็นศูนย์ 74AS74 จะถูกพรีเซ็ท (PRESET) Q=1,~Q=0 เมื่อ Q เปลี่ยนจาก"0"เป็น"1" ADC ก็จะเริ่มทำการแปลงข้อมูล สัญญาณ ~BUSY จะเปลี่ยนจาก"1"เป็น"0" ทำให้ LF398 โฮลสัญญาณอนาลอกไว้ และจาก ~Q="0" ดังนั้น 74LS373 ก็จะไม่รับข้อมูลจาก ADC เมื่อการแปลงข้อมูลเสร็จสิ้น สัญญาณ ~BUSY จะเปลี่ยนจาก "0" เป็น "1" ทำให้ 74AS74 รับข้อมูลจากขา D มาออกที่ขา Q ซึ่งข้อมูลที่ D="0" ดังนั้น Q="0" ,~Q="1" ซึ่งก็จะเป็นการอ่านข้อมูล ข้อมูลจึงถูกส่งไปเก็บใน 74LS373 แล้วก็คอยการอ่านเข้าไปประมวลผลครั้งต่อไป

3.5 <u>ส่วนแปลงสัญญาณจากดิจิตอลเป็นอนาลอก</u>

ส่วนนี้เป็นอุปกรณ์อย่างหนึ่งบน Port Header จะเป็นชนิดส่งข้อมูลออกจากระบบประมวลผล โดยส่วนนี้จะรับข้อมูลมาจาก TMS32010

ในส่วนนี้เราจะเลือกใช้ไอซีเบอร์ DAC0800 ซึ่งเป็นไอซีแปลงสัญญาณดิจิตอล 8 บิท ให้เป็นสัญญาณอนาลอก มีเซตตลิ้งไทม์เท่ากับ 100 นาโนวินาที จากการทำงานของ DAC0800 ซึ่งจะแปลงข้อมูลดิจิตอลที่อยู่ที่ขาดาต้าของ DAC0800 ดังนั้นเพื่อความต่อเนื่องของสัญญาณเอาท์พุทอนาลอก ข้อมูลที่ขาดาต้าของ DAC0800 จึงจำเป็นต้องมีอยู่ตลอดเวลา แต่ข้อมูลที่มาจากดาต้าบัส ของ TMS จะอยู่เพียงชั่วระยะเวลาหนึ่งเท่านั้น จึงจะต้องมีวงจรที่จะเก็บค่าข้อมูลนั้นเอาไว้โดยใช้ไอซี 74LS373 แล้ว DAC0800 ค่อยนำข้อมูลจากเอาท์พุทของ 74LS373 มาแปลงเป็นอนาลอกอีกทีหนึ่ง ซึ่งผลของการแปลงเป็นอนาลอกของ DAC0800 จะได้อยู่ในรูปของกระแส ดังนั้นจึงต้องมีวงจรส่วนแปลงค่ากระแสที่ได้ ออกมาเป็นแรงดันอีกทีหนึ่ง วงจรจะอยู่ในภาคผนวก ก ซึ่งเราจะคำนวณได้ดังนี้

I_{FS} = (V_{REF} / R_{REF}) * (255/256)

I_{FS} = I_o + $\tilde{I}_o$

V_{REF} = 10.000 V

R_{REF} = 5.000 KOhm

R_{15} = REF

C_c = 0.01 uF

V_{LC} = 0 V

ส่วนออปแอมป์นั้น จะเป็นตัวแปลงสัญญาณที่อยู่ในรูปของกระแสให้เป็นสัญญาณแรงดัน โดยสัญญาณเอาท์พุทที่ได้จะเป็นแบบไบโพล่าร์

ส่วนสัญญาณ V_{REF} 10.000 โวลท์ สามารถสร้างได้โดยใช้ LM317

โดยเอาท์พุทสัญญาณอนาลอกที่ได้จากวงจรเปลี่ยนกระแสเป็นแรงดันนั้น จะมีลักษณะเป็นขั้นๆที่ไม่ต่อเนื่องกัน ดังนั้นเราก็จะขจัดความถี่เพื่อให้เอาท์พุทที่ได้มีลักษณะราบเรียบ โดยใช้วงจร Low Pass Filter เราจะตั้งความถี่คัทออฟเท่ากับ 100 KHz เนื่องจากระบบของเราทำงานได้ในช่วงความถี่ที่กว้าง เราก็จะได้วงจรรวม ดังแสดงในภาคผนวก ก

3.6 <u>ส่วนติดต่อกับ IBM PC (Interface)</u>

การติอต่อกับ IBM PC สามารถทำได้หลายทาง เช่น การติดต่อผ่านพอร์ทอนุกรม (RS-232) การติดต่อผ่านพอร์ทที่สร้างขึ้นโดยตรง และ การติดต่อผ่านพอร์ทขนาน (IBM PRINTER PORT) และอื่นๆ

จากการติดต่อกับ IBM ในหลายๆวิธีที่กล่าวมาข้างต้น ด้วยความเหมาะสมและเหตุผลต่างๆ เราจะเลือกการติดต่อโดยจะใช้การติดต่อผ่านทางบัสของ IBM PC โดยตรง โดยใช้ 8255 เป็น Peripheral device ซึ่งมีการนำเอาแอดเดรสบัสมาผ่านวงจร decode ทำให้ได้หมายเลขพอร์ทของ 8255 เป็น 0FF0h-0FF3h และ 0FF4h-0FF7h ตามลำดับ

สำหรับการทำงานของ 8255 นั้นเรากำหนดให้ทำงานในโหมด 0. สัญญาณต่าง ๆ ที่ได้จาก 8255 เป็นดังนี้

1. D0...D15 มาจาก พอร์ท A และ พอร์ท B ของ 8255 ตัวที่1
2. A0...A11 มาจาก พอร์ท A และ พอร์ท B (PB0..PB3) ของ 8255 ตัวที่2
3. สัญญาณคอนโทรล มาจากพอร์ท C ของ 8255 ตัวที่ 2

PC0 = สัญญาณ ~IRAN

PC1 = สัญญาณ R/~W

PC2 = สัญญาณ IBM/~TMS

PC3 = สัญญาณ P/~R

การที่จะกำหนดว่า พอร์ทใดควรเป็นอินพุทหรือเอาท์พุทพอร์ทนั้น จะกำหนดด้วยซอฟแวร์ ซึ่งจะกล่าวต่อไป

บทที่ 4 ตัวกรองไม่ป้อนกลับเชิงเลข

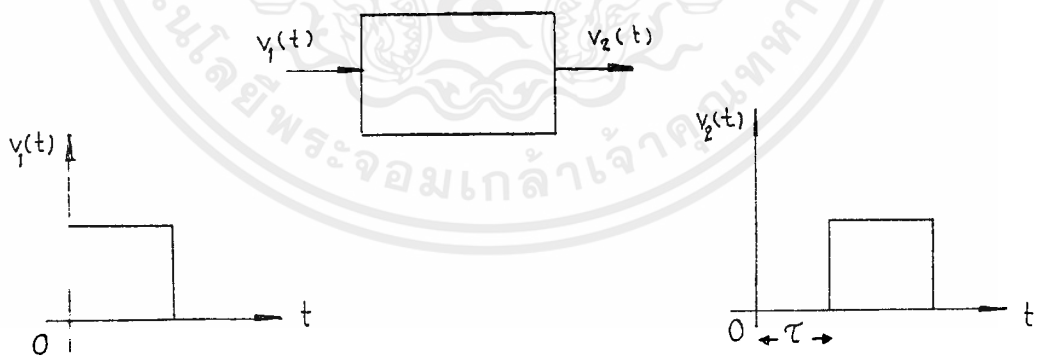
Non-Recursive Filter

ในการออกแบบตัวกรองเชิงอุปมาน หรือตัวกรองเชิงเลขโดยทั่วไป เรามักให้ความสำคัญแก่ผลตอบสนองแอมพลิจูดมากกว่า นั่นคือ ออกแบบให้มีผลตอบสนองแอมพลิจูดตามต้องการโดยยอมให้ผลตอบสนองเฟสเป็นอย่างไรก็ได้ แต่ในงานบางลักษณะนั้น เช่น การประมวลสัญญาณ อีซีจี (ECG: electrocardiogram) ที่ขนาดและลักษณะรูปคลื่นของสัญญาณมีความสำคัญมาก ในสถานการณ์แบบนี้ถ้าใช้ตัวกรองที่มีผลตอบสนองเฟสไม่เป็นเชิงเส้นมาทำการประมวลผลสัญญาณ ก็จะทำให้ค่าขนาด ลักษณะรูปคลื่น หรือเวลาในการเกิดสัญญาณนี้ผิดเพี้ยนไปได้ ดังนั้นจึงต้องใช้ตัวกรองไม่ป้อนกลับเชิงเลข ซึ่งสามารถออกแบบให้มีผลตอบสนองเฟสเป็นเชิงเส้นได้ง่าย

4.1 วงจรกรองแบบหน่วงเวลา (Delay Filters)

วงจรกรองหน่วงเวลา มีคุณสมบัติดังแสดงในรูปที่ 1 คือ ยอมให้สัญญาณผ่านออกไปได้โดยที่ลักษณะรูปคลื่นของสัญญาณออกไม่ผิดเพี้ยน เพียงแต่เวลาของการเกิดรูปคลื่นจะมีเวลาหน่วงออกไป τ วินาที โดยที่ τ เป็นค่าเวลาคงตัวที่กำหนดให้ ถ้าให้ $v_2(t)$ เป็นสัญญาณออกจากวงจรกรอง และ $v_1(t)$ เป็นสัญญาณเข้าของวงจรกรอง เราสามารถเขียนความสัมพันธ์ของทั้งสองสัญญาณนี้ได้เป็น

$$v_2(t) = v_1(t-\tau) \qquad (4.1.1)$$



รูป 4.5.1 แสดงคุณสมบัติของตัวกรองหน่วงเวลา

จากการวิเคราะห์โดยการประยุกต์ใช้ผลการแปลงฟูริเยอร์ ทำให้เรารู้ว่าสัญญาณรูปสี่เหลี่ยมตามรูป 4.1.1 สามารถแตกย่อยออกเป็น สัญญาณซายน์จำนวนอนันต์พจน์มาประกอบรวมกัน ดังนั้น เราสา

มารถศึกษาถึงผลตอบสนองเฟสของวงจรโดยนำมาพิจารณาแต่ละความถี่ได้ ถ้าเรานำสัญญาณที่ความถี่เดียวคือ ที่ความถี่ w มาพิจารณา หรือให้สัญญาณเข้าเป็น

$$v_1 = A \sin(wt + \phi) \qquad (4.1.2)$$

ดังนั้นเราจะได้สัญญาณออกเป็น

$$v_1 = A \sin\{w(t - \tau) + \phi\} \qquad (4.1.3)$$

หรือ $$v_1 = A \sin(wt - w\tau + \phi) \qquad (4.1.4)$$

และถ้าให้ เป็นมุมเฟสที่แตกต่างกันระหว่าง v_1 และ v_2 ได้

$$\theta = -w\tau \qquad (4.1.5)$$

ซึ่งจะเห็นว่ามุมเฟสมีค่าเท่ากับค่าคงตัว คูณกับความถี่ สำหรับสัญญาณชายน์ตัวอื่น ก็สามารถพิสูจน์ได้ในทำนองเดียวกันว่า วงจรกรองให้ผลตอบสนองเป็นแบบหน่วงเวลาตามสมการ (4.1.1) ได้ก็ต่อเมื่อ **มี ผลตอบสนองเฟสเป็นเชิงเส้น (linear phase response)** กับความถี่ดังสมการ 4.1.5) โดยที่ w เป็นค่าความถี่ใดๆ

คุณสมบัติของวงจรนี้อานศึกษาให้ละเอียดลงไปโดยการเขียนความสัมพันธ์ของสัญญาณเข้าและสัญญาณออกจากวงจรกรองใหม่ในรูปของเฟสเซอร์ (phasor) คือ

$$v_1(\tau) = A\angle\phi \quad \text{และ} \quad v_2(\tau) = A\angle(\phi - w\tau)$$

และฟังก์ชันถ่ายโอนเขียนได้เป็น

$$T(s) = v_2(s)/v_1(s) = 1 - w \qquad (4.1.6a)$$

หรือ $$T(s) = \exp(-jw\tau) \qquad (4.1.6b)$$

ซึ่งพจน์ $s=jw$ และ $T(s)$ ก็จะเป็นจำนวนเชิงซ้อน

$$T(w) = R(w) + jU(w) \qquad (4.1.7)$$

ดังนั้นจึงหาผลตอบสนองความถี่ได้ โดยพิจารณาเป็นสองส่วนคือ ผลตอบสนองแอมพลิจูด และผลตอบสนองเฟส ซึ่งหาได้โดย

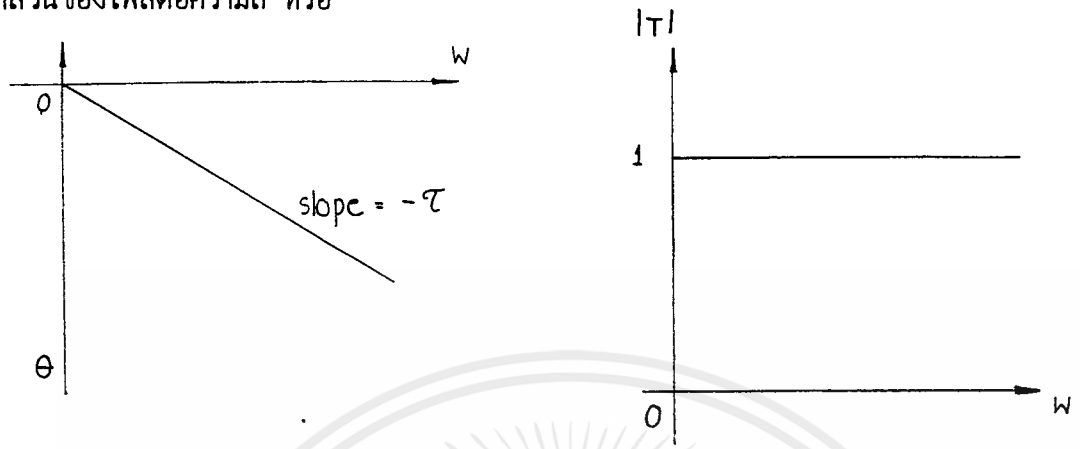
ผลตอบสนองแอมพลิจูด $= V_2(s)/V_1(s) = T(\exp\{-jw\})$ (4.1.8)

ผลตอบสนองเฟส $= \theta(w) = \tan^{-1}\{U(w)/R(w)\}$ (4.1.9)

ผลตอบสนองความถี่ของวงจรหน่วงเวลา ได้แสดงไว้ในรูป 4.1.2 ซึ่งนอกจากแสดงให้เห็นในรูป 4.1.2a ว่ามีผลตอบสนองเฟสเป็นเชิงเส้นแล้ว ยังให้ผลตอบสนองแอมพลิจูดเป็นแบบ ตัวกรองผ่านทุก

ความถี่ (all pass filter) ด้วย

การกล่าวถึงคุณสมบัติของวงจรกรองนั้น ยังมีคำอีกสองคำที่มักใช้ในการบ่งถึงคุณสมบัติของผลตอบสนองเฟสด้วย คำแรกคือความหน่วงเฟส (phase delay) จะเขียนแทนด้วย τ_P ซึ่งนิยามให้เป็นอัตราส่วนของเฟสต่อความถี่ หรือ



รูป 4.1.2 แสดง (a) ผลตอบสนองเฟส (b) ผลตอบสนองแอมพลิจูดของตัวกรองหน่วงเวลา

$$\tau_P \triangleq -\theta(w)/w \qquad (4.1.10)$$

เทอมที่สองคือ ความหน่วงกลุ่ม (group หรือ signal หรือ envelope delay) จะเขียนแทนด้วย $_G$ ซึ่งนิยามให้เป็นค่าอนุพันธ์ของเฟส เมื่อเทียบกับความถี่ หรือ

$$\tau_G \triangleq -d\theta(w) / dw \qquad (4.1.11)$$

ถ้าหากนำนิยามทั้งสองนี้มาประยุกต์ใช้กับ ผลตอบสนองเฟสของวงจรกรองหน่วงเวลา ที่มีคุณสมบัติตามสมการ (4.1.11) แล้วเห็นได้ว่า ตัวกรองหน่วงเวลามีผลตอบสนองเฟสเป็นแบบ ความหน่วงเฟสคงตัว (constant group delay) ด้วย ซึ่งตัวกรองโดยทั่วไปอาจให้คุณสมบัติเพียงความหน่วงเฟสคงตัว หรือ ความหน่วงกลุ่มคงตัว เพียงอย่างเดียวก็ได้

4.2 ตัวกรองผลตอบสนองเฟสเชิงเส้น (Linear phase filter)

จากสมการฟังก์ชันถ่ายโอนของตัวกรองไม่ป้อนกลับเชิงเลข เราหาผลตอบสนองความถี่ได้โดยแทน $Z = e^{-jwT}$ และให้

$$H(e^{jwT}) = M(w).\exp\{j\theta(w)\} \qquad (4.2.1)$$

$$= \sum_{n=0}^{N-1} h(nT)\, e^{-jwT}$$

โดยกำหนดให้

ผลตอบสนองแอมพลิจูด = $M(w) = H(e^{jwT})$ (4.2.2a)

ผลตอบสนองเฟส = $\theta(w) = \angle H(e^{jwT})$

$= \tan^{-1} \{ \text{Im } H(e^{jwT}) / \text{Re } H(e^{jwT}) \}$ (4.2.2b)

ในเบื้องแรกมาดูกรณี ตัวกรองที่ให้ผลตอบสนองเฟสเป็นแบบ **ความหน่วงเฟสคงตัว และ ความหน่วงกลุ่มคงตัว** พร้อมกัน ซึ่งตามนิยามแล้วผลตอบสนองเฟสต้องมีคุณสมบัติคือ

$$(w) = -\tau w \quad ; \quad -\pi < w < \pi \tag{4.2.3}$$

เมื่อประยุกต์ใช้นิยามของผลตอบสนองเฟสจาก (4.2.2b) และ (4.2.3) ได้ผลลัพธ์

$$(w) = -w = \tan^{-1} \{ -\sum_{n=0}^{N-1} h(nT) \sin nwT \; / \sum_{n=0}^{N-1} h(nT) \cos nwT \} \tag{4.2.4}$$

หรือ

$$\tan w = \sin w / \cos w = \{ \sum_{n=0}^{N-1} h(nT) \sin nwT \; / \sum_{n=0}^{N-1} h(nT) \cos nwT \} \tag{4.2.5}$$

ทำการคูณไขว้ ได้เงื่อนไขที่กำหนดถึง คุณสมบัติอิมพัลส์ของตัวกรองไม่ป้อนกลับเชิงเลขนี้เป็น

$$\sum_{n=0}^{N-1} h(nT).\{\cos nwT \sin w - \sin nwT \cos w \} = 0 \tag{4.2.6}$$

หรือ $$\sum_{n=0}^{N-1} h(nT) \sin \{ (\tau - nT) w \} = 0$$

เห็นได้ว่าการที่ตัวกรองไม่ป้อนกลับเชิงเลขจะมีคุณสมบัติ ผลตอบสนองเฟสเชิงเส้น ได้นั้นผลตอบสนองอิมพัลส์ต้องถูกถ่วงน้ำหนักด้วยพจน์ $\sin\{(\tau - nT) w \}$ ซึ่งเป็นฟังก์ชันซายน์ และนำผลบวกมารวมกันได้ผลรวมเป็นศูนย์

4.3 การออกแบบโดยใช้อนุกรมฟูริเยอร์

ตัวกรองไม่ป้อนกลับนี้สามารถเขียนได้ตามสมการทั่วไป (4.3.1) ซึ่งจะเห็นว่าความยาวของตัวกรองจำกัด หรือ ซีพรีจิสเตอร์ มีจำนวนจำกัด ถ้าให้ x(n) เป็นลำดับสัญญาณเข้าและให้ y(n) เป็นลำดับสัญญาณออก เพราะฉะนั้นเขียนสมการผลต่างสืบเนื่องได้

$$y(n) = \sum_{k=-M}^{+M} C_k \cdot x(n-k) \qquad (4.3.1)$$

$$y(n) = C_0\, x(n) + \sum_{k=1}^{-M} C_k \cdot x(n-k) + \sum_{k=1}^{M} C_k \cdot x(n-k)$$

$$= C_0\, x(n) + \sum_{k=1}^{M} \{ C_{-k} \cdot x(n+k) + C_k \cdot x(n-k) \}$$

โดยการเขียนเราสมมติให้ T = 1 และประยุกต์ใช้ การแปลง-แซด เพื่อเขียนฟังก์ชันถ่ายโอนได้

$$H(Z) = C_0 + \sum_{k=1}^{M} \{ C_{-k}Z^{k} + C_k Z^{-k} \} = \sum_{k=1}^{M} \{ C_k Z^{-k} \}$$

หรือ

$$H(w) = C_0 + \sum_{k=1}^{M} \{ C_{-k}e^{Jwk} + C_k e^{-Jwk} \} = \sum_{k=1}^{M} \{ C_k e^{-Jwk} \}$$

ถ้าพิจารณากรณีตัวกรองแบบสมมาตรคู่ หรือ $C_k = C_{-k}$

$$H(w) = C_0 + \sum_{k=1}^{M} \{ C_{-k}e^{Jwk} + C_k e^{-Jwk} \} \qquad (4.3.2a)$$

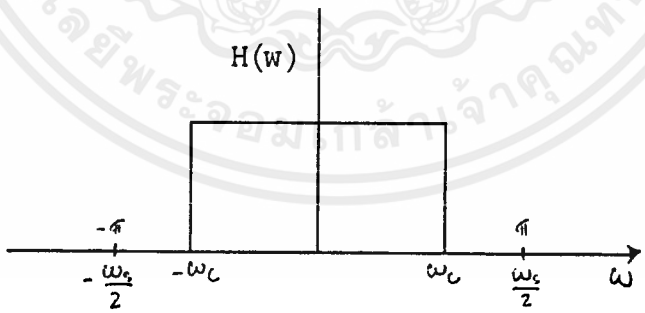
$$= C_0 + \sum_{k=1}^{M} 2C_{-k} \left(\{ e^{jwk} + C_k e^{-jwk} \} / 2 \right) \qquad (4.3.2b)$$

$$H(w) = a_0 + \sum_{k=1}^{M} a_k \cos wk$$

โดยที่กำหนดให้ $a_0 = C_0$ และ $a_k = C_k$ และ $k > 0$

ผลที่ได้จากสมการ (4.3.2b) เห็นได้ชัดว่า ตัวกรองไม่ป้อนกลับเชิงเลขชนิดสมมาตรคู่นั้น สามารถเขียนอธิบายได้โดยใช้ อนุกรมฟูริเยอร์ ที่ประกอบด้วยสัมประสิทธิ์เฉพาะพจน์โคไซน์ เท่านั้น และค่าสัมประสิทธิ์ของอนุกรมฟูริเยอร์ a_k ตาม (4.3.2b) ก็จะเป็นค่าสัมประสิทธิ์ของตัวกรองด้วย ในทำนองเดียวกันเราอาจแสดงได้ว่า สำหรับตัวกรองไม่ป้อนกลับเชิงเลขแบบปฏิสมมาตรคู่ สามารถเขียนแทนด้วย อนุกรมฟูริเยอร์ ที่มีเฉพาะพจน์ชายน์เท่านั้น และค่าสัมประสิทธิ์จะเป็นค่าจินตภาพ (imaginary) เสมอ

หลักการในการออกแบบ โดยปกติแล้วการออกแบบตัวกรองสัญญาณนั้น กระทำโดยการทำการประมาณการให้ตัวกรองที่ออกแบบมีผลตอบสนองความถี่ได้ใกล้เคียงกับผลตอบสนองอุดมคติ รูป 4.3.1 ให้มากที่สุด โดยตามรูป w_c เป็นค่าความถี่ตัด (cut off frequency) จากผลตอบสนองความถี่อุดมคติ $H_d(w)$ สามารถกระจายให้อยู่ในพจน์ของโดเมนเวลาโดยการประยุกต์ใช้ การแปลงฟูริเยอร์ได้



รูป 4.3.1 ผลตอบสนองแอมพลิจูดอุดมคติ

$$H_d(w) = \sum_{n=-\infty}^{\infty} h_d(n)\, e^{-JwnT} \qquad (4.3.3)$$

โดยที่ในกรณีนี้ผลตอบสนองอิมพัลส์ $h_d(n)$ หาได้จาก

$$h_d(n) = (1/2\pi) \int_{-w_s/2}^{w_s/2} H_d(w).e^{\,jnwT}\, dw \qquad (4.3.4)$$

แต่จากสมการ (4.3.2a) ผลตอบสนองความถี่ของตัวกรองไม่ป้อนกลับเชิงเลขเป็น

$$H(w) = \sum_{k=-M}^{M} C_k \,.\, e^{-JwT} \qquad (4.3.5)$$

เมื่อเปรียบเทียบสมการ (4.3.3) และ (4.3.5) เห็นได้ว่า ผลตอบสนองความถี่ของตัวกรองจะใกล้เคียงผลตอบสนองอุดมคติก็ต่อเมื่อ $M = \infty$ แต่ในทางปฏิบัติเป็นไปได้ยาก จึงมีการ ตัดปลาย (truncate) อนุกรมฟูริเยอร์ให้เหลือเท่าที่จำเป็น และจัดให้

$$C_k = h_d(k) \qquad (4.3.6)$$

โดยที่ $k = -M,..,0,..,M$ ส่วนความยาวของตัวกรอง $N = (2M+1)$ ซึ่ง $M = (n-1)/2$

จากผลตอบสนองแอมพลิจูดจะพบว่า ถ้า M หรือ N มีค่ามากมีผลทำให้

1) ผลตอบสนองแอมพลิจูดใกล้เคียงผลตอบสนองอุดมคติมากขึ้น

2) จำนวน ลูกคลื่น (ripple) มากขึ้น โดยที่คาบของลูกคลื่นลดลง

3) ผลตอบสนองแอมพลิจูดมีความคมมากขึ้น หรือ ปริเวณความถี่ที่ผลตอบสนองเปลี่ยนจากค่าหนึ่งไปเป็นค่าใกล้ศูนย์ หรือเรียก **แถบเปลี่ยนสถานะ** (transition Band) แคบลง

โดยทั่วไปในการออกแบบตัวกรองไม่ป้อนกลับเชิงเลขเราต้องการ ลดจำนวนลูกคลื่นให้น้อยที่สุด และให้มีช่วงแถบเปลี่ยนค่ากว้างขึ้น ดังนั้นในการออกแบบจึงจำเป็นต้องเลือกอย่างใดอย่างหนึ่ง สำหรับการออกแบบตัวกรองไม่ป้อนกลับเชิงเลขโดยวิธีเขียนอนุกรมฟูริเยอร์ อาจสรุปขั้นตอนได้ดังนี้

1) กำหนดผลตอบสนองอุดมคติที่ต้องการโดยกำหนดความถี่ตัด

2) ทำการอินทิเกรตผลตอบสนองอุดมคติเพื่อหา สำดับผลตอบสนองอิมพัลส์ $h_d(k)$

3) เลือกให้สัมประสิทธิ์ $C_k = h_d(k)$

4.4 การออกแบบโดยใช้วินโดว์ (Window methods)

จากขั้นตอนการออกแบบที่ต้องทำการตัดปลายอนุกรมฟูริเยอร์ เพื่อให้ได้ความยาวตามต้องการนั้น เปรียบเสมือนกับว่าเราทำการเจาะช่องคล้ายเป็น หน้าต่าง (window) ที่มีรูปร่างต่างๆ และเมื่ออนุกรมฟูริเยอร์จำนวนอนันต์พจน์ผ่านหน้าต่างนี้ออกไป อนุกรมนี้จะถูกถ่วงน้ำหนัก หรือถูกตัดปลายตามต้องการ

เริ่มจาก ถ้าให้ $H_d(w)$ แทนผลตอบสนองความถี่อุดมคติ และผลตอบสนองนี้ สามารถเขียนแทนด้วยอนุกรมฟูริเยอร์ยาวอนันต์พจน์ได้คือ

$$H_d(w) = \sum_{n=-\infty}^{\infty} h_d(n)\, e^{-JnwT} \qquad (4.4.1)$$

โดยที่

$$h_d(n) = (1/2) \int_{-\pi}^{\pi} H_d(w).e^{JnwT}\, dw \qquad (4.4.2a)$$

หรือ

$$h_d(n) = (1/w_s) \int_{-w_s/2}^{w_s/2} H_d(w).e^{JnwT}\, dw \qquad (4.4.2b)$$

โดยที่ w_s เป็นค่าความถี่ในการสุ่มตัวอย่าง ตามที่ทราบกันแล้วว่าการนำสมการ (4.4.1) และ (4.4.2) ไปใช้ในการออกแบบตัวกรองไม่ป้อนกลับเชิงเลขที่ผ่านมาเกิดปัญหา 2 ประการคือ ประการแรกต้องทำการตัดปลายอนุกรม (4.4.1) ให้เป็นอนุกรมจำนวนพจน์จำกัด หรือ

$$H_d(w) = \sum_{n=-(N-1)/2}^{(N-1)/2} h_d(n)\, e^{-JnwT} \qquad (4.4.3)$$

ซึ่งผลลัพธ์ก็คือ ทำให้เกิดลูกคลื่นบนผลตอบสนองแอมพลิจูด ประการที่สองก็คือ เมื่อตัดปลายอนุกรมแล้ว อนุกรม (4.4.3) จะยังนำไปสร้างใช้งานไม่ได้ เพราะว่าผลตอบสนองอิมพัลส์เริ่มจากลำดับที่ $n=-(n-1)/2$ ซึ่งสำหรับระบบเวลาจริงไม่สามารถสร้างลำดับที่เวลาเป็นลบได้ ข้อนี้แก้ไขได้โดยการเลื่อนหรือหน่วงผลตอบสนองอิมพัลส์ออกไปโดยให้เริ่มจากค่า $n = 0$ หรือ

$$H(w) = \sum_{n=0}^{N-1} h(n)\, e^{-JnwT} \qquad (4.4.4)$$

ซึ่งการทำแบบนี้มีผลดีคือทำให้ตัวกรองมีผลตอบสนองเฟสเป็นเชิงเส้น

สังเกตจากสมการ (4.4.4) เห็นได้ว่าการที่เราตัดปลายอนุกรมฟูริเยอร์ก็คือ การที่เราทำการเลือกให้

$$h(n) = h_d(n) \quad \text{เมื่อ} \quad 0 < n < N-1 \qquad (4.4.5)$$
$$= 0 \quad \text{เมื่อ} \quad n \text{ เป็นค่าอื่น}$$

ซึ่งสมการนี้ ถ้าหากเขียนให้อยู่ในรูปของสมการทั่วไปก็คือ การนำเอาลำดับ $h_d(n)$ มาคูณกับลำดับจำกัด $w(n)$ หรือ

$$h(n) = h_d(n).w(n) \qquad (4.4.6)$$

โดยที่ $w(n)$ แทนลำดับของวินโดว์แบบต่างๆ สำหรับในกรณี (4.4.5) นั้น วินโดว์เป็นวินโดว์สี่เหลี่ยม ซึ่งลำดับ $w(n)$ มีคุณสมบัตินิยามได้เป็น

$$w_R(n) = 1 \quad \text{สำหรับ} \quad 0 > n > (N-1)/2 \qquad (4.4.7)$$
$$= 0 \quad \text{สำหรับ} \quad n \text{ กรณีอื่น}$$

แต่สำหรับโครงงานนี้ ไม่ได้ออกแบบโดยใช้วินโดว์สี่เหลี่ยม จึงไม่ขอกล่าวถึงการใช้งานวินโดว์ชนิดนี้

4.5 ไคเซอร์วินโดว์ (Kaiser window)

4.5.1 คุณสมบัติของไคเซอร์วินโดว์

การออกแบบวงจรไม่ป้อนกลับเชิงเลขจะสรุปขั้นตอนได้คือ ตอนแรกตั้งเป้าหมายผลตอบสนองความถี่ที่ต้องการไว้ จากนั้นทำการหาผลตอบสนองอิมพัลส์หรือกล่าวอีกนัยหนึ่งคือ หาอนุกรมฟูริเยอร์ของผลตอบสนองความถี่นี้ ตัดปลายอนุกรมให้ได้จำนวนพจน์เป็น N ที่ต้องการ นำลำดับฟูริเยอร์ที่ถูกตัดปลายมาคูณค่าพจน์ต่อพจน์กับลำดับของวินโดว์ และถ้าหากต้องการให้ตัวกรองมีผลตอบสนองเฟสเชิงเส้นด้วย ก็ต้องทำการเลื่อนหรือหน่วงลำดับออกไป $(N-1)/2$ ลำดับ ผลลัพธ์ที่ได้คือ ค่าสัมประสิทธิ์ของตัวกรองไม่ป้อนกลับเชิงเลข อย่างไรก็ตามการออกแบบโดยใช้คุณสมบัติของวินโดว์มีข้อน่าสังเกต 2 ประการคือ ประการแรก ความกว้างของโหลบหลักของวินโดว์ แปรตามค่า $1/N$ นั่นหมายถึงว่า ถ้า

ต้องการออกแบบตัวกรองให้มีแถบเปลี่ยนค่าแคบ ก็ต้องใช้ N ค่ามาก ประการที่สอง ค่าขนาดของไหลบข้างที่เป็นส่วนที่ข้างที่ทำให้เกิดลูกคลื่นบนผลตอบสนองความถี่ไม่ได้แปรตาม N แต่ขึ้นอยู่กับชนิดของวินโดว์ที่ใช้ ดังนั้นการออกแบบตัวกรองให้มี **ค่าลดทอนในแถบหยุดน้อยที่สุด (minimum stopband attenuation)** ตามต้องการ อาจทำได้โดยต้องเลือกชนิดของวินโดว์ให้เหมาะสม

ด้วยข้อด้อยดังกล่าว **ไคเซอร์ (J.F. Kaiser)** จึงได้ทำการค้นคว้าหาวินโดว์ใหม่ที่มีคุณสมบัติดีขึ้น ให้มีคุณสมบัติปรับค่าได้ และการออกแบบสามารถทำได้เป็นระบบมากขึ้น ไคเซอร์ใช้ **ฟังก์ชันทรงคล้ายทรงกลมแบนข้าง (prolate spheroidal function)** มาทำการประมาณค่า ผลที่ได้คือได้วินโดว์ที่มีคุณสมบัติแลกเปลี่ยน (trade-off) ระหว่างค่าขนาดของไหลบหลักกับค่าขนาดของไหลบข้าง โดยที่มีพารามิเตอร์ เป็นตัวควบคุมค่าเพื่อปรับค่าขนาดไหลบข้างเมื่อเทียบกับค่าขนาดของไหลบหลักได้ ส่วนความกว้างของไหลบหลักก็เหมือนกับวินโดว์อื่นๆ คือปรับค่าได้โดยการเลือก N วินโดว์แบบไคเซอร์นี้อาจเรียกได้ว่ามีคุณสมบัติ **เกือบเหมาะสมที่สุด (nearly optimum window)** อย่างไรก็ตาม ฟังก์ชันทรงคล้ายคล้ายทรงกลมแบนข้าง เป็นฟังก์ชันที่ไม่เป็นที่รู้จักกันทั่วไป อีกทั้งการคำนวณทำได้ยาก ไคเซอร์จึงได้พัฒนาขั้นตอนการออกแบบโดยใช้วินโดว์ขึ้นมา ซึ่งจะได้กล่าวต่อไป

ลำดับของไคเซอร์วินโดว์ ที่อยู่ในรูปของฟังก์ชันคล้ายทรงกลมแบนข้างคือ

$$w_k(nT) = I_0(\beta) \,/\, I_0(\alpha) \qquad \text{เมื่อ} \quad n < N-1/2 \tag{4.5.1}$$

$$= 0 \qquad \text{กรณี n ค่าอื่น}$$

โดยที่ เป็นพารามิเตอร์อิสระที่สัมพันธ์กับพารามิเตอร์ β โดย

$$= \alpha \cdot \sqrt{1-(2n/(n-1))^2} \tag{4.5.2}$$

และ $I_0(x)$ เป็น **เบสเซลฟังก์ชันชนิดที่หนึ่งอันดับศูนย์ (zero th - order Bessel function of the first kind)** ที่สามารถคำนวณได้โดยใช้อนุกรมกำลังคือ

$$I_0(x) = 1 + \{ (1/k!)\,(x/2)^k \}^2 \tag{4.5.3a}$$

$$= 1 + \{(x/2)^2\} + \{(x/2)^4 /(2!)^2\}$$

$$+ \{(x/2)^6 /(3!)^2\} + \{(x/2)^8 /(4!)^2\} + \ldots \tag{4.5.3b}$$

ในการประยุกต์ใช้งานทั่วไป สำหรับความละเอียดในการคำนวณตามสมการ (4.5.3b) อาจใช้จำนวนพจน์ระหว่าง 15 ถึง 25 พจน์ก็เพียงพอ

ต่อไปจะดูว่าพจน์ มีผลต่อการแปรคุณสมบัติของไคเซอร์วินโดว์อย่างไร พิจารณากรณี = 0 เห็นได้ว่า พจน์เศษและพจน์ส่วน ของสมการ (4.5.1) มีค่าเป็นหนึ่ง นั่นคือ $w_k(nT)$ ทำตัวเป็นวินโดว์สี่เหลี่ยม ถ้าหาก มีค่าเพิ่มขึ้นมาก โหลบหลักของวินโดว์กว้างขึ้น ส่วนค่าขนาดของโหลบข้างมีค่าลดลง จากการพิจารณาอย่างคร่าวๆ นี้เห็นได้ว่า พจน์ สามารถใช้เป็นพารามิเตอร์ในการควบคุมคุณสมบัติของไคเซอร์วินโดว์ ให้มีอัตราลูกคลื่นน้อยลงได้

สเปกตรัมของไคเซอร์วินโดว์ หาได้โดย การประยุกต์ผลการแปลง-แซด กับสมการ (4.5.1) และให้ $Z = e^{jwT}$ ได้ผลว่า

$$W_k(e^{jwT}) = W_k(0) + 2.\sum_{n=1}^{(N-1)/2} w_k(nT)\cos nwT \qquad (4.5.4)$$

ซึ่งสามารถแสดงได้ว่า

$$W_k(w) = \{2/I_0(\beta)\}\{\sin(\pi\sqrt{w^2 - w_b^2})\} / \sqrt{w^2 - w_b^2} \qquad (4.5.5)$$

โดยที่ให้ $w_b = \alpha/\pi$ และ $= (N-1)T/2$ และ w_b เป็นค่าที่กำหนดขึ้นมาเพื่อเป็นการแลกเปลี่ยนกันระหว่างความกว้างของโหลบหลักกับค่าขนาดของโหลบข้าง

สมการ (4.5.5) สามารถนำไปหาค่าความกว้างของโหลบหลัก โดยการหาตำแหน่งที่ให้ $W(jw) = 0$ ครั้งแรก แล้วคูณค่าด้วย 2 ได้ผลเป็น

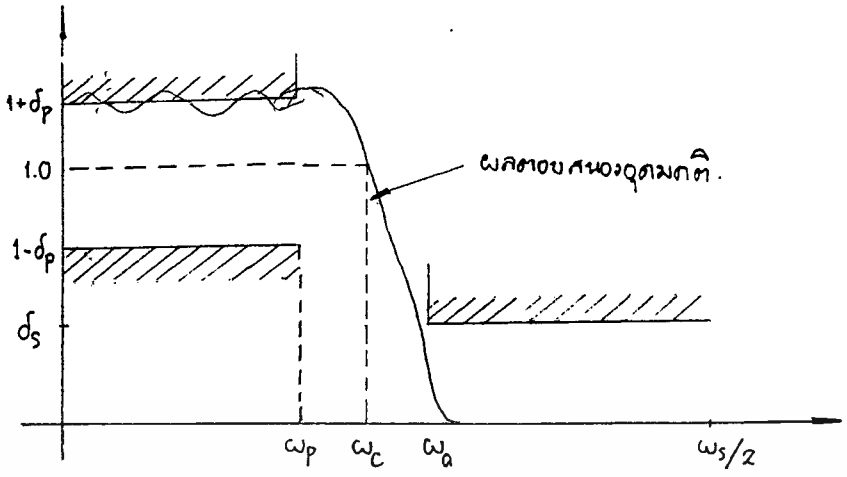
$$C_w = 2w_b\sqrt{1 + (\pi/\alpha)^2} \qquad (4.5.6)$$

โดยที่ C_w คือค่า ความกว้างของโหลบหลัก (center-lobe or main-lobe width) และจาก (4.4.6) ถ้าเราแก้สมการหาค่า $W(jw)$ ที่ตำแหน่งที่ค่าอนุพันธ์ของ $W(jw)$ เป็นศูนย์ครั้งแรก แล้วนำไปคิดเป็นเปอร์เซนต์กับค่า $W(0)$ ได้ค่าอัตราส่วนลูกคลื่นเป็น

$$RR = \frac{0.218\,\alpha}{\sin h(\alpha)} \times 100\% \qquad (4.5.7)$$

4.5.2 การออกแบบโดยใช้ไคเซอร์วินโดว์

ตามรูป 4.5.1 เป็นการแสดงถึงข้อกำหนดในการออกแบบ วงจรกรองผ่านความถี่ต่ำ โดยทั่วไป พารามิเตอร์ δ_p และ δ_s เป็นตัวกำหนดคุณสมบัติของค่าขนาดของลูกคลื่นใน แถบผ่าน (pass



รูปที่ 4.5.1 ข้อกำหนดของผลตอบสนองความถี่ที่ต้องการออกแบบ

band) และในแถบหยุด (stopband) ตามลำดับ ถ้ากำหนดให้ A_p เป็นค่าขนาดของลูกคลื่นในแถบผ่าน A_a เป็นค่าลดทอนในแถบหยุดน้อยที่สุด และ B_t เป็นค่าความกว้างของแถบเปลี่ยนสถานะ เราสามารถเขียนความสัมพันธ์ได้ว่า

$$A_p = 20 \log \{ (1 + \delta_p)/(1 - \delta_p) \} \quad (4.5.8)$$

$$A_a = -20 \log \delta_s \quad (4.5.9)$$

และ $$B_t = w_a - w_p \quad (4.5.10)$$

ในการเขียนต่อไปนี้ ถ้าสมมติว่าให้ A_p' และ A_a' เป็นค่าขนาดของลูกคลื่นในแถบผ่าน และค่าลดทอนในแถบหยุดน้อยที่สุด ที่กำหนดมาจากผลตอบสนองรูป 4.5.1 และให้ A_p และ A_a เป็นค่าที่ได้จากการออกแบบ ดังนั้นขั้นตอนในการออกแบบตัวกรองโดยใช้ไคเซอร์วินโดว์ เป็นดังนี้

1) หาผลตอบสนองอิมพัลส์ h(n) โดยการประยุกต์ใช้อนุกรมฟูริเยอร์กับผลตอบสนองอุดมคติคือ

$$H_d(w) = 1 \quad \text{เมื่อ} \quad |w| < w_c$$

$$= 0 \quad \text{เมื่อ} \quad w_c < |w| < w_s/2$$

โดยที่จากรูป 4.5.1

$$w_c = (1/2)(w_p + w_a) \quad (4.5.11)$$

2) เลือกพารามิเตอร์จากสมการ (4.5.8) และ (4.5.9) จนได้ค่าที่ทำให้

$$A_p < A_p' \quad ; \quad A_a > A_a' \tag{4.5.12}$$

และเลือก

$$\delta = \min(\delta_p, \delta_s) \tag{4.5.13}$$

การเลือก ตามสมการ (4.5.13) เนื่องมาจากว่า การเลือกตามเงื่อนไขสมการ (4.5.8) และ (4.5.9) ทำให้เกิดข้อด้อยคือได้ตัวกรองที่ทำให้ $\delta = \delta_p = \delta_s$ เสมอ

3) คำนวณหา A_a จากสมการ (4.5.9) โดยใช้ค่า δ จาก (4.5.13)

4) เลือกพารามิเตอร์ α ให้สอดคล้องกับสมการข้างล่างนี้

$$\alpha = \begin{cases} 0 & \text{เมื่อ } A_a < 21 \\ 0.5842(A_a-21)^{0.4} + 0.07886(A_a-21) & \text{เมื่อ } 21 < A_a < 50 \\ 0.1102(A_a-8.7) & \text{เมื่อ } A_a > 50 \end{cases} \tag{4.5.14}$$

5) เลือกพารามิเตอร์ D ให้สอดคล้องกับ

$$D = \begin{cases} 0.9222 & \text{เมื่อ } A_a < 21 \\ (A_a - 7.95)/14.36 & \text{เมื่อ } A_a > 21 \end{cases} \tag{4.5.15}$$

6) คำนวณหาค่า N ที่เป็นเลขคี่ที่ต่ำสุดที่ทำให้ได้เงื่อนไข

$$N \geq (w_s \cdot D / B_t) + 1 \tag{4.5.16}$$

7) คำนวณหา $w_k(nI)$ จากสมการ (4.5.1) และดัดแปลงผลตอบสนองอิมพัลส์ตามสมการ (4.4.5) และ (4.4.6) ได้เป็น

$$h(n) = h_d(n) \cdot w_k(n) \qquad \text{สำหรับ } n \leq (N-1)/2 \tag{4.5.17}$$

8) ฟังก์ชันถ่ายโอนของตัวกรองเขียนได้เป็น

$$H(z) = z^{-(n-1)/2} \{ h(0) + 2 \sum_{n=0}^{(N-1)/2} h(n) (z^{n} + z^{-n}) \qquad (4.5.18)$$

โดยที่ $h(0) = h_d(0) \cdot w_k(0)$ (4.5.19)

$h(n) = h_d(n) \cdot w_k(n)$

และผลตอบสนองแอมพลิจูดของวงจรกรองหาได้จาก

$$H(w) = h(0) + 2 \sum_{n=1}^{(N-1)/2} h(n) \cos 2n\, fT \qquad (4.5.20)$$

ตัวอย่าง การออกแบบวงจรกรองผ่านความถี่ต่ำ

ถ้ากำหนดให้ ค่าขนาดของลูกคลื่นในแถบความถี่ผ่าน $A_p < 0.1$ dB ในช่วงความถี่ 1 ถึง 1.5 เรเดียนต่อวินาที และกำหนดให้ ค่าลดทอนน้อยสุดในแถบหยุด $A_a > 40$ dB ในช่วงความถี่ 2.5 ถึง 5.0 เรเดียนต่อวินาที โดยที่ให้ค่าความถี่ในการสุ่มตัวอย่างสัญญาณเป็น 10 ต่อวินาที

วิธีทำ การคำนวณทำตามลำดับดังนี้

ขั้นตอนที่ 1 ได้ $h(nT) = (1/n) \sin w_c nT$

โดยที่ จาก (4.5.11) ให้ค่า $w_c = (1.5+2.5)/2 = 2.0$ เรเดียนต่อวินาที

ขั้นตอนที่ 2 สมการ (4.5.8) ให้ $\delta_1 = \{ (10^{0.05(0.1)}-1)/(10^{0.05(0.1)}+1)$

$= 5.7564 \times 10^{-3}$

สมการ (4.5.9) ให้ $\delta_2 = 10^{-0.05(40)} = 0.01$

สมการ (4.5.13) เลือก $\delta = \min(\delta_1, \delta_2) = 5.7564 \times 10^{-3}$

ขั้นตอนที่ 3 ถึง 6 ได้ $A_a = 44.979$ dB

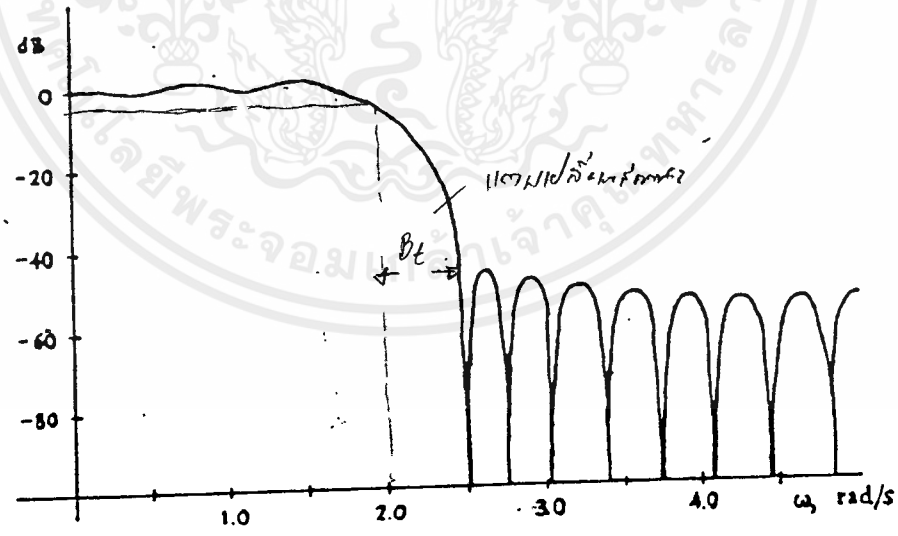
ขั้นตอนที่ 4 และ 5 ให้ $\alpha = 3.9524$; $D = 2.566$

และ $N > \{10(2.566)/1\} + 1 = 26.66 = 27$

ขั้นตอนที่ 6 และ 7 ได้สัมประสิทธิ์ของตัวกรอง ซึ่งขอเขียนเฉพาะ 8 ตัวแรก ดังนี้

n	h(nT)	$w_k(nT)h(nT)$
0	$4x10^{-1}$	$4x10^{-1}$
1	$3.02731x10^{-1}$	$2.99692x10^{-1}$
2	$9.35489x10^{-2}$	$8.98359x10^{-2}$
3	$-6.2366x10^{-2}$	$-5.69018x10^{-2}$
4	$-7.56827x10^{-2}$	$-6.42052x10^{-2}$
5	0	0
6	$5.04551x10^{-2}$	$3.45003x10^{-2}$
7	$2.67283x10^{-2}$	$1.57769x10^{-2}$

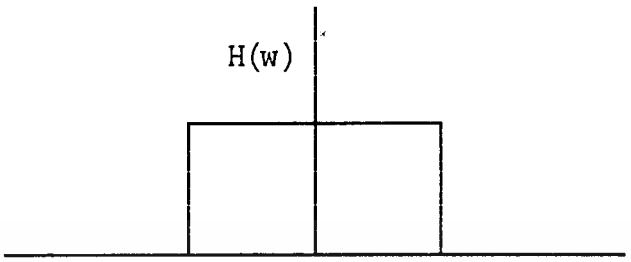
และผลตอบสนองความถี่เป็นดังรูป 4.5.2



รูป 4.5.2 ผลตอบสนองแอมพลิจูด

4.6 การหาสมการสำเร็จสำหรับการเขียนโปรแกรม

เนื่องจากการออกแบบตัวกรอง เราจะทำในลักษณะของความถี่นอร์มอไลซ์ กล่าวคือ ให้ความถี่สุ่มมีค่าเป็น 2π และเหตุผลที่ความถี่ที่ถูกประมวลผลจะต้องมีค่าไม่เกินครึ่งหนึ่งของความถี่สุ่ม ดังนั้นช่วงความถี่ที่ใช้ในการออกแบบของเราจึงอยู่ในช่วง - ถึง + ดังแสดงในรูปข้างล่างนี้



ดังนั้นในการสร้าง Lowpass filter prototype เราจึงหาได้จากการใช้การแปลงอินเวอร์สฟูริเยอร์ ของรูปผลตอบสนองความถี่ข้างบน ดังต่อไปนี้

$$h(n) = 1/2\pi \int_{-W_c}^{W_c} H(w)\, e^{jwn}\, dw \quad ;\{ H(w) = 1 \ , \ -W_c < w < W_c \ \}$$

ดังนั้น

$$h(n) = 1/2\pi \int_{-W_c}^{W_c} e^{jwn}\, dw$$

$$= 1/2\pi \ .1/jn \left[e^{jwn} \Big|_{-W_c}^{W_c} \right]$$

$$= 1/2\pi \ .1/jn \ [e^{jw_c n} - e^{-jw_c n}]$$

$$= 1/2\pi \ .1/jn \ [2j \sin w_c n]$$

$$h(n) = 1/\pi n \ . \sin w_c n$$

$$h(0) = [\tfrac{d}{dn} \sin w_c n] / [\tfrac{d}{dn} \pi n]$$

$$= (w_c \cos w_c n) / \pi$$

$$h(0) = (w_c \cos 0) / \pi$$

ดังนั้น $$h(0) = w_0 / \pi$$

$$h(n) = (\sin w_c n) / n\pi$$

สูตรสำเร็จที่ได้นี้จะใช้ในการหาผลตอบสนองอิมพัลส์ของโปรแกรม โดยที่เราสามารถแปรค่าของ w_c ได้ตามต้องการ สำหรับตัวกรองชนิดอื่น ก็มีวิธีการหาได้จากการดัดแปลงผลตอบสนองอิมพัลส์ของตัวกรองผ่านความถี่ต่ำ ดังสูตรต่อไปนี้

ตัวกรองผ่านความถี่สูง (High Pass Filter)

$$h(n)_{HP} = (-1)^n h(n) \qquad \text{เมื่อ } n = \ldots,-2,-1,0,1,2,\ldots$$

ตัวกรองผ่านแถบความถี่ (Band Pass Filter)

$$h(n)_{BP} = (2\cos nw_0T)h(n)_{LP} \qquad \text{เมื่อ } n = \ldots,-2,-1,0,1,2,\ldots$$

ตัวกรองกำจัดแถบความถี่ (Band Stop Filter)

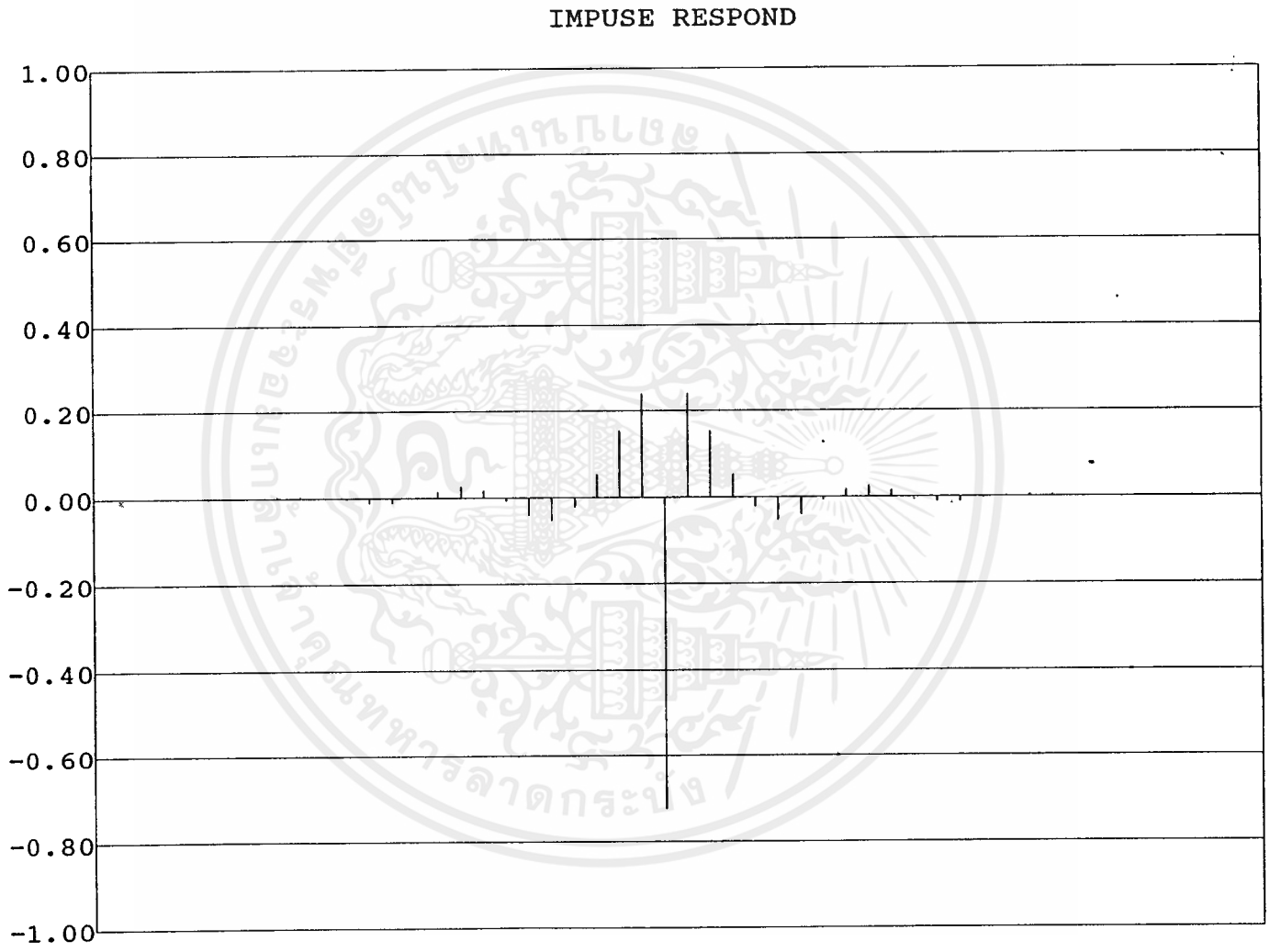
$$h(0)_{BS} = 1 - h(0)_{BP}$$

$$h(n)_{BS} = -h(n)_{BP} \qquad \text{เมื่อ } n = \ldots,-2,-1,1,2,\ldots$$

บทที่ 5 บันทึกผลการทดลอง

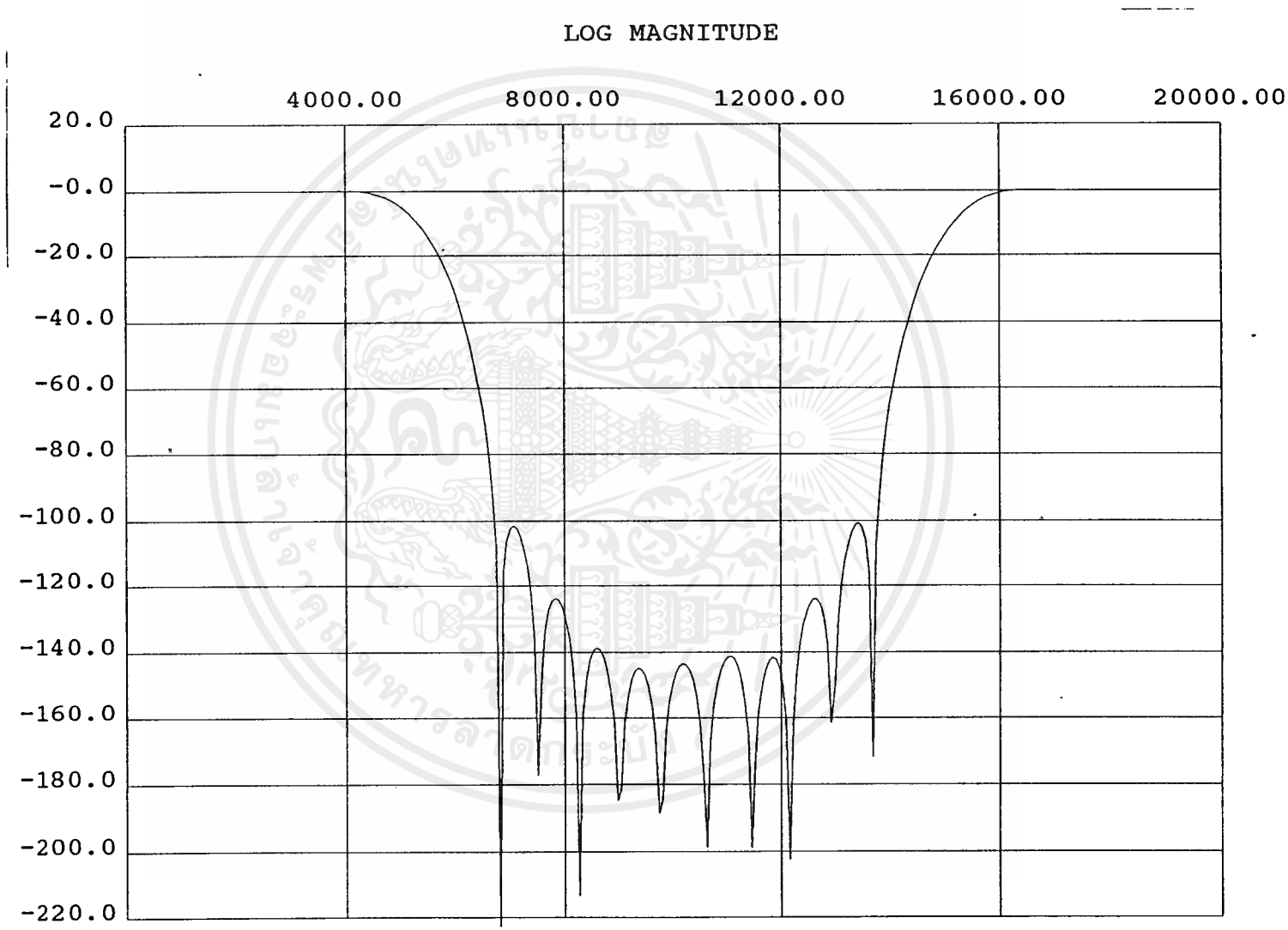
กราฟต่อไปนี้ แสดงคุณสมบัติต่างๆ ที่ได้จากการออกแบบตัวกรองชนิดต่างๆ โดยมีคุณสมบัติดังต่อไปนี้

1. ชนิดของตัวกรอง (Type of Filter)
2. ชนิดของวินโดว์ (Window)
3. ความถี่สุ่ม (Sampling Frequency)
4. จำนวนความยาวของผลตอบสนองอิมพัลส์ (Length)

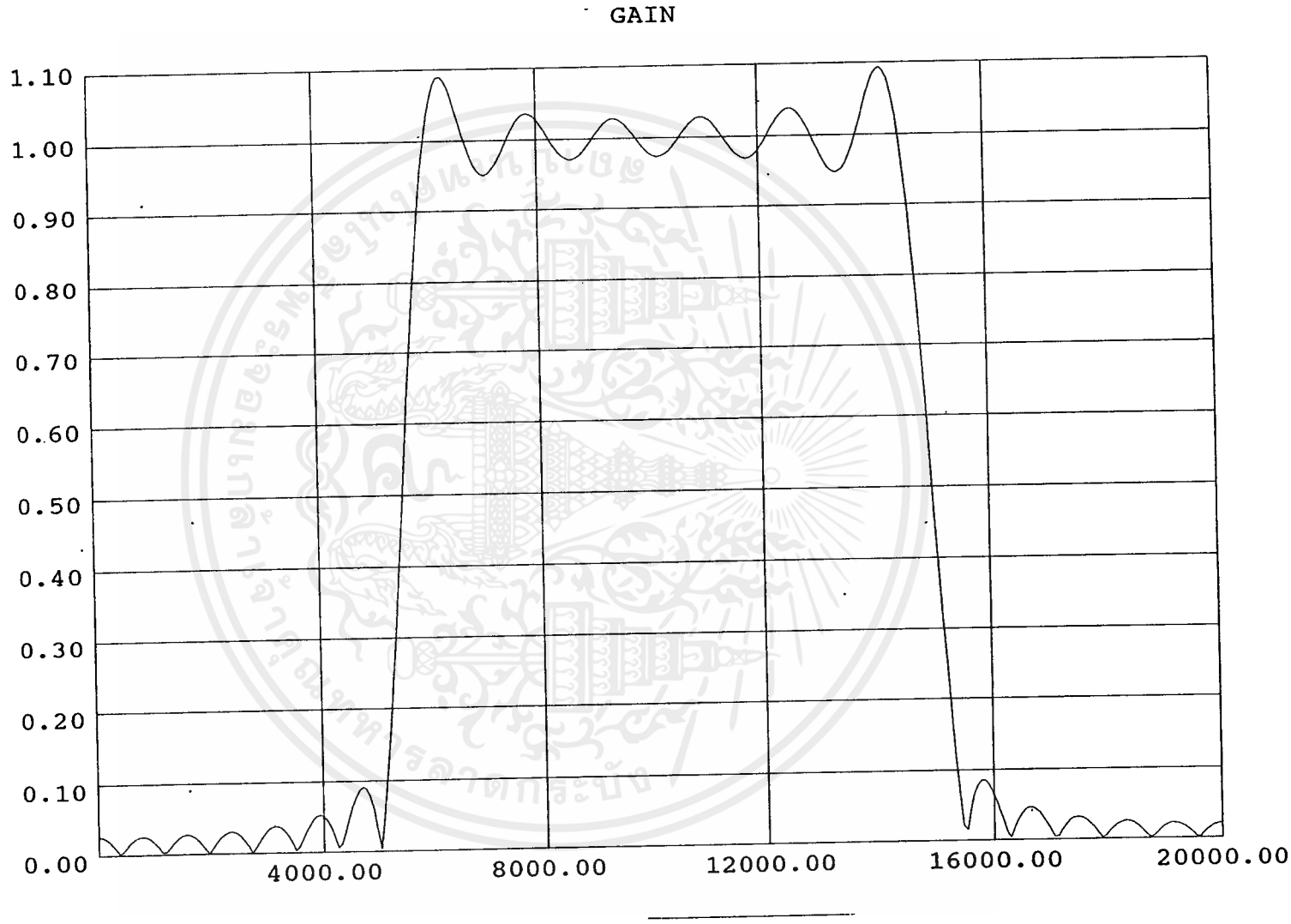


Type of Filter : High Pass Filter
Sampling Frequency : 40 KHz

Window : Hamming
Length : 51 points

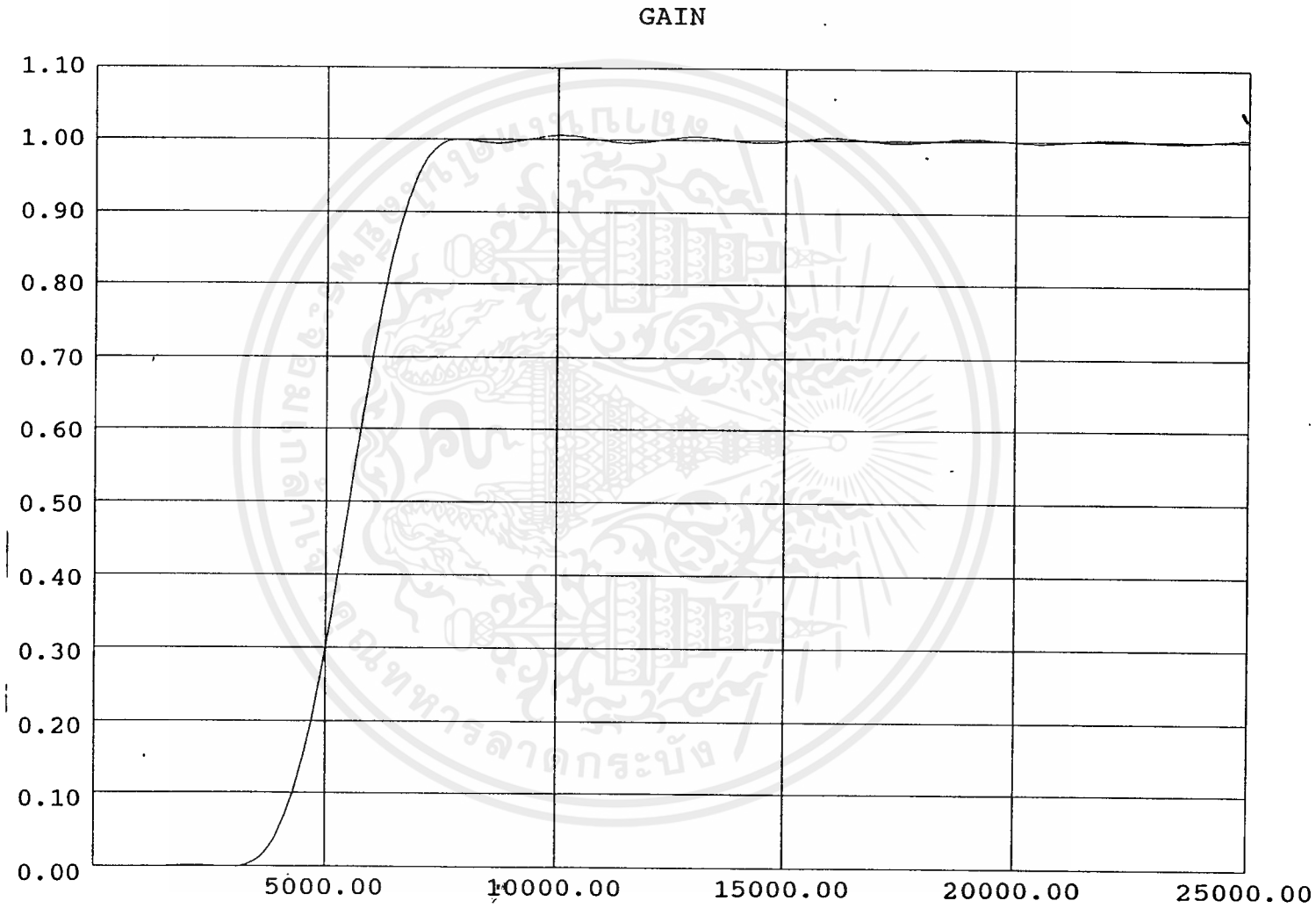


Type of Filter : Band Rejection Filter Window : Hann
Sampling Frequency : 40 KHz Length : 51 points



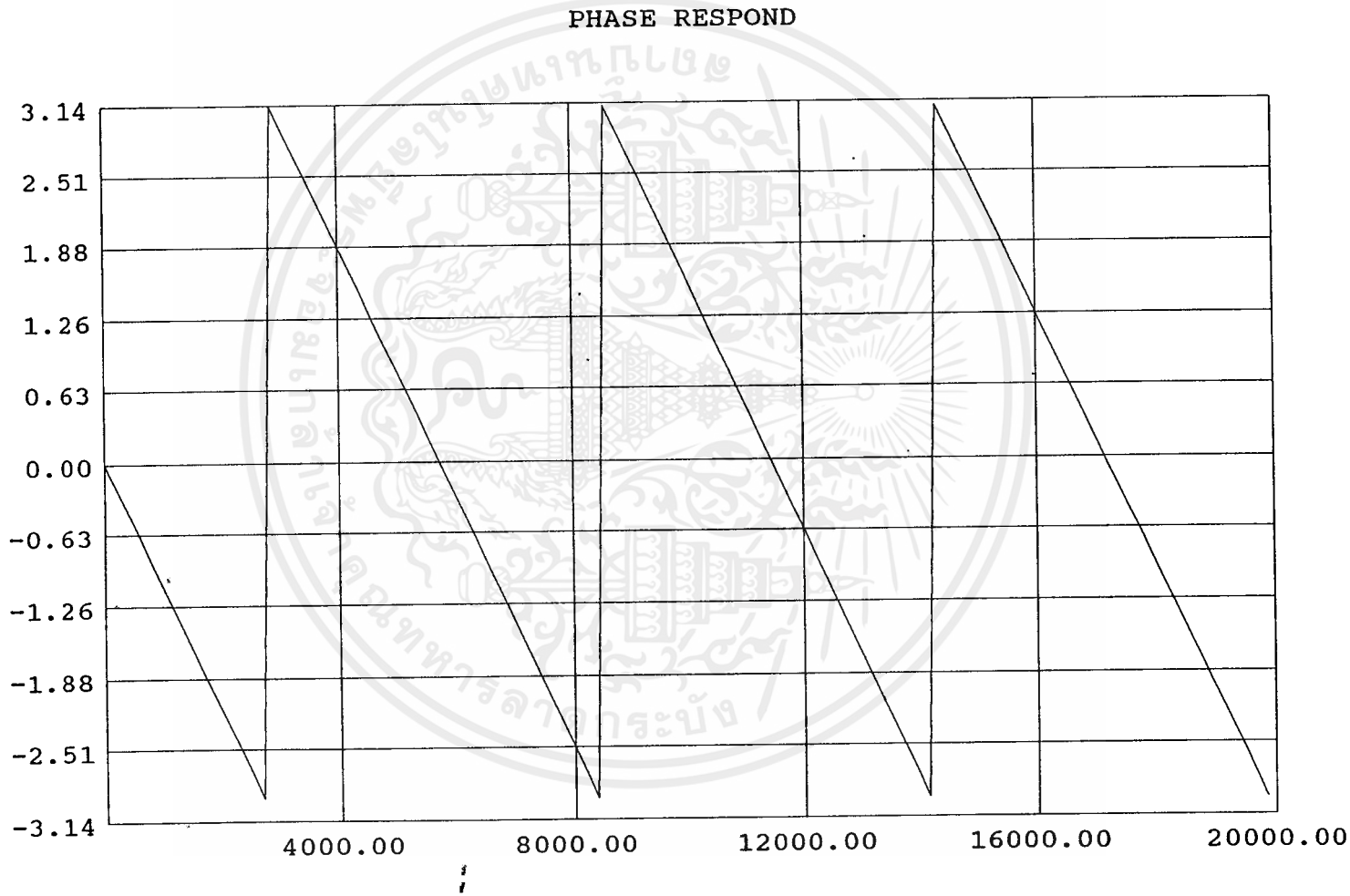
Type of Filter : Band Pass Filter
Sampling Frequency : 40 KHz

Window : Rectangular
Length : 51 points



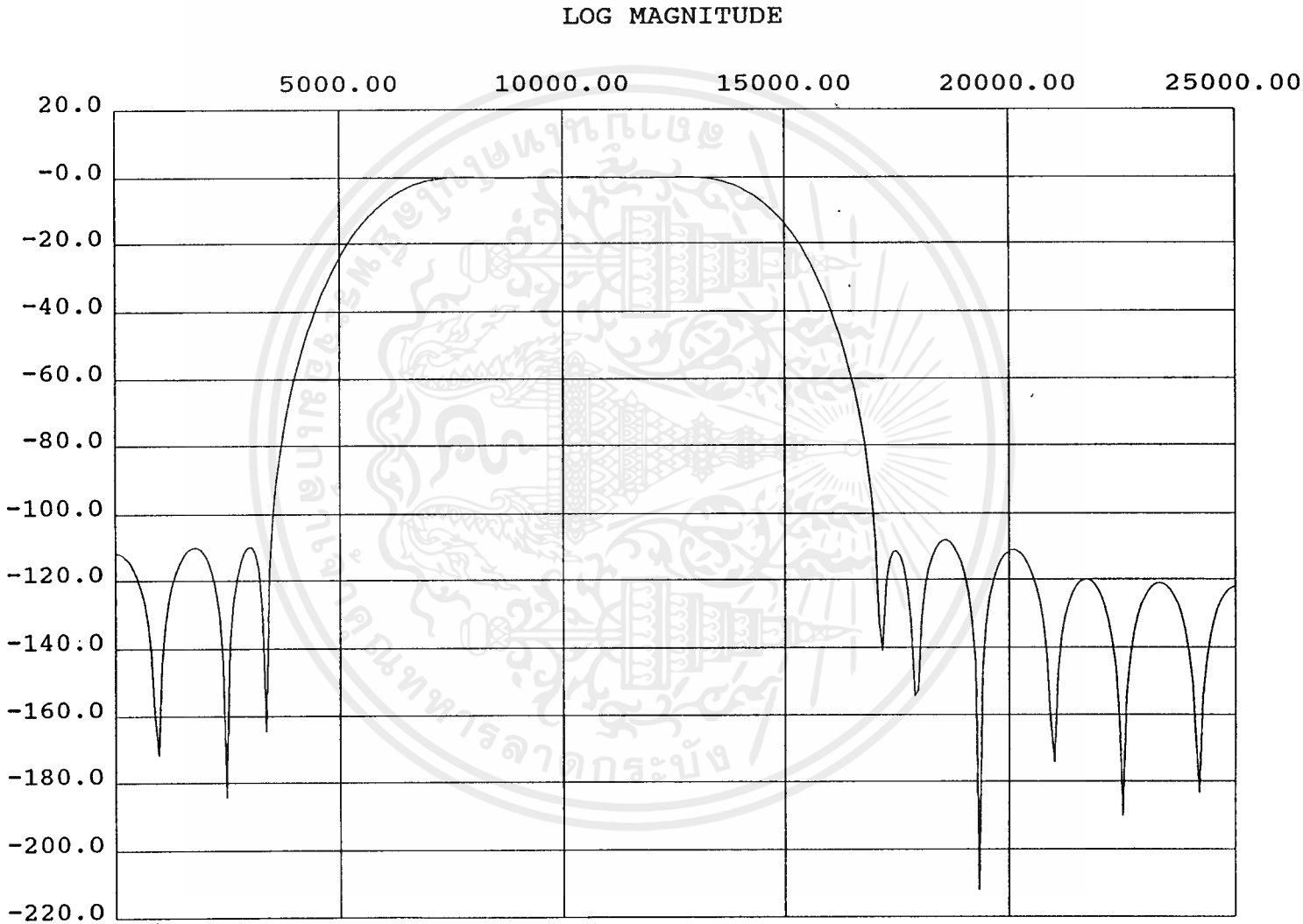
Type of Filter : High Pass Filter
Sampling Frequency : 50 KHz

Window : Kaiser
Length : 35 points



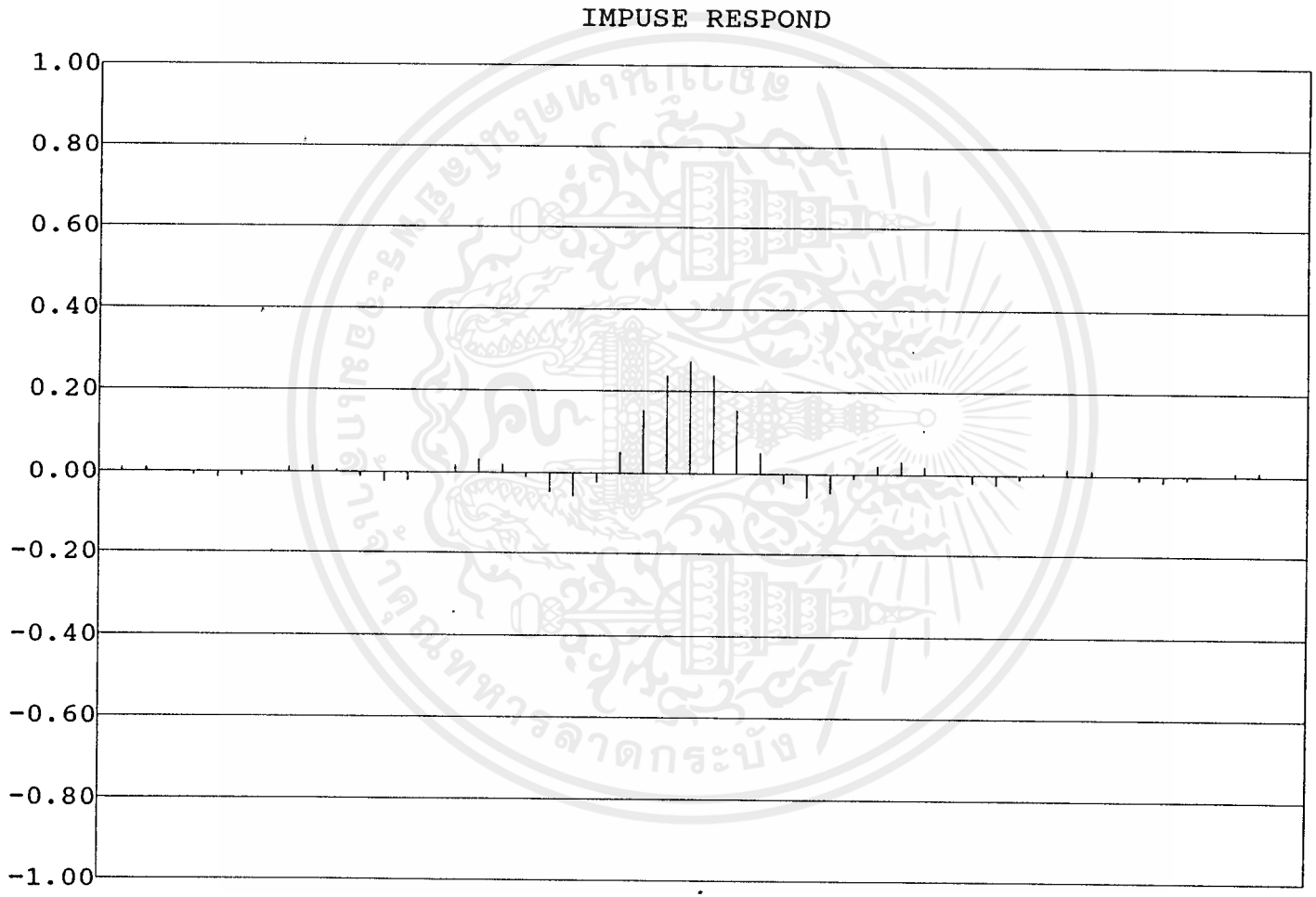
Type of Filter : Band Pass Filter
Sampling Frequency : 40 KHz

Window : Kaiser
Length : 15 points



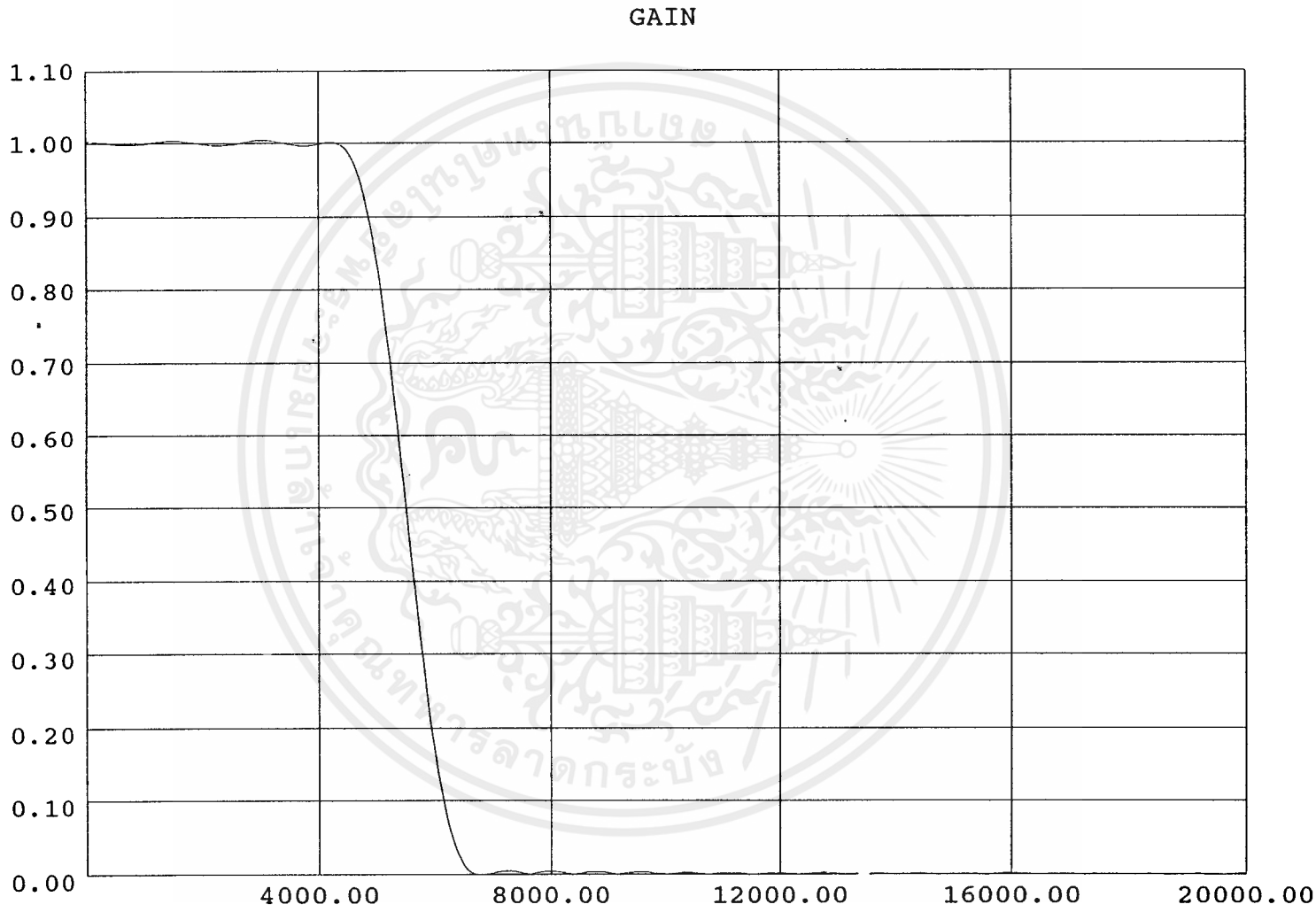
Type of Filter : Band Pass Filter
Sampling Frequency : 50 KHz

Window : Kaiser
Length : 35 points



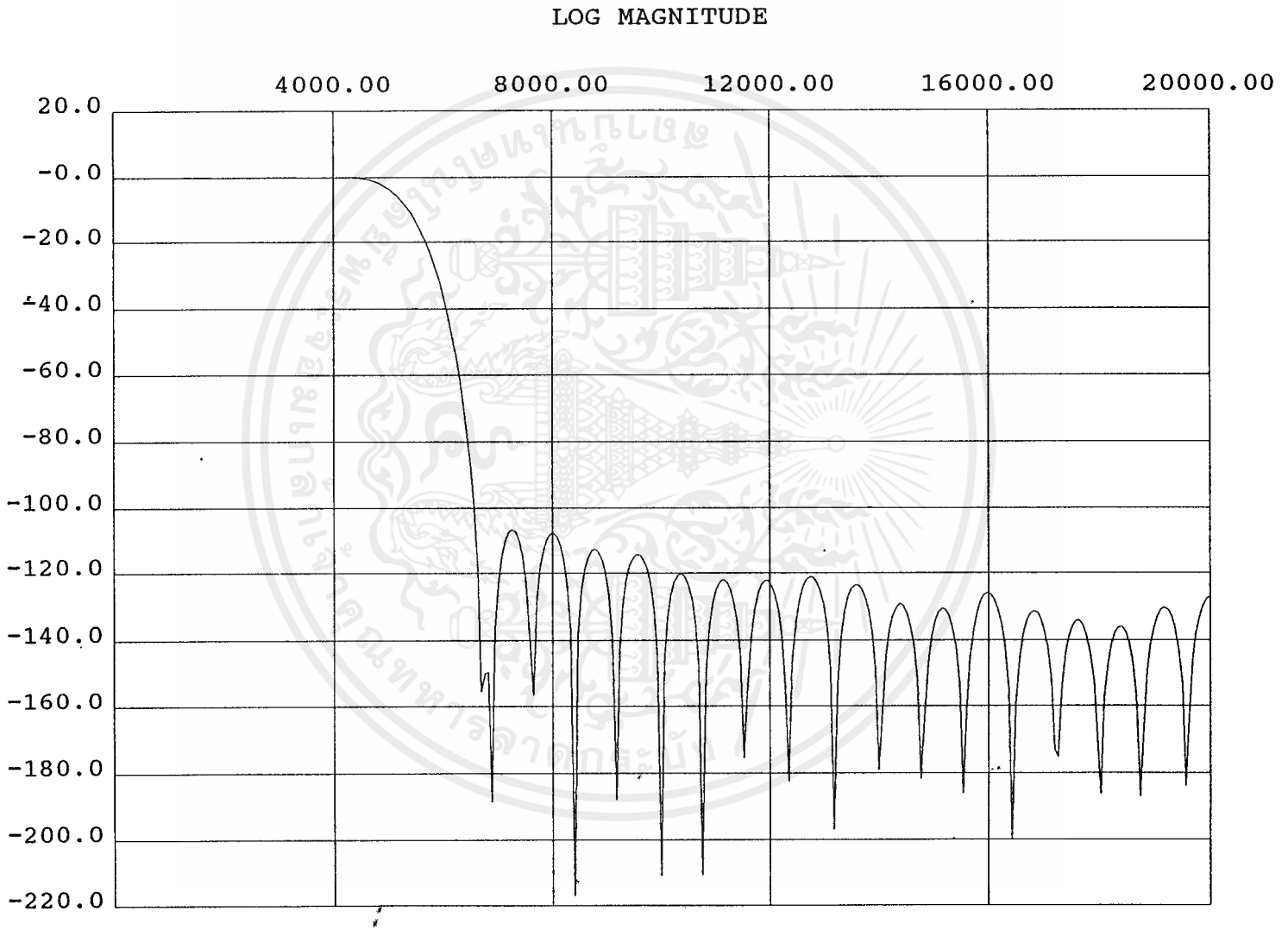
Type of Filter : Low Pass Filter
Sampling Frequency : 40 KHz

Window : Rectangular
Length : 51 points



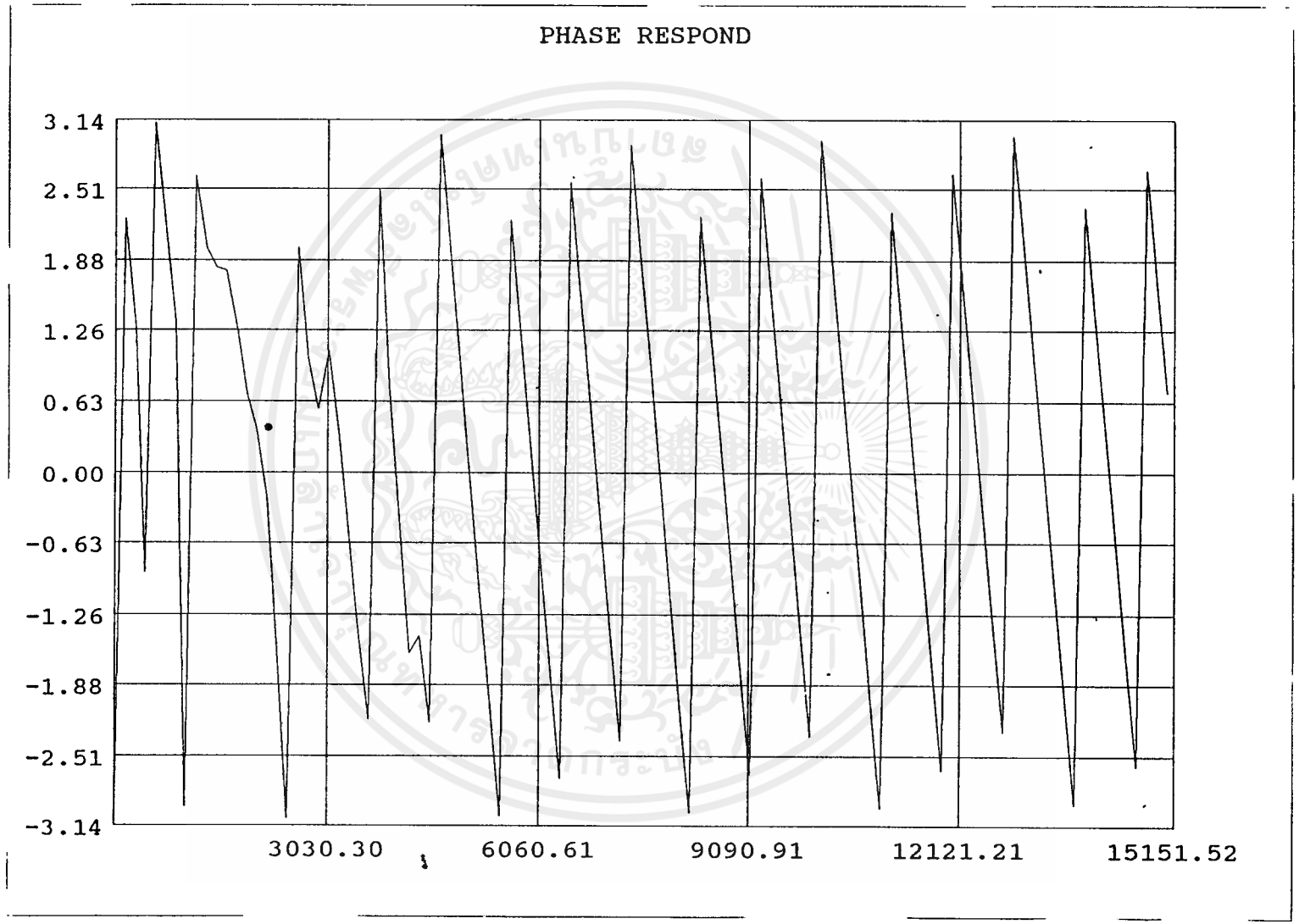
Type of Filter : Low Pass Filter
Sampling Frequency : 40 KHz

Window : Kaiser
Length : 51 points



Type of Filter : Low Pass Filter
Sampling Frequency : 40 KHz

Window : Kaiser
Length : 51 points



Type of Filter : High Pass Filter
Sampling Frequency : 30 KHz

Window : Blackman
Length : 67 points

บทที่ 6 สรุปผลการทดลอง

ตามรายละเอียดและผลการทดลองที่ได้กล่าวมาข้างต้นแล้ว เป็นการนำเอาหลักการและทฤษฎีทางการประมวลผลสัญญาณแบบดิจิตอล มาใช้ในการสร้างดิจิตอลฟิลเตอร์ที่สามารถโปรแกรมได้ (Programable Digital Filter) ซึ่งมีคุณสมบัติหลายประการที่ดีกว่า Analog Filter เช่น ความเป็นเชิงเส้นของการตอบสนองทางเฟส (Linear Phase Response) และที่สำคัญก็คือ สามารถโปรแกรมให้ระบบทำหน้าที่เป็นวงจรกรองแบบต่างๆ ได้ และมี specification ต่างๆ ได้โดยสะดวก และไม่ต้องเปลี่ยนวงจรใหม่ เพราะใช้การโปรแกรมทางซอฟต์แวร์เท่านั้น ตามที่กล่าวรายละเอียดมาข้างต้นแล้ว การเปลี่ยนชนิดและคุณสมบัติของระบบทำได้โดยการคำนวณ impulse response ของระบบตามที่ต้องการ แล้วเปลี่ยนผลที่ได้นั้นให้เป็นสัมประสิทธิ์ของ Transfer function ในรูปแบบของระบบ FIR (Finite Impulse Response) ซึ่งรายละเอียดในการคำนวณต่างๆ นั้น ได้กล่าวไว้แล้วในส่วนของทฤษฎี

อย่างไรก็ตาม โครงงานนี้ยังสามารถทำการพัฒนาให้มีความสามารถและมีฟังก์ชันในการทำงานหลากหลายขึ้น เช่น การพัฒนาโปรแกรมให้เป็นระบบ IIR Filter หรือ Infinite Impulse Response ซึ่งมีหลักการคล้ายระบบทางอนาลอก กล่าวคือ เป็นระบบที่มีการป้อนกลับ ดังนั้น IIR Filter จะมีลักษณะอัลกอริทึมที่ได้จากการแปลงจากระบบอนาลอก ซึ่งมีข้อดีคือ สามารถสร้าง Filter ให้ได้ Transition Band ที่แคบกว่าได้ โดยใช้ order น้อยกว่าของ FIR Filter อย่างไรก็ดี การออกแบบ IIR Filter นั้น การจะทำให้ได้ linear phase response นั้น ทำได้ยากกว่าแบบ FIR มาก

การประยุกต์ใช้งานตัวกรองเชิงเลขที่ได้จากโครงงานนี้ ยังถูกจำกัดในด้านความเร็วในการประมวลผล ดังนั้น จึงสามารถประยุกต์ใช้งานกับสัญญาณในย่านความถี่เสียง (Audio Signal) แต่ในการประยุกต์ขั้นต่อไปนั้น ด้านความเร็วสามารถเพิ่มได้โดยการใช้ไมโครโปรเซสเซอร์ที่มีความเร็วสูงขึ้น พร้อมทั้งพัฒนาด้านอัลกอริทึมของโปรแกรมต่อไป.

ภาคผนวก ก.

ในภาคผนวกนี้จะแสดงรูปรายละเอียดของ วงจรที่ใช้ในส่วนต่างๆทั้งหมด ซึ่งจะมีดังนี้

1) **วงจรในบอร์ด TMS** ซึ่งได้แก่

1.1 วงจรหลักของ TMS (DSP HARDWARE[TMS])

1.2 วงจรกำเนิดความถี่ให้ TMS (OSCILATOR CIRCUIT)

1.3 วงจรเลือกแอดเดรส (ADDRESS SELECTOR)

1.4 วงจรเลือกข้อมูล (DATA SELECTOR)

1.5 วงจรรีเซ็ทการทำงาน (POWER ON RESET)

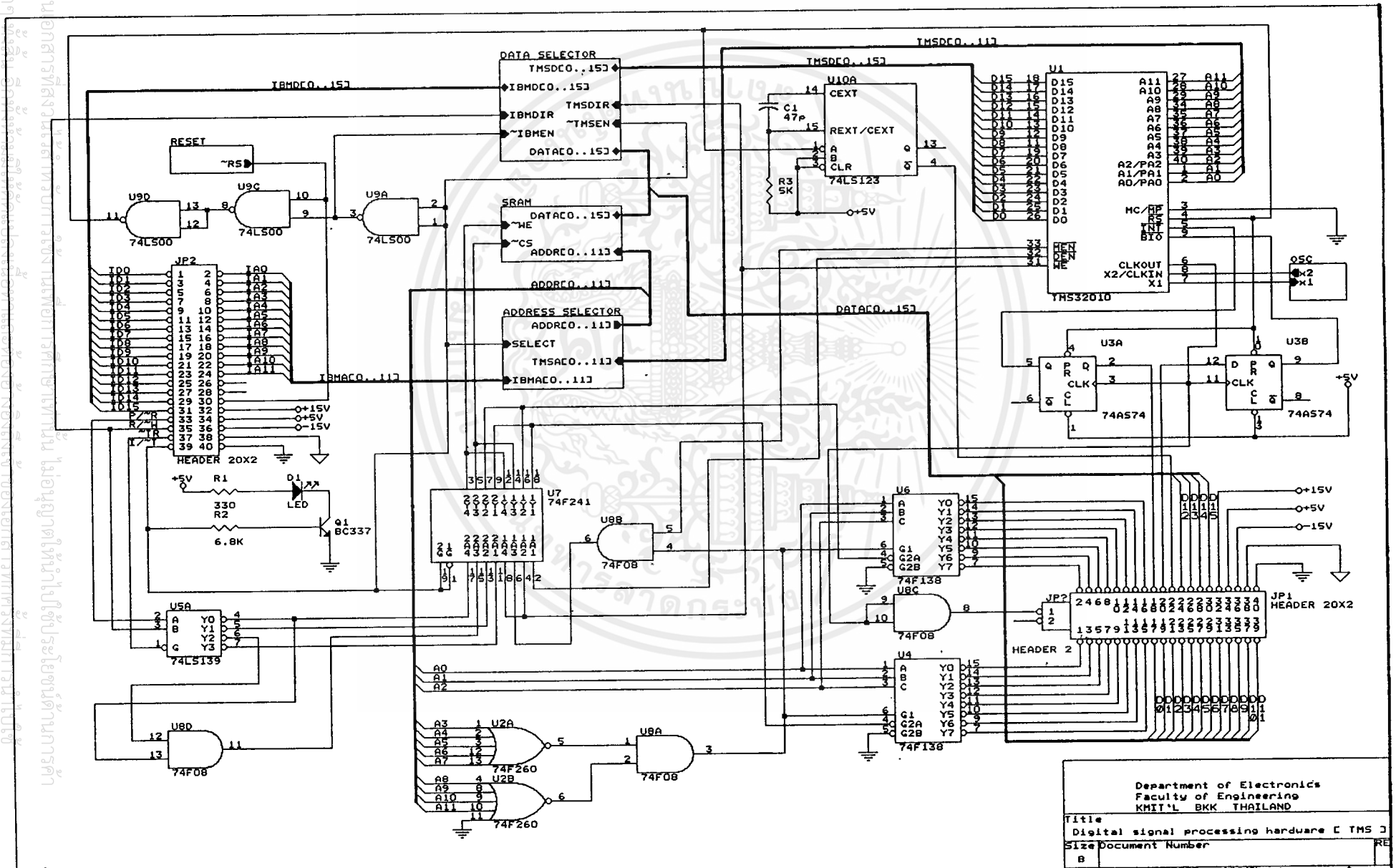
1.6 วงจรหน่วยความจำ (MEMORY UNIT)

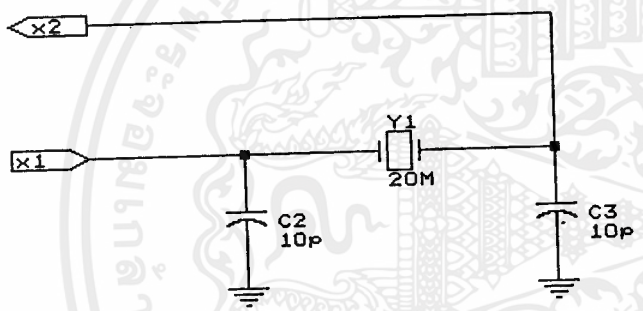
1.7 วงจรสร้างฐานเวลา (TIME BASE)

2) **วงจรในบอร์ด การติดต่อกับไอบีเอ็ม (IBM INTERFACE)**

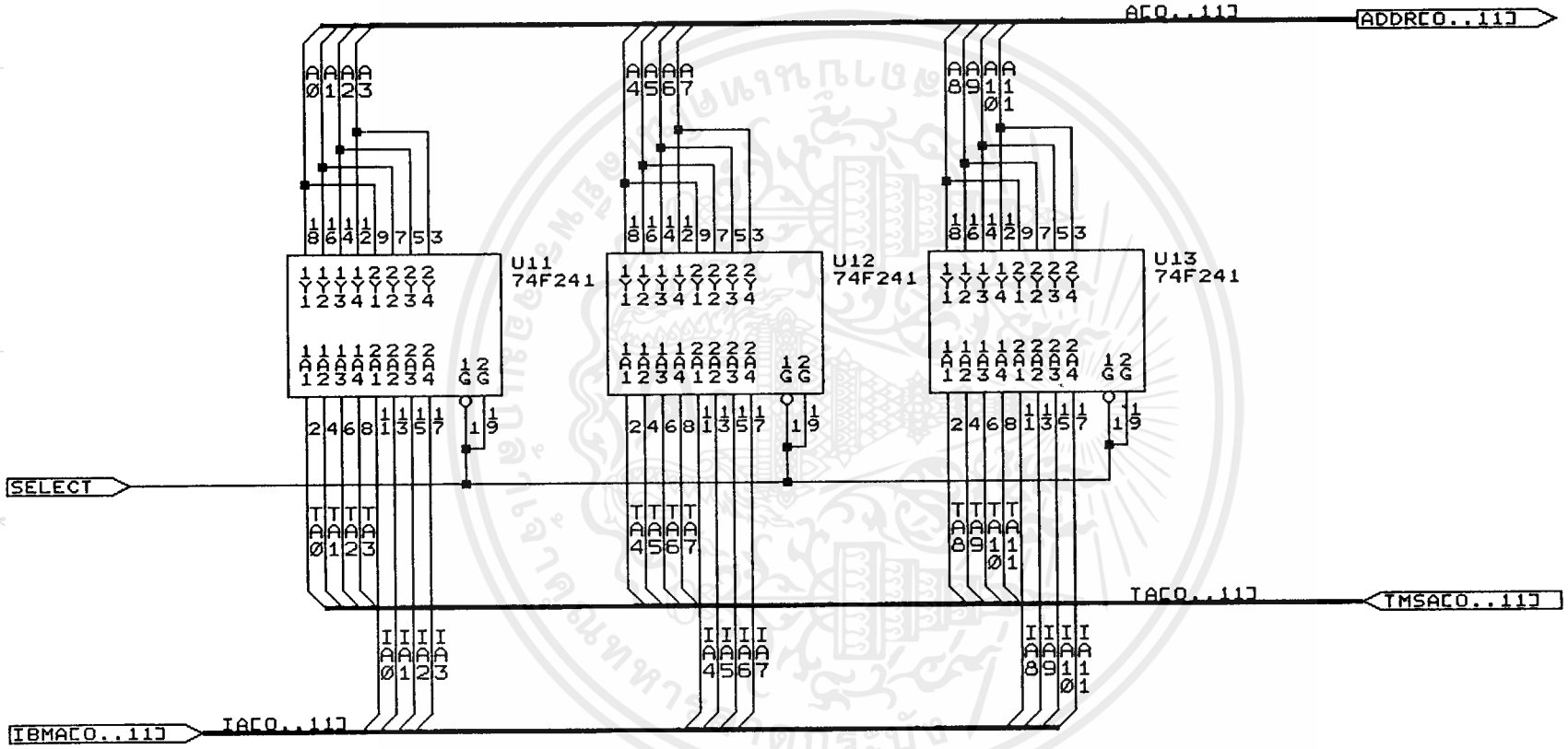
3) **วงจรในบอร์ดอินพุท (ANALOG TO DIGITAL)**

4) **วงจรในบอร์ดเอาท์พุท (DIGITAL TO ANALOG)**

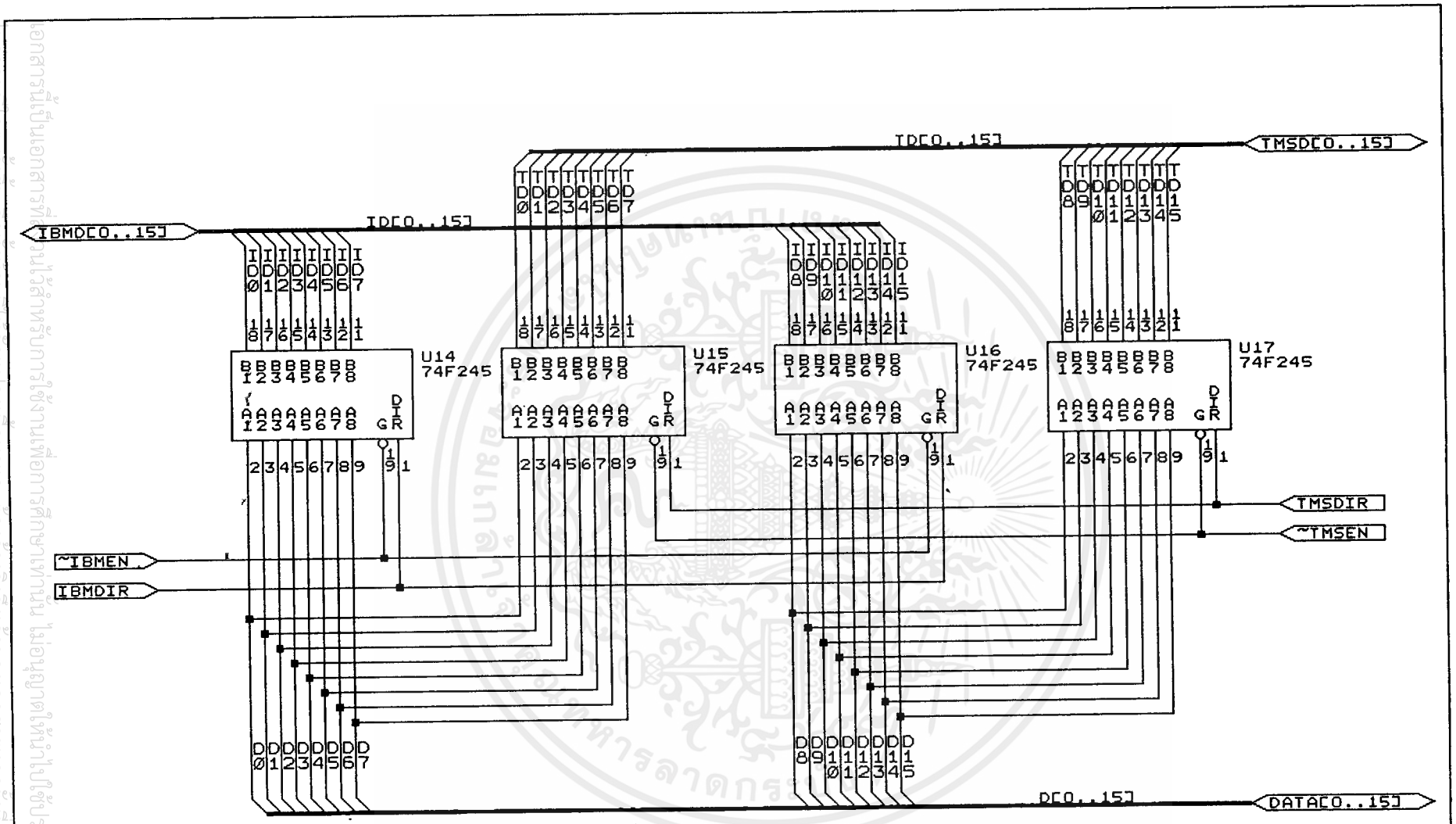

DATA SELECTOR
TMSDCO..15]
IBMDCO..15]
TMSDIR
~TMSEN
IBMDIR
~IBMEN
DATACO..15]
SRAM
DATACO..15]
~WE
~CS
ADDRCO..11]
ADDRESS SELECTOR
ADDRCO..11]
SELECT
TMSACO..11]
IBMACO..11]
RESET
~RS
U9D
U9C
U9A
74LS00
U10A
CEXT
REXT/CEXT
74LS123
C1
47p
R3
5K
+5V
U1
TMS32010
MC/MP
RS
INT
BIO
CLKOUT
X2/CLKIN
X1
OSC
x2
x1
U3A
U3B
74AS74
JP2
HEADER 20X2
+15V
+5V
-15V
R1
330
R2
6.8K
D1
LED
Q1
BC337
U7
74F241
U8B
74F08
U6
74F138
U8C
74F08
JP?
HEADER 2
JP1
HEADER 20X2
U5A
74LS139
U8D
74F08
U4
74F138
U2A
U2B
74F260
74F260
U8A
74F08
Department of Electronics
Faculty of Engineering
KMIT'L BKK THAILAND
Title
Digital signal processing harduare [TMS]
Size
B
Document Number



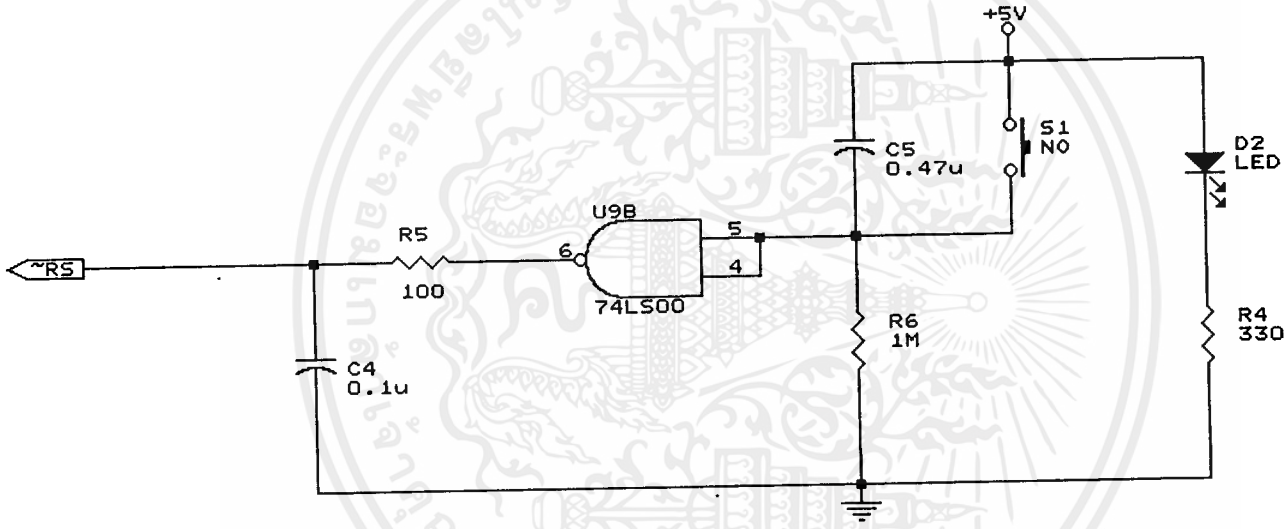
Title	OSCILATOR CIRCUIT	
Size A	Document Number	REV
Date: September 17, 1992	Sheet 3 of 6	



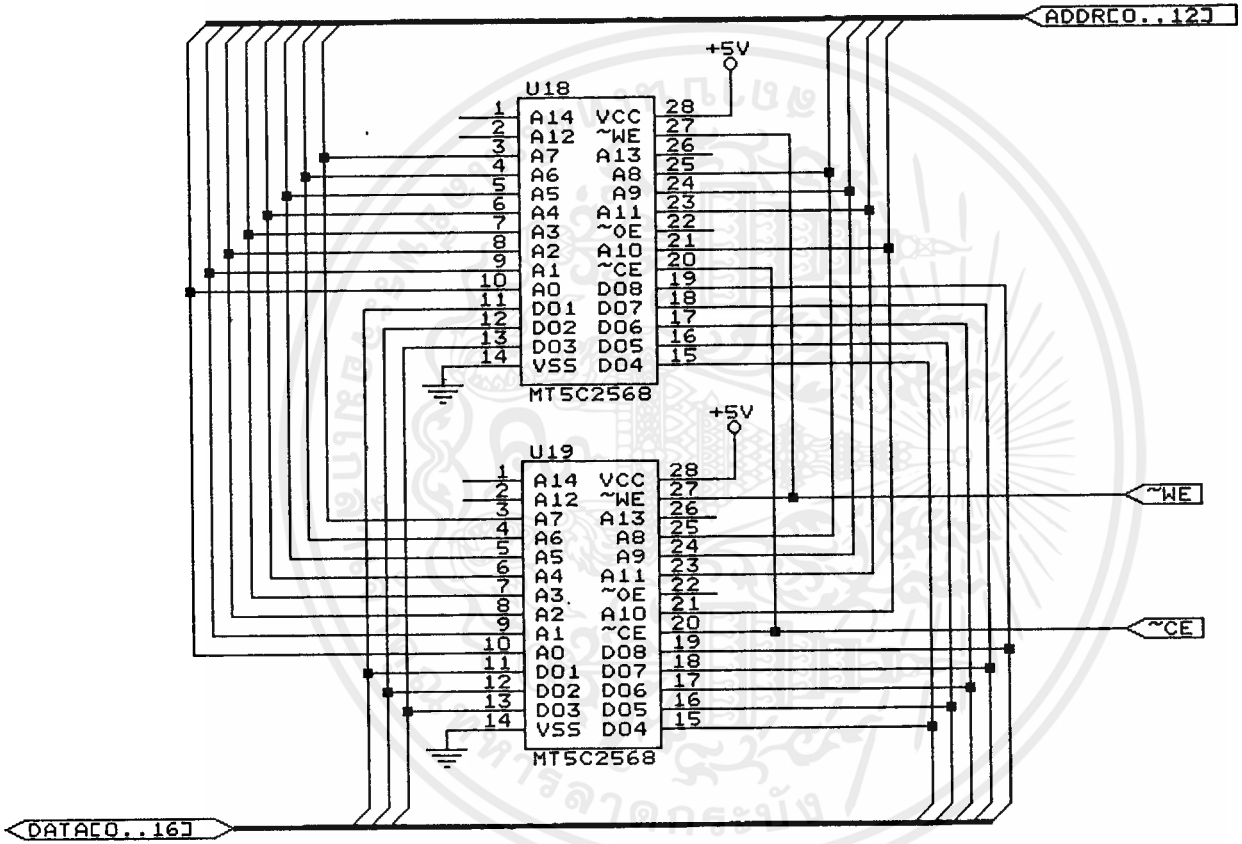
Title: ADDRESS SELECTOR		
Size: A	Document Number	REV
Date: September 17, 1992	Sheet 2 of 6	



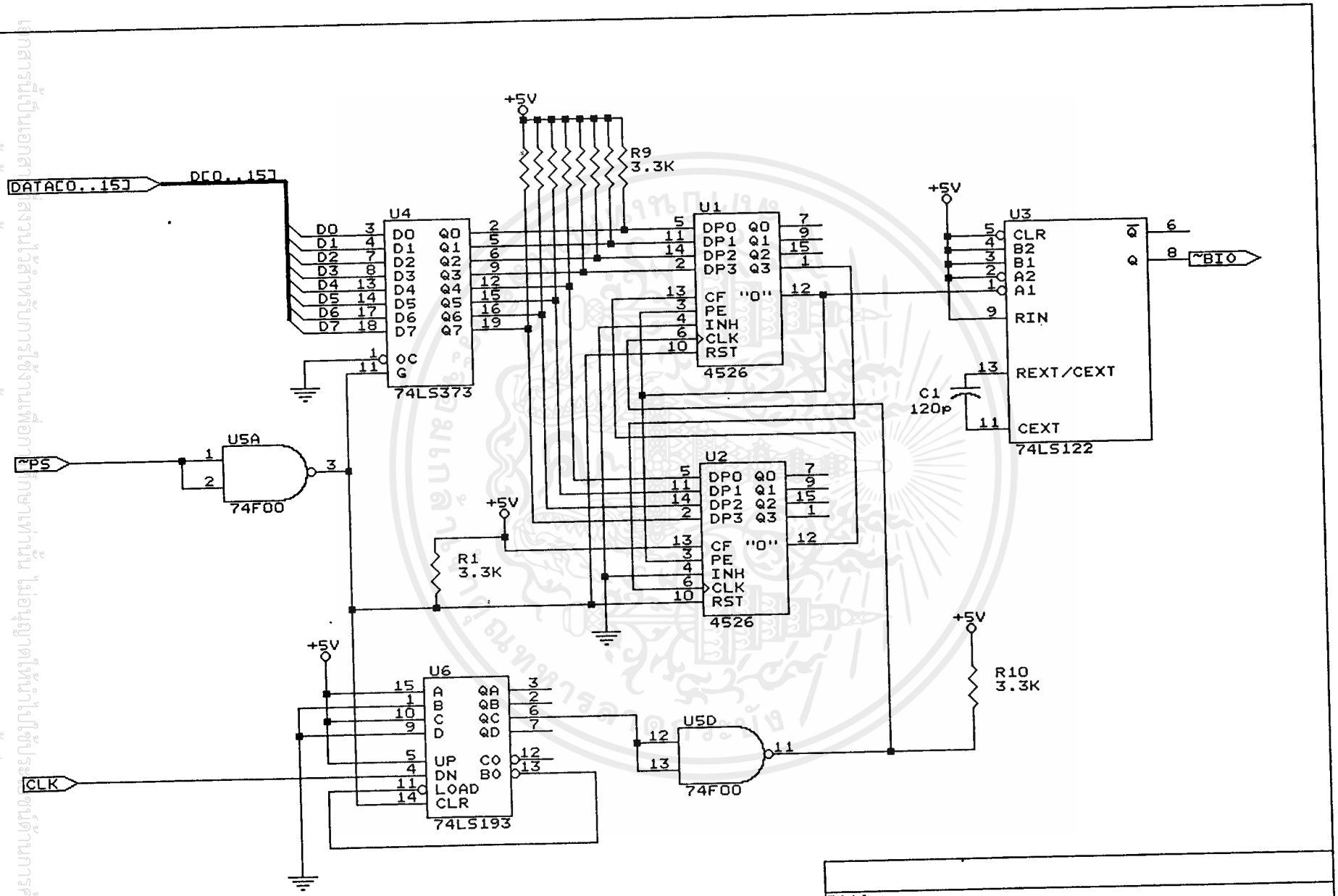
Title: 3-STATE 2-DIRECTION DATA SELECTOR		
Size: A	Document Number	REV
Date: September 17, 1992	Sheet 4 of 6	



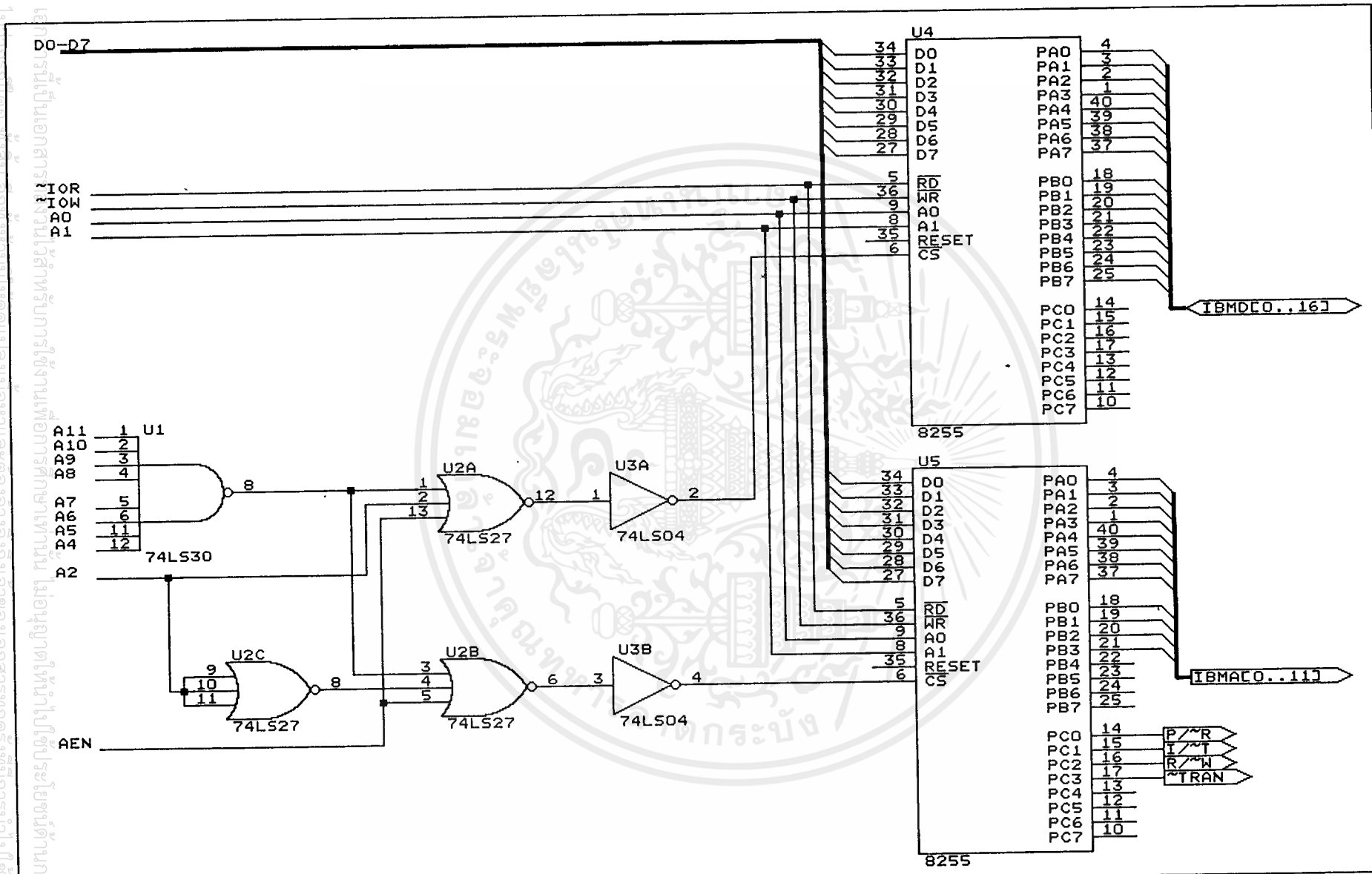
Title	MANUAL & POWER ON RESET	
Size A	Document Number	REV
Date: September 17, 1992	Sheet 6 of 6	



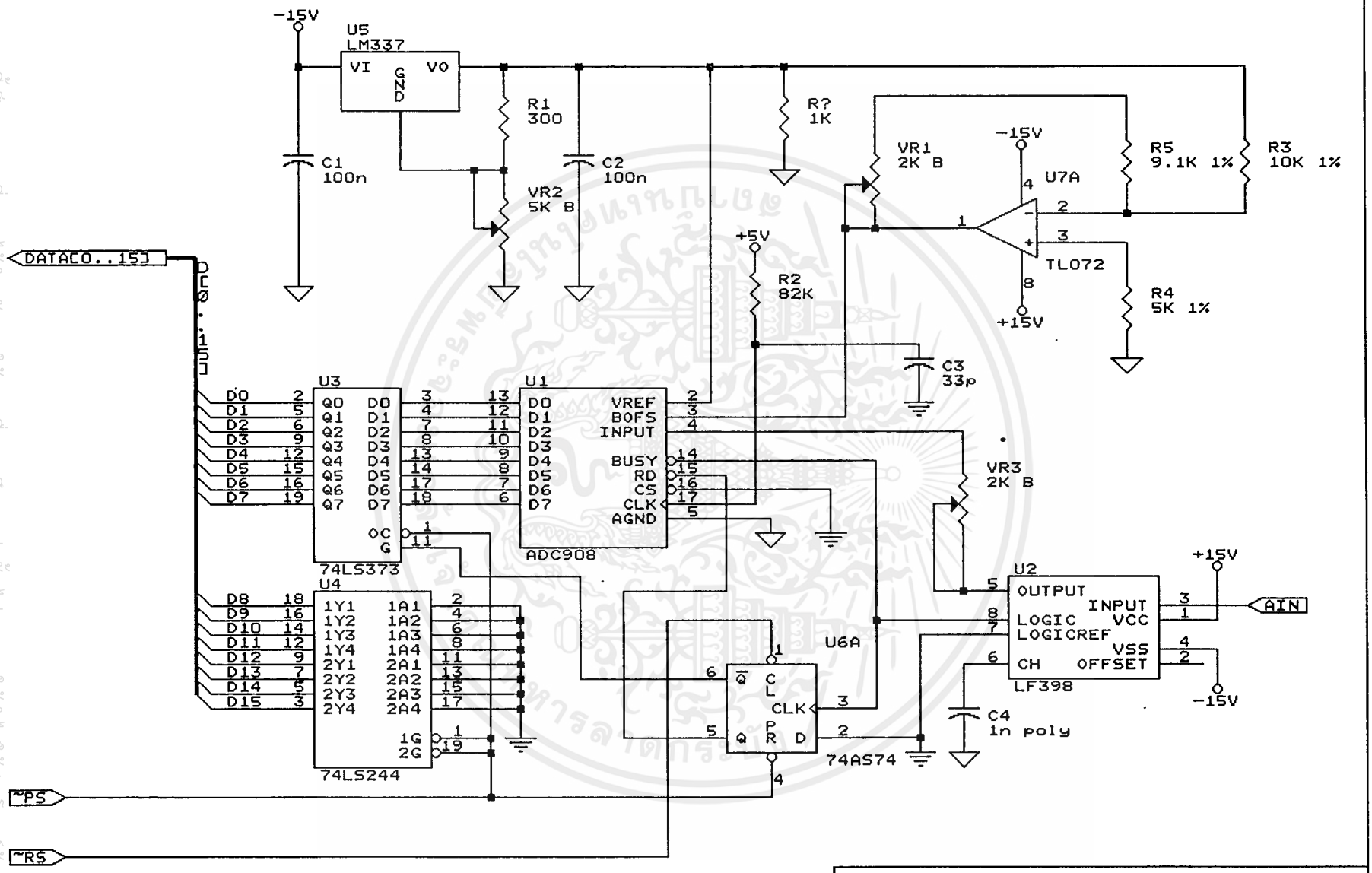
Title: MEMORY UNIT (32K x 8)		
Size: A	Document Number	REV
Date: September 17, 1992	Sheet of	



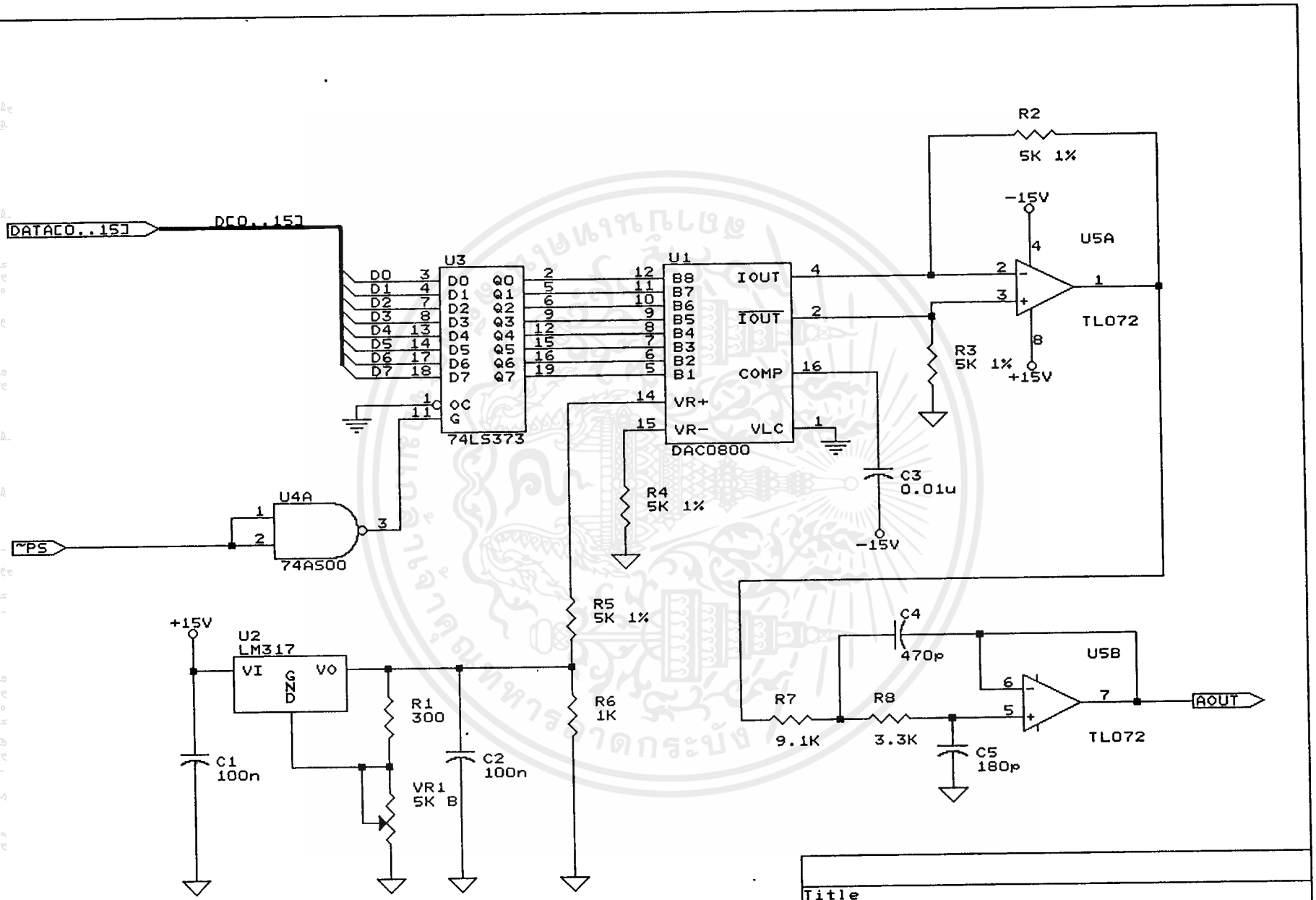
Title	TIME BASE	
Size A	Document Number	REV
Date: September 17, 1992	Sheet 1 of 1	



Title	8255 INTERFACE	
Size A	Document Number	REV
Date: September 17, 1992	Sheet of	



Title	ANALOG TO DIGITAL	
Size A	Document Number	REV
Date: September 24, 1992	Sheet 1 of	


DATA[0..15]
D[0..15]
U3
74LS373
U1
DAC0800
U4A
74A500
~PS
R2
5K 1%
-15V
U5A
TL072
R3
5K 1%
+15V
C3
0.01u
R4
5K 1%
R5
5K 1%
+15V
U2
LM317
VI
GND
VO
R1
300
R6
1K
C1
100n
C2
100n
VR1
5K B
C4
470p
R7
9.1K
R8
3.3K
C5
180p
U5B
TL072
AOUT
Title
DIGITAL TO ANALOG
Size
A
Document Number
REV
Date: September 17, 1992
Sheet 1 of 1

ภาคผนวก ข.

LIST PROGRAM

```
#include <owl.h>
#include <bwcc.h>
#include <print.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <string.h>

#define MOUSEX(arg) (arg.LP.Lo)
#define MOUSEY(arg) (arg.LP.Hi)
#define COMMANDMSG(arg) (arg.WParam)

#define EXITTO     0

#define FNOPEN     10
#define FNSAVEAS   20
#define FNSAMPLING 30
#define FNPASSBR   40
#define FNSTBND    50
#define FNFEG      60
#define FNSEGE     70
#define FNTEDG     80
#define FNFEDGE    90

#define NEW       101
#define OPEN      102
#define SAVE      103
#define SAVEAS    104
#define PRINT     105
#define SAMPLING  106
#define PRIPPLE   107
#define SRIPPLE   108
#define FEDGE     109
#define SEDGE     110
#define TEDGE     111
#define FTHEDGE   112

#define LPP       113
#define HP        114
#define BP        115
#define BR        116

#define RCT       117
#define KAISER    118
#define HANN      119
#define HAMM      120
#define BLKMAN    121

#define GAIN      122
#define LOGM      123
#define PHASE     124
#define IMPULSE   125

#define HELPP     126
#define AB        127
#define COMPI     128
#define SENDOP    129

#define INFOM     130

#define OPTIMUM   131

#define VER       132

#define   PA1   0xFF0
#define   PB1   0xFF1
#define   PC1   0xFF2
#define   CW1   0xFF3
#define   PA2   0xFF4
#define   PB2   0xFF5
#define   PC2   0xFFE
#define   CW2   0xFF7


LPSTR lpBuffer;
WORD  cbBuffer;
const int TEXTBUFFER = 81;
const int MAXPOINTS = 32;
const int MARKERSIZE = 30;
HDC   OOHDC;
HDC   PRHDC;
HWND  OOHWND;
COLORREF BACKGROUND;
COLORREF Graphplot[5];
char testtext[TEXTBUFFER];

class Filter
```

```
{
  public:

    Filter();
    ~Filter();

    int   dx(float VX);
    int   dy(float VY);
    void  coordinate(int startx,int starty,int endx,int endy);
    void  setfactors(float lefta1,float bottomb1,
                  float righta2,float upperb2,HWND hwnd);
           void  MakeFilter(int num,int timebase,HWND hwnd);
    void  quantize(float *h,int num,int *hh);
    void  converse(float *h,int *hh,int num,HWND hwnd);
    int   KaiserOptimize(int type,float center,float Aa,float Ap,float Fp,
                 float Fa,float sampling,float *h,int *hh,
                 HWND hwnd,int wintype);
    float I(float x);
    unsigned long factorial(int x);
    HDC    GetPrinterDC (void);
    void   PageGDICalls(HDC hdcPrn,short cxPage,short cyPage,HWND hwnd);
    BOOL   PrintMyPage(HWND hwnd);

    int    Filtertype;
    float  Sampling;
    float  Pbripple;
    float  Stopbatt;
    float  Firstedge;
    float  Secondedge;
    float  Thirdedge;
    float  Fourthedge;
    int    Windowtype;
    int    Display;
    int    Optimize;

    int    TIMEBASE;
    int    LENGTH;

    float *h;
    int   hh[200];


    int   VH,VW,PX,PY;
    float X1,X2,Y1,Y2;
    float XF,YF;
    int   Canplot;
    int   Change;
    int   DisplayChange;
    int   WindowChange;
    int   Success;
    int   first;
    int   Printer;
    int   Repaint;

    char  Savename[80];
    struct spec {
      int    type;
      float  sampling;
      float  pbripple;
      float  stopbatt;
      float  firstedge;
      float  secondedge;
      float  thirdedge;
      float  fourthedge;
      int    windowtype;
      int    display;
      int    optimize;
    } savespec;


};

class TMarkerApplication : public TApplication
{
  public:
    TMarkerApplication( LPSTR lpszName, HANDLE hInstance,
                        HANDLE hPrevInstance, LPSTR lpszCmdLine,int nCmdShow);
    virtual void InitMainWindow();
};

class TMarkerWindow : public TWindow
{
  public:
    TMarkerWindow (PTWindowsObject pwParent, LPSTR lpszTitle,
                PTModule pmModule);
    virtual LPSTR GetClassName();
    virtual void GetWindowClass(WNDCLASS&);
    virtual void WMLButtonDown (TMessage& Msg) = [WM_LBUTTONDOWN];
    virtual void Paint(HDC hdc,PAINTSTRUCT& ps);
```

```
    virtual void WMCommand(TMessage& Msg)  = [WM_COMMAND];
    virtual void WMSetfocus(TMessage& Msg) = [WM_SETFOCUS];
    virtual void WMErasebkgnd(TMessage& Msg) = [WM_ERASEBKGND];

    int cpt;

    TDialog *ptTestDialog;
    TDialog *ptSaveasDialog;
    TDialog *ptSamplingDialog;
    TDialog *ptPassbandDialog;
    TDialog *ptStopbandDialog;
    TDialog *ptFirstedgeDialog;
    TDialog *ptSecondedgeDialog;
    TDialog *ptThirdedgeDialog;
    TDialog *ptFourthedgeDialog;

    POINT pt[MAXPOINTS];
};

class TTestDialog : public TDialog
{
  public:

    TTestDialog (PTWindowsObject  ptParent, LPSTR lpszName,
             LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TSaveasDialog : public TDialog
{
  public:

    TSaveasDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                   LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TSamplingDialog : public TDialog
{
  public:

    TSamplingDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                           LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TPassbandDialog : public TDialog
{
  public:

    TPassbandDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                           LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TStopbandDialog : public TDialog
{
  public:

    TStopbandDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                           LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TFirstedgeDialog : public TDialog
{
  public:

    TFirstedgeDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                            LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TSecondedgeDialog : public TDialog
{
  public:

    TSecondedgeDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                           LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TThirdedgeDialog : public TDialog
```

```
{
  public:

    TThirdedgeDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                      LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};

class TFourthedgeDialog : public TDialog
{
  public:

    TFourthedgeDialog (PTWindowsObject  ptParent, LPSTR lpszName,
                       LPSTR lpTestData, WORD cbBufSize);
    virtual void WMCommand(TMessage& Msg) = [WM_COMMAND];
    virtual void WMInitDialog(TMessage& Msg) = [WM_INITDIALOG];
};


void DrawMarker (HDC hDC,int x,int y);

char samm[80];
Filter FIR;



int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                                LPSTR lpszCmdLine, int nCmdShow)
{
  HANDLE hDLL;
  TMarkerApplication Marker("Marker",hInstance,hPrevInstance,lpszCmdLine,nCmdShow);

    hDLL = LoadLibrary("BWCC.DLL");
    Marker.Run();
    if(hDLL) FreeLibrary(hDLL);
    return Marker.Status;
}

TMarkerApplication::TMarkerApplication(LPSTR lpszName,
               HANDLE hInstance, HANDLE hPrevInstance,
               LPSTR lpszCmdLine, int nCmdShow)
              :TApplication(lpszName,hInstance,
                 hPrevInstance,lpszCmdLine,nCmdShow)

{
}

void TMarkerApplication::InitMainWindow()
{
 MainWindow = new TMarkerWindow( NULL,"FIR FILTER",NULL);
}

TMarkerWindow::TMarkerWindow(PTWindowsObject pwParent,
          LPSTR lpszTitle, PTModule pmModule)
         :TWindow(pwParent, lpszTitle, pmModule)
{
  cpt =0;
  ptTestDialog       = (TTestDialog *) 0L;
  ptSaveasDialog     = (TSaveasDialog *)0L;
  ptSamplingDialog   = (TSamplingDialog *)0L;
  ptPassbandDialog   = (TPassbandDialog *)0L;
  ptStopbandDialog   = (TStopbandDialog *)0L;
  ptFirstedgeDialog  = (TFirstedgeDialog *)0L;
  ptSecondedgeDialog = (TSecondedgeDialog *)0L;
  ptThirdedgeDialog  = (TThirdedgeDialog *)0L;
  ptFourthedgeDialog = (TFourthedgeDialog *)0L;


}

LPSTR TMarkerWindow::GetClassName()
{
  return "Marker:Main";
}

void TMarkerWindow::GetWindowClass( WNDCLASS& wc )
{
  TWindow::GetWindowClass(wc);
  wc.hIcon=LoadIcon(wc.hInstance,"snapshot");
  wc.lpszMenuName = "#1";
}

void TTestDialog::TTestDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)

{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}
```

```
void TSaveasDialog::TSaveasDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TSamplingDialog::TSamplingDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TPassbandDialog::TPassbandDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TStopbandDialog::TStopbandDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TFirstedgeDialog::TFirstedgeDialog(PTWindowsObject  ptParent, LPSTR lpszName,
                        LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TSecondedgeDialog::TSecondedgeDialog(PTWindowsObject  ptParent,
                    LPSTR lpszName,LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TThirdedgeDialog::TThirdedgeDialog(PTWindowsObject  ptParent,
                    LPSTR lpszName,LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void TFourthedgeDialog::TFourthedgeDialog(PTWindowsObject  ptParent,
                    LPSTR lpszName,LPSTR lpTestData, WORD cbBufSize)
                        :TDialog(ptParent,lpszName)
{
     lpBuffer =  lpTestData;
     cbBuffer =  cbBufSize;
}

void Filter::Filter(void)
{
     Filtertype      = 0;
     Sampling        = 40000.0;
     Pbripple        = 5.0;
     Stopbatt        = 20.0;
     Firstedge       = 1000;
     Secondedge      = 10000.0;
     Thirdedge       = 11000.0;
     Fourthedge      = 19000.0;
     Windowtype      = 1;
     Display         = 0;
     Canplot         = 0;
     Success         = 0;
     Change          = 0;
     DisplayChange   = 0;
     WindowChange    = 0;
     first           = 0;
     Optimize        = 0;
     Printer         = 0;
     Repaint         = 1;
     BACKGROUND      = RGB(0,255,255);
     strcpy(Savename,"NONAME.FIR");
     Graphplot[0] = RGB(0,0,255);
     Graphplot[1] = RGB(200,200,0);
```

```
    Graphplot[2] = RGB(255,20,20);
    Graphplot[3] = RGB(255,0,255);
    Graphplot[4] = RGB(0,255,255);

    h = (float *)calloc(3002,sizeof(float));
}

void Filter::~Filter(void)
{

    free(h);

}

void TTestDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  hCtl = Msg.Receiver;
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}

void TSaveasDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  hCtl = Msg.Receiver;
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}

void TSamplingDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  char buf[30];

  sprintf(buf,"%.2f",FIR.Sampling);
  hCtl = Msg.Receiver;
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);

}

void TPassbandDialog::WMInitDialog(TMessage& Msg)
{

  HWND hCtl;
  char buf[30];

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Pbripple);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}

void TStopbandDialog::WMInitDialog(TMessage& Msg)
{
  char buf[20];
  HWND hCtl;

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Stopbatt);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(1);
  EnableWindow(hCtl,TRUE);
}

void TFirstedgeDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  char buf[20];

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Firstedge);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}

void TSecondedgeDialog::WMInitDialog(TMessage& Msg)
{
  char buf[20];
  HWND hCtl;

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Secondedge);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
```

```
}

void TThirdedgeDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  char buf[20];

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Thirdedge);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}

void TFourthedgeDialog::WMInitDialog(TMessage& Msg)
{
  HWND hCtl;
  char buf[20];

  hCtl = Msg.Receiver;
  sprintf(buf,"%.2f",FIR.Fourthedge);
  SetDlgItemText(hCtl,2,buf);
  hCtl = GetItemHandle(2);
  EnableWindow(hCtl,TRUE);
}


void TTestDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;
      GetDlgItemText(HWindow;2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,10,0L);
      break;
    }
}

void TSaveasDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,20,0L);

      break;
    }
}

void TSamplingDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,30,0L);

      break;
    }
}

void TPassbandDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,40,0L);

      break;
    }
}

void TStopbandDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;
```

```
      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,50,0L);

      break;
    }
}

void TFirstedgeDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,60,0L);

      break;
    }
}

void TSecondedgeDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,70,0L);

      break;
    }
}

void TThirdedgeDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,80,0L);

      break;
    }
}

void TFourthedgeDialog::WMCommand(TMessage& Msg)
{
    switch(Msg.WParam) {
       case 1:
      HWND hwndParent;

      GetDlgItemText(HWindow,2,lpBuffer,cbBuffer);
      hwndParent = GetParent(Msg.Receiver);
      CloseWindow(Msg.Receiver);
      SendMessage(hwndParent,WM_COMMAND,90,0L);

      break;
    }
}

void TMarkerWindow::WMLButtonDown(TMessage& Msg)
{

  PAINTSTRUCT ps;

  if(cpt<MAXPOINTS) {
    pt[cpt].x = MOUSEX(Msg);
    pt[cpt].y = MOUSEY(Msg);
    InvalidateRect(HWindow,NULL,FALSE);
  }
}

void TMarkerWindow::Paint(HDC hdc,PAINTSTRUCT& ps)
{

  OOHDC   = ps.hdc;
  OOHWND  = HWindow;

  RECT    r;
  if(!FIR.Canplot) {
```

```
            SelectObject(hdc,GetStockObject(LTGRAY_BRUSH));
      GetClientRect(HWindow,&r);
      Rectangle(ps.hdc,r.left,r.top,r.right,r.bottom);
    }

}

void TMarkerWindow::WMSetfocus(TMessage& Msg)
{
     UpdateWindow(HWindow);
     if(FIR.Canplot&&(COMMANDMSG(Msg)==WA_ACTIVE) ) {
        FIR.Change  =  1;
     }
}

void TMarkerWindow::WMErasebkgnd(TMessage& Msg)
{
  HWND hwnd;
     hwnd = GetActiveWindow();
     UpdateWindow(Msg.Receiver);
     if(FIR.Canplot&&FIR.Repaint) {
        FIR.Change  =  1;
     SendMessage(Msg.Receiver,WM_COMMAND,GAIN+(int)FIR.Display,0L);
            SetActiveWindow(hwnd);
     }
     FIR.Repaint = 1;

}

void TMarkerWindow::WMCommand(TMessage& Msg)
{
  char buffer[80];
  HMENU hmenu;
  HMENU hmenuTop;

  if(COMMANDMSG(Msg)==EXITTO)
     SendMessage(HWindow,WM_SYSCOMMAND,SC_CLOSE,0L);
  else {
     switch(COMMANDMSG(Msg)) {


    case  FNOPEN:
     {
       FILE *fp;
       char title[80];


       if( !strcmp(testtext,"") ) strcpy(testtext,"NONAME.FIR");
       if( (fp = fopen(testtext,"rb")) != NULL ) {

           strcpy(FIR.Savename,testtext);
           sprintf(title,"FIR FILTER - ( %s )",FIR.Savename);
           SetCaption(title);
           fread(&FIR.savespec,sizeof(FIR.savespec),1,fp);
           FIR.Sampling   = FIR.savespec.sampling;
           FIR.Pbripple   = FIR.savespec.pbripple;
           FIR.Stopbatt   = FIR.savespec.stopbatt;
           FIR.Firstedge  = FIR.savespec.firstedge;
           FIR.Secondedge = FIR.savespec.secondedge;
           FIR.Thirdedge  = FIR.savespec.thirdedge;
           FIR.Fourthedge = FIR.savespec.fourthedge;

           hmenuTop = GetMenu(HWindow);

           hmenu = GetSubMenu(hmenuTop,2);
           CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
           FIR.Filtertype = FIR.savespec.type;
           CheckMenuItem(hmenu,2*FIR.Filtertype,MF_CHECKED|MF_BYPOSITION);

           hmenu = GetSubMenu(hmenuTop,3);
           CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
           FIR.Windowtype = FIR.savespec.windowtype;
           CheckMenuItem(hmenu,FIR.Windowtype,MF_CHECKED|MF_BYPOSITION);

           hmenu = GetSubMenu(hmenuTop,5);
           FIR.Optimize   = FIR.savespec.optimize;
           if(FIR.Optimize) {
            CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
           }
           else {
             CheckMenuItem(hmenu,2,MF_UNCHECKED|MF_BYPOSITION);
           }

           hmenu = GetSubMenu(hmenuTop,4);
           CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
           FIR.Display    = FIR.savespec.display;
           CheckMenuItem(hmenu,FIR.Display,MF_CHECKED|MF_BYPOSITION);
           //SendMessage(Msg.Receiver,WM_COMMAND,GAIN+(int)FIR.Display,0L);
           fclose(fp);
```

```
        FIR.Change = 1;
        FIR.WindowChange = 0;
        FIR.first = 0;
        SendMessage(HWindow,WM_COMMAND,COMPI,0L);

      }
      else {
        strcpy(FIR.Savename,testtext);
        sprintf(title,"FIR FILTER - ( %s )",FIR.Savename);
        SetCaption(title);

        FIR.Sampling   = 40000.0;
        FIR.Pbripple   = 5.0;
        FIR.Stopbatt   = 20.0;
        FIR.Firstedge  = 1000.0;
        FIR.Secondedge = 10000.0;
        FIR.Thirdedge  = 11000.0;
        FIR.Fourthedge = 19000.0;

        hmenuTop = GetMenu(HWindow);

        hmenu = GetSubMenu(hmenuTop,2);
        CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Filtertype = 0;
        CheckMenuItem(hmenu,2*FIR.Filtertype,MF_CHECKED|MF_BYPOSITION);

        hmenu = GetSubMenu(hmenuTop,3);
        CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Windowtype = 1;
        CheckMenuItem(hmenu,FIR.Windowtype,MF_CHECKED|MF_BYPOSITION);

        hmenu = GetSubMenu(hmenuTop,5);
        FIR.Optimize   = 0;
        if(FIR.Optimize) {
          CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
        }
        else {
            CheckMenuItem(hmenu,2,MF_UNCHECKED|MF_BYPOSITION);
        }

        hmenu = GetSubMenu(hmenuTop,4);
        CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Display    = 0;
        CheckMenuItem(hmenu,FIR.Display,MF_CHECKED|MF_BYPOSITION);
        FIR.Change = 1;
        FIR.WindowChange = 0;
        FIR.first = 1;
        SendMessage(HWindow,WM_COMMAND,COMPI,0L);
      }
      break;
    }
  case  NEW:
    {
      char title[80];

        strcpy(FIR.Savename,"NONAME.FIR");
        sprintf(title,"FIR FILTER - ( %s )",FIR.Savename);
        SetCaption(title);
        FIR.Sampling   = 40000.0;
        FIR.Pbripple   = 5.0;
        FIR.Stopbatt   = 20.0;
        FIR.Firstedge  = 1000.0;
        FIR.Secondedge = 10000.0;
        FIR.Thirdedge  = 11000.0;
        FIR.Fourthedge = 19000.0;

        hmenuTop = GetMenu(HWindow);

        hmenu = GetSubMenu(hmenuTop,2);
        CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Filtertype = 0;
        CheckMenuItem(hmenu,2*FIR.Filtertype,MF_CHECKED|MF_BYPOSITION);

        hmenu = GetSubMenu(hmenuTop,3);
        CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Windowtype = 1;
        CheckMenuItem(hmenu,FIR.Windowtype,MF_CHECKED|MF_BYPOSITION);

        hmenu = GetSubMenu(hmenuTop,5);
        FIR.Optimize   = 0;
        if(FIR.Optimize) {
         CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
        }
        else {
          CheckMenuItem(hmenu,2,MF_UNCHECKED|MF_BYPOSITION);
        }

        hmenu = GetSubMenu(hmenuTop,4);
        CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
        FIR.Display    = 0;
```

```
        CheckMenuItem(hmenu,FIR.Display,MF_CHECKED|MF_BYPOSITION);
        FIR.Change = 1;
        FIR.WindowChange = 0;
        FIR.first = 1;
        SendMessage(HWindow,WM_COMMAND,COMPI,0L);

           break;
      }
case  FNSAVEAS:
 {
   FILE *fp;
   char title[80];

   if( (fp = fopen(testtext,"wb")) != NULL ) {
       strcpy(FIR.Savename,testtext);
       FIR.savespec.type = FIR.Filtertype;
       FIR.savespec.sampling = FIR.Sampling;
       FIR.savespec.pbripple = FIR.Pbripple;
       FIR.savespec.stopbatt = FIR.Stopbatt;
       FIR.savespec.firstedge= FIR.Firstedge;
       FIR.savespec.secondedge=FIR.Secondedge;
       FIR.savespec.thirdedge= FIR.Thirdedge;
       FIR.savespec.fourthedge=FIR.Fourthedge;
       FIR.savespec.windowtype=FIR.Windowtype;
       FIR.savespec.display  = FIR.Display;
       FIR.savespec.optimize = FIR.Optimize;

       fwrite(&FIR.savespec,sizeof(FIR.savespec),1,fp);
       fclose(fp);
       sprintf(title,"FIR FILTER - ( %s )",FIR.Savename);
       SetCaption(title);
   }
   else MessageBox(Msg.Receiver,"Can not write to file","MESSAGE",MB_OK);
   break;
 }

case  FNSAMPLING:
 {
  float tmp;
  char  *er;
   tmp = strtod(testtext,&er);
   if( *testtext == NULL ) tmp = FIR.Sampling;
   if( (*er != NULL) ) tmp = FIR.Sampling;
   if( (tmp < 4000.0)||(tmp <= 1.0) ) {
     FIR.Sampling = 4000.0;
     FIR.WindowChange = 0;
       }
   if(tmp > 110000.0) {
     FIR.Sampling = 110000.0;
     FIR.WindowChange = 0;
       }
   else {
     FIR.Sampling = tmp;
     FIR.WindowChange = 0;
       }

   SendMessage(HWindow,WM_COMMAND,COMPI,0L);
   break;
     }
case  FNPASSBR:
     {
  float tmp;
      char *er;
   tmp = strtod(testtext,&er);
   if( *testtext == NULL ) tmp = FIR.Pbripple;
   if(*er != NULL) tmp = FIR.Pbripple;
   if(tmp < 0.001) {
     FIR.Pbripple = 0.001;
     FIR.WindowChange = 0;
     SendMessage(HWindow,WM_COMMAND,COMPI,0L);
     break;
       }
   if(tmp > 20) {
     FIR.Pbripple = 20;
     FIR.WindowChange = 0;
       }
   else {
     FIR.Pbripple = tmp;
     FIR.WindowChange = 0;
       }

   SendMessage(HWindow,WM_COMMAND,COMPI,0L);
   break;
 }
case  FNSTBND:
     {
  float tmp;
      char  *er;
   tmp = strtod(testtext,&er);
   if( *testtext == NULL ) tmp = FIR.Stopbatt;
```

```
    if(*er != NULL) tmp = FIR.Stopbatt;
    if(tmp < 5.0) {
      FIR.Pbripple = 5.0;
      FIR.WindowChange = 0;
      SendMessage(HWindow,WM_COMMAND,COMPI,0L);
          break;
        }
    if(tmp > 200.0) {
      FIR.Pbripple = 200.0;
      FIR.WindowChange =  0;
        }
    else {
      FIR.Stopbatt = tmp;
      FIR.WindowChange = 0;
        }
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
  }
 case  FNFEG:
  {
   float tmp;
   char  *er;

    tmp = strtod(testtext,&er);
    if( *testtext == NULL ) tmp = FIR.Firstedge;
    if(*er != NULL) tmp = FIR.Firstedge;
    if((tmp<(FIR.Sampling/2.0)))
    {
       /* &&
       (tmp<FIR.Secondedge) &&
       (tmp<FIR.Thirdedge) &&
       (tmp<FIR.Fourthedge) )*/

          FIR.Firstedge = tmp;
          FIR.WindowChange = 0;
          SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    }
    else putch(7);
    break;
   }
 case  FNSEGE:
  {
   float tmp;
       char  *er;
    tmp = strtod(testtext,&er);
    if( *testtext == NULL ) tmp = FIR.Secondedge;
    if(*er != NULL) tmp = FIR.Secondedge;
    if( tmp<((FIR.Sampling/2.0)))
    {   /*&&
       (tmp>FIR.Firstedge) &&
       (tmp<FIR.Thirdedge) &&
       (tmp<FIR.Fourthedge) )*/


         FIR.Secondedge = tmp;
         FIR.WindowChange = 0;
         SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    }
        else putch(7);
    break;
      }
 case  FNTEDG:
  {
   float tmp;
       char  *er;
    tmp = strtod(testtext,&er);
    if( *testtext == NULL ) tmp = FIR.Thirdedge;
    if(*er != NULL) tmp = FIR.Thirdedge;
    if( tmp<((FIR.Sampling/2.0)))
        {
       /*&&
       (tmp>FIR.Firstedge) &&
       (tmp>FIR.Secondedge) &&
       (tmp<FIR.Fourthedge) ) */

         FIR.Thirdedge = tmp;
         FIR.WindowChange = 0;
         SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    }
        else putch(7);
    break;
      }
 case  FNFEDGE:
      {
   float tmp;
       char  *er;
    tmp = strtod(testtext,&er);
    if( *testtext == NULL ) tmp = FIR.Fourthedge;
    if(*er != NULL) tmp = FIR.Fourthedge;
    if( tmp<((FIR.Sampling/2.0)))
```

```
    {
       /*&&
       (tmp>FIR.Firstedge) &&
       (tmp>FIR.Secondedge) &&
       (tmp>FIR.Thirdedge) ) */

          FIR.Fourthedge = tmp;
              FIR.WindowChange = 0;
          SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    }
    else putch(7);
    break;

      }
 case HELPP:
    TDialog *PHelp;
    /*PHelp = new TDialog(this,"HELP");*/
    PHelp = new TDialog(this,"HELP");
    GetApplication()->ExecDialog(PHelp);
    break;

 case AB:
    TDialog *PAbout;
    PAbout = new TDialog(this,"ABOUT");
    GetApplication()->ExecDialog(PAbout);
    break;
 case VER:
    TDialog *PAversion;
    PAversion = new TDialog(this,"VERSION");
    GetApplication()->ExecDialog(PAversion);
    break;

 case OPEN:
    FIR.Repaint = 0;

    if(ptTestDialog && IsWindow(ptTestDialog->HWindow)) {
       SetActiveWindow(ptTestDialog->HWindow);
    }
    else {
       ptTestDialog = new TTestDialog(this,"TEST",testtext,TEXTBUFFER);
    }
    GetApplication()->MakeWindow(ptTestDialog);
    break;
 case SAVEAS:
    if(ptSaveasDialog && IsWindow(ptSaveasDialog->HWindow)) {
       SetActiveWindow(ptSaveasDialog->HWindow);
    }
    else {
       ptSaveasDialog = new TSaveasDialog(this,"SAVEAS",testtext,TEXTBUFFER);
    }
    GetApplication()->MakeWindow(ptSaveasDialog);
    break;
 case SAMPLING:
    FIR.Repaint = 0;

    if(ptSamplingDialog && IsWindow(ptSamplingDialog->HWindow)) {
       SetActiveWindow(ptSamplingDialog->HWindow);
    }
    else {
       ptSamplingDialog = new TSamplingDialog(this,"SAMPLING",testtext,TEXTBUFFER);
    }
    GetApplication()->MakeWindow(ptSamplingDialog);

    break;

 case PRIPPLE:
        FIR.Repaint = 0;

    if(ptPassbandDialog && IsWindow(ptPassbandDialog->HWindow)) {
       SetActiveWindow(ptPassbandDialog->HWindow);
    }
    else {
       ptPassbandDialog = new TPassbandDialog(this,"PASSBR",
                                              testtext,TEXTBUFFER);
    }
    GetApplication()->MakeWindow(ptPassbandDialog);
    break;
 case SRIPPLE:
    FIR.Repaint = 0;

    if(ptStopbandDialog && IsWindow(ptStopbandDialog->HWindow)) {
       SetActiveWindow(ptStopbandDialog->HWindow);
    }
    else {
       ptStopbandDialog = new TStopbandDialog(this,"STBND",
                                              testtext,TEXTBUFFER);
    }
    GetApplication()->MakeWindow(ptStopbandDialog);

    break;
```

```
case FEDGE:
   FIR.Repaint = 0;

   if(ptFirstedgeDialog && IsWindow(ptFirstedgeDialog->HWindow)) {
      SetActiveWindow(ptFirstedgeDialog->HWindow);
   }
   else {
      ptFirstedgeDialog = new TFirstedgeDialog(this,"FEG",testtext,TEXTBUFFER);
   }
   GetApplication()->MakeWindow(ptFirstedgeDialog);
   break;

case SEDGE:
   FIR.Repaint = 0;

   if(ptSecondedgeDialog && IsWindow(ptSecondedgeDialog->HWindow)) {
      SetActiveWindow(ptSecondedgeDialog->HWindow);
   }
   else {
      ptSecondedgeDialog = new TSecondedgeDialog(this,"SEGE",testtext,TEXTBUFFER);
   }
   GetApplication()->MakeWindow(ptSecondedgeDialog);
   break;

case TEDGE:
   FIR.Repaint = 0;

   if(ptThirdedgeDialog && IsWindow(ptThirdedgeDialog->HWindow)) {
      SetActiveWindow(ptThirdedgeDialog->HWindow);
   }
   else {
      ptThirdedgeDialog = new TThirdedgeDialog(this,"TEDG",testtext,TEXTBUFFER);
   }
   GetApplication()->MakeWindow(ptThirdedgeDialog);
   break;

case FTHEDGE:
   FIR.Repaint = 0;

   if(ptFourthedgeDialog && IsWindow(ptFourthedgeDialog->HWindow)) {
      SetActiveWindow(ptFourthedgeDialog->HWindow);
   }
   else {
      ptFourthedgeDialog = new TFourthedgeDialog(this,"FEDGE",
                                                 testtext,TEXTBUFFER);
   }
   GetApplication()->MakeWindow(ptFourthedgeDialog);
   break;

case LPP:
  {
   hmenuTop = GetMenu(HWindow);
   hmenu = GetSubMenu(hmenuTop,2);
   CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
   CheckMenuItem(hmenu,0,MF_CHECKED|MF_BYPOSITION);
   if(FIR.Filtertype!=0) FIR.WindowChange = 0;
   FIR.Filtertype = 0;

   SendMessage(HWindow,WM_COMMAND,COMPI,0L);
       break;
  }

case HP:
  {
   hmenuTop = GetMenu(HWindow);
   hmenu = GetSubMenu(hmenuTop,2);
   CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
   CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
   if(FIR.Filtertype!=1) FIR.WindowChange = 0;
   FIR.Filtertype = 1;
   SendMessage(HWindow,WM_COMMAND,COMPI,0L);
       break;
  }

 case BP:
  {
   hmenuTop = GetMenu(HWindow);
   hmenu = GetSubMenu(hmenuTop,2);
   CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
   CheckMenuItem(hmenu,4,MF_CHECKED|MF_BYPOSITION);
   if(FIR.Filtertype!=2) FIR.WindowChange = 0;
   FIR.Filtertype = 2;
   SendMessage(HWindow,WM_COMMAND,COMPI,0L);
       break;
  }

 case BR:
  {
   hmenuTop = GetMenu(HWindow);
```

```
    hmenu = GetSubMenu(hmenuTop,2);
    CheckMenuItem(hmenu,2*FIR.Filtertype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,6,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Filtertype!=3) FIR.WindowChange = 0;
    FIR.Filtertype = 3;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case RCT:
   {
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,3);
    CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,0,MF_CHECKED|MF_BYPOSITION);
        if(FIR.Windowtype!=0) FIR.WindowChange = 0;
    FIR.Windowtype = 0;
    FIR.WindowChange = 1;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case KAISER:
   {
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,3);
    CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,1,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Windowtype!=1) FIR.WindowChange = 0;
    FIR.Windowtype = 1;
    FIR.WindowChange = 1;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case HANN:
   {
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,3);
    CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Windowtype!=2) FIR.WindowChange = 0;
    FIR.Windowtype = 2;
    FIR.WindowChange = 1;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case HAMM:
   {
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,3);
    CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,3,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Windowtype!=3) FIR.WindowChange = 0;
    FIR.Windowtype = 3;
    FIR.WindowChange = 1;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case BLKMAN:
   {
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,3);
    CheckMenuItem(hmenu,FIR.Windowtype,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,4,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Windowtype!=4) FIR.WindowChange = 0;
    FIR.Windowtype = 4;
    FIR.WindowChange = 1;
    SendMessage(HWindow,WM_COMMAND,COMPI,0L);
    break;
   }

  case GAIN:
   {
    HBRUSH hbr;
    RECT   r;
    HPEN   pen;
    int    cnt;
    int    x,y;
    float  w,result;
    float  Re,Im;
    char   tout[80];

    SetActiveWindow(HWindow);
    InvalidateRect(HWindow,NULL,FALSE);
    SendMessage(HWindow,WM_PAINT,0L,0L);

        hmenuTop = GetMenu(HWindow);
```

```
    hmenu = GetSubMenu(hmenuTop,4);
  CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
  CheckMenuItem(hmenu,0,MF_CHECKED|MF_BYPOSITION);
  if(FIR.Display!=0) FIR.Change = 1;
  FIR.Display = 0;

  hbr = GetStockObject(LTGRAY_BRUSH);
  SelectObject(OOHDC,hbr);
  GetClientRect(HWindow,&r);
  if(FIR.Change) Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);


  if(FIR.Canplot) {
      if(FIR.Change)  {
    if( 0x80000000 == SetBkColor(OOHDC,RGB(180,180,180))) putch(7);
    FIR.setfactors(-0.25,-0.1,3.4,1.2,HWindow);
        SetTextAlign(OOHDC,TA_RIGHT|TA_BASELINE);
    pen = GetStockObject(WHITE_PEN);
    SelectObject(OOHDC,pen);
    for(cnt=0;cnt<=11;cnt++) {
        sprintf(tout,"%.2f", 1.1*(float)cnt/11.0);
        TextOut(OOHDC,FIR.dx(-0.03),FIR.dy(-0.02 + 1.11*(float)(cnt)/11.0),
                tout,strlen(tout));
        MoveTo(OOHDC,FIR.dx(0),FIR.dy(1.1*(float)(cnt)/11.0));
        LineTo(OOHDC,FIR.dx(M_PI),FIR.dy(1.1*(float)(cnt)/11.0));
    }
    SetTextAlign(OOHDC,TA_CENTER|TA_TOP);
    for(cnt=0;cnt<=5;cnt++) {
        sprintf(tout,"%.2f",(M_PI*((float)cnt/5.0))*
                            (FIR.Sampling/(2.0*M_PI)));
        if(cnt) TextOut(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),
                FIR.dy(-0.01),tout,strlen(tout));
        MoveTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(0));
        LineTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(1.1));
    }
    TextOut(OOHDC,FIR.dx(1.57),FIR.dy(1.19),"GAIN",4);
   }

    pen = CreatePen(PS_SOLID,2, Graphplot[FIR.Windowtype]);
    SelectObject(OOHDC,pen);
    for(w=0.0;w<M_PI;w+=0.02) {
      Re=0.0;
      Im=0.0;
      for(cnt=0;cnt<FIR.LENGTH;cnt++) {
        Re += FIR.h[cnt]*cos(w*cnt);
        Im += FIR.h[cnt]*sin(w*cnt);
      }
      result = sqrt( (Re*Re)+(Im*Im) );
      x = FIR.dx(w);
      y = FIR.dy(result);
      if(w==0) MoveTo(OOHDC,x,y);
      LineTo(OOHDC,x,y);
    }
    DeleteObject(pen);
   }
   else Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);

  break;
 }

case LOGM:
 {
  HBRUSH hbr;
  RECT   r;
  HPEN   pen;
  int    cnt;
  int    x,y;
  float  w,result;
  float  Re,Im;
      char   tout[20];

  InvalidateRect(HWindow,NULL,FALSE);
  SendMessage(HWindow,WM_PAINT,0L,0L);
  hmenuTop = GetMenu(HWindow);
  hmenu = GetSubMenu(hmenuTop,4);
  CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
  CheckMenuItem(hmenu,1,MF_CHECKED|MF_BYPOSITION);
  if(FIR.Display!=1) FIR.Change = 1;
  FIR.Display = 1;

  hbr = GetStockObject(LTGRAY_BRUSH);
  SelectObject(OOHDC,hbr);
      GetClientRect(HWindow,&r);
  if(FIR.Change) Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);

  if(FIR.Canplot) {
   if(FIR.Change)  {
    if( 0x80000000 == SetBkColor(OOHDC,RGB(180,180,180))) putch(7);
    FIR.setfactors(-0.35,-225,3.4,55,HWindow);
```

```
        SetTextAlign(OOHDC,TA_RIGHT|TA_BASELINE);
      pen = GetStockObject(WHITE_PEN);
      SelectObject(OOHDC,pen);
      for(cnt=-1;cnt<=11;cnt++) {
          sprintf(tout,"%.1f", -240*(float)cnt/12.0);
          TextOut(OOHDC,FIR.dx(-0.03),FIR.dy(-0.02 - 240*(float)(cnt)/12.0),
                  tout,strlen(tout));
          MoveTo(OOHDC,FIR.dx(0),FIR.dy(-240*(float)(cnt)/12.0));
          LineTo(OOHDC,FIR.dx(M_PI),FIR.dy(-240*(float)(cnt)/12.0));
      }
      SetTextAlign(OOHDC,TA_CENTER|TA_BOTTOM);
      for(cnt=0;cnt<=5;cnt++) {
          sprintf(tout,"%.2f",(M_PI*((float)cnt/5.0))*
                              (FIR.Sampling/(2.0*M_PI)));
          if(cnt) TextOut(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),
                  FIR.dy(23.5),tout,strlen(tout));
          MoveTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(20));
          LineTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(-240.0));
      }
      SetTextAlign(OOHDC,TA_CENTER|TA_TOP);
      TextOut(OOHDC,FIR.dx(1.57),FIR.dy(52),"LOG MAGNITUDE",13);
         }


      pen = CreatePen(PS_SOLID,2,Graphplot[FIR.Windowtype]);
      SelectObject(OOHDC,pen);
      for(w=0.0;w<M_PI;w+=0.01) {
        Re=0.0;
        Im=0.0;
        for(cnt=0;cnt<FIR.LENGTH;cnt++) {
          Re += FIR.h[cnt]*cos(w*cnt);
          Im += FIR.h[cnt]*sin(w*cnt);
        }
        result = sqrt( (Re*Re) + (Im*Im) );
        result = 20 * log(result);
        x = FIR.dx(w);
        y = FIR.dy(result);
        if(w==0) MoveTo(OOHDC,x,y);
        LineTo(OOHDC,x,y);
      }

      DeleteObject(pen);
    }
    else Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);
    break;
   }

  case PHASE:
   {
    HBRUSH hbr;
    RECT   r;
    HPEN   pen;
    int    cnt;
    int    x,y;
    float  w,result;
    float  Re,Im;
    char   tout[20];

    InvalidateRect(HWindow,NULL,FALSE);
    SendMessage(HWindow,WM_PAINT,0L,0L);
    hmenuTop = GetMenu(HWindow);
    hmenu = GetSubMenu(hmenuTop,4);
    CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
    CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
    if(FIR.Display!=2) FIR.Change = 1;
    FIR.Display = 2;

    hbr = GetStockObject(LTGRAY_BRUSH);
    SelectObject(OOHDC,hbr);
        GetClientRect(HWindow,&r);
    if(FIR.Change) Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);

    if(FIR.Canplot) {
      FIR.setfactors(-0.33,-1*M_PI-0.8,3.5,M_PI+1.0,HWindow);
      if( 0x80000000 == SetBkColor(OOHDC,RGB(180,180,180))) putch(7);
      SetTextAlign(OOHDC,TA_RIGHT|TA_BASELINE);
      pen = GetStockObject(WHITE_PEN);
      SelectObject(OOHDC,pen);
      for(cnt=-10;cnt<=10;cnt+=2) {
                sprintf(tout,"%.2f", M_PI*(float)cnt/10.0);
              TextOut(OOHDC,FIR.dx(-0.03),FIR.dy(-0.09 + M_PI*(float)(cnt)/10.0),
                           tout,strlen(tout));
                MoveTo(OOHDC,FIR.dx(0),FIR.dy(M_PI*(float)(cnt)/10.0));
              LineTo(OOHDC,FIR.dx(M_PI),
                    FIR.dy(M_PI*(float)(cnt)/10.0));
      }
      SetTextAlign(OOHDC,TA_CENTER|TA_TOP);
      for(cnt=0;cnt<=5;cnt++) {
              sprintf(tout,"%.2f",
                           (M_PI*((float)cnt/5.0))*(FIR.Sampling/(2.0*M_PI)));
```

```
            if(cnt) TextOut(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),
                  FIR.dy(-1*M_PI - 0.1),tout,strlen(tout));
            MoveTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(-1*M_PI));
            LineTo(OOHDC,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(M_PI));
        }
        TextOut(OOHDC,FIR.dx(M_PI/2.0),FIR.dy(M_PI+0.85),"PHASE RESPOND",13);


        pen = CreatePen(PS_SOLID,1,Graphplot[FIR.Windowtype]);
        SelectObject(OOHDC,pen);
        for(w=0.0;w<M_PI;w+=0.03) {
          Re=0.0;
          Im=0.0;
          for(cnt=0;cnt<FIR.LENGTH;cnt++) {
            Re += FIR.h[cnt]*cos(w*cnt);
              Im += FIR.h[cnt]*sin(w*cnt);
                    }
          result = sqrt(Re*Re+Im*Im);
          result = atan2(-1*Im,Re);
          x = FIR.dx(w);
          y = FIR.dy(result);
          if(w==0) MoveTo(OOHDC,x,y);
          LineTo(OOHDC,x,y);
        }

        DeleteObject(pen);
      }
      else Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);
      break;
     }

    case IMPULSE:
     {
      HBRUSH hbr;
      RECT   r;
      HPEN   pen;
      int    cnt;
      char   tout[20];

      InvalidateRect(HWindow,NULL,FALSE);
      SendMessage(HWindow,WM_PAINT,0L,0L);

      hmenuTop = GetMenu(HWindow);
      hmenu = GetSubMenu(hmenuTop,4);
      CheckMenuItem(hmenu,FIR.Display,MF_UNCHECKED|MF_BYPOSITION);
      CheckMenuItem(hmenu,3,MF_CHECKED|MF_BYPOSITION);
      if(FIR.Display!=3) FIR.Change = 1;
      FIR.Display = 3;

      if( 0x80000000 == SetBkColor(OOHDC,RGB(180,180,180))) putch(7);
      hbr = GetStockObject(LTGRAY_BRUSH);
      SelectObject(OOHDC,hbr);
          GetClientRect(HWindow,&r);
      if(FIR.Change) Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);
      if(FIR.Canplot) {

        FIR.setfactors(-1*( (float)FIR.LENGTH/10.0 ),
                      -1.1,(float)FIR.LENGTH+( (float)FIR.LENGTH/10.0 ),
                      1.2,HWindow);
        SetTextAlign(OOHDC,TA_RIGHT|TA_BASELINE);
        pen = GetStockObject(WHITE_PEN);
                    SelectObject(OOHDC,pen);
        for(cnt=-10;cnt<=10;cnt+=2) {
                sprintf(tout,"%.2f", 1.0*(float)cnt/10.0);
                TextOut(OOHDC,FIR.dx(-0.07),FIR.dy(-0.02 + 1.0*(float)(cnt)/10.0),
                                              tout,strlen(tout));
           MoveTo(OOHDC,FIR.dx(0),FIR.dy(1.0*(float)(cnt)/10.0));
                LineTo(OOHDC,FIR.dx((float)FIR.LENGTH),
                      FIR.dy(1.0*(float)(cnt)/10.0));
        }
        MoveTo(OOHDC,FIR.dx(0),FIR.dy(1.0));
        LineTo(OOHDC,FIR.dx(0),FIR.dy(-1.0));
        MoveTo(OOHDC,FIR.dx((float)FIR.LENGTH),FIR.dy(1.0));
        LineTo(OOHDC,FIR.dx((float)FIR.LENGTH),FIR.dy(-1.0));
        SetTextAlign(OOHDC,TA_CENTER|TA_TOP);
                    TextOut(OOHDC,FIR.dx( (float)FIR.LENGTH/2.0),
                                                  FIR.dy(1.15),"IMPUSE RESPOND",14);


        pen = CreatePen(PS_SOLID,2,Graphplot[FIR.Windowtype] /*RGB(255,0,255)*/);
        SelectObject(OOHDC,pen);
            MoveTo(OOHDC,FIR.dx(0),FIR.dy(FIR.h[0]));
        for(cnt=0;cnt<FIR.LENGTH;cnt++){
          MoveTo(OOHDC,FIR.dx((float)cnt),FIR.dy(0));
          LineTo(OOHDC,FIR.dx((float)cnt),FIR.dy(FIR.h[cnt]));
        }
        MoveTo(OOHDC,FIR.dx((float)cnt),FIR.dy(1.0));
      }
      else Rectangle(OOHDC,r.left,r.top,r.right,r.bottom);
      break;
```

```
 }
 case INFOM:
  {
     char informations[500];
     char type[4][20] = {"LOW PASS:","HIGH PASS:",
                         "BAND PASS:","BAND REJECT:"};
     char typewin[5][20] = {"Rectangular","Kaiser","Hann","Hamming",
                            "Blackman"};

     sprintf(informations,"Filter type of %s \n"
                             "Weigthed by %s window.\n\n"
                          "The Sampling rate is\t%.3f\tHz.\n"
                          "Pass band ripple  is\t%.3f        \tdB.\n"
                          "Stop  band ripple  is\t%.3f      \tdB.\n"
                          "First  edge frequency is\t%.3f\tHz.\n"
                          "Second edge frequency is\t%.3f\tHz.\n"
                          "Third  edge frequency is\t%.3f\tHz.\n"
                          "Forth  edge frequency is\t%.3f\tHz.\n"
                          "Length is %u.\n",
                          type[FIR.Filtertype],
                          typewin[FIR.Windowtype],
                          FIR.Sampling,
                          FIR.Pbripple,
                          FIR.Stopbatt,
                          FIR.Firstedge,
                          FIR.Secondedge,
                          FIR.Thirdedge,
                          FIR.Fourthedge,
                          FIR.LENGTH);
   wsprintf(informations,"%s",informations);
     MessageBox(Msg.Receiver,informations,"FILTER'S INFORMATION",MB_OK);
     break;
  }

case COMPI:
  {
   float temp1=0,temp2=0,temp3=0;
   float center=0,bt=0;
       char  nw[20];

   if(FIR.first) {
       FIR.first = 0;
       FIR.Canplot = 0;
       wsprintf(nw,"Welcome to new filter...");
       MessageBox(HWindow,nw,"NEW",MB_OK);
       FIR.Change  =  1;
       SendMessage(HWindow,WM_COMMAND,GAIN+(int)FIR.Display,0L);
       break;
   }

    if(FIR.WindowChange) {
        FIR.WindowChange = 0;
        FIR.Change       = 0;
    }
    else FIR.Change = 1;
        if( (FIR.Filtertype==2)||(FIR.Filtertype==3) ){
      temp1 = (FIR.Firstedge+FIR.Secondedge)/2.0;
          temp2 = (FIR.Thirdedge+FIR.Fourthedge)/2.0;
          center = (temp1+temp2)/2.0;
          if((FIR.Secondedge-FIR.Firstedge)<(FIR.Fourthedge-FIR.Thirdedge)) {
        temp3 =  center - temp1;
            bt = (FIR.Secondedge-FIR.Firstedge)/2.0;
          }
          else {
            temp3 = temp2-center;
            bt = (FIR.Fourthedge-FIR.Thirdedge)/2.0;
      }

      FIR.KaiserOptimize(FIR.Filtertype,center,FIR.Stopbatt,FIR.Pbripple,
                    temp3-bt,temp3+bt,FIR.Sampling,
                    FIR.h,FIR.hh,HWindow,FIR.Windowtype);
    }
    else {
      FIR.KaiserOptimize(FIR.Filtertype,0.0,FIR.Stopbatt,FIR.Pbripple,
                    FIR.Firstedge,FIR.Secondedge,FIR.Sampling,
                    FIR.h,FIR.hh,HWindow,FIR.Windowtype);
  }

       SendMessage(HWindow,WM_COMMAND,GAIN+(int)FIR.Display,0L);

   }
 break;
 case SENDOP:
  {
        char buff[80];
        if(FIR.Success) {
          disable();
          FIR.MakeFilter(FIR.LENGTH,FIR.TIMEBASE,HWindow);
          enable();
```

```
            wsprintf(buff,"%s","Send assembly language of current filter complete...");
          MessageBox(HWindow,buff,"MESSAGE",MB_OK);
        }
        else {
          wsprintf(buff,"%s","Can not send to DSP board because length of "
                             "filter over capability of hardware.");
          MessageBox(HWindow,buff,"MESSAGE",MB_OK);
           }
      }
         break;

    case OPTIMUM:

         hmenuTop = GetMenu(HWindow);
         hmenu = GetSubMenu(hmenuTop,5);
             if(FIR.Optimize) {
           CheckMenuItem(hmenu,2,MF_UNCHECKED|MF_BYPOSITION);
           FIR.Optimize = 0;
             }
         else {
           CheckMenuItem(hmenu,2,MF_CHECKED|MF_BYPOSITION);
           FIR.Optimize = 1;
         }
         FIR.WindowChange = 0;
         SendMessage(HWindow,WM_COMMAND,COMPI,0L);
         break;
    case PRINT:
     {
                    FIR.Printer = 1;
         if(FIR.PrintMyPage(HWindow)) MessageBox(HWindow,"Could not print",
                                  "MESSAGE",MB_OK|MB_ICONEXCLAMATION);
         FIR.Printer = 0;
         FIR.WindowChange = 0;
         SendMessage(HWindow,WM_COMMAND,COMPI,0L);
             break;

     }

    case SAVE:
     {
      FILE *fp;
      char title[80];

       if(strcmp(FIR.Savename,"NONAME.FIR")) {
         if( (fp = fopen(FIR.Savename,"wb")) != NULL ) {
           FIR.savespec.type = FIR.Filtertype;
           FIR.savespec.sampling = FIR.Sampling;
           FIR.savespec.pbripple = FIR.Pbripple;
           FIR.savespec.stopbatt = FIR.Stopbatt;
           FIR.savespec.firstedge= FIR.Firstedge;
           FIR.savespec.secondedge=FIR.Secondedge;
           FIR.savespec.thirdedge= FIR.Thirdedge;
           FIR.savespec.fourthedge=FIR.Fourthedge;
           FIR.savespec.windowtype=FIR.Windowtype;
           FIR.savespec.display  = FIR.Display;
           FIR.savespec.optimize = FIR.Optimize;

           fwrite(&FIR.savespec,sizeof(FIR.savespec),1,fp);
           fclose(fp);
           sprintf(title,"FIR FILTER - ( %s )",FIR.Savename);
           SetCaption(title);
             }
         break;
       }
       else {
         SendMessage(HWindow,WM_COMMAND,SAVEAS,0L);
         break;
           }
     }

        default:
           wsprintf(buffer, "Command = %d",COMMANDMSG(Msg));
       MessageBox(HWindow,buffer,"WM_COMMAND",MB_OK);
     }
  }
}


void DrawMarker(HDC hdc,int x,int y)
{
   int i;
   int rop;

   for(i=0;i<=MARKERSIZE;i++) {
    rop = SetROP2(hdc, R2_COPYPEN);
    SetPixel(hdc,x+i,y,RGB(200,0,0));
    SetPixel(hdc,x-i,y,RGB(0,200,0));
    SetPixel(hdc,x,y+i,RGB(0,0,200));
    SetPixel(hdc,x,y-i,RGB(200,0,200));
   }
```

```
   SetROP2(hdc,rop);
}

void Filter::setfactors(float lefta1,float bottomb1,
                        float righta2,float upperb2,HWND hwnd)
{
   RECT r;
   short cx,cy;

   GetClientRect(hwnd,&r);
   if(!FIR.Printer) {
     X1 = lefta1;
     X2 = righta2;
     Y1 = bottomb1;
     Y2 = upperb2;
     PY = (int)(r.bottom - r.top)/1;
     XF = (float) ((float)(r.right - r.left))/(float)(righta2 - lefta1);
     YF = (float) ((float)(r.bottom - r.top))/(float)(upperb2 - bottomb1);
   }
     else {
     X1 = lefta1;
     X2 = righta2;
     Y1 = bottomb1;
     Y2 = upperb2;
     cx = GetDeviceCaps(PRHDC,HORZRES);
     cy = (GetDeviceCaps(PRHDC,VERTRES))/2;

     PY = (int)(cy)/1;
     XF = (float) ((float)(cx))/(float)(righta2 - lefta1);
     YF = (float) ((float)(cy))/(float)(upperb2 - bottomb1);
   }
}

int Filter::dx(float VX)
{
     return ( (int) (XF*(VX-X1)));       /* return x coordinated */
}

int Filter::dy(float VY)
{
    return ( (int) ((PY-( (YF*VY) - (YF*Y1) )))/1 );
}

void Filter::coordinate(int startx,int starty,int endx,int endy)
{
   char NUM[5];
   int CNTR;

   for(CNTR=startx;CNTR<=endx;CNTR++) {
       MoveTo(OOHDC,dx(CNTR),dy(0));
       LineTo(OOHDC,dx(CNTR),(dy(0)+3));
   }
   for(CNTR=starty;CNTR<=endy;CNTR++) {
       MoveTo(OOHDC,dx(0)-3,dy(CNTR));
       LineTo(OOHDC,dx(0)+3,dy(CNTR));
   }
   MoveTo(OOHDC,dx(X1),dy(0));
   LineTo(OOHDC,dx(X2),dy(0));
   MoveTo(OOHDC,dx(0),dy(Y1));
   LineTo(OOHDC,dx(0),dy(Y2));
}

int Filter::KaiserOptimize(int type,float center,float Aa,float Ap,float Fp,
                float Fa,float sampling,float *h,int *hh,
                HWND hwnd,int wintype)
{
    int cnt;
    int linear;
    unsigned int num,timeb,N;
    float Io;
    float tmp;
    float thethaa,thethap,Wc,Ws;
    float temp;
    float alpha,beta;
    float D;
    char  status[200];

            for(cnt=0;cnt<3000;cnt++) {
       *(h+cnt) = 0;
    }

    sampling = 1000000.0/sampling;
    timeb = (int)(sampling/1);
    if(timeb>255) {
      timeb = 255;
    }
    sampling = 1000000/(float)timeb;
    FIR.Sampling = sampling;

    if( (FIR.Filtertype<2)&&((FIR.Firstedge>=FIR.Secondedge)||
```

```
  (FIR.Firstedge>=FIR.Sampling/2.0)||
  (FIR.Secondedge>=FIR.Sampling/2.0)) )
  {
   wsprintf(status,"%s","Specification of filter error...");
       FIR.Canplot = 0;
   MessageBox(hwnd,status,"Error",MB_OK);
   return(0);
  }

  if( (FIR.Filtertype>1)&&( (FIR.Firstedge>=FIR.Secondedge)||
  (FIR.Secondedge>=FIR.Thirdedge)||
  (FIR.Thirdedge>=FIR.Fourthedge)||
  (FIR.Fourthedge>=FIR.Sampling/2.0) ) )
  {
   wsprintf(status,"%s","Specification of filter error...");
       FIR.Canplot = 0;
   MessageBox(hwnd,status,"Error",MB_OK);
   return(0);
  }

Fa = 2*Fa/sampling;
Fp = 2*Fp/sampling;
Ws = 2.0;
center = 2*center/sampling;
Wc = 0.5*(Fa+Fp)*M_PI;
if( (type == 1) ) Wc = M_PI-Wc;
temp = pow(10,0.05*Ap);
thethap =  (temp-1)/(temp+1);
thethaa =  pow(10,-1*0.05*Aa);
if(thethap<thethaa) thethaa = thethap;
Aa = -1*20*log(thethaa);
if(Aa<21.0) {
  alpha = 0.0;
  D = 0.9222;
}
else  {
  D = (Aa-7.95)/14.36;
  if( (Aa>21.0)&&(Aa<50) ) {
        alpha = (0.5842*pow(Aa-20.0,0.4)) + 0.07886*(Aa-21);
  }
  else alpha = 0.1102*(Aa-8.7);
}
Ws = (Ws*D/(Fa-Fp)) + 1;
num = (int)Ws/1;
if(num%2 == 0) num++;
N = num;

if(FIR.Optimize) {
  /* extend to the maximum capability */
  while(  (1.0/sampling) > ((23+(2*num))*200.0/1000000000.0) && (num<81) ) {
   num+=2;
  }
  /* reduce to the maximum capability of hardware */
  if( (1.0/sampling) < ((23+(2*num))*200.0/1000000000.0) ) num-=4;
}



N = num;
if(N>=2000) N = num = 2000;

        linear  = (num-1)/2;
h[linear] = Wc/M_PI;
for(cnt=1;cnt<linear;cnt++) {
    h[cnt] = h[num-cnt-1] = sin(Wc*(num-cnt-linear-1))/((num-cnt-linear-1)*M_PI);
}

// Kaiser window
if(wintype==1) {
  Io = FIR.I(alpha);

  for(cnt=0;cnt<num;cnt++) {
        tmp = (((float)2*(cnt-linear)/(float)(num-1)));
   tmp *= tmp;
      //if(tmp>1.0) tmp = 0.9999999999999999999;
        tmp = FIR.I(alpha*sqrt( fabs(1.0-tmp) ));
                        h[cnt] = h[cnt] * tmp / Io;
  }
}

// Hann window
if(wintype==2) {
  for(cnt=0;cnt<num;cnt++) {
      h[cnt] = h[cnt] * (0.5 + (0.5*cos(2*M_PI*(cnt-linear)/num)));
  }
}

// Hamming window
if(wintype==3) {
  for(cnt=0;cnt<num;cnt++) {
```

```
            h[cnt] = h[cnt] * (0.54 + (0.46*cos(2*M_PI*(cnt-linear)/num)));
          }
        }

        if(wintype==4) {
          for(cnt=0;cnt<num;cnt++) {
            h[cnt] = h[cnt] * ( (0.42 + (0.5*cos(2*M_PI*(cnt-linear)/num))) +
                               (0.08*cos(4*M_PI*cnt/num)));
          }
        }

        if(type==1) {
          for(cnt=0;cnt<num;cnt++) {
             if(cnt%2 != 0) h[cnt] = -1*h[cnt];
          }
        }

        if(type==2) {
          for(cnt=0;cnt<num;cnt++) {
             h[cnt] = 2*h[cnt]*cos((cnt-linear)*M_PI*center);
          }
        }

        if(type==3) {
          for(cnt=0;cnt<num;cnt++) {
                h[cnt] = 2*h[cnt]*cos((cnt-linear)*M_PI*center);
          }
          temp = h[linear];
          for(cnt=0;cnt<num;cnt++) {
                h[cnt] = -1 * h[cnt];
          }
          h[linear] = 1-temp;
        }

        FIR.LENGTH = N;
        if( (N<=81)&&((1.0/sampling) >= (23+(2*num))*200.0/1000000000.0) ) {
          Success = 1;
          FIR.TIMEBASE = timeb;

          FIR.quantize(h,N,hh);
          FIR.converse(h,hh,N,hwnd);
        }
                else Success = 0;
        if(!Success) {
         sprintf(status,"Filter's length of %d is over capability of hardware\n",N);
         wsprintf(status,"%s",status);
         MessageBox(hwnd,status,"MESSAGE",MB_OK);
        }
        FIR.Canplot = 1;
        return(N);
}

void Filter::quantize(float *h,int num,int *hh)
{
   int cnt,cnt1;
   float temp;
   int flag=0;
   unsigned int quantized=0;

   for(cnt=0;cnt<num;cnt++){
     flag=0;
     temp = h[cnt];
     if(temp<0) {
            flag = 1;
                           temp = 0-temp;
     }

     for(cnt1=0;cnt1<15;cnt1++){
       temp *= 2;
       if(temp>=1) {
              quantized |= 0x01<<(14-cnt1);
           temp -= 1.0;
       }
     }
     if(flag) {
       quantized ^= 0xffff;
       flag=0;
     }
     /*printf("%x ",quantized);*/
     hh[cnt]=quantized;
     quantized=0.0;
   }
}

void Filter::converse(float *h,int *hh1,int num,HWND hwnd)
{
  int cnt,cnt1;
  int flag=0;
    float temp=0;
    int *hh;
```

```
    char buff[80];

    if( (hh=(int *)malloc(5*num*sizeof(int))) == NULL ) {
            putch(7);
            wsprintf(buff,"%s","not enough memory...");
            GetWindow(hwnd,GW_HWNDFIRST);
            MessageBox(hwnd,buff,"ERROR",MB_OK);
            SendMessage(hwnd,WM_SYSCOMMAND,SC_CLOSE,0L);
       }
  for(cnt=0;cnt<num;cnt++) *(hh+cnt) = *(hh1+cnt);
  for(cnt=0;cnt<num;cnt++) {
    temp = 0;
    if(hh[cnt]<0) {
            flag=1;
            hh[cnt] = 0-hh[cnt];
    }
    for(cnt1=0;cnt1<15;cnt1++) {
       if(hh[cnt]&(0x01<<cnt1)) {
           temp += (float)1/(0x01<<(15-cnt1));
       }
    }
    if(flag) {
      temp = 0-temp;
      flag=0;
            }
    h[cnt] = temp;
  }
}


void Filter::MakeFilter(int num,int timebase,HWND hwnd)
{
   char buff[80];
   int cnt,count;
   int *stream;
   int offx,off,offh;
   int prog[50] = {
     0x707F, 0x6880, 0x6E00, 0x7F89, 0x5088, 0xF400, 0x0000, 0x7E01,
     0x507F, 0x7E80, 0x507C, 0x7E02, 0x6880, 0x0000, 0x0000, 0x6881,
     0x67A0, 0x007F, 0xF400, 0x0000, 0x0000, 0x677E, 0x4F7E, 0x6880,
     0x4000, 0x2000, 0x107C, 0x5000, 0x7F89, 0x0000, 0x0000
   };

   disable();
   outportb(CW1,0x80);       /* initialize all ports */
   outportb(CW2,0x80);
   outportb(PC2,0x07);

     if( (stream=(int *)malloc(5*num*sizeof(int))) == NULL ) {
                 putch(7);
      wsprintf(buff,"%s","not enough memory...");
      GetWindow(hwnd,GW_HWNDFIRST);
      MessageBox(hwnd,buff,"ERROR",MB_OK);
                 SendMessage(hwnd,WM_SYSCOMMAND,SC_CLOSE,0L);
   }

   *(stream) = 0xf900;
   *(stream+1) = num+3;

   for(cnt=0;cnt<= num ;cnt++) {
            *(stream+2+cnt) = FIR.hh[cnt];
   }

   *(stream+num+2) = timebase;     /* Time base */

   prog[6] = num+7;
   prog[13] = 0x7000 | (num-1)/2;
   prog[14] = 0x7100 | (num+1);
   prog[19] = num+18;
   prog[20] = 0x7E00 | (num+2);
   prog[29] = 0x6A00|(num-1);
   prog[30] = 0x6D00|(num+1);
   for(cnt=0;cnt<31;cnt++) {
     *(stream+cnt+num+3) = prog[cnt];
   }

   offx=num-2;
   off =34;
   offh=num+2;
   for(cnt=1;cnt<= (num-1)/2; cnt++) {
     *(stream+num+off) = 0x6B00|offx;
     off++;  offx--;
     *(stream+num+off) = 0x6D00|offh;
     off++;  offh++;
   }
   offh-=2;
   for(cnt=0;cnt<(num-1)/2;cnt++) {
     *(stream+num+off) = 0x6B00|offx;
     off++; offx--;
     *(stream+num+off) = 0x6D00|offh;
```

```
     off++; offh--;
    }
    *(stream+num+off) = 0x7F8F; off++;
    *(stream+num+off) = 0x0F7C; off++;
    *(stream+num+off) = 0x597D; off++;
    *(stream+num+off) = 0x487D; off++;
    *(stream+num+off) = 0xF600; off++;
    *(stream+num+off) = (num+off-1); off++;
    *(stream+num+off) = 0xF900; off++;
    *(stream+num+off) = num+27;
    count=num+off;

    for(cnt=0;cnt <= count;cnt++){
        outportb(PB2,(unsigned char)(cnt>>8));
        outportb(PA2,(unsigned char)(cnt&0xff));
        outportb(PB1,(unsigned char)( *(stream+cnt)>>8 ));
        outportb(PA1,(unsigned char)( *(stream+cnt)&0xff ));
        outportb(PC2,0x05);
        outportb(PC2,0x04);
        outportb(PC2,0x04);
        outportb(PC2,0x07);
    }
    free(stream);
    outportb(CW1,0x80);
      outportb(PC2,0x03);
      enable();
}

float Filter::I(float x)
{
    float temp;
    int   m=1;
    float result = 1.0;

    temp = x/2.0;
    while((temp>=0.0000000000001)&&(m<16)) {
       result += temp;
       m++;
       temp =    (pow(x/2.0,m)/(float)FIR.factorial(m));
       temp *= temp;
    }
    return(result);
}

unsigned long Filter::factorial(int x)
{
     unsigned long result=1;
     int cnt;

     if(x==0) return(1);
     for(cnt=1;cnt<=x;cnt++) {
       result  *= cnt;
     }
     return(result);
}

HDC Filter::GetPrinterDC(void)
{
     static char szPrinter[80];
     char   *szDevice, *szDriver, *szOutput;

     GetProfileString("windows","device",",,,",szPrinter,80);

     if( NULL != (szDevice = strtok(szPrinter,","))&&
     NULL != (szDriver = strtok(NULL,","))&&
     NULL != (szOutput = strtok(NULL,",")))

          return CreateDC(szDriver,szDevice,szOutput,NULL);

     return 0;
}

void Filter::PageGDICalls(HDC hdcPrn,short cxPage,short cyPage,HWND hwnd)
{

      int    cnt;
      char   tout[80];
      float  Re=0,Im=0,w=0,result=0;
      int    x,y;


      Rectangle(hdcPrn,0,0,cxPage,cyPage/2);
      if(FIR.Display == 1) {
       FIR.setfactors(-0.35,-225,3.4,55,hwnd);
       SetTextAlign(hdcPrn,TA_RIGHT|TA_BASELINE);
       for(cnt=-1;cnt<=11;cnt++) {
          sprintf(tout,"%.1f", -240*(float)cnt/12.0);
             TextOut(hdcPrn,FIR.dx(-0.03),FIR.dy(-0.02 - 240*(float)(cnt)/12.0),
                          tout,strlen(tout));
          MoveTo(hdcPrn,FIR.dx(0),FIR.dy(-240*(float)(cnt)/12.0));
```

```
        LineTo(hdcPrn,FIR.dx(M_PI),FIR.dy(-240*(float)(cnt)/12.0));
    }
    SetTextAlign(hdcPrn,TA_CENTER|TA_BOTTOM);
    for(cnt=0;cnt<=5;cnt++) {
     sprintf(tout,"%.2f",(M_PI*((float)cnt/5.0))*(FIR.Sampling/(2.0*M_PI)));
                  if(cnt) TextOut(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),
                                  FIR.dy(23.5),tout,strlen(tout));
       MoveTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(20));
       LineTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(-220.0));
    }
    SetTextAlign(hdcPrn,TA_CENTER|TA_TOP);
    TextOut(hdcPrn,FIR.dx(1.57),FIR.dy(52),"LOG MAGNITUDE",13);
    for(w=0.0;w<M_PI;w+=0.01) {
         Re=0.0;
         Im=0.0;
         for(cnt=0;cnt<FIR.LENGTH;cnt++) {
           Re += FIR.h[cnt]*cos(w*cnt);
           Im += FIR.h[cnt]*sin(w*cnt);
         }
         result = sqrt( (Re*Re) + (Im*Im) );
         result = 20 * log(result);
         x = FIR.dx(w);
         y = FIR.dy(result);
         if(w==0) MoveTo(PRHDC,x,y);
         LineTo(PRHDC,x,y);
    }

    SaveDC(hdcPrn);
    RestoreDC(hdcPrn,-1);
   }

   if(FIR.Display == 0)  {
       FIR.setfactors(-0.25,-0.1,3.4,1.2,hwnd);
     SetTextAlign(hdcPrn,TA_RIGHT|TA_BASELINE);
       for(cnt=0;cnt<=11;cnt++) {
                sprintf(tout,"%.2f", 1.1*(float)cnt/11.0);
                TextOut(hdcPrn,FIR.dx(-0.03),FIR.dy(-0.02 + 1.11*(float)(cnt)/11.0),
                                tout,strlen(tout));
           MoveTo(hdcPrn,FIR.dx(0),FIR.dy(1.1*(float)(cnt)/11.0));
                                LineTo(hdcPrn,FIR.dx(M_PI),FIR.dy(1.1*(float)(cnt)/11.0));
       }
       SetTextAlign(hdcPrn,TA_CENTER|TA_TOP);
       for(cnt=0;cnt<=5;cnt++) {
                sprintf(tout,"%.2f",(M_PI*((float)cnt/5.0))*
                                (FIR.Sampling/(2.0*M_PI)));
           if(cnt) TextOut(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),
                        FIR.dy(-0.01),tout,strlen(tout));
           MoveTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(0));
                                LineTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(1.1));
       }
       TextOut(hdcPrn,FIR.dx(1.57),FIR.dy(1.19),"GAIN",4);

       for(w=0.0;w<M_PI;w+=0.01) {
         Re=0.0;
         Im=0.0;
         for(cnt=0;cnt<FIR.LENGTH;cnt++) {
           Re += FIR.h[cnt]*cos(w*cnt);
                  Im += FIR.h[cnt]*sin(w*cnt);
         }
         result = sqrt( (Re*Re)+(Im*Im) );
         x = FIR.dx(w);
         y = FIR.dy(result);
         if(w==0) MoveTo(hdcPrn,x,y);
         LineTo(hdcPrn,x,y);
       }
       SaveDC(hdcPrn);
       RestoreDC(hdcPrn,-1);
                }

    if(FIR.Display == 2) {
       FIR.setfactors(-0.33,-1*M_PI-0.8,3.5,M_PI+1.0,hwnd);
       SetTextAlign(hdcPrn,TA_RIGHT|TA_BASELINE);
       for(cnt=-10;cnt<=10;cnt+=2) {
                sprintf(tout,"%.2f", M_PI*(float)cnt/10.0);
               TextOut(hdcPrn,FIR.dx(-0.03),FIR.dy(-0.09 + M_PI*(float)(cnt)/10.0),
                                tout,strlen(tout));
           MoveTo(hdcPrn,FIR.dx(0),FIR.dy(M_PI*(float)(cnt)/10.0));
           LineTo(hdcPrn,FIR.dx(M_PI),
                                FIR.dy(M_PI*(float)(cnt)/10.0));
       }
       SetTextAlign(hdcPrn,TA_CENTER|TA_TOP);
       for(cnt=0;cnt<=5;cnt++) {
                 sprintf(tout,"%.2f",(M_PI*((float)cnt/5.0))*(FIR.Sampling/(2.0*M_PI)));
                if(cnt) TextOut(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),
                                  FIR.dy(-1*M_PI - 0.1),tout,strlen(tout));
                  MoveTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(-1*M_PI));
                                LineTo(hdcPrn,FIR.dx(M_PI*(float)cnt/5.0),FIR.dy(M_PI));
       }
       TextOut(hdcPrn,FIR.dx(M_PI/2.0),FIR.dy(M_PI+0.85),"PHASE RESPOND",13);
```

```
          for(w=0.0;w<M_PI;w+=0.03) {
            Re=0.0;
            Im=0.0;
            for(cnt=0;cnt<FIR.LENGTH;cnt++) {
              Re += FIR.h[cnt]*cos(w*cnt);
                Im += FIR.h[cnt]*sin(w*cnt);
            }
              result = sqrt(Re*Re+Im*Im);
              result = atan2(-1*Im,Re);
              x = FIR.dx(w);
              y = FIR.dy(result);
              if(w==0) MoveTo(hdcPrn,x,y);
              LineTo(hdcPrn,x,y);
            }
            SaveDC(hdcPrn);
            RestoreDC(hdcPrn,-1);
          }
          if(FIR.Display == 3) {

            FIR.setfactors(-1*( (float)FIR.LENGTH/10.0 ),
                           -1.1,(float)FIR.LENGTH+( (float)FIR.LENGTH/10.0 ),
                           1.2,hwnd);
            SetTextAlign(hdcPrn,TA_RIGHT|TA_BASELINE);
                       for(cnt=-10;cnt<=10;cnt+=2) {
                       sprintf(tout,"%.2f", 1.0*(float)cnt/10.0);
                     TextOut(hdcPrn,FIR.dx(-0.03),FIR.dy(-0.02 + 1.0*(float)(cnt)/10.0),
                                 tout,strlen(tout));
                     MoveTo(hdcPrn,FIR.dx(0),FIR.dy(1.0*(float)(cnt)/10.0));
                       LineTo(hdcPrn,FIR.dx((float)FIR.LENGTH),
                                     FIR.dy(1.0*(float)(cnt)/10.0));
            }
            MoveTo(hdcPrn,FIR.dx((float)FIR.LENGTH),FIR.dy(1.0));
            LineTo(hdcPrn,FIR.dx((float)FIR.LENGTH),FIR.dy(-1.0));
            MoveTo(hdcPrn,FIR.dx(0),FIR.dy(1.0));
            LineTo(hdcPrn,FIR.dx(0),FIR.dy(-1.0));
            SetTextAlign(hdcPrn,TA_CENTER|TA_TOP);
                       TextOut(hdcPrn,FIR.dx( (float)FIR.LENGTH/2.0),
                                                FIR.dy(1.15),"IMPUSE RESPOND",14);


            MoveTo(hdcPrn,FIR.dx(0),FIR.dy(FIR.h[0]));
            for(cnt=0;cnt<FIR.LENGTH;cnt++){
              MoveTo(hdcPrn,FIR.dx((float)cnt),FIR.dy(0));
              LineTo(hdcPrn,FIR.dx((float)cnt),FIR.dy(FIR.h[cnt]));
            }
            MoveTo(hdcPrn,FIR.dx((float)cnt),FIR.dy(1.0));
          }

}

BOOL Filter::PrintMyPage(HWND hwnd)
{
     static char szMessage [] = "Print1: Printing";
     BOOL   bError = FALSE;
     HDC    hdcPrn;
     short  xPage,yPage;

     if(NULL == (hdcPrn = GetPrinterDC())) {
       putch(7);
       return TRUE;
     }
     PRHDC = hdcPrn;
     xPage = GetDeviceCaps(hdcPrn,HORZRES);
     yPage = GetDeviceCaps(hdcPrn,VERTRES);

     if(Escape(hdcPrn,STARTDOC,sizeof szMessage-1,szMessage,NULL) > 0){
            PageGDICalls(hdcPrn,xPage,yPage,hwnd);
            if(Escape(hdcPrn,NEWFRAME,0,NULL,NULL) > 0)
            Escape(hdcPrn,ENDDOC,0,NULL,NULL);
            else bError = TRUE;
     }
     else bError = TRUE;
     DeleteDC(hdcPrn);
     return bError;
}
```

เอกสารอ้างอิง

1. Texas Instrument, "First Generation TMS32010 User's Guide", 1988.
2. Kun-Shan Lin Editor, "digital Signal Processing Application with the TMS32010 Family, Volumn 1", 1987.
3. Louis Nashelsky Editor, "Introduction to Digital Technology".
4. International Business Machines Corporation, "IBM Technical Referrence", 1983.
5. Eugene L. Zuch Editor, "Data Acquisition and Conversion Handbook"
6. David J. Defatta, Joseph G. Lucas, William S. Hodgkiss ,"Digital signal processing : A System Design Approach" , JOHN WILEY & SONS ,1988.
7. DAVE WILLIAMS, "The Programer's Technical Reference" , JOHN WILEY & SONS ,1991.
8. Texas Instrument, " Digital Signal Processing Products and Applications ", 1988.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้ด้วยความช่วยเหลือเป็นอย่างดียิ่ง จาก อาจารย์ มนัส สังวรศิลป์ และ อาจารย์ ชินภัทร นันทจิวากรชัย ซึ่งเป็นอาจารย์ที่ปรึกษาตลอดการทำปริญญานิพนธ์นี้ ได้กรุณาให้ยืมเอกสาร และตำราต่างๆที่เกี่ยวข้องกับการทำปริญญานิพนธ์ และคณาจารย์ท่านอื่นๆ ที่ได้ให้คำแนะนำอันมีค่าและเป็นประโยชน์อย่างมาก ข้าพเจ้าจึงขอขอบพระคุณคณาจารย์ทุกท่านที่ได้กล่าวนามมาข้างต้น และบุคคลอื่นๆ ที่ได้ช่วยเหลือสนับสนุนการทำปริญญานิพนธ์นี้ซึ่งไม่สามารถกล่าวรายนามได้ครบถ้วนทุกท่าน เป็นอย่างสูงมา ไว้ ณ.ที่นี้

SRAM

PM: Barbara (AFORCE)

32K x 8 SRAM

(Total 1 Page)

FEATURES

- High speed: 15, 20, 25, 30, 35 and 45ns
- High-performance, low-power, CMOS double-metal process
- Single +5V ±10% power supply
- Easy memory expansion with $\overline{CE}$ and $\overline{OE}$ options
- All inputs and outputs are TTL compatible

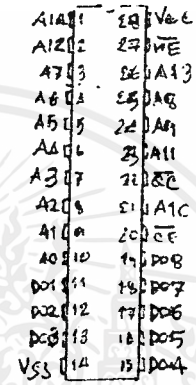
OPTIONS

	MARKING
Timing	
15ns access	-15
20ns access	-20
25ns access	-25
30ns access	-30
35ns access	-35
45ns access	-45
Package	
Plastic DIP (300 mil)	None
Plastic DIP (600 mil)	W
Plastic SOJ (300 mil)	DJ
Plastic ZIP	Z
2V data retention	L
Temperature	
Industrial (-40°C to +85°C)	IT
Automotive (-40°C to +125°C)	AT
Extended (-55°C to +125°C)	XT

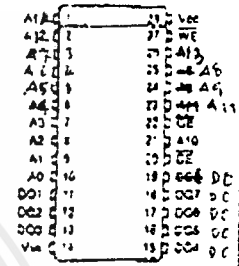
Available in ceramic packages tested to meet military specifications. Please refer to Micron's *Military Data Book*.

PIN ASSIGNMENT (Top View)

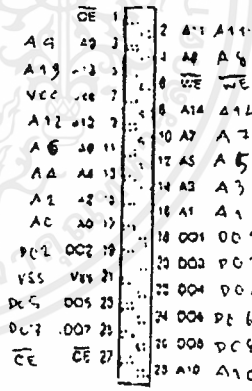
28-Pin DIP (A-9, A-11)



28-Pin SOJ (E-8)



28-Pin ZIP (C-5)



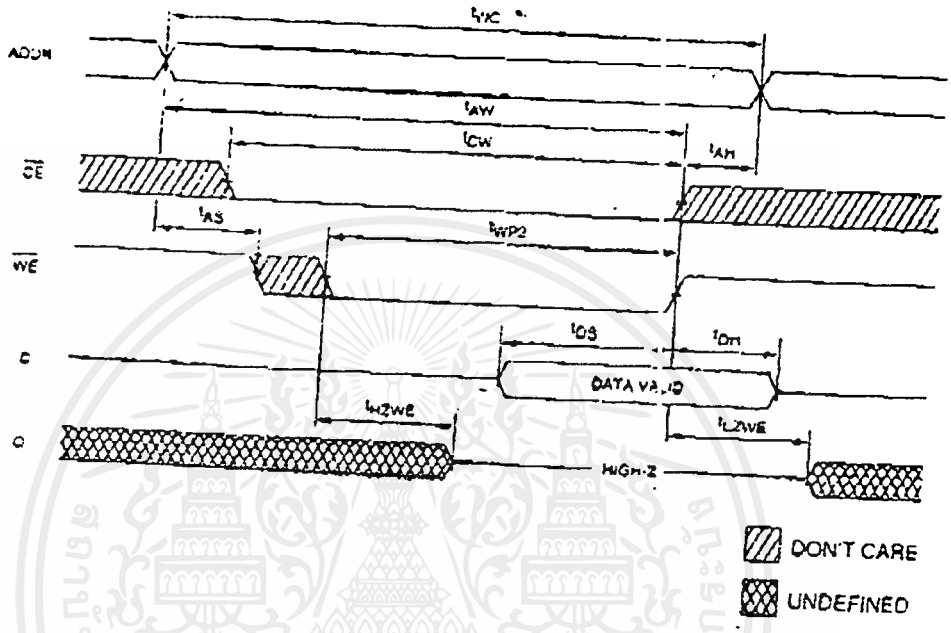
GENERAL DESCRIPTION

The Micron SRAM family employs high-speed, low-power CMOS designs using a four-transistor memory cell. Micron SRAMs are fabricated using double-layer metal, double-layer polysilicon technology.

For flexibility in high-speed memory applications, Micron offers chip enable ($\overline{CE}$) and output enable ($\overline{OE}$) options. This organization has enhancements can place the outputs in High-Z for additional flexibility in system design. Writing to these devices is accomplished when write enable ($\overline{WE}$) and $\overline{CE}$ inputs are both LOW. Reading is accomplished when $\overline{WE}$ remains HIGH and $\overline{CE}$ and $\overline{OE}$ go LOW. The device offers a reduced power standby mode when disabled. This allows system designers to achieve their low standby power requirements.

All devices operate from a single +5V power supply and all inputs and outputs are fully TTL compatible.

WRITE CYCLE NO. 3
(Write Enable Controlled) 7, 12



AC TEST CONDITIONS

Input pulse levels	Vss to 3.0V
Input rise and fall times	5ns
Input timing reference levels	1.5V
Output reference levels	1.5V
Output load	See Figures 1 and 2

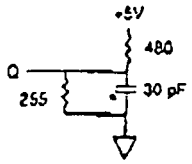


Fig. 1 OUTPUT LOAD EQUIVALENT

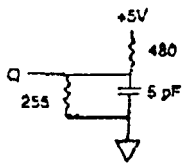


Fig. 2 OUTPUT LOAD EQUIVALENT

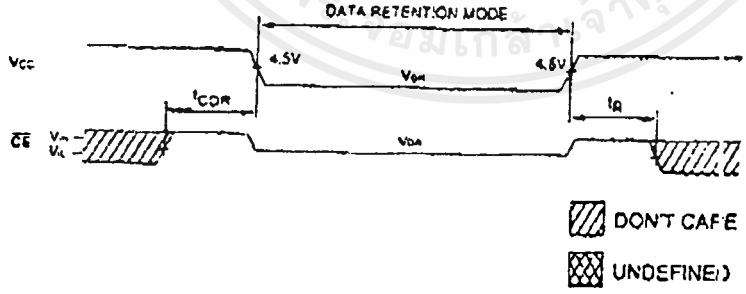
NOTES

1. All voltages referenced to Vss (GND).
2. -3V for pulse width < 20ns
3. Icc is dependent on output loading and cycle rates.
4. This parameter is sampled.
5. Test conditions as specified with the output loading as shown in Fig. 1 unless otherwise noted.
6. tHZCE, tHZOE and tHZWE are specified with CL = 5pF as in Fig. 2. Transition is measured ±500mV from steady state voltage.
7. At any given temperature and voltage condition, tHZCE is less than tLZCE and tHZWE is less than tLZWE
8. $\overline{WE}$ is HIGH for READ cycle
9. Device is continuously selected. All chip enables are held in their active state.
10. Address valid prior to or coincident with latest occurring chip enable.
11. tRC = Read Cycle Time.
12. Chip enable ($\overline{CE}$) and write enable ($\overline{WE}$) can initiate and terminate a WRITE cycle.
13. For automotive, industrial and extended temperature specifications, refer to page 1-175.
14. Typical values are measured at 5V, 25°C and 20ns cycle time

DATA RETENTION ELECTRICAL CHARACTERISTICS (L Version Only)

DESCRIPTION	CONDITIONS		SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Vcc for Retention Data			VDR	2		—	V	
Data Retention Current	$\overline{CE}$ ≥ (Vcc -0.2V) VIN ≥ (Vcc -0.2V) or ≤ 0.2V	Vcc = 2V	ICCDR		95	300	µA	
		Vcc = 3V			350	400	µA	
Chip Deselect to Data Retention Time			tCDR	0		—	ns	4
Operation Recovery Time			tR	tRC			ns	4, 11

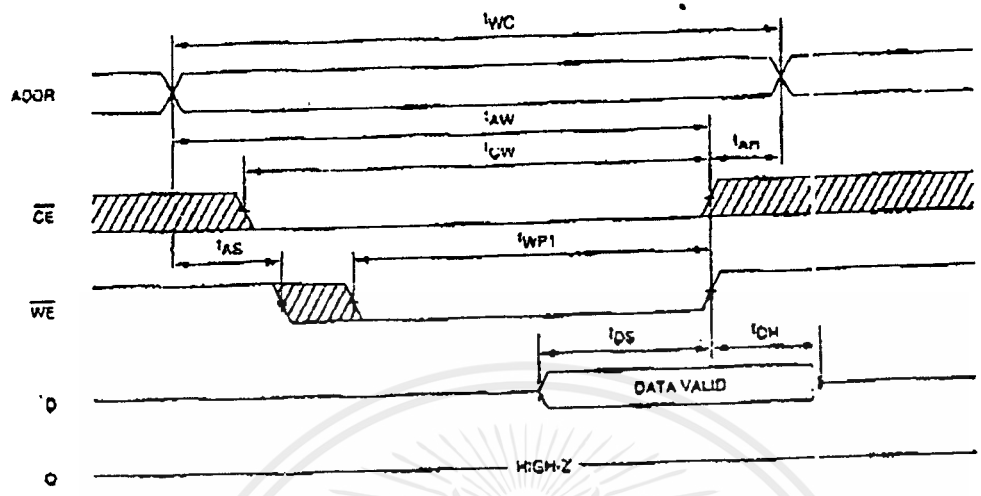
LOW Vcc DATA RETENTION WAVEFORM



DON'T CARE

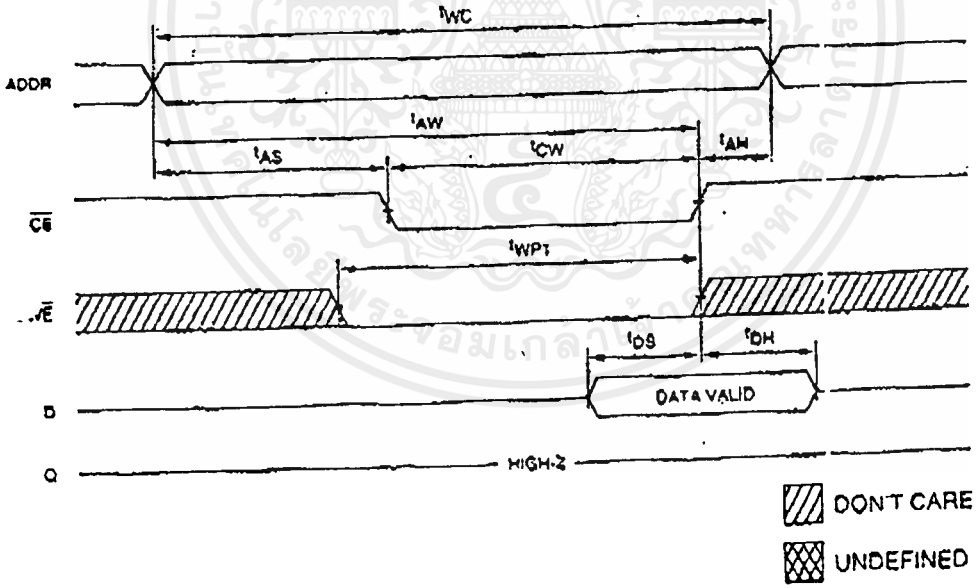
UNDEFINED

WRITE CYCLE NO. 1
(Write Enable Controlled) [12]

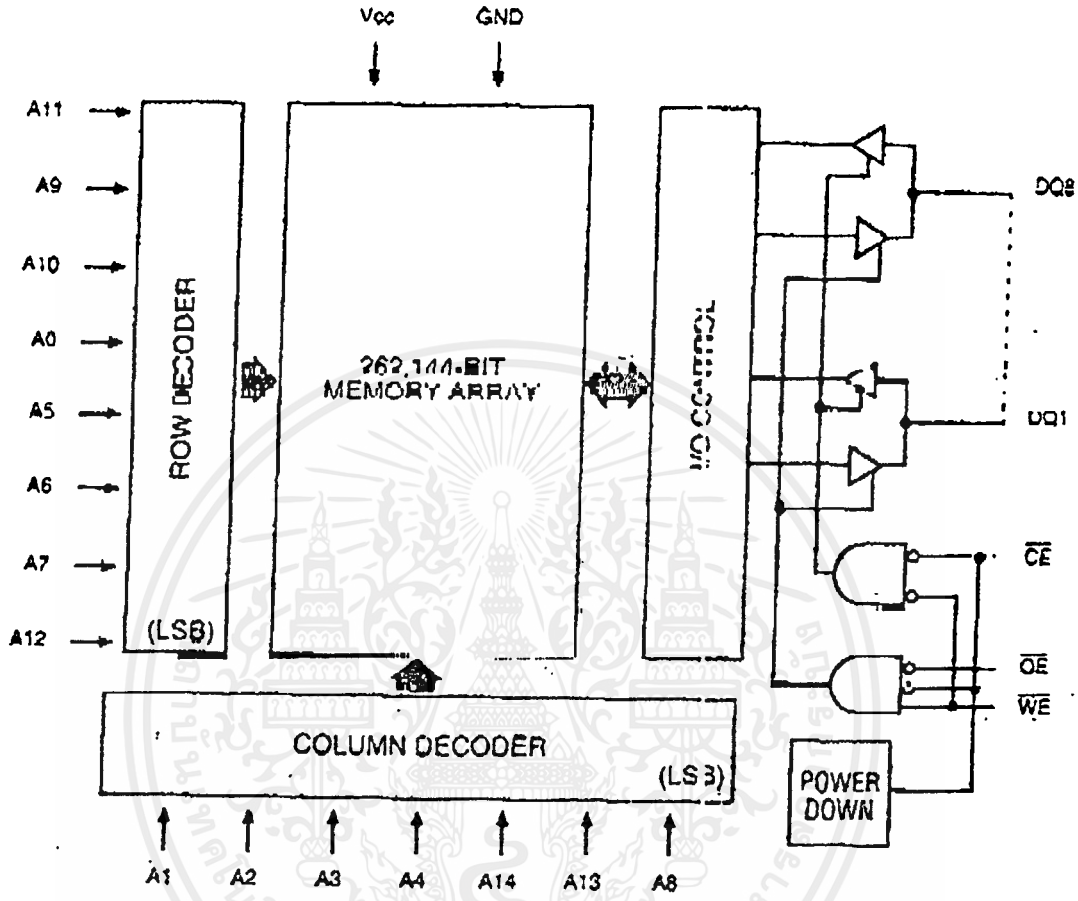


NOTE: Output enable ($\overline{OE}$) is inactive (HIGH).

WRITE CYCLE NO. 2
(Chip Enable Controlled)



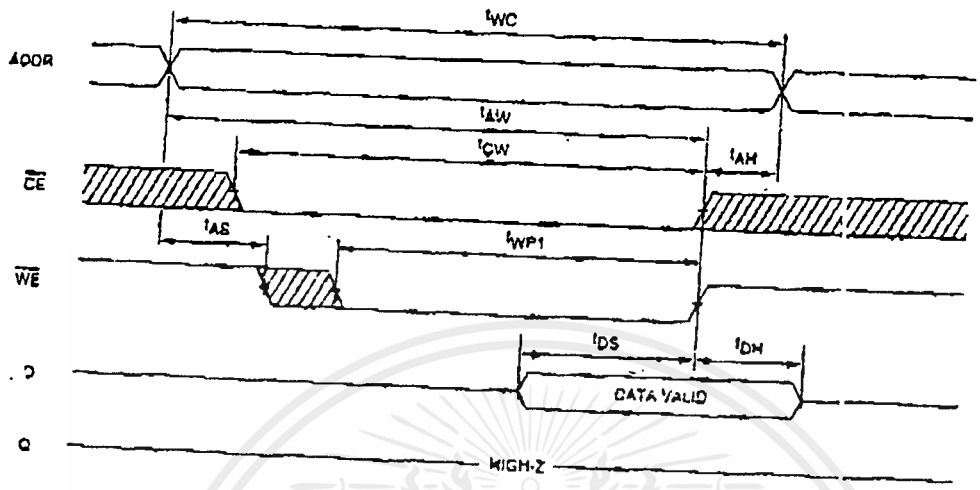
FUNCTIONAL BLOCK DIAGRAM



TRUTH TABLE

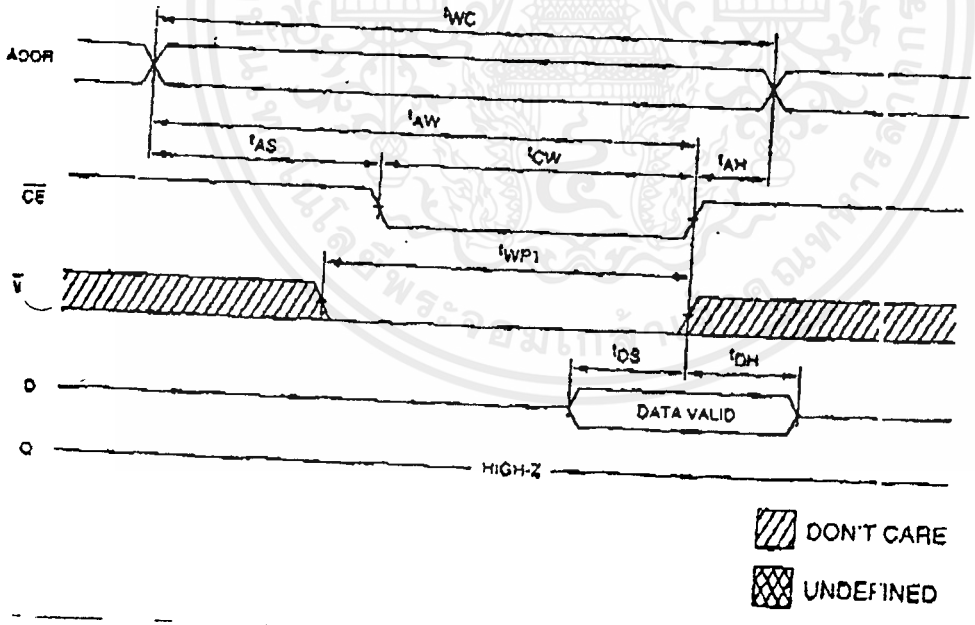
MODE	OE	CE	WE	DQ	POWER
STANDBY	X	H	X	HIGH-Z	STANDBY
READ	L	L	H	Q	ACTIVE
READ	H	L	H	HIGH-Z	ACTIVE
WRITE	X	L	L	D	ACTIVE

WRITE CYCLE NO. 1
(Write Enable Controlled) [12]



TE: Output enable ($\overline{OE}$) is inactive (HIGH).

WRITE CYCLE NO. 2
(Chip Enable Controlled)



DON'T CARE

UNDEFINED

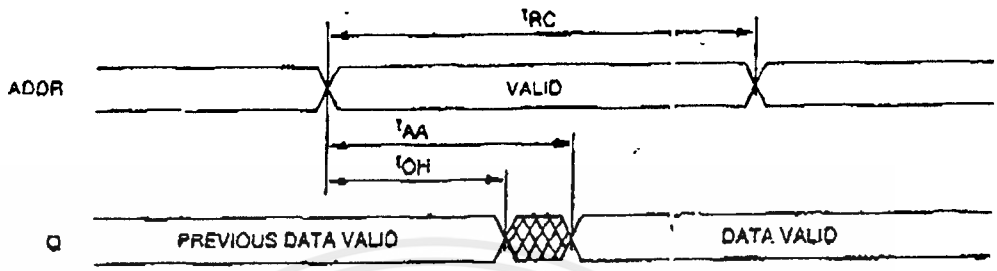
FAST SRAM

ELECTRICAL CHARACTERISTICS AND RECOMMENDED AC OPERATING CONDITIONS

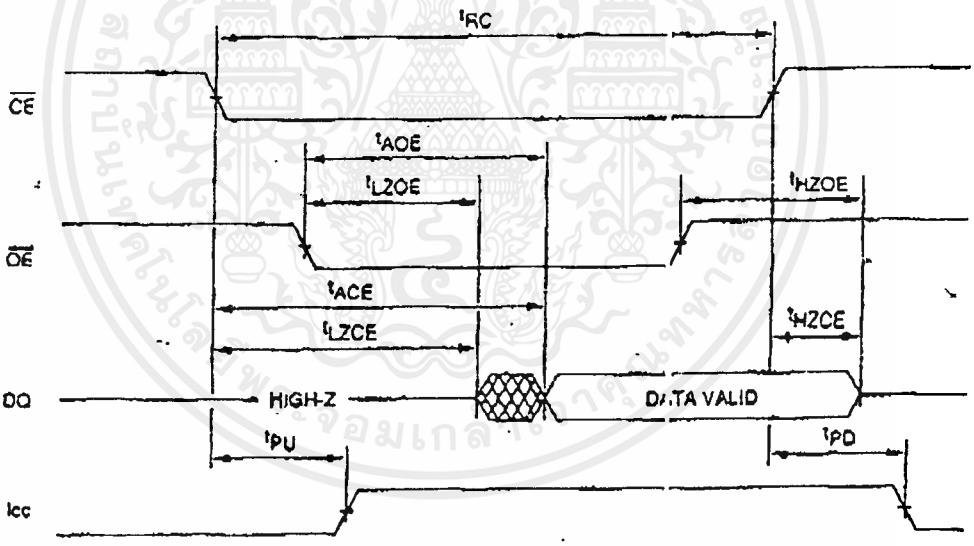
(Note 5, 13) (0°C ≤ T_A ≤ 70°C, Vcc = 5V ±10%)

DESCRIPTION		-15		-20		-25		-30		-35		-45			
	SYM	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	UNITS	NOTES
READ Cycle															
READ cycle time	tRC	15		20		25		30		35		45		ns	
Address access time	tAA		15		20		25		30		35		45	ns	
Chip Enable access time	tACE		15		20		25		30		35		45	ns	
Output hold from address change	tOH	3		3		5		5		5		5		ns	
Chip Enable to output in Low-Z	tLZCE	4		6		6		6		6		6		ns	
Chip disable to output in High-Z	tHZCE		8		9		9		12		15		18	ns	6,7
Chip Enable to power-up time	tPU	0		0		0		0		0		0		ns	
Chip disable to power-down time	tPD		15		20		25		30		35		45	ns	
Output Enable access time	tAOE		8		8		8		10		12		15	ns	
Output Enable to output in Low-Z	tLZOE	0		0		0		0		0		0		ns	
Output disable to out put in High-Z	tHZOE		6		7		7		10		12		15	ns	
WRITE Cycle															
WRITE cycle time	tWC	15		20		20		25		30		35		ns	
Chip Enable to end of write	tCW	10		15		15		18		20		25		ns	
Address valid to end of write	tAW	10		15		15		18		20		25		ns	
Address setup time	tAS	0		0		0		0		0		0		ns	
Address hold from end of write	tAH	0		0		0		0		0		0		ns	
WRITE pulse width	tWP1	10		15		15		18		20		25		ns	
WRITE pulse width	tWP2	12		15		15		18		20		25		ns	
Data setup time	tDS	7		10		10		12		15		20		ns	
Data hold time	tDH	0		0		0		0		0		0		ns	
Write disable to output in Low-Z	tLZWE	4		5		5		5		5		5		ns	
Write Enable to output in High-Z	tHZWE		7		10		10		12		15		18	ns	6

READ CYCLE NO. 1 [8, 9]



READ CYCLE NO. 2 [7, 8, 10]



DON'T CARE

UNDEFINED

FAST SRAM

ABSOLUTE MAXIMUM RATINGS*

Voltage on Vcc Supply Relative to Vss ..... -1V to +7V
Storage Temperature (Plastic) ..... -55°C to +150°C
Power Dissipation 1W
Short Circuit Output Current 50mA

*Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

ELECTRICAL CHARACTERISTICS AND RECOMMENDED DC OPERATING CONDITIONS

(0°C ≤ T_A ≤ 70°C; Vcc = 5V ±10%)

DESCRIPTION	CONDITIONS	SYMBOL	MIN	MAX	UNITS	NOTES
Input High (Logic 1) Voltage		V_{IH}	2.2	Vcc+1	V	1
Input Low (Logic 0) Voltage		V_{IL}	-0.5	0.8	V	1, 2
Input Leakage Current	0V ≤ V_{IN} ≤ Vcc	I_{LI}	-5	5	µA	
Output Leakage Current	Output(s) Disabled 0V ≤ V_{OUT} ≤ Vcc	I_{LO}	-5	5	µA	
Output High Voltage	I_{OH} = -4.0mA	V_{OH}	2.4		V	1
Output Low Voltage	I_{OL} = 8.0mA	V_{OL}		0.4	V	1

DESCRIPTION	CONDITIONS	SYMBOL	TYP	MAX						UNITS	NOTES
				-15	-20	-25	-30	-35	-45		
Power Supply Current: Operating	$\overline{CE}$ ≤ V_{IL}; Vcc = MAX f = MAX = 1/t_{RC} Outputs Open	Icc	75	140	120	110	90	90	90	mA	3, 14
Power Supply Current: Standby	$\overline{CE}$ ≥ V_{IH}; Vcc = MAX f = MAX = 1/t_{RC} Outputs Open	I_{SB1}	11	30	30	25	25	25	25	mA	14
	$\overline{CE}$ ≥ Vcc -0.2V; Vcc = MAX V_{IL} ≤ Vss +0.2V V_{IH} ≥ Vcc -0.2V; f = 0	I_{SB2}	.04	5	5	5	5	7	7	mA	14

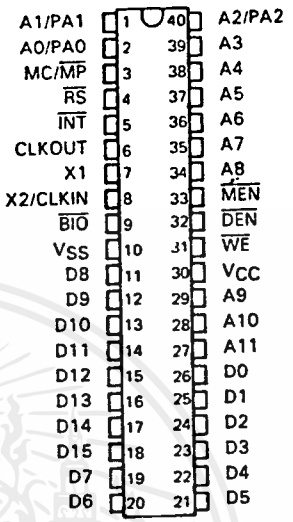
CAPACITANCE

DESCRIPTION	CONDITIONS	SYMBOL	MAX	UNITS	NOTES
Input Capacitance	T_A = 25°C, f = 1 MHz Vcc = 5V	Ci	6	pF	4
Output Capacitance		Co	5	pF	4

JANUARY 1987 – REVISED OCTOBER 1987

- 160-ns Instruction Cycle
- 144/256-Word On-Chip Data RAM
- 1.5K/4K-Word On-Chip Program ROM
- 4K-Word On-chip Program EPROM (TMS320E15/E17)
- EPROM Code Protection for Copyright Security
- 4K-Word Total External Memory at Full Speed
- 32-Bit ALU/Accumulator
- 16 x 16-Bit Multiplier with a 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- Dual-Channel Serial Port (TMS32011/C17/E17)
- 16-Bit Bidirectional Data Bus with 50-Mbps Transfer Rate
- Single 5-V Supply
- Packaging: 40-Pin DIP and 44-Pin PLCC
- Commercial and Military Versions Available
- NMOS Technology:
  - TMS32010-25 . . . . . 160-ns cycle time
  - TMS32010 . . . . . 200-ns cycle time
  - TMS32010-14 . . . . . 280-ns cycle time
  - TMS32011 . . . . . 200-ns cycle time

TMS32010, TMS320C10
N PACKAGE
(TOP VIEW)



- CMOS Technology:
  - TMS320C10-25 . . . . . 160-ns cycle time
  - TMS320C10 . . . . . 200-ns cycle time
  - TMS320C15-25 . . . . . 160-ns cycle time
  - TMS320C15 . . . . . 200-ns cycle time
  - TMS320E15 (EPROM) . 200-ns cycle time
  - TMS320C17-25 . . . . . 160-ns cycle time
  - TMS320C17 . . . . . 200-ns cycle time
  - TMS320E17 (EPROM) . 200-ns cycle time

This data sheet provides complete design documentation for all the first-generation devices of the TMS320 family. This facilitates the selection of the devices best suited for user applications by providing all specifications and special features for each TMS320 member. This data sheet is divided into four major sections: architecture, electrical specifications (NMOS and CMOS), timing diagrams, and mechanical data. In each of these sections, generic information is presented first, followed by specific device information. An index is provided for quick reference to specific information about a device.

description

The TMS320 family of 16/32-bit single-chip digital signal processors combines the flexibility of a high-speed controller with the numerical capability of an array processor, thereby offering an inexpensive alternative to multichip bit-slice processors. The highly paralleled architecture and efficient instruction set provide speed and flexibility to produce a MOS microprocessor family capable of executing 6.4 MIPS (million instructions per second). The TMS320 family optimizes speed by implementing functions in hardware that other processors implement through microcode or software. This hardware-intensive approach provides the design engineer with processing power previously unavailable on a single chip.

CLOCK CHARACTERISTICS AND TIMING

The TMS32010 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Crystal frequency f_x	0°C to 70°C	6.7		20.5	6.7		25.0	MHz
C1, C2	0°C to 70°C		10			10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, See Figure 2	195.12	200		160			ns
$t_{r(C)}$	CLKOUT rise time			10			10		ns
$t_{f(C)}$	CLKOUT fall time			8			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92			74		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90			72		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	25‡		60‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4 * t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	40		150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.475t_{c(MC)}$†		$0.525t_{c(MC)}$†	$0.475t_{c(MC)}$†		$0.525t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low, $t_{c(MC)}$ = 50 ns		20†			18†		ns
$t_{w(MCH)}$	Pulse duration master clock high, $t_{c(MC)}$ = 50 ns		20†			18†		ns

†Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	TMS32010 MIN	TMS32010 TYP	TMS32010 MAX	TMS32010-25 MIN	TMS32010-25 TYP	TMS32010-25 MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, See Figure 2	10†		50	10†		40	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ − 12	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		− 10†		15	− 10†		12	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ − 12	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		− 10†		15	− 10†		12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 15	½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 12	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		− 10†		15	− 10†		12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				¼$t_{c(C)}$ + 65			¼$t_{c(C)}$ + 52	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		¼$t_{c(C)}$ − 5†			¼$t_{c(C)}$ − 5†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				¼$t_{c(C)}$ + 30†			¼$t_{c(C)}$ + 30†	ns
t_v	Data bus OUT valid after CLKOUT↓		¼$t_{c(C)}$ − 10			¼$t_{c(C)}$ − 10			ns
$t_{h(A-WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑ or $\overline{DEN}$↑ (see Note 1)		0			0			ns
$t_{su(A-MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		¼$t_{c(C)}$ − 45			¼$t_{c(C)}$ − 35			ns

†Values derived from characterization data and not tested.
NOTE 1: Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS32010 MIN	TMS32010 NOM	TMS32010 MAX	TMS32010-25 MIN	TMS32010-25 NOM	TMS32010-25 MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF, See Figure 2	50			40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11} Delay time $\overline{DEN}$†, $\overline{WE}$†, and $\overline{MEN}$† from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 2			$\frac{1}{2}t_{c(C)} - 50$†	ns
$t_{dis(R)}$ Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} - 50$†	ns

†Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

	TMS32010			TMS32010-25			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$ Reset $\overline{RS}$ setup time prior to CLKOUT (see Note 3)	50			40			ns
$t_{w(R)}$ $\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 3: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

	TMS32010			TMS32010-25			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$ Fall time ($\overline{INT}$)			15			15	ns
$t_{w(INT)}$ Pulse duration $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$ Setup time $\overline{INT}$↓ before $\overline{CLKOUT}$↓	50			40			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

	TMS32010			TMS32010-25			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$ Fall time $\overline{BIO}$			15			15	ns
$t_{w(IO)}$ Pulse duration $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$ Setup time $\overline{BIO}$↓ before CLKOUT↓	50			40			ns