การประยุกต์ใช้งาน A/D & D/A

(A/D & D/A APPLICATION)

โดย

นายธวัชชัย คุณพิทักษ์วัฒนา

นายเลิศ แซ่ตัน

นายสุปรีดี วัฒนการุณ

ปริญญานิพนธ์สำหรับปริญญาอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปีการศึกษา 2535

การประยุกต์ใช้งาน A/D & D/A

(A/D & D/A APPLICATION)

โดย

นายธวัชชัย คุณพิทักษ์วัฒนา

นายเลิศ แซ่ตัน

นายสุปรีดี วัฒนการุณ

อาจารย์ที่ปรึกษา

อ. ภากร หุตะสังกาศ

ปริญญานิพนธ์ปีการศึกษา 2535

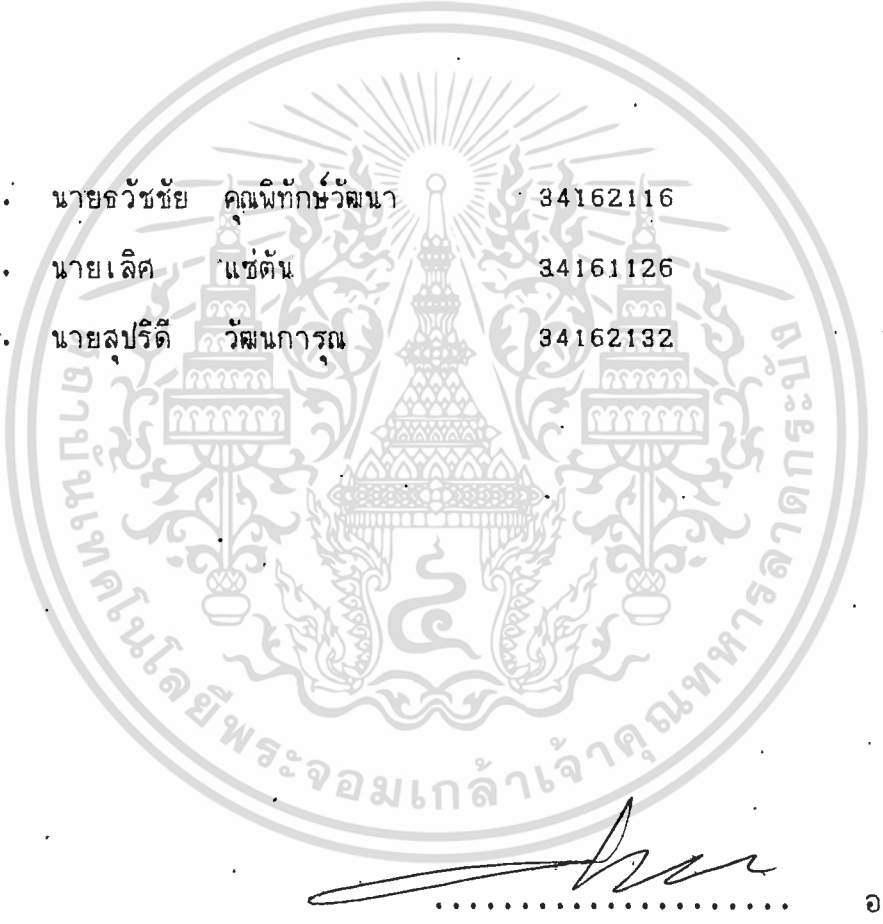
ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน A/D & D/A

ผู้จัดทำ

1. นายธวัชชัย คุณพิทักษ์วัฒนา 34162116
2. นายเลิศ แซ่ตัน 34161126
3. นายสุปริดี วัฒนการุณ 34162132



..... อาจารย์ที่ปรึกษา

(อ. ภากร หุตะสังกาศ)

การประยุกต์ใช้งาน A/D & D/A

ชวัชชัย คุณพิทักษ์วัฒนา

เลิศ แซ่ตัน

สุปรีดิ วัฒนการุณ

อ.ภากร หุตะสังกาศ อาจารย์ที่ปรึกษา

ปีการศึกษา 2535

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการนำเสนอในด้านของการประยุกต์ใช้งานเครื่องคอมพิวเตอร์ร่วมกับวงจรที่ใช้ในการทดลองแลปฯ โดยใช้โปรแกรมภาษาปาสคาลในการควบคุมการทำงาน เพราะเป็นภาษาแบบโครงสร้างที่ง่ายต่อความเข้าใจ โดยแลปที่นำมารวมเป็นปริญญานิพนธ์ชิ้นนี้คือ ADC (ANALOG TO DIGITAL CONVERTER) , DAC (DIGITAL TO ANALOG CONVERTER) และได้เพิ่มเติม การประยุกต์ใช้งานทั้งสองแลปร่วมกัน โดยทำเป็นเครื่องบันทึกเสียงพูด โดยสามารถเก็บเสียงไว้ในหน่วยความจำ หรือ แผ่นจานแม่เหล็กก็ได้ โดยในช่วงแรก สามารถเก็บสัญญาณเสียงได้ประมาณ 8 วินาที และหลังจากนั้นก็จะพัฒนาขีดความสามารถให้สูงขึ้นต่อไป

A/D & D/A Application

TAWATCHAI KUNPITAKWATTANA

LERT SAETAN

SUPREEDEE WATTANAKAROON

PHAKORN HOOTASANGKART ADVISOR

ABSTRACT

This thesis present the computer application with the circuit that is used in engineering laboratory . And the PASCAL language is used to control the operation because it is structure program that is simply to understand . The LAB include that Analog to Digital Converter , Digital to Analog Converter and apply of the both circuits . For Example in the thesis , It is "SPEECH RECORDER" . The circuit can record the speech in memory or in disk for 8 second . And we will develop the ability of the circuit .

วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อนำปริญญานิพนธ์ชิ้นนี้ไปใช้ในการทดลองการแปลงสัญญาณจากอนาล็อกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก

2. เพื่อให้นักศึกษาสามารถเข้าใจถึงการทำงานของการแปลงสัญญาณอนาล็อกเป็นดิจิตอลและดิจิตอลเป็นอนาล็อกได้เร็วขึ้น

3. เพื่อให้สามารถนำไปประยุกต์ใช้งานในด้านการแปลงสัญญาณระหว่าง ADC และ DAC

ขอบเขตของปริญญานิพนธ์

1. สามารถสร้างวงจรแปลงสัญญาณจากอนาล็อกไปเป็นสัญญาณดิจิตอล
2. สามารถสร้างวงจรแปลงสัญญาณจากดิจิตอลไปเป็นสัญญาณอนาล็อก
3. สามารถนำวงจรได้ศึกษามา มาประยุกต์ใช้เป็นตัวบันทึกสัญญาณเสียงพูดลงบนแผ่นดิสก์
4. สามารถทำการอ่านข้อมูลที่ได้จากการบันทึกในข้อ 3. มากลับเป็นเสียงพูดดังเดิมได้
5. มีความเข้าใจในการจัดสรรหน่วยความจำ เพื่อสร้างขอบเขตในการสร้างย่านบันทึกข้อมูลได้เกินขอบเขต 1 เซกเมนต์(64K)

สารบัญ

บทนำ

ปริญญานิพนธ์ฉบับนี้ เป็นตัวสรุปโปรเจคต์การประยุกต์การทดลอง lab ในหัวข้อการประยุกต์การใช้งาน A/D และ D/A (ANALOG TO DIGITAL & DIGITAL TO ANALOG) ซึ่งที่แล้วมาในการทดลองแต่ละครั้ง น.ศ. จะต้องเสียเวลาไปมากในการที่จะต้องต่อวงจร เบิกอุปกรณ์ รวมถึงความไม่พร้อมของอุปกรณ์ที่จะรองรับต่อการทดลองในแต่ละครั้ง และเมื่อต้องการทดลองการประยุกต์ใช้งานก็จะต้องพบกับปัญหาที่ยุ่งยากเพิ่มขึ้นไปอีก ดังนั้น เพื่อความสะดวกในหลาย ๆ ด้าน และลดความเสี่ยงในความไม่รอบคอบของผู้ประกอบวงจรซึ่งอาจจะทำให้วงจรไม่ทำงาน , อุปกรณ์เกิดความเสียหาย ดังนั้นจึงได้เกิดแนวความคิดที่จะทำการรวบรวมเอาการทดลองต่าง ๆ ซึ่งต้องใช้อุปกรณ์ย่อย ๆ เป็นจำนวนมากมาทำเป็นโมดูลของชุดทดลองนั้น ๆ

ในโปรเจคต์ชิ้นนี้จะเป็นการรวบรวมเอาการทดลองต่างๆ จำนวน 3 การทดลองมารวมเป็นชุดทดลองชุดเดียว การทดลองทั้งสามนั้นประกอบด้วย ADC (ANALOG TO DIGITAL CONVERTER) DAC (DIGITAL TO ANALOG CONVERTER) และการนำเอา ADC และ DAC มาประยุกต์ใช้งาน

ชื่อสัญญาณ	I/O	ความหมาย
OSC	O	สัญญาณนาฬิกาที่มีความกว้าง 70 ns ความถี่ 14.31818 MHz
CLK	O	สัญญาณนาฬิกาของระบบ มีความถี่ 4.77 MHz มีช่วงคาบ 210 ns
RESET DRV	O	สายสัญญาณนี้ใช้ในการรีเซตระบบในขณะเริ่มเปิดเครื่อง
A0 - A19	O	ADDRESS BUS A0 - A19
D0 - D7	I/O	DATA BUS BIT 0 - 7
ALE	O	ทำการแลทช์แอดเดรสเป็นสัญญาณที่กำหนดค่าแอดเดรส
$\overline{\text{I/O CH CK}}$	I	เป็นสัญญาณตรวจสอบแชนแนล I/O สัญญาณนี้จะมีผลต่อเนื่องมาเมื่อควบคุมระบบ โดยส่งผลมาในลักษณะ PARITY ERROR
I/O CHRDY	I	สัญญาณนี้ปกติเป็น "0" สัญญาณนี้จะทำให้เกิดการ SYNCHRONIZE อุปกรณ์อินพุตเอาท์พุทที่ทำงานช้าให้เข้ากับระบบได้
IRQ2 - IRQ7	I	เป็นสัญญาณของอินเตอร์รัพท์ 2-7
$\overline{\text{IOR}}$	O	สัญญาณอ่านอินพุทเอาท์พุท
$\overline{\text{IOW}}$	O	สัญญาณเขียนอินพุทเอาท์พุท
$\overline{\text{MEMR}}$	O	สัญญาณอ่านหน่วยความจำ
$\overline{\text{MEMW}}$	O	สัญญาณเขียนหน่วยความจำ
DRQ1 - DRQ3	I	สัญญาณขอ DMA1 - DMA3
$\overline{\text{DACK0}}$ - $\overline{\text{DACK3}}$	O	สัญญาณตอบรับการขอ DMA1 - DMA3
AEN	O	สัญญาณการอีเนเบิลแอดเดรส
T/C	O	สัญญาณการนับ TERMINAL
CARD SLCTD	I	สัญญาณการเลือกการ์ด

สายสัญญาณที่เหลือคือสายสัญญาณไฟเลี้ยงที่จะต้องจ่ายให้กับระบบที่จะเชื่อมต่อและสายกราวด์

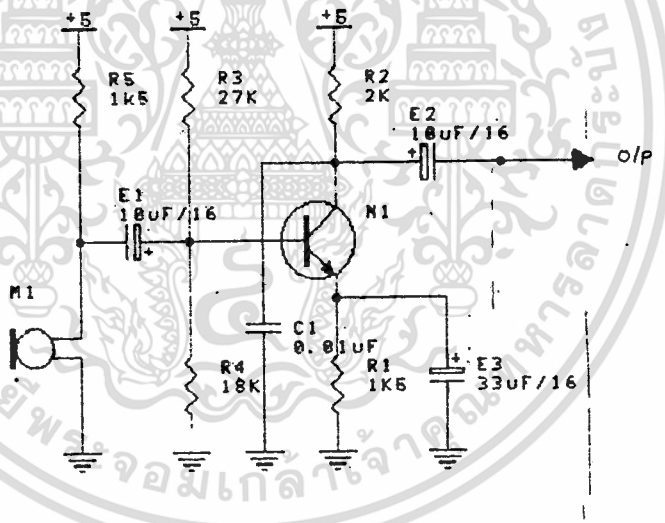
HEX RANGE	DEVICE
000-01F	DMA CONTROLLER 1 , 8237A-5
020-03F	INTERRUPT CONTROLLER 1 , 8259A , MASTER
040-05F	TIMER , 8254.2
060-06F	8042 (KEY BOARD)
070-07F	REAL TIME CLOCK , NMI (NON MASKABLE INTERRUPT) MASK
080-09F	DMA PAGE REGISTER , 74LS612
0A0-0BF	INTERRUPT CONTROLLER 2 , 8259A
0C0-0DF	DMA CONTROLLER 2 , 8237A-5
0F0	CLEAR MATH COPROCESSOR BUSY
0F1	RESET MATH COPROCESSOR
0F8-0FF	MATH COPROCESSOR
1F0-1F8	FIXED DISK
200-207	GAME I/O
278-27F	PARALLEL PRINTER PORT 2
2F8-2FF	SERIAL PORT 2
300-31F	PROTOTYPE CARD
360-36F	RESERVED
378-37F	PARALLEL PRINTER PORT 1
380-38F	SDLC , BISYNCHRONOUS 2
3A0-3AF	BISYNCHRONOUS 1
3B0-3BF	MONOCHROME DISPLAY AND PRINTER ADAPTER
3C0-3CF	RESERVED
3D0-3DF	COLOR/GRAPHICS MONITER ADAPTER
3F0-3F7	DISKETTE CONTROLLER
3F8-3FF	SERIAL PORT 1

วงจรที่จะนำมารวมเป็นการทดลองทั้งสามการทดลองนั้น จะประกอบด้วย ADC , DAC และการประยุกต์ใช้งาน ADC และ DAC ทั้งสามการทดลองนี้ จะร่วมกันเป็น "เครื่องบันทึกเสียงพูด" ซึ่งจะมีส่วนที่จะ SAMPLING สัญญาณ ในส่วนนี้จะใช้ SOFTWARE ในการ SAMPLING เป็นภาษา PASCAL วงจรย่อย ๆ ที่ประกอบกันเป็นเครื่องบันทึกเสียงพูดมีดังต่อไปนี้

- วงจร PREE MIC
- วงจร ADC
- วงจร DAC
- ส่วน DECODE PORT

วงจร PRE MIC

วงจร PRE MIC นี้ จะทำหน้าที่ขยายสัญญาณจาก MICROPHONE ซึ่งมีขนาดเล็กให้มีขนาดใหญ่พอที่จะนำมาใช้งานได้ ลักษณะของวงจรจะเป็นดังรูป



การทำงานของวงจรจะเป็นดังนี้

เมื่อมีการพูด ก็จะเกิดสัญญาณออกมาจาก MICROPHONE ไปเข้าขาเบสของทรานซิสเตอร์ ทรานซิสเตอร์ก็จะทำการขยายสัญญาณให้มีขนาดใหญ่ เพื่อที่จะนำเอาสัญญาณนี้ไปทำการ แปลงเป็นสัญญาณดิจิตอลต่อไป

การ DECODE PORT

การ DECODE PORT คือการทำให้เครื่องคอมพิวเตอร์สามารถติดต่อกับอุปกรณ์ภายนอก เพื่อจุดประสงค์ต่าง ๆ เช่น ใช้ในการควบคุมอุปกรณ์ , ใช้ในการแสดงข้อมูลต่าง ๆ เป็นต้น การ DECODE PORT ทำได้โดยการต่อขาสัญญาณต่าง ๆ บน SLOT ของคอมพิวเตอร์ ออกมาเพื่อทำการรควบคุม PORT ต่าง ๆ สัญญาณต่าง ๆ ที่ต้องใช้ในการ DECODE PORT มีดังต่อไปนี้

- ขาสัญญาณ ADDRESS ต่าง ๆ ใช้กำหนดหมายเลข PORT
- ~IOW ใช้เมื่อเขียนข้อมูลไปยัง PORT
- ~IOR ใช้เมื่ออ่านข้อมูลจาก PORT
- ALE ใช้บอกอุปกรณ์ต่าง ๆ ว่าสัญญาณที่ BUS ในขณะนั้น ๆ เป็นสัญญาณ ADDRESS

การกำหนดหมายเลข PORT

ในการ INTERFACE กับเครื่องคอมพิวเตอร์นั้น หมายเลข PORT ที่จะใช้งานนั้นจะต้องเป็น PORT ที่เครื่องคอมพิวเตอร์ไม่ได้ใช้งาน เนื่องจากถ้าหากเราไปใช้ PORT ซ้ำกับ PORT ที่เครื่องคอมพิวเตอร์ใช้งานอยู่ จะทำให้การทำงานของทั้งเครื่องคอมพิวเตอร์ และวงจรที่ออกแบบไว้ผิดพลาด จาก SPEC ของเครื่องคอมพิวเตอร์ PORT ที่ไม่ได้ใช้งานจะอยู่ในช่วง 300H ถึง 30FH ดังนั้น ในโปรเจคต์ชิ้นนี้ จะใช้หมายเลข PORT เบอร์ 300H และ 301H

การกำหนดสัญญาณที่จะใช้ในการ DECODE PORT

สัญญาณ ADDRESS จากหมายเลข PORT ที่ใช้งานคือ 300H และ 301H ดังนั้น LOGIC ที่ ADDRESS แต่ละเส้นจะเป็นดังนี้

3 0 0 3 0 1

0011 0000 0000 0011 0000 0001

จะเห็นว่า A1-A7 และ A10-A11 จะต้องเป็น "0" A8-A9 จะต้องเป็น "1" PORT 300H นั้น A0 จะต้องเป็น "0" และ PORT 301H นั้น A0 จะต้องเป็น "1"

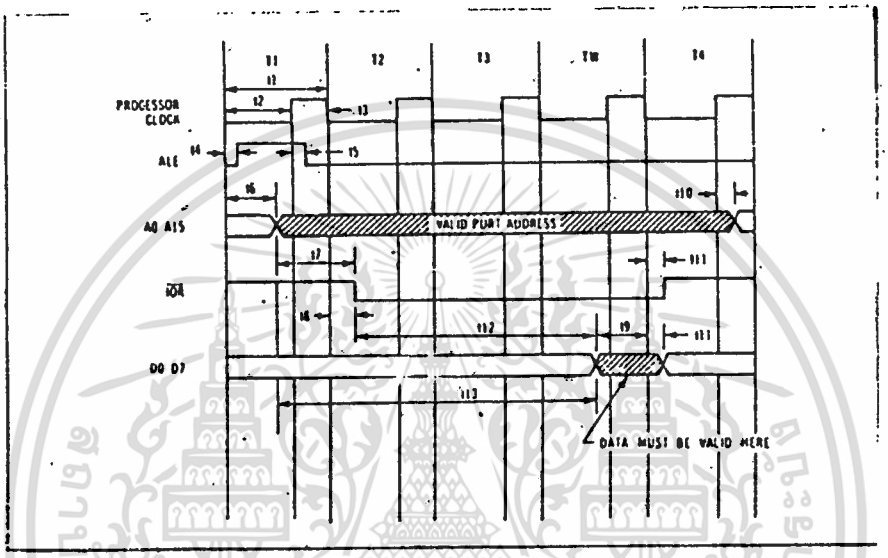
สัญญาณ ~IOR และ ~IOW

สองสัญญาณนี้จะ ACTIVE LOW เป็นสัญญาณที่จะบอกว่าเป็นการติดต่อกับอุปกรณ์ I/O ในการ DECODE PORT นี้ จะนำเอาสัญญาณทั้งสองนี้มา AND กันแล้วป้อนเข้าร่วมกับสัญญาณอื่น ๆ ที่จำเป็น

TIMING DIAGRAM

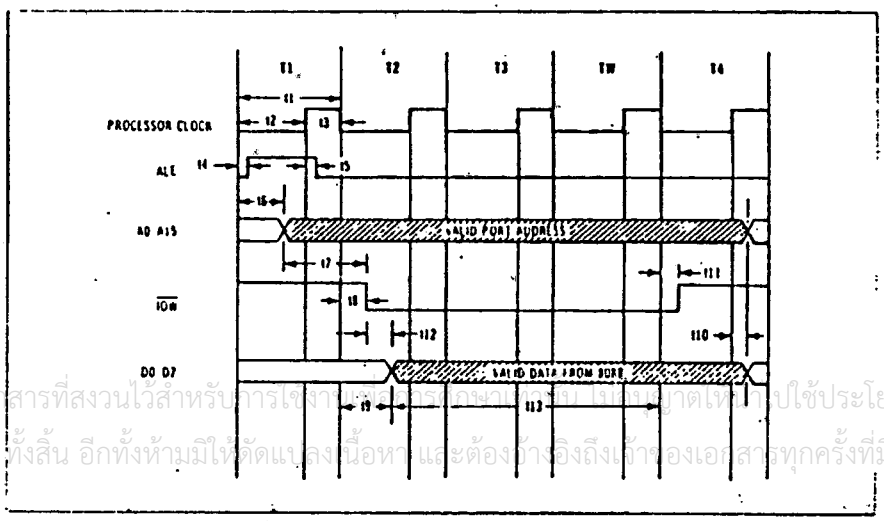
การ READ PORT

จาก TIMING DIAGRAM เมื่อมีการ READ PORT สัญญาณ ALE จะ ACTIVE เพื่อที่จะบอกอุปกรณ์ต่าง ๆ ว่า สัญญาณที่ขา AD0-AD7 นั้น เป็นสัญญาณ ADDRESS ให้อุปกรณ์ภายนอก LATCH สัญญาณนี้ไว้ จากนั้นสัญญาณ ~IOR ก็จะ ACTIVE เพื่อทำการอ่านข้อมูลจาก PORT INPUT จากนั้นสัญญาณ DATA ก็จะถูกส่งออกมา ข้อมูลก็จะถูกอ่านไปเก็บไว้ เป็นการสิ้นสุดกระบวนการ READ PORT

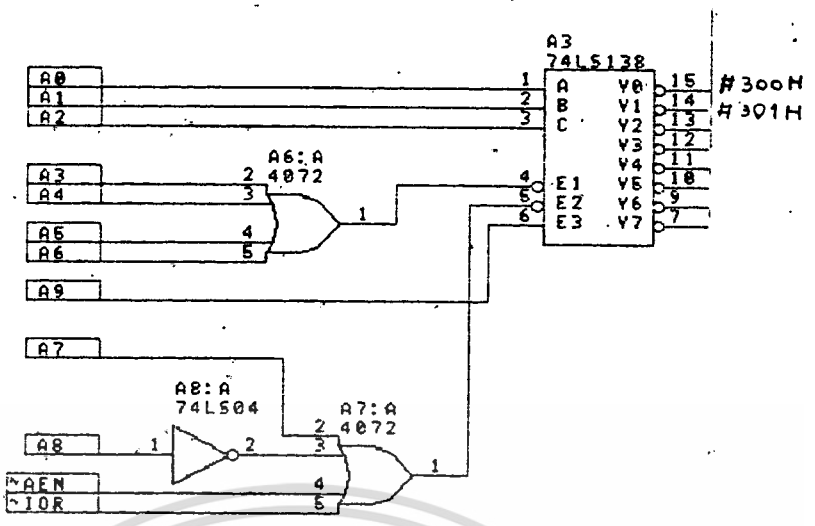


การ WRITE PORT

เมื่อมีการ WRITE PORT สัญญาณ ALE จะ ACTIVE เพื่อที่จะบอกอุปกรณ์ต่างๆ ว่า สัญญาณที่ขา AD0-AD7 นั้น เป็นสัญญาณ ADDRESS ให้อุปกรณ์ภายนอก LATCH สัญญาณนี้ไว้ จากนั้นสัญญาณ ~IOW ก็จะ ACTIVE เพื่อทำการเขียนข้อมูลจาก PORT OUTPUT จากนั้นสัญญาณ DATA ก็จะถูกส่งออกมาจากขา AD0 AD7 ข้อมูลก็จะถูกเขียนไปยัง PORT เป็นการสิ้นสุดกระบวนการ WRITE PORT



จาก TIMING DIAGRAM และสัญญาณต่าง ๆ สามารถต่อเป็นวงจร DECODE PORT 300H และ PORT 301H ได้ดังนี้



สำหรับการทำงานและ FUNCTION การทำงานของ 74LS138 3 TO 8 LINE DECODER นั้น สามารถดูได้จาก DATA SHEET ที่ภาคผนวกด้านหลังของปริญญานิพนธ์ฉบับนี้

วงจรเอทูดีคอนเวอร์ทเตอร์

วงจรเปลี่ยนสัญญาณอะนาลอกเป็นดิจิตอล (analog to digital converter) ที่ใช้กันอยู่ทั่วๆไป มีหลายแบบคือ

แบบใช้วงจรเปรียบเทียบขนานหรือแบบ "แฟลช" (Parallel Comparator Simultaneous Flash A/D Converter)

วงจรเอทูดีแบบนี้ใช้หลักการง่ายๆ อีกทั้งยังเป็นวิธีที่รวดเร็วที่สุด คือใช้วงจรเปรียบเทียบที่ต่อขนานกันดังรูปที่1ก ประกอบด้วยออปแอมป์ที่ต่อเป็นวงจรเปรียบเทียบและตัวต้านทานต่อไว้เพื่อแบ่งแรงดันที่ขาอินพุตแบบกลับ (inverting) ให้มีขนาดต่างๆ กัน

จากหลักการของวงจรเปรียบเทียบทั่วไป เมื่อแรงดันอินพุตที่ขาอินพุตแบบไม่กลับ(noninverting) มีค่าสูงกว่าที่ขาอินพุตแบบกลับ (inverting) เอาต์พุตจะได้แรงดันค่าสูง ดูได้จากตาราง จะเข้าใจยิ่งขึ้นว่าที่แรงดันค่าต่างๆ มีผลต่อเอาต์พุตของวงจรเปรียบเทียบแต่ละตัวอย่างไร ซึ่งเอาต์พุตที่ได้จากวงจรเปรียบเทียบนี้จะนำไปเข้ารหัสให้เป็นเลขฐานสองต่อไปจำนวนของวงจรเปรียบเทียบที่ต้องใช้ในวงจรขึ้นอยู่กับขนาดของสัญญาณอะนาลอกที่อินพุต

จากวงจรรูปที่1ถ้าแรงดันอินพุตมีค่า1โวลต์ ไม่เพียงพอที่จะทำให้วงจรเปรียบเทียบตัวใดให้ค่าเอาต์พุตเป็น "high"

ที่แรงดันระหว่าง 1 ถึง 2 โวลต์ วงจรเปรียบเทียบที่มีระดับเทรชโฮลด์* (threshold) ต่ำสุดก็จะให้เอาต์พุตเป็น "high"

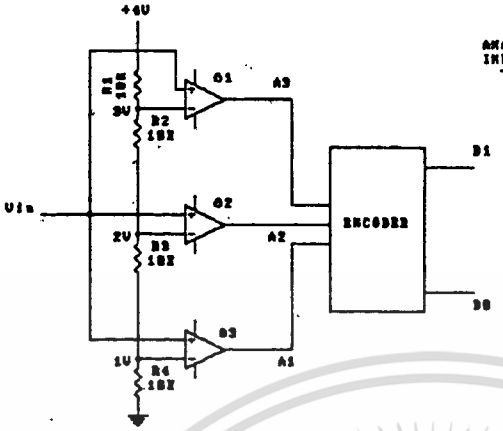
แรงดัน 2 - 3 โวลต์ วงจรเปรียบเทียบทั้ง A1 และ A2 ให้เอาต์พุตเป็น "high" ถ้าแรงดันอินพุตมากกว่า 3 โวลต์ วงจรเปรียบเทียบก็จะให้เอาต์พุตเป็น "high" ทั้งหมด

เมื่อต้องการวงจรที่มีความละเอียดสูงขึ้น จำเป็นต้องใช้วงจรเปรียบเทียบเพิ่มขึ้น เช่น ถ้าต้องการความละเอียด 3 บิต ต้องใช้วงจรเปรียบเทียบ 7 ตัว ความละเอียด 4 บิต ต้องใช้วงจรเปรียบเทียบ 15 ตัว (16 ระดับ) โดยหาจำนวนวงจรเปรียบเทียบได้จาก $2^N - 1$ เมื่อ N แทนจำนวนบิตหรือความละเอียดที่ต้องการ

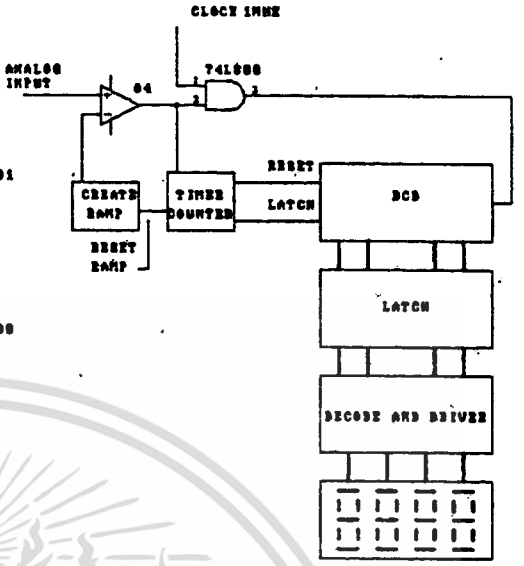
จะเห็นได้ว่าที่ความละเอียด 8 บิตต้องใช้วงจรเปรียบเทียบมากถึง 255 ตัวซึ่งเป็นข้อเสียของวงจรเอทูดีแบบนี้

ข้อเสียอีกประการหนึ่งคือ เอาต์พุตที่ได้ไม่เป็นเลขฐานสอง ต้องมีวงจรเพิ่มเติมไปทำการเข้ารหัส

ข้อดีของวงจรเอทูดีแบบขนานนี้คือ ความเร็วสูงมาก บางครั้งจึงเรียกวงจรเอทูดีแบบนี้ว่า"แฟลช" (flash type A/D converter) วงจรเอทูดิชนิดนี้ใช้เวลาในการแปลงได้เร็วใน ระดับนาโนวินาที



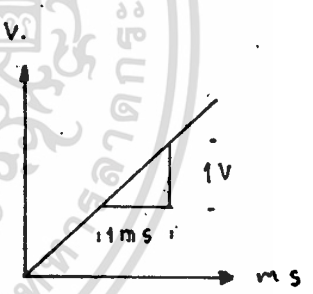
รูปที่ 1ก



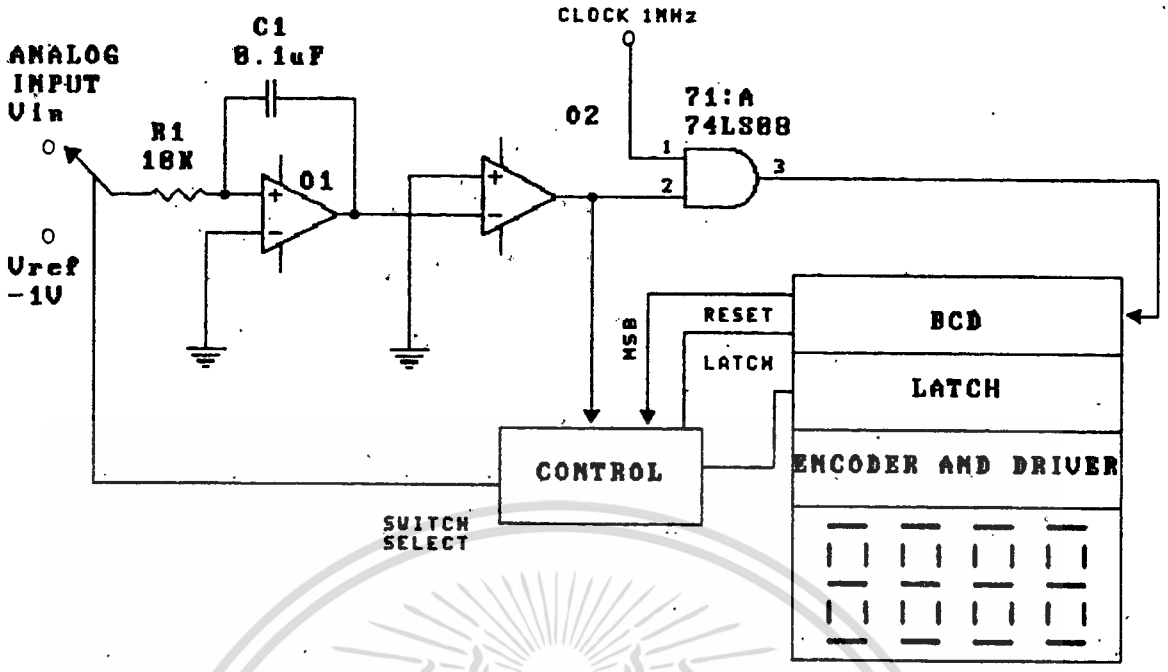
รูปที่ 2ก

แรงดันอินพุต (โวลท์)	o/p ของ COMPARATOR			o/p	
	A_1	A_2	A_3	D_0	D_1
0 - 1	0	0	0	0	0
1 - 2	1	0	0	0	1
2 - 3	1	1	0	1	0
3 - 4	1	1	1	1	1

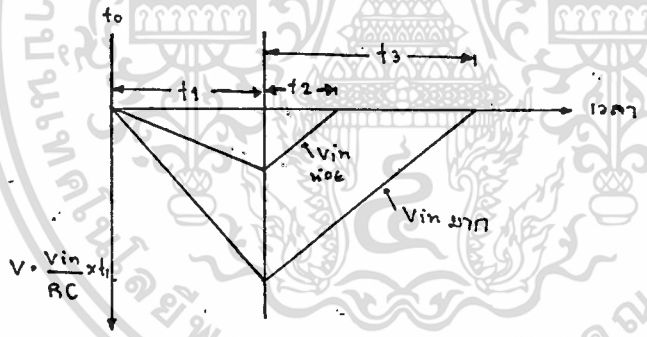
รูปที่ 1ก



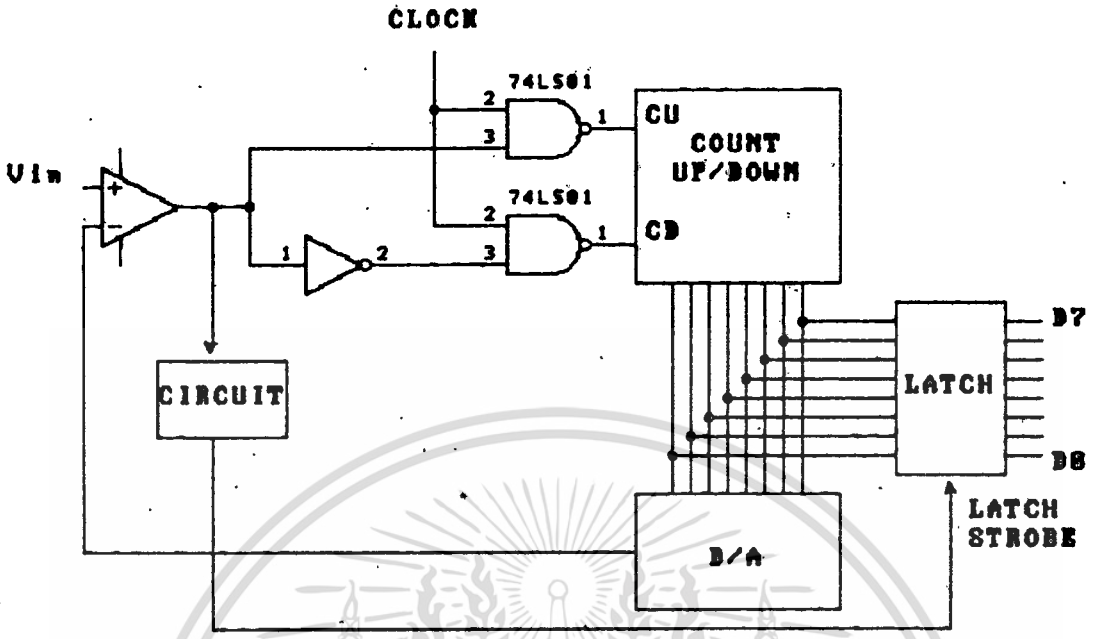
รูปที่ 2ข



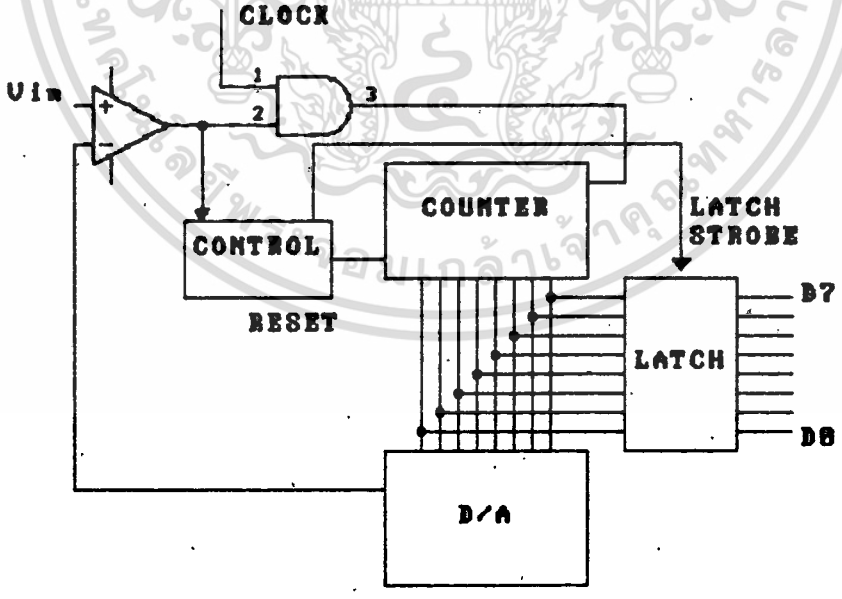
รูปที่3ก



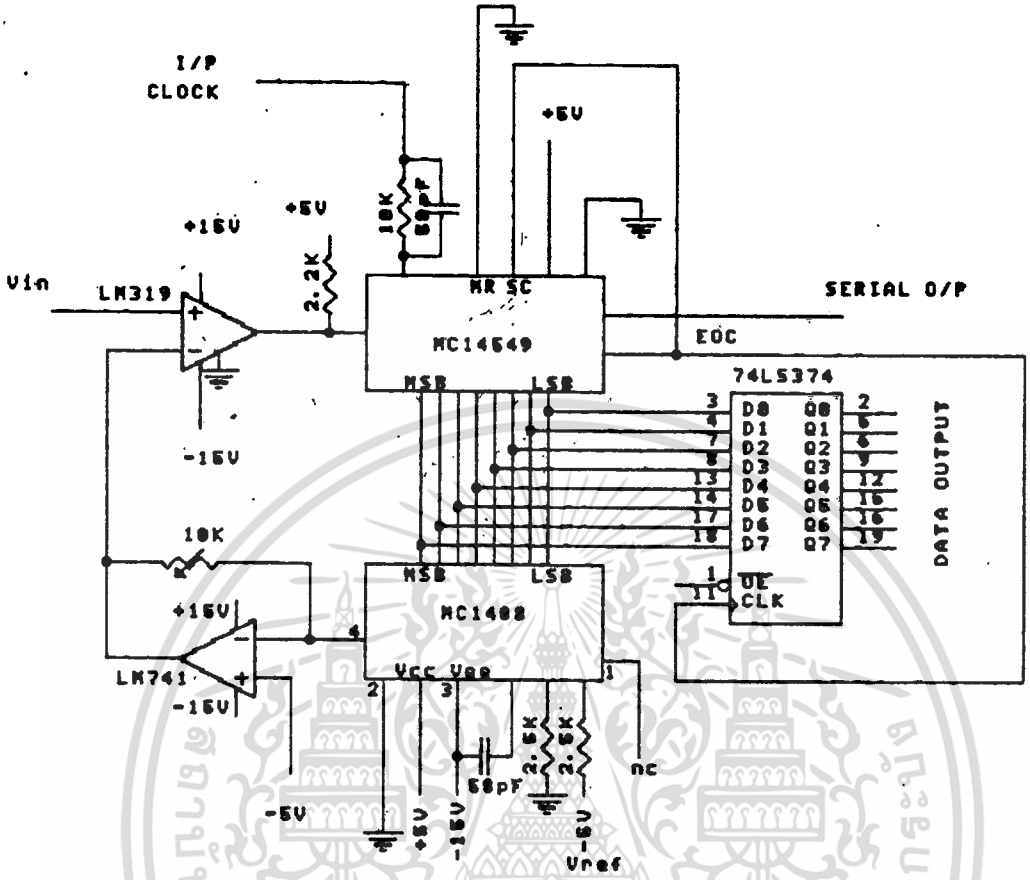
รูปที่3ข



รูปที่5



รูปที่6



รูปที่7

วงจรเอทูดีที่ใช้การอินทีเกรต

วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้เทคนิคการอินทีเกรตสัญญาณมี 3 แบบ คือแบบสโลปเดี่ยวหรือแบบแรมป์ (Single Ramp หรือ Single Slope A/D Converter)

วงจรเอทูดีแบบนี้แสดงไว้ดังรูปที่ 2 ประกอบด้วยวงจรกำเนิดสัญญาณแรมป์ , วงจรเปรียบเทียบ, วงจรนับบีซีดี(BCD) หรือ นับเลขฐานสอง

เมื่อเริ่มทำการเปลี่ยนสัญญาณ สัญญาณแรมป์และวงจรนับจะถูกรีเซตให้เป็น 0 แรงดันอะนาลอกถูกป้อนไปยังวงจรเปรียบเทียบทางขาอินพุตแบบไม่กลับ เมื่อแรงดันอินพุตที่ขานี้เป็นบวกมากกว่าที่ขาอินพุตแบบกลับ วงจรเปรียบเทียบก็จะให้เอาต์พุตเป็นระดับ "high" ทำให้แอนด์เกตปล่อยสัญญาณนาฬิกาผ่านไปยังวงจรนับได้ และทำให้เริ่มเกิดสัญญาณแรมป์

สัญญาณแรมป์มีแรงดันเป็นบวกขึ้นเรื่อย ๆ จนมากกว่าระดับแรงดันอินพุตเอาต์พุตของวงจรเปรียบเทียบก็ตกลงมาเป็นระดับ "low" ปิดแอนด์เกตไม่มีสัญญาณผ่านไปให้วงจรนับ

วงจรนับจะหยุดนับและเก็บค่าไว้ที่วงจรแลตช์ จากนั้นจึงทำการรีเซตวงจรนับและวงจรกำเนิดสัญญาณแรมป์

สมมติให้สัญญาณนาฬิกามีความถี่ 1 MHZ, วงจรนับบีซีดี 4 หลัก, แรงดันอินพุต Vin 2 โวลต์, สัญญาณแรมป์มีความชัน 1 V/ms ดังแสดงในรูปที่ 2.ข

จากจุดเริ่มต้นจนถึงแรงดันสูงสุด (2โวลต์) สัญญาณแรมป์ใช้เวลา 2 ms หลังจากนี้จึงปิด สัญญาณนาฬิกาที่ส่งไปให้วงจรนับ

ในช่วง 2ms นี้มีการส่งพัลส์ไปให้วงจรนับทำการนับถึง 200 ลูก เอาต์พุตของวงจรเปรียบเทียบที่มีระดับ "high" เป็นการสั่งสัญญาณให้วงจรแลตช์ส่งค่าที่นับได้ไปยังภาคแสดงผล และเติมจุดทศนิยมที่ตำแหน่งที่เหมาะสมของตัวแสดงผลได้เป็นค่า 2,000 ที่แรงดันอินพุต 2 โวลต์

วงจรแบบนี้เป็นหลักการเบื้องต้นของดิจิตอลโวลต์มิเตอร์ ซึ่งถ้าใช้วงจรนับเลขฐานสองแทนแบบบีซีดี เอาต์พุตก็จะอ่านได้ค่าเลขฐานสองโดยตรง

วงจรลักษณะนี้มักนำไปใช้งานในการเปลี่ยนเวลาเป็นขนาดของสัญญาณ (time to amplitude conversion)หรืออาจใช้ในดิจิตอลโวลต์มิเตอร์ แต่ไม่ใช้กับงานที่ต้องการความถูกต้องสูง เนื่องจากการเปลี่ยนแปลงในแหล่งกำเนิดสัญญาณแรมป์ขึ้นกับอุณหภูมิและผลตอบสนองต่อสัญญาณอินพุต ทำให้ไม่มีความคงที่ ดังนั้นจึงมีการปรับปรุงให้ดีขึ้นกลายเป็นแบบสโลปคู่ (dual - slope)

แบบสโลปคู่ (Dual - Slope A/D converters)

จากรูปที่ 3.ก แสดงบล็อกไดอะแกรมของวงจรเอทูดีแบบสโลปคู่ ซึ่งวงจรส่วนใหญ่คล้ายกับแบบ

สโลปเดี่ยว แต่มีสวิตช์ที่อินพุตเพิ่มขึ้นเพื่อทำการเลือกระหว่างแรงดันอินพุตกับแรงดันอ้างอิง(วงจรเปรียบเทียบต่อขาสัญญาณอินพุตกลับกันกับแบบสโลปเดี่ยว)

ส่วนแรกของวงจรคือ วงจรกำเนิดสัญญาณแรมป์หรือวงจรอินทีเกรเตอร์นั่นเอง ที่อินพุตแบบกลับของออปแอมป์มีสภาพเป็นกราวด์เทียม (virtual ground)ถ้ามีแรงดันอินพุต 2 โวลต์จะได้กระแสไหลผ่านตัวต้านทาน 10k เท่ากับ 0.2 mA ไปยังจุดรวม (summing point) เนื่องจากค่าความต้านทานอินพุตของออปแอมป์นั้นสูงมาก กระแสที่ไหลจึงเกิดขึ้นผ่านตัวเก็บประจุ

ขณะที่ตัวเก็บประจุทำการชาร์จ (รับประจุ) แรงดันที่เอาต์พุตของออปแอมป์ก็จะยิ่งเป็นลบมากขึ้นเรื่อยๆ เพื่อรักษาระดับกระแสให้คงที่ แรงดันคร่อมตัวเก็บประจุจึงได้เป็นสัญญาณแรมป์ที่เป็นเชิงเส้น (linear ramp)

ถ้าแรงดันอินพุตเป็นบวก วงจรอินทีเกรเตอร์จะให้เอาต์พุตเป็นสัญญาณแรมป์ทางลบดังแสดงไว้ในช่วง t1 ดังรูปที่ 3ข. หากแรงดันอินพุตเป็นลบก็จะทำให้เอาต์พุตได้แรมป์ทางบวก

ความชันของสัญญาณแรมป์ สามารถคำนวณได้จากความสัมพันธ์ของประจุ q = cv และ q = It โดยจับสองสมการมาเท่ากัน

$$\frac{V}{t} = \frac{I}{C}$$

เมื่อรู้ว่ากระแสเท่ากับ $\frac{Vin}{R}$ เราก็รู้ว่า

$$\frac{V}{t} = \frac{Vin}{RC}$$

จากรูปให้แรงดันอินพุต +2 โวลต์ ก็จะได้ความชันของสัญญาณแรมป์ทางเอาต์พุตเท่ากับ -2 V/ms

จากวงจรในรูปที่ 3 อธิบายได้คือ เมื่อสวิตช์ต่อกับสัญญาณอินพุตจะทำให้มีแรงดันบวกจากอินพุตป้อนเข้าสู่วงจรอินทีเกรเตอร์ ได้เอาต์พุตออกมาเป็นแรมป์ทางลบ

วงจรเปรียบเทียบก็จะได้แรงดันลบจากวงจรอินทีเกรเตอร์ แล้วให้เอาต์พุตเป็นบวกทำการเปิดแฮนด์เกตให้สัญญาณนาฬิกาผ่านเข้าไปสู่วงจรนับ วงจรนับจะนับไปยังค่าที่กำหนดไว้คงที่ (t1)แล้วทำการสับสวิตช์ต่อเข้ากันกับแรงดันอ้างอิง

ในช่วงที่วงจรนับด้วยค่าคงที่นั้น วงจรอินทีเกรตจะให้สัญญาณแรมป์ทางลบที่มีค่าไต่สูงสุดตามแต่ระดับแรงดันอินพุต เมื่อทำการสวิตช์อินพุตของวงจรอินทีเกรเตอร์ให้ไปที่แรงดันอ้างอิงค่าลบ เอาต์พุตของวงจรจึงได้เป็นแรมป์ทางบวกคือ ช่วง t2 รูปที่ 3ข. พร้อมๆ กับรีเซตค่าของวงจรนับลงเป็นศูนย์เพื่อเริ่มนับใหม่

เมื่อเอาต์พุตของวงจรอินทีเกรเตอร์เพิ่มขึ้นถึงแรงดัน 0 อีกครั้ง เอาต์พุตของวงจรเปรียบเทียบก็จะเป็นลบ(หรือเป็นศูนย์) วงจรควบคุมจับการเปลี่ยนแปลงอันนี้ได้ก็ส่งสัญญาณสโตรบ (STROBE) ให้วงจรนับเก็บค่าที่ได้ไว้ที่วงจรแลตช์ จากนั้นจึงรีเซตให้เป็นศูนย์แล้วทำการสวิตช์ให้อินพุตของวงจรอินทีเกรเตอร์ต่อกับแรงดันอินพุตเป็นการเริ่มทำการเปลี่ยนสัญญาณอีกรอบหนึ่งจำนวนที่นับได้ที่เก็บไว้ในวงจรแลตช์ก็จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุต Vin

สัญญาณแรมป์ทางเอาต์พุตของวงจรอินทีเกรเตอร์ในช่วงเวลาคงที่ t1 จะลดลงสู่แรงดัน V ซึ่ง

$$V = \frac{(Vin \times t1)}{RC}$$

เพื่อให้กลับไปสู่ระดับ 0 หน้าที่ของวงจรอินทีเกรเตอร์จึงต้องสร้างแรมป์ทั้งทางบวกทางลบให้เพิ่มขึ้นเท่าๆ กันในช่วงเวลา t2 (ที่เกิดจากแรงดันอินพุตอ้างอิง) แรงดัน V เท่ากับ

$$V = \frac{(VREF \times t2)}{RC}$$

สูตรทั้ง 2 ของ V สามารถจับมาเท่ากันได้เป็น

$$\frac{Vin}{RC} \times t1 = \frac{VREF}{RC} \times t2$$

$$Vin \times t1 = VREF \times t2$$

$$t2 = Vin \times \frac{t1}{VREF}$$

เห็นได้ว่าRC ปรากฏอยู่ทั้ง 2 ข้างของสมการ จึงสามารถตัดทิ้งได้หมายถึงว่าเมื่อช่วงเวลาอินทีเกรตสัญญาณและช่วงเวลาอินทีเกรตอ้างอิง ใช้ตัวต้านทานและตัวเก็บประจุค่าเดียวกัน การเปลี่ยนแปลงของค่าทั้งสองนี้ก็จะไม่มีผลต่อความถูกต้องของสัญญาณเอาต์พุต ซึ่งเป็นข้อดีที่เหนือ กว่าแบบสโลปเดียว คือค่าที่ได้ไม่ขึ้นกับความถี่ของรอบการทำงานสมการท้ายสุดแสดงให้เห็นว่า เอาต์พุตของวงจรนับในช่วงเวลา t2 เป็นสัดส่วนโดยตรงกับแรงดันอินพุต Vin เมื่อ Vref และ t1 คงที่

จากวงจรในรูปที่ 3 t1 เท่ากับ 1000 รอบ เมื่อป้อนสัญญาณนาฬิกา 1 MHZ

(= $\frac{1}{1000}$ = 1 ms) และ VREF มีค่า -1 โวลต์

ถ้าสัญญาณอินพุตมีขนาด 2 โวลต์ จะได้ช่วงเวลา t2 = $\frac{2V \times 1000}{1V}$ = 2000 รอบ

(รอบของการนับ) จุดทศนิยมที่อยู่ทางขวาทำให้ได้ผลลัพธ์ที่ภาคแสดงผล 2.000

กราฟในรูป 3ข. แสดงว่าเมื่อสัญญาณอินพุตน้อยกว่านี้จะมีการเปลี่ยนแปลงอย่างไรบ้าง

เช่นอินพุต 0.88 โวลต์ t2 จะได้ $\frac{0.8\ V}{1\ V} \times 1000$ เท่ากับ 800 รอบ

ก็จะอ่านค่าได้ 0.800 หลักการเช่นนี้ถูกนำไปใช้อย่างแพร่หลายในดิจิตอลโวลต์มิเตอร์และเครื่องมืออื่นๆ อีกหลายชนิด

สรุปอีกครั้งได้ว่า แรงดันอินพุตที่ไม่รู้ค่าถูกป้อนเข้าไปในวงจรอินทีเกรเตอร์ เมื่อครบช่วง

เวลา t1 วงจรนับจึงถูกรีเซตให้เป็น0 อินพุตของวงจรอินทีเกรเตอร์ก็จะถูกสวิตช์ต่อกลับมาที่แรงดันอ้างอิง(ที่มีแรงดันคงที่)ให้ความชันของสัญญาณแรมป์คงที่เพิ่มค่าขึ้นไปจนถึงระดับ 0 ช่วงเวลา t2 นี้เป็นสัดส่วนโดยตรงกับสัญญาณอินพุต ถ้าดูรูป 3ข. อีกครั้งพิจารณาช่วง t1 ซึ่งเป็นช่วงเวลางที่ และ t2 ซึ่งความชันคงที่แล้วจะเข้าใจยิ่งขึ้น

ข้อดีของวงจรเปลี่ยนสัญญาณแบบสโลปคู่นี้คือ ความถูกต้องสูง, ราคาถูก, เสถียรภาพทางด้านอุณหภูมิดี ข้อเสียคือความเร็วต่ำ ในการเปลี่ยนสัญญาณ 1 ครั้งอาจใช้เวลาถึง 100 ms (ในขณะที่แบบ "แฟลช" ใช้เวลาประมาณ 30 ns)

แบบชาร์จบาลานซ์ (Charge Balance A/D Converters)

วงจรเปลี่ยนสัญญาณเอทูดีแบบชาร์จบาลานซ์ใช้วงจรสำคัญคล้ายกับแบบสโลปคู่นั่นเองแต่แทนที่จะให้อินพุตสวิตช์ไปมาระหว่างแรงดันที่ไม่รู้ค่ากับแรงดันอ้างอิง ก็ทำการแทรกพลัส์ของกระแสอ้างอิงมาตรงๆ ที่จุดรวมของวงจรอินทีเกรเตอร์ในช่วงเวลาที่คงที่ โดยที่จำนวนของพัลส์จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุตที่ไม่รู้ค่า

ประโยชน์ของเทคนิคนี้คือ แรงดันตกคร่อมตัวเก็บประจุของวงจรอินทีเกรเตอร์จะมีค่าใกล้เคียง 0V ดังนั้นจึงไม่เกิดความผิดพลาดจากผลของกระแสรั่วไหล เอทูดีชนิดนี้จึงมีความถูกต้องสูงกว่าแบบสโลปคู่

แบบเดลต้า - ซิกม่า (Delta - Sigma A/D Converters)

จากวงจรรูปที่ 4 เมื่อมีแรงดันอินพุตป้อนเข้าไปที่วงจรอินทีเกรเตอร์ จะให้เอาต์พุตไปเข้าวงจรเปรียบเทียบ เปรียบเทียบกับแรงดันคงที่ (จากรูปคือ กราวด์) พัลส์ของกระแสที่ได้ขึ้นอยู่กับเอาต์พุตของวงจรเปรียบเทียบ โดยสวิตช์ที่ทำขึ้นจากเฟตจะควบคุมให้กระแสเข้าไปยังที่จุดรวมหรือลงกราวด์ไป ส่วนวงจรนับจะนับจำนวนพัลส์ด้วยหลักการที่คล้ายกัน

ข้อสรุปของเอทูดีแบบอินทีเกรตสัญญาณ

จุดสำคัญของอินทีเกรติ้งเทคนิคคือ อินพุตที่ให้กับวงจรอินทีเกรเตอร์ต้องเป็นกระแส ไอซีคอนเวอร์เตอร์บางตัวอาจมีอินพุตให้สองขา แต่จะมีขาหนึ่งต่อตรงกับจุดรวมโดยใช้กับอุปกรณ์ที่เป็นแหล่งจ่ายกระแสโดยตรง

ถ้าให้อินพุตเป็นกระแสก็ไม่ต้องคำนึงถึงแรงดันออฟเซตของวงจรอินทีเกรเตอร์ แต่ถ้าหากใช้กับอินพุตที่เป็นแรงดัน(ที่ต้องมีตัวต้านทานต่ออนุกรมเพื่อให้ได้เป็นกระแส) ต้องปรับออฟเซตของออปแอมป์เสียก่อน

การใช้อินพุตเป็นกระแสทำให้ย่านการใช้งานทางไฟสลับกว้าง

ไอซีแบบชาร์จ - บาลานซ์มักประกอบด้วยวงจรแปลงแรงดันเป็นความถี่อยู่ด้วย ดังนั้นถ้าหากต้องการเอาต์พุตเป็นความถี่ก็สามารถเลือกได้

วงจรเปลี่ยนสัญญาณเอทูดี ที่ใช้วงจรนับและวงจรดีทูเอประกอบกัน

แบบวงจรนับเดี่ยว (Single - Counter)

แท้ที่จริงแล้วสัญญาณแรมป์เชิงเส้น (linear ramp) อาจประกอบขึ้นด้วยสัญญาณขั้นบันไดเล็ก ๆ จำนวนมากที่เกิดจากการต่อเอาต์พุตของวงจรนับเข้ากับวงจรแปลงดีทูเอ โดยขนาดของขั้นบันไดแต่ละขั้นขึ้นอยู่กับจำนวนบิตหรือความละเอียดของวงจรดีทูเอนั้นๆ

รูปที่ 5 แสดงการกำเนิดสัญญาณแรมป์เดี่ยวด้วยวงจรนับ และวงจรดีทูเอ (แทนวงจรอินทีเกรเตอร์) เมื่อเริ่มแปลงสัญญาณวงจรนับจะถูกรีเซต เอาต์พุตของวงจรดีทูเอมีระดับ 0 เมื่อแรงดันถูกป้อนเข้าไปยังอินพุตของวงจรเปรียบเทียบ เอาต์พุตก็จะขึ้นสู่ระดับ "high" และเปิดสัญญาณนาฬิกาไปสู่วงจรนับ แต่ละพัลส์ของสัญญาณนาฬิกา ทำให้เกิดการนับและเพิ่มแรงดันขึ้น 1 ขั้น เมื่อเอาต์พุตของดีทูเอมีค่ามากกว่าอินพุต Vin เอาต์พุตของวงจรเปรียบเทียบก็จะกลายเป็น "low" ทำให้สัญญาณนาฬิกาไม่อาจผ่านไปยังวงจรนับได้ ดังนั้นวงจรควบคุมจะทำการแลตช์เอาต์พุตของวงจรนับและรีเซตวงจรนับให้เริ่มต้นรอบใหม่อีกครั้งหนึ่ง

แบบแทร็กกิ้ง (Tracking A/D Converter)

การทำงานจะคล้ายกับแบบใช้วงจรนับเดี่ยว แต่การนับจะไม่ได้เริ่มจากศูนย์แต่จะทำการนับขึ้น หรือนับลงจากค่าล่าสุดไปยังค่าใหม่ แล้วแต่ว่าแรงดันอินพุตในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้ว

ข้อดีของเอทูดีแบบแทร็กกิ้งคือ ทำงานได้เร็วขึ้น

วงจรเปลี่ยนสัญญาณเอทูดีแบบใช้การประมาณค่า (Successive Approximation A/D Con - verter)

วงจรเอทูดีแบบนี้มีข้อได้เปรียบทางด้านความละเอียด เพราะความละเอียด n บิตสามารถกำหนดได้จากสัญญาณนาฬิกา N ลูก ตัวอย่างเช่น วงจรแปลงขนาด 8 บิต ต้องการพัลส์ของสัญญาณนาฬิกาเพียง 8 ลูก ในขณะที่แบบใช้วงจรนับต้องใช้พัลส์ถึง 256 ลูก วงจร SA (Successive - Approximation) นี้แสดงไว้ดังรูปที่ 7 หัวใจของวงจรคือ successive approximation - register (SAR) เช่นเบอร์ MC14549 ที่มีการทำงานดังต่อไปนี้

เมื่อเริ่มทำการเปลี่ยนสัญญาณ พัลส์ลูกแรกจะทำการส่งบิตที่มีนัยสำคัญสูงสุดไปยังดีทูเอเบอร์ MC1408โดย SAR จะรอสัญญาณจากวงจรเปรียบเทียบ LM319 ซึ่งทำการตรวจสอบว่าเอาต์พุตของวง

จรดีทูเอมากกว่าหรือน้อยกว่าแรงดันอินพุต Vin ถ้าเอาต์พุตของวงจรเปรียลเทียบมีระดับ "high" เอาต์พุตของดีทูเอจิงต่ำกว่า Vin - SAR จะทำการเก็บบิตที่มีนัยสำคัญสูงสุดไว้ ถ้าเอาต์พุตของวงจรเปรียบเทียบเป็นระดับ "low" เอาต์พุตของดีทูเอจิงมากกว่า Vin SAR จะทำการรีเซตบิตที่มีนัยสำคัญสูงสุดนั้น

พัลส์ลูกต่อมาก็จะทำงานเช่นเดียวกัน โดยบิตที่ได้คือ บิตที่มีนัยสำคัญรองลงมา SAR ทำงานแบบนี้ไปจนถึงบิตที่มีนัยสำคัญต่ำสุด แต่ละบิตใช้สัญญาณนาฬิกาเพียงลูกเดียวครบทุกบิตแล้ว SAR ก็ทำการส่งสัญญาณ EOC (end of conversion) ออกไป

สัญญาณ EOC เป็นตัวบอกว่าสายสัญญาณเอาต์พุตที่ขนานกันมาทุกเส้นมีข้อมูลดิจิตอลของสัญญาณอินพุตครบถ้วนแล้ว ถ้าสัญญาณ EOC ถูกต่อไปยังอินพุตที่เป็นจุดเริ่มการเปลี่ยนสัญญาณการเปลี่ยนสัญญาณก็จะเกิดขึ้นอย่างต่อเนื่อง

MC14549 ยังสามารถส่งเอาต์พุตดิจิตอลแบบขนานกรมได้อีกด้วย ขึ้นอยู่กับการควบคุมของSAR

วงจรในรูปที่ 7 ใช้แรงดันอินพุตสูงสุด $\pm$5 โวลต์ อินพุตแบบไม่กลับของออปแอมป์ที่ทำหน้าที่เปลี่ยนกระแสเป็นแรงดัน (current to voltage converter) ต่อเข้ากับ -5 โวลต์ แทนที่จะต่อลงกราวด์เป็นการยกระดับแรงดันอะนาลอกจาก -5 โวลต์ถึง +5 โวลต์แทน 0 ถึง 10โวลต์ สัญญาณไฟสลับรูปชายน์จึงสามารถต่อโดยตรงเข้ากับอินพุตของเอทูดีวงจรนี้ได้

วงจรแปลงเอทูดีชนิดนี้มีความเร็วสูง และความละเอียดสูง จึงเป็นวงจรที่นำมาใช้กันอย่างแพร่หลาย

การสุ่มและการคงค่า (Sample and Holds)

วงจรเอทูดีต้องการเวลาในการแปลงสัญญาณ หรือที่เรียกว่า conversion timeเพื่อเปลี่ยนสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลที่เหมาะสม ถ้าสัญญาณอะนาลอกมีการเปลี่ยนแปลงในระหว่างช่วงเวลาการแปลง เอาต์พุตของวงจรเปลี่ยนสัญญาณอาจเกิดความผิดพลาด จึงต้องมีการป้องกันด้วยวงจรสุ่มและคงค่าสัญญาณ (sample and hold) เพื่อใช้จับสัญญาณอะนาลอกที่จุดเริ่มของการเปลี่ยนสัญญาณ แล้วเก็บไว้ในตัวเก็บประจุระหว่างช่วงเวลาการแปลง หลังจากที่การเปลี่ยนสัญญาณเสร็จสิ้น จึงจับสัญญาณอะนาลอกค่าใหม่มาเก็บไว้อีกครั้งเป็นเช่นนี้ต่อๆ ไปเราจึงมักได้เห็นวงจรสุ่มและคงค่าสัญญาณปรากฏคู่กับวงจรเอทูดีอยู่เสมอ

วงจร ดีทูเอคอนเวอร์ทเตอร์

แบบใช้ตัวต้านทานหลายค่า (Binary Weighted Resistor D/A converter)

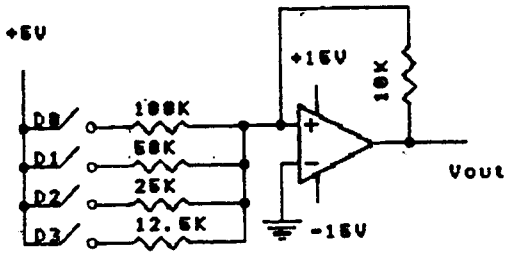
วงจรเปลี่ยนสัญญาณดีทูเอชนิดนี้ใช้ตัวต้านทานต่างๆ และออปแอมป์เพื่อเปลี่ยนระดับสัญญาณลอจิก 2 ระดับเป็นแรงดันที่ได้สัดส่วนกัน รูปที่ 1 แสดงวงจรเปลี่ยนสัญญาณดิจิตอลขนาด 4 บิต ออปแอมป์ที่ใช้มีอัตราขยายสูงมาก(โดยทั่วไปจะสูงกว่า 1000,000 เท่า) มีความต้านทานเอาต์พุตต่ำ ความต้านทานด้านอินพุตมีค่าสูงมาก สิ่งสำคัญที่สุดที่จะต้องตระหนักไว้ก็คือ สัญญาณที่เอาต์พุตถูกป้อนกลับมายังอินพุตแบบกลับเฟส (การป้อนกลับแบบลบ) เพื่อเปรียบเทียบกับสัญญาณที่ขาอินพุตแบบไม่กลับเฟส เอาต์พุตของออปแอมป์จะเป็นตัวจ่ายหรือรับกระแส (Source or sink)เพื่อให้แรงดันที่เปรียบเทียบกันนั้นมีค่าเดียวกันวงจรในรูปที่ 1 ต่อขาไม่กลับเฟสลงกราวนด์ ดังนั้นที่ขากลับเฟสก็จะมีแรงดัน 0 โวลต์ด้วย การที่อินพุตที่ขากลับเฟสเป็น 0 โวลต์ด้วยโดยไม่ได้ต่อลงกราวนด์โดยตรงจึงถูกเรียกว่า กราวด์เทียม

มาดูกันตอนที่สวิตช์ D0 ปิด ตัวต้านทาน R1 ค่า 100 กิโลโอห์มจะมีแรงดัน 5โวลต์ ที่ปลายข้างหนึ่งอีกข้างหนึ่งเป็น 0 โวลต์(กราวด์เทียม)จากกฎของโอห์มจะมีแรงดันตกคร่อม 5โวลต์ซึ่งให้กระแสไหลผ่าน 0.05 มิลลิแอมป์ กระแสนี้ไม่อาจเข้าไปยังอินพุตของออปแอมป์ได้ เนื่องจากออปแอมป์มีความต้านทานอินพุตสูงมาก และไม่สามารถส่งหรือรับกระแสมากๆ ได้ ดังนั้นกระแส 0.05 มิลลิแอมป์ จึงต้องไหลผ่านไปยังเอาต์พุต โดยผ่านตัวต้านทานป้อนกลับ Rf 10 กิโลโอห์มจะได้แรงดันเอาต์พุตเท่ากับ (10 กิโลโอห์ม) x (-0.05 มิลลิแอมป์) = -0.5 โวลต์เพื่อรับกระแสผ่านสวิตช์ D0 และรักษาสภาวะกราวด์เทียมไว้ แต่ถ้าหากยังสงสัยในภาวะกราวด์เทียมก็ลองวาดเป็นวงจรดิไวเดอร์ที่มีแรงดันข้างหนึ่ง +5 โวลต์ ตรงกลางเป็น 0 โวลต์ และอีกปลายหนึ่งมีค่า -0.5 โวลต์

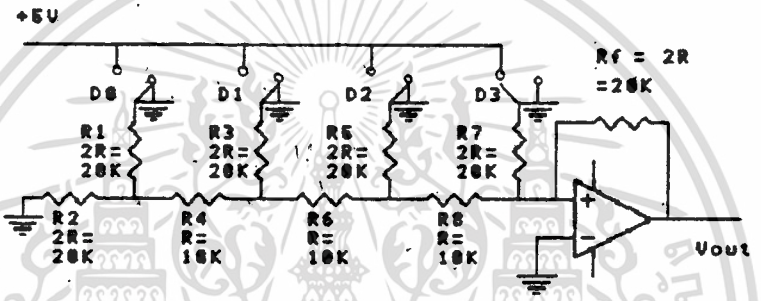
เมื่อเปิดวงจรที่สวิตช์ D0 และปิดวงจรที่สวิตช์ D1 (ขณะที่ R2 มีค่าเป็นครึ่งหนึ่งของ R1) กระแสเพิ่มเป็น 2 เท่า หรือ 0.1 มิลลิแอมป์ ไหลผ่าน RF, กราวด์เทียมและ R2 ทำให้มีแรงดันเอาต์พุต -1 โวลต์ ต่อไปก็ปิดวงจรทั้งที่ D0 และ D1 จะได้กระแส 0.05 มิลลิแอมป์ไหลผ่าน R1 และ 0.1 มิลลิแอมป์ ผ่าน R2 รวมกระแส 0.15 มิลลิแอมป์ ได้แรงดันเอาต์พุต -1.5 โวลต์

เมื่อเปลี่ยนการปิดเปิดสวิตช์ไปเรื่อยๆ จะได้แรงดันเอาต์พุตค่าต่างๆ กันกระแสที่ผ่านสวิตช์แต่ละตัวจะถูกรวมกันที่จุดกราวด์เทียมแล้วเปลี่ยนเป็นแรงดันที่เอาต์พุต โดยตัวต้านทานป้อนกลับ RF

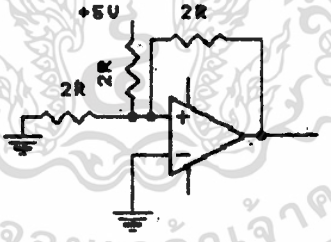
แรงดันเอาต์พุตจะเพิ่มขึ้นเป็นระดับๆ เหมือนขั้นบันได ดังนั้น 4 บิตจึงได้ 15 ระดับ แต่ละระดับต่างกัน -0.5 โวลต์ อาจกำหนดระยะห่างของแต่ละระดับได้โดยเปลี่ยนขนาดของ RF แต่ถ้า RF มีค่ามากเกินไประดับบนสุดจะขับออป-แอมป์ถึงจุดอิ่มตัว (ที่แรงดัน -14 โวลต์)



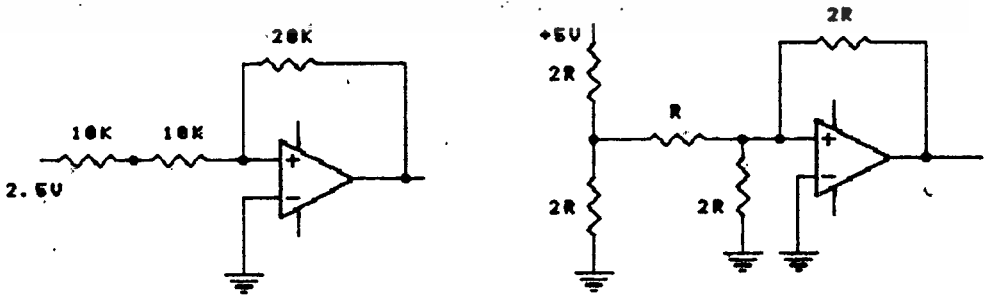
รูปที่1



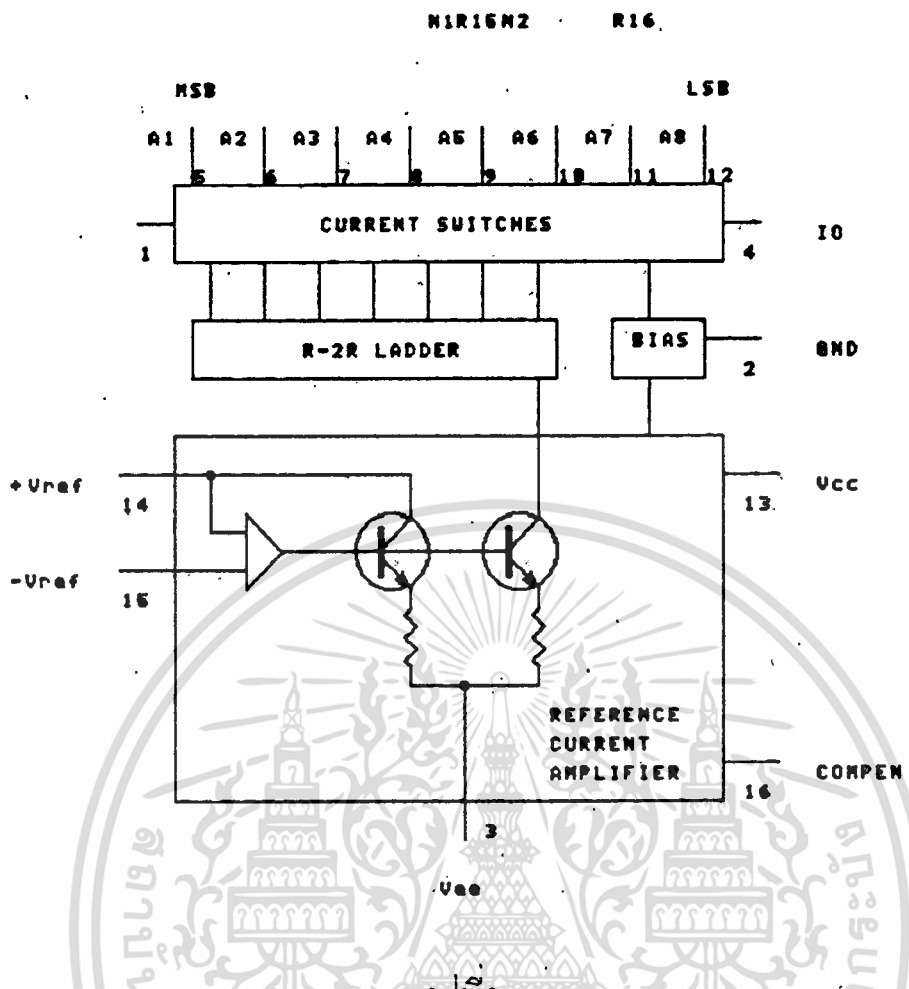
รูปที่ 2ก



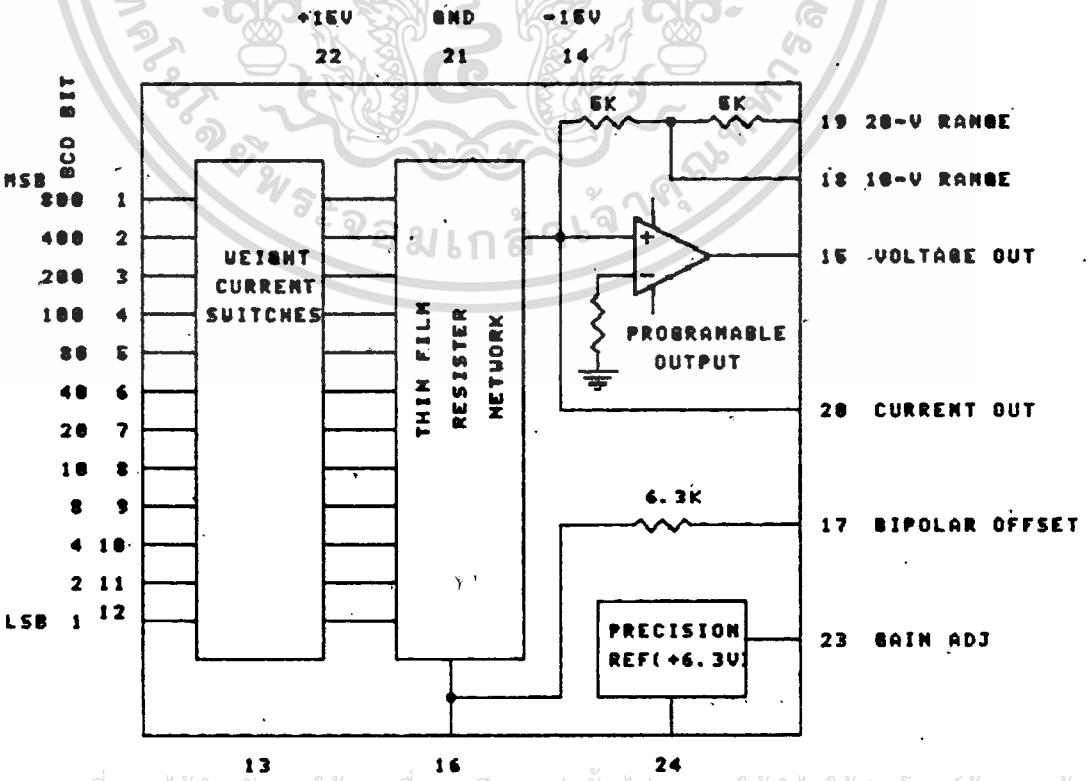
รูปที่ 2 ข



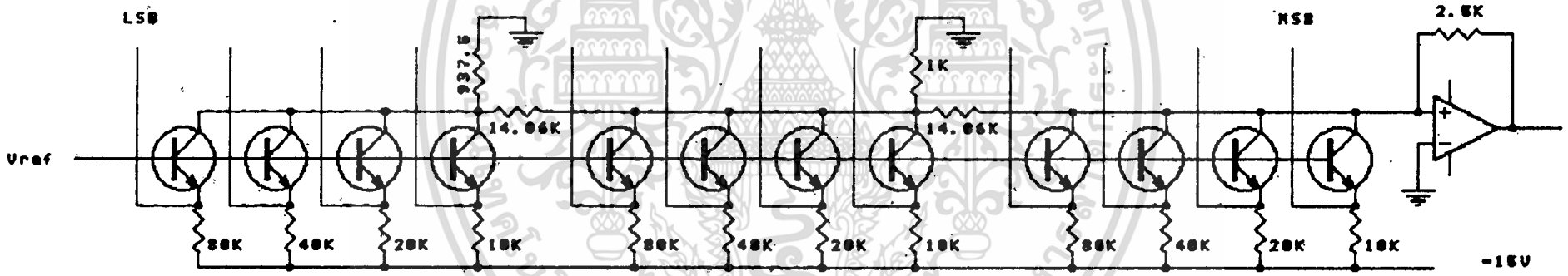
รูปที่2ค



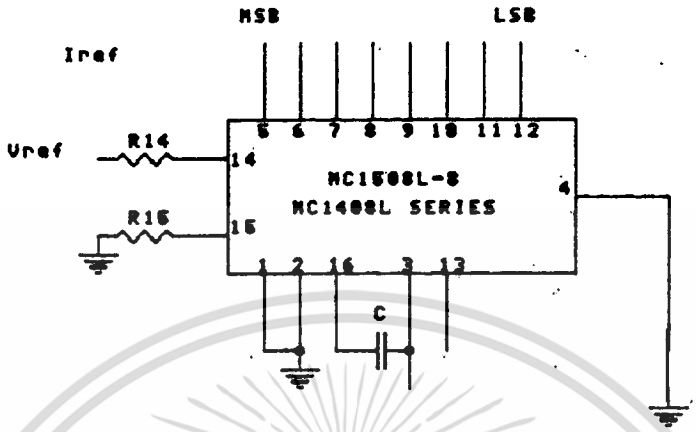
รูปที่3



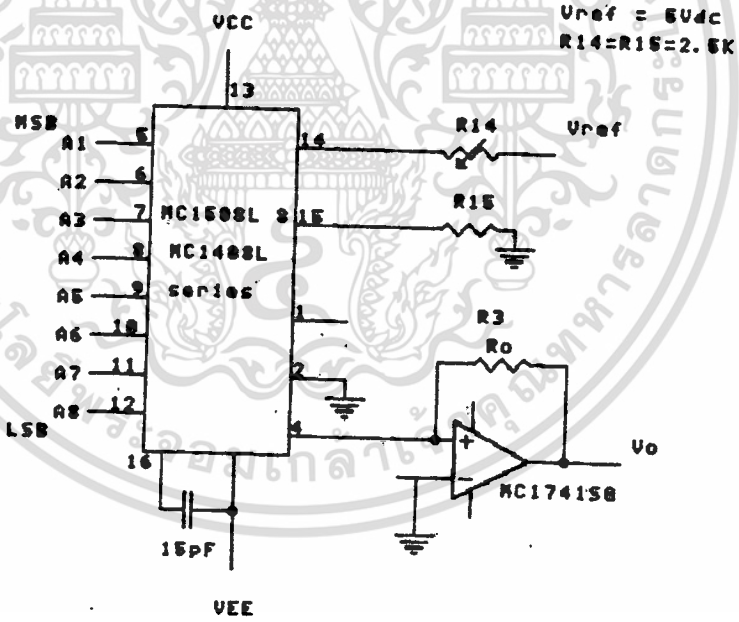
รูปที่4


LSB
MSB
2.5K
1K
14.86K
14.86K
Vref
80K
40K
20K
10K
80K
40K
20K
10K
80K
40K
20K
10K
-15V

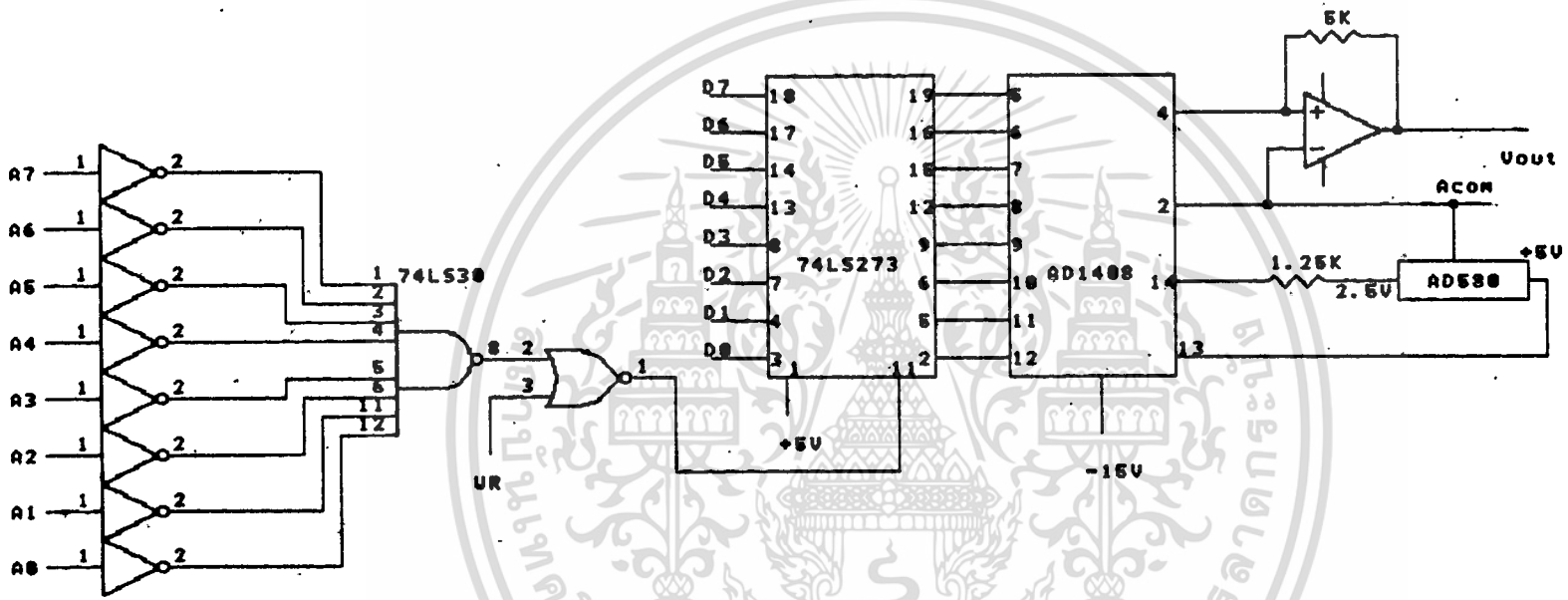
รูปที่ 5



รูปที่6ก



รูปที่6ข


A7
A6
A5
A4
A3
A2
A1
A0
74LS30
UR
D7
D6
D5
D4
D3
D2
D1
D0
74LS273
+5V
AD1408
-15V
5K
Vout
ACOM
1.25K
2.5V
AD580
+5V

รูปที่ 7

แบบใช้ตัวต้านทาน 2 ค่า (R/2R Ladder D/A Converter)

เมื่อวงจรดีทูเอ มีขนาดมากกว่า 4 บิต วงจรตามรูปที่1 จะเกิดปัญหาเนื่องจากต้องการค่าความต้านทานที่มีช่วงกว้างมาก วิธีที่ใช้หลักการไบนารี่เวตเหมือนกัน แต่ใช้ความต้านทานเพียง 2 ค่า แสดงในรูปที่ 2ก. ซึ่งกระแสจะถูกเปลี่ยนค่าแรงดันโดยออปแอมป์และตัวต้านทานป้อนกลับ RF เหมือนวงจรในรูปที่ 1 วิธีนี้เรียกว่าการใช้ความต้านทาน 2 ค่า

สังเกตให้ดี หลักการความต้านทาน 2 ค่า ดูไปก็คล้ายกับกฎของเคียร์โอฟเพียงแต่ว่าค่าความต้านทานที่ใช้เป็นอัตราส่วนที่ทำให้คำนวณได้ง่าย แรกเลยสมมติว่าสวิตช์ D3 ซึ่งเป็นสวิตช์ในบิตที่มีนัยสำคัญสูงสุดนั้นต่อกับแรงดันอ้างอิง 5 โวลต์ ในขณะที่สวิตช์ตัวอื่นปิดลงกราวด์ดังนั้น R1 และ R2 จึงต่อขนานกันลงกราวด์ สังเกตตัวต้านทาน 2R ต่อขนาน กับ 2R อีกตัวหนึ่งจึงมีค่าเท่ากับ R ค่า R นี้จะถูกบวกกับ R4 กลายเป็นค่า 2R แล้วขนานกันกับ R3 ลงกราวด์ การรวมของ R3 และตัวต้านทานก่อนหน้าจึงทำให้เหลือเพียงค่า R ต่ออนุกรมกับ R6 พิจารณาเช่นเดียวกันกับวงจรส่วนที่เหลือก็จะได้เป็นวงจรง่ายขึ้น ดังรูปที่ 2 ข.

โดยเหตุที่กราวด์เทียมของออปแอมป์มีแรงดัน 0 โวลต์ ทำให้ไม่มีกระแสไหลผ่านค่าความต้านทานเหล่านี้ลงกราวด์จึงไม่ต้องสนใจส่วนนี้ ดังนั้นแรงดัน 5 โวลต์ ที่ปลายข้างหนึ่งของR7 ค่า 20 กิโลโอห์ม ทำให้มีกระแส 0.25 มิลลิแอมป์ ผ่านที่จุดต่อและผ่าน RF 20 กิโลโอห์ม แรงดันเอาต์พุตที่ได้จากบิตที่มีนัยสำคัญสูงสุดจึงมีค่า -5 โวลต์

แรงดันที่ได้จากบิตที่มีนัยสำคัญรองลงมาก็หาได้โดยบิตสวิตช์ D2 ไปยัง +5 โวลต์ และ D3 ลงกราวด์ ตัวต้านทานทั้งหมดที่อยู่ทางซ้ายของ R5 ในรูปที่ 2 ก. ลดรูปลงเหลือเพียง 2R ต่อลงกราวด์ การวิเคราะห์วงจรสามารถนำทฤษฎีของเทวินินมาใช้ได้ โดยมีการแบ่งแรงดันระหว่าง R5และ 2R ที่ต่อลงกราวด์ ดังรูป 2ค. แรงดันของเทวินินคือแรงดันที่รอยต่อหรือ 2.5 โวลต์ ตัวต้านทานเทวินินมีค่าเท่ากับตัวต้านทาน 2 ตัวต่อขนานกัน (หรือ R) อุปกรณ์ที่อยู่ทางด้านซ้ายของ R6 สามารถลดรูปได้เหลือค่า R ต่อกับ 2.5 โวลต์ เราสามารถละทิ้ง R7 ได้เพราะจุดปลายทั้งสองค่าต่อลงกราวด์ค่าความต้านทานรวมระหว่างที่จุดรวม (จุดกราวด์เทียม) และแรงดันเทวินินคือ 2R หรือ 20 กิโลโอห์ม กระแสที่จุดรวม คือ 2.5 โวลต์ หรือ 0.125 มิลลิแอมป์ กระแสที่ผ่าน RF 20 กิโลโอห์ม ทำให้เกิดแรงดันเอาต์พุต -2.5 โวลต์ (สำหรับบิตที่มีนัยสำคัญถัดมา)

ด้วยการวิเคราะห์ในทำนองเดียวกันนี้ สามารถหาแรงดันเอาต์พุตที่บิตต่ำลงมาอีกได้ 1.25 โวลต์ และที่ค่าดิจิตอลต่ำสุดได้ 0.625 โวลต์ ในขณะที่ค่าดิจิตอลสูงสุด (สวิตช์ทุกตัวต่อไปที่ +5 โวลต์) ได้เอาต์พุตเต็มสเกลคือ 9.375 โวลต์

แม้ว่าดีทูเอ คอนเวอร์เตอร์ แบบ R/2R แสดเดอร์ จะวิเคราะห์ยากกว่าแบบใช้ตัวต้านทานหลายค่า (Weighted resistor) แต่จะง่ายกว่าสำหรับการต่อวงจรให้ถูกต้อง เพราะใช้ค่าความต้านทานเพียง 2 ค่าเท่านั้น จำนวนบิตก็เพิ่มได้โดยเพิ่มส่วนของ R/2R ลงไปวงจรนับ 4 บิตที่เป็น TTL หรือ CMOS อาจนำมาต่อแทนตำแหน่งของสวิตช์ในวงจรรูป 2 ก. เพื่อให้แรงดันเอาต์พุตเป็นขั้นบันไดได้

แบบใช้ไอซี (Monotithic and hybrid D/A converters)

โมโนลิทิค (Monolithic) หมายถึง "หินก้อนเดียว" เมื่อนำมาใช้ในวงจรรวมจะเป็นการชี้บอกว่าวงจรๆ หนึ่งถูกบรรจุอยู่บนสารกึ่งตัวนำเพียงชิ้นเดียว

ส่วนไฮบริด (hybrid) บรรจุสารกึ่งตัวนำที่เรียกว่า ชิพ (chip) เพียงชิ้นเดียวหรือมากกว่า มีตัวต้านทานหรือตัวประกอบวงจรอื่นๆ อยู่ในกรอบของไอซีตัวเดียว

ตัวอย่างวงจรเปลี่ยนสัญญาณดีทูเอโมโนลิทิคขนาด 8 บิตคือ MC1408 ซึ่งมีผังการทำงานดังแสดงในรูปที่ 3ก. 1408L เป็น DIP (Dual Inline Package) 16 ขา ใช้ VCC + 5 โวลต์ และ VEE จาก -5 โวลต์ (ต่ำสุด) ถึง - 15 โวลต์ (สูงสุด)

ใน 1408L R/2R แสดเดอร์ แบ่งกระแสที่ได้จากภาคขยายเป็น 8 ระดับขึ้นอยู่กับค่าทางเลขฐานสอง(binary) ทรานซิสเตอร์แบบไบโพลาร์จะสวิตช์ให้กระแสที่ได้สอดคล้องกับอินพุต A1 ถึง A8 การเรียงจากบิตที่มีนัยสำคัญสูงสุดถึงบิตที่มีนัยสำคัญต่ำสุดจะกลับกันกับของวงจรนับทั่วๆ ไปแต่วงจรเปลี่ยนสัญญาณดิจิตอลเป็นอะนาลอกบางตัวก็จะไม่ได้เรียงอย่างนี้ ดังนั้นควรอ่านคู่มืออย่างละเอียดถี่ถ้วนเสียก่อน

1408L มีกระแสเอาต์พุตที่สามารถเปลี่ยนเป็นแรงดันได้ด้วยออป-แอมป์และตัวต้านทานดังแสดงในรูป 3ข. แรงดันนี้สามารถคำนวณโดยใช้สูตร

$$V_{out} = \frac{VREF}{R14} \times RO\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right)$$

จากค่าที่เลือกไว้ ได้แรงดันเอาต์พุตเต็มสเกล (อินพุต A1 ถึง A8 เป็น "1") คือ

$$2 \text{ โวลต์} \times \frac{5 \text{ กิโลโอห์ม}}{1 \text{ กิโลโอห์ม}} \times \frac{255}{256}$$

ซึ่งเราถือว่าเป็นวงจรเปลี่ยนสัญญาณแบบ 10 โวลต์ เต็มสเกล

ตัวอย่างการนำไปใช้งานของ 1408L เช่น วงจรกำเนิดเสียง โดยรูปคลื่นเอาต์พุตของวงจรนับ 8 บิต สามารถนำมาต่อกับอินพุตของดีทูเอ เมื่อวงจรนับเริ่มนับดีทูเอคอนเวอร์เตอร์ก็จะให้รูปคลื่นรูปสามเหลี่ยมทางเอาต์พุต ซึ่งประกอบขึ้นด้วยขั้นบันไดเล็กๆ ถึง 255 ขั้น ความถี่เอาต์พุตเท่ากับความถี่ของสัญญาณนาฬิกาทางอินพุต หารด้วย 256 คลื่นรูปอื่น ๆ ก็อาจทำได้โดยต่อดีทูเอกับเอาต์พุตของหน่วยความจำ รอม หรือ แรม แบบ 8บิต หน่วยความจำได้ถูกโปรแกรมด้วยค่าเลขฐานสองค่าต่างๆ ดังนั้นเสียงที่ได้จะมีความหลากหลาย ฟังดูแปลกหูหรืออาจโปรแกรมเสียงตามที่ต้องการก็ได้ ข้อจำกัดของเสียงที่ออกมาก็ขึ้นอยู่กับจินตนาการของผู้ที่โปรแกรม, จำนวนหน่วยความจำ และความสามารถในการโปรแกรมเอง

ถ้าต้องการใช้งานที่บิตมากกว่านี้ก็อาจเปลี่ยนเป็นของบริษัท Datel เบอร์ DAC - HZ 12 BGC ซึ่งมี 24 ขา รูปร่างได้แสดงไว้แล้วในรูปที่ 4ก. และผังการทำงานได้แสดงไว้ในรูป 4ข. เป็นวงจรเปลี่ยนสัญญาณ 12 บิต ต้องการไฟเลี้ยง +15 โวลต์ และ -15 โวลต์ มีตัวต้านทานและออปแอมป์รวมไว้ในตัวแล้ว โดยจะให้เอาต์พุตออกมาเป็นกระแสหรือแรงดันก็ได้ แรงดันเต็มสเกลสามารถเปลี่ยนได้โดยการเปลี่ยนค่าตัวต้านทานป้อนกลับที่ออปแอมป์

แบบ R/2R แลดเดอร์ใช้ตัวต้านทาน 2 ตัวต่อ 1 บิต ในขณะที่แบบไบนารี่เวตใช้ 1 ค่าต่อ 1 บิต ข้อเสียของแบบไบนารี่เวตคือต้องใช้ค่าตัวต้านทานที่มีช่วงกว้างมากหากเกินกว่า 4บิต ปัญหานี้อาจแก้ไขได้ โดยใช้ขนาด 4 บิต 3 ชุด ต่อกันดังรูปที่ 4ค.

ลักษณะสมบัติของดีทูเอ

ลักษณะสมบัติอันแรกของการแปลงสัญญาณดิจิตอลเป็นอะนาลอกที่จะพูดถึงก็คือ ความละเอียด (resolution) ซึ่งขึ้นกับจำนวนของบิตทางด้านอินพุตตัวอย่างเช่น วงจรเปลี่ยนสัญญาณ 8 บิต มีระดับเอาต์พุต 2^8 หรือ 256 ระดับ ดังนั้นความละเอียดคือ 1 ใน 256 วงจรเปลี่ยนสัญญาณ 12 บิต มีความละเอียด 2^{12} หรือ 4096 ความละเอียดบางครั้งจะคิดเป็นเปอร์เซ็นต์ คือ 1/4096 = 0.024 %

ลักษณะสมบัติข้อต่อมาคือความถูกต้อง (accuracy)ของดีทูเอ ความถูกต้องจากการเปรียบเทียบระหว่างเอาต์พุตจริงและเอาต์พุตที่ปรากฏโดยคิดที่เต็มสเกล ถ้าวงจรเปลี่ยนสัญญาณมีเอาต์พุตเต็มสเกล 10โวลต์มีความถูกต้อง $\pm$0.2 % ดังนั้นความผิดพลาดสูงสุดคือ 0.002 x 10 โวลต์ หรือ 20 มิลลิโวลต์ ในทางทฤษฎีแล้ว ความถูกต้องของวงจรเปลี่ยนสัญญาณดิจิตอลเป็นอะนาลอกไม่ควรต่ำกว่า $\pm$/2 ของค่าที่ LSB (บิตที่มีนัยสำคัญต่ำสุด)

ความผิดพลาดเชิงเส้น (Linearity errors)ค่าจริงที่ได้จากเอาต์พุตจริงต่างจากเอาต์พุตตามทฤษฎีที่ควรจะเป็นเส้นตรง ความผิดพลาดนี้มักจะมาจากความผิดพลาดจากแหล่งจ่ายกระแสหรือค่าความต้านทาน

ความผิดพลาดแบบที่สองคือ ความผิดพลาดทางอัตราขยาย (gain error) ความผิดพลาดนี้มักเกิดจาก ความผิดพลาดของตัวต้านทานป้อนกลับของออป-แอมป์ ที่ทำหน้าที่เปลี่ยนจากกระแสเป็นแรงดัน

ความผิดพลาดออฟเซตหรือ offset error คือเมื่ออินพุตทุกตัวเป็นศูนย์แล้วเอาต์พุตไม่เป็นศูนย์ ทำให้เอาต์พุตมีค่าแรงดันผิดพลาดค่าหนึ่งบวกกับค่าจริงอยู่ตลอดเวลา ดังในรูปที่ 5ค. ความผิดพลาดนี้เกิดจากความผิดพลาดของการขยายของออปแอมป์ และกระแสรั่วไหลที่การสวิตช์

ลักษณะสมบัติต่อมาคือ โมโนโทนิค (monotonicity) จะเรียกว่าเป็นโมโนโทนิคก็ต่อเมื่อไม่มีการกระโดดข้ามขั้นตลอดย่านการใช้งาน

ส่วนเวลาเซตเอาต์พุต (output setting time) เป็นเวลาที่เอาต์พุตของวงจรเปลี่ยนสัญญาณใช้ในการเพิ่มขึ้นถึง $\pm 1/2$ของ LSB หลังจากมีการเปลี่ยนแปลงทางอินพุต ถ้าวงจรเปลี่ยนสัญญาณถูกใช้งานย่านความถี่สูง อาจทำให้มีการเพิ่มแรงดันไม่ถึงค่าที่ถูกต้องทำให้เกิดความผิดพลาดขึ้นได้อีกประการหนึ่ง

การใช้ดีทูเอทวีคูณสัญญาณ (Multipling D/A)

วงจรเปลี่ยนสัญญาณดีทูเอสามารถใช้เป็นวงจรทวีคูณสัญญาณ (multiplier) ได้โดยปรับเปลี่ยนแรงดันอ้างอิง เอาต์พุตที่เป็นสัญญาณอะนาลอกของดีทูเอเป็นสัดส่วนกับผลคูณของแรงดันอ้างอิงกับสัญญาณดิจิตอลทางอินพุต

จากวงจรในรูปที่ 6ข. ถ้าแรงดันอ้างอิงมีค่าลดลงจาก 5 โวลต์เหลือ 2.5 โวลต์ เอาต์พุต VO สำหรับอินพุตค่าสูงสุด (11111111) ก็จะลดลงเหลือ 4.98 โวลต์ เป็นครึ่งหนึ่งของเมื่อแรงดันอ้างอิงมีค่า 5 โวลต์ ดังนั้นดีทูเอทุกตัวจึงสามารถทำเป็นวงจรทวีคูณได้ อย่างไรก็ตามหากใช้ดีทูเอธรรมดา การทวีคูณสัญญาณจะมีข้อจำกัดของช่วงสัญญาณ

ดีทูเอที่ถูกออกแบบไว้สำหรับใช้ในการทวีคูณสัญญาณ จะให้เอาต์พุตเป็นเชิงเส้นและเปลี่ยนแปลงแรงดันอ้างอิงได้ในย่านกว้าง

การเชื่อมต่อกับไมโครโปรเซสเซอร์

รูปที่ 7 แสดงตัวอย่างการเชื่อมต่อดีทูเอเบอร์ 1408 กับไมโครโปรเซสเซอร์เบอร์ 8085

วงจรประกอบด้วยวงจรเปลี่ยนสัญญาณ,วงจรแลตช์แบบขนานขนาด 8 บิต(74LS273)และออปแอมป์ทางเอาต์พุต

จากไมโครโปรเซสเซอร์ใช้สายแอดเดรส 8 เส้น และใช้สัญญาณควบคุม WR , แนนด์เกต 8 อินพุตประกอบกันเป็นพอร์ตเอาต์พุต วงจรจะแสดงแรงดันอะนาลอกที่สอดคล้องกับอินพุตที่เป็นดิจิตอลจากไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์เริ่มทำงานจากอินพุตค่า 00_{16} ปรากฏที่แอดเดรสบัส (K0 - K7) เอาต์พุตของแนนด์เกตที่ส่งไปยังอินพุตของนอร์เกตจะมีระดับ "0" เอาต์พุตของนอร์เกตจึงมีระดับ "1" เมื่อมีสัญญาณ WR จากไมโครโปรเซสเซอร์ทำให้วงจรแลตช์ 74LS273 ทำงาน ข้อมูลที่บัสข้อมูลจะเก็บไว้ป้อนให้วงจรเปลี่ยนสัญญาณจนกว่าอินพุตใหม่จะเข้ามา ถ้าการตอบสนองของวงจรเปลี่ยนสัญญาณช้ากว่าความเร็วอินพุตของไมโครโปรเซสเซอร์ จะต้องเพิ่มลูปหน่วงสัญญาณไว้ในโปรแกรม

วงจรเปลี่ยนสัญญาณดิทูเอใช้เป็นพอร์ตเอาต์พุตที่ตำแหน่ง 01_{16} โดยใช้เทคนิค I/O แบบนี้ไมโครโปรเซสเซอร์สามารถใช้พอร์ตอินพุต/เอาต์พุตได้ถึง 256 พอร์ต จาก 00_{16} ถึง FF_{16}

ดิทูเอเบอร์1408 ต้องการแลตช์ที่จะเชื่อมต่อกับไมโครโปรเซสเซอร์ แต่อย่างไรก็ตามวงจรเปลี่ยนสัญญาณตัวใหม่ๆเช่น AD7524ก็มีวงจรแลตช์อยู่ภายในแล้ว วงจรรูปที่ 8 แสดงการเชื่อมต่อกับ AD7524

AD7524 มีขาสำหรับสัญญาณควบคุม WR และ chip select (CS) สำหรับถอดรหัสแอดเดรส การต่อวงจรและโปรแกรมก็คล้ายกันกับที่ได้อธิบายไว้แล้ว

การทำงานของ A/D CIRCUIT (ADC 0809)

ADC 0809 จะรับสัญญาณอนาล็อกเข้าเป็นอินพุทได้ 8 แชนแนล(channel)ในการทดลองนี้จะยกตัวอย่างมาเพียง 5 channel คือ

1. ch 0 (simulated dc. voltage 0-5 v), #00
2. ch 1 (simulated dc. voltage 0-5 v), #01
3. ch 2 (simulated dc. voltage 0-5 v), #02
4. ch 3 (sine wave หรือ i/p ใด ๆ), #03 ต่อตรง
5. ch 4 (speech lab.), #04

วิธีการเลือกแชนแนลของอินพุทใช้การสั่งงานมาจากคอมพิวเตอร์โดยใช้คำสั่ง outport [302]:= data จากนั้นจะมีสัญญาณไปสั่งขา ALE ให้แลตช์ดาต้านั้นไว้ ทำให้เอทูดีทราบว่า อินพุทมาจากแชนแนลใด

สัญญาณอินพุทจะถูกแซมปลิ้งด้วยความถี่ประมาณ 10KHz ที่เข้ามาทางขา 6 ดังนั้นสัญญาณดิจิตอลจะปรากฏที่ขา 21, 20, 19, 18, 8, 15, 14, 17 โดยขา 21 เป็น MSB และขา 17 เป็น LSB ตามลำดับ ส่วนความละเอียดของขั้นต่ำหนดได้โดยแรงดันที่ขา ref(+) และ ref(-) โดยถ้าอินพุทเท่ากับ ref(+) เอาท์พุทดิจิตอลจะได้เป็น "1" หมด (FFH) และถ้าอินพุทเท่ากับ ref(-) เอาท์พุทดิจิตอลจะได้เป็น "0" หมด (00H) ดังนั้นจะเห็นได้ว่า ref(+) และ ref(-) จะเป็นตัวกำหนดค่าอนาล็อกอินพุทสูงสุดและต่ำสุด

สัญญาณ CLK (ขา 10) ได้จากการหารความถี่ออสซิลเลเตอร์ของเครื่องคอมพิวเตอร์ด้วย 8 โดยการใช้ไอซีเบอร์ 74393 เป็นตัวหารจะได้ความถี่เอาท์พุทออกทางขา 6 ประมาณ 1 MHz ซึ่งเพียงพอกับการใช้งานความถี่แซมปลิ้ง 10 MHz

ขา start ต่อกับเอาท์พุทของ 555 ซึ่งต่อเป็นวงจรอะสเตเบิล มัลติไวเบรเตอร์ (ASATABLE MULTIVIBRATOR ได้เอาท์พุทประมาณ 10 MHz (DUTY CYCLE = 50%) เป็นสัญญาณความถี่ฉมปลิ้งมาตรฐานของวงจร

เอาท์พุทของ ADC 0809 จะถูกแยกออกจากด้าต้าบัสของคอมพิวเตอร์ด้วยไอซีเบอร์ 74244 ซึ่งบัฟเพอร์แบบไตรสเตทเกท(TRISTATE GATE) เอาท์พุทของ 74244 จะยังไม่ปรากฏจนกว่าขา 1,19 (ของ 74244) จะถูกอิเนเบิ้ล สัญญาณอิเนเบิ้ลจะได้จากการดีโค้ดพอร์ท(decode port) เบอร์ #300 ของไอซีเบอร์ 74138

การทำงาน D/A CIRCUIT (DAC 0800)

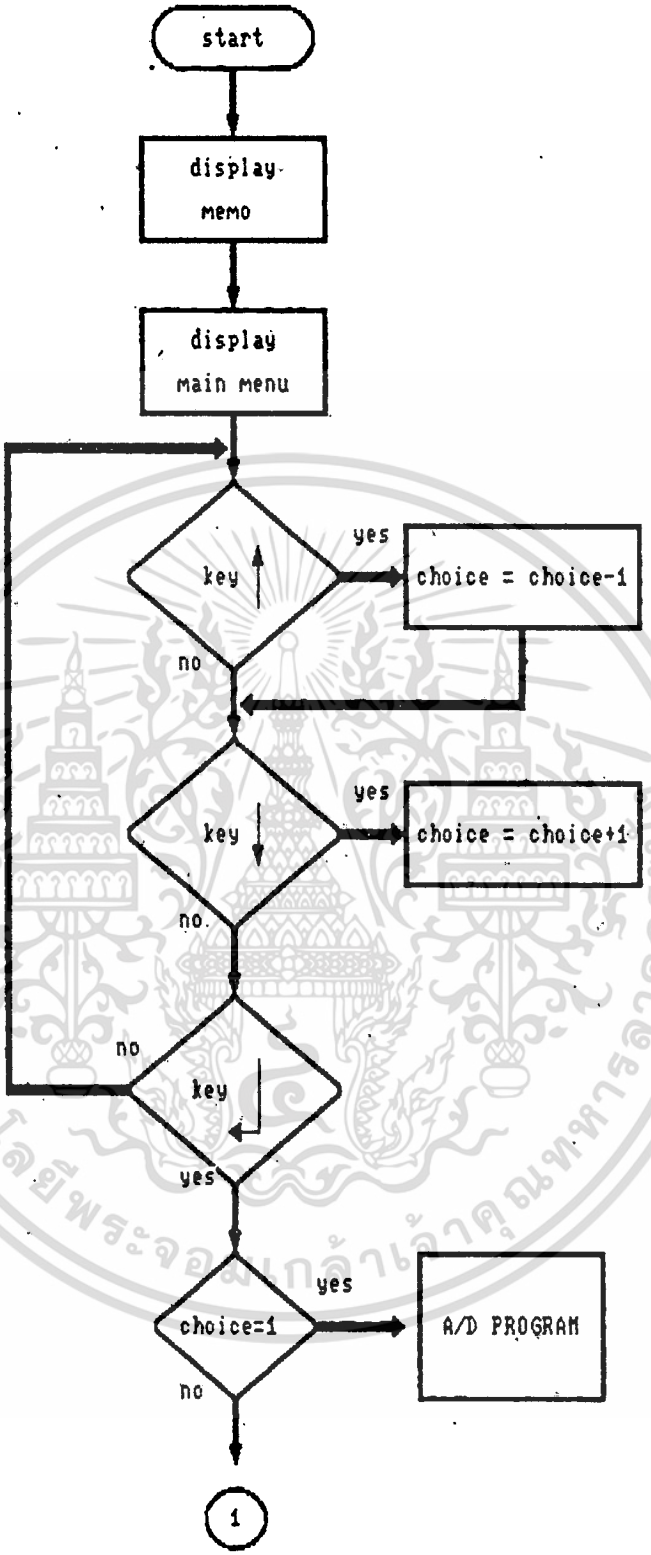
ข้อมูลดิจิตอลที่เก็บไว้ภายในคอมพิวเตอร์จะส่งออกทางดาต้าบัสเข้าสู่ดีฟลิปฟลอป (74374) เพื่อแลทช์ข้อมูลไว้โดยการดีโค้ดพอร์ทเบอร์ #301 ของไอซีเบอร์ 74138 ดังนั้นเอาท์พุทของ 74374 จะส่งออกมาได้ก็ต่อเมื่อได้รับการอิเนเบิ้ลจากนั้นสัญญาณจะถูกส่งไปยังอินพุทของ DAC 0800 ได้เอาท์พุทเป็นสัญญาณอนาล็อกที่มีลักษณะใกล้เคียงกับอินพุทที่เข้าเอทูดี

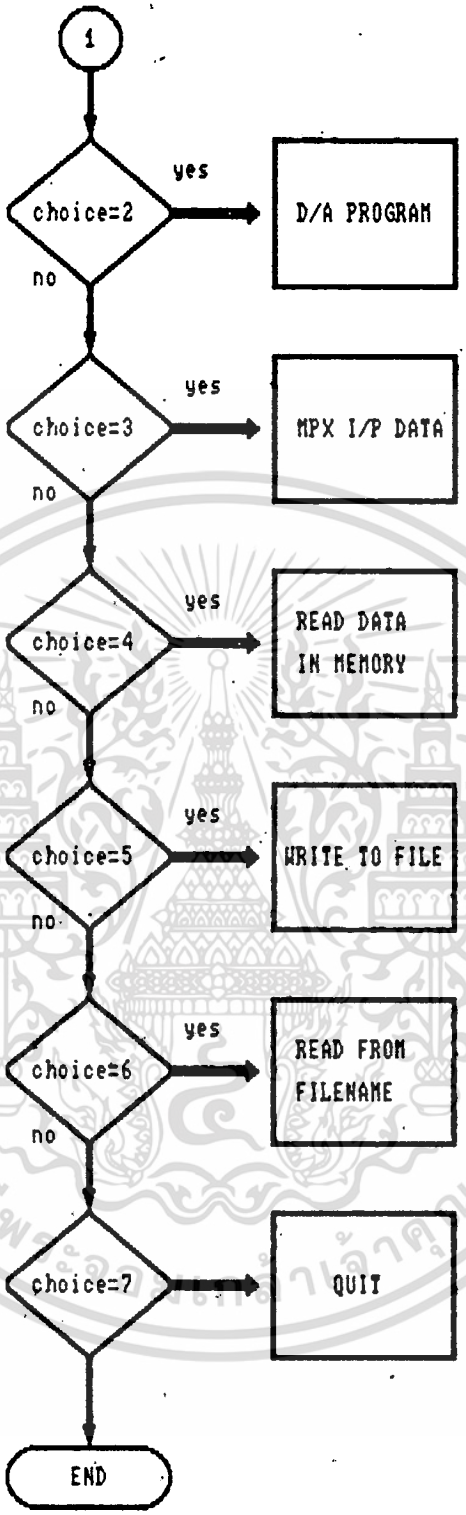
ขาref(+)ได้รับกระแสอ้างอิงขนาด 1 mA และ ref(-) ก็เป็นลักษณะเช่นเดียวกัน แต่ต่อลงกราวด์เท่านั้น ดังนั้นอินพุท 00 - FF ก็จะถูกแปลงเป็นกระแสอนาล็อกมีช่วงระหว่าง 0 - 1 mA แล้วส่งไปยัง ออป-แอมป์ เพื่อแปลงกระแส 0 - 1 mA ให้เป็นแรงดัน 0 - 5 V

เอาท์พุทของดีทูเอจะถูกแยกเป็น 2 ทาง คือ

1. ถ้าอินพุทเป็นสัญญาณเสียงจะนำเอาท์พุทนี้ไปผ่านฟิลเตอร์แบบแบนด์พาส(band pass filter (300Hf - 3KHf))เพื่อกรองเอาเพียงสัญญาณเสียง จากนั้นก็จะส่งไปที่ไอซีแอมป์ปลิฟายเออร์ขนาดกำลังขับ 10 W ขยายเสียงให้แรงขึ้นก่อนส่งออกลำโพง

2. ถ้าอินพุทเป็นแรงดันdc หรืออินพุทใดๆจะนำเอาท์พุทนี้ไปผ่านฟิลเตอร์แบบโลว์พาส (low pass filter (500 Hf)) ก็จะได้เอาท์พุทเหมือนกับอินพุทที่ป้อนเข้าเอทูดี





การทำงานของโปรแกรมมีส่วนประกอบดังนี้

โปรแกรมที่ใช้ในโปรเจกต์ชิ้นนี้ บางส่วนได้นำมาคอมพลายท์(Compile) เป็นยูนิต เพื่อประโยชน์ในการใช้งานในอนาคต และเพื่อให้โปรแกรมสั้นกระทัดรัด สามารถอิดิต(Edit)โปรแกรมได้ง่าย โปรแกรมที่ใช้งานในโปรเจกต์ประกอบด้วย

1.PHK10.EXE เป็นโปรแกรมที่เกิดจากการคอมไพล์ภาษาปาสคาลจากโปรแกรมจุด PAS มาเป็นจุด EXE

2. UNIT DOS,GRAPH,CRT เป็นยูนิตมาตราฐานของเทอร์โบปาสคาล

3. UNIT KEYBOARD เป็นยูนิตที่ใช้ในการเช็คการกดปุ่มบนแป้นพิมพ์

4. UNIT SCREEN เป็นยูนิตที่ใช้ควบคุมเคอร์เซอร์ และการทำงานของจอภาพต่างๆ

5. UNIT WIN เป็นยูนิตที่ใช้แสดงจอภาพแบบต่างๆ รวมถึงการสร้างกรอบหน้าต่าง(WINDOWS) และพูลดาวน์เมนู(PULL DOWN MENU)ในโปรแกรม

6. DOHELP เป็นส่วนช่วยให้คำอธิบายในแต่ละเมนู

7. A_D.PAS เป็นโปรแกรมที่ใช้ในเมนูเอ็ทดีและดีทูเอ คอนเวอร์เตอร์

8. BARCHART.PAS เป็นโปรแกรมที่ใช้ในเมนูมัลติเพล็กซ์อินพุทดาต้า

9. MODEL.PAS เป็นโปรแกรมแสดงภาพตอนเริ่มต้นของโปรแกรม

10. PJCT1.PAS เป็นโปรแกรมที่ใช้ในการอ่านเขียนข้อมูลจากอินพุท นอกจากนี้ยังมีไฟล์ที่มีการใช้อยู่แล้วในเทอร์โบปาสคาล ซึ่งใช้ในโหมด(MODE)กราฟิกส์ คือ

11. ไฟล์ .CHR ใช้กำหนดแอตตริบิวท์เกี่ยวกับรูปแบบตัวอักษรต่างๆ

12. ไฟล์ .BGI ใช้ในการแสดงกราฟิกส์ตามชนิดของจอภาพ

สามารถแบ่งรายละเอียดของโปรแกรมออกได้เป็น

- PHK10.EXE

เป็นโปรแกรมเมนที่ใช้ควบคุมโปรแกรมต่างๆ โปรแกรมนี้จะเรียกใช้โปรแกรมจากยูนิตต่างๆ แล้วนำมาสร้างเป็นพูลดาวน์เมนู

เริ่มต้นโปรแกรมขั้นแรกจะเรียกโปรแกรมย่อย TitleCom ในไฟล์ MODEL.PAS เพื่อแสดงภาพการต่อเชื่อมเครื่องคอมพิวเตอร์ กับชุดต่อเชื่อม จากนั้นก็จะไปเรียกโปรแกรมย่อย DispFuncKey และ MenuActive เพื่อแสดง main menu เมื่อแสดงเสร็จ ก็จะอ่านคีย์ที่ถูกกดจากคีย์บอร์ด ด้วยโปรแกรมย่อย ReadFuncKey ใน UNIT KEYBOARD แล้วก็จะตรวจว่าเป็นคีย์อะไรด้วยโปร

แกรมย่อย TestKey การกดคีย์จะมีผลทำให้ตัวแปร choicemenu [statusmenu] ซึ่งเป็นตัวแปรที่ใช้เช็คในการเลือกโปรแกรมย่อยต่าง ๆ เป็นดังนี้

up_key	เลื่อนเมนูปัจจุบันไปที่เมนูก่อนหน้านี้
down_key	เลื่อนเมนูปัจจุบันไปที่เมนูหลังเมนูนี้
pageup_key	เลื่อนเมนูปัจจุบันไปที่เมนูแรก
pagedown_key	เลื่อนเมนูปัจจุบันไปที่เมนูสุดท้าย
F1_key	แสดง Help ของเมนูปัจจุบันออกมา โปรแกรมย่อยเหล่านี้ จะอยู่ในไฟล์ DOHELP.PAS
enter_key	จะไปเรียกใช้โปรแกรมย่อยซึ่งเป็น choice ที่เลือก ซึ่งมีการทำงานของโปรแกรมย่อยต่างๆ ดังนี้

- โปรแกรมย่อย TitleCom โปรแกรมย่อยนี้ จะอยู่ในไฟล์ MODEL.PAS ซึ่งโปรแกรมย่อยนี้ จะเป็นภาพกราฟฟิกส์ แสดงถึงการต่อเชื่อมระหว่างเครื่องคอมพิวเตอร์กับชุดต่อเชื่อม

- โปรแกรมย่อย A_D จะเป็นโปรแกรมย่อยที่ใช้ในการทดลองเอทดิคอนเวอร์ทเตอร์ซึ่งจะแสดงภาพด้วยโปรแกรมย่อย FramePicture และ OutTextXY แล้วก็จะอ่านข้อมูลมาจากพอร์ทอินพุตด้วยคำสั่ง inp:=port[$300] แล้วนำข้อมูลมาแปลงจากข้อมูลแบบไบท์ มาเป็นเลขฐานสิบหกที่โปรแกรมย่อย ByteToHex ถ้ามีการกด คีย์ ก็จะค้างค่าอินพุตเอาไว้ที่หน้าจอ จนกว่าจะมีการกด key และจะเช็ค key อีกครั้งว่าเป็น y หรือ n ถ้าเป็น y ก็จะอ่านข้อมูลใหม่มาแสดงอีกครั้ง แต่ถ้าเป็น n ก็จะจบโปรแกรมย่อยนี้ แล้วกลับเข้าสู่เมนเมนูอีกครั้ง

- โปรแกรมย่อย D_A จะเป็นโปรแกรมย่อยที่ใช้ในการทดลอง ดีทูเอคอนเวอร์ทเตอร์แล้วจะรับค่าข้อมูลเป็นเลขฐานสิบหกมาจากคีย์บอร์ด แล้วนำข้อมูลนั้นไปแปลงเป็นไบท์ด้วยโปรแกรมย่อย HexTo Byte แล้วเอาท์ออกไปที่เอาท์พุตพอร์ท ด้วยคำสั่ง port[$301]:=date จนกว่าจะมีการตอบปฏิเสธการรันโปรแกรมต่อไปด้วยการกดคีย์ n ก็จะเป็นการสิ้นสุดโปรแกรมย่อยนี้ แล้วกลับสู่เมนเมนูต่อไป

- โปรแกรมย่อย BarChart จะเป็นโปรแกรมที่ใช้ในการทดลองการมัลติเพลกซ์ ข้อมูลอินพุตโปรแกรมจะอ่านค่าจำนวนช่องข้อมูลมาจากคีย์บอร์ด จากนั้นก็จะมาอ่านข้อมูลเข้ามาจากอินพุตช่องต่างๆ แล้วแสดงค่าอินพุตเหล่านั้นเป็นกราฟแท่ง เราสามารถทดสอบได้จากการลองปรับอินพุตที่แชนแนล 0 จนถึงแชนแนล 2 แล้วดูผลที่จอภาพ ถ้ามีการกดคีย์ก็จะเป็นการสิ้นสุดโปรแกรมย่อยนี้ แล้วกลับไปสู่เมนเมนูต่อไป

- โปรแกรมย่อย Do_Help จะเป็นโปรแกรมที่จะเรียกโปรแกรมย่อย Help ต่างๆ ในไฟล์ DOHELP.PAS ตามค่าของ choice[statusmenu] ในไฟล์ DOHELP.PAS นี้ จะมีโปรแกรมย่อย Help1 ถึง Help7 ซึ่งแต่ละโปรแกรมย่อยนี้ จะอธิบายถึงการใช้งานเมนูต่างๆ ตามที่แสดงที่หน้าจอ

- โปรแกรมย่อย InMem โปรแกรมนี้จะอ่านข้อมูลจากไมค์ แล้วนำมาเก็บไว้ในหน่วยความจำที่ได้จองไว้ที่ส่วนหัวของโปรแกรม แล้วจากนั้นก็จะเอาท์ออกไปที่เอาท์พุต เพื่อจะได้แปลงเสียงที่ได้เก็บไว้กลับเป็นสัญญาณเสียงตามเดิม

- โปรแกรมย่อย WriteFile จะอ่านชื่อไฟล์ที่ต้องการเขียนมาจากคีย์บอร์ด แล้วก็จะทำการเขียนข้อมูลที่อ่านเข้ามาจากอินพุตแชนแนล 4 ลงบนไฟล์นี้ด้วยโปรแกรมย่อย ListDirWrite โปรแกรมย่อย ListDirWrite นี้จะตรวจดูว่า ชื่อไฟล์ซ้ำกับไฟล์เดิมที่มีอยู่ในแผ่นนั้นเดิมหรือไม่ ถ้าซ้ำ โปรแกรมก็จะถามว่าจะเขียนทับหรือไม่จากการอ่านคีย์บอร์ด ถ้าเป็น y ก็จะเขียนทับ ถ้าเป็น n ก็จะจบโปรแกรมย่อยนี้ แล้วกลับสู่เมนเมนู

- โปรแกรมย่อย ReadFile จะอ่านชื่อไฟล์ที่ต้องการเขียนมาจากคีย์บอร์ด แล้วทำการตรวจว่ามีไฟล์นี้อยู่ในแผ่นหรือไม่ ถ้าไม่มีก็จะแสดงข้อความว่า "File Not Found" ออกมา ถ้าพบก็จะเอาท์ข้อมูลออกมาที่เอาท์พุตเป็นสัญญาณเดิมที่ได้บันทึกไว้ แล้วกลับสู่เมนเมนู

- โปรแกรมย่อย Quit โปรแกรมนี้ จะทำให้ตัวแปรบูลีนที่ชื่อ finish เป็นจริง ตัวแปร finish นี้จะเป็นการสั่งให้จบโปรแกรมเมนเมนูลงเมื่อมันเป็น "true" ซึ่งทำได้โดยการเลือกเมนูนี้

- unit screen

มีโปรแกรมย่อยที่ต้องใช้งานคือ

- โปรแกรมย่อย cursor off จะเป็นตัวทำให้เคอร์เซอร์(cursor) หายไปจากจอภาพ

- โปรแกรมย่อย cursor on จะเป็นตัวที่เรียกให้เคอร์เซอร์ปรากฏบนจอภาพอีกครั้ง

- โปรแกรมย่อย set attr จะทำให้ตัวอักษรเกิดแอตตริบิวท์(attribute)ตามที่เราได้กำหนดไว้

- unit keyboard

มีโปรแกรมย่อยที่ต้องใช้งานคือ

- โปรแกรมย่อย readfunckey เป็นกระบวนการที่จะรอรับการกดปุ่มบนแป้นพิมพ์และตรวจสอบว่าปุ่มที่กดเป็นปุ่มฟังก์ชั่นหรือปุ่มตัวอักษร ถ้าผู้ใช้กดปุ่มฟังก์ชั่น พารามิเตอร์ KEY จะเก็บสแกนโค้ดของปุ่มและตัวแปร funckey ภายในยูนิตจะมีค่าเป็น true ถ้าเป็นปุ่มตัวอักษรพารามิเตอร์ key จะเก็บรหัสแอสกีของตัวอักษร และตัวแปร funckey จะมีค่า false

- unit win

มีโปรแกรมที่ต้องใช้งานคือ

- โปรแกรมย่อย SetBoxAttr มีหน้าที่กำหนดแอตตริบิวท์ของกรอบหน้าต่าง

- โปรแกรมย่อย SetCharAttr มีหน้าที่กำหนดแอตตริบิวท์ของตัวอักษรในช่องหน้าต่าง

- โปรแกรมย่อย SetBoxStyle เป็นตัวกำหนดแบบของกรอบหน้าต่าง โดยมีให้เลือก 4 แบบคือ single , double , mix1 , mix2

- โปรแกรมย่อย SetHeadAttr

มีหน้าที่เป็นตัวกำหนดแอตตริบิวท์ของเฮดเดอร์(Header)ของช่องหน้าต่าง

- โปรแกรมย่อย SetWinAttr ยามเมื่อเรากำหนดช่องหน้าต่าง และทำการเปิดช่องหน้าต่าง พื้นหลังของหน้าต่างจะมีแอตตริบิวท์ตามที่ได้กำหนดด้วยกระบวนการนี้

- โปรแกรมย่อย SetWinHeader ข้อความที่กำหนดนี้จะแสดงอยู่ที่กรอบหน้าต่างด้านบนและมีแอตตริบิวท์ตามที่ได้กำหนดจาก SetHeadAttr และข้อความนี้จะถูกจัดให้อยู่กึ่งกลางของเส้นกรอบหน้าต่างด้านบนโดยอัตโนมัติ

- โปรแกรมย่อย WindowClose ทำหน้าที่ยกเลิกการใช้ช่องหน้าต่างปัจจุบัน และจะเรียกคืนจอภาพที่เก็บเอาไว้มาอยู่ในสภาพเดิม

- โปรแกรมย่อย WindowOpen เป็นการเปิดช่องหน้าต่างขึ้นมาใช้งาน โดยจะสร้างช่องหน้าต่างตามขนาดที่กำหนด และจะมีการเก็บจอภาพเอาไว้ทุกครั้ง การเปิดช่องหน้าต่างทำได้ไม่จำกัดขึ้นอยู่กับหน่วยความจำที่มีอยู่

การทดลองที่ 1 พื้นฐานการแปลงสัญญาณอนาล็อกเป็นดิจิตอล

จุดประสงค์ 1. เพื่อให้ทราบหลักการแปลงเอทูดีแบบต่าง ๆ

2. เพื่อให้นักศึกษาได้คุ้นเคยกับไอซีเบอร์ ADC 0809

ทฤษฎี ADC 0809 เป็นไอซีเอทูดีแบบชิ้กเซสชีฟแอบพล็อคชิเมทขนาด 8 BIT อินพุทสามารถมีได้ 8 แชนแนล วิธีการเลือกแชนแนลอาศัยวิธีการมัลติเพล็กซ์(MULTIPLEX) โดยป้อนสัญญาณเลือกที่ขา A, B, C ประกอบกับการป้อนสัญญาณแอดเดรสแลทช์อิเนเบิ้ล(address latch enable หรือ ALE) ข้อดีของไอซีตัวนี้คือสามารถนำไปต่อใช้งานกับไมโครโปรเซสเซอร์ได้หลายเบอร์ ระดับสูงต่ำของสัญญาณอนาล็อกอินพุทกำหนดได้โดยแรงดันขา ref(+) และ ref(-) บัฟเฟอร์อยู่และได้รับการอิเนเบิ้ลด้วยคอมพิวเตอร์พอร์ท 300 ดังนั้นเราสามารถจำลองการแซมปลิ้งได้โดยการสั่งจากซอฟท์แวร์ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. ดิจิตอล โวลท์ มิเตอร์
2. ชุดทดลองอินเตอร์เฟส
3. คอมพิวเตอร์
4. แผ่นดิสก์เกตโปรแกรมประกอบด้วยไฟล์
   - PHK10.EXE
   - *.BGI
   - *.CHR
   - *.TPU

ลำดับขั้นการทดลอง

1. เสียบปลั๊กของอุปกรณ์ทั้งหมดเข้าไป 220 V ก่อน
2. บู๊ท ดอส
3. เรียกไฟล์ A>PHK10 <--: หน้าจอจะปรากฏโลโก้ ให้กดคีย์ใด ๆ เพื่อผ่านไปยังส่วนต่อไปของโปรแกรม
4. ขณะนี้หน้าจอจะปรากฏ MAIN MENU ดังรูปที่ 1 ในการทดลองนี้จะใช้เพียงเมนูที่มีชื่อว่า ANALOG TO DIGITAL ให้ใช้คีย์ เลื่อนแถบแสงไปยังเมนูดังกล่าวแล้วกด <--:
5. หน้าจอจะเป็นการทดลองนี้ให้ใช้นอร์ท CH0 เป็นตัวป้อนอินพุทโดยป้อนอินพุทค่าต่างๆ

ตามตาราง แล้วกด enter ดูค่าของดิจิตอลที่ได้มา และก่อนที่จะป้อนค่าใหม่ให้กด Y เมื่อครบทุกค่าแล้วให้กด N

MAIN MENU

ANALOG TO DIGITAL
DIGITAL TO ANALOG
MULTIPLEX I/P CHANNEL
READ IN MEMORY
READ & WRITE TO FILE
READ FRON FILE
QUIT

รูปที่ 1

วิธีการคำนวณ

การใช้งานเอดีซีตามแรงดันที่ใช้ในชุดต่อเชื่อมนี้ แรงดันจะอยู่ในย่าน 0 ถึง 5 โวลท์ ดังนั้น ช่วงการวัดแรงดันจะเป็น 5 โวลท์ เนื่องจากเอดีซีมีขนาด 8 บิท ทางด้านดิจิตอล จึงทำให้ย่านการวัดดังกล่าวถ้าเปลี่ยนเป็นดิจิตอลจะได้ข้อมูลเป็น 00 ถึง OFFH หรือมีช่วงการวัดเท่ากับ 256 สเต็ป ความละเอียดของการแปลงสัญญาณ V/STEP เป็น 5/256 = 20.5 มิลลิโวลท์/สเต็ป

ตัวอย่าง

สัญญาณอนาลอกมีแรงดัน 3 โวลท์ ผ่านวงจรเอดีซี อยากทราบว่าทางด้านเอาท์พุตของเอดีซีจะมีข้อมูลเท่ากับเท่าใด

วิธีหา จากแรงดันอินพุตจะมีช่วงของแรงดันเป็น 3 โวลท์

ดังนั้นเมื่อคิดเป็นสเต็ปจะได้ = 3/20.5E-3

= 146.3 สเต็ป

นำค่า 146 สเต็ปไปแปลงเป็นเลขฐาน16 ได้ 92H

ดังนั้นแรงดันอนาล็อก 3 โวลท์จะเป็นข้อมูลดิจิตอล 92H

ANALOG – DIGITAL		
อนาล็อกอินพุท	ดิจิตอลเอาท์พุท	ดิจิตอลเอาท์พุท (คำนวณ)
0		
0.5		
1		
1.5		
2		
2.5		
3		
3.5		
4		
4.5		
5		

ตารางที่ 1 แสดงผลการทดลองเอทูดี

สรุปผลการทดลอง

คำถาม 1.ถ้า ref(-) = 0 และ ref(+) = 10 ข้อมูลเอาท์พุทเป็น 8 บิท (00 - FF) จะมีค่ารีโซลูชั่นเท่าไร

2. จากข้อ 1 ถ้าอินพุท = 3.5 V จะได้ข้อมูลเอาท์พุทค่าเท่าไร

3. จงอธิบายความหมาย

3.1 RESOLUTION

3.2 LINEARITY

3.3 ACCURACY

3.4 CONVERSION TIME

3.5 EXTERNAL INPUT IMPEDANCE และ EXTERNAL OUTPUT IMPEDANCE

4. จงยกตัวอย่างวิธีการแปลงเอทุดิมา 3 วิธี อธิบาย

การทดลองที่ 2 พื้นฐานการแปลงสัญญาณดิจิตอลเป็นอนาล็อก

จุดประสงค์ 1. เพื่อให้ทราบหลักการแปลงดีทูเอแบบต่าง ๆ

2. เพื่อให้นักศึกษาได้คุ้นเคยกับอินพุทเอาท์พุทเบอร์ DAC 0800

ทฤษฎี DAC 0800 เป็นดีทูเอแบบ 8 บิท ใช้ ref(+) และ ref(-) เป็นกระแสและให้เอาท์พุทเป็นกระแสอนาล็อกแบบกลับขั้วและไม่กลับขั้ว (มี 2 ขา) ในการใช้งานจุดนี้ให้ศึกษาได้จากดาต้าชีท ลักษณะของดีทูเอจะมีข้อดีคือ มีคอนเวอร์ชั่น ไทม์(CONVERSION TIME)ต่ำมากๆ ดังนั้นการป้อนอินพุทจึงใช้การแลทช์ข้อมูล โดยใช้ดีฟลิปฟลอป ข้อมูลจากคอมพิวเตอร์จึงถูกแปลงออกมาตลอดเวลา และเพื่อให้ได้อัตราการแปลงข้อมูลเท่ากับเอทูดี จึงต้องอาศัยการอิเนเบิ้ลด้วยอัตราเดียวกับเอทูดี

อุปกรณ์ที่ใช้ในการทดลอง เหมือนกับการทดลองที่ 1

ลำดับขั้นการทดลอง

1. ทำตามขั้นตอนที่ 1 และ 3 ของการทดลองที่ 1 จนเข้าสู่เมนเมนู
2. ใช้คีย์ เลื่อนแถบแสงไปยังเมนู DIGITAL TO ANALOG แล้วกด
3. หน้าจอจะปรากฏเป็นการทดลองนี้ ให้ป้อนข้อมูล 00 - FF ตามตาราง โดยเมื่อป้อนค่าถูกต้องแล้วให้กด ข้อมูลจะออกที่เอาท์พุทของดีทูเอทันทีแล้วใช้มิเตอร์วัดสัญญาณอนาล็อคเอาท์พุท

วิธีการคำนวณ

สมมุติว่าข้อมูลจากการป้อนเป็น 0C0H ส่งออกไปยังวงจรดีเอซี จะได้เอาท์พุตเป็นเท่าใด

วิธีหา ค่าของข้อมูล 0C0H จะได้ 128+64 = 192 สเต็ป

แรงดันเอาท์พุต = 192*20.5E-3

= 3.936 โวลท์

DIGITAL - ANALOG		
ดิจิตอลอินพุท	อนาล็อกเอาท์พุท	อนาล็อกเอาท์พุท (คำนวณ)
00		
1F		
3F		
5F		
7F		
9F		
AF		
CF		
EF		
FF		

ตารางที่ 2 แสดงผลการทดลองดีทูเอ

สรุปผลการทดลอง

คำถาม 1. ถ้า ref(-) = 1 และ ref(+) = 5V ข้อมูลอินพุทเป็น 8 บิท จะได้ริโซลูชั่นเท่าใด และถ้าดาต้าเท่ากับ 5FH จะได้ค่าอนาล็อกเอาท์พุทเท่าใด

2. จงยกตัวอย่างวิธีการแปลงดีทูเอมา 2 วิธี อธิบาย

การทดลองที่ 3

จุดประสงค์ 1. เพื่อให้เข้าใจการใช้งานเอทูดีแบบหลายอินพุทโดยใช้การมัลติเพล็กซ์ แชนแนล

2. เข้าใจวิธีการเลือกแต่ละแชนแนลด้วยซอฟท์แวร์ ภาษาปาสคาล

ทฤษฎี ในโรงงานอุตสาหกรรม ถ้าหากเราต้องการให้มีการประมวลผล ,วิเคราะห์ หรือควบคุมโปรเซสต่างๆ ด้วยคอมพิวเตอร์ เราจำเป็นต้องใช้วงจรเอทูดี เพื่อเปลี่ยนสัญญาณอะนาลอก เป็นสัญญาณดิจิตอลเพราะคอมพิวเตอร์จะรู้จักแต่สัญญาณดิจิตอลเท่านั้น อีกทั้งในโรงงานอุตสาหกรรม โปรเซสต่างๆไม่ได้มีเพียงโปรเซสเดียวเท่านั้น เราจึงจำเป็นต้องใช้วงจรเอทูดีที่มีหลายๆ อินพุท เพื่อจะเป็นการประหยัดวงจรต่างๆได้ ในการทดลองนี้จึงเป็นการทดลองวงจรเอทูดีคอนเวอร์เตอร์เบอร์ ADC 0809 ซึ่งมีอยู่ 8 อินพุท แต่การทดลองนี้จะจำลองการทำงานเพียง 3 อินพุทเท่านั้น

อุปกรณ์ในการทดลอง

เหมือนกับการทดลองที่ 1

ลำดับขั้นการทดลอง

1. ทำเหมือนการทดลองที่ 1 ข้อ 1 - 2
2. เลื่อนแถบแสงไปที่เมนูที่ 3 MULTIPLEX I/P CHANNEL
3. ที่หน้าจอจะปรากฏคำถามว่าต้องการกี่อินพุท ก็ให้ใส่ตามต้องการ แต่เนื่องจากได้จำลองการทำงานมาแค่ 3 อินพุท จึงเลือกได้แค่สูงสุด 3 อินพุท ถ้าเลือกเกินจะไม่ปรากฏค่าแสดงที่จอ
4. ลองปรับ VOLUME ที่ CH0 - CH3 ดูว่าเกิดอะไรขึ้นที่หน้าจอ
5. ให้ลองศึกษาและทำความเข้าใจการทำงานของฮาร์ดแวร์และซอฟท์แวร์ ให้เข้าใจ

สรุปผลการทดลอง

คำถาม 1. ค่าของอินพุทกับ SCALE บนหน้าจอ 0 -- 255 มีความสัมพันธ์กันอย่างไร ให้แสดงวิธีการคำนวณออกมา

2. ในกรณีที่มีการใช้อินพุทเกินกว่า 8 CHANNEL จะต้องทำอย่างไร อธิบาย

การทดลองที่ 4 การประยุกต์ใช้งาน วงจรแอนดีและดีทูเอ

จุดประสงค์ 1. เพื่อให้นักศึกษาสามารถประยุกต์ใช้งานวงจรแอนดีและดีทูเอได้

อุปกรณ์ที่ใช้ในการทดลอง เหมือนกับการทดลองที่ 1

ลำดับขั้นการทดลอง

1. READ IN MEMORY

การทดลองนี้เป็นการทดลองในการเก็บข้อมูลซึ่งเป็นสัญญาณเสียงพูดเข้าไว้ในหน่วยความจำ (memory) ในขั้นตอนแรก และขั้นตอนถัดไปจะเป็นการนำข้อมูลที่เก็บไว้นำมาแปลงออกคืนสู่ลักษณะเดิม

1. เลือกเมนูชื่อ READ IN MEMORY แล้วกด

2. เครื่องจะถามว่าให้อ่านข้อมูลออกมาซ้ำกี่ครั้ง ให้ป้อนค่าตัวเลขแล้วกด

3. เครื่องจะถามว่าพร้อมจะพูดรึยัง ถ้าพร้อมให้กด แล้วพูดได้เลย (จะพูดได้ประมาณ 18 วินาที) พอหมดเครื่องจะพูดเหมือนกับที่เราบันทึกได้ตามจำนวนรอบที่กำหนดจากข้อ 2 พอหมดจะกลับสู่เมนูเมนูดังเดิม

2. READ & WRITE TO FILE

การทดลองนี้ จะเป็นการทดลองที่คล้ายกับการทดลอง READ IN MEMORY แต่มีข้อแตกต่างกันคือเมื่อหมดเวลาพูดแล้วเครื่องจะเก็บข้อมูลนั้นลงไปในไฟล์

1. เลือกเมนูชื่อ READ & WRITE TO FILE แล้วกด

2. เครื่องจะถามว่าให้เก็บข้อมูลในไฟล์ชื่ออะไร ตอนนี้ให้นำดิสก์ใส่ไดร์ว A แล้วป้อน A : LONGDU.BIN แล้วกด

3. เครื่องจะถามว่าพร้อมที่จะบันทึกรึยัง ถ้าพร้อมให้กดแล้วพูดได้เลย พอพูดจบเครื่องจะกลับสู่ลักษณะเดิมตอนนี้ให้กด และข้อมูลจะถูกเก็บในดิสก์เรียบร้อยแล้ว

3. READ FROM FILE

การทดลองนี้จะนำข้อมูลจากการทดลอง READ & WRITE TO FILE มาแสดงผล

1. เลือกเมนูชื่อ READ FROM FILE

2. เครื่องจะถามว่าให้อ่านข้อมูลจากไฟล์ชื่ออะไร ให้นำดิสก์ที่เก็บข้อมูลจากการทดลองที่แล้วใส่ไดร์ว A แล้วป้อน A: LONGDU.BIN แล้วกด

3. เครื่องจะถามว่าให้พูดซ้ำกี่รอบ ให้ใส่ตัวเลขเข้าไปแล้วกด แล้วเครื่องจะพูดออกมาทันที

สรุปผลการทดลอง

คำถาม

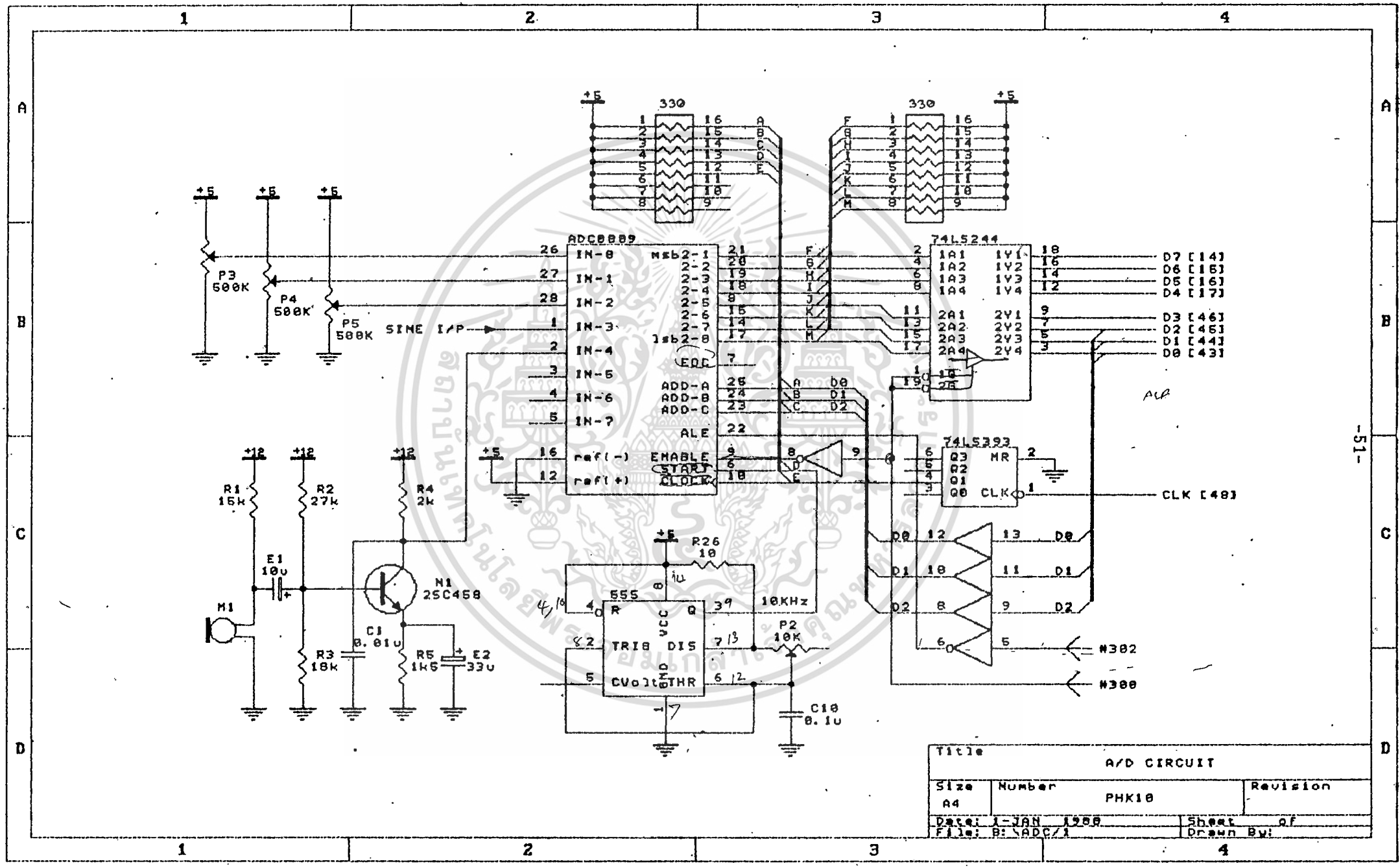
1. จงออกแบบวงจรการประยุกต์ใช้งาน ADC 0809 และ DAC 0800 มา 1วงจร พร้อมทั้งอธิบายการทำงาน
2. ให้ลองสังเกตขนาดของไฟล์ที่ได้ทำการบันทึกข้อมูลลงไป และพิจารณาจากซอฟท์แวร์ว่า หลักการของการจองหน่วยความจำเพื่อใช้ในการเก็บข้อมูลมีหลักการอย่างไร อธิบาย และ ถ้าต้องการที่จะลดหรือเพิ่มขนาดของหน่วยความจำจะต้องแก้ไขโปรแกรมอย่างไร
3. ให้เขียนโปรแกรมเพื่อลองเก็บข้อมูลในหน่วยความจำขนาด 64 K(1เซกเมนต์)

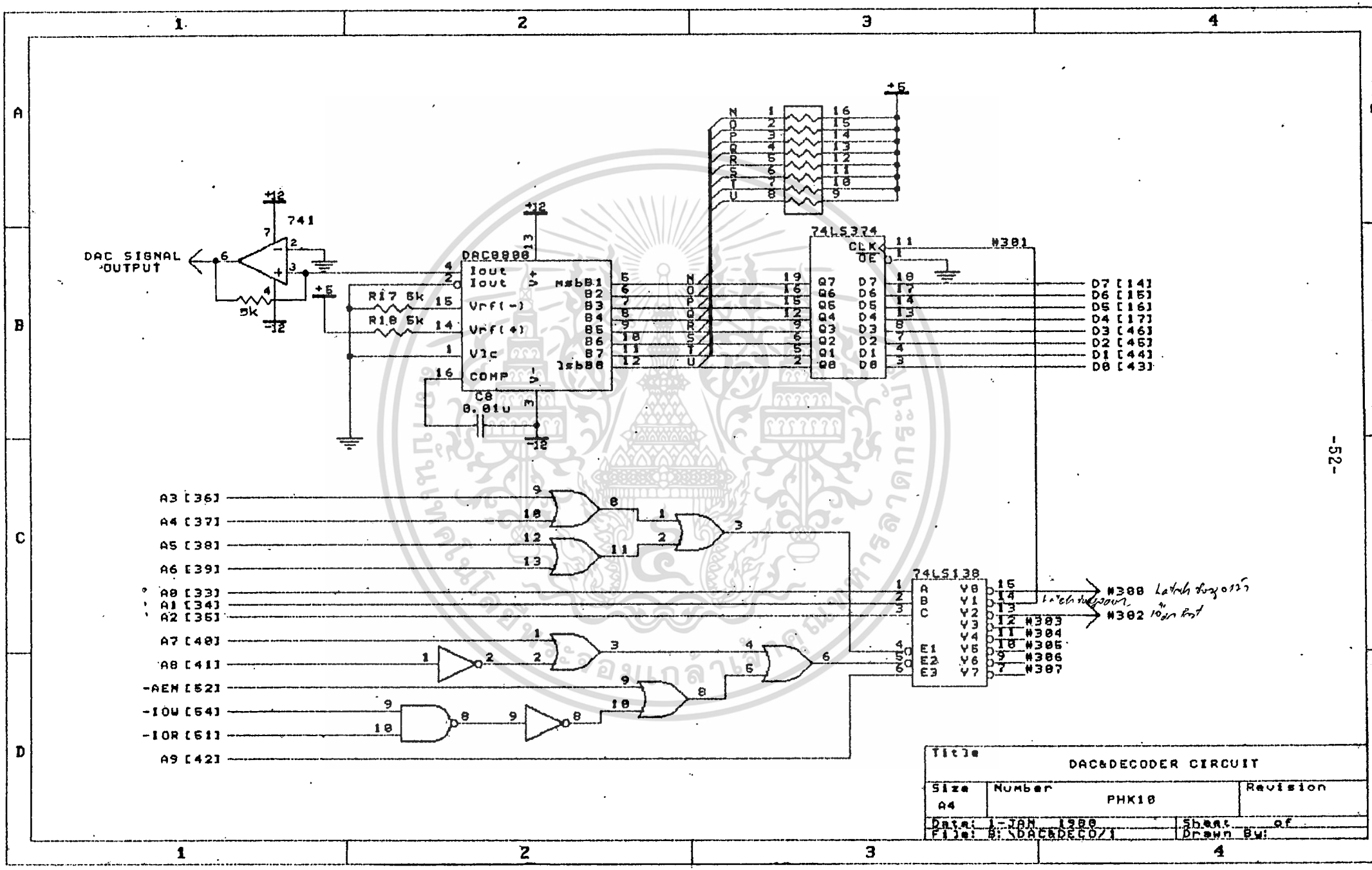
สรุปผลการทดลอง

ฮาร์ดแวร์ ในส่วนฮาร์ดแวร์นั้นสามารถแปลงสัญญาณจากเอทูดีเป็นดีทูเอได้ แต่ในการมัลติเพล็กซ์ดาต้า หลายๆ แชนแนลนั้นจะเกิดการกวนกันในแต่ละแชนแนล ต้องแก้ไขด้วยการใส่ R PULL UPไปที่ขา A B C ของ ADC 0809 ก็สามารถแก้ปัญหาได้ อีกส่วนหนึ่งเกิดจากการออกแบบลายปรินท์ในส่วนไฟบวก,ไฟลบ และกราวด์ นั้นลายทองแดงมีขนาดเล็กเกินไป ทำให้เกิดปัญหาดาต้าที่อ่านเข้ามาไม่แน่นอน จะกระเพื่อมตลอดเวลา ต้องแก้ไขโดยการบัดกรีสายไฟลงบนไฟบวก,ไฟลบ และกราวด์ทั้งหมดเพื่อขยายขนาดของผิวทองแดงจึงสามารถแก้ปัญหาได้

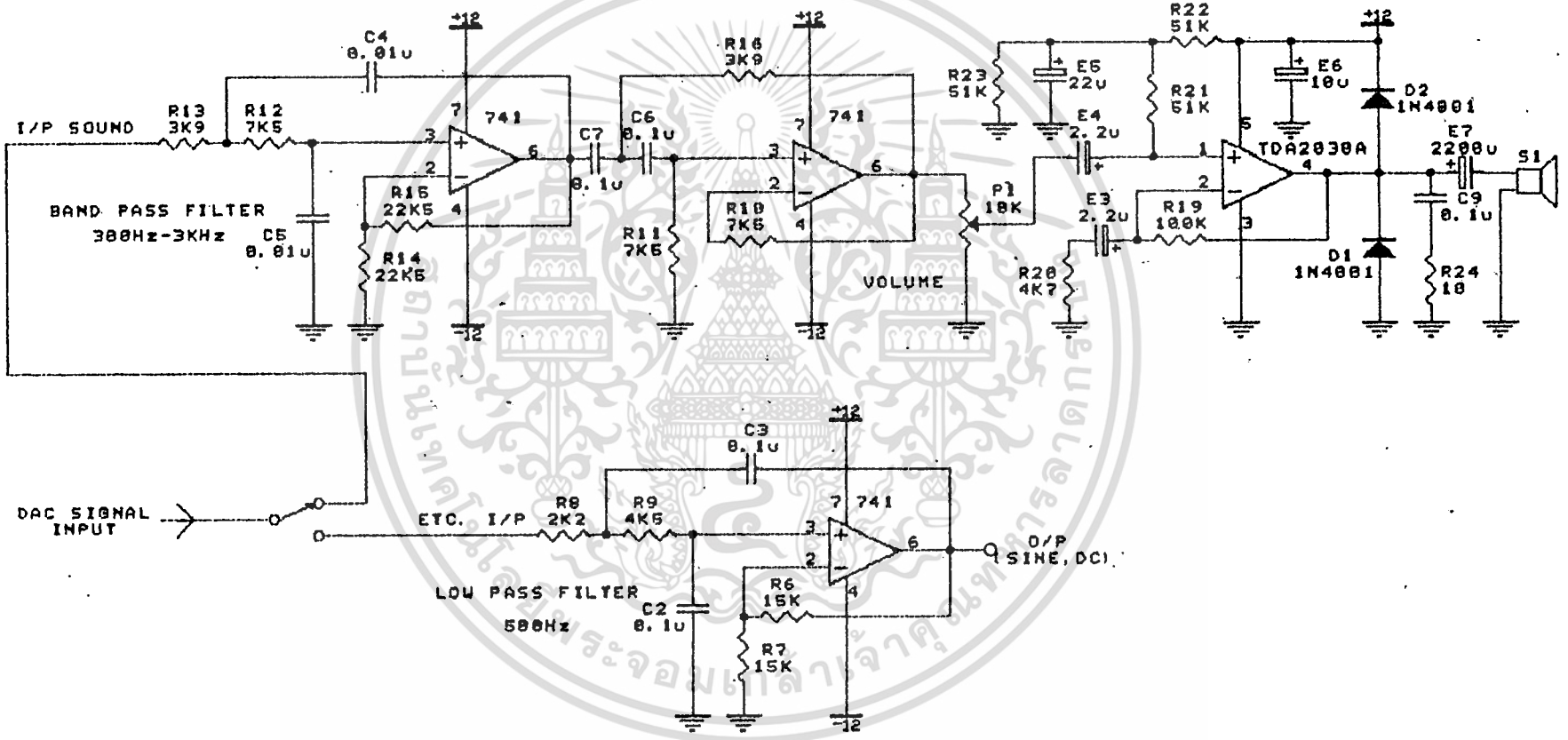
ในอีกปัญหาที่พบก็คือการจัดการกับหน่วยความจำ เนื่องจากเมนบอร์ดของคอมพิวเตอร์ยี่ห้อ "ถาวร"นั้นไม่สามารถใช้งานในส่วนที่เป็น HIGH DOS MEMORY คือ384K ข้างบนได้ ซึ่งระบบปฏิบัติการที่ใช้คือเอ็มเอสดอส 5.0 โดยเมื่อสั่ง HIMEM.SYS ซึ่งในเครื่องทั่วๆ ไปสามารถใช้ได้ แต่กับเมนบอร์ดรุ่นนี้ใช้ไม่ได้ ซึ่งทำให้ระบบทั้งหมดไปรวมอยู่ที่ส่วน BASE MEMORY 640K ซึ่งต้องมีทั้งระบบดอส,เทอร์โบปาสคาล จึงมีพื้นที่เหลือประมาณแค่ 300K กว่าๆ ซึ่งถ้าส่วน 384K ข้างบนใช้ได้แล้วก็จะมีส่วนของเบสเมโมรี่เหลือเพิ่มขึ้นก็จะทำให้สามารถจองหน่วยความจำได้มากขึ้น เวลาที่ใช้ในการเก็บก็จะได้มากขึ้น

ซอฟท์แวร์ ส่วนนี้เกิดปัญหาในการเปลี่ยนไปเปลี่ยนมาระหว่างโหมดกราฟิกส์กับโหมดเท็กซ์ ซึ่งเมื่อเปลี่ยนไปแล้ว ภาพในหน้าจอในโหมดเท็กซ์เดิมจะหายหมด การแก้ปัญหานั้นทำได้โดยการเปิดวินโดวส์ขึ้นอีกหนึ่งวินโดวส์ให้มีขนาดใหญ่กว่าเมนเมนู แล้วจึงเข้าโหมดกราฟิกส์ เมื่อสั่งโครส(CLOSE)กราฟิกส์กลับมายังเท็กซ์โหมด จึงทำการปิดวินโดวส์ เมื่อทำเช่นนี้แล้วก็จะทำให้โปรแกรมกลับสู่เมนเมนูได้โดยไม่มีปัญหา

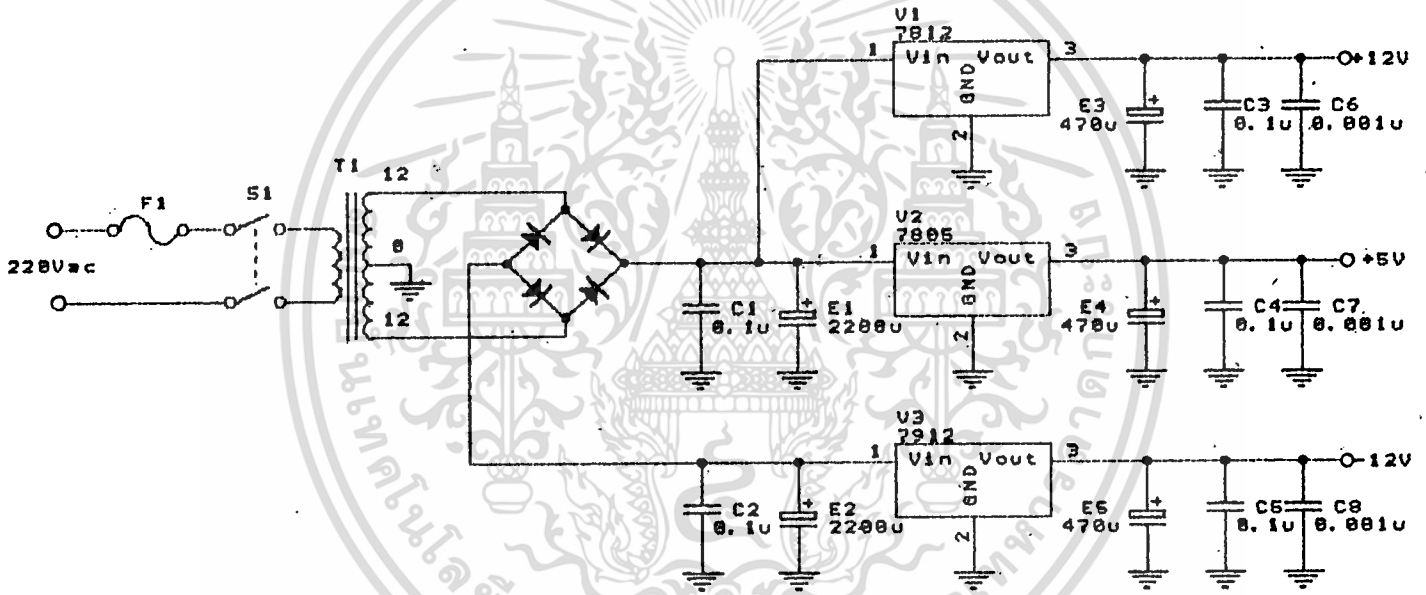

330
330
+5
ADC0809
74LS244
74LS393
IN-0
IN-1
IN-2
IN-3
IN-4
IN-5
IN-6
IN-7
SINE I/P
P3 500K
P4 500K
P5 500K
msb2-1
2-2
2-3
2-4
2-5
2-6
2-7
lsb2-8
EOC
ADD-A
ADD-B
ADD-C
ALE
ENABLE
START
CLOCK
ref(-)
ref(+)
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
D7 [14]
D6 [15]
D5 [16]
D4 [17]
D3 [46]
D2 [45]
D1 [44]
D0 [43]
Q3
Q2
Q1
Q0
MR
CLK
CLK [48]
+12
R1 15k
R2 27k
R4 2k
E1 10u
M1
N1 2SC458
C1 0.01u
R3 18k
R5 1k5
E2 33u
R26 10
555
R
Q
TRIG
DIS
CVolt
THR
10KHz
P2 10K
C10 0.1u
D0
D1
D2
M302
M300
Title
A/D CIRCUIT
Size
A4
Number
PHK10
Revision
Date: 1-JAN 1980
File: B:\ADC/1
Sheet of
Drawn By:



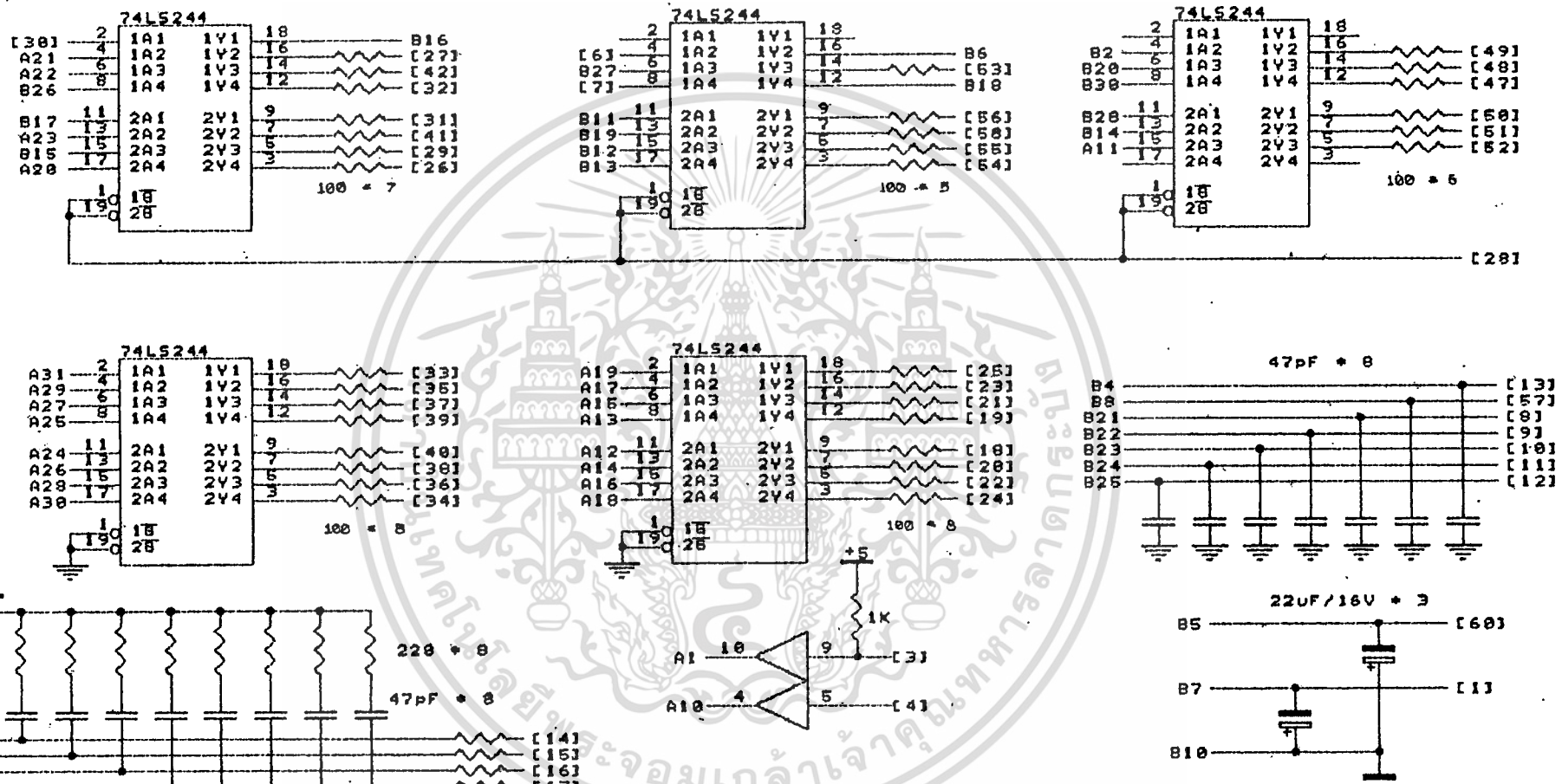
Title	DAC&DECODER CIRCUIT		
Size A4	Number PHK10		Revision
Date: 1-JAN 1980		Sheet of	
File: B:\DAC&DEC01		Drawn By:	

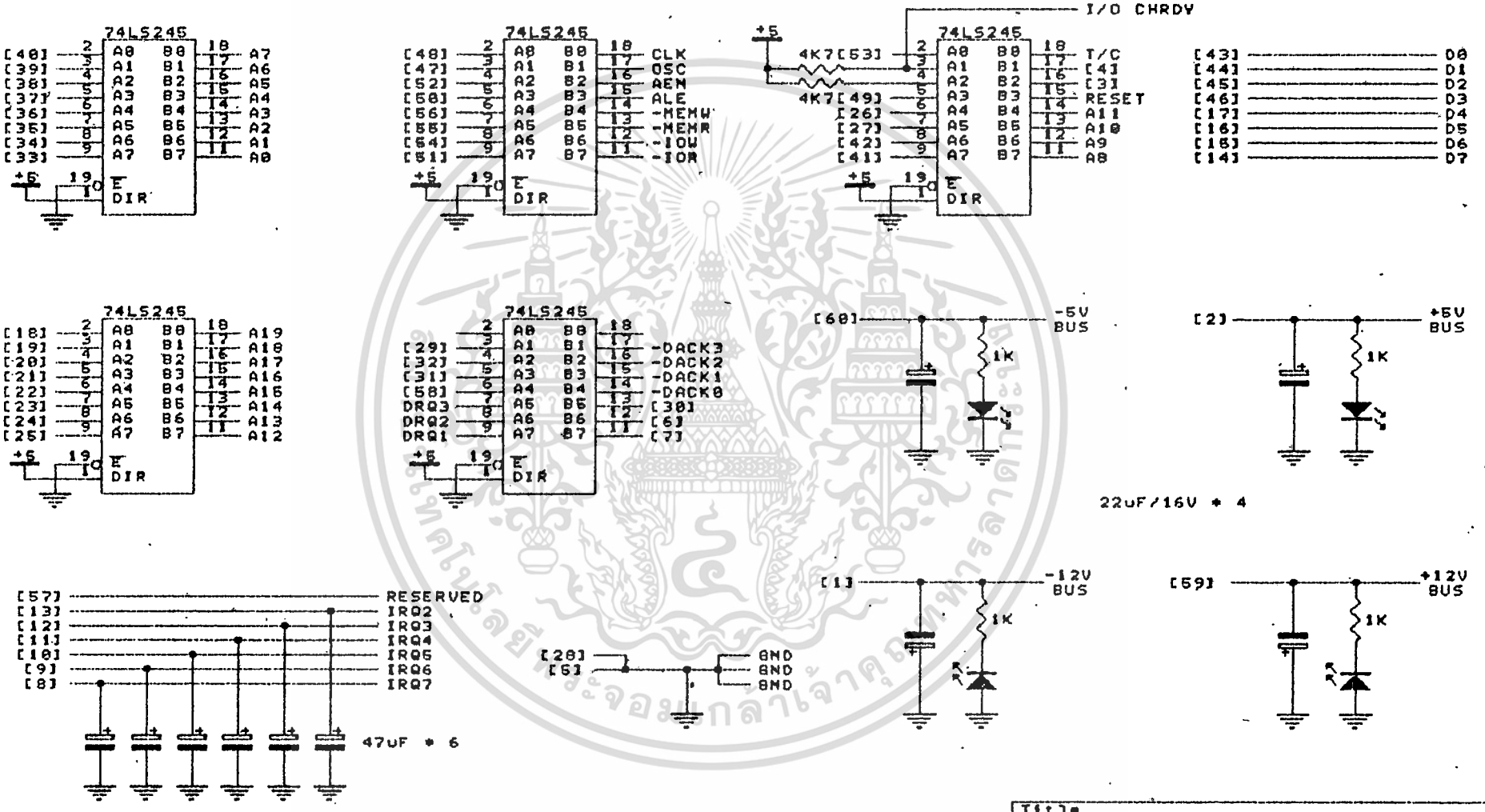


Title	FILTER&AMP CIRCUIT	
Size A4	Number PHK10	Revision
Date: 1-JAN 1980		Sheet of
File: B:\FIL&AMP/1		Drawn By:



Title	POWER SUPPLY CIRCUIT	
Size A4	Number PHK10	Revision
Date: 1-JAN 1980		Sheet of
File: B:\POWER-SU/1		Drawn By:


74LS244
100 * 7
100 * 5
100 * 8
220 * 8
47pF * 8
100 * 8
1K
47pF * 8
22uF/16V * 3
BUFFER CARD CIRCUIT
Size A4
Number PHK10
Revision
Date: 1-JAN 1980
File: B:\BUFF-CAR/1
Sheet of
Drawn By:



Title: LABORATORY BOX		
Size: A4	Number: PHK18	Revision:
Date: 1-JAN 1988		Sheet: of
File: B:\LAB-BOX/1		Drawn By:

```
program pulldownmenu;
uses dos,crt,screen,keyboard,win,graph;
{*************list file****************}


const maxchoice = 15;
      xdim = 5;
      maxmenu = 4;
      dt=100;
      bb=-28000;
      aa=28000;
type  dirptr = ^dirlist;
      fname = string[20];
      dirname =string[12];
      dirlist = record
                  name :dirname;
                  attr : byte;
                  left,
                  right : dirptr;
                end;
      chr2 = array[1..2] of char;
      inp1 = array[bb..aa] of byte;
      F_name = file of inp1;
      string15 = string[15];
      string6 = string[6];
      string80 = string[80];
      structuremenu = record
                        win : array[1..4] of byte;
                        col : array[0..maxchoice] of byte;
                        row : array[0..maxchoice] of byte;
                        msg : array[0..maxchoice] of string80;
                        lastchoice : byte;
                      end;
strucmenu = array[1..maxmenu] of structuremenu;
functionmenu = record
                 col : array[1..10] of byte;
                 row : array[1..10] of byte;
                 msg : array[1..10] of string6;
               end;
const  menu : strucmenu = (
                          (win : (28,09,54,17);
                           col : (28,02,02,02,02,02,02,02,00,00,00,00,00,
0);
                           row : (08,01,02,03,04,05,06,07,00,00,00,00,00,
0);
                           msg : ('         MAIN  MENU          ',
                                  ' ANALOG TO DIGITAL       ',
                                  ' DIGITAL TO ANALOG       ',
                                  ' MULTIPLEX I/P CHANNEL   ',
                                  ' READ IN MEMORY          ',
                                  ' READ & WRITE TO FILE    ',
                                  ' READ FROM FILE          ',
                                  ' QUIT                    ',
                                  '                         ');
                           lastchoice : 7),
                          (win : (19,02,40,07);
                           col : (19,02,02,02,02,00,00,00,00,00,00,00,00,
0);
                           row : (01,01,02,03,04,00,00,00,00,00,00,00,00,
0);
                           msg : ('      menu2              ',
                                  '      number 1           ',
```

```
                                    '   number 2   ',
                                    '   number 3   ',
                                    '   number 4   ',
                                    '              ');
                                lastchoice : 4),
                               (win : (36,02,60,06);
                                col : (39,02,02,02,00,00,00,00,00,00,00,00,00,0
0);
                                row : (01,01,02,03,00,00,00,00,00,00,00,00,00,0
0);
                                msg : (' menu3 ',
                                       ' number 1 ',
                                       ' number 2 ',
                                       ' number 3 ',
                                       '          ');
                                lastchoice : 3),
                               (win : (59,02,80,06);
                                col : (59,02,02,02,00,00,00,00,00,00,00,00,00,0
0);
                                row : (01,01,02,03,00,00,00,00,00,00,00,00,00,0
0);
                                msg : (' menu4 ',
                                       ' number 1 ',
                                       ' number 2 ',
                                       ' number 3 ',
                                       '          ');
                                lastchoice : 3)
                                );
fkey : functionmenu = (col : (01,09,17,25,33,41,49,57,65,72);
                       row : (25,25,25,25,25,25,25,25,25,25);
                       msg : (' func1 ',
                              ' func2 ',
                              ' func3 ',
                              ' func4 ',
                              ' func5 ',
                              ' func6 ',
                              ' func7 ',
                              ' func8 ',
                              ' func9 ',
                              ' quit ')
                        );
const choicemenu  : array[1..maxmenu] of 1..maxchoice = (1,1,1,1);
      statusmenu : 1..maxmenu = 1;
      finish      : boolean    = false;
var   key,ch          : char;
      i,j,m:integer;
      dircount  : byte;
      dirinfo : searchrec;
      headnode : dirptr;
      pointer : dirptr;
      name,path : string;
      inp : inp1;
      FiPtr :F_name;
      filename:fname;
      check,dsp:boolean;
      grdriver,grmode : integer;
{$I a:model.pas}
{$I a:dohelp.pas}
{$I a:a_d.pas}
{$I barchart.pas}
{*****************************************************************}
```

```
function getnode : dirptr;
var p :dirptr;
begin
   new(p);
   getnode := p;
end;

procedure freedir(var p : dirptr);
begin
   dispose(headnode);
end;

procedure initheader;
begin
   headnode := getnode;
   headnode^.left := headnode;
   headnode^.right:= headnode;
end;

procedure insertright(atnode : dirptr;name:dirname;attr:byte);
var newnode,temp : dirptr;
begin
   newnode := getnode;
   newnode^.name := name;
   newnode^.attr := attr;
   temp := atnode^.right;
   temp^.left := newnode;
   newnode^.right := temp;
   newnode^.left := atnode;
   atnode^.right := newnode;
end;

procedure insertafter(name : dirname;attr:byte);
var last:dirptr;
begin
   last := headnode^.left;
   insertright(last,name,attr);
end;

procedure searchdir(name:dirname;var post:byte;var pt:dirptr);
begin
   post := 0;
   pt := headnode;
   repeat
      post := post+1;
      pt := pt^.right;
   until ( (name = pt^.name) or (pt = headnode));
   if pt = headnode then post := 0 ;
end;

procedure displaydir(xindex,yindex : byte);
var tail :char;
    col,row : byte;
begin
   col := (xindex-1)*14+2;
   row := yindex;
   gotoxy (col,row);
   if pointer^.attr = directory then tail :='\' else tail := ' ';
   write (pointer^.name,tail,' ':11-length(pointer^.name));
end;
```

```
procedure putdirtolist;
begin
   with dirinfo do
   begin
      if( (attr and sysfile) = 0) then
      begin
         insertafter(name,attr);
         dircount := dircount + 1;
      end;
   end;
end;


procedure listdir(mask : string);
begin
   initheader;
   dircount := 0;
   findfirst(mask,anyfile,dirinfo);
   while doserror = 0 do
   begin
      findnext(dirinfo);
      if (doserror = 0) then putdirtolist;
   end;
end;

procedure maptomatrix(size,post : byte;var x,y : byte);
var i : byte;
begin
   x := (post mod size);
   y := (post div size);
   if (x = 0) then x := size;
   if (post mod size) <> 0 then  y := y+1;
   pointer := headnode;
   for i := 1 to post do pointer := pointer^.right;
end;

procedure dir(mask : string);
var xindex,yindex,post : byte;
begin
   listdir(mask);
   for post := 1 to dircount do
   begin
      maptomatrix(xdim,post,xindex,yindex);
      displaydir(xindex,yindex);
   end;
end;

function getmatrix(xdim,num,post:byte;var pt : dirptr): dirname;
var xindex,yindex,currentpost : byte;
    key : char;
    selectok : boolean;

procedure displayname(old,new :byte);
var x,y : byte;
begin
   maptomatrix(xdim,old,x,y);
   setattr(lowdisplay);
   displaydir(x,y);
   maptomatrix(xdim,new,x,y);
   setattr(reverselow);
   displaydir(x,y);
end;
```

```
procedure nameup;
begin
   if post > xdim then
   begin
      currentpost := post;
      post := post - xdim;
      displayname(currentpost,post);
   end;
end;

procedure namedn;
begin
   if (post + xdim <= num) then
   begin
      currentpost := post;
      post := post + xdim;
   end;
end;

procedure namelf;
begin
   if post <> 1 then
   begin
      currentpost := post;
      post := post -1;
   end;
end;

procedure namert;
begin
   if ( (post +1) <= num) then
   begin
     currentpost := post;
     post := post +1;
   end;
end;

procedure firstname;
begin
   currentpost := post;
   post := 1;
end;

procedure lastname;
begin
  currentpost := post;
  post := num;
end;

procedure returnvaule;
begin
  pt := pointer;
  getmatrix := pointer^.name;
  selectok := true;
end;

procedure exitesc;
begin
   getmatrix := #27;
   selectok := true;
end;
```

```
{************** main of function getmatrix ***************** }
begin
  setattr(reverselow);
  currentpost := post;
  displayname(post,post);
  selectok := false;
  repeat
    readfunckey(key);
    if funckey then
       begin
          case key of
             up_key : nameup;
             lt_key : namelf;
             rt_key : namert;
             dn_key : namedn;
             home_key : firstname;
             end_key : lastname;
          end;
         displayname(currentpost,post);
       end
    else
       case key of
         return_key :
         begin
            returnvaule;
          {  load_file;}
         end;
         esc_key : exitesc;
       end;
  until selectok;
end;
{************ end of function getmatrix ***************}
function getpath :string;
var path : string;
begin
   getdir(0,path);
   if path[length(path)]='\' then
      getpath := path
   else
      getpath := path +'\';
end;

procedure readdir (mask : string;var path,name: string);
type string80 = string[80];
var post,p,xindex,yindex : byte;
    n : dirname;
    curdir : string80;
    startdir : string;
    pt : dirptr;
    finished : boolean;
begin
   cursoroff;
   getdir(0,startdir);
   setwinheader('directory of ' +getpath+mask);
   windowopen(04,08,78,22);
   post := 1;
   curdir := ' ';
   finished := false;
   repeat
      listdir(mask);
      if dircount = 0 then listdir('*.*');
      for p := 1 to dircount do
```

```
      begin
         maptomatrix(xdim,p,xindex,yindex);
         displaydir(xindex,yindex);
      end;
      searchdir(curdir,post,pt);
      if post = 0 then post := 1;
      n := getmatrix(xdim,dircount,post,pt);
      case n[1] of
        #27 : begin
                 name := #27;
                 finished := true;
              end;
        '.' : begin
                 getdir(0,curdir);
                 p := length(curdir);
                 repeat p := p - 1; until curdir[p] = '\';
                 curdir := copy (curdir,p+1,length(curdir)-p+1);
                 chdir('..');
              end
       else
              begin
                if (pt^.attr and directory) = directory then
                begin
                   chdir(n);
                   curdir := '..';
                end
                else
                begin
                   name := n;
                   path := getpath;
                   finished := true;
                end;
              end;
        end;
        freedir(headnode);
    until  finished;
    chdir(startdir);
    windowclose;
end;
{****************************************}
procedure choiceactive(old,new:byte);
begin
  with menu[statusmenu] do
  begin
    setattr(lowdisplay);
    gotoxy(col[old],row[old]);
    write(msg[old]);
    setattr(reverselow);
    gotoxy(col[new],row[new]);
    write(msg[new]);
  end;
end;

{****************************************}
procedure menuactive(new : byte);
var i : byte;
begin
  setattr(lowdisplay);
  setwinattr(lowdisplay);
  setattr(reverselow);
  setboxattr(lowdisplay);
  setcharattr(lowdisplay);
```

```
  with menu[new] do
  begin
    gotoxy(col[0],row[0]);
    write(msg[0]);
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setcharattr(lowdisplay);
    setboxstyle(single);
    setwinheader('');
    windowopen(win[1],win[2],win[3],win[4]);
    for i := 1 to lastchoice do
      begin
        gotoxy(col[i],row[i]);
        write(msg[i]);
      end;
    end;
    choiceactive(1,choicemenu[new]);
end;

{**********************************}

{$R+}
procedure inmem;
const   max_row = 300;
        max_col = 350;
        dt = 100;
type    row_array = array[1..max_col] of byte;
        row_ptr = ^row_array;
        matrix = array[1..max_row] of row_ptr;
var     result : matrix;
        index:integer;
        row,col :integer;
        rnd,k,l:integer;
        carac:char;

begin
    port[$302]:=$04;
    setattr(lowdisplay);
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setcharattr(lowdisplay);
    setboxstyle(double);
    setwinheader('');
    windowopen(23,13,80,18);
    write('    HOW MANY ROUND THAT YOU WANT TO PLAY : ');
    cursoroff;
    readln(rnd);
    clrscr;cursoron;
    writeln(' WHEN YOU READY TO SPEAK');
    writeln(' PRESS ANY KEY AND SPEAK TO RECORD YOUR SOUND IN MEMORY');
    carac:=readkey;
       clrscr;i := bb;
       writeln(' SPEAK!   NOW....');
{       repeat
            inp[i] := port[$300];
            i := i+1;
            for j := 1 to dt do
       until (i>aa);}
  {************ in data ***************}
    for index := 1 to max_row do
        new(result[index]);
```

```
          j:=0;
        for row := 1 to max_row do
           begin
           for col :=  1 to max_col do
              begin
              result[row]^[col] := port[$300];
              for k := 1 to dt do
              end;
           end;

    {***************** end in data ********************}

        writeln(' PRESS ANY KEY TO PLAY YOUR SOUND ',rnd,' ROUND');
        carac:=readkey;
        clrscr;
        writeln('                          PLAY SOUND ',rnd,' ROUND');


{        for m := 1 to rnd do
        for i:=bb to aa do
           begin
           port[$301] := inp[i];
           for j := 1 to dt do
           end;
        end;
}
         for l := 1 to rnd do
            begin
            for row := 1 to max_row do
               begin
               for col := 1 to max_col do
                  begin
                  port[$301] := result[row]^[col];
                  for k :=1 to dt do
                  end;
               end;
            end;
        for index := 1 to max_row do
          dispose(result[index]);
          j:=0;
        for row := 1 to max_row do
           begin
           for col :=  1 to max_col do
           end;

    windowclose;
end;
{*********************************}
procedure InData;
begin
  for i:=bb to aa do
    begin
    inp[i]:=port[$300];
         for j := 1 to dt do
    end;
end;

{*********************************}
procedure ovwrite(filename : fname);
var carac:char;
begin
    setattr(lowdisplay);
```

```
     setwinattr(lowdisplay);
     setboxattr(lowdisplay);
     setcharattr(highdisplay);
     setboxstyle(double);
     setwinheader('');
     windowopen(25,13,80,17);writeln(' WHEN YOU READY TO SPEAK');
     writeln(' PRESS ANY KEY AND THEN SPEAK');
     writeln(' PRESS ESC TO EXIT');
     carac:=readkey;
     if carac <> #27 then
        begin
        clrscr;
        writeln(' SPEAK NOW !');
        InData;
        clrscr;writeln(' PRESS ANY KEY TO WRITE YOUR SOUND IN THE FILE');
        writeln(' PRESS ESC TO EXIT');
        ch:=readkey;
        if ch <> #27 then
              begin
              write(FiPtr,inp);
              close(FiPtr);windowclose;
              end
           else
             begin
             close(FiPtr);windowclose;
             end;
        end
        else
          begin
          close(FiPtr);windowclose;
          end;
end;
{*********************************}
procedure OverWrite(filename : fname);
begin
         assign(FiPtr,filename);
         {$I-}
         ReWrite(FiPtr);
         {$I+}
         ovwrite(filename);
end;


{*********************************}
procedure FindName(filename : fname);
begin
       setattr(lowdisplay);setwinattr(lowdisplay);setboxattr(lowdisplay);
       setcharattr(lowdisplay);setboxstyle(double);
       setwinheader('verify');
       windowopen(28,13,48,15);
       gotoxy(1,1);write(' Overwrite (Y/N)');
       ch:=readkey;
       windowclose;
       if (ch = 'y') or (ch = 'Y') then OverWrite(filename);
end;


{*********************************}
procedure listdirwrite(filename : fname);
var cou,tru:integer;
begin
   initheader;
```

```
    dircount := 0;tru:=0;
    findfirst(filename,anyfile,dirinfo);
    if doserror = 0 then
        begin
        tru:=3;
        FindName(filename)
        end
        else
          begin
          listdir('*.');
          for cou:=1 to dircount do
             begin
             findnext(dirinfo);
             if doserror = 0 then tru := 1
                else tru :=0;
             end;
          end;
    if tru <> 3 then
    if tru=1 then findname(filename)
       else overwrite(filename);
end;

{***************************}
procedure readfilename(var mask:fname);
begin
   while mask = ''  do
      begin
      clrscr;
      readln(mask);
      end;
end;
{***************************}
procedure WinReadFileName(var filename:fname);
begin
    setattr(lowdisplay);setwinattr(reverselow);setboxattr(lowdisplay);
    setcharattr(reverselow);setboxstyle(double);
    setwinheader('Input File Name');
    windowopen(28,13,60,15);filename:='';
    readfilename(filename);
    windowclose;
end;

procedure WriteFile;
var y,n:char;
begin
    port[$302]:=04;
    WinReadFileName(filename);
    listdirwrite(filename);
end;
{***************************}
procedure Readfile;
const dl = 80;
var c,k:integer;
begin
  port[$302]:=04;
  winreadfilename(filename);
  assign(FiPtr,filename);
  {$I-}
  reset(FiPtr);
  {$I+}
  if IOresult = 0 then
    begin
```

```
    reset(FiPtr);
    read(FiPtr,inp);
    setattr(lowdisplay);setwinattr(lowdisplay);setboxattr(lowdisplay);
    setcharattr(highdisplay);setboxstyle(double);
    setwinheader('');
    windowopen(28,13,60,15);
    write('      HOW  MANY ROUNDS : ');
    read(m);clrscr;write('            ','RUN ',M,' ROUND');
    for c:=1 to m do
    for i:=bb to aa do
      begin
      port[$301]:=inp[i];
      for j:=1 to dt do;
      end;
    close(FiPtr);
    windowclose;
    end
    else
    begin
       setattr(lowdisplay);setwinattr(lowdisplay);setboxattr(lowdisplay);
       setcharattr(highdisplay);setboxstyle(double);
       setwinheader('');
       windowopen(28,13,60,16);
       writeln('         FILE NOT FOUND');
       write('           PRESS ENTER ');
       readln;windowclose;
    end;
end;

{*********************************}

procedure InOut(i:integer);
  begin
    clrscr;
    i := bb;
    writeln('When you speak  your sound will out to speaker ');
    writeln('if you will out of program PRESS ANYKEY');
    repeat
         inp[i] := port[$300];
         port[$301] := inp[i];
         i := i+0;
         for j := 1 to dt do
    until (i>aa) or (keypressed);
  end;

{*********************************}

  procedure shownumber(i:integer);
  var j:integer;
  begin
    clrscr;writeln('press enter to goto menu');
      for j := bb to aa do
        begin
        repeat
        inp[j] := port[$300];
        writeln(inp[j]);
        port[$301] := inp[j];j := j+1;
        until (j > aa) or (keypressed);
        end;
  end;

{*********************************}
```

```
procedure quit;
begin
  setattr(highdisplay);
  windowclose;
  cursoron;
  clrscr;
  finish := true;
end;

{*********************************}
procedure displaymainmenu;
var i : byte;
begin
  window(1,1,80,25);
  setattr(lowdisplay);
  for i := 1 to maxmenu do

    with menu[i] do
    begin
      gotoxy(col[0],row[0]);
      write(msg[0]);
    end;
end;


{procedure DispInfor(currentchoice:byte);
begin
    setattr(lowdisplay);
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setcharattr(lowdisplay);
    setboxstyle(single);
    setwinheader('');
  windowopen(1,22,80,24);write('jfjfgjkfjk');readln;
    windowclose;

end;}

{*****************************************}
procedure moveup;
var currentchoice : byte;
begin
  currentchoice := choicemenu[statusmenu];
  {DispInfor(currentchoice);}
  if currentchoice = 1 then
     choicemenu[statusmenu] := menu[statusmenu].lastchoice
  else
    choicemenu[statusmenu] := currentchoice - 1;
  choiceactive(currentchoice,choicemenu[statusmenu]);
end;

{*****************************************}
procedure movedown;
var currentchoice : byte;
begin
  currentchoice := choicemenu[statusmenu];
  if currentchoice = menu[statusmenu].lastchoice then
    choicemenu[statusmenu] := 1
  else
    choicemenu[statusmenu] := currentchoice + 1;
  choiceactive(currentchoice,choicemenu[statusmenu]);
end;
```

```
{*****************************************}
procedure movetofirstchoice;
begin
  choiceactive(choicemenu[statusmenu],1);
  choicemenu[statusmenu] := 1;
end;
{*****************************************}
procedure movetolastchoice;
begin
  with menu[statusmenu] do begin
   choiceactive(choicemenu[statusmenu],lastchoice);
   choicemenu[statusmenu] := lastchoice;
  end;
end;

procedure barchart1;
begin
    setwinattr(lowdisplay);
    setboxattr(nodisplay);
    windowopen(1,1,80,25);
    barchart;
    windowclose;
end;

procedure d_a1;
begin
    setboxattr(nodisplay);
    windowopen(1,1,80,25);
    d_a;
    windowclose;

end;
procedure a_d1;
begin
    setwinattr(lowdisplay);
    setboxattr(nodisplay);
    windowopen(1,1,80,25);
    a_d;
    windowclose;
    closegraph;
end;

{*****************************************}
procedure DoHelp;
begin
    writeln('Help');
end;
{*****************************************}
procedure processmenu(num : byte);
var ch :char;
begin
  setwinattr(reverselow);
  setboxattr(reverselow);
  setcharattr(reverselow);
  setboxstyle(single);
  {windowopen(20,09,50,14);
  gotoxy(08,02);
  write('process menu ',num);
  gotoxy(08,3);
  write ('option number '+chr(choicemenu[statusmenu]+48));
  ch := readkey;}
```

```
  if (chr(choicemenu[statusmenu]+48) = '1') then a_d1
       else if (chr(choicemenu[statusmenu]+48) = '2') then d_a1
       else if (chr(choicemenu[statusmenu]+48) = '3') then barchart1
       else if (chr(choicemenu[statusmenu]+48) = '4') then inmem
       else if (chr(choicemenu[statusmenu]+48) = '5') then writefile
       else if (chr(choicemenu[statusmenu]+48) = '6') then readfile
       else if (chr(choicemenu[statusmenu]+48) = '7') then quit
       else num:=num;
end;

{****************************************}
procedure domenu1;
begin
  processmenu(1);
end;

{**********************}
{procedure domenu2;
begin
  processmenu(2);
end;}

{**********************}
{procedure domenu3;
begin
  processmenu(3);
end;}

{**********************}
{procedure domenu4;
begin
  processmenu(4);
end;}

{**********************}
procedure domenu;
begin
  case statusmenu of
    1  :  domenu1;
  end;
end;

{**********************}
procedure processfunction(num : byte);
var ch : char;
begin
  setwinattr(reverselow);
  setboxattr(reverselow);
  setcharattr(reverselow);
  setboxstyle(single);
  windowopen(25,12,55,14);
  gotoxy(08,02);
  write('    process function ',num);
  ch := readkey;
  windowclose;
end;
{**************************}
procedure do_help;
begin
    if (chr(choicemenu[statusmenu]+48) = '1') then help1
       else if (chr(choicemenu[statusmenu]+48) = '2') then help2
       else if (chr(choicemenu[statusmenu]+48) = '3') then help3
```

```
        else if (chr(choicemenu[statusmenu]+48) = '4') then help4
        else if (chr(choicemenu[statusmenu]+48) = '5') then help5
        else if (chr(choicemenu[statusmenu]+48) = '6') then help6
        else if (chr(choicemenu[statusmenu]+48) = '7') then help7
        else check:=check;
end;


{**************************}
procedure testkey(key : char);
begin
  if funckey then
    case (key) of
        up_key     : moveup;
        dn_key     : movedown;
        pgup_key   : movetofirstchoice;
        pgdn_key   : movetolastchoice;
        f1_key     : Do_Help;
    end
  else
    if key = return_key then domenu;
end;

procedure dispfunckey;
begin
    gotoxy(1,25);
    setattr(reverselow);
    write('F1-HELP');
end;


begin
  setboxattr(nodisplay);
  windowopen(2,2,79,24);
  titlecom;
  windowclose;
  setattr(lowdisplay);clrscr;
  directvideo := true;
  checksnow   := true;
  dispfunckey;
  menuactive(1);
  repeat
    cursoroff;
    readfunckey(key);
    testkey(key);
  until finish;
  cursoron;
end.
```

```
procedure clsscreen(x1,y1,x2,y2:integer);
var oldfill:fillsettingstype;
begin
     getfillsettings(oldfill);
     setfillstyle(0,oldfill.color);
     bar(x1,y1,x2,y2);
     setfillstyle(oldfill.pattern,oldfill.color);
end;
procedure opengraph;
begin
    grdriver:=detect;
    initgraph(grdriver,grmode,'');
end;
procedure LineBlock;
begin
    line(0,0,getmaxx,0);
    line(getmaxx,0,getmaxx,getmaxy);
    line(getmaxx,getmaxy,0,getmaxy);
    line(0,getmaxy,0,0);
end;
procedure ByteToHex(inp:byte;var data:chr2);
var dummyh,dummyl:byte;
begin
    dummyh:=inp and $F0;
    dummyl:=inp and $0F;
    case dummyh of
       $00 : data[1]:='0';
       $10 : data[1]:='1';
       $20 : data[1]:='2';
       $30 : data[1]:='3';
       $40 : data[1]:='4';
       $50 : data[1]:='5';
       $60 : data[1]:='6';
       $70 : data[1]:='7';
       $80 : data[1]:='8';
       $90 : data[1]:='9';
       $A0 : data[1]:='A';
       $B0 : data[1]:='B';
       $C0 : data[1]:='C';
       $D0 : data[1]:='D';
       $E0 : data[1]:='E';
       $F0 : data[1]:='F';
    end;
    case dummyl of
       $00 : data[2]:='0';
       $01 : data[2]:='1';
       $02 : data[2]:='2';
       $03 : data[2]:='3';
       $04 : data[2]:='4';
       $05 : data[2]:='5';
       $06 : data[2]:='6';
       $07 : data[2]:='7';
       $08 : data[2]:='8';
       $09 : data[2]:='9';
       $0A : data[2]:='A';
       $0B : data[2]:='B';
       $0C : data[2]:='C';
       $0D : data[2]:='D';
       $0E : data[2]:='E';
       $0F : data[2]:='F';
    end;
```

```
end;

procedure hextobyte(var data:chr2;var outport:byte);
var op1,op2:byte;
begin
   case data[1] of
       '0'  : op1 :=0;
       '1'  : op1 :=16;
       '2'  : op1 :=32;
       '3'  : op1 :=48;
       '4'  : op1 :=64;
       '5'  : op1 :=80;
       '6'  : op1 :=96;
       '7'  : op1 :=112;
       '8'  : op1 :=128;
       '9'  : op1 :=144;
   'a','A'  : op1 :=160;
   'b','B'  : op1 :=176;
   'c','C'  : op1 :=192;
   'd','D'  : op1 :=208;
   'e','E'  : op1 :=224;
   'f','F'  : op1 :=240;
        end;
   case data[2] of
       '0'  : op2 :=0;
       '1'  : op2 :=1;
       '2'  : op2 :=2;
       '3'  : op2 :=3;
       '4'  : op2 :=4;
       '5'  : op2 :=5;
       '6'  : op2 :=6;
       '7'  : op2 :=7;
       '8'  : op2 :=8;
       '9'  : op2 :=9;
   'a','A'  : op2 :=10;
   'b','B'  : op2 :=11;
   'c','C'  : op2 :=12;
   'd','D'  : op2 :=13;
   'e','E'  : op2 :=14;
   'f','F'  : op2 :=15
      end;
      outport:=op1+op2;
end;

procedure framepicture;
var colornum:word;
begin
    lineblock;

{*******  small rectangle  *********}
    rectangle(205,190,395,310);
    setpalette(colornum,$f);
    setcolor($9);
    setbkcolor($8);
    settextstyle(1,0,3);
    outtextxy(225,150,'DIGITAL DATA');
end;
procedure a_d;
var data:chr2;
    x,y,i,j,k:integer;
    curr_pall:palettetype;
    colornum:word;
```

```
    color:shortint;
    inp:byte;
    ch:char;
begin
    opengraph;
    port[$302]:=$00;
    x:=getmaxx;y:=getmaxy;
    framepicture;
    setcolor(red);
    settextstyle(4,0,5);
    outtextxy(183,70,'A / D LAB');
    setcolor(yellow);
    settextstyle(1,0,1);
    outtextxy(60,350,'PRESS ANY KEY WHEN YOU WANT TO LOOK AT THE DATA');
    repeat
       repeat
          inp:=port[$300];
          ByteToHex(inp,data);
          setcolor(green);
          settextstyle(1,0,10);
          outtextxy(230,180,data);
          clsscreen(210,400,400,425); {clear test again}
          clsscreen(207,191,393,308);
       until keypressed;
       ch:=readkey;
       outtextxy(230,180,data);
       setcolor(magenta);
       settextstyle(1,0,2);
       outtextxy(220,400,'TEST AGAIN (Y/N)');
       repeat
          ch:=readkey;
       until (ch='y') or (ch='Y') or (ch='n') or (ch='N');
    until (ch<>'y') and (ch<>'Y');
    closegraph;
end;

procedure d_a;
{type  hexa}
const hexnum:set of '0'..'z' = ['0'..'9','a'..'f','A'..'F'];
var data:chr2;
    x,y,i,j,k:integer;
    curr_pall:palettetype;
    colornum:word;
    color:shortint;
    inp:byte;
    ch:char;
    outport:byte;
begin
    clrscr;
    opengraph;
    framepicture;
    setcolor(red);
    settextstyle(4,0,5);
    outtextxy(183,70,'D / A LAB');
    setcolor(yellow);
    settextstyle(1,0,3);
    outtextxy(160,350,'ENTER DIGITAL DATA = ');
    repeat
    i:=1;
       repeat
          repeat{for i:=1 to 2 do}
             data[i]:=readkey;
```

```
            data[i]:=upcase(data[i]);
            settextstyle(1,0,3);
            setcolor(yellow);
            if data[i] in hexnum then
               begin
                  check:=true;
                  if i=1 then
                       begin
                       outtextxy(430,350,data[i]);
                       i:=i+1;
                       end
                       else
                       begin
                       outtextxy(450,350,data[i]);
                       i:=i+1;
                       end;
               end
               else
                  begin
                  check:=false;
                  clsscreen(429,350,470,375);
                  i:=1;
                  end;
         until i>2;
      until check=true;
      setcolor(green);
      settextstyle(1,0,10);
      outtextxy(230,180,data);
      settextstyle(1,0,2);
      hextobyte(data,outport);
      setcolor(magenta);
      outtextxy(210,400,'TEST AGAIN (Y/N)');
      repeat
         port[$301]:=outport;
         ch:=readkey;
      until (ch='y') or (ch='Y') or (ch='n') or (ch='N');
      clsscreen(429,350,470,375); {clear hexnum}
      clsscreen(207,191,393,308); {clear big alphanumeric}
      clsscreen(210,400,395,425); {clear test again}
   until (ch<>'y') and (ch<>'Y');
   closegraph;
end;
```

```
procedure barchart;
type  inpu = set of 1..3;
const {numbars = 4;}          {set now}
      depth = 0;
      step_a = 90;
      step_b = 80;
      inp1 : inpu = [1..3];
var
     DATA : ARRAY[0..10] OF INTEGER;
     graphdriver,graphmode:integer;
     width,height,numbar_a,inp:integer;
     l,x1,y1,x2,y2,xmax,ymax,ganx,frame,frame_a,i,j,k,q,r:integer;
     numbars,st:string[1];
     y3 :real;
     ch : char;
     check: boolean;
begin
      clrscr;{windowopen(2,2,79,24);}
      setwinheader('');
      setboxstyle(double);
      setcharattr(highdisplay);
      windowopen(35,12,60,14);
      clrscr;
      write(' HOW MANY INPUT NOW : ');
      repeat
      readln(inp);
        if inp in inp1 then
           check:=true
           else
             begin
             check:=false;
             write(' HOW MANY INPUT NOW : ');
             end;
      until check;
      windowclose;
     graphdriver := detect;
     initgraph(graphdriver,graphmode,'');
{     xmax:=getmaxx;
     ymax:=getmaxy;}
     setbkcolor(lightblue);
     width := 80   {xmax div (numbars * 2)};
     height := ymax-(ymax div 4);
     setlinestyle(solidln,0,thickwidth);
     line(25,5,25,450);

     line(25,450,620,450);
     moveto(1,10); outtext('255');          {distance step = 90}
     moveto(1,100);outtext('200');
     moveto(1,190);outtext('150');
     moveto(1,280);outtext('100');
     moveto(9,370);outtext('50');
     moveto(13,460);outtext('0');
     settextstyle(4,0,2);
     outtextxy(350,220,'PRESS ANY KEY TO EXIT');
     settextstyle(smallfont,horizdir,6);
     ganx := 90;
      for k:= 0 to inp-1 do
       begin
       str(k:1,st);
       moveto(ganx,460);outtext('IN');          {distance = 80}
       moveto(ganx+20,460);outtext(st);
       ganx:=ganx+80;
```

```
        end;
    {********* read input ********}
       repeat
       for i := 0 to inp-1 do
       begin
        port[$302] := i;
        delay(5);
        data[i] := port[$300];
       end;
       i := 0;
      {***** display  input*******}
     for l := 1 to inp do
       begin
          x1 := width*(1+(l-1));
          y1 := 450;
          x2 := x1+(width div 2);
          y3 := 450 - (data[i]*1.7);
          y2 := trunc(y3);
          inc(i);
          bar3d(x1,y1,x2,y2,depth,topoff);
       end;
       inc(i);
     delay(50);
     for i := 1 to inp do
     begin
     setfillstyle(solidfill,lightblue);
     bar(x1-1,y1-2,x2+1,y2-2);
     x1:=x1-80;x2:=x2-80;y2:=y2*0;
     end;
     until keypressed;
     ch := readkey;

  closegraph;

end;
```

```
procedure help1;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 1 ');
    setcharattr(highdisplay);
    windowopen(20,5,60,18);
    writeln;
    writeln('This menu is the Analog to Digital Lab.');
    writeln('     This menu simulates the analog to');
    writeln(' digital converter that concist of the');
    writeln(' conversion  from  the  analog data to');
    writeln(' the hexadecimal data (digital data) .');
    writeln(' The analog data is read from the port');
    writeln(' and it is converted to  the hexadeci-');
    writeln(' mal number and display on the monitor');
    writeln(' by this program.');
    ch:=readkey;
    windowclose;
end;
procedure help2;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 2 ');
    setcharattr(highdisplay);
    windowopen(20,5,60,18);
    writeln;
    writeln('This menu is the Digital to Analog Lab.');
    writeln('    This menu simulates the digital to');
    writeln(' analog  converter that consist of the');
    writeln(' conversion  from  the digital data to');
    writeln(' the analog data . The digital data is');
    writeln(' read from the user   ( the user enter');
    writeln(' the  digital data  by press  the key-');
    writeln(' board  )  .  The program will out the');
    writeln(' digital  data  to  the D/A  converter');
    writeln(' through the port . The analog data is');
    write(' acording with the digital data');
    ch:=readkey;
    windowclose;
end;
procedure help3;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 3 ');
    setcharattr(highdisplay);
    windowopen(20,5,61,16);
    writeln;
    writeln('This menu is the A/D application program');
    writeln('     This menu is the program that dis-');
    writeln(' play the data from each channel of A/D');
    writeln(' converter . You will test this program');
    writeln(' by rotate the volume of each input ch-');
    writeln(' annel, the analog data will display on');
```

```
    write(' the monitor.                                ');
    ch:=readkey;
    windowclose;
end;
procedure help4;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 4 ');
    setcharattr(highdisplay);
    windowopen(20,5,61,18);
    writeln('This is read the data to  memory program');
    writeln('   This menu is the program  that reads');writeln;
    writeln(' the  data from  the input port and out');writeln;
    writeln(' and write to the memory .The data will');writeln;
    writeln(' be out to the output port (D/A conver-');writeln;
    writeln(' through the amplifier and the speaker.');
    ch:=readkey;
    windowclose;
end;

procedure help5;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 5 ');
    setcharattr(highdisplay);
    windowopen(20,5,61,18);
    writeln('This is a  read & write to file program.');
    writeln('   This menu is the program that writes');writeln;
    writeln(' the sound in the disk. The A/D conver-');writeln;
    writeln(' ter is sampling the sound and converts');writeln;
    writeln(' to  the digital data  and  the program');writeln;
    write(' writes it in the disk .');
    ch:=readkey;
    windowclose;
end;

procedure help6;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 6 ');
    setcharattr(highdisplay);
    windowopen(20,5,61,18);
    writeln('   This is a  read from file program');writeln;
    writeln('   This menu is the program  that reads');
    writeln(' the sound in the disk that  is written');
    writeln(' by  the read & write to file program ,');
    writeln(' The program reads the sound,that is in');
    writeln(' digital form ,and out to the D/A conv-');
    writeln(' ert  port to  convert the digital data');
    writeln(' to sound(analog form).');
    writeln(' The sound  is  amplified  by  the amp-');
    writeln(' lifier that is in the module.');
    ch:=readkey;
```

```
    windowclose;
end;
procedure help7;
var ch:char;
begin
    setwinattr(lowdisplay);
    setboxattr(lowdisplay);
    setboxstyle(mix1);
    setwinheader(' Help Menu 7 ');
    setcharattr(highdisplay);
    windowopen(20,5,60,15);
    writeln;
    writeln('          This menu is the quit          ');
    writeln('     When you run this menu , you will');writeln;
    writeln(' you will exit from the main menu      ');writeln;
    ch:=readkey;
    windowclose;
end;
```

```
{program interfacegraph;}
procedure  titlecom;
var
     count,test : integer;
     ddd : word;
     ch:char;
begin
     count := 0;
     grdriver := detect;
     initgraph(grdriver,grmode,'');
     setbkcolor(5);
     setcolor(lightblue);
     rectangle(400,70,530,170);
     setcolor(white);
     settextstyle(triplexfont,horizdir,2);
     outtextxy(410,80,'MODULE');
     outtextxy(410,100,'BOARD');
     outtextxy(410,120,'INTERFACE');
     setlinestyle(solidln,0,thickwidth);
     setcolor(yellow);
     arc(300,350,250,334,70);
     arc(300,380,250,360,80);
     line(130,361,272,455);
     line(170,361,276,417);
     setcolor(white);
     setcolor(yellow);
     line(135,335,165,335);
     line(135,350,165,350);
     arc(250,160,70,135,70);
     arc(250,190,70,135,70);
     arc(350,90,250,315,70);
     arc(350,60,250,315,70);
     line(273,95,327,127);
     line(273,125,327,157);
     line(65,237,200,110);
     line(100,237,200,140);
{    outtextxy(122,180,'b');}    {display position}
     setcolor(green);
     rectangle(350,280,490,380);
     setcolor(white);
     outtextxy(360,290,'PROTO');
     outtextxy(360,310,'BOARD');
     outtextxy(360,330,'EXPERIMENT');
     setcolor(lightgray);
     line(50,240,50,320);
     line(55,237,160,237);
     line(165,240,165,320);
     line(55,323,160,323);
     line(50,240,55,237);
     line(160,237,165,240);
     line(165,320,160,323);
     line(50,320,55,323);
     setcolor(lightgray);
     rectangle(40,324,175,360);
     setcolor(white);
     outtextxy(56,245,'COMPUTER');
     outtextxy(56,270,'CONTROL ');
     setlinestyle(solidln,0,normwidth);
     outtextxy(490,430,'Press anykey');
     test:=1;
     setcolor(green);
     repeat
```

```
      line(200,200,270,190);
      line(200,200,200,180);
      line(200,180,170,210);
      line(170,210,205,230);
      line(205,230,200,210);
      line(200,210,270,200);
      line(270,200,275,220);
      line(275,220,305,190);
      line(270,190,265,170);
      line(265,170,305,190);
      {********** arrow top *****}
      line(230,350,300,360);
      line(230,350,232,370);
      line(232,370,212,340);
      line(212,340,236,310);
      line(236,310,234,330);
      line(234,330,302,340);
      line(302,340,300,320);
      line(300,320,320,350);
      line(300,360,298,380);
      line(298,380,320,350);
      if test = 1 then
        begin
         setcolor(magenta);
         test := 0;
        end
      else
        begin
        setcolor(green);
        test := 1;
        end;
      delay(100);
      until keypressed;
      ch:=readkey;
      closegraph;
end;
```

```
unit screen;
interface
uses crt,dos;
  const   nodisplay        = $00;
          lowdisplay       = $07;
          highdisplay      = $0f;
          underlinelow     = $01;
          underlinehigh    = $09;
          reverselow       = $70;
          reversehigh      = $78;
          blinklow         = $87;
          blinkhigh        = $8f;
          undblinklow      = $81;
          undblinkhigh     = $89;
          revblinklow      = $f0;
          revblinkhigh     = $f8;
          colorseg         = $b800;
          monoseg          = $b000;
   var    videoseg         : word;
          crttype          : byte absolute $0040:$0049;
          cursormode       : word absolute $0040:$0060;
          vport            : word absolute $0040:$0063;

 procedure setattr (attrib : byte);
 procedure setcursor (top,bottom : byte);
 procedure cursoron;
 procedure cursoroff;

{ end of interface section }

implementation

   var regs : registers;

   procedure setattr (attrib : byte);
   begin
     textattr := attrib;
   end;
 procedure setcursor (top,bottom : byte);
 begin
     regs.ah := 1;    { function set cursor mode }
     regs.ch := top;
     regs.cl := bottom;
     intr($10,regs);
 end;

 procedure cursoron;
 begin
     port[vport]   := 10;
     port[vport+1] := hi(cursormode) and $df;
     port[vport]   := 11;
     port[vport+1] := lo(cursormode);
 end;

 procedure cursoroff;
 begin
     port[vport]   := 10;
     port[vport+1] := hi(cursormode) or $20;
     port[vport]   := 11;
     port[vport+1] := lo(cursormode)
 end;
```

```
procedure identifycrt;
begin
    case crttype of
       0..3        : videoseg := colorseg;
          7        : videoseg := monoseg;
    end;
end;

begin
    identifycrt;
end. { of unit }
```

```
unit keyboard;
interface
uses crt;
   var   key     :  char;
         funckey :  boolean;
   const return_key  = #13;   esc_key  =  #27;
         shift_tab   = #15;
         alt_q_key  =  #16; alt_w_key  = #17;
         alt_e_key  =  #18; alt_r_key  = #19;
         alt_t_key  =  #20; alt_y_key  = #21;
         alt_u_key  =  #22; alt_i_key  = #23;
         alt_o_key  =  #24; alt_p_key  = #25;

         alt_a_key  =  #30; alt_s_key  = #31;
         alt_d_key  =  #32; alt_f_key  = #33;
         alt_g_key  =  #34; alt_h_key  = #35;
         alt_j_key  =  #36; alt_k_key  = #37;
         alt_l_key  =  #38;

         alt_z_key  =  #44; alt_x_key  = #45;
         alt_c_key  =  #46; alt_v_key  = #47;
         alt_b_key  =  #48; alt_n_key  = #49;
         alt_m_key  =  #50;

         f1_key  = #59; f2_key  = #60;
         f3_key  = #61; f4_key  = #62;
         f5_key  = #63; f6_key  = #64;
         f7_key  = #65; f8_key  = #66;
         f9_key  = #67; f10_key = #68;

         home_key  = #71; up_key   = #72;
         pgup_key  = #73; lt_key   = #75;
         rt_key    = #77; end_key  = #79;
         dn_key    = #80; pgdn_key = #81;
         ins_key   = #82; del_key  = #83;

         shift_f1_key  = #84; shift_f2_key  = #85;
         shift_f3_key  = #86; shift_f4_key  = #87;
         shift_f5_key  = #88; shift_f6_key  = #89;
         shift_f7_key  = #90; shift_f8_key  = #91;
         shift_f9_key  = #92; shift_f10_key = #93;

         ctrl_f1_key  = #94;  ctrl_f2_key  = #95;
         ctrl_f3_key  = #96;  ctrl_f4_key  = #97;
         ctrl_f5_key  = #98;  ctrl_f6_key  = #99;
         ctrl_f7_key  = #100; ctrl_f8_key  = #101;
         ctrl_f9_key  = #102; ctrl_f10_key = #103;

         alt_f1_key  = #104; alt_f2_key  = #105;
         alt_f3_key  = #106; alt_f4_key  = #107;
         alt_f5_key  = #108; alt_f6_key  = #109;
         alt_f7_key  = #110; alt_f8_key  = #111;
         alt_f9_key  = #112; alt_f10_key = #113;

         ctrl_prtsc_key  = #114; ctrl_lt_key   = #115;
         ctrl_rt_key     = #116; ctrl_end_key  = #117;
         ctrl_pgdn_key   = #118; ctrl_home_key = #119;

         alt_1_key  = #120; alt_2_key  = #121;
         alt_3_key  = #122; alt_4_key  = #123;
         alt_5_key  = #124; alt_6_key  = #125;
         alt_7_key  = #126; alt_8_key  = #127;
```

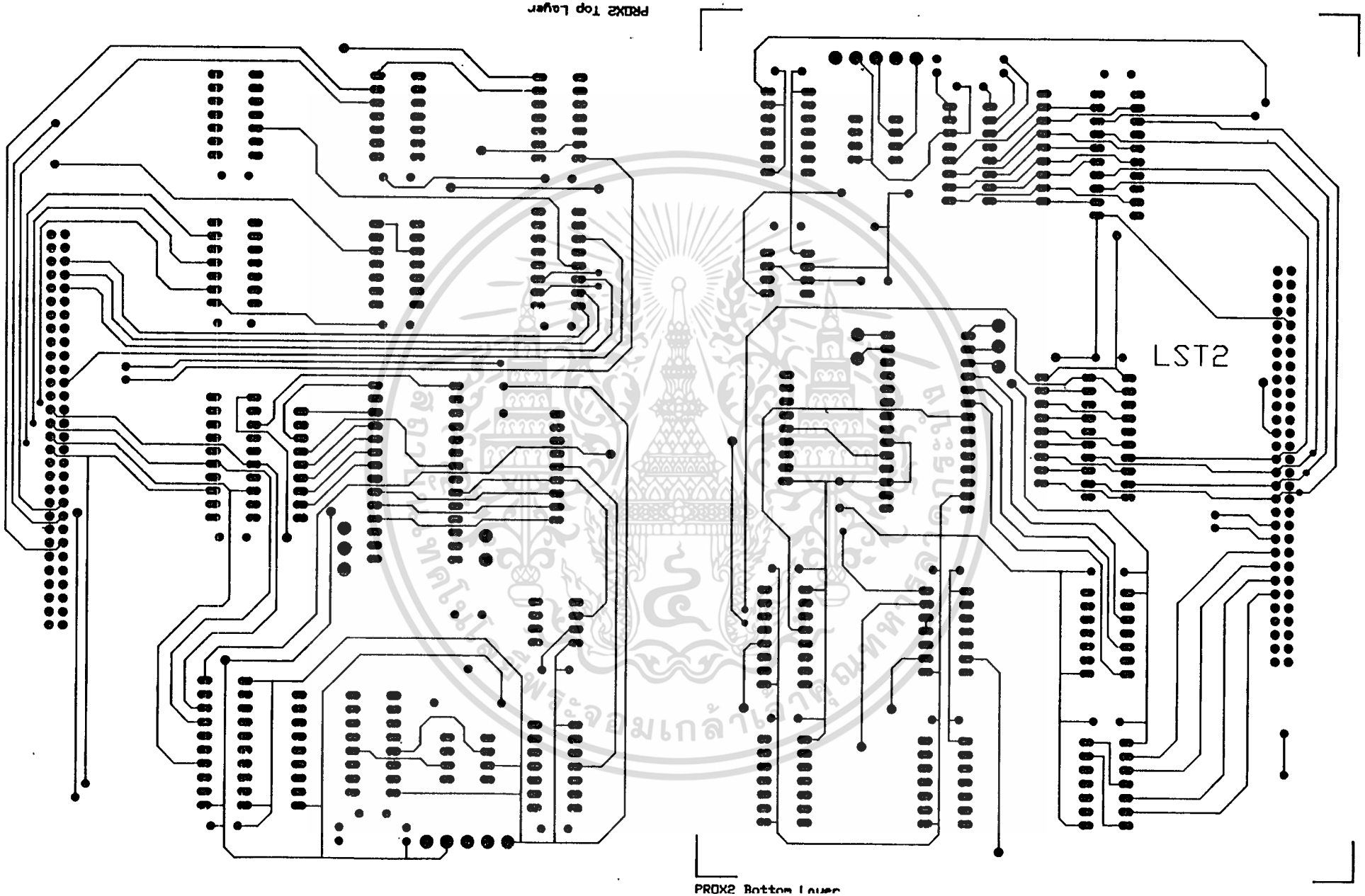
```
          alt_9_key  = #128; alt_10_key = #129;

          ctrl_pgup_key  = #132;
          f11_key        = #133; f12_key        = #134;
          shift_f11_key  = #135; shift_f12_key  = #136;
          ctrl_f11_key   = #137; ctrl_f12_key   = #138;
          alt_f11_key    = #139; alt_f12_key    = #140;
procedure readfunckey (var key :char);
procedure readstring(var st :string);
procedure readintnum(var num,code :integer);
procedure readrealnum(var num :real ;var code : integer);
implementation
  procedure readfunckey(var key :char);
  begin
    key := readkey;
    if key = #0 then
       begin
          funckey := true;
          key     := readkey;
       end
     else
       funckey  := false;
  end;

  procedure  mapkey(var key :char);
  begin
     if funckey then
        case key  of
          home_key  : key := ^a;
          end_key   : key := ^z;
          lt_key    : key := ^s;
          rt_key    : key := ^d;
          del_key   : key := ^g
      else
          key := #00;
      end;
  end;
  procedure backspace(var st :string; var ptr :byte);
   var tail  : string;
   begin
     if ptr <> 1 then
     begin
       tail  :=  copy (st,ptr,length(st)-ptr+1);
       delete (st,ptr-1,1);
       ptr  := ptr-1;
       write (^h);
       clreol;
       write (tail);
       gotoxy(wherex-length(tail),wherey);
     end;
  end;
  procedure leftarrow(var ptr :byte);
  begin
     if  ptr <> 1 then
     begin
        ptr := ptr-1;
        write(^h);
     end;
  end;
  procedure rightarrow (st :string;var ptr : byte);
  var len : byte;
  begin
```

```
     len := length(st);
     if (ptr <= len) and (len <> 0) then
     begin
        ptr := ptr+1;
        gotoxy(wherex+1,wherey);
     end;
end;
procedure tohome (var ptr : byte);
begin
   gotoxy(wherex-ptr+1,wherey);
   ptr := 1;
end;
procedure toend (st :string; var ptr : byte);
var len : byte;
begin
  len := length(st);
  if ptr <= len then
  begin
     tohome(ptr);
     gotoxy(wherex+len,wherey);
     ptr := len + 1;
  end;
end;
procedure truncate (var st :string;var ptr :byte);
begin
  st  := copy (st,1,ptr-1);
  ptr := length(st)+1;
  if ptr = 1 then st := '';
  clreol;
end;
procedure del (var st :string;var ptr : byte);
var tail : string;
    len  : byte;
begin
  len := length(st);
  if (ptr <= len) and (len <> 0) then
  begin
    tail := copy (st,ptr+1,length(st)-ptr);
    delete (st,ptr,1);
    clreol;
    write (tail);
    gotoxy(wherex - length(tail),wherey);
  end;
end;
procedure clear (var st :string;var ptr : byte);
begin
  tohome (ptr);
  clreol;
  st := '';
end;
procedure character(var st:string; ch : char;var ptr : byte);
var p,x,len : byte;
begin
  len := length(st);
  if ptr > len then
    begin
      if (wherex <= (lo(windmax)-lo(windmin))) then
        begin
          st := st+ch;
          ptr := ptr+1;
         write (ch);
        end;
```



```
setattr(attrofbox);
gotoxy(x1,y1);
```

```
    write(typeofbox[boxstyle,upleft]);
    for x := x1+1 to x2-1 do
        write(typeofbox[boxstyle,top]);
    write(typeofbox[boxstyle,upright]);
    for y := y1+1 to y2-1 do begin
        gotoxy(x1,y);write(typeofbox[boxstyle,left]);
        gotoxy(x2,y);write(typeofbox[boxstyle,right]);
    end;
    gotoxy(x1,y2);
    write(typeofbox[boxstyle,loleft]);
    for x := x1+1 to x2-1 do
         write(typeofbox[boxstyle,bottom]);
    write(typeofbox[boxstyle,loright]);
    setattr(attrofheader);
    gotoxy((x1+x2-length(headerofwindow)) div 2,y1);
    write(headerofwindow);
    window(x1+1,y1+1,x2-1,y2-1);
    setattr(attrofchar);
end;
procedure windowopen(x1,y1,x2,y2:byte);
var  block       : windowlink;
     linelength,windowsize,i: integer;
     y   : byte;
begin
    linelength  := x2-x1+1;
    windowsize := linelength*(y2-y1+1)*2+fixedsize;
    if (x2 > 80) or (y2 > 25) or (x2-x1 < 2) or (y2-y1 < 2) then
       errorwindow  := 1
    else
       if (abs(memavail) < windowsize) then
           errorwindow := 2
       else
           errorwindow := 0;
    if errorwindow = 0 then
       begin
         getmem (block,windowsize);
         block^.x1   := x1;
         block^.x2   := x2;
         block^.y1   := y1;
         block^.y2   := y2;
         block^.x    := wherex;
         block^.y    := wherey;
         block^.backlink  :=  activewindow;
         activewindow     :=  block;
         windowcount      :=  windowcount + 1;
         block^.id        :=  windowcount;
         i := 1;
         for y := y1 to y2 do begin
            move(screenptr^[y,x1],block^.screencontents[i],linelength*2);
            i := i + linelength;
         end;
       windowbox(x1,y1,x2,y2);
    end;
 end;
 procedure windowclose;
 var  block    : windowlink;
      linelength,windowsize,i  : integer;
      y : byte;
begin
    if activewindow <> nil then
    begin
      block      := activewindow;
      linelength := block^.x2-block^.x1+1;
```


PROX2 Top Layer
LST2
PROX2 Bottom Layer

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้สำเร็จลุล่วงไปได้ด้วยดี ก็ต้องขอขอบคุณ อาจารย์ภากร หุตะสังกาศ และเพื่อนๆ ที่คอยให้กำลังใจในยามที่เกิดปัญหาในทุกๆ ด้าน จึงขอขอบคุณมา. ณ.ที่นี้อีกครั้ง

หนังสืออ้างอิง

1. ซีเอ็ดยูเคชั่น,คู่มือ/เทียบเบอร์ไอซี TTL,บริษัท ซีเอ็ดยูเคชั่น จำกัด,338 หน้า,2529
2. เปรมจิตร วิสุทธิศิริ,พื้นฐานวงจร A/D & D/A,เซมิคอนดักเตอร์ อิเล็กทรอนิกส์,บริษัท ซีเอ็ดยูเคชั่น จำกัด,หน้า 272 - 279,ฉบับ 102,2533
3. เปรมจิตร วิสุทธิศิริ,พื้นฐานวงจร A/D & D/A,เซมิคอนดักเตอร์ อิเล็กทรอนิกส์,บริษัท ซีเอ็ดยูเคชั่น จำกัด,หน้า 302 - 309,ฉบับ 103,2533
4. IBM,PC/AT Technical Reference,IBM Corp.,Personal Computer,1984
5. KENT PORTER,Streching Turbo Pascal 5.5,SIMON & SCHUSTER,421 หน้า,1990
6. National Semiconductor,DATA BOOK 2,National Semiconductor Corp.,1989