ระบบวัดความเข้มของคลื่นวิทยุ

FIELD STRENGTH MEASURING SYSTEM

โดย

นายวระ	ศิวะบวร	37014378
นายวิศิษฎ์	รัตนชัยฤทธิ์	37014414
นายวีระศักดิ์	จงไพโรจน์โฆษิต	37014423

อาจารย์ที่ปรึกษา

ผ.ศ.นิภา ลีลารุจิ

ร.ศ.ณรงค์ เหมกรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ระบบวัดความเข้มของคลื่นวิทยุ

FIELD STRENGTH MEASURING SYSTEM

โดย

นายวระ ศิวะบวร

นายวิศิษฎ์ รัตนชัยฤทธิ์

นายวีระศักดิ์ จงไพโรจน์โมษิต

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ระบบวัดความเข้มของคลื่นวิทยุ

FIELD STRENGTH MEASURING SYSTEM

โดย

นายวระ	ศิวะบวร	37014378
นายวิศิษฎ์	รัตนชัยฤทธิ์	37014414
นายวีระศักดิ์	จงไพโรจน์โมษิต	37014423

อาจารย์ที่ปรึกษา

ผ.ศ.นิภา ฉลารุจิ

ร.ต.ณรงค์ เหมกรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบวัดความเข้มของคลื่นวิทยุ

FIELD STRENGTH MEASURING SYSTEM

ผู้จัดทำ

1. นายวระ ศิวะบวร 37014378
2. นายวิศิษฏ์ รัตนชัยฤทธิ์ 37014414
3. นายวีระศักดิ์ จงไพโรจน์โฆษิต 37014423

อาจารย์ที่ปรึกษา

(ผศ.นิภา ฉัตรุจิ)

อาจารย์ที่ปรึกษา

(รศ.ณรงค์ เหมกรณ์)

ระบบวัดความแข็งของคลื่นวิทยุ

Field Strength Measuring System

โดย นายวระ ศิวะบวร 37014378

นายวิศิษฏ์ รัตนชัยฤทธิ์ 37014414

นายวีระศักดิ์ จงไพโรจน์โฆษิต 37014423

อาจารย์ที่ปรึกษา ผศ.นิภา ลีลารุจิ

รศ.ณรงค์ เหมกรณ์

บทคัดย่อ

โครงงานนี้เป็นการสร้างอุปกรณ์วัดความเข้มของคลื่นวิทยุเอฟเอ็ม ซึ่งมีความถี่อยู่ในช่วง 87.5-108 MHz โดยใช้สายอากาศกวาดรับสัญญาณจากทิศทางต่าง ๆ หลักการทำงาน จะใช้สายอากาศเป็นตัวรับสัญญาณ และใช้สเต็ปปิ้งมอเตอร์เป็นตัวขับสายอากาศให้หมุนไปรอบ ๆ ทีละสเต็ป ความแรงของสัญญาณที่รับได้จากสายอากาศในแต่ละสเต็ป จะถูกเปลี่ยนเป็นสัญญาณดิจิตอลแล้วส่งไปให้คอมพิวเตอร์โดยผ่านทางสล็อตของคอมพิวเตอร์ เพื่อนำข้อมูลความแรงของสัญญาณไปแสดงผลทางหน้าจอคอมพิวเตอร์ ในส่วนของซอฟท์แวร์ที่ใช้ควบคุมการทำงานของอุปกรณ์จะเขียนโดยภาษาซี

ABSTRACT

This project is the instrument that can measure the field strength of frequency modulation wave in range 87.5-108 MHz in any direction. By using stepping motor to step the receiving antenna into 0°-360°. Signal levels in each step are converted into digital signal and tranfer to computer slot to show the value at monitor display. C compiler is used for control devices.

สารบัญ

บทที่ 1
บทนำ

ปกติพื้นฐานของการติดต่อสื่อสารทางไกลนั้น ต้องมีทั้งด้านส่งและด้านรับ โดยด้านส่งจะทำหน้าที่ส่งสัญญาณ (โดยใช้คลื่นความถี่ใด ๆ ที่เหมาะสม) ออกไปในทิศทางที่กำหนด หรือทุกทิศทาง เช่น ระบบกระจายเสียงเอฟเอ็ม (FM Broadcast), ระบบการส่งสัญญาณโทรทัศน์ ส่วนทางด้านรับนั้นก็จะทำหน้าที่รับสัญญาณที่มาจากด้านส่ง ทั้งนี้ไม่ว่าทางด้านส่งจะส่งสัญญาณมาในแบบใด การที่เครื่องรับจะรับสัญญาณได้แรงที่สุดนั้น สายอากาศทางด้านรับจะต้องหันไปทางสถานีส่ง

ผลของการติดตั้งสายอากาศเข้ากับอุปกรณ์โดยที่ไม่สนใจทิศทางของสถานีส่งอาจทำให้เครื่องรับไม่ได้รับสัญญาณที่แรงที่สุด จะทำให้คุณภาพของสัญญาณลดลงไป เช่น สายอากาศของเครื่องรับโทรทัศน์ ถ้าปรับสายอากาศไม่ตรงกับสถานีส่งก็จะทำให้ได้รับสัญญาณภาพ, เสียงไม่ชัดเจน หรือถ้าเป็นเครื่องรับวิทยุเอฟเอ็ม เสียงที่รับได้อาจเบา และมีเสียงรบกวน (เนื่องจากสัญญาณอื่นแรงกว่า) ได้

จากสาเหตุข้างต้นเพื่อให้สายอากาศติดตั้งในทิศทางที่เหมาะสมจึงมีการผลิตอุปกรณ์ที่ใช้หาทิศทางที่คลื่นมีขนาดสัญญาณที่แรงที่สุด โดยใช้หลักการคือ ใช้สายอากาศหันไปในทิศทางใดทิศทางหนึ่งแล้วค่อย ๆ หมุนสายอากาศให้เปลี่ยนทิศทางไปพร้อมๆกับการวัดความเข้มของคลื่นไปด้วย เมื่อสายอากาศหมุนครบรอบก็จะได้ความเข้มของคลื่นทุก ๆ ทิศทาง จากนั้นนำความเข้มในทิศทางต่าง ๆ มาเปรียบเทียบกันก็จะหาทิศทางที่สัญญาณแรงที่สุด (หรือทิศทางของสถานีส่ง) ได้

ในโครงงานนี้จึงเป็นการนำเอาหลักการดังกล่าวมาประยุกต์ใช้ในการทดลองสร้างอุปกรณ์วัดหาความเข้มของคลื่นวิทยุ เรียก " ระบบวัดความเข้มของคลื่นวิทยุ (Field-Strength Measuring System) " โดยจะใช้วัดในช่วงคลื่นกระจายเสียงเอฟเอ็ม (FM Broadcast) ที่มีความถี่ตั้งแต่ 88 ถึง 108 MHz ใช้สายอากาศแบบมีทิศทาง (Directional Antenna) ในการรับคลื่น ใช้สเต็ปปิ้งมอเตอร์เป็นตัวหมุนสายอากาศซึ่งจะทำให้ได้ความแน่นอนสูง การควบคุมสเต็ปปิ้งมอเตอร์จะควบคุมจากคอมพิวเตอร์แทนการควบคุมจากไมโครคอนโทรลเลอร์ และแสดงผลที่ได้จากการวัดทางหน้าจอคอมพิวเตอร์ จะทำให้สะดวกต่อการใช้งานคือสั่งงานและดูผลได้จากหน้าจอคอมพิวเตอร์เลย

ประโยชน์อื่นๆของอุปกรณ์นี้คือสามารถนำไปประยุกต์ใช้หาตำแหน่งของสถานีส่งได้ โดยใช้เครื่องรับ 2-3 ตัว วางให้ห่างกันแล้วทำการวัดหาสัญญาณที่แรงที่สุดจากแต่ละเครื่อง ทิศทางของสัญญาณที่แรงที่สุดและระยะห่างของเครื่องรับแต่ละเครื่องที่สามารถนำมาคำนวณหาตำแหน่งของสถานีส่งได้

บทที่ 2
ทฤษฎี และหลักการ

2.1 การสร้างพอร์ทเพื่ออินเตอร์เฟซ

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเตอร์เฟซเข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล็อตบนเมนบอร์ดนี้จะมีจำนวน 5 สล็อต (สำหรับใน IBM PC/XT จะมี 8 สล็อต) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้าง ๆ ละ 31 ขา ส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้าย หรือขวา) ของสล็อต โดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" และขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของสล็อตขาที่ 24 (นับจากทางด้านท้ายของเครื่อง)

แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่าง ๆ บนเมนบอร์ด ทำให้การสร้างวงจรอินเตอร์เฟซกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอดเดรส (Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมสำหรับการเขียน อ่านข้อมูลจากหน่วยความจำ หรือพอร์ทอินพุทเอาท์พุท เส้นสัญญาณสำหรับการขออินเตอร์รัพท์ของวงจรอินเตอร์เฟซ, เส้นสัญญาณสำหรับการขอเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access : DMA), สัญญาณฐานเวลา (Timing Signal) ต่าง ๆ ที่ใช้ในระบบ, เส้นสัญญาณแสดงการรีเฟรชหน่วยความจำ และสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่าง ๆ ที่ใช้ในระบบอีกด้วย คือ $+5V_{dc}$, $-5V_{dc}$, $+12V_{dc}$ และ $-12V_{dc}$

2.1.1 รายละเอียดเกี่ยวกับสัญญาณต่าง ๆ

OSC (Oscillator; ขา B30)

ขานี้เป็นเอาท์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีความถี่สูงสุดบนเมนบอร์ด คือ 14.31818 MHz ซึ่งมีคาบเวลาประมาณ 70 ns และมีดิวตี้ไซเกิลประมาณ 50% สัญญาณคล็อกอื่น ๆ ของระบบ เช่น คล็อกที่ป้อนให้กับ 8088 หรือชิพซัพพอร์ทต่าง ๆ นั้น จะถูกสร้างขึ้นโดยการหารสัญญาณคล็อกนี้ อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงในการใช้งานสัญญาณ OSC ก็คือ สัญญาณนี้จะไม่ซิงโครไนซ์ (Synchronize) กับสัญญาณอื่น ๆ บนบัสของระบบ ดังนั้นจึงไม่ควรที่จะนำสัญญาณจากขา OSC นี้ไปใช้เป็นสัญญาณคล็อกสำหรับวงจรภายนอกอื่น ๆ ที่ทำงานร่วมกับระบบ

CLK (Clock; ขา B20)

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหารสัญญาณ OSC ด้วย 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz (14.31818 MHz/3) หรือมีช่วงเวลาใน 1 คาบ เท่ากับ 210 ns สำหรับค่าดิวตี้ไซเกิลของสัญญาณนี้จะมีค่าประมาณ 1/3 คือใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิก "1" เท่ากับ 1/3 ของคาบเวลา

ทั้งหมด หรือประมาณ 70 ns และช่วงเวลาที่เป็นลอจิก "0" เท่ากับ 2/3 ของคาบเวลาทั้งหมด หรือประมาณ 140 ns

สัญญาณนี้เป็นสัญญาณที่ถูกใช้เป็นคล็อกของระบบ

RESET DRV (ขา B2) IBM/PC

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และจะยังคงแอคทีฟไปจนกว่าระบบต่าง ๆ ภายใน IBM/PC จะพร้อมที่จะทำงานได้ จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ ถ้าระดับแรงดันของแหล่งจ่ายไฟตกลง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน โดยทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซ็ทวงจรอินเตอร์เฟซหรืออุปกรณ์อินพุทเอาท์พุทต่าง ๆ ในช่วงที่เริ่มจ่ายไฟให้กับระบบ ซึ่งจะเป็นการทำให้วงจร หรืออุปกรณ์เหล่านั้นถูกปรับให้อยู่ในสภาวะที่แน่นอน ก่อนที่จะเริ่มต้นการทำงานในระบบ (สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซ็ท)

A0-A19 (Address Bus ; ขา A31-A12)

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำ หรืออุปกรณ์อินพุทเอาท์พุทที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0-A19 นี้ จะถูกกำหนดโดย 8088 ในระหว่างกระบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำ หรืออุปกรณ์อินพุทเอาท์พุท แต่ในช่วงของขบวนการเข้าถึงหน่วยความจำโดยตรง นั้น ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรง จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbytes แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำพลวัตร (Random Access Memory : RAM) บนเมนบอร์ดที่ถูกใช้โดยระบบ จำนวน 64 kbytes (สำหรับ IBM PC/XT จะเป็นจำนวน 256 kbytes) และแอดเดรสสำหรับหน่วยความจำสถิต (Read Only Memory : ROM) อีก 48 kbytes ซึ่งถูกจัดในช่วงของแอดเดรสบนสุดใน 1 Mbytes คือ 0FC00H จนถึง 0FFFFFH (สำหรับ IBM PC/XT จะเป็น 64 kbytes)

สำหรับการอ้างแอดเดรสของพอร์ทอินพุท/เอาท์พุท นั้นจะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A0-A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64 k พอร์ท โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH เท่านั้น

D0-D7 (Data Bus; ขา A9-A2) :

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ทอินพุท/เอาท์พุท กับ IBM/PC โดยบิท D0 จะมีนัยสำคัญต่ำสุด และบิท D7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกสร้างออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{IOW}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ $\overline{MEMW}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ $\overline{IOW}$ หรือ $\overline{MEMW}$ นี้จะถูกใช้เพื่อสั่งให้ I/O พอร์ท หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้น พอร์ทอินพุท/เอาท์พุท หรือหน่วยความจำที่ถูกอ้างถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{IOR}$ (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท หรือ $\overline{MEMR}$ (ในกรณีที่ต้องการอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น)

ALE (Address Latch Enable; ขา B28) :

ขาสัญญาณนี้เป็นสัญญาณเอาท์พุทที่ตัวควบคุมบัส 8288 (8288 Bus Controller) สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าแอดเดรสจากบัสแอดเดรส/ข้อมูล (Address/Data Bus; AD0-AD7) ของ 8088 ทำให้สามารถแยกค่าแอดเดรส (A0-A19) และข้อมูล (A0-A7) ออกจากกันได้ อย่างไรก็ตามสัญญาณ ALE จะแอคทีฟเฉพาะในบัสไซเคิลที่สร้างขึ้นโดย 8088 เท่านั้น โดยจะไม่แอคทีฟในระหว่างขบวนการเข้าถึงหน่วยความจำโดยตรง

$\overline{I/O\ CHCK}$ (I/O Channel Check; ขา A1) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้ในการแสดงความผิดพลาดเกี่ยวกับพาริตี้ ที่เกิดขึ้นในการทำงานของวงจรอินเตอร์เฟซหรืออุปกรณ์ I/O เมื่อขาสัญญาณนี้ได้รับลอจิก "0" จะทำให้ 8088 ถูกอินเตอร์รัพท์แบบนอน-มาสเคเบิล (Non-Maskable) อย่างไรก็ตามเราสามารถที่จะกำหนดให้วงจรภายในของ IBM/PC ทำการขออินเตอร์รัพท์ (เมื่อได้รับสัญญาณ $\overline{I/O\ CHCK}$) หรือไม่ก็ได้ โดยการกำหนดลอจิกของบิทข้อมูลของพอร์ทที่ควบคุมการขออินเตอร์รัพท์แบบนอน-มาสเคเบิล คือบิท D7 ของพอร์ท 00A0H ในกรณีที่บิท D7 ของพอร์ท 00A0H ถูกเซ็ทเป็น "1" ก็จะทำให้วงจรภายนอกขออินเตอร์รัพท์แบบนอน-มาสเคเบิล ได้ (Enable) แต่ถ้าบิท D7 ของพอร์ท 00A0H ถูกเซ็ทเป็น "0" ก็จะเป็นการดิสเอเบิล (Disable) การขออินเตอร์รัพท์แบบนอน-มาสเคเบิล ดังนี้

Enable : ใช้คำสั่ง OUT ส่งข้อมูล 80H ไปยังพอร์ท 00A0H

Disable : ใช้คำสั่ง OUT ส่งข้อมูล 00H ไปยังพอร์ท 00A0H

และเนื่องจากยังมีอุปกรณ์อื่นที่สามารถขออินเตอร์รัพท์แบบนอน-มาสเคเบิล ได้อีก ดังนั้นซอฟท์แวร์ที่ใช้งานจะต้องสามารถตรวจสอบว่าการขออินเตอร์รัพท์นั้นเกิดขึ้นจากแหล่งใดได้ด้วย

I/O CHRDY (I/O Channel Ready; ขา A10) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือ หน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้น ไม่สามารถทำงานทันตามช่วงเวลาปกติของบัสไซเคิลนั้น ๆ

ได้ (ช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูก หรือ 840 ns ในขณะที่บัสไซเคิลที่เกี่ยวกับ I/O จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 5 ลูก หรือ 1.05 μs)

เมื่ออุปกรณ์ I/O หรือหน่วยความจำต้องการที่จะเพิ่มช่วงเวลาในบัสไซเคิลให้นานขึ้นอีกนั้น จะสามารถทำได้โดยการป้อนลอจิก "0" ให้กับขา I/O CHRDY ในช่วงเวลาที่ I/O หรือ หน่วยความจำที่ถูกกำหนดนั้น ได้รับสัญญาณจากการถอดรหัสแอดเดรส และสัญญาณ $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ หรือ $\overline{\text{IOW}}$ แอดทีฟ

IRQ2-IRQ7 (Interrupt Request 2 Through 7; ขา B4 และ B25-B21) :

ขาสัญญาณทั้ง 6 นี้เป็นขาอินพุทที่ใช้สำหรับการขออินเตอร์รัพท์จาก 8088 โดยสัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรง โปรแกรมในส่วนไบออส ของ IBM/PC จะทำการโปรแกรม 8259A ให้ IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญต่ำสุด ในกรณีที่มีการขออินเตอร์รัพท์เกิดขึ้นคือระดับลอจิกที่ขา IRQ ขาใดขาหนึ่งถูกเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) 8259A ก็จะทำการส่งสัญญาณ INT ให้กับ 8088 เพื่อทำการขออินเตอร์รัพท์

สิ่งสำคัญในการขออินเตอร์รัพท์โดยผ่านทาง IRQ2-IRQ7 นี้ ก็คืออุปกรณ์ที่ทำการขออินเตอร์รัพท์โดยผ่านทาง IRQ ขาใดก็จะต้องรักษาระดับสัญญาณที่ขา IRQ นั้น ให้แอดทีฟ (ลอจิก "1") อยู่จนกว่าจะได้รับสัญญาณ INTA (Interrupt Acknowledge) จาก 8088 เสียก่อน ถ้าไม่เช่นนั้นการขออินเตอร์รัพท์จะถูกยกเลิกและอินเทอร์รัพท์ระดับ 7 (IRQ7) ก็จะถูกสร้างขึ้นโดยอัตโนมัติ ไม่ว่าการขออินเตอร์รัพท์ที่ถูกยกเลิกนั้นจะเป็นการขออินเตอร์รัพท์ใน Level หรือขาใด

แต่อย่างไรก็ตามสัญญาณ INTA นี้จะไม่ถูกต่อออกมาที่ขาของสล็อตด้วย ดังนั้นโปรแกรมที่ทำการตอบสนองต่อการขออินเตอร์รัพท์ (Interrupt Service Routine) จะต้องทำการรีเซ็ทสัญญาณ IRQ เอง โดยใช้คำสั่ง OUT ไปยังพอร์ทอินพุท/เอาท์พุท ที่เกี่ยวข้อง

$\overline{\text{IOR}}$ (I/O Read; ขา B14) :

ขาสัญญาณนี้เป็นเอาท์พุทแอดทีฟที่ลอจิก "0" ที่สร้างขึ้นโดยตัวควบคุมบัส 8288 เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ทอินพุท/เอาท์พุท เพื่อให้พอร์ทอินพุท/เอาท์พุท ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น ส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ $\overline{\text{IOR}}$ ประมาณ 30 ns เพื่อให้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการเข้าถึงหน่วยความจำโดยตรง ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรง 8237A-5 (8237A-5 DMA Controller) จะทำการสร้างสัญญาณ $\overline{\text{IOR}}$ เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ทอินพุท/เอาท์พุท) ที่พอร์ทอินพุท/เอาท์พุท ที่ขอเข้าถึงหน่วยความจำโดยตรง ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ทใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะอาศัยสัญญาณ DACK จาก ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรง Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอดทีฟก็จะแสดงว่าพอร์ทอินพุท/เอาท์พุท ที่จะต้องส่งข้อมูลออกมาบนบัสข้อมูลก็คือพอร์ทอินพุท/เอาท์พุท ที่ขอเข้าถึงหน่วยความจำโดยตรง ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

$\overline{\text{IOW}}$ (I/O Write; ขา B13) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดยตัวควบคุมบัส 8288 เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ทอินพุท/เอาท์พุท เพื่อให้พอร์ทอินพุท/เอาท์พุท ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ $\overline{\text{IOW}}$ นี้แอคทีฟ (ลอจิก "0") นั้นข้อมูลบนบัสข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ $\overline{\text{IOW}}$ แทนขอบขาลงในการทำให้พอร์ทอินพุท/เอาท์พุท ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการเข้าถึงหน่วยความจำโดยตรง นั้น ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรงจะทำการสร้างสัญญาณ $\overline{\text{IOW}}$ เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ทอินพุท/เอาท์พุท ที่ขอเข้าถึงหน่วยความจำโดยตรงต้องการจะอ่านข้อมูล

$\overline{\text{MEMW}}$ (Memory Write; ขา B11) :

ขานี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งตัวควบคุมบัส 8288 สร้างขึ้นในระหว่างบัสไซเคิลในการเขียนข้อมูลลงในหน่วยความจำของ 8088 สัญญาณ $\overline{\text{MEMW}}$ นี้จะถูกส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการรับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ โดยทั่วไปหน่วยความจำจะรับข้อมูลในช่วงขอบขาขึ้นของสัญญาณ $\overline{\text{MEMW}}$

สำหรับในระหว่างขบวนการเข้าถึงหน่วยความจำโดยตรงนั้น ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรง จะทำการควบคุมบัสต่าง ๆ ของระบบแทน 8088 และสัญญาณ $\overline{\text{MEMW}}$ จะถูกใช้ในบัสไซเคิลของการเขียนข้อมูลลงในหน่วยความจำ (ข้อมูลถูกส่งจากอุปกรณ์ I/O ไปให้กับหน่วยความจำ)

$\overline{\text{MEMR}}$ (Memory Read; ขา B12) :

ขานี้เป็นเอาท์พุทจาก 8088 ซึ่งสัญญาณนี้จะแอคทีฟ (ลอจิก "0") ในระหว่างบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำของ 8088 เพื่อให้หน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการส่งข้อมูลออกมาบนบัสข้อมูล โดยหน่วยความจำนั้นจะต้องส่งข้อมูลออกมาในช่วงเวลา 30 ns ก่อนที่สัญญาณ $\overline{\text{MEMW}}$ จะกลับเป็นลอจิก "1" ทั้งนี้ก็เพื่อให้ 8088 ได้รับข้อมูลที่ถูกต้อง

สำหรับในระหว่างขบวนการเข้าถึงหน่วยความจำโดยตรง นั้น ตัวควบคุมขบวนการเข้าถึงหน่วยความจำโดยตรง จะควบคุมบัสต่าง ๆ ของระบบแทน 8088 และสัญญาณ $\overline{\text{MEMR}}$ จะถูกใช้ในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำ (ข้อมูลถูกส่งจากหน่วยความจำไปให้กับอุปกรณ์ I/O)

DRQ1-DRQ3 (DMA Request 1-3; ขา B18, B6 และขา B16) :

ขาสัญญาณทั้งสามนี้เป็นสัญญาณอินพุทแอคทีฟที่ลอจิก "1" ซึ่งอุปกรณ์ภายนอกสามารถใช้ในการขอเข้าถึงหน่วยความจำโดยตรงจากระบบ โดยการป้อนระดับสัญญาณลอจิก "1" ให้กับขา DRQ ขาใดขาหนึ่ง (ขา DRQ ทั้งสามนี้จะต่อเข้ากับ DRQ1-DRQ3 ของ 8237A-5)

เมื่อ 8237A-5 ได้รับสัญญาณนี้แล้วก็จะตรวจสอบว่ามีการขอเข้าถึงหน่วยความจำโดยตรง ในแชนแนลที่มีลำดับความสำคัญ (Priority) สูงกว่าหรือไม่ ถ้าไม่มีก็จะทำการขอเข้าถึงหน่วยความจำโดยตรง จาก 8088 และตอบรับการขอเข้าถึงหน่วยความจำโดยตรง จากอุปกรณ์ภายนอก (สัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ขอเข้าถึงหน่วยความจำโดยตรง จะแอคทีฟ) แต่ถ้ามี 8237A-5 ก็จะทำการขอเข้าถึงหน่วยความจำโดยตรง ให้กับแชน

แนลที่มีลำดับความสำคัญสูงกว่าก่อน แล้วจึงทำการขอเข้าถึงหน่วยความจำโดยตรง ให้กับแชนแนลที่มีลำดับความสำคัญต่ำกว่า ภายในรอมไบออส (ROM BIOS) ของ IBM/PC จะโปรแกรม 8237A-5 ให้ DRQ1 มีลำดับความสำคัญสูงสุด และDRQ3 มีลำดับความสำคัญต่ำสุด ดังนั้นถ้ามีการขอเข้าถึงหน่วยความจำโดยตรง ของอุปกรณ์ภายนอกผ่านทางแชนแนลที่ 1 (DRQ1) และแชนแนลที่ 2 (DRQ2) 8237A-5 ก็จะทำการขอเข้าถึงหน่วยความจำโดยตรง ให้กับแชนแนลที่ 1 ก่อน จากนั้นเมื่อเสร็จขบวนการเข้าถึงหน่วยความจำโดยตรง ของแชนแนลที่ 1 แล้ว จึงจะทำการขอเข้าถึงหน่วยความจำโดยตรง ให้กับแชนแนลที่ 2

อย่างไรก็ตาม 8237A-5 ยังมีแชนแนลสำหรับการขอเข้าถึงหน่วยความจำโดยตรง อยู่อีก 1 แชนแนล คือแชนแนลที่ 0 (DRQ0) ซึ่งในความเป็นจริงแล้วแชนแนลนี้จะมีลำดับความสำคัญที่สูงกว่าแชนแนลที่ 1 แต่จะไม่ถูกต่อออกมายังขาของสล็อต เนื่องจาก IBM/PC จะใช้แชนแนลที่ 0 นี้ในการรีเฟรชหน่วยความจำที่เป็นหน่วยความจำพลวัตร

ในการขอเข้าถึงหน่วยความจำโดยตรง นั้นสัญญาณ DRQ นี้ จะต้องแอคทีฟอยู่ในช่วงระยะเวลาหนึ่งเท่านั้น ถ้าสัญญาณนี้แอคทีฟอยู่นานเกินไป จะทำให้เกิดขบวนการเข้าถึงหน่วยความจำโดยตรง ขึ้นมากกว่า 1 ขบวนการได้ สำหรับวงจรที่ขอเข้าถึงหน่วยความจำโดยตรง โดยทั่วไปแล้วจะใช้สัญญาณตอบรับการขอเข้าถึงหน่วยความจำโดยตรง หรือสัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ขอเข้าถึงหน่วยความจำโดยตรง นั้น ในการรีเซ็ทสัญญาณ DRQ เช่นอุปกรณ์ภายนอกที่ขอเข้าถึงหน่วยความจำโดยตรง ผ่านทางแชนแนลที่ 1 (DRQ1) ก็จะคอยตรวจสอบการตอบรับการขอ เข้าถึงหน่วยความจำโดยตรง จากสัญญาณ $\overline{\text{DACK}}$ ของแชนแนลที่ 1 ($\overline{\text{DACK}}1$) เมื่อได้รับสัญญาณจาก $\overline{\text{DACK}}1$ แล้วก็จะรีเซ็ทสัญญาณ DRQ1 (เปลี่ยนจากลอจิก "1" เป็น "0")

$\overline{\text{DACK}}$0-$\overline{\text{DACK}}$3 (DMA Acknowledge 0-3; ขา B19, B17, B26 และ B15) :

สัญญาณทั้ง 4 นี้ เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่ง 8237A-5 สร้างขึ้นเพื่อแสดงให้วงจรภายนอกที่ขอเข้าถึงหน่วยความจำโดยตรง ทราบว่าการขอเข้าถึงหน่วยความจำโดยตรง นั้นได้รับการตอบสนองแล้ว และ 8237A-5 จะเข้าสู่ขบวนการเข้าถึงหน่วยความจำโดยตรง เพื่อให้การส่งผ่านข้อมูลระหว่างอุปกรณ์ I/O ที่ขอเข้าถึงหน่วยความจำโดยตรง กับหน่วยความจำเกิดขึ้นได้โดยตรง (คือไม่ต้องผ่าน 8088) โดยสัญญาณ $\overline{\text{DACK}}$ นี้จะแอคทีฟในแชนแนลใดก็ขึ้นอยู่กับว่าขบวนการเข้าถึงหน่วยความจำโดยตรง ที่จะเกิดขึ้นนั้น เป็นการตอบสนองต่อการขอเข้าถึงหน่วยความจำโดยตรง ในแชนแนลใด เช่นถ้าขบวนการเข้าถึงหน่วยความจำโดยตรง ที่จะเกิดขึ้นนั้นเป็นการตอบสนองต่อการขอเข้าถึงหน่วยความจำโดยตรง ในแชนแนลที่ 2 (DRQ2) สัญญาณ $\overline{\text{DACK}}2$ ก็จะแอคทีฟ เป็นต้น

ดังที่ได้กล่าวแล้วว่าสัญญาณ DRQ0 นั้น จะไม่ถูกต่อออกมายังขาของสล็อต ดังนั้นวงจรอินเตอร์เฟซจึงไม่สามารถจะขอเข้าถึงหน่วยความจำโดยตรง ผ่านทางแชนแนล 0 ได้ แต่สัญญาณ $\overline{\text{DACK}}0$ จะถูกต่อออกมายังสล็อตด้วย (ขา B19) ทั้งนี้ก็เพื่อที่จะแสดงให้วงจรอินเตอร์เฟซต่าง ๆ ทราบว่าขบวนการเข้าถึงหน่วยความจำโดยตรง ที่เกิดขึ้นในช่วงเวลาที่ $\overline{\text{DACK}}0$ แอคทีฟนั้น เป็นขบวนการที่ใช้สำหรับการรีเฟรชหน่วยความจำที่เป็นหน่วยความจำพลวัตร ซึ่งวงจรอินเตอร์เฟซที่ใช้หน่วยความจำประเภทนี้สามารถจะนำไปใช้ในการรีเฟรชหน่วยความจำพลวัตรที่อยู่ในวงจรได้

โดยที่การรีเฟรชหน่วยความจำนั้นจะเกิดขึ้นในทุก ๆ 15.12 μs หรือทุก ๆ 72 คล็อก ดังนั้นสัญญาณ $\overline{\text{DACK}}0$ นี้ก็จะแอคทีฟในทุก 15.12 μs ด้วย

AEN (Address Enable; ขา A11) :

สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้น เป็นบัสไซเคิลของขบวนการเข้าถึงหน่วยความจำโดยตรง

สำหรับบนเมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการดิสเอเบิล (Disable) ตัวควบคุมบัส 8288 และจะใช้ดิสเอเบิลพอร์ทอินพุท/เอาท์พุทต่าง ๆ ที่ไม่เกี่ยวข้องกับขบวนการเข้าถึงหน่วยความจำโดยตรง ที่เกิดขึ้นนี้ ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการเข้าถึงหน่วยความจำโดยตรง นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรส และจะทำให้สัญญาณ $\overline{\text{IOR}}$ หรือ $\overline{\text{IOW}}$ แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ทอินพุท/เอาท์พุท ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ทอินพุท/เอาท์พุท ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดได้

T/C (Terminal Count; ขา B27) :

สัญญาณนี้ถูกสร้างขึ้นจากการนำเอาสัญญาณเอาท์พุทที่ขา EOP ของ 8237A-5 มากลับลอจิก (โดยใช้เกทอินเวอร์ทเตอร์) ทำให้สัญญาณ T/C นี้แอคทีฟที่ลอจิก "1"

สำหรับสัญญาณนี้จะแอคทีฟเมื่อจำนวนไบท์ในการส่งผ่านข้อมูลของขบวนการเข้าถึงหน่วยความจำโดยตรง ในแชนแนลใดแชนแนลหนึ่ง ครบตามจำนวนที่กำหนดไว้ โดยทั่วไปแล้วสัญญาณที่จะถูกใช้ในการสิ้นสุดขบวนการเข้าถึงหน่วยความจำโดยตรง ที่ทำการส่งผ่านข้อมูลเป็นบล็อก เนื่องจากสัญญาณนี้จะแอคทีฟโดยไม่แสดงว่าเป็นสัญญาณของแชนแนลใด ดังนั้นจึงต้องทำการนำสัญญาณ T/C นี้ผ่านเกทอินเวอร์ทเตอร์แล้วนำไปออร์กับสัญญาณ $\overline{\text{DACK}}$ เพื่อให้สามารถทราบได้ว่า สัญญาณ T/C ที่เกิดขึ้นนั้นเป็นสัญญาณของแชนแนลใด สำหรับในแชนแนลที่ 0 นั้นสัญญาณ T/C จะแอคทีฟในช่วงเวลาที่คงที่คือ ทุก ๆ 990.804 ms ซึ่งก็คือช่วงเวลาที่ใช้ในการรีเฟรชหน่วยความจำขนาด 64 kbyte นั่นเอง

บัสของแหล่งจ่ายไฟของระบบ

+5 V_{dc} (ขา B3 และ B29) :

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟ DC +5 V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±5% คือ อยู่ในช่วง +4.75 ถึง +5.25 V_{dc}

+12 V_{dc} (ขา B9) :

ขานี้จะต่อกับแหล่งจ่ายไฟ DC +12 V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±5% คืออยู่ในช่วง +11.4 ถึง +12.6 V_{dc}

-5 V_{dc} (ขา B3 และ B29) :

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟ DC -5 V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±10% คือ อยู่ในช่วง -5.5 ถึง -4.5 V_{dc}

-12 V_{dc} (ขา B9) :

ขานี้จะต่อกับแหล่งจ่ายไฟ DC -12 V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±10% คืออยู่ในช่วง -13.2 ถึง -10.8 V_{dc}

2.1.2 การอ้างแอดเดรสของพอร์ทอินพุท/เอาท์พุท

ในการควบคุม และตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซัพพอร์ท หรือการ์ดต่าง ๆ ที่ใช้ในระบบของ IBM/PC นั้น จะกระทำโดยผ่านทางพอร์ทอินพุท/เอาท์พุท ของระบบ ดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ทอินพุท/เอาท์พุท ต่าง ๆ ของระบบด้วย และเนื่องจากการควบคุมหรือติดต่อกับพอร์ทเหล่านี้ ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ทอินพุท/เอาท์พุท เหล่านั้นโดยตรง เราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ทอินพุท/เอาท์พุท ต่าง ๆ นั้น จะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ทอินพุท/เอาท์พุท โดยเฉพาะ คือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้กับพอร์ทเหล่านี้จะทำได้โดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยังแอดเดรสของพอร์ทที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ท ก็จะทำได้โดยการใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการเช่นกัน

ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ทอินพุท/เอาท์พุท อยู่ทั้งสิ้น 65,356 หรือ 64 k แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1 Mbytes) ซึ่งทำให้การอ้างแอดเดรสของพอร์ทอินพุท/เอาท์พุท ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้น ดังนั้นในการอ้างถึงแอดเดรสของพอร์ทอุปกรณ์หรือชิพซัพพอร์ทใด ๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วย โดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำไปใช้งาน แต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วย เพียงแต่ไม่ได้ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่างเช่นในการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0010H นั้นจะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0410H, 0810H, 0C10H ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งาน จึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A10-A15 นั้น ไม่ทำให้เกิดความแตกต่างใด ๆ ขึ้น

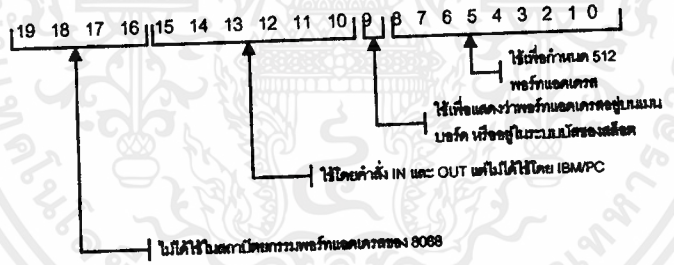
เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น (คือ A0-A9) ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1,024 พอร์ท (จากจำนวน 64 k พอร์ท) เท่านั้น นอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1,024 พอร์ทออกเป็น 2 ส่วน (ส่วนละ 512 พอร์ท) อีกด้วย กล่าวคือถ้าข้อมูลในบิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทต่าง ๆ ที่อยู่บนเมนบอร์ด (Main Board) ของ IBM/PC เช่น 8253-5,

8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทที่อยู่บนการ์ดต่าง ๆ เท่านั้น

จากที่ได้กล่าวมานั้นจะสรุปได้ว่าพอร์ทบน IBM/PC ทั้ง 1,024 พอร์ท ถูกแบ่งออกเป็น 2 กลุ่ม โดยที่กลุ่มแรกเป็นกลุ่มของพอร์ทที่อยู่บนเมนบอร์ด และกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ทที่อยู่บนการ์ดต่าง ๆ

สำหรับในกรณีของการส่งข้อมูลให้กับพอร์ททั้ง 1,024 พอร์ท เราสามารถที่จะเลือกส่งไปยังพอร์ทใด ๆ ใน IBM/PC ได้ ดังนั้นการเลือกแอดเดรสสำหรับพอร์ทที่อยู่บนการ์ดจึงสามารถทำได้โดยสะดวก แต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือถ้าแอดเดรสที่เราเลือกให้กับพอร์ทนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่ในตำแหน่งแอดเดรสนี้ ก็จะเท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ทที่อยู่บนเมนบอร์ด และพอร์ทที่อยู่บนการ์ดด้วย ซึ่งในกรณีเช่นนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นแอดเดรสบิท A9 มีค่าเป็น "1" คือ แอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น (แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น "1" ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น "1" หรือ "0" ก็ได้)

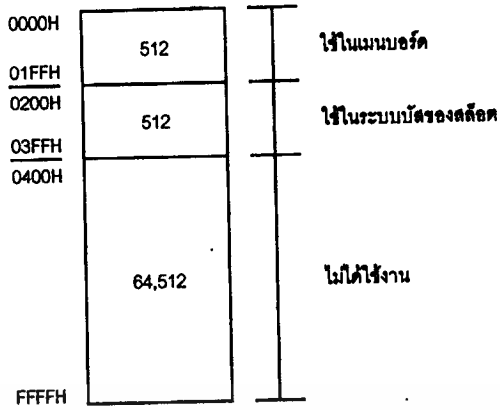
สำหรับรูปที่ 2.1 นี้ จะแสดงถึงการใช้งานแอดเดรสบิทต่าง ๆ ในการอ้างแอดเดรสของพอร์ทใน IBM/PC



รูปที่ 2.1 การใช้แอดเดรสบิทต่าง ๆ ในการอ้างแอดเดรสของพอร์ทใน IBM/PC

2.1.3 การใช้งานแอดเดรสสำหรับพอร์ทอินพุท/เอาท์พุท ใน IBM/PC

จากที่ได้กล่าวไว้ข้างต้นนั้น พอร์ทอินพุท/เอาท์พุท ทั้ง 1,024 พอร์ทใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่ม ๆ ละ 512 พอร์ท สำหรับในหัวข้อนี้จะกล่าวถึงการใช้งานพอร์ทต่าง ๆ เหล่านี้ โดยจะแบ่งออกเป็น 2 กลุ่มตามที่ได้อธิบายไว้ในหัวข้อที่ผ่านมาดังนี้



รูปที่ 2.2 การใช้งานแอดเดรสของพอร์ทบน IBM/PC

1. ในกลุ่มแรกนี้เป็นกลุ่มของ I/O ที่อยู่บนเมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (ขอให้ระลึกอยู่เสมอว่า A10-A15 นั้นจะไม่ถูกใช้งาน) หรือแอดเดรสที่มีบิท A9 เป็น "0" นั่นเอง

สำหรับแอดเดรสของพอร์ทอินพุท/เอาท์พุท ในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสของชิพซัพพอร์ท และอุปกรณ์ที่เป็น I/O ต่าง ๆ บนเมนบอร์ดของ IBM/PC เช่นแอดเดรส 0000H จนถึง 01FFH ในการอ้างแอดเดรสของชิพซัพพอร์ท และอุปกรณ์ต่าง ๆ ที่ทำหน้าที่เป็น I/O บนเมนบอร์ดของ IBM/PC

แอดเดรสที่มีค่าตั้งแต่ 00C0H จนถึงแอดเดรส 01FFH นั้นไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM/PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งานแอดเดรสต่าง ๆ เหล่านี้ได้ แต่อย่างไรก็ตามแอดเดรสเหล่านี้ยังคงถูกดีโค้ดให้เป็นแอดเดรสที่ใช้ในการอ่านข้อมูลจากพอร์ทอินพุท/เอาท์พุท บนเมนบอร์ดเท่านั้น ดังนั้นการใช้ค่าแอดเดรส 0C00H-01FFH กับพอร์ทอินพุท/เอาท์พุท บนการ์ด หรือวงจรอินเตอร์เฟซที่เราสร้างขึ้นนั้น ต้องเป็นพอร์ทเอาท์พุทเพียงชนิดเดียวเท่านั้น กล่าวคือ จะทำการอ่านข้อมูลจากพอร์ทอินพุท/เอาท์พุท (ที่ไม่ได้อยู่บนเมนบอร์ด) ที่มีค่าแอดเดรสอยู่ในช่วง 00C0H-01FFH ไม่ได้

2. ในกลุ่มที่สองนี้ จะเป็นกลุ่มของพอร์ทอินพุท/เอาท์พุท ที่ถูกใช้งานอยู่บนการ์ดที่ใช้เสียบบนสล็อตต่าง ๆ ของ IBM/PC สำหรับแอดเดรสของพอร์ทเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเอง สำหรับการใช้งานแอดเดรสของพอร์ทอินพุท/เอาท์พุท ทั้งสองกลุ่มจะแสดงได้ดังตารางที่ 1

ตารางที่ 2.1 แสดงการจัดตำแหน่งพอร์ทของระบบ

หมายเลขพอร์ท	การใช้งาน
000-01F	ตัวควบคุมดีเอ็มเอ 1, 8237A-5
020-03F	ตัวควบคุมอินเตอร์รัปต์ 1, 8259 (มาสเตอร์)
040-05F	ตัวควบคุมไทเมอร์เคาน์เตอร์8254-2

060-06F	ตัวควบคุมพอร์ทขนาน และคีย์บอร์ด 8042
070-07F	Real Time Clock, NMI ของระบบ
080-09F	ดีเอ็มเอเพจรีจิสเตอร์ 74LS612
0A0-0BF	ตัวควบคุมอินเตอร์รัปต์ 2, 8259 (สเลฟ)
0C0-0DF	ตัวควบคุมดีเอ็มเอ 2, 8237A-5
0F0	เคลียร์แมธโคโปรเซสเซอร์
0F1	รีเซตแมธโคโปรเซสเซอร์
0F8-0FF	แมธโคโปรเซสเซอร์ 80287
1F0-1FB	ฮาร์ดดิสก์
200-207	เกมอินพุท/เอาท์พุท
278-27F	เครื่องพิมพ์ขนาน พอร์ท 2
2F8-2FF	เครื่องพิมพ์อนุกรม พอร์ท 2
300-31F	การ์ดโปรโตไทป์ (Prototype)
360-36F	สงวนไว้
378-37F	เครื่องพิมพ์ขนาน พอร์ท 1
380-38F	SDLC ไบต์ซิงโครไนซ์ 1
3A0-3AF	ไบต์ซิงโครไนซ์
3B0-3BF	อะแดปเตอร์โมโนโครม และเครื่องพิมพ์
3C0-3CF	สงวนไว้
3D0-3DF	อะแดปเตอร์สี/กราฟฟิกส์
3F0-3F7	ตัวควบคุมดิสก์ไดรฟ์
3F8-3FF	พอร์ทอนุกรม 1

อย่างไรก็ตามการใช้งานแอดเดรสในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ ทั้งนี้ขึ้นอยู่กับการใช้งานการ์ดต่าง ๆ ร่วมกับ IBM/PC โดยการ์ดที่ถูกออกแบบผลิตขึ้นใหม่นั้นอาจจะใช้ค่าแอดเดรสต่าง ๆ ที่เหลืออยู่นี้ได้ ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเตอร์เฟซที่จำเป็นต้องใช้ค่าแอดเดรสสำหรับพอร์ทอินพุท/เอาท์พุท จึงควรจะตรวจสอบดูก่อนว่าการ์ดต่าง ๆ ที่ใช้อยู่ในระบบของ IBM/PC ที่เราใช้งานอยู่นั้นมีการ์ดใดบ้าง และการ์ดเหล่านั้นใช้งานแอดเดรสใดบ้าง จากนั้นจึงทำการออกแบบวงจรอินเตอร์เฟซโดยเลือกใช้เฉพาะแอดเดรสที่ยังไม่ถูกใช้งาน

2.2 วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog to Digital Converters : ADC)

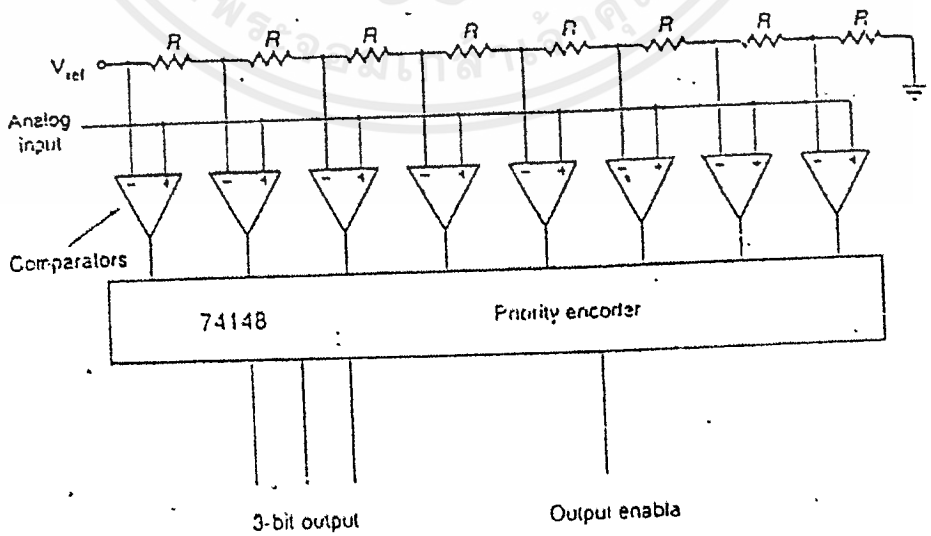
การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลนั้น เป็นสิ่งจำเป็นมาก เพราะจะทำให้เครื่องมือทางดิจิตอล หรือเครื่องคอมพิวเตอร์รับรู้ และทำการตอบสนองกับสัญญาณอนาล็อกที่เข้ามาได้ ดังนั้นจึงต้องทำการเปลี่ยนสัญญาณอนาลือกเป็นสัญญาณดิจิตอลเสียก่อน วิธีที่ใช้ในการเปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลที่สำคัญมีดังนี้คือ

2.2.1. แบบแฟลชคอนเวอร์เตอร์ (Flash Converters)

การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลด้วยวิธีแฟลชคอนเวอร์เตอร์ ได้แสดงไว้ในรูปที่ 2.3 เป็นวงจร ADC ขนาด 3 บิท มีส่วนประกอบที่สำคัญดังต่อไปนี้

- รีซิสเตอร์เน็ตเวอร์ก (Resistors Network)
- วงจรเปรียบเทียบแรงดัน
- วงจรเข้ารหัส

หลักการทำงาน ตัวต้านทาน 8 บิทต่อกันเป็นวงจรแบ่งแรงดัน โดยแบ่งค่าแรงดันของแรงดันอ้างอิงออกเป็น 8 ส่วน ถ้า V_{ref} มีค่าเท่ากับ +10 V แรงดันถูกแบ่งออกเป็นค่าต่าง ๆ ดังต่อไปนี้ 10.00, 8.75, 7.50, 6.25, 5.00, 3.75, 2.50 และ 1.25 ค่าของแรงดันที่แบ่งแล้วถูกต่อเข่ากับขา (-) ของออปแอมป์ ส่วนขา (+) ของออปแอมป์ต่อถึงกันทุกตัว และยังใช้ขานี้เป็นขาอินพุทโวลท์เตจของวงจรอีกด้วย ออปแอมป์ที่เห็นในวงจรทุกตัวทำหน้าที่เป็นวงจรเปรียบเทียบแรงดันทั้งหมด โดยจะเปรียบเทียบระหว่างแรงดันอินพุทกับแรงดันที่ได้จากการแบ่งของรีซิสเตอร์เน็ทเวอร์ก สมมติว่าขณะนี้มีแรงดัน V_{in} มีค่าเท่ากับ 0 V เห็นได้ว่าขณะนี้แรงดันที่ขา (+) ของออปแอมป์ทุกตัวจึงมีค่าเป็น "0" เอาท์พุทของออปแอมป์มีค่าเป็น "1" ได้ก็ต่อเมื่อแรงดันที่ขา (+) มีมากกว่าแรงดันที่ขา (-) ตารางที่ 2.2 แสดงว่าเอาท์พุทของออปแอมป์เมื่อ V_{in} มีค่าต่าง ๆ



รูปที่ 2.3 วงจร ADC แบบแฟลชคอนเวอร์เตอร์

จากนั้นเอาท์พุทที่ได้จากออปแอมป์ก็จะถูกส่งไปให้กับวงจรเข้ารหัสให้เป็นเลขฐาน 16 ต่อไป ADC แบบแฟลช หรือ ขนาน เป็น ADC ที่มีความเร็วสูงที่สุดเมื่อเทียบกับ ADC ชนิดอื่น และยังมีค่าความเป็นเชิงเส้นดีกว่า ADC ชนิดอื่นอีกมาก ยกตัวอย่างถ้าเอาท์พุทมี 8 บิท จะต้องใช้ออปแอมป์ถึง 225 ตัว หรือ 2^n-1 นั่นเอง เมื่อ n คือจำนวนบิท

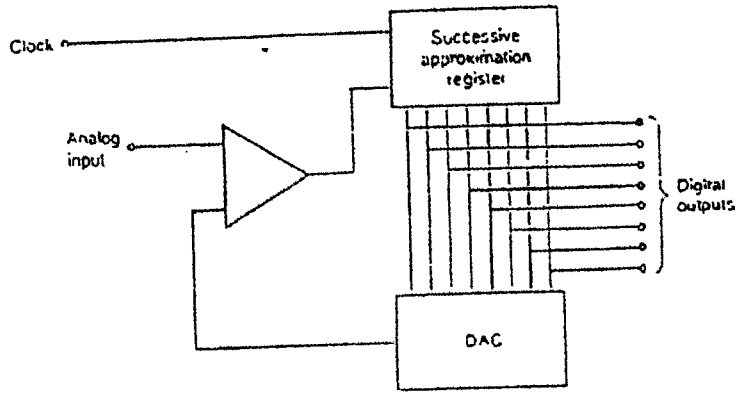
ตารางที่ 2.2 เอาท์พุทของออปแอมป์ที่ V_{in} ค่าต่าง ๆ

	V_{o1}	V_{o2}	V_{o3}	V_{o4}	V_{o5}	V_{o6}	V_{o7}	V_{o8}
$V_{in} < V_1$	0	0	0	0	0	0	0	0
$V_2 < Vin < V_1$	1	0	0	0	0	0	0	0
$V_3 < V_{in} < V_2$	1	1	0	0	0	0	0	0
$V_4 < V_{in} < V_3$	1	1	1	0	0	0	0	0
$V_5 < V_{in} < V_4$	1	1	1	1	0	0	0	0
$V_6 < V_{in} < V_5$	1	1	1	1	1	0	0	0
$V_7 < V_{in} < V_6$	1	1	1	1	1	1	0	0
$V_8 < V_{in} < V_7$	1	1	1	1	1	1	1	0
$V_o < V_8$	1	1	1	1	1	1	1	1

2.2.2 แบบซักเซสซีฟแอพพร็อกซิเมชั่นคอนเวอร์เตอร์ (Successive Approximation Converters)

วงจรซักเซสซีฟแอพพร็อกซิเมชั่นแสดงดังรูปที่ 2.4 หัวใจของวงจรก็คือ DAC และยังมีส่วนควบคุมลอจิก ซึ่งเรียกว่าซักเซสซีฟแอพพร็อกซิเมชั่นรีจิสเตอร์ (Successive Approximation Register : SAR) จะทำให้บิทที่มีนัยสำคัญมากที่สุดเป็นลอจิก "1" เมื่อได้รับพัลส์ถูกแรก ส่วนบิทที่เหลือทั้งหมดจะเป็นลอจิก "0" เลขไบนารีจะถูกส่งจาก SAR ไปยัง DAC ทำหน้าที่เปลี่ยนไบนารีเป็นโวลท์เตจ สัญญาณเอาท์พุทของ DAC จะถูกนำไปเปรียบเทียบกับอนาล็อกอินพุท ถ้าแรงดันของ DAC สูงกว่าแรงดันของอนาล็อกอินพุท คอมแพเรเตอร์เอาท์พุทจะเป็นลอจิก "0" ไปรีเซ็ทบิทให้เป็น 0 ถ้าแรงดันเอาท์พุทของ DAC น้อยกว่าแรงดันอินพุท บิทที่มีนัยสำคัญสูงสุดจะคงสภาวะ "1"

เมื่อมีคล็อกลูกใหม่เข้ามา บิทที่มีนัยสำคัญมากที่สุดถัดมาจะถูกเซ็ทเป็น 1 การเปรียบเทียบเกิดขึ้นเหมือนเดิม จนกระทั่งบิทสุดท้าย ที่จุดนี้ SAR จะให้เอาท์พุทที่เรียกว่า เอนด์ออฟคอนเวอร์เตอร์ (end-of-converters : eoc) จำนวนไบนารีถูกส่งออกที่เอาท์พุทรีจิสเตอร์ของ SAR

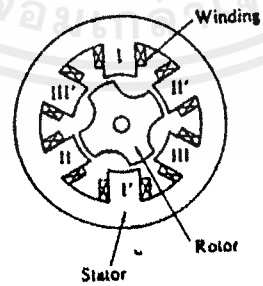


รูปที่ 2.4 ADC แบบ Successive Approximation

2.3 สเต็ปปิ้งมอเตอร์ (Stepping Motor)

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ชนิดที่หมุนทีละสเต็ป โดยแต่ละสเต็ป มอเตอร์จะหมุนด้วยมุมคงที่ค่าหนึ่งซึ่งในการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์นั้นจะอาศัยวงจรควบคุมทางดิจิตอล โดยที่วงจรทางดิจิตอลนี้จะทำหน้าที่ในการจัดลำดับการกระตุ้นในแต่ละเฟสของสเต็ปปิ้งมอเตอร์ ซึ่งจะทำให้สามารถกำหนดทิศทางการหมุน, ความเร็วในการหมุนและตำแหน่งที่ต้องการจะเลื่อนไปของสเต็ปปิ้งมอเตอร์ได้อย่างถูกต้องแม่นยำ

เนื่องจากวงจรทางดิจิตอลที่ใช้ในการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์สามารถกำหนดความเร็วในการหมุนและตำแหน่งที่ต้องการจะเลื่อนไปของสเต็ปปิ้งมอเตอร์ได้อย่างถูกต้องแม่นยำ ดังนั้นจึงไม่จำเป็นต้องมีการป้อนกลับ (Feedback Control) เพื่อควบคุมความเร็วและตำแหน่งในการหมุน

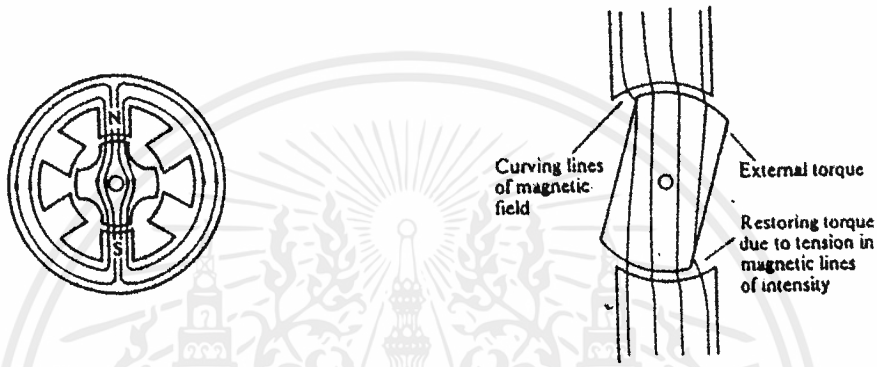


รูปที่ 2.5 โครงสร้างของสเต็ปปิ้งมอเตอร์

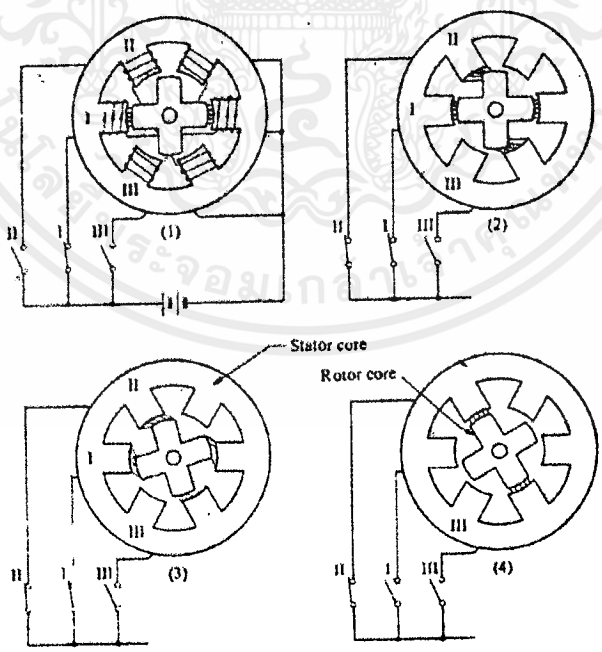
2.3.1 โครงสร้างและหลักการทำงานของสเต็ปปิ้งมอเตอร์

ภายในสเต็ปปิ้งมอเตอร์ประกอบด้วย สเตเตอร์ (stator), โรเตอร์ (rotor) และขดลวด (coil) ประกอบเข้าด้วยกันดังรูปที่ 2.5

เนื่องจากโรเตอร์เป็นเหล็กอ่อน ซึ่งมีคุณสมบัติที่พยายามปรับตัวเองให้อยู่ในแนวที่เส้นแรงแม่เหล็กผ่านมากที่สุด ดังในรูปที่ 2.6 เมื่อเกิดเส้นแรงแม่เหล็กขึ้นที่ตัวสเตเตอร์ผ่านโรเตอร์ ตัวโรเตอร์ก็จะพยายามปรับตัวเองให้เส้นแรงแม่เหล็กผ่านมากที่สุดโดยการหมุนตัวเองทำให้เคลื่อนที่ไปเกิดมุมของการหมุน และจะหยุดหมุน เมื่อเส้นแรงแม่เหล็กที่ตัดผ่านตัวมันถึงจุดมากที่สุด



รูปที่ 2.6 เส้นแรงแม่เหล็กที่ทำให้เกิดแรงบิด



รูปที่ 2.7 แสดงการทำงานของสเต็ปปิ้งมอเตอร์

การทำให้สเต็ปปิ้งมอเตอร์หมุนนั้นสามารถทำได้โดยอาศัยหลักการนี้แต่ต้องให้เส้นแรงแม่เหล็กเกิดขึ้นโดยรับช่วงต่อกันไปเรื่อยๆ ดังรูปที่ 2.7 เป็นสเต็ปปิ้งมอเตอร์ที่แกนเหล็กสเตเตอร์ (Stator Core) มีซี่ฟัน (Teeth) 6 ซี่ ขณะที่โรเตอร์ (Rotor) มีฟัน 4 ซี่ ทั้งโรเตอร์และสเตเตอร์เป็นเหล็กอ่อน ขดลวด (Core) 3 ชุดถูกต่ออยู่ดังรูป แต่ะชุดขดลวดมี 2 ขดลวดต่ออนุกรมกัน เรียกแต่ละชุดว่า เฟส (Phase) และผลจากการต่อแบบนี้เรียกว่า มอเตอร์ 3 เฟส (3 Phase Motor) กระแสถูกจ่ายไปยังขดลวดแต่ละขดลวดผ่านสวิตช์ I , II และ III ในสภาวะ (1) ชุดขดลวดของเฟส 1 ได้รับกระแสโดยผ่านสวิตช์ I หรือเฟส I ถูกกระตุ้น เส้นแรงแม่เหล็กที่เกิดขึ้นในช่องอากาศ (Air Gap) เกิดขึ้นเนื่องจากการกระตุ้น ซึ่งแสดงด้วยลูกศร ในสภาวะนี้สเตเตอร์ 2 ขั้วของเฟส 1 จะอยู่ในแนวเดียวกับ 2 ซี่ที่อยู่ตรงข้ามกันของโรเตอร์ นี่เป็นสภาวะสมดุลย์ ซึ่งอยู่ในเทอมของไดนามิค (Dynamics) เมื่อสวิตช์ II ปิดเพื่อกระตุ้นเฟส 2 กับเฟส 1 เส้นแรงแม่เหล็กจะถูกสร้างขึ้นที่ขั้วของสเตเตอร์ของเฟส 2 ในลักษณะซึ่งแสดงในสภาวะ (2) ทอร์ค (Torgue) ทิศทางทวนเข็มนาฬิกาจะถูกสร้างขึ้นมาจากความเครียด (Tension) ในฟลักซ์แม่เหล็กเอียงไปยังแกนมอเตอร์ที่อยู่ใกล้ หลังจากนั้นโรเตอร์จะมาอยู่ในสภาวะ (3)

ดังนั้นโรเตอร์จะหมุนไปด้วยมุมการเปลี่ยนแปลงคงที่ ซึ่งเรียกว่ามุมสเต็ป (Step angle) ในที่นี้คือ 15 องศา ขณะที่สวิตช์จะมีการเปลี่ยนแปลงอีกครั้งหนึ่งคือ สวิตช์ I จะถูกเปิด เพื่อลดพลังงานในเฟส โดยโรเตอร์จะหมุนไป 15 องศามาอยู่ในสภาวะ (4) ตำแหน่งมุมของโรเตอร์จะถูกควบคุมโดยการเปิด-ปิดสวิตช์ ถ้าสวิตช์ถูกปิด-เปิดเป็นลำดับ โรเตอร์จะหมุนไปในลักษณะที่เป็นสเต็ป ความเร็วเฉลี่ยจะสามารถควบคุมได้ด้วยการเปิด-ปิดสวิตช์ดังรูปที่ 2.7

จากที่กล่าวพอที่จะกล่าวถึงคุณสมบัติสเต็ปปิ้งมอเตอร์ได้ว่า

1) การหมุนของมอเตอร์จะเป็นสเต็ป (เป็นขั้นๆ) สเต็ปละกี่องศาขึ้นอยู่กับชนิดของมอเตอร์

2) ความเร็วในการหมุนขึ้นกับสัญญาณพัลส์ที่ให้เข้ามาทางอินพุทของมอเตอร์ (ความถี่)

3) ความผิดพลาดในการหมุน 1 สเต็ปมีค่าน้อยมาก แต่ต้องจำกัดอยู่ในความเร็วที่พอดีด้วย

4) คุณสมบัติการตอบสนองสัญญาณที่มอเตอร์เริ่มทำงานและหยุดทำงานดีมาก

5) เนื่องจากไม่มีแปรงถ่าน (Commutator) เหมือนมอเตอร์ในระบบดีซี ดังนั้นจึงมีความแน่นอนในการทำงานสูง

6) แหล่งจ่ายแรงดันไฟที่ใช้ในการขับมอเตอร์มีค่าไม่มาก

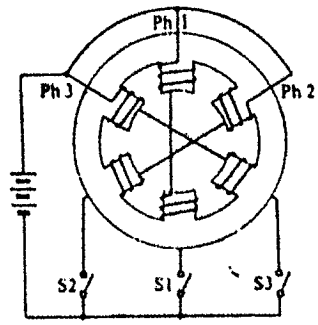
7) ไม่ทำให้เกิดเสียงรบกวนและการสั่นได้ง่าย

8) ทำงานแบบ โอเพ่นลูป (Open Loop)

2.3.2 ชนิดของสเต็ปปิ้งมอเตอร์

2.3.2.1 วาริเบิลรีลัคแทนซ์สเต็ปปิ้งมอเตอร์ (Variable Reluctance Stepping Motor)

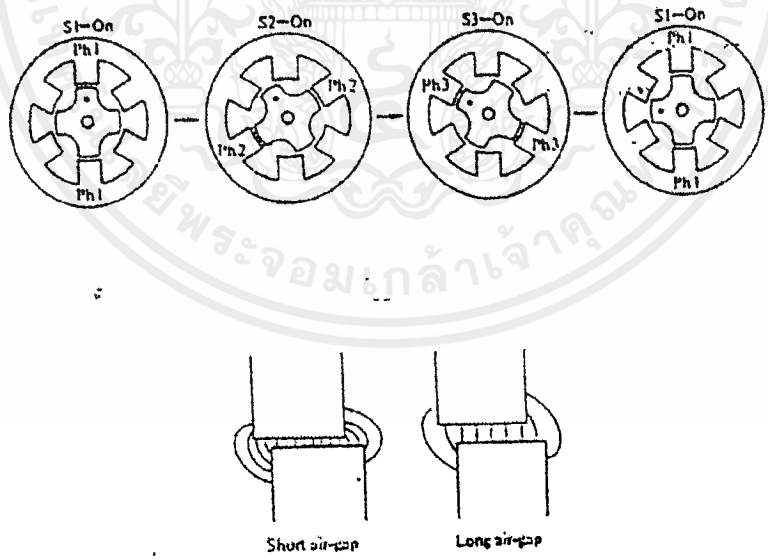
มอเตอร์แบบนี้ทำด้วยเหล็กอ่อน ซึ่งค่าซึมซาบแม่เหล็ก (Permeability) สูง และสามารถให้ฟลักซ์แม่เหล็กผ่านได้มาก โดยโรเตอร์จะติดอยู่กับแกนมอเตอร์ และสเตเตอร์ติดอยู่กับโครงของตัวมอเตอร์ จากรูป 2.8 เป็นภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบนี้ ซึ่งเป็นมอเตอร์ 3 เฟส มีซี่ฟัน 6 ซี่ ฟันของสเตเตอร์จะอยู่ตรงข้ามจะต่อกันเป็นอนุกรม หรือขนานก็ได้ (ในที่นี้ต่อแบบอนุกรม)



รูปที่ 2.8 โครงสร้างวาริเอเบิลรีลัคแทนซ์มอเตอร์

เราจะเห็นได้ว่าฟันของสเตเตอร์ 2 ซี่ ที่มีเฟสเดียวกัน จะมีขั้วแม่เหล็กตรงข้ามกันและกันดังแสดงดังรูป สมมุติว่าฟัน 1,2 และ 3 มีขั้วเป็นขั้วเหนือ ฟัน 1, 2 และ 3 จะเป็นขั้วได้เมื่อถูกกระตุ้น

กระแสแต่ละเฟสจะถูกกระตุ้น ฟลักซ์แม่เหล็กก็จะเกิดดังรูป 2.9 ฟันของโรเตอร์ก็จะมีตำแหน่งในแนวเดียวกันกับฟันของสเตเตอร์ ซึ่งจะมีผลให้แมกเนติกรีลัคแทนซ์ (Megnetic Reluctance) น้อยที่สุดสภาวะนี้คือตำแหน่งสมดุลย์

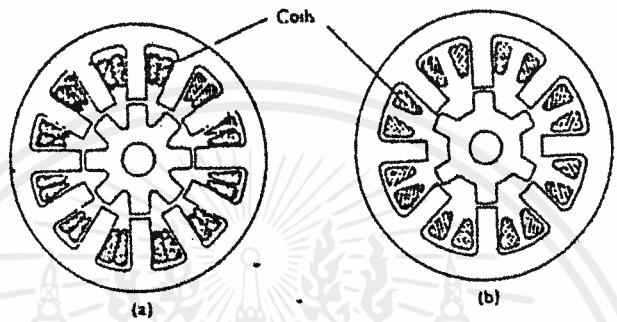


รูปที่ 2.9 ฟลักซ์แม่เหล็กที่เกิดขึ้น

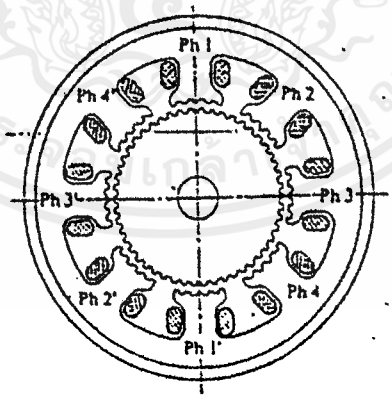
โครงสร้างเบื้องต้นของมอเตอร์แบบนี้ จะมีลักษณะ

1) ช่องว่างอากาศควรจะเล็กที่สุด เท่าที่จะเป็นไปได้ช่องว่างอากาศระหว่างฟันของโรเตอร์กับฟันของสเตเตอร์ควรมีค่าความห่างกันน้อยมาก เพื่อที่จะได้ทอร์คสูง และตำแหน่งที่แน่นอนขึ้น

2) สำหรับมุมสเต็ปเล็ก ๆ จากรูป 2.10 ก) แสดง 3 เฟสมอเตอร์ที่สเตเตอร์มีฟัน 12 ซี่และโรเตอร์มีฟัน 8 ซี่ รูป 2.10 ข) เป็นรูป 4 เฟส มอเตอร์ที่สเตเตอร์มีฟัน 8 ซี่และโรเตอร์มีฟัน 6 ซี่ ซี่ทั้งสองรูปนี้มีมุมสเตปเท่ากับ 15 องศา



รูปที่ 2.10 โรเตอร์ และสเตเตอร์



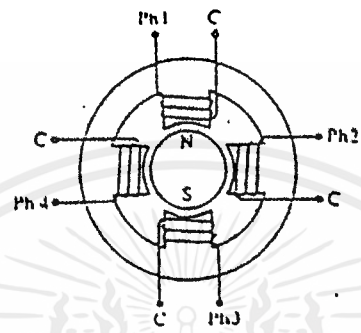
รูปที่ 2.11 หน้าตัดของสเต็ปปิ้งมอเตอร์ 4 เฟสที่มีฟันโรเตอร์ 50 ซี่ มุมสเต็ป 1.8 องศา

2.3.2.2 สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร (Permanent Magnatic Stepping Motor)

สเต็ปปิ้งมอเตอร์แบบนี้ใช้แม่เหล็กแบบถาวร รูปที่ 2.11 เป็นตัวอย่างของสเต็ปปิ้งมอเตอร์แบบถาวรแบบ 4 เฟส โรเตอร์เป็นทรงกระบอก สเตเตอร์มีฟัน 4 ซี่ โดยที่แต่ละซี่มีขดลวดพันรอบ ถ้าจำนวนซี่บน

สเตเตอร์และขั้วแม่เหล็กบนโรเตอร์เพิ่มขึ้นเป็น 2 เท่า มุมแต่ละสเต็ปลดลงจากเดิมครึ่งหนึ่ง ดังนั้นเพื่อที่จะลดมุมสเต็ปลงไปอีก ในสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรจะต้องเพิ่มจำนวนแม่เหล็กและซี่ฟัน อย่างไรก็ตามขั้วแม่เหล็กที่จะเพิ่มขึ้นได้นั้นมีจำนวนจำกัด

ลักษณะทั่วไปของมอเตอร์แบบนี้ก็คือ โรเตอร์จะถูกยึดอยู่กับที่ แม้ว่าไม่มีการกระตุ้นเฟส ลักษณะเช่นนี้เรียกว่า ดีเทนซ์ แมคคานิคซึ่ม (Detent Mechanism)

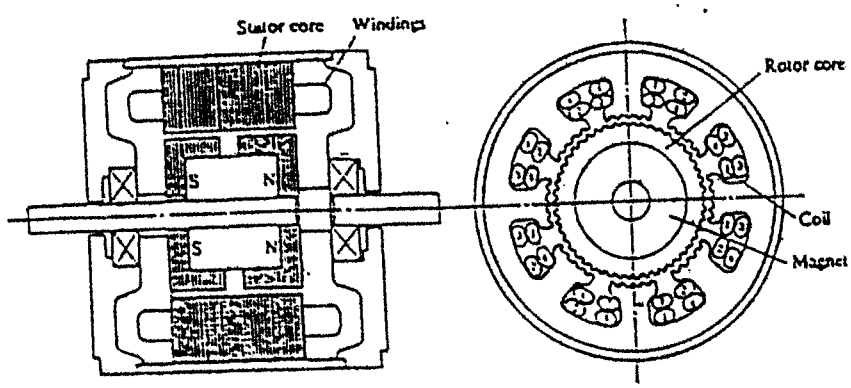


รูปที่ 2.12 โครงสร้างมอเตอร์แบบแม่เหล็กถาวร

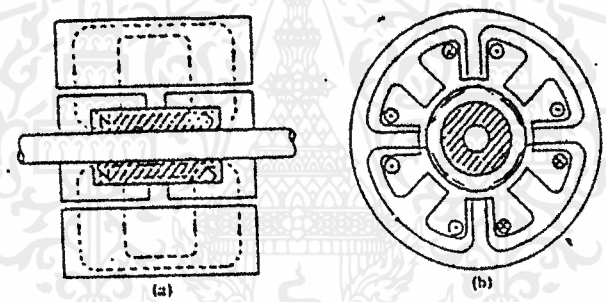
2.3.2.3 สเต็ปปิ้งมอเตอร์แบบไฮบริด (Hybrid Stepping Motor)

เป็นสเต็ปปิ้งมอเตอร์แบบหนึ่งที่มีโรเตอร์เป็นแบบแม่เหล็กถาวร การใช้ชื่อไฮบริดได้มาจากการรวมหลักสำคัญของมอเตอร์แบบแม่เหล็ถาวรและแบบวาริเอเบิลรีลัคแทนซ์ โครงสร้างแกนของสเตเตอร์จะคล้ายกับแบบวาริเอเบิลรีลัคแทนซ์ แต่การพันและการต่อขดลวดจะต่างจากแบบวาริเอเบิลรีลัคแทนซ์มอเตอร์ ซึ่งมีเพียง 1 ขดจาก 2 ขดที่ถูกพันเป็นขั้วเดียวในขณะที่ 4 เฟส ไฮบริดมอเตอร์ ชุดขดลวด 2 เฟสที่แตกต่างกันจะถูกพันบนขั้วเดียวกัน ดังรูป 2.13 เพราะฉะนั้น ขั้วหนึ่งจะไม่มีเพียงเฟสเดียว ขดลวด 2 ขด จะถูกพันเห็นขั้วเดียวกันแบบ ไบฟิลา (Bifilar Winding) ซึ่งจะทำให้แม่เหล็กต่างกันขณะมีการกระตุ้น

ลักษณะที่สำคัญอีกอย่างหนึ่งของไฮบริดมอเตอร์คือ โคเตอร์นั้นจะเป็นแม่เหล็กรูปร่างทรงกระบอกอยู่ในแกนเหล็กของโรเตอร์ มันถูกทำให้เป็นแม่เหล็กตามยาวเพื่อสร้างสนามขั้วเดียวดังรูปที่ 2.14 แต่ละขั้วของแม่เหล็กจะถูกล้อมรอบด้วยฟันเหล็กอ่อน ซึ่งฟันของโรเตอร์กับสเตเตอร์อยู่เหลื่อมกันอยู่ครึ่งช่วงฟัน



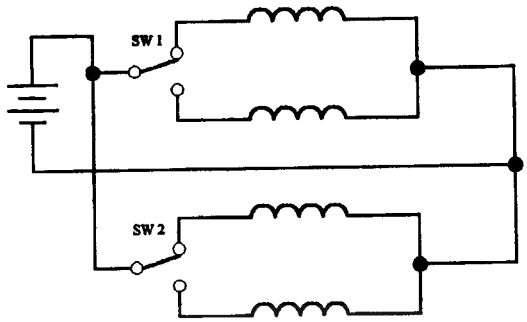
รูปที่ 2.13 โครงสร้างของสเต็ปปิ้งมอเตอร์แบบไฮบริด



รูปที่ 2.14 การวางแม่เหล็กตามยาวเพื่อสร้างสนามขั้วเดียวกัน

2.3.3 การทำงานพื้นฐาน

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ที่ถูกใช้ทำงานโดยสัญญาณอินพุทที่เป็นพัลส์ ทุกๆสัญญาณพัลส์ที่ให้เข้ามาจะให้การเปลี่ยนแปลงสถานะและสนามแม่เหล็กไฟฟ้าที่เกิดขึ้น และให้การหมุนของมอเตอร์เป็นมุมที่คงที่ ซึ่งจะแตกต่างกับการหมุนของมอเตอร์แบบธรรมดาในระบบควบคุมที่ใช้สัญญาณดิจิตอลนี้ เช่น ในการส่งข้อมูลการควบคุมข้อมูลโดยที่เป็นปริมาณค่าสัญญาณดิจิตอลทั้งหมด จึงทำให้การควบคุมการทำงานของสเต็ปปิ้งมอเตอร์โดยตรงได้เป็นอย่างดี การทำงานพื้นฐานของสเต็ปปิ้งมอเตอร์แสดงในรูปที่ 2.15 ซึ่งแสดงบล็อกขอวงจรขับมอเตอร์ ทำให้กระแสไฟดีซีเรียงลำดับเข้าตามเฟสต่างๆของมอเตอร์หมุนไปตามองศาที่กำหนดไว้ การที่จะทำให้กระแสไฟดีซีเรียงลำดับเข้าไปที่มอเตอร์ได้ ทำได้ด้วยการใช้งานของสวิตช์ และเพื่อให้มอเตอร์หมุน จำเป็นต้องมีวงจรขับสเต็ปปิ้งที่จะควบคุมการไหลของกระแสไฟ คือ สวิตช์ 1 และ 2 ถ้าให้การทำงานของสวิตช์ตามตารางการทำงานที่แสดงไว้ ถ้าให้การทำงานโดยใช้รีเลย์ (Relay) หรือมือโยก สเต็ปปิ้งมอเตอร์ก็จะหมุนเหมือนกันแต่หมุนช้าเนื่องจากรีเลย์หรือมือโยกให้การเปลี่ยนแปลงของสวิตช์ช้า ซึ่งจะทำได้เร็วเท่ากับความถี่สูงสุดของสัญญาณพัลส์ที่มอเตอร์ทำงานสามารถทำงานได้ (เป็นจำนวนพัลส์ต่อวินาที)



รูปที่ 2.15 การทำงานพื้นฐานของสเต็ปปิ้งมอเตอร์.

2.3.4 ชนิดของวงจรขับมอเตอร์ที่นิยมใช้มีอยู่ 3 ระบบ คือ

2.3.4.1 ระบบการกระตุ้นแม่เหล็กเฟสเดียว (Single Phase Excitation)

ตารางที่ 2.3 แสดงการกระตุ้นแม่เหล็กเฟสเดียว

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

2.3.4.2 ระบบการกระตุ้นสนามแม่เหล็ก 2 เฟส (2-Phase Excitation)

ตารางที่ 2.4 แสดงการกระตุ้นสนามแม่เหล็ก 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-

3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

2.3.4.3 ระบบการกระตุ้นสนามแม่เหล็กครึ่งเฟส (Half-Step Excitation)

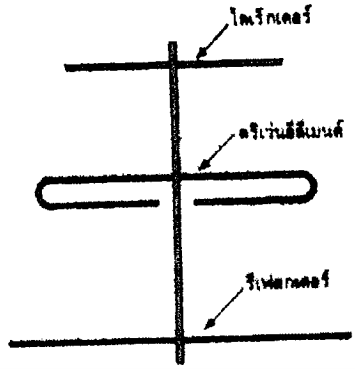
ตารางที่ 2.5 แสดงการกระตุ้นสนามแม่เหล็กครึ่งเฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

2.4 สายอากาศแบบยากิ (Yagi-Uda Antenna)

2.4.1 ลักษณะทั่วไปของสายอากาศยากิ

สายอากาศยากิ ประกอบด้วย 3 ส่วนคือ ไดเร็กเตอร์ ดริเว่นอีลีเมนต์ และรีเฟลกเตอร์ ดังรูปที่2.16

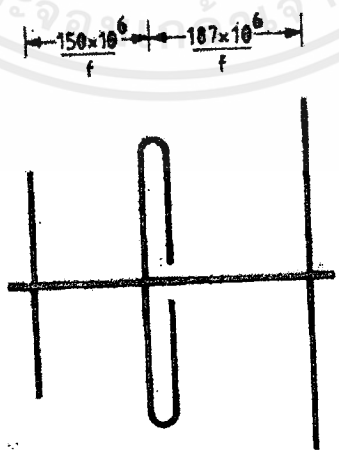


รูปที่ 2.16 ลักษณะทั่วไปของสายอากาศยากิ

จากรูปที่ 2.16 จะเห็นว่าส่วนที่สั้นที่สุดเรียกว่าไดเร็กเตอร์ ทำหน้าที่นำทางคลื่นให้เข้ามาที่ดริเว่นอีลีเมนต์ซึ่งเป็นส่วนที่ถัดมา ดริเว่นอีลีเมนต์นี้ทำหน้าที่นำคลื่นที่ได้ไปเข้าเครื่องรับ ส่วนสุดท้ายที่เป็นส่วนที่ยาวที่สุด ทำหน้าที่สะท้อนคลื่นที่อาจเลยมาจากดริเว่นอีลีเมนต์ให้กลับเข้าไปดริเว่นอีลีเมนต์อีกครั้ง เพื่อทำให้รับคลื่นได้เต็มที่นั่นเอง

ส่วนสำคัญที่สุดของสายอากาศยากิอยู่ที่ตัวดริเว่นอีลีเมนต์ การสร้างจะต้องเริ่มสร้างตัวดริเว่นอีลีเมนต์นี้ก่อนซึ่งตัวนี้จะมีความยาวเป็นครึ่งหนึ่งของความยาวคลื่น ไดเร็กเตอร์จะสั้นกว่าดริเว่นอีลีเมนต์ 5 เปอร์เซ็นต์ ส่วนรีเฟล็กเตอร์จะยาวกว่าดริเว่นอีลีเมนต์ 5 เปอร์เซ็นต์ และระยะห่างระหว่างส่วนประกอบทั้งสามก็ไม่เท่ากันด้วย

จากรูปที่ 2.17 ระยะห่างระหว่างไดเร็กเตอร์กับตัวดริเว่นอีลีเมนต์ และดริเว่นอีลีเมนต์กับรีเฟลกเตอร์มีหน่วยเป็นฟุต



รูปที่ 2.17 ระยะต่าง ๆ ของสายอากาศแบบยากิ

การคำนวณหาความยาวของส่วนประกอบทั้งสามโดยเริ่มจากตัวดริเว่นอีลีเมนต์ ซึ่งจะต้องหาค่าความยาวคลื่นก่อน

จากสูตร $\lambda = v/f = 300 \times 10^6/F$

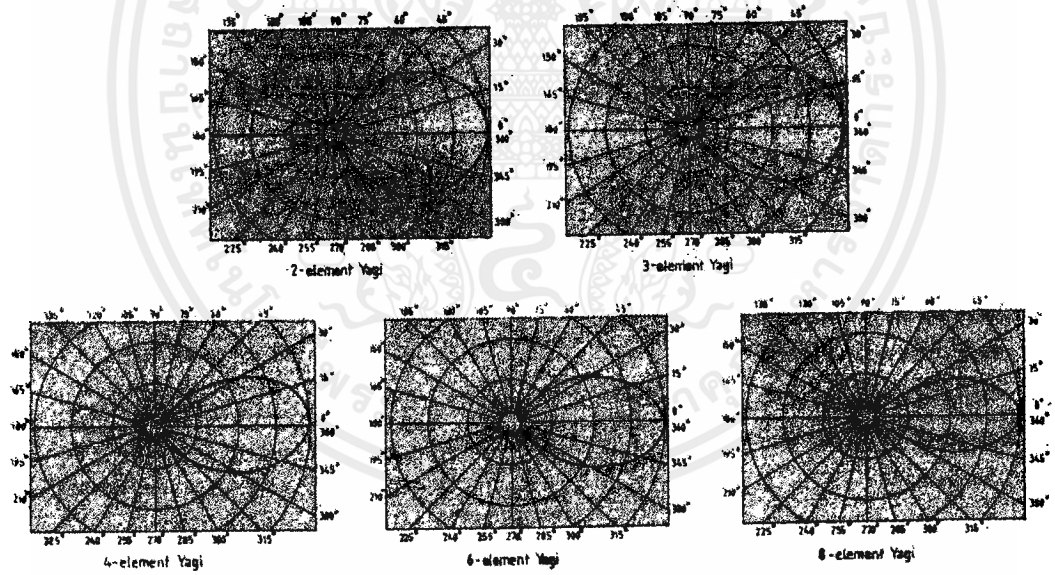
จากสายอากาศยาว $\lambda/2$ เพราะฉนั้น $\lambda = 3 \times 10^6/2F = 150 \times 10^6/F$ เมตร

ทำให้เป็นหน่วยฟุต คูณด้วย3.28 $\lambda = 150 \times 10^6 \times 3.28/F$ ฟุต $= 492 \times 10^6/F$ ฟุต

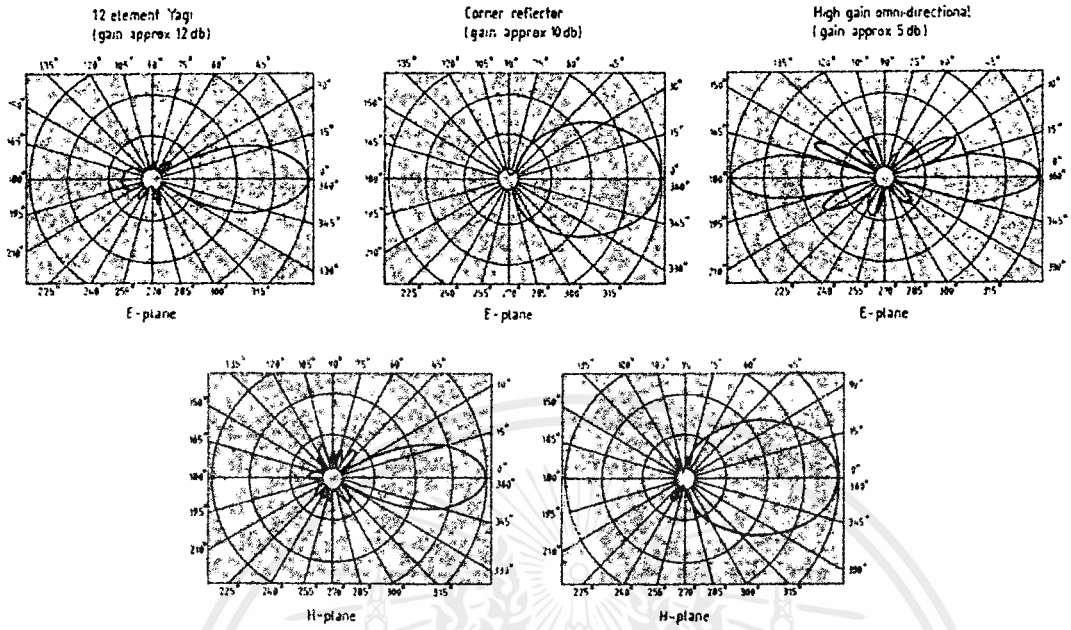
แต่เนื่องจากคลื่นวิทยุซึ่งเป็นคลื่นแม่เหล็กไฟฟ้าเดินท้างในอากาศได้เร็วกว่าในตัวนำที่นำมาทำสายอากาศ ดังนั้นจะต้องคูณด้วยค่าคงที่ความเร็ว 0.95 เข้าไป

สายอากาศยาว $492 \times 10^6 \times 0.95/F = 468 \times 10^6/F$ ฟุต

จากรูปที่ 2.18 ซึ่งเป็นลักษณะการแพร่กระจายคลื่นของสายอากาศในย่านวีเอชเอฟ และยูเอชเอฟ จะสังเกตเห็นว่าถ้าจำนวนชิ้น ของสายอากาศยากิมากขึ้น จะทำให้การรับคลื่นเป็นลำแคบลง นั่นคือ จะทำให้การรับสัญญาณจะชัดเจนยิ่งขึ้น เช่นเดียวกับรูปที่ 2.19 ซึ่งประสิทธิภาพการรับนี้ก็ขึ้นอยู่กับการจัดระยะห่างของส่วนประกอบของสายอากาศให้เหมาะสม



รูปที่ 2.18 ลักษณะการแพร่กระจายคลื่นในย่านวีเอชเอฟ



รูปที่ 2.19 ลักษณะการแพร่กระจายคลื่นในย่านยูเอชเอฟ

2.4.2 ผลของตัวจับยึดของสายอากาศย่านวีเอชเอฟ และยูเอชเอฟ

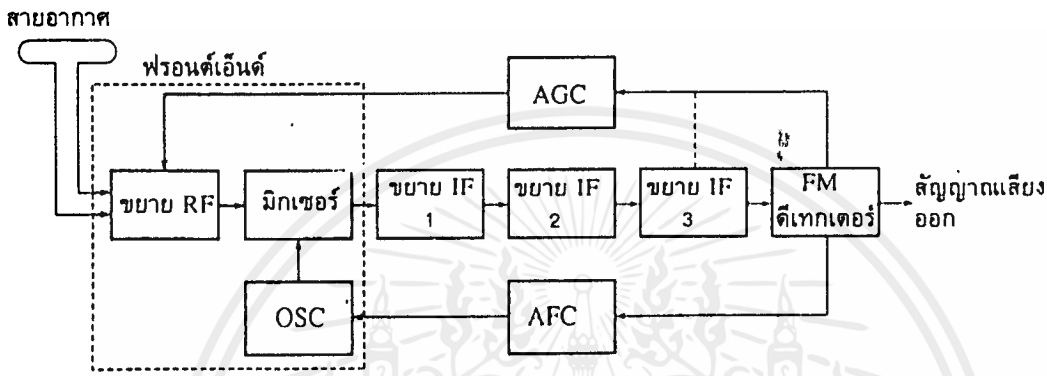
1. ผลกระทบทางด้านอิมพีแดนซ์ของสายอากาศ โดยทั่วไปสายอากาศยากิจะมีอิมพีแดนซ์ประมาณ 70 โอห์ม ขณะอยู่ในอากาศ แต่ถ้านำสายอากาศนี้วางไว้ใกล้โลหะ (ในที่นี้คือ ตัวจับยึด) ที่มีผิวหน้าไม่น้อยกว่า 1 ความยาวคลื่น จะทำให้ค่าอิมพีแดนซ์ของสายอากาศนั่นเปลี่ยนแปลงไปได้ และทำให้ค่า VSWR เปลี่ยนแปลงไปด้วย ดังแสดงในตารางที่ 2.6
2. ผลกระทบเกี่ยวกับรูปลักษณะการแพร่กระจายคลื่น ถ้าโครงสร้างของตัวจับยึดมีขนาดใหญ่ขึ้นเพียงใด การแพร่กระจายคลื่นทางด้านหลังของสายอากาศก็จะถูกจำกัดลงไปเพียงนั้น โดยเฉพาะเสาอากาศที่มีโครงสร้างเป็นโลหะทึบ หรือส่วนเป็นตะแกรงลวดโลหะ ซึ่งมีความยาวด้านทะแยงน้อยกว่า $\lambda/4$

ตารางที่ 2.6 การเปลี่ยนแปลงของอิมพีแดนซ์ อันเนื่องมาจากตัวจับยึด

ระยะห่างจากโครงสร้าง	อิมพีแดนซ์ของสายอากาศ
$\lambda/2$	70 Ω
$\lambda/3$	100 Ω
$\lambda/4$	83 Ω
$\lambda/6$	50 Ω
$\lambda/10$	20 Ω

2.5 เครื่องรับเอฟเอ็ม

การทำงานของเครื่องรับวิทยุเอฟเอ็ม ตามบล็อกไดอะแกรมในรูปที่ 2.20 อธิบายได้ดังต่อไปนี้



รูปที่ 2.20 บล็อกไดอะแกรมของเครื่องรับเอฟเอ็ม

เสาอากาศ จำเป็นมากสำหรับเครื่องรับเอฟเอ็ม ทำหน้าที่รับคลื่นวิทยุเอฟเอ็มจากสถานีส่งต่าง ๆ ความถี่ระหว่าง 88-108 MHz ให้มีความแรงเท่า ๆ กันทุกสถานี เสาอากาศจึงต้องรับคลื่นในแบนด์กว้างตลอดย่านเอฟเอ็ม เสาอากาศอาจเป็นแบบติดอยู่กับตัวเครื่องรับ หรือเสาอากาศภายนอก

ภาคขยายอาร์เอฟ หรือภาคอาร์เอฟแอมป์ ทำหน้าที่ขยายสัญญาณความถี่ที่รับมาจากสายอากาศโดยเลือกขยายทีละ 1 ความถี่ ดังนั้นจึงต้องมีวงจรจูนสัญญาณอินพุทที่รับมาจากสายอากาศ และจูนสัญญาณเอาท์พุทที่ส่งให้ภาคมิกเซอร์ ซึ่งการจูนต้องได้ความถี่ต่ำเดียวกัน ตัวขยายสัญญาณอาร์เอฟ สมัยก่อนใช้หลอดสุญญากาศ ปัจจุบันใช้ทรานซิสเตอร์หรือเฟต บางรุ่นจะใช้ไอซี การขยายอาร์เอฟจะทำให้สัญญาณที่รับได้มีขนาดแรงขึ้น สัญญาณที่ออกจากภาคขยายอาร์เอฟนี้เป็นสัญญาณความถี่ของสถานีที่ต้องการรับ และส่งออกไปเข้าภาคมิกเซอร์เพียง 1 ความถี่เท่านั้น อัตราการขยายของภาคขยายอาร์เอฟถูกควบคุมโดยวงจรเอจีซี

ภาคออสซิลเลเตอร์ หรือโลคอลออสซิลเลเตอร์ (local oscillator) ทำหน้าที่ผลิตความถี่อาร์เอฟย่านวีเอชเอฟขึ้นมา 1 ความถี่ เพื่อที่จะส่งเข้าไปผสมกับคลื่นที่รับมาจากภาคขยายอาร์เอฟอยู่ 10.7 MHz นั่นคืออยู่ในช่วง 98.7 ถึง 118.7 MHz ความถี่ออสซิลเลเตอร์ผลิตจะต้องคงที่เสมอ ซึ่งควบคุมโดยวงจรเอเอฟซี

ภาคมิกเซอร์ ทำหน้าที่รับคลื่นจากภาคขยายอาร์เอฟ และภาคออสซิลเลเตอร์ มาผสมกันตามกระบวนการซูเปอร์เฮเทอร์ดายน์ โดยใช้ผลต่างของออสซิลเตอร์กับสัญญาณจากภาคขยายอาร์เอฟ มีค่า 10.7 MHz ถูกจูนออกทางเอาท์พุทของมิกเซอร์

ภาคขยายอาร์เอฟ ภาคออสซิลเลเตอร์ และภาคมิกเซอร์ เมื่อรวมกันจะเรียกว่า ภาคฟรอนต์เอ็นด์ (front end) อาจบรรจุไว้ในกล่องโลหะชีลด์ เพื่อป้องกันความถี่อื่นรบกวน โดยจะโผล่ออกมาเฉพาะจุดต่อสาย และแกนของตัวเก็บประจุแบบปรับค่าได้ที่หมุนเปลี่ยนความถี่

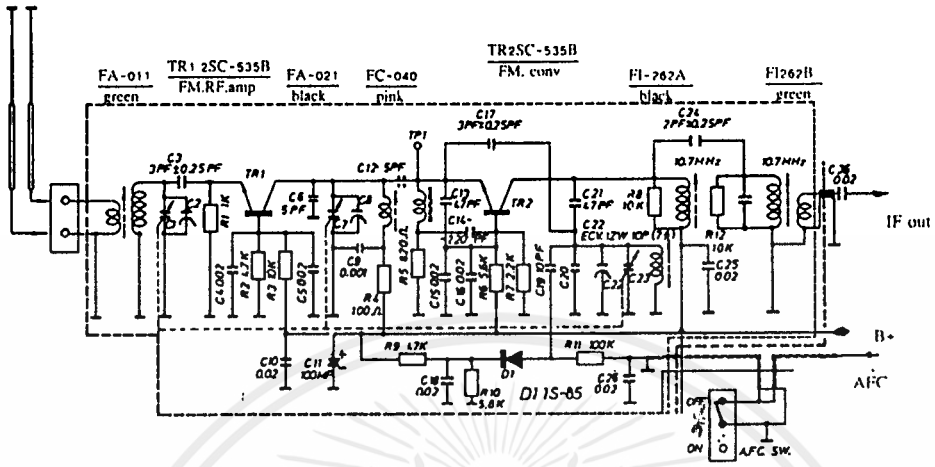
ภาคขยายสัญญาณความที่ไอเอฟ โดยรับสัญญาณไอเอฟจากเอาท์พุทของภาคมิกเซอร์ด้วยความถี่ 10.7 MHz มาทำการขยายติดต่อกัน 3 ครั้ง (บางวงจรอาจขยายน้อย หรือมากครั้งกว่านี้) การขยายมีหม้อแปลงจูนเชื่อมโยงระหว่างภาคแบนด์วิดธ์ของสัญญาณไอเอฟต้องกว้างประมาณ 200 kHz จึงจะได้สัญญาณเสียงที่ครบถ้วน การขยายสัญญาณไอเอฟ อาจใช้หลอดสุญญากาศ ทรานซิสเตอร์ หรือไอซี การขยายสัญญาณไอเอฟครั้งที่ 3 อาจจะทำหน้าที่เป็นลิมิตเตอร์ด้วยก็ได้ โดยตั้งจุดไบแอสของทรานซิสเตอร์ให้สูงกว่าปกติ

ภาคเอฟเอ็มดีเทกเตอร์ ทำหน้าที่ตรวจจับเพื่อแยกสัญญาณเสียงออกจากคลื่นไอเอฟของเอฟเอ็มความถี่ 10.7 MHz การดีเทกเอฟเอ็มมีหลายวิธี บางวิธีต้องเพิ่มวงจรลิมิตเตอร์ เพื่อตัดแต่งสัญญาณเอฟเอ็มให้มีแอมปลิจูดเท่ากันเสียก่อน แต่วิธีที่นิยมกันคือ เรโชดีเทกเตอร์ โดยประหยัดไม่ต้องมีภาคลิมิตเตอร์ สัญญาณเสียงที่ได้จะส่งไปเข้าวงจรขยายเสียงต่อไป ผลของการดีเทกเตอร์จะได้สัญญาณบางส่วนนำไปสร้างเป็นสัญญาณเอเอฟซีและสัญญาณเอจีซี

ภาคเอเอฟซี (automatic frequency control) ทำหน้าที่ควบคุมความถี่ของออสซิลเลเตอร์ให้พอดีล็อคกับคลื่นอาร์เอฟตลอดเวลา เพราะถ้าออสซิลเลเตอร์ผลิตความถี่ไม่สัมพันธ์กับคลื่นอาร์เอฟ สัญญาณเสียงที่รับฟังจะจางหายไม่ชัดเจน ซึ่งคล้ายกับรับไม่ตรงสถานี สัญญาณแรงดันไฟเอเอฟซีได้มาจากภาคเอฟเอ็มดีเทกเตอร์

ภาคเอจีซี ทำหน้าที่ควบคุมอัตราการขยายของภาคขยายอาร์เอฟให้คงที่สม่ำเสมอกันทุก ๆ สถานี สัญญาณที่นำไปควบคุมเอจีซีอาจนำมาจากภาคขยายไอเอฟหรือภาคเอฟเอ็มดีเทกเตอร์ก็ได้

2.5.1 ภาคฟรอนต์เอ็นด์ ภาคฟรอนต์เอ็นด์ประกอบด้วยวงจรขยายอาร์เอฟ วงจรออสซิลเลเตอร์ และวงจรมิกเซอร์ เนื่องจากวงจรจะทำงานเกี่ยวข้องกับความถี่สูงย่านวีเอชเอฟ โดยขาของอุปกรณ์ในภาคนี้จะสั้นและอยู่ชิดกัน อาจมีกระป๋องหรือแผ่นโลหะปิดกั้นสำหรับชีลด์ความถี่ต่าง ๆ ไม่ให้รบกวน จากวงจรรูปที่ 2.21 ใช้ทรานซิสเตอร์เพียง 2 ตัวเท่านั้น TR_1 ทำหน้าที่ขยายอาร์เอฟ โดยต่อเป็นวงจรคอมมอนเบส สัญญาณวิทยุเอฟเอ็มจากสายอากาศจะเชื่อมโยงผ่านหม้อแปลงอาร์เอฟ (RA-001) แล้วถูกจูนเลือกความถี่ด้วย C_1 สัญญาณวิทยุที่เลือกแล้วจะส่งไปขยายใน TR_1 ผ่านทาง C_3 (3pF) เข้าขาอีมิตเตอร์ของ TR_1 สัญญาณที่ขยายแล้วออกทางขาคอลเลกเตอร์ซึ่งมีวงจรจูนที่เอาท์พุท C_6, C_7, C_8 ขดลวด FR-021 จูนเลือกสัญญาณอาร์เอฟส่งผ่าน C_{12} (5pF) ไปเข้าภาคคอนเวอร์เตอร์ทางขาอีมิตเตอร์ของ TR_2 ทั้ง TR_1 และ TR_2 ทำงานแบบคอมมอนเบส



รูปที่ 2.21 วงจรฟรอนต์เอ็นด์แบบทรานซิสเตอร์ 2 ตัว

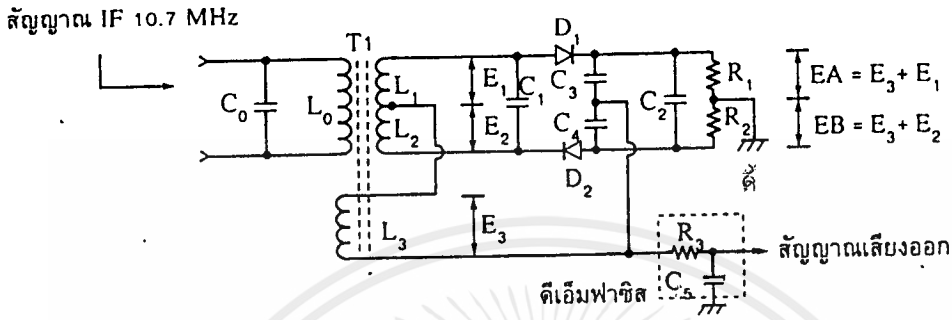
TR_2 ทำหน้าที่เป็นตัวคอนเวอร์เตอร์ คือเป็นทั้งออสซิลเลเตอร์ และมิกเซอร์ ความถี่ออสซิลเลเตอร์ถูกำหนดด้วยวงจรเรโซแนนซ์ C_{20}, C_{22}, C_{23} และขดลวด FO-301 การเปลี่ยนความถี่ต้องปรับความจุของ C_{23} ซึ่งมีแกนกลางร่วมกันกับ C_1 และ C_7 จึงทำให้ความถี่ที่ออสซิลเลเตอร์ได้มีความสัมพันธ์กันพอดี โดยส่งผ่าน C_{17} (3pF) เข้าวงจรมิกเซอร์ สัญญาณอาร์เอฟที่ขยายใน TR_2 ทั้งหมดถูกส่งออกทางขาดอลเลกเตอร์ สัญญาณไอเอฟ 10.7 MHz และถูกจูนออกด้วยหม้อแปลงไอเอฟเบอร์ FI - 262A แล้วเชื่อมโยงด้วย C_{24} (2pF) ไปเข้าวงจรเรโซแนนซ์ของหม้อแปลงไอเอฟเบอร์ FI - 262B ถูกจูน 10.7 MHz ออกทางขดทุติยภูมิ เพื่อออกไปยังภาคขยายไอเอฟต่อไป R_8 และ R_{12} ต่อขนานกับขดลวดหม้อแปลงไอแอฟทั้งสอง เพื่อเพิ่มความกว้างของแบนด์วิดธ์ไอเอฟ

การควบคุมความถี่ของออสซิลเลเตอร์ กระทำโดยวงจรเอเอฟซีซึ่งประกอบด้วย C_{19} (10 pF) วาแรกเตอร์ไดโอด D_1 (1S-85) และ C_{18} (0.02) ต่ออนุกรมกันแล้วขนานอยู่กับวงจรแท้งก์ของออสซิลเลเตอร์ R_9 และ R_{10} เป็นตัวจัดไบแอสให้กับ D_1 ส่วน R_{11} เป็นตัวรับสัญญาณเอเอฟซีจากภาคเอฟเอ็มดีเทกเตอร์ เมื่อแรงดันเอเอฟซีเปลี่ยนแปลง ค่า C ภายใน D_1 จะเปลี่ยนแปลงความถี่เรโซแนนซ์ของวงจรแท้งก์ จึงทำให้ความถี่ของออสซิลเลเตอร์เปลี่ยนแปลงเบี่ยงเบนตามการควบคุมของเอเอฟซีให้ได้ความถี่ไอเอฟเข้าหาความถี่ที่ถูกต้องอยู่เสมอ

2.5.2 เรโชดีเทกเตอร์

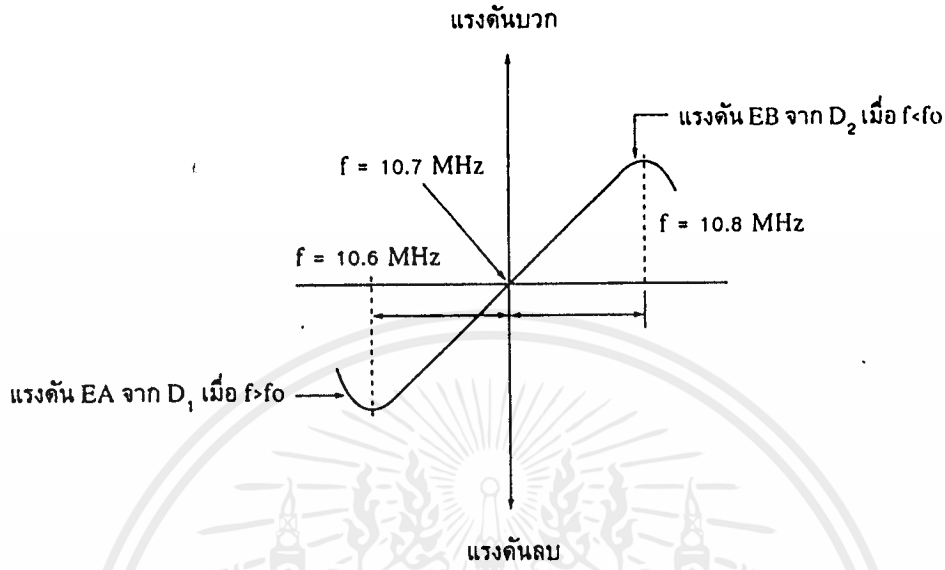
เรโชดีเทกเตอร์จะใช้ในวงจรเครื่องรับวิทยุเอฟเอ็ม เพื่อถอดเอาสัญญาณเสียงออกมาจากสัญญาณไอเอฟ กว่าที่จะพัฒนามาเป็นเรโชดีเทกเตอร์มีการแยกสัญญาณเอฟเอ็มที่เรียกชื่ออย่างอื่น เช่น ทราวิสดิสคริมิเนเตอร์ สโลปดีเทกเตอร์ เป็นต้น แต่วงจรดังกล่าวจะมีข้อเสียคือ สร้างและปรับแต่งยาก รวมทั้งต้องเพิ่มวงจรลิมิเตอร์อยู่ข้างหน้าทำให้สิ้นเปลือง เครื่องรับวิทยุเอฟเอ็ม จึงหันมาใช้เรโชดีเทกเตอร์ ซึ่งถอดแยกเอาสัญญาณ

เสียงออกจากดีเทคเตอร์เอฟเอ็มได้ดี การรบกวนขจัดได้ง่าย ปรับแต่งง่ายไม่ต้องมีภาคลิมิเตอร์ และผลจากการถอดแยกสัญญาณยังได้สัญญาณเพื่อการควบคุมเอเอฟซี และควบคุมเอจีซี อีกด้วย



รูปที่ 2.22 วงจรเอฟเอ็มเรโชดีเทกเตอร์

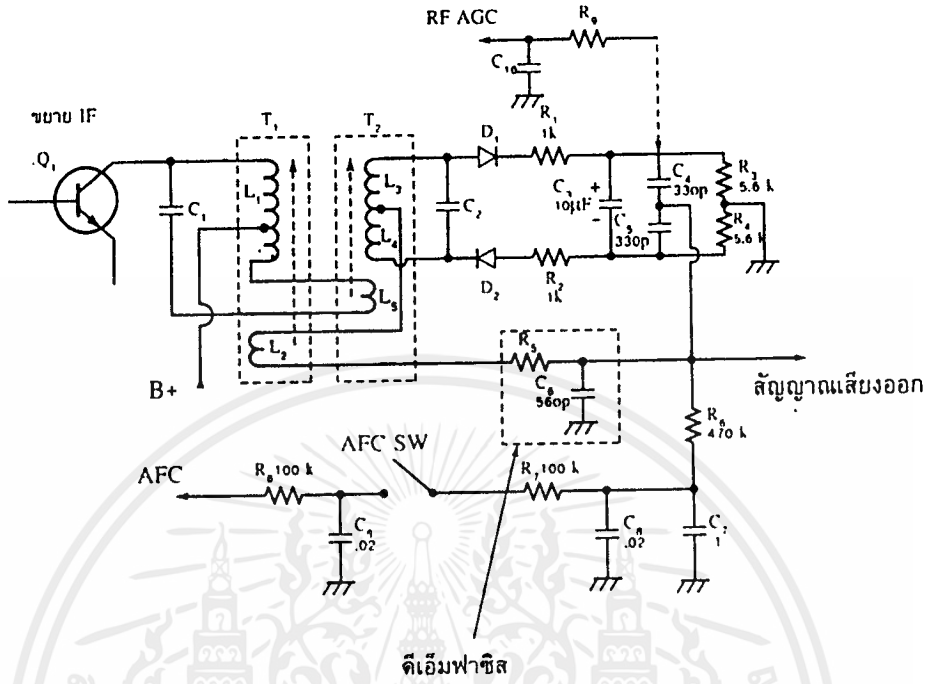
วงจรดีเทกเตอร์มี 2 แบบ คือ วงจรแบบสมดุลย์ และ วงจรแบบไม่สมดุลย์ (balanced and unbalanced) วงจรในรูปที่ 2.22 แสดงวงจรแบบสมดุลส่วนสำคัญคือ T_1 เป็นหม้อแปลงเรโชดีเทกเตอร์ มี D_1 และ D_2 เป็นไดโอดดีเทกเตอร์ T_1 จูนให้เรโซแนนซ์กับความถี่ไอเอฟ 10.7 MHz L_0 - C_0 จูนด้านปฐมภูมิ ส่วน C_1 -L_1 -L_2 จูนด้านทุติยภูมิ L_3 พันอยู่ใกล้ L_0 และมีเฟสเดียวกับ L_0 ขดปลายของ L_3 ต่อกับกึ่งกลางของขด L_1 - L_2 เมื่อ L_0 เชื่อมโยงสัญญาณจะเหนี่ยวนำให้เกิดแรงดันขึ้นในขดลวด L_1-L_2 และ L_3 ซึ่งเฟสต่างกัน สัญญาณ EA = E3 + E1 และ EB = E3 + E2 เมื่อสัญญาณไอเอฟ 10.7 MHz เข้ามา การตัดแยกสัญญาณจะได้แรงดัน EA = EB แรงดันทั้งสองหักล้างกันพอดี เพราะไหลสวนทางกัน และที่เอาท์พุทไม่มีสัญญาณเสียงออกจึงเป็นศูนย์ถ้าหากมีสัญญาณ 10.8 MHz เข้ามาหรือสัญญาณที่สูงกว่า 10.7 MHz เข้ามากระแสจะผ่าน D_2 มาก การตัดแยกสัญญาณได้แรงดัน EB สั้นกว่า EA ได้สัญญาณเสียงช่วงบวกออกทาง L_3 และในกรณีความถี่ต่ำกว่า 10.7 MHz เข้ามา D1 จะตัดแยกสัญญาณได้กระแสไหลผ่านมาก แรงดัน EA จะสูงกว่า EB ได้สัญญาณเสียงช่วงลบออกทาง L3 และก่อนที่สัญญาณเสียงจะออกไปจากวงจรจะต้องผ่านวงจรดีเอ็มฟาซิสประมาณ 75ms ด้วย R_3 และ C_5 เพื่อลดระดับความถี่สูงลงให้พอเหมาะกับการฟัง ถ้าหากการปรับแต่งวงจรเอฟเอ็มดีเทกเตอร์ถูกต้องกราฟสัญญาณเสียงจะเป็นเอสเคิร์ฟ (S-curve) ดังรูปที่ 2.23



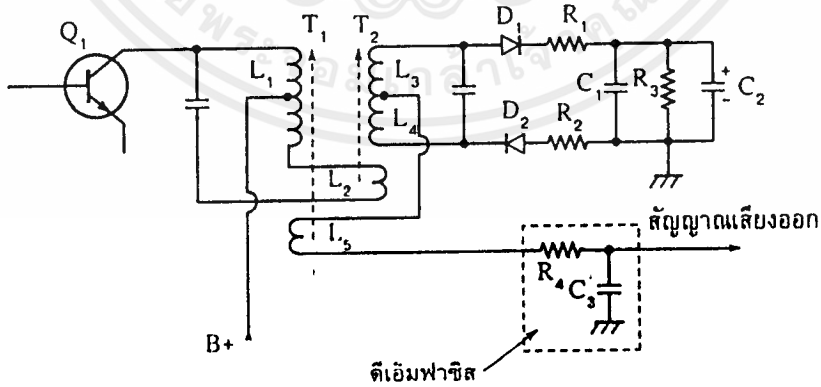
รูปที่ 2.23 เอสเคิร์ฟจากเอฟเอ็มเรโชดีเทกเตอร์

จากรูปที่ 2.24 เป็นวงจรเรโชดีเทกเตอร์แบบสมดุลซึ่งใช้งานในเครื่องรับทั่ว ๆ ไป ที่เรียกว่าวงจรแบบสมดุลก็เพราะใช้ค่าต่าง ๆ เท่ากัน เช่น ขดลวด $L_3 = L_4$, $D_1 = D_2$, $R_1 = R_2$, $C_4 = C_5$, $R_3 = R_4$ วงจรนี้แยกหม้อแปลงเป็น 2 ตัว T_1 มีขดลวด L_1 และ L_2 ส่วน T_2 มีขดลวด L_3 - L_4 และ L_5 ขดลวด L_5 ต่ออนุกรมกับ L_1 เป็นวงจรเรโซแนนซ์ร่วมกับ C_1 เพื่อจูนเอาความถี่ไอเอฟจาก Q_1 แล้วส่งผ่านไปให้ขดลวด L_3-L_4 ส่วนขดลวด L_2 นั้นต่ออยู่กับขากลางของขดลวด L_3 - L_4 เพื่อต่อแยกเอาสัญญาณเสียงออก และที่แปลกไปก็คือ มี R_1 อนุกรมกับ D_1 และ R_2 อนุกรมกับ D_2 เพื่อการประจุกระแสแก่ C_3 การทำงานในวงจรนี้เช่นเดียวกับที่ได้อธิบายไว้แล้วในรูปที่ 2.22

ตัวอย่างเรโชดีเทกเตอร์อีกแบบหนึ่งดังรูปที่ 2.25 เป็นแบบไม่สมดุลซึ่งจะประหยัดอุปกรณ์ในภาคดีเทกเตอร์คือ C_1, R_3 และ C_2 จะมีอย่างละตัว อุปกรณ์อื่นยังคงเดิม เช่น D_1, D_2, T_1, T_2 สัญญาณเสียงที่ตัดแยกได้ผ่านออกทางจุดกึ่งกลางของ L_3 กับ L_4 ผ่าน L_5 เข้าวงจรดีเอ็มฟาซิส R_4 กับ C_3 เป็นสัญญาณเสียงออกไปเพื่อเข้าภาคขยายเสียงต่อไป วงจรเรโชดีเทกเตอร์แบบไม่สมดุลนิยมใช้กับการแยกสัญญาณเสียงในเครื่องรับโทรทัศน์บางรุ่น



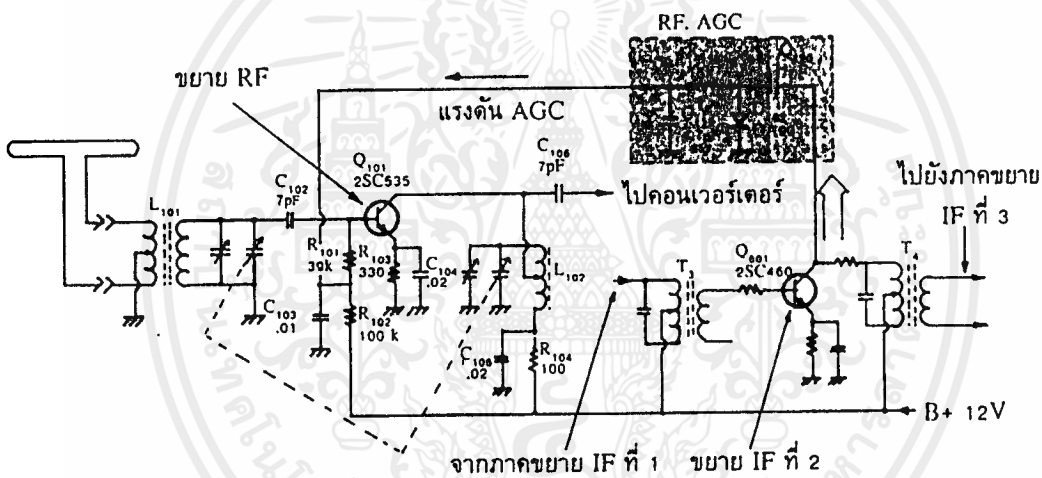
รูปที่ 2.24 วงจรเรโซดีเทกเตอร์ แบบสมดุล



รูปที่ 2.25 เรโซดีเทกเตอร์แบบไม่สมดุล

2.5.3 วงจรอาร์เอฟ-เอจีซี

การขยายคลื่นอาร์เอฟของเอฟเอ็มอาจจะใช้ทรานซิสเตอร์ธรรมดาขยายแบบคอมมอนอีมิตเตอร์หรือคอมมอนเบส หรือใช้ทรานซิสเตอร์แบบเฟต ปัญหาที่เกิดขึ้นกับภาคขยายอาร์เอฟก็คือมีอัตราขยายคงที่เมื่อมีสัญญาณเอฟเอ็มจากสถานีที่มีกำลังส่งแรงจะได้เอาท์พุทแรง และเมื่อรับคลื่นจากสถานีกำลังส่งต่ำ เอาท์พุทที่ได้ก็จะต่ำ นั่นก็คือ ความไวในการรับคลื่นวิทยุจะน้อยนั่นเอง เพื่อให้การรับคลื่นวิทยุเอฟเอ็มมีความไวเท่าเทียมกันทุก ๆ สถานี จำเป็นจะต้องควบคุมอัตราการขยายของวงจรขยายอาร์เอฟให้เป็นไปโดยอัตโนมัติด้วยวงจรเอจีซี ความต้องการของเราก็คือ เมื่อรับคลื่นสถานีที่มีความแรงมาก การขยายของภาคอาร์เอฟจะลดลง และเมื่อรับคลื่นสถานีที่มีกำลังอ่อน การขยายของภาคอาร์เอฟจะเพิ่มขึ้นเพื่อชดเชยกัน และให้ได้เอาท์พุทมีความแรงพอ ๆ กัน สัญญาณควบคุมเอจีซีจะนำมาจากภาคขยายไอเอฟ หรือนำมาจากภาคเอฟเอ็มดีเทกเตอร์ก็ได้



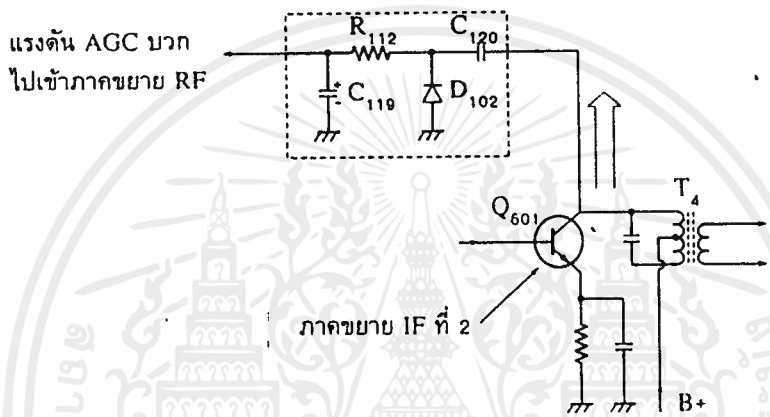
รูปที่ 2.26 การตัดแยกสัญญาณเอจีซี จากวงจรขยายไอเอฟ

รูปที่ 2.26 ทรานซิสเตอร์ Q_{101} เบอร์ 2SC535 เป็นชนิดเอ็นพีเอ็น ต่อแบบคอมมอนอีมิตเตอร์ร่วม แรงดันไฟตรงไบแอสขาเบสเป็นไฟบวก ได้จากแหล่งจ่ายไฟ 12 โวลท์ผ่านทาง R_{102} (100k) และ R_{101} (39k) เป็นแรงดันคงที่ ถ้ามีแรงดันอยู่อย่างนี้ การขยายของ Q_{101} ก็จะคงที่ เอาท์พุทที่ได้ขึ้นอยู่กับขนาดของสัญญาณอินพุท

Q_{601} เบอร์ 2SC460 ทำหน้าที่ขยายสัญญาณไอเอฟครั้งที่ 2 มี T_4 เป็นหม้อแปลงไอเอฟสำหรับจูนเอาสัญญาณ 10.7 MHz ออกไปเข้าภาคขยายไอเอฟที่ 3 ที่ขาคอลเลกเตอร์ของ Q_{601} มี C_{120} (7pF) ต่อไว้เพื่อดึงเอาสัญญาณไอเอฟออกไปเข้าวงจรเอจีซี AGC ดีเทกเตอร์ D_{102} (OA90) ตัดแยกสัญญาณซีกบวกลงกราวด์ที่แอโนดของ D_{102} จึงเป็นไฟขั้วลบ ผ่าน R_{112} (15k) แล้วกรองแรงดันด้วย C_{119} (4.7MFD/25V)เพื่อให้เป็นไฟ DC ที่เรียบ แรงดันไฟลบนี้คือแรงดันเอจีซี ใช้สำหรับป้อนเข้าขาเบสของทรานซิสเตอร์ขยาย อาร์เอฟผ่านทาง R_{101} (39k) แรงดันเอจีซีนี้จะมากขึ้นหรือน้อยลงได้ขึ้นอยู่กับความแรงของสัญญาณเอฟเอ็มที่รับเข้ามา เช่น เมื่อรับสัญญาณ

จากสถานีที่มีความแรงมากสัญญาณที่ผ่านไปถึงภาคขยายไอเอฟก็จะสูง ทำให้การแยกสัญญาณเอจีซี AGC ได้แรงดันไฟลบสูง แรงดันลบนี้จะเข้ามาหักล้างกับไฟบวกที่ไบแอสขาเบสของ Q_{101} อาจเกิดเพียงเล็กน้อย Q_{101} ก็ได้รับไบแอสบวกเพิ่มขึ้น การขยายสัญญาณอาร์เอฟ ก็แรงขึ้น ได้สัญญาณเอาต์พุตมีความแรงพอดีเท่าเทียมกันทุก ๆ สถานี

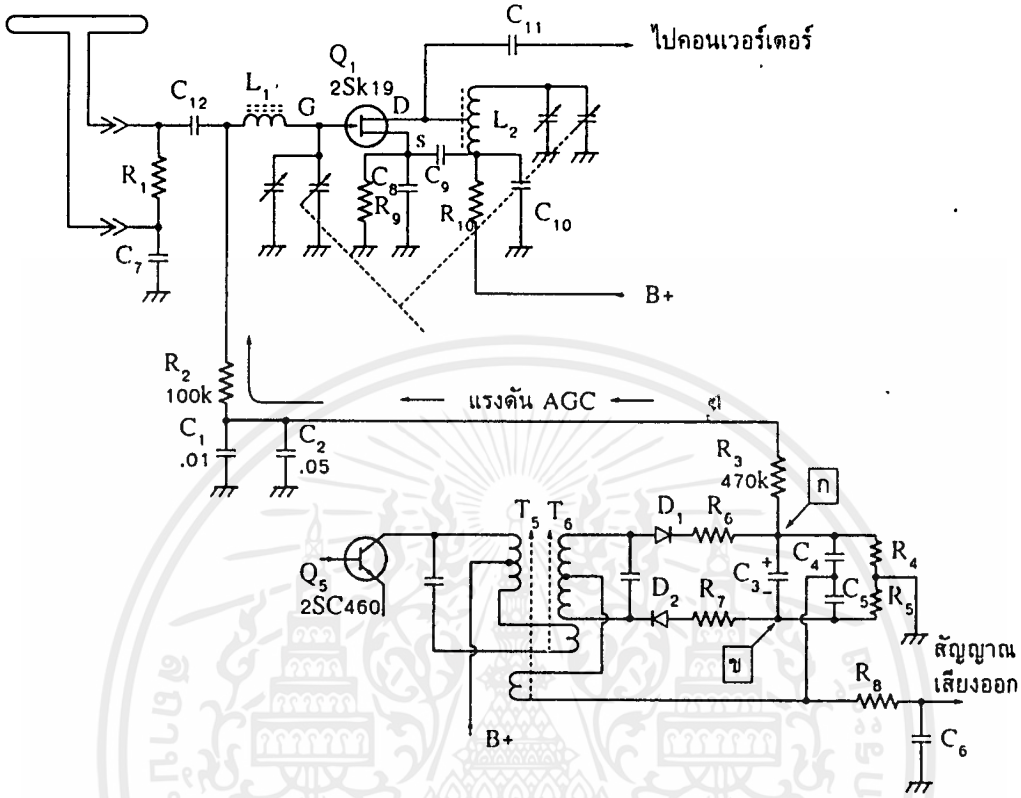
แรงดันเอจีซีถ้าหากต้องการแรงดันบวกเพื่อไปควบคุมทรานซิสเตอร์แบบพีเอ็นพี ให้ทำการแยกสัญญาณเอจีซีใหม่โดยกลับขั้วของไดโอดและกลับขั้วของตัวเก็บประจุกรองแรงดันใหม่ ดังรูปที่ 2.27



รูปที่ 2.27 วงจรเอจีซีดีเทกเตอร์ได้สัญญาณบวก

2.5.4 สัญญาณเอจีซีจากเรโชดีเทกเตอร์

การแยสัญญาณเอฟเอ็มโดยใช้เรโชดีเทกเตอร์ตามรูปที่ 2.28 ไดโอด D_1 และ D_2 ได้กระแสประจุใน C_3 จุด ก เป็นขั้วบวก และจุด ข เป็นขั้วลบ ถ้าหากสัญญาณที่รับเข้ามาแรง การแยกสัญญาณจะได้ความต่างศักย์ระหว่างจุด ก กับ ข นี้มาก และเมื่อสัญญาณเข้ามาอ่อน ความต่างศักย์จุด ก กับ ข ก็จะน้อย เมื่อวัดเทียบกราวด์จุด ก จะมีศักย์เป็นบวก และจุด ข จะมีศักย์เป็นลบ แรงดันทั้งสองนี้คือแรงดันคร่อม R_4 และ R_5 จากรูปจะเห็นว่ามี R_3 (470k) ต่อกับจุด ก เพื่อเอาแรงดันไฟบวกเป็นแรงดันเอจีซีไปป้อนเข้าขาเกทของ Q_1 ผ่านทาง R_2(100k) และ L_1 ส่วน C_1 (0.01) และ C2 (0.05) เป็นตัวเก็บประจุบายพาสความถี่ไอเอฟ เมื่อแรงดันจุด ก เปลี่ยนแปลงตามความแรงของสัญญาณเอฟเอ็มที่รับจากสถานี จึงทำให้ควบคุมการขยายของ Q_1 ได้อย่างอัตโนมัติโดยใช้สัญญาณจากภาคเรโชดีเทกเตอร์ และถ้าต้องการแรงดันเอจีซีเป็นลบสามารถต่อแรงดันออกไปจากจุด ข ได้

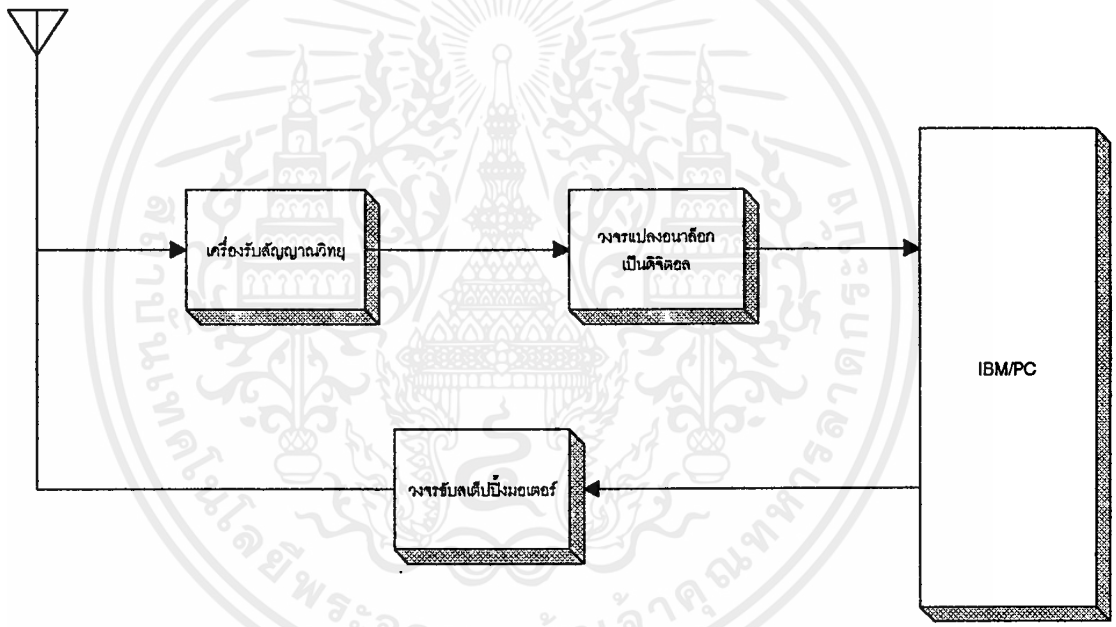


รูปที่ 2.28 วงจรอาร์เอฟ เอจีซี ได้สัญญาณควบคุมจากภาคเรโชดีเทกเตอร์

บทที่ 3
การสร้าง

3.1 หลักการทำงานของเครื่องวัดความเข้มของคลื่นวิทยุ

จากรูปที่ 3.1 เป็นบล็อกไดอะแกรมของเครื่องวัดความเข้มของคลื่นวิทยุ ซึ่งประกอบด้วย สายอากาศซึ่งจะถูกหมุนด้วยสเต็ปปิ้งมอเตอร์ โดยสายอากาศจะทำการรับสัญญาณวิทยุความถี่ 88 - 108 MHz ผ่านเครื่องรับวิทยุ จากนั้นนำระดับสัญญาณเอจีซี ของเครื่องรับที่เปลี่ยนแปลงตามความแรงของสัญญาณที่รับได้มาเข้าวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล รหัสสัญญาณที่ได้จะนำไปแปลงเป็นระดับโวลท์เตจหรือเดซิเบล แล้วแสดงผลทางหน้าจอคอมพิวเตอร์



รูปที่ 3.1 บล็อกไดอะแกรมแสดงการทำงานของเครื่องวัดความเข้มของคลื่นวิทยุ

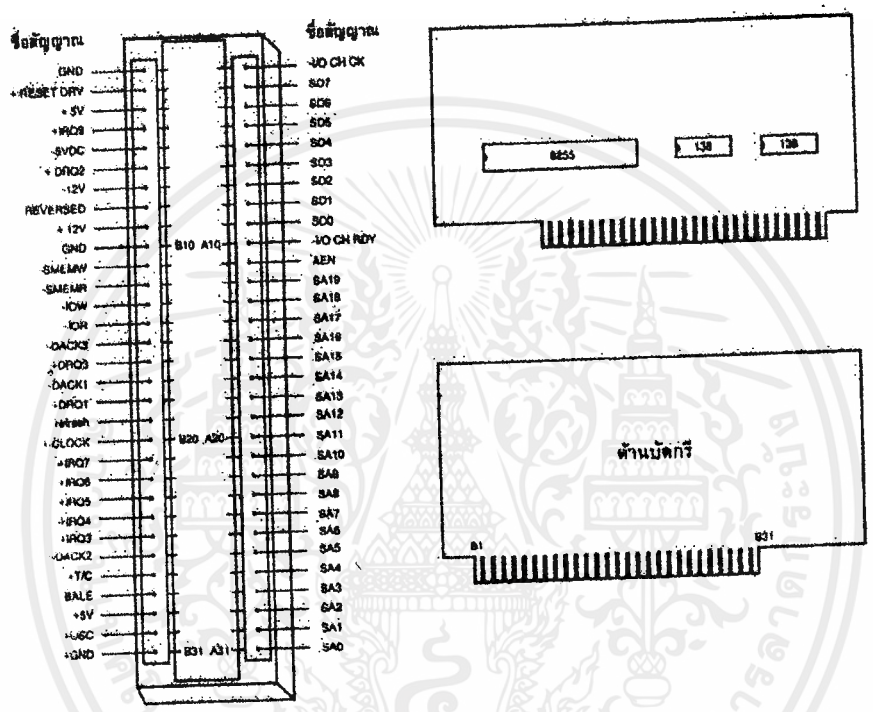
สัญญาณที่ได้จะผ่านเครื่องรับสัญญาณซึ่งจะเลือกรับความถี่ที่ต้องการซึ่งจะนำระดับสัญญาณ AGC ที่ได้จากเครื่องรับส่งไปยังวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

ในการควบคุมของทิศทางให้เป็นไปตามต้องการจะใช้สเต็ปปิ้งมอเตอร์ เป็นตัวหมุนสายอากาศ ซึ่งสเต็ปปิ้งมอเตอร์นี้จะควบคุมโดยซอฟท์แวร์อีกที

การทำงานของเครื่องวัดความเข้มของคลื่นวิทยุ ซึ่งจะประกอบไปด้วยฮาร์ดแวร์หลักสองส่วน คือ การ์ดอินเตอร์เฟซที่ทำหน้าที่เชื่อมต่อสล็อตของคอมพิวเตอร์ เข้ากับฮาร์ดแวร์ภายนอก และวงจรขับสเต็ปปิ้งมอเตอร์ทำหน้าที่ควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ให้เป็นไปตามต้องการ

3.2 การ์ดอินเตอร์เฟซ

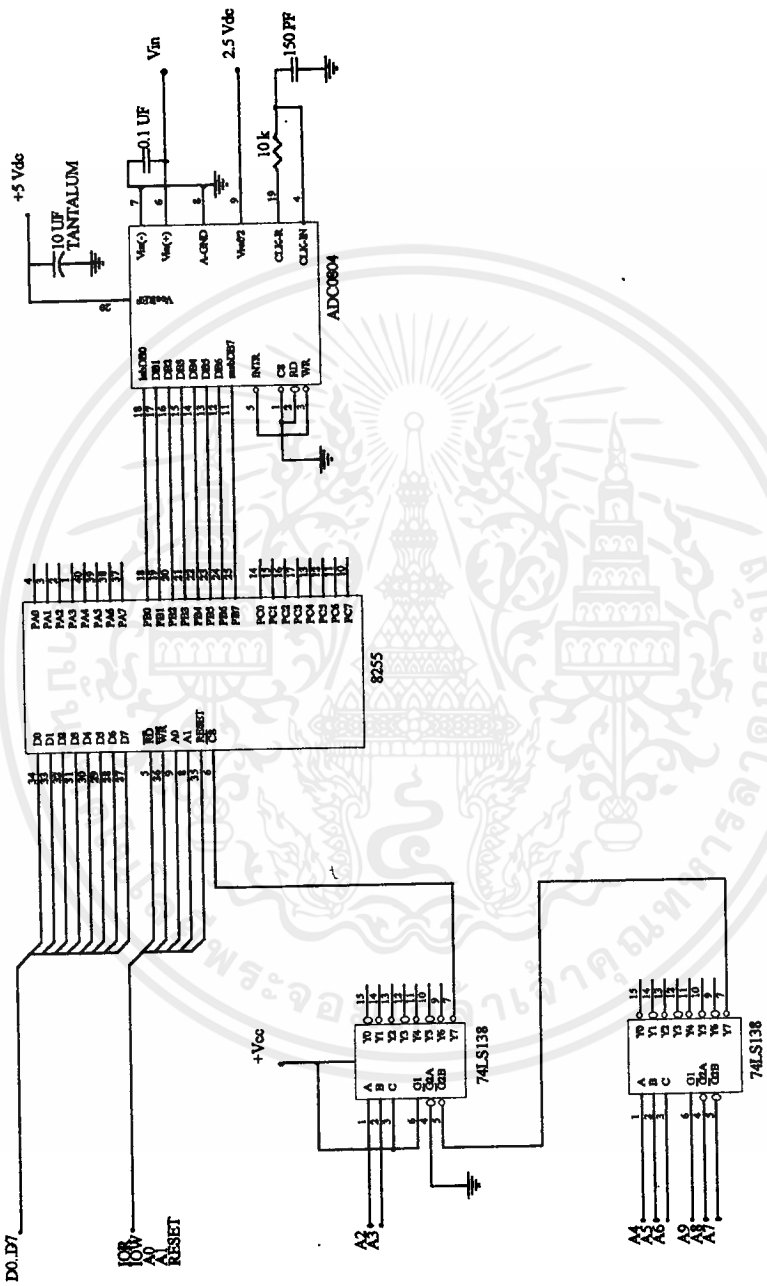
สำหรับการ์ดอินเตอร์เฟซ ซึ่งทำหน้าที่เชื่อมต่อระหว่างไมโครคอมพิวเตอร์ กับอุปกรณ์ภายนอก สามารถสร้างขึ้นมาได้ โดยใช้ชิป 8255 ทำหน้าที่เป็นพอร์ตอินพุท/เอาท์พุท ซึ่งมีระดับสัญญาณเป็นทีทีแอล เช่นกัน จะเป็นการ์ดอินเตอร์เฟซอเนกประสงค์ ซึ่งใช้งานกับไมโครคอมพิวเตอร์ PC/XT/AT ได้ทุกรุ่น ซึ่งจะต้องนำสัญญาณควบคุมบัสข้อมูล แอดเดรสบัส สัญญาณรีเซต ไฟ V_{cc} กราวนด์ของไมโครคอมพิวเตอร์ เชื่อมต่อกับการ์ดนี้ให้ถูกต้องเสียก่อน



รูปที่ 3.2 แสดงขาสัญญาณบนสล็อต และการ์ดอินเตอร์เฟซ

เครื่องไมโครคอมพิวเตอร์ทุกรุ่น (PC/AT 286, 386, 486) จะมีหมายเลขพอร์ตสำหรับใช้งานต่าง ๆ ดังตารางที่ 1 จะเห็นว่ามีบางพอร์ตที่ไม่ได้ถูกใช้งาน (สงวนไว้) เช่น 360-36F, 3C0-3CF เราสามารถที่จะนำหมายเลขพอร์ตที่ถูกกำหนดเอาไว้แล้ว แต่เครื่องไมโครคอมพิวเตอร์ของเราไม่ได้ต่ออุปกรณ์ใช้งานกับพอร์ตนั้น เช่น พอร์ตหมายเลข 278-27F (เครื่องพิมพ์ขนาน พอร์ต2) เพราะโดยทั่วไปจะนิยมต่อเครื่องพิมพ์ที่พอร์ตหมายเลข 3BC หรือ 378

ในกรณีนี้เราจะเลือกใช้แอดเดรสของพอร์ททั้งหมด 4 พอร์ท คือ 27C, 27D, 27E และ 27F เพื่อให้เป็นหมายเลขพอร์ทของ 8255 ทั้ง 4 พอร์ทเช่นกันคือ พอร์ท A, พอร์ท B, พอร์ท C, และพอร์ทควบคุม ตามลำดับ จากวงจรรูปที่ 2 จะเห็นว่าใช้ 74LS138 ในการถอดรหัสหมายเลข 27X และเมื่อรวม A0-A19 จะได้หมายเลข 27C-27F ดังตารางที่ 3.1



รูปที่ 3.3 แสดงวงจรของการ์ดอินเตอร์เฟซ

เมื่อเรากำหนดหมายเลขพอร์ทให้กับพอร์ททั้ง 4 ของ 8255 ได้แล้ว เราก็จะสามารถควบคุมพอร์ทของ 8255 ให้เป็นอินพุท หรือ เอาท์พุทได้ตามต้องการ ซึ่งในที่นี้จะกำหนดให้ห้พอร์ท B เป็นอินพุทพอร์ท เพื่อรับสัญญาณข้อมูลดิจิตอล 8 บิท มาแสดงผลทางหน้าจอคอมพิวเตอร์

ตารางที่ 3.1 แสดงการถอดรหัสแอดเดรสพอร์ต

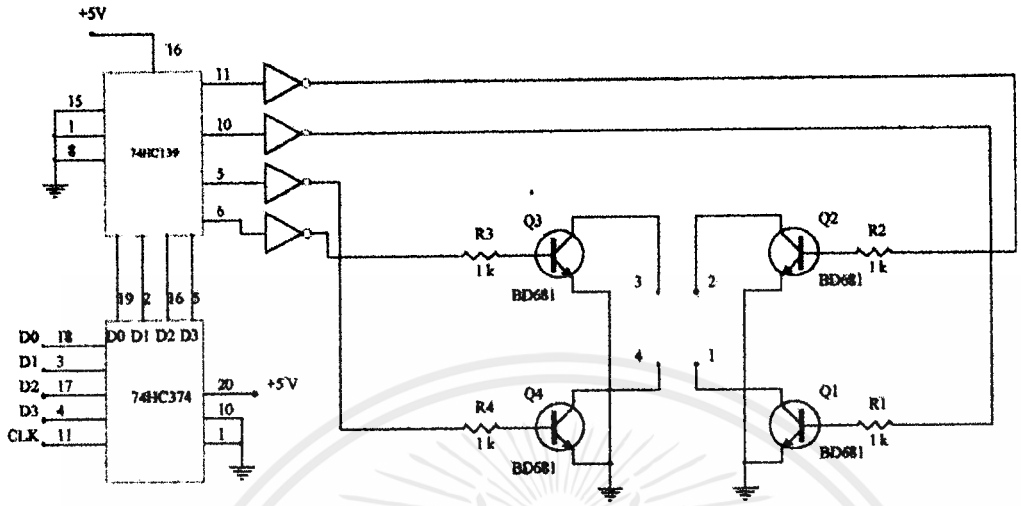
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	หมายเลขพอร์ต	หมายเหตุ
G1	$G2_A$	$G2_B$	C_2	B_2	A_2	B_1	A_1				
1	0	0	1	1	1	1	1	0	0	27C	พอร์ต A
1	0	0	1	1	1	1	1	0	1	27D	พอร์ต B
1	0	0	1	1	1	1	1	1	0	27E	พอร์ต C
1	0	0	1	1	1	1	1	1	1	27F	พอร์ตควบคุม

จะสังเกตได้ว่าการ์ดอินเตอร์เฟซที่สร้างขึ้นนี้ ไม่ได้เป็นเพียงการ์ดที่ทำหน้าที่เป็นพอร์ทของเครื่องไมโครคอมพิวเตอร์อย่างเดียว แต่จะรวมส่วนที่การแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลเข้าไปด้วยเนื่องจากในกรณีของการส่งผ่านข้อมูลผ่านทางสล็อตของคอมพิวเตอร์นั้น ถ้าหากเราแยกส่วนที่ทำหน้าที่แปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลให้อยู่ห่างจากสล็อตของคอมพิวเตอร์มากเกินไป จะทำให้ข้อมูลดิจิตอลเกิดการผิดพลาดได้

ในส่วนของการแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล จะใช้ไอซีเบอร์ ADC0804 ทำหน้าที่แปลงสัญญาณอินพุท ซึ่งได้จากระดับแรงดัน AGC ของเครื่องรับมาแปลงเป็นสัญญาณดิจิตอลขนาด 8 บิท ซึ่งเราจะต่อให้เป็นการทำงานในโหมดฟรีรันนิ่ง (Free Running Mode) การทำงานในโหมดนี้จะสามารถแปลงสัญญาณอนาล็อกอินพุทให้เป็นสัญญาณดิจิตอลได้อย่างต่อเนื่อง โดยไม่ต้องใช้สัญญาณควบคุมใด ๆ และสัญญาณดิจิตอลที่แปลงมาได้แล้วนั้น จะคงสถานะอยู่จนกว่าจะมีสัญญาณอนาล็อกอินพุทค่าใหม่เข้ามา

3.3 วงจรขับสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ที่ใช้เป็นแบบไฮบริดซึ่งมีข้อดีคือ มีแรงยึดหน่วงสูง, มีแรงบิดดึงและผลักได้ดี ซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง วงจรสมบูรณ์ของบอร์ดควบคุมสเต็ปเปอร์มอเตอร์แสดงดังรูปที่ 3.4 เริ่มต้นจาก ไอซี 74HC374 เป็นตัวเชื่อมต่อระหว่างพอร์ตเครื่องพิมพ์และบอร์ดควบคุม โดยไอซี 74HC374 เป็นไอซีที่ทำหน้าที่แลตช์ข้อมูลซึ่งภายในเป็นฟลิปฟลอปแบบ D ฟลิปฟลอป จำนวน 8 ตัวอยู่ภายใน โดยใช้สัญญาณควบคุมแลตช์อีนาเบิลหรือสัญญาณนาฬิการ่วมกันที่ขา 11 ข้อมูลทั้ง 4 บิตจะถูกโหลดเข้าเมื่อขาสัญญาณแลตช์อีนาเบิลมีสถานะเป็นลอจิกไฮ และปรากฏทางขาเอาท์พุททั้ง 4 ขาคือขา 19, 2, 16 และ 5



รูปที่ 3.4 แสดงวงจรขับสเต็ปปิ้งมอเตอร์

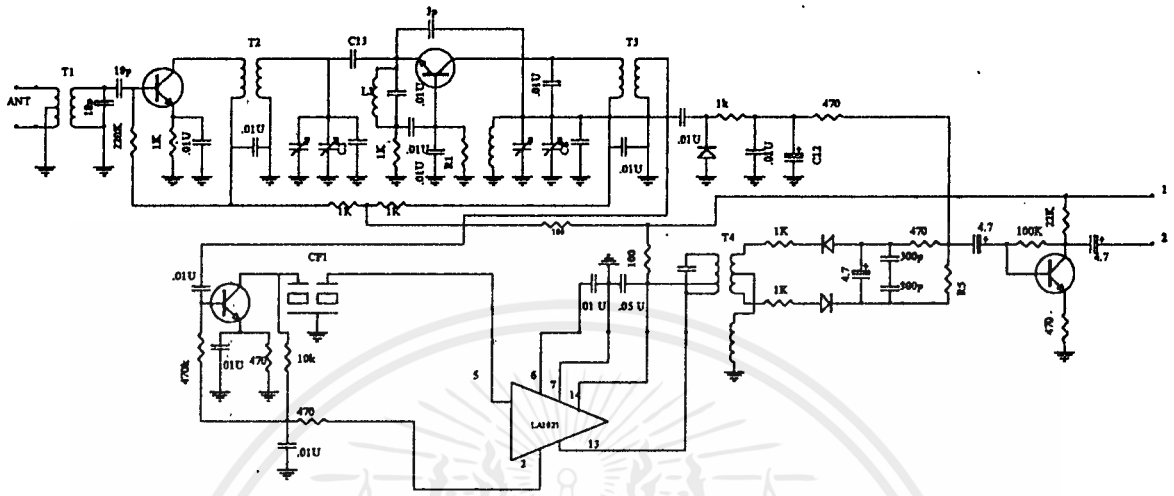
ข้อมูลที่จะส่งไปยังบอร์ดควบคุมทำได้โดยให้คอมพิวเตอร์ส่งข้อมูลขนาด 1 ไบต์ (8 บิต) ออกทางพอร์ตเครื่องพิมพ์และส่งสัญญาณทำให้ขา 11 ของ 7HC374 เป็นลอจิก "1" ซึ่งข้อมูลนี้จะปรากฏที่เอาท์พุทของ 74HC374 ทันที สัญญาณเอาท์พุทที่ได้ป้อนให้กับไอซี 74LS139 ซึ่งเป็นไอซีทำหน้าที่ดีโค้ดเดอร์สายสัญญาณจาก 2 อินพุตออกเป็น 4 เอาท์พุท 2 วงจรในตัวเดียว

ที่ขา 5 และ 6 ของ 74LS139 ถูกนำมาใช้งานซึ่งเป็นเอาท์พุทของดีโค้ดเดอร์ตัวแรกโดยที่อีก 2 ขาเอาท์พุทไม่นำมาใช้งาน และที่ขา 10 และ 11 ก็ถูกนำมาใช้งานเช่นกันสำหรับดีโค้ดเดอร์ตัวที่สองอินเวอร์เตอร์ ทำหน้าที่กลับสถานะลอจิกจากขาเอาท์พุทของดีโค้ดเดอร์เป็นตรงกันข้าม จากแอกตีฟลอจิก "0" เป็นแอกตีฟลอจิก "1" สำหรับป้อนให้วงจรขับ

เนื่องจากสเต็ปปิ้งมอเตอร์ที่ใช้เป็นแบบยูนิโพลาร์ ทรานซิสเตอร์จึงต่อเป็นวงจรสวิตช์แบบธรรมดาดังรูปซึ่งง่ายกว่าการต่อทรานซิสเตอร์แบบไฮบริดที่ใช้กับสเต็ปปิ้งมอเตอร์แบบไบโพลาร์

3.4 วงจรเครื่องรับเอฟเอ็ม

วงจรเครื่องรับที่ใช้จะเป็นวงจรที่มีฟรอนต์เอ็นด์เป็นแบบทรานซิสเตอร์ 2 ตัวแสดงดังรูปที่ 3.5

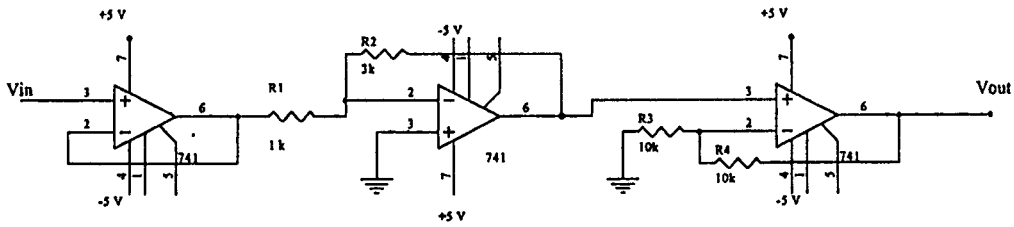


รูปที่ 3.5 แสดงวงจรเครื่องรับเอฟเอ็มที่ใช้กับระบบวัดความเข้มของคลื่นวิทยุ

จากวงจรนี้เราจะนำจุดที่เป็นระดับแรงดันเอจีซีมาใช้ ซึ่งวงจรเอจีซีนี้จะมีระดับแรงดันเปลี่ยนแปลงตามความแรงของสัญญาณที่รับเข้ามาได้ทางสายอากาศ มีหลักการทำงานดังนี้คือ เมื่อรับสัญญาณจากสถานีที่มีความแรงมาก สัญญาณที่ผ่านไปถึงภาคขยายไอเอฟก็จะสูง ทำให้การแยกสัญญาณเอจีซีได้แรงดันไฟลบสูง แรงดันลบนี้ก็จะเข้ามาหักล้างกับไฟบวกที่ไบแอสขาของทรานซิสเตอร์ในภาคขยายอาร์เอฟทำให้ไบแอสต่ำลง ทรานซิสเตอร์ก็จะลดการขยายลงได้สัญญาณเอาท์พุทพอดี แลกรณีรับสัญญาณสถานีที่มีขนาดสัญญาณต่ำ สัญญาณที่ผ่านไปยังภาคขยายไอเอฟก็ต่ำ การแยกสัญญาณเอจีซีจะได้ระดับแรงดันเอจีซีเป็นลบต่ำด้วย การหักล้างที่ขาของทรานซิสเตอร์อาจเกิดเพียงเล็กน้อย ทรานซิสเตอร์ก็ได้รับไบแอสบวกเพิ่มขึ้น การขยายสัญญาณอาร์เอฟก็จะแรงขึ้น ได้สัญญาณเอาท์พุทมีความแรงพอดีเท่าเทียมกันทุก ๆ สถานี

3.5 วงจรขยาย และกลับขั้วแรงดันเอจีซี

เนื่องจากแรงดันเอจีซีที่เรานำมาใช้เป็นแรงดันไฟลบที่มีขนาดแรงดันต่ำมาก ดังนั้นเราจึงนำมาผ่านวงจรขยาย และกลับขั้วของวงจรเพื่อให้ได้แรงดันไฟบวกที่เปลี่ยนแปลงไปตามความแรงของสัญญาณ สำหรับวงจรที่ใช้ขยาย และกลับขั้วแรงดันเอจีซีแสดงดังรูป 3.6



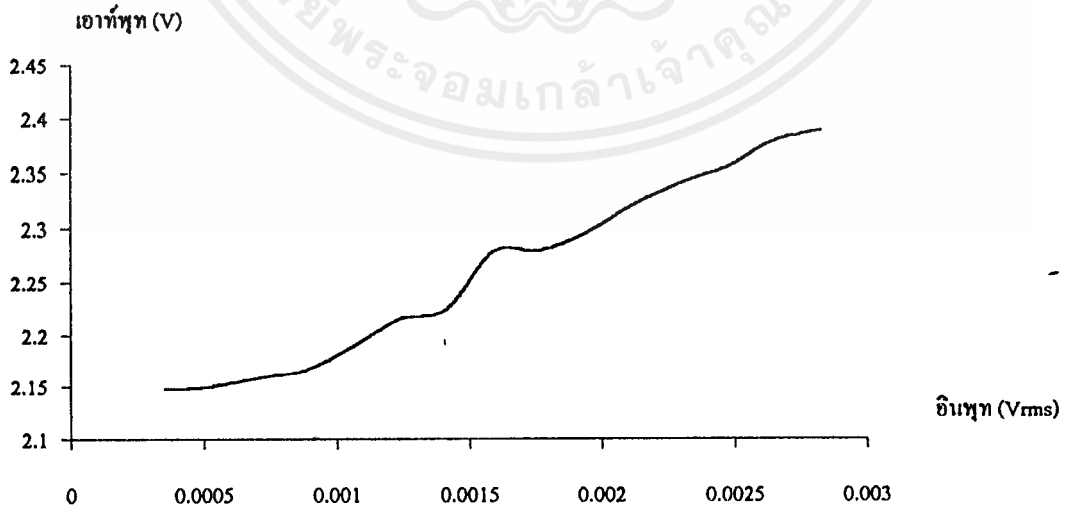
รูปที่ 3.6 แสดงวงจรขยาย และกลับขั้วแรงดันเอจีซี

ในวงจรนี้จะประกอบด้วยไอซีออปแอมป์เบอร์ 741 สามตัว ออปแอมป์ตัวแรกทำหน้าที่เป็นบัฟเฟอร์ ออปแอมป์ตัวที่สองจะทำหน้าที่กลับขั้วของแรงดันไฟลบให้เป็นไฟบวก ออปแอมป์ตัวที่สามทำหน้าที่ขยายแรงดันที่กลับขั้วแล้วให้มีระดับสูงขึ้น

อัตราขยายของออปแอมป์ตัวที่สองมีค่าเท่ากับ $-R_2/R_1 = -3k/1k = -3$

อัตราขยายของออปแอมป์ตัวที่สามมีค่าเท่ากับ $(R_4/R_3)+1 = (10k/10k)+1 = 2$

3.6 การคำนวณหาระดับความแรงของสัญญาณ

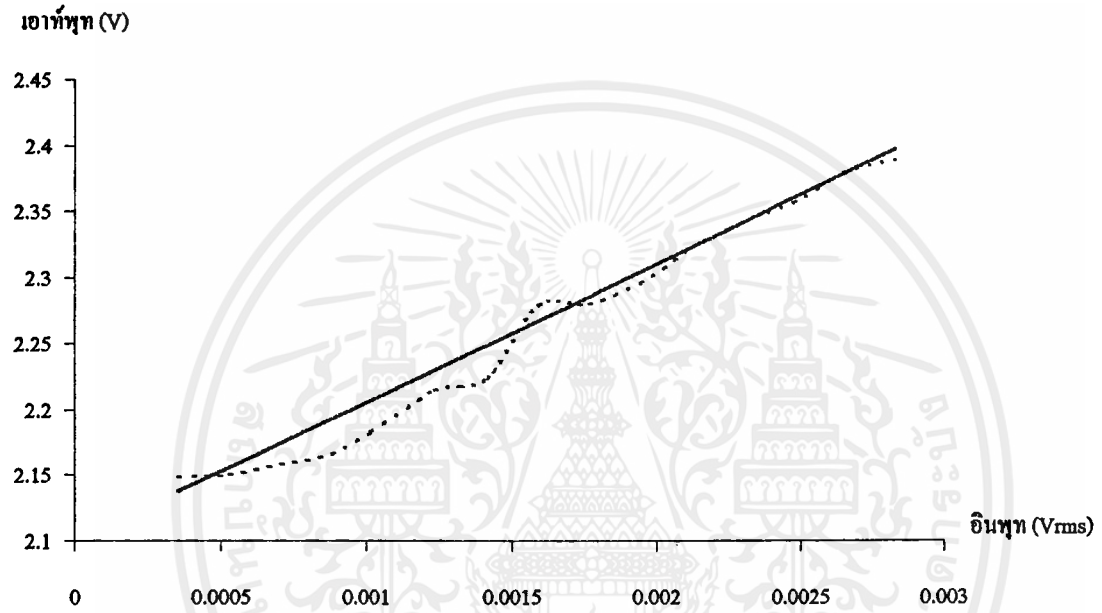


รูปที่ 3.7 รูปกราฟเปรียบเทียบระหว่างอินพุท และเอาท์พุท (แรงดันเอจีซี) ของเครื่องรับเอฟเอ็ม

ในการคำนวณหาค่าความแรงของสัญญาณที่รับเข้ามาได้นั้น เราจะใช้วิธีการคำนวณจากอัตราส่วนของความสัมพันธ์ของอินพุทซึ่งเป็นระดับสัญญาณที่รับเข้ามาได้ และเอาท์พุทซึ่งเป็นระดับแรงดันเอจีซี

ความสัมพันธ์ของอินพุท และเอาท์พุทข้างต้นหาได้จากการทดลองป้อนสัญญาณอินพุท ให้มีระดับสัญญาณค่าต่าง ๆ จากนั้นทำการวัดเอาท์พุทเอจีซี ซึ่งจากการทดลองเราจะได้รูปกราฟแสดงความสัมพันธ์ของอินพุท และเอาท์พุทดังกราฟรูปที่ 3.7

จากกราฟเราจะเห็นได้ว่าเมื่อระดับอินพุทเพิ่มขึ้น ระดับเอาท์พุทเอจีซีก็จะเพิ่มขึ้นตามไปด้วย ดังนั้นเพื่อหาความสัมพันธ์หรืออัตราส่วนของอินพุทกับเอาท์พุท เราจะประมาณกราฟเป็นเส้นตรงดังนี้



รูปที่ 3.8 แสดงรูปกราฟที่ประมาณเป็นเส้นตรงเทียบกับกราฟจริง

จากกราฟเส้นตรงที่ประมาณ จะได้ความสัมพันธ์ระหว่างอินพุทกับเอาท์พุทคือ

$$\text{เอาท์พุท} = (105 \times \text{อินพุท}) + 2.1$$

ดังนั้น เมื่อเราทราบค่าแรงดันเอจีซี (เอาท์พุท) เราจะสามารถหาค่าระดับความแรงของสัญญาณที่รับเข้ามาได้ดังนี้

$$\text{ระดับความแรงของสัญญาณที่รับเข้ามา} = (\text{ระดับแรงดันเอจีซี} - 2.1)/105$$

ในการแสดงความแรงของสัญญาณที่รับเข้ามาได้ที่หน้าจอคอมพิวเตอร์ เราจะแสดงในหน่วย dBmV โดย

$$\text{ระดับความแรงของสัญญาณที่รับเข้ามา} = 20 \log [(\text{ระดับแรงดันเอจีซี} - 2.1) / (105 \times 10^{-3})]$$

ซึ่งเราจะนำความสัมพันธ์นี้ไปใช้ในการเขียนโปรแกรมแสดงผล

บทที่ 4

การทดลอง และผลการทดลอง

4.1 การทดลองที่ 1

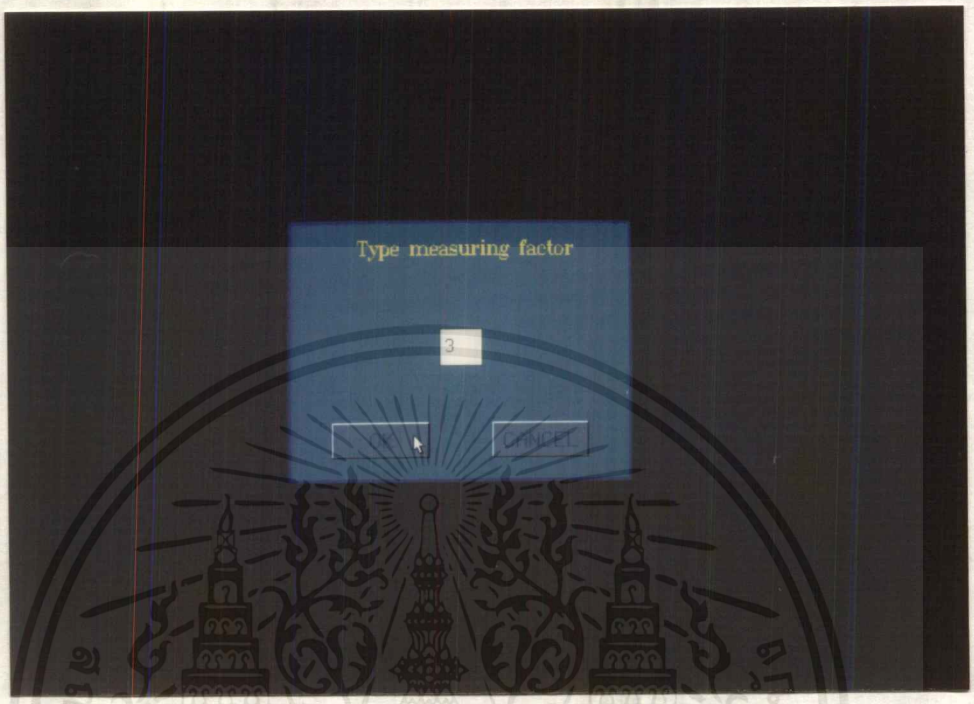
จุดประสงค์ - เพื่อทดสอบการทำงานของสเต็ปปิ้งมอเตอร์ และสายอากาศ
- เพื่อหาจำนวนสเต็ปต่อรอบของมอเตอร์

อุปกรณ์
1) ชุดสเต็ปปิ้งมอเตอร์พร้อมเสาอากาศ
2) วงจรขับสเต็ปปิ้งมอเตอร์
3) การ์ดเชื่อมต่อกับคอมพิวเตอร์
4) โปรแกรมสั่งงาน
5) คอมพิวเตอร์
6) แหล่งจ่ายไฟ 12 โวลท์

วิธีการทดลอง
1) ต่อวงจรขับสเต็ปปิ้งมอเตอร์เข้ากับพอร์ตปริ้นเตอร์ของคอมพิวเตอร์และต่อมอเตอร์เข้ากับวงจรขับ
2) เสียบการ์ดเชื่อมต่อกับคอมพิวเตอร์
3) ต่อแหล่งจ่ายไฟ 12 โวลท์ให้กับวงจรขับสเต็ปปิ้งมอเตอร์
4) ใช้โปรแกรมภาษาซีสั่งงานให้ป้อนรหัสควบคุมให้กับวงจรขับ โดยจะสามารถกำหนดจำนวนสเต็ปและทิศทาง (ตามเข็มหรือทวนเข็ม) ได้ การสั่งงานจะสั่งให้มอเตอร์หมุนดังนี้
   - ให้มอเตอร์หมุนแบบตามเข็มนาฬิกา จำนวน 5 สเต็ป
   - ให้มอเตอร์หมุนแบบตามเข็มนาฬิกา จำนวน 10 สเต็ป
   - ให้มอเตอร์หมุนแบบทวนเข็มนาฬิกา จำนวน 5 สเต็ป
   - ให้มอเตอร์หมุนแบบทวนเข็มนาฬิกา จำนวน 10 สเต็ป
4) สังเกตผลที่ได้จากหน้าจอแสดงผลของคอมพิวเตอร์
5) จากนั้นทำการทดลองหมุนมอเตอร์ 1 รอบ แล้วนับจำนวนสเต็ปที่ได้

รูปที่ 4.1 แสดงหน้าจอสั่งงานสเต็ปปิ้งมอเตอร์ (หน้าจอหลัก)

รูปที่ 4.2 แสดงหน้าจอสั่งงานสต็ปปิ้งมอเตอร์ (หน้าจอติดตั้ง)



รูปที่ 4.3 แสดงหน้าจอการตั้งค่าแฟกเตอร์ (ค่าที่รับในแต่ละสเต็ปซึ่งจะนำมาเฉลี่ยแล้วบันทึกลงคอมพิวเตอร์)

ผลการทดลอง

จากการทดสอบการหมุนของสเต็ปปิ้งมอเตอร์แบบตามเข็มนาฬิกาจำนวน 5 สเต็ป มอเตอร์หมุนเป็นจำนวน 5 สเต็ป ตามที่กำหนด และเสาอากาศหมุนเป็นจำนวน 5 สเต็ป

จากการทดสอบการหมุนของสเต็ปปิ้งมอเตอร์แบบตามเข็มนาฬิกาจำนวน 10 สเต็ป มอเตอร์หมุนเป็นจำนวน 10 สเต็ป แต่เสาอากาศหมุนได้ประมาณ 7-8 สเต็ป

จากการทดสอบการหมุนของสเต็ปปิ้งมอเตอร์แบบทวนเข็มนาฬิกาจำนวน 5 สเต็ป มอเตอร์หมุนเป็นจำนวน 5 สเต็ป ตามที่กำหนด และเสาอากาศหมุนเป็นจำนวน 5 สเต็ป

จากการทดสอบการหมุนของสเต็ปปิ้งมอเตอร์แบบทวนเข็มนาฬิกาจำนวน 10 สเต็ป มอเตอร์หมุนเป็นจำนวน 10 สเต็ป และเสาอากาศหมุนเป็นจำนวน 10 สเต็ป

เมื่อสเต็ปปิ้งมอเตอร์หมุนครบรอบ (360 องศา) นับจำนวนสเต็ปได้ 1175 สเต็ป ซึ่งเมื่อคิดเป็นองศาที่หมุนได้ต่อ 1 สเต็ป จาก

$$360 / 1175 = 0.3064$$

เพราะฉะนั้นจะได้เท่ากับ 0.3064 องศาต่อสเต็ป

และจากการสังเกตการหมุนของเสาอากาศพบว่า หากสั่งให้มอเตอร์หมุนในทิศทางตามเข็มนาฬิกา จะมีช่วงหนึ่งที่เสาอากาศหยุดหมุน แต่หากสั่งให้มอเตอร์หมุนในทิศทางทวนเข็มนาฬิกา มอเตอร์จะหมุนได้ครบรอบ

สรุปผลการทดลอง

จากการทดลองที่ 1 มอเตอร์สามารถทำงานได้ตามที่กำหนด แต่เสาอากาศจะมีการสะดุดบ้าง เนื่องจากฟันเฟืองของเสาอากาศกับฟันเฟืองของมอเตอร์ไม่ลงล็อกกันพอดี คือระยะห่างของฟันเฟืองของเสาอากาศกว้างกว่าระยะห่างของฟันเฟืองของมอเตอร์ ทำให้การหมุนของเสาอากาศมีการติดขัด

จากลักษณะอาการดังกล่าว ได้ทำการแก้ไขโดยทำให้เฟืองของเสาอากาศอยู่ชิดกับเฟืองของมอเตอร์มากขึ้น และหลังจากแก้ไขแล้วได้ทำการทดลองอีกครั้ง จึงได้ผลการทดลองที่ถูกต้องตามที่ต้องการ

4.2 การทดลองที่ 2

จุดประสงค์ เพื่อทดสอบการทำงานของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล

อุปกรณ์

1) วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล (อยู่ในการ์ดเชื่อมต่อกับคอมพิวเตอร์)
2) คอมพิวเตอร์
3) โปรแกรมสั่งงาน
4) แหล่งจ่ายไฟ 0-5 โวลท์

วิธีการทดลอง

1) ต่อการ์ดเชื่อมต่อเข้ากับคอมพิวเตอร์ (ซึ่งมีวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลอยู่ในการ์ด
2) ต่อแหล่งจ่ายไฟเข้ากับการ์ด
3) ใช้โปรแกรมภาษาซีควบคุมการรับข้อมูลของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล
4) ป้อนระดับโวลท์เตจจากแหล่งจ่ายไฟ จากนั้นกด Enter ให้แสดงผลการแปลงจากสัญญาณอนาล็อกเป็นรหัสไบนารีและระดับเลขฐานสิบ 0-255

ตารางที่ 4.2 แสดงผลการเปรียบเทียบแรงดัน AGC ที่ได้จากเครื่องวัดความแรงของคลื่นและเครื่องรับวิทยุ FM

	เครื่องวัดความเข้มของคลื่น		เครื่องรับวิทยุ
องศา	dBμV	แรงดัน AGC (mV)	แรงดัน AGC (mV)
0	54	675	144
20	58	716	148
40	59	720	155
60	55	680	171
80	50	630	179
100	48	600	180
120	54	670	184
140	57	705	193
160	56	700	189
180	55	690	160
200	53	650	158
220	47	600	148
240	38	500	157
260	46	590	162
280	53	660	164
300	54	670	163
320	55	690	165
340	53	670	167
360	54	680	168

สรุปผลการทดลอง

จากการทดลองในขั้นตอนแรกจะเห็นว่าเครื่องรับ FM สามารถให้ผลการวัด (แรงดัน AGC) ตรงกับเครื่องวัดคือมีค่ามากในทิศทางเดียวกัน และมีค่าน้อยในทิศทางเดียวกัน

จากตารางที่ 4.2 ในส่วนของเครื่องวัดความแรงของคลื่นจะเห็นว่าค่าแรงดัน AGC จะมีค่ามากขึ้นหรือลดลงในทำนองเดียวกับ dBμV ที่เป็นดังนี้เพราะแรงดัน AGC ที่ได้จากเครื่องวัดจะเป็นตัวบอกความแรงของเครื่องวัด แต่ผลที่ได้จากเครื่องรับ FM กลับมีค่าที่ไม่สัมพันธ์กับค่าที่อ่านได้จากเครื่องวัด

บอกความแรงของเครื่องวัด แต่ผลที่ได้จากเครื่องรับ FM กลับมีค่าที่ไม่สัมพันธ์กับค่าที่อ่านได้จากเครื่องวัดทั้งๆที่ทำการทดสอบเหมือนกันที่ความถี่เดียวกัน ทั้งนี้อาจเป็นเพราะการวัดครั้งที่ 2 (เครื่องรับ FM) สายที่ต่อจากสายอากาศเกิดการพันกับเสามอเตอร์ซึ่งทำให้สัญญาณที่รับได้มีความเพี้ยนไป

4.5 การทดลองที่ 5

จุดประสงค์ เพื่อทดสอบความแรงของสัญญาณในทิศทางต่าง ๆ

อุปกรณ์

1) สายอากาศแบบยากิ ชุดควบคุมและหมุนสายอากาศ
2) โปรแกรมการสั่งงานการแปลงสัญญาณอนาล็อกเป็นดิจิตอล
3) เครื่องวัดระดับความเข้มสัญญาณ
4) วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล

วิธีการทดลอง

1) ทำการต่อสายอากาศเข้ากับชุดหมุนสายอากาศ
2) ต่อสายเคเบิลเข้ากับสายอากาศ
3) นำสายเคเบิลที่ต่อกับสายอากาศมาต่อเข้ากับเครื่องวัดระดับความเข้มสัญญาณ
4) ควบคุมการหมุนสายอากาศโดยให้หมุนทีละ 3 องศา
5) ใช้โปรแกรมภาษาซีควบคุมการแปลงสัญญาณอนาล็อกที่รับได้จากสายอากาศเป็นระดับเลขฐานสิบ

ผลการทดลอง

จากตารางที่ 4.3 เป็นตารางแสดงระดับสัญญาณที่ได้จากเครื่องวัดระดับความเข้มสัญญาณและระดับเลขฐานสิบที่ได้จากวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลที่ความถี่ 88.0 MHz โดยจะแสดงค่า dBμV/m และค่าที่เป็นระดับเลขฐานสิบ

ตารางที่ 4.3 ระดับสัญญาณที่ได้จากเครื่องวัดระดับความเข้มสัญญาณที่ความถี่ 88.0 MHz

องศาที่หมุนไป	dBμV	เลขฐานสิบ	องศาที่หมุนไป	dBμV	เลขฐานสิบ
0	38	8	63	38	10
3	38	8	66	37	9
6	37	9	69	38	9
9	38	11	72	37	9
12	36	9	75	38	10
15	38	9	78	38	9
18	37	9	81	38	10
21	36	8	84	37	10
24	35	8	87	37	10
27	35	9	90	36	9
30	35	8	93	40	10
33	37	7	96	39	11
36	36	8	99	39	10
39	35	8	102	39	11
42	36	8	105	39	10
45	36	7	108	40	10
48	35	9	111	39	10
51	37	8	114	38	10
54	35	9	117	38	10
57	36	9	120	39	10
60	36	9			

ตารางที่ 4.3 ระดับสัญญาณที่ได้จากเครื่องวัดระดับความเข้มสัญญาณที่ความถี่ 88.0 MHz

องศาที่หมุนไป	dBμV	เลขฐานสิบ	องศาที่หมุนไป	dBμV	เลขฐานสิบ
123	38	10	183	36	7
126	39	10	186	35	6
129	37	9	189	36	7
132	37	9	192	35	7
135	38	9	195	35	7
138	36	9	198	35	7
141	35	8	201	36	6
144	34	7	204	34	7
147	34	8	207	36	7
150	35	7	210	36	6
153	35	7	213	32	7
156	36	7	216	36	6
159	33	7	219	35	7
162	34	7	222	33	7
165	35	8	225	35	7
168	35	8	228	36	7
171	35	8	231	33	7
174	36	8	234	37	7
177	38	9	237	36	6
180	37	8	240	36	6

ตารางที่ 4.3 ระดับสัญญาณที่ได้จากเครื่องวัดระดับความเข้มสัญญาณที่ความถี่ 88.0 MHz

องศาที่หมุนไป	dBμV	เลขฐานสิบ	องศาที่หมุนไป	dBμV	เลขฐานสิบ
243	36	6	303	36	9
246	35	7	306	36	9
249	34	7	309	37	9
252	35	8	312	35	8
255	36	8	315	37	9
258	35	8	318	37	8
261	35	8	321	37	8
264	36	8	324	37	8
267	38	9	327	37	9
270	37	9	330	38	9
273	36	9	333	38	9
276	35	8	336	39	10
279	35	8	339	38	7
282	36	9	342	37	8
285	37	8	345	38	9
288	36	9	348	37	9
291	38	9	351	36	8
294	36	9	354	37	9
297	37	9	357	38	8
300	37	9	360	39	9

สรุปผลการทดลอง

การทดลองนี้จะเป็นการทดลองวัดสัญญาณที่รับได้ โดยการหมุนสายอากาศเป็นจำนวน 1 รอบ เพื่อดูว่าสายอากาศจะรับสัญญาณได้ดีเพียงใด จากตาราง 4.3 ที่ได้ จะเห็นว่าจากจุด 0 องศา ลักษณะของสัญญาณจะมีการเพิ่มขึ้นเรื่อยๆ ซึ่งจะสูงสุดที่ช่วง 90 – 120 องศา จากนั้นสัญญาณจะค่อยๆต่ำลงจนถึงจุดต่ำสุด จากนั้นจึงเพิ่มขึ้นอีกครั้ง จากการทดลองนี้ทำให้เห็นว่าสายอากาศสามารถรับคลื่นเป็นทิศทางได้ ถึงแม้ว่าทิศทางด้านหน้าจะไม่แตกต่างกับด้านหลังมากก็ตาม ซึ่งทั้งนี้อาจเป็นเพราะจำนวนเอเลเมนต์ของสายอากาศน้อยเกินไป

4.6 การทดลองที่ 6

จุดประสงค์ เพื่อวัดความแรงของสัญญาณในทิศทางต่างๆที่วัดจากระบบวัดความเข้มของคลื่นวิทยุ แล้วแสดงผลออกมาเป็นกราฟ

อุปกรณ์

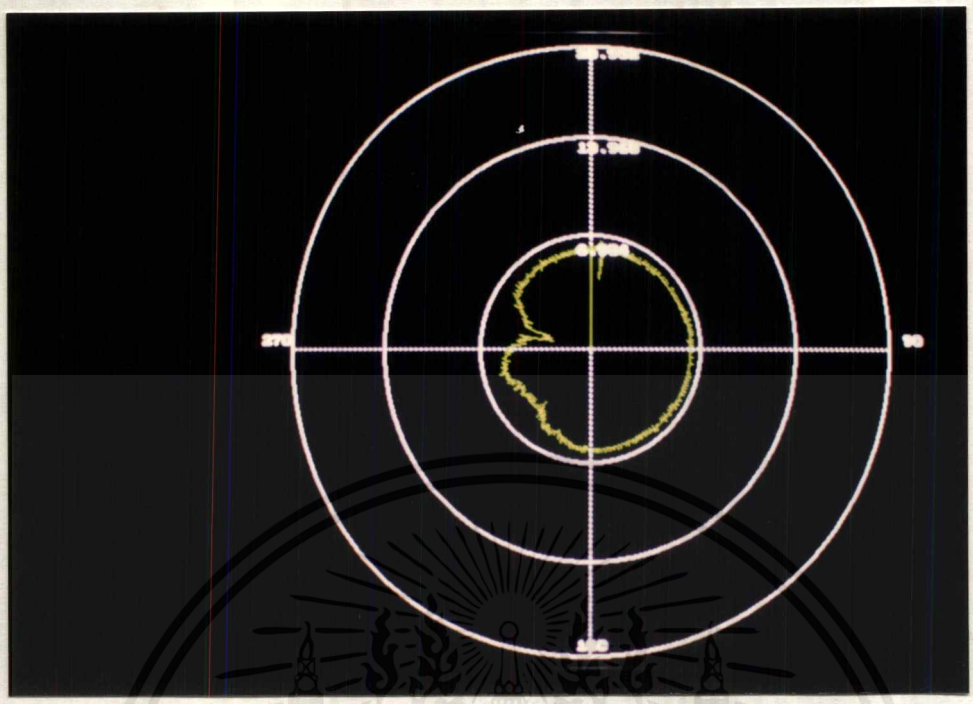
1) สายอากาศแบบยากิ
2) โปรแกรมการสั่งงานการแปลงสัญญาณอนาล็อกเป็นดิจิตอล
3) คอมพิวเตอร์
4) ชุดสเต็ปปิ้งมอเตอร์
5) เครื่องรับวิทยุ FM
6) วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล

วิธีการทดลอง

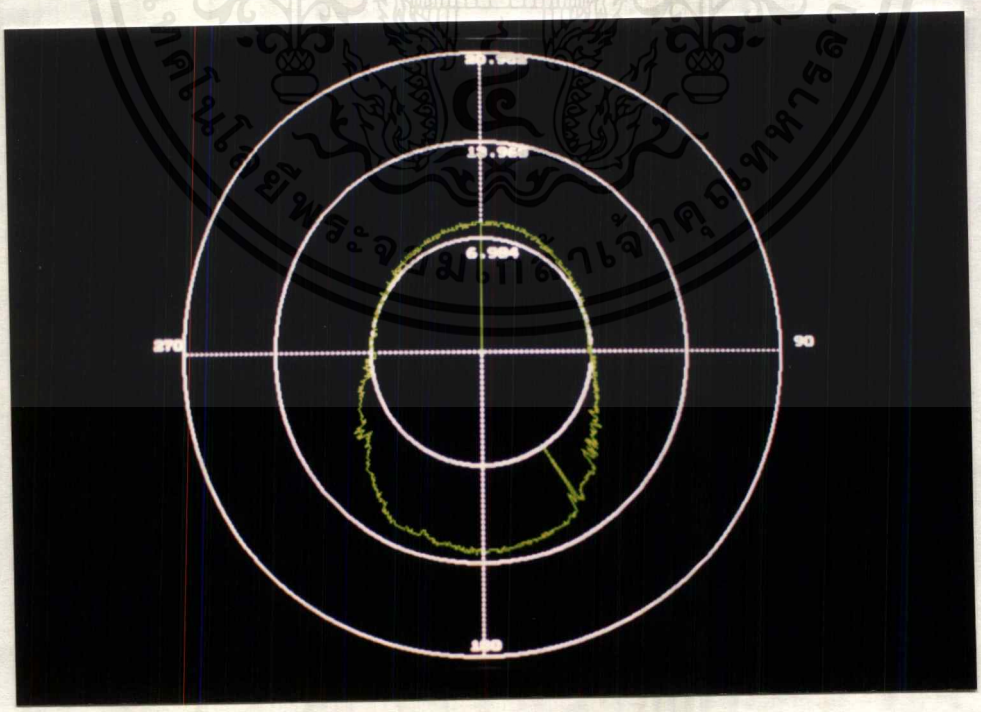
1) ทำการประกอบสายอากาศเข้ากับชุดหมุนสายอากาศ (สเต็ปปิ้งมอเตอร์)
2) ต่อสายอากาศเข้ากับเครื่องรับวิทยุ FM
3) นำสัญญาณ AGC จากเครื่องรับวิทยุ FM ไปเข้าคอมพิวเตอร์โดยผ่านทางการ์ดอินเตอร์เฟซ
4) ควบคุมการหมุนสายอากาศโดยให้หมุนและรับค่าทุกๆสเต็ป ผลที่ได้จะถูกบันทึกเข้าเครื่องคอมพิวเตอร์

ผลการทดลอง

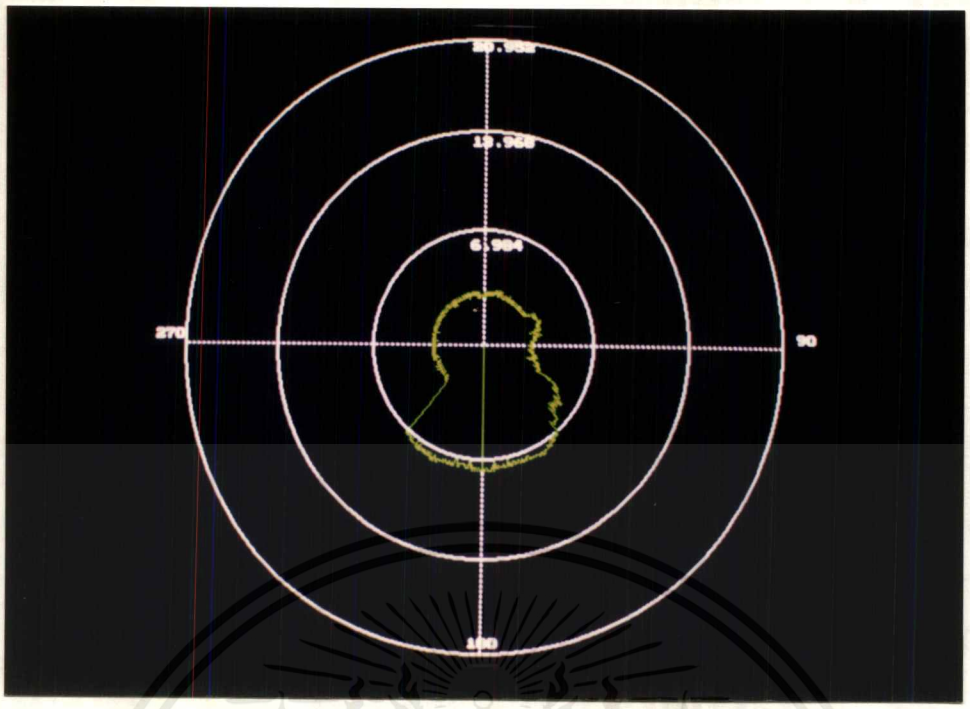
ผลการทดลองที่ได้จากการทดลองวัดความแรงของคลื่น FM 95.0 MHz เป็นดังกราฟรูปที่ 4.4 และคลื่น 93.5 MHz จะเป็นดังกราฟรูปที่ 4.5 ส่วนรูปที่ 4.6 เป็นกราฟแสดงแพทเทิร์นของสายอากาศ



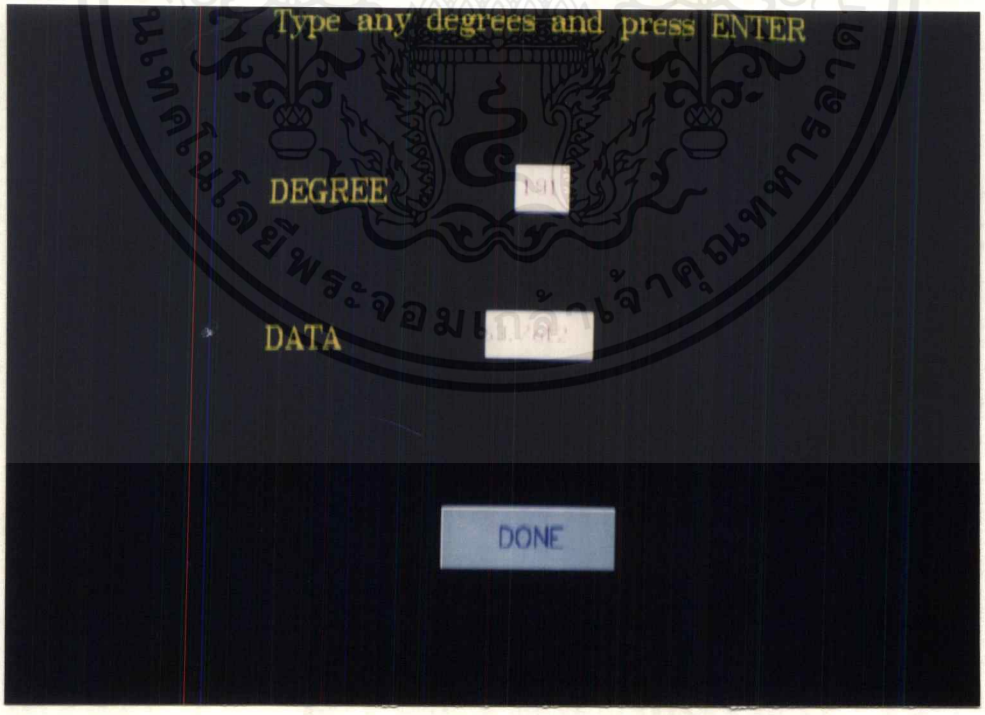
รูปที่ 4.4 กราฟแสดงผลการวัดความเข้มของคลื่นวิทยุ FM 95.0 MHz



รูปที่ 4.5 กราฟแสดงผลการวัดความเข้มของคลื่นวิทยุ FM ความถี่ 93.5 MHz จาก
ระบบวัดความเข้มของคลื่น



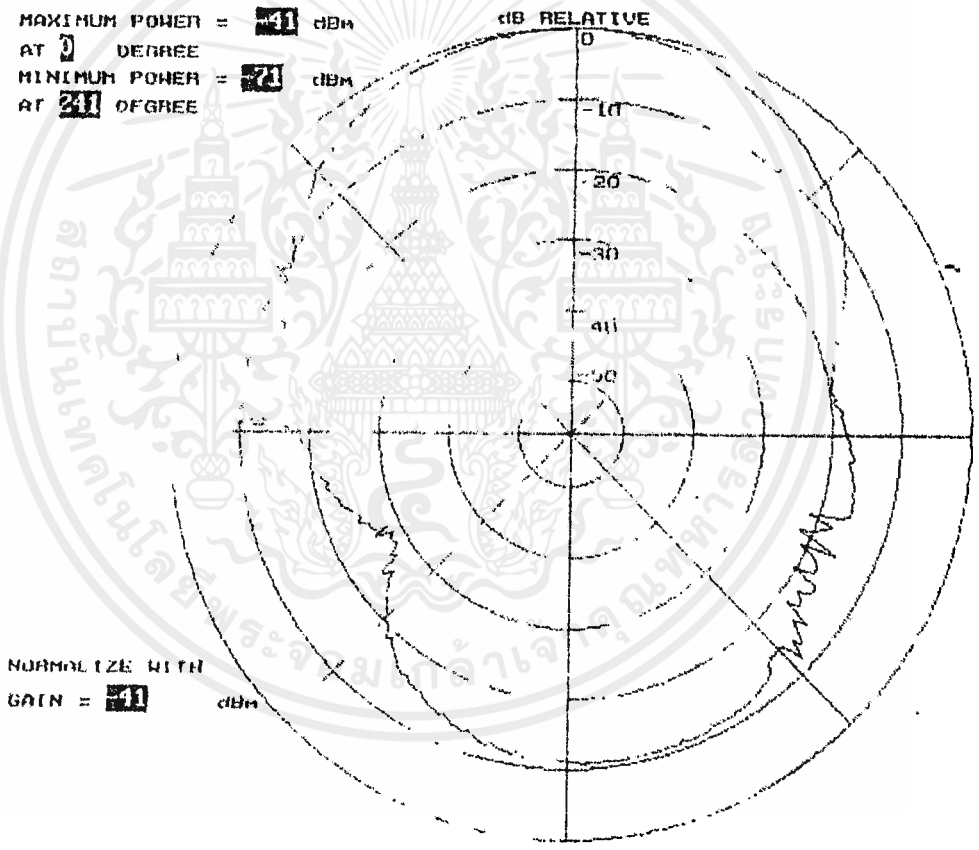
รูปที่ 4.6 กราฟแสดงผลการวัดความเข้มของคลื่นวิทยุ FM ความถี่ 93.5 MHz จากเครื่อง Field Strength Meter



รูปที่ 4.7 หน้าจอแสดงผลโดยละเอียด

สรุปผลการทดลอง

จากกราฟทั้งสองรูปจะมีความแตกต่างกัน กราฟรูปที่ 4.4 จะมีรูปแบบที่ไม่แน่นอน ไม่สามารถบอกทิศทางที่คลื่นแรงสุดได้ ทั้งนี้อาจเป็นเพราะทำการวัดในห้อง ส่วนกราฟรูปที่ 4.5 และ 4.6 ทำการวัดบนหลังคาตึก ผลที่ได้จึงมีลักษณะที่เป็นทิศทางมากกว่า แต่ทิศทางที่คลื่นแรงสุดนั้นก็ยังไม่แตกต่างจากทิศทางที่คลื่นเบาสุด ทั้งนี้อาจเนื่องมาจากตัวสายอากาศมีแพทเทิร์นไม่ดีพอ



รูปที่ 4.8 แสดงแพทเทิร์นของสายอากาศที่ใช้ในการทดลอง

บทที่ 5
บทสรุป และวิจารณ์

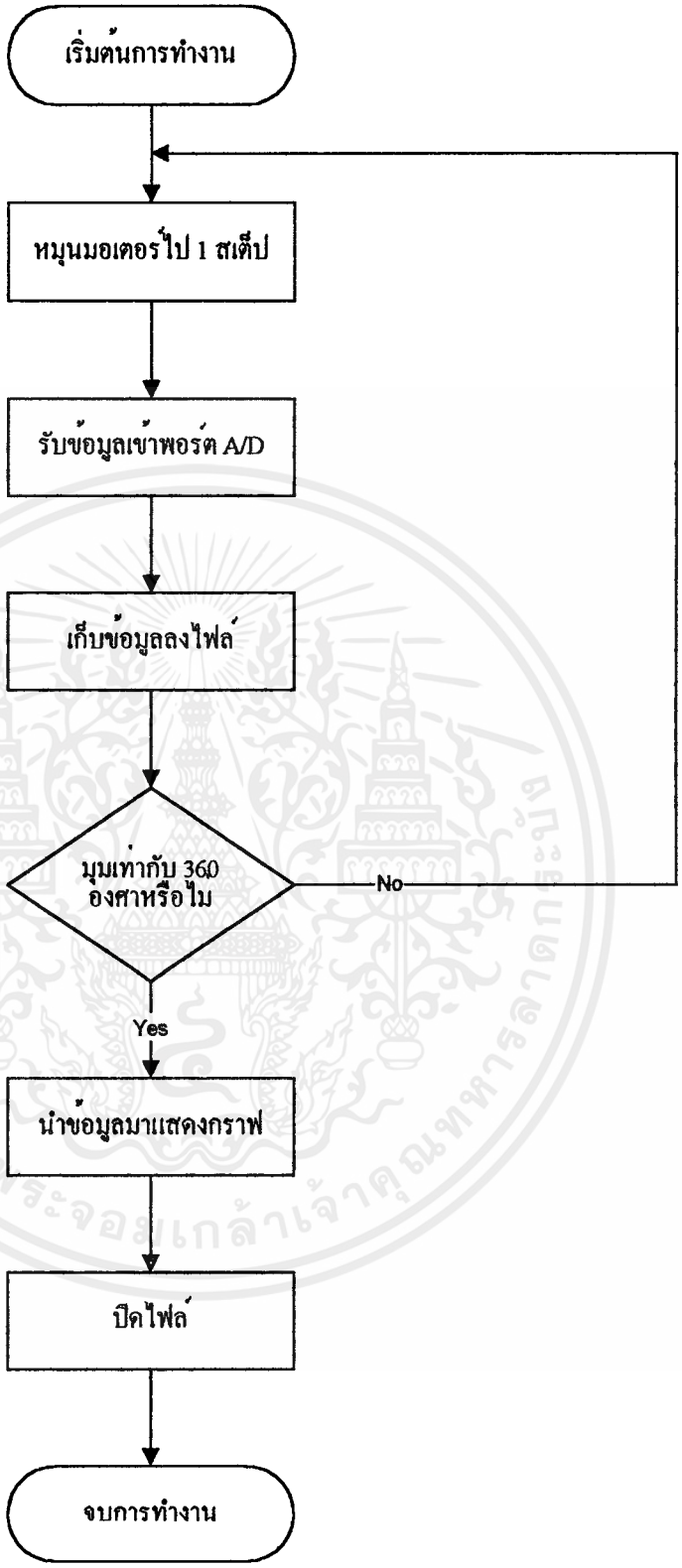
จากการทดลองที่ 1 จะเห็นว่ามอเตอร์สามารถหมุนตามสเต็ปที่ต้องการได้ แสดงว่าโปรแกรมควบคุมการทำงานด้วยภาษาซีนั้น ทำงานได้อย่างถูกต้อง และวงจรขับสเต็ปปิ้งมอเตอร์ สามารถขับกระแสให้กับมอเตอร์ได้อย่างเพียงพอ และสำหรับชุดมอเตอร์ที่ต่อกับสายอากาศ ถึงแม้จะมีปัญหาที่รอยต่อของเฟือง แต่ก็สามารถแก้ไขให้หมุนได้อย่างถูกต้อง

จากการทดลองที่ 2 วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล จะแปลงระดับโวลเตจที่ป้อนให้จาก 0 ถึง 5 โวลท์ให้เปลี่ยนเป็นเลขฐานสอง และเลขฐานสิบซึ่งอยู่ในช่วง 0 ถึง 255

จากการทดลองที่ 3, 4, 5 และ 6 เป็นการวัดความแรงของสัญญาณ จะเห็นว่าลักษณะข้อมูลที่ได้จะยังไม่มีทิศทางที่ชัดเจน เนื่องจากสายอากาศที่ใช้รับ มีแพทเทิร์นที่มีโลบทางด้านหลังสูง ทำให้ระดับสัญญาณที่วัดได้จากทางด้านหน้าและด้านหลังแตกต่างกันไม่มากดังตารางที่ 4.3 แต่ก็ยังพอเห็นว่าสัญญาณด้านหน้าจะแรงกว่าด้านหลังเล็กน้อย

สำหรับสเกลแสดงความแรงของสัญญาณเป็นเดซิเบลในโครงงานนี้นั้นยังไม่แม่นยำเท่าที่ควร แต่ก็ยังสามารถใช้บอกทิศทางที่มีสัญญาณแรงได้

ภาคผนวก ก. โปรแกรมสั่งงาน และแสดงผล



แผนผังขั้นตอนการทำงานของระบบ

```
#include <dos.h>
#include <graphics.h>
#include <bios.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int mx,my,button;
union REGS regs;
char value[5] = " ",factor[5] = " ";


int main(void)
{
   int select_button;

   void Mouse(void);
   void Set_Limits(int min_X_limit,int max_X_limit,
                   int min_Y_limit,int max_Y_limit);
   void Show_Mouse(void);
   void Hide_Mouse(void);
   void Event(void);

   void Start_Graphics(void);
   void Main_Menu(void);
   void Measuring_Menu(void);
   void Direction_Set(void);
   void Draw_Button(int x_left,int y_top,
                    int x_right,int y_bottom);
   void Click_Button(int x_left,int y_top,
                     int x_right,int y_bottom);
   int Check_Step_Data(int y_top,int y_bottom,int x_left1,
                       int x_right1,int x_left2,int x_right2);
   int Check_Button(int x_left,int x_right,int y_top1,int y_bottom1,
                    int y_top2,int y_bottom2,int y_top3,int y_bottom3);
   int Key_Event(void);
   void Show_Steps(void);
   int Check_Factor_Data(int y_top,int y_bottom,int x_left1,
                         int x_right1,int x_left2,int x_right2);
   void Measure_Set(void);
   void Field_Strength(void);
   void Arrange_Data(void);
   void Show_Graph(void);
   void More_Info(int x_left,int y_top,int x_right,int y_bottom);
   void Field_Info(void);

   Start_Graphics();
   First:Main_Menu();
       select_button = Check_Button(200,430,170,213,270,313,370,413);
       if (select_button == 1)
       {
          Second:Measuring_Menu();
               select_button =
Check_Button(200,430,170,213,270,313,370,413);
               if (select_button == 1)
               {
                  Measure_Set();
                  select_button =
Check_Factor_Data(300,330,200,290,350,440);
                  if (select_button == 1)
                  {
```

```
                    goto Second;
                   }
                   if (select_button == 2)
                   {
                    goto Second;
                   }
                }
                if (select_button == 2)
                {
                   Field_Strength();
                   Show_Graph();
                   Arrange_Data();
                   More_Info(250,325,380,368);
                   Field_Info();
                   goto First;
                }
                if (select_button == 3)
                {
                   select_button = 0;
                   goto First;
                }
             }
        if (select_button == 2)
        {
           Direction_Set();
           select_button = Check_Step_Data(300,330,200,290,350,440);
           if (select_button == 1)
           {
              Show_Steps();
           }
           if (select_button == 2)
           {
              goto First;
           }
        }
        if (select_button == 3)
        {
           select_button = 0;
        }
        closegraph();
        return 0;
}

void Mouse(void)
{
   regs.x.ax = 0x00;
   int86(0x33,&regs,&regs);
}

void Set_Limits(int min_X_limit,int max_X_limit,
                int min_Y_limit,int max_Y_limit)
{
   regs.x.ax = 0x07;
   regs.x.cx = min_X_limit;
   regs.x.dx = max_X_limit;
   int86(0x33,&regs,&regs);
   regs.x.ax = 0x08;
   regs.x.cx = min_Y_limit;
   regs.x.dx = max_Y_limit;
   int86(0x33,&regs,&regs);
}
```

```
void Show_Mouse(void)
{
   regs.x.ax = 0x01;
   int86(0x33,&regs,&regs);
}

void Hide_Mouse(void)
{
   regs.x.ax = 0x02;
   int86(0x33,&regs,&regs);
}

void Event(void)
{
   regs.x.ax = 0x03;
   int86(0x33,&regs,&regs);
   button = regs.x.bx;
   mx = regs.x.cx;
   my = regs.x.dx;
}

void Start_Graphics(void)
{
   int grdriver = DETECT,grmode;
   initgraph(&grdriver,&grmode," ");
}

void Main_Menu(void)
{
   cleardevice();
   setbkcolor(BLUE);
   setcolor(WHITE);
   rectangle(9,9,630,471);
   setfillstyle(SOLID_FILL,LIGHTGRAY);
   bar(20,20,620,100);
   setlinestyle(SOLID_LINE,0,3);
   setcolor(DARKGRAY);
   rectangle(20,20,620,100);
   setlinestyle(SOLID_LINE,0,1);
   Draw_Button(200,170,430,213);
   Draw_Button(200,270,430,313);
   Draw_Button(200,370,430,413);
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
   settextjustify(CENTER_TEXT,CENTER_TEXT);
   setcolor(YELLOW);
   outtextxy(320,55,"FIELD STRENGTH MEASURING SYSTEM");
   settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
   outtextxy(320,190,"MEASURING");
   outtextxy(320,290,"DIRECTION SET");
   outtextxy(320,390,"EXIT");
   Mouse();
   Set_Limits(0,getmaxx(),0,getmaxy());
   Show_Mouse();
}

void Measuring_Menu(void)
{
   cleardevice();
   setbkcolor(BROWN);
   setcolor(WHITE);
   setlinestyle(SOLID_LINE,0,1);
```

```
    rectangle(9,9,630,471);
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(80,20,560,100);
    setlinestyle(SOLID_LINE,0,3);
    setcolor(DARKGRAY);
    rectangle(80,20,560,100);
    setlinestyle(SOLID_LINE,0,1);
    Draw_Button(200,170,430,213);
    Draw_Button(200,270,430,313);
    Draw_Button(200,370,430,413);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(BLUE);
    outtextxy(320,55,"MEASURING MENU");
    settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
    outtextxy(320,190,"SETUP");
    outtextxy(320,290,"START MEASURING");
    outtextxy(320,390,"RETURN TO MAIN MENU");
    Mouse();
    Set_Limits(0,getmaxx(),0,getmaxy());
    Show_Mouse();
}

void Direction_Set(void)
{
    cleardevice();
    setbkcolor(BLACK);
    setfillstyle(SOLID_FILL,CYAN);
    bar(160,130,480,350);
    setlinestyle(SOLID_LINE,0,3);
    setcolor(MAGENTA);
    rectangle(160,130,480,350);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(YELLOW);
    outtextxy(328,150,"Type any step degrees (0 to 360)");
    Draw_Button(200,300,290,330);
    Draw_Button(350,300,440,330);
    settextstyle(SMALL_FONT,HORIZ_DIR,7);
    setcolor(BLUE);
    outtextxy(246,313,"OK");
    outtextxy(396,313,"CANCEL");
    Set_Limits(160,480,130,350);
    Show_Mouse();
}

void Measure_Set(void)
{
    cleardevice();
    setbkcolor(BLACK);
    setfillstyle(SOLID_FILL,CYAN);
    bar(160,130,480,350);
    setlinestyle(SOLID_LINE,0,3);
    setcolor(MAGENTA);
    rectangle(160,130,480,350);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(YELLOW);
    outtextxy(328,150,"Type measuring factor");
    Draw_Button(200,300,290,330);
    Draw_Button(350,300,440,330);
```

```
    settextstyle(SMALL_FONT,HORIZ_DIR,7);
    setcolor(BLUE);
    outtextxy(246,313,"OK");
    outtextxy(396,313,"CANCEL");
    Set_Limits(160,480,130,350);
    Show_Mouse();
}

void Draw_Button(int x_left,int y_top,int x_right,int y_bottom)
{
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(x_left,y_top,x_right,y_bottom);
    setlinestyle(SOLID_LINE,0,1);
    setcolor(WHITE);
    line(x_left,y_top,x_right,y_top);
    line(x_left,y_top,x_left,y_bottom);
    setcolor(DARKGRAY);
    line(x_right,y_top,x_right,y_bottom);
    line(x_left,y_bottom,x_right,y_bottom);
}

void Click_Button(int x_left,int y_top,int x_right,int y_bottom)
{
    int *buff;
    int size;
    size = imagesize(x_left+2,y_top+2,x_right-2,y_bottom-2);
    buff = (int *)malloc(size);
    Hide_Mouse();
    if (buff != 0)
    {
       getimage(x_left+2,y_top+2,x_right-2,y_bottom-2,buff);
       putimage(x_left+3,y_top+3,buff,COPY_PUT);
       free(buff);
    }
    setcolor(DARKGRAY);
    line(x_left,y_top,x_right,y_top);
    line(x_left,y_top,x_left,y_bottom);
    setcolor(WHITE);
    line(x_right,y_top,x_right,y_bottom);
    line(x_left,y_bottom,x_right,y_bottom);
    delay(1500);
}

int Check_Step_Data(int y_top,int y_bottom,int x_left1,
                    int x_right1,int x_left2,int x_right2)
{
    #define BACKSP 8
    #define ENTER 13
    #define ESC 27

    int mbutton = 0,i = 0,k,start = 310;

    Hide_Mouse();
    settextstyle(SMALL_FONT,HORIZ_DIR,6);
    setfillstyle(SOLID_FILL,WHITE);
    bar(start-10,220,start+30,251);
    setlinestyle(SOLID_LINE,0,1);
    setcolor(GREEN);
    rectangle(start-10,220,start+30,251);
    setcolor(RED);
    Show_Mouse();
```

```
   while (!mbutton)
      {
       Event();
       mbutton = button;
       if ((mbutton == 1)&&(mx > x_left1)&&(mx < x_right1)&&(my >
y_top)&&(my < y_bottom))
       {
          mbutton = 1;
          Click_Button(x_left1,y_top,x_right1,y_bottom);
       }
       else
       if ((mbutton == 1)&&(mx > x_left2)&&(mx < x_right2)&&(my >
y_top)&&(my < y_bottom))
       {
          mbutton = 2;
          Click_Button(x_left2,y_top,x_right2,y_bottom);
       }
       else
       if (((k = Key_Event()) != 0)&&(i <= 3))
       {
          setcolor(RED);
          if ((k > 47)&&(k < 58)&&(i <= 2))
          {
             Hide_Mouse();
             value[i] = (char)k;
             value[i+1] = '\0';
             outtextxy(start,233,&value[i]);
             start += 10;
             i++;
             Show_Mouse();
          }
          if ((i > 0)&&(k == BACKSP))
          {
             Hide_Mouse();
             i--;
             start -= 10;
             value[i] = '\0';
             bar(start-4,222,start+8,250);
             Show_Mouse();
          }
          if ((k == ENTER))
          {
             mbutton = 1;
             Click_Button(x_left1,y_top,x_right1,y_bottom);
          }
          if ((k == ESC))
          {
             mbutton = 2;
             Click_Button(x_left2,y_top,x_right2,y_bottom);
          }
       }
       else mbutton = 0;
      }
   return mbutton;
}
```

```
int Check_Factor_Data(int y_top,int y_bottom,int x_left1,
                      int x_right1,int x_left2,int x_right2)
{
   #define BACKSP 8
   #define ENTER 13
   #define ESC 27

   int mbutton = 0,i = 0,k,start = 310;

   Hide_Mouse();
   settextstyle(SMALL_FONT,HORIZ_DIR,6);
   setfillstyle(SOLID_FILL,WHITE);
   bar(start-10,220,start+30,251);
   setlinestyle(SOLID_LINE,0,1);
   setcolor(GREEN);
   rectangle(start-10,220,start+30,251);
   setcolor(RED);
   Show_Mouse();
   while (!mbutton)
      {
       Event();
       mbutton = button;
       if ((mbutton == 1)&&(mx > x_left1)&&(mx < x_right1)
                         &&(my > y_top)&&(my < y_bottom))
       {
          mbutton = 1;
          Click_Button(x_left1,y_top,x_right1,y_bottom);
       }
       else
       if ((mbutton == 1)&&(mx > x_left2)&&(mx < x_right2)
                         &&(my > y_top)&&(my < y_bottom))
       {
          mbutton = 2;
          Click_Button(x_left2,y_top,x_right2,y_bottom);
       }
       else
       if (((k = Key_Event()) != 0)&&(i <= 3))
       {
          setcolor(RED);
          if ((k > 47)&&(k < 58)&&(i <= 2))
          {
             Hide_Mouse();
             factor[i] = (char)k;
             factor[i+1] = '\0';
             outtextxy(start,233,&factor[i]);
             start += 10;
             i++;
             Show_Mouse();
          }
          if ((i > 0)&&(k == BACKSP))
          {
             Hide_Mouse();
             i--;
             start -= 10;
             factor[i] = '\0';
             bar(start-4,222,start+8,250);
             Show_Mouse();
          }
          if ((k == ENTER))
          {
             mbutton = 1;
```

```
            Click_Button(x_left1,y_top,x_right1,y_bottom);
         }
         if ((k == ESC))
         {
            mbutton = 2;
            Click_Button(x_left2,y_top,x_right2,y_bottom);
         }
       }
       else mbutton = 0;
      }
   return mbutton;
}

void Show_Steps(void)
{
   #define DELAY_ON1 3000
   #define DELAY_OFF1 3000
   #define DATA_PORT 0x378
   #define CTRL_PORT 0x37A
   #define RISING_EDGE 0x04
   #define FALLING_EDGE 0x05

   int step_data,x;
   unsigned char state;

   setcolor(YELLOW);
   step_data = (int)(atof(value)*1180/360);
   state = 0x01;
   for (x = 0;x < step_data;x++)
   {
      if (state == 0x08)
      state = 0x01;
      else state = state << 1;
      itoa(x+1,value,10);
      cleardevice();
      outtextxy(260,240,"STEP OF MOTOR NOW :");
      outtextxy(375,240,value);
      outportb(CTRL_PORT,RISING_EDGE);
      outportb(CTRL_PORT,FALLING_EDGE);
      outportb(DATA_PORT,state);
      outportb(CTRL_PORT,RISING_EDGE);
      delay(DELAY_ON1);
      outportb(CTRL_PORT,FALLING_EDGE);
      outportb(DATA_PORT,0x00);
      outportb(CTRL_PORT,RISING_EDGE);
      delay(DELAY_OFF1);
   }
}
```

```
void Field_Strength(void)
{
   #define DELAY_ON2 30
   #define DELAY_OFF2 30
   #define DATA_PORT 0x378
   #define CTRL_PORT 0x37A
   #define RISING_EDGE 0x04
   #define FALLING_EDGE 0x05
   #define CYCLE 1180
   #define A_8255 0x27C
   #define B_8255 0x27D
   #define C_8255 0x27E
   #define CTRL_8255 0x27F

   int x,count,times_measure,strength_data = 0,avg_strength;
   char cycle_step[5];
   float average_strength;
   unsigned char state;
   FILE *fpt;

   fpt = fopen("field.dat","w");
   times_measure = atoi(factor);
   outportb(CTRL_8255,130);
   state = 0x01;
   for (x = 0;x < CYCLE;x++)
   {
      if (state == 0x08)
      state = 0x01;
      else state = state << 1;
      itoa(x+1,cycle_step,10);
      cleardevice();
      outtextxy(260,240,"STEP OF MOTOR NOW :");
      outtextxy(400,240,cycle_step);
      outportb(CTRL_PORT,RISING_EDGE);
      outportb(CTRL_PORT,FALLING_EDGE);
      outportb(DATA_PORT,state);
      outportb(CTRL_PORT,RISING_EDGE);
      delay(DELAY_ON2);
      outportb(CTRL_PORT,FALLING_EDGE);
      outportb(DATA_PORT,0x00);
      outportb(CTRL_PORT,RISING_EDGE);
      delay(DELAY_OFF2);
      for (count = 0;count < times_measure;++count)
      {
       strength_data += (int)inportb(B_8255);
      }
      average_strength = strength_data/times_measure;
      strength_data = 0;
      avg_strength = (int)average_strength;
      fprintf(fpt,"%d\n",avg_strength);
   }
   fclose(fpt);
}
```

```
void Show_Graph(void)
{
   FILE *fpt;
   int strength_data= 0,x,y,cycle = 0,r;
   int x_previous = 320,y_previous = 240;
   float theta;

   cleardevice();
   setbkcolor(BLACK);
   circle(320,240,232);
   circle(320,240,158);
   circle(320,240,84);
   setlinestyle(1,0,1);
   line(320,8,320,472);
   line(88,240,550,240);
   settextstyle(SMALL_FONT,HORIZ_DIR,5);
   setcolor(WHITE);
   outtextxy(320,1,"0");
   outtextxy(564,240,"90");
   outtextxy(320,478,"180");
   outtextxy(74,240,"270");
   outtextxy(310,79,"6.984");
   outtextxy(310,153,"13.968");
   outtextxy(310,227,"20.952");

   fpt = fopen("field.dat","r");
   for (cycle = 0;cycle < 1180;cycle++)
   {
      setcolor(GREEN);
      setlinestyle(SOLID_LINE,0,1);
      fscanf(fpt,"%d",&strength_data);
      agc_voltage = (((float)strength_data)*5)/255;
      ant_input = (agc_voltage-1.8)/105;
      r = (int)(ant_input*234/20.952e-3);
      theta = cycle*(2*3.1416/1180);
      x = (int)(320+((r*sin(theta))));
      y = (int)(240-((r*cos(theta))));
      line(x,y,x_previous,y_previous);
      x_previous = x;
      y_previous = y;
   }
   fclose(fpt);
   while (bioskey(1) == 0);
}

void Arrange_Data(void)
{
   int i;
   int j;
   float count;
   int data,time;
   FILE *fpt1,*fpt2;

   fpt2 = fopen("strength.dat","w");
   for (i = 0;i < 360;i++)
   {
      count = i*(1175/360);
      time = (int)count;
      time++;
      time--;
      fpt1 = fopen("field.dat","r");
```

```
        for (j = 0;j < 1180;j++)
        {
        fscanf(fpt1,"%d",&data);
        if (j == time)
        {
           fprintf(fpt2,"%d\n",data);
           goto check;
        }
        }
        check :fclose(fpt1);
     }
     fclose(fpt2);
  }

  void More_Info(int x_left,int y_top,int x_right,int y_bottom)
  {
     #define DELAY_ON1 3000
     #define DELAY_OFF1 3000
     #define DATA_PORT 0x378
     #define CTRL_PORT 0x37A
     #define RISING_EDGE 0x04
     #define FALLING_EDGE 0x05

     FILE *fpt;
     int data,dataold = 0,maxdegree,i,j = 0,k = -1,mbutton = 0;
     float maxdata;
     char data_max[6],degree_max[6];
     int step_data,x;
     unsigned char state;

     fpt = fopen("strength.dat","r");
     for (i = 0;i < 360;i++)
     {
        k++;
        fscanf(fpt,"%d",&data);
        if (data > dataold)
        {
         dataold = data;
         j += k;
         k = 0;
        }
     }
     maxdata = dataold*0.4;
     maxdegree = j;
     cleardevice();
     setbkcolor(DARKGRAY);
     setcolor(WHITE);
     setlinestyle(SOLID_LINE,0,1);
     rectangle(9,9,630,471);
     settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
     setcolor(LIGHTBLUE);
     outtextxy(320,150,"MAXIMUM DATA :");
     outtextxy(320,200,"MAXIMUM DEGREE :");
     Draw_Button(250,325,380,368);
     setcolor(YELLOW);
     outtextxy(320,345,"DONE");
     gcvt(maxdata,6,data_max);
     itoa(maxdegree,degree_max,10);
     setcolor(WHITE);
     outtextxy(435,150,data_max);
     outtextxy(445,200,degree_max);
```

```
    Show_Mouse();
    while (!mbutton)
       {
        Event();
        mbutton = button;
        if ((mbutton == 1)&&(mx > x_left)&&(mx < x_right)
                          &&(my > y_top)&&(my < y_bottom))
        {
           mbutton = 1;
           Click_Button(x_left,y_top,x_right,y_bottom);
        }
        else mbutton = 0;
       }
    cleardevice();
    setcolor(YELLOW);
    step_data = maxdata;
    state = 0x01;
    for (x = 0;x < step_data;x++)
    {
       if (state == 0x08)
       state = 0x01;
       else state = state << 1;
       itoa(x+1,value,10);
       cleardevice();
       outtextxy(260,240,"STEP OF MOTOR NOW :");
       outtextxy(375,240,value);
       outportb(CTRL_PORT,RISING_EDGE);
       outportb(CTRL_PORT,FALLING_EDGE);
       outportb(DATA_PORT,state);
       outportb(CTRL_PORT,RISING_EDGE);
       delay(DELAY_ON1);
       outportb(CTRL_PORT,FALLING_EDGE);
       outportb(DATA_PORT,0x00);
       outportb(CTRL_PORT,RISING_EDGE);
       delay(DELAY_OFF1);
       mbutton = 0;
    }
 }

 void Field_Info(void)
 {
    #define BACKSP 8
    #define ENTER 13

    FILE *fpt;
    int mbutton = 0,start = 310,degree_data[361],
        count,data,k,i = 0,degree_read;
    char user_degree[5],show_field[5];
    float user_field;

    cleardevice();
    setbkcolor(BLACK);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    outtextxy(320,30,"Type any degrees and press ENTER");
    outtextxy(160,150,"DEGREE");
    outtextxy(145,250,"DATA");
    Draw_Button(250,370,380,413);
    settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
    setcolor(BLUE);
    outtextxy(318,388,"DONE");
```

```
setfillstyle(SOLID_FILL,WHITE);
settextstyle(SMALL_FONT,HORIZ_DIR,6);
bar(start-10,135,start+30,165);
bar(start-10,235,start+50,265);
setlinestyle(SOLID_LINE,0,1);
fpt = fopen("strength.dat","r");
for (count = 0;count < 360;count++)
{
   fscanf(fpt,"%d",&data);
   degree_data[count] = data;
}
fclose(fpt);
setcolor(RED);
Show_Mouse();
while (!mbutton)
{
   Event();
   mbutton = button;
   if ((mbutton == 1)&&(mx > 250)&&(mx < 380)&&(my > 370)&&(my < 413))
   {
    mbutton = 1;
    Click_Button(250,370,380,413);
   }
   else
   if (((k = Key_Event()) != 0)&&(i <= 3))
   {
    if ((k > 47)&&(k < 58)&&(i <= 2))
    {
       Hide_Mouse();
       user_degree[i] = (char)k;
       user_degree[i+1] = '\0';
       outtextxy(start,148,&user_degree[i]);
       start += 10;
       i++;
       Show_Mouse();
    }
    if ((i > 0)&&(k == BACKSP))
    {
       Hide_Mouse();
       i--;
       start -= 10;
       user_degree[i] = '\0';
       bar(start-4,137,start+8,250);
       Show_Mouse();
    }
    if ((k == ENTER))
    {
       Hide_Mouse();
       start = 310;
       setfillstyle(SOLID_FILL,WHITE);
       bar(300,135,340,165);
       bar(300,235,360,265);
       degree_read = atoi(user_degree);
       user_field = 20*(log10((degree_data[degree_read])-2.1)/
                    (105e-3)));
       gcvt(user_field,6,show_field);
       outtextxy(start,248,show_field);
       i = 0;
       start = 310;
       Show_Mouse();
    }
```

```
        }
        else mbutton = 0;
    }
}

int Check_Button(int x_left,int x_right,int y_top1,int y_bottom1,
                 int y_top2,int y_bottom2,int y_top3,int y_bottom3)
{
    int mbutton = 0;

    while (!mbutton)
        {
         Event();
         mbutton = button;
         if ((mbutton == 1)&&(mx > x_left)&&(mx < x_right)
                          &&(my > y_top1)&&(my < y_bottom1))
         {
            mbutton = 1;
            Click_Button(x_left,y_top1,x_right,y_bottom1);
         }
         else
         if ((mbutton == 1)&&(mx > x_left)&&(mx < x_right)
                          &&(my > y_top2)&&(my < y_bottom2))
         {
            mbutton = 2;
            Click_Button(x_left,y_top2,x_right,y_bottom2);
         }
         else
         if ((mbutton == 1)&&(mx > x_left)&&(mx < x_right)
                          &&(my > y_top3)&&(my < y_bottom3))
         {
            mbutton = 3;
            Click_Button(x_left,y_top3,x_right,y_bottom3);
         }
         else mbutton = 0;
        }
    return mbutton;
}

int Key_Event(void)
{
    unsigned int key;

    key = bioskey(1);
    if (key)
    {
       key = bioskey(0);
       key &= 0xFF;
    }
    return key;
}
```

ภาคผนวก ข. คาต้าชีท

HARRIS SEMICONDUCTOR

ADC0802, ADC0803 ADC0804

8-Bit μP Compatible A/D Converters

December 1993

Features

- 80C48 and 80C80/85 Bus Compatible - No Interfacing Logic Required
- Conversion Time < 100μs
- Easy Interface to Most Microprocessors
- Will Operate in a "Stand Alone" Mode
- Differential Analog Voltage Inputs
- Works with Bandgap Voltage References
- TTL Compatible Inputs and Outputs
- On-Chip Clock Generator
- 0V to 5V Analog Voltage Input Range (Single + 5V Supply)
- No Zero-Adjust Required

Description

The ADC0802 family are CMOS 8-Bit successive approximation A/D converters which use a modified potentiometric ladder and are designed to operate with the 8080A control bus via three-state outputs. These converters appear to the processor as memory locations or I/O ports, and hence no interfacing logic is required.

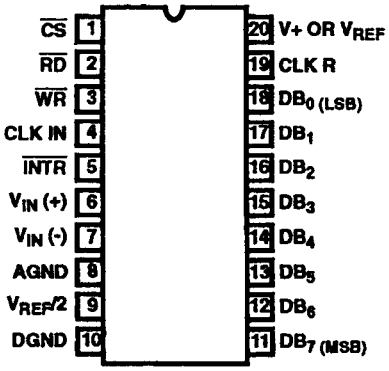
The differential analog voltage input has good common-mode-rejection and permits offsetting the analog zero-input-voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8-Bits of resolution.

Ordering Information

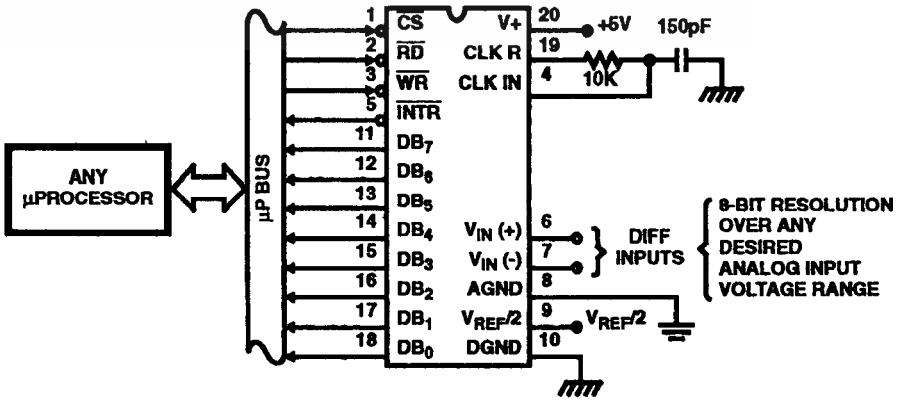
PART NUMBER	ERROR	EXTERNAL CONDITIONS	TEMPERATURE RANGE	PACKAGE
ADC0802LCN	$\pm^1/_2$ LSB	$V_{REF}/2 = 2.500\ V_{DC}$ (No Adjustments)	0°C to +70°C	20 Lead Plastic DIP
ADC0802LCD	$\pm^3/_4$ LSB		-40°C to +85°C	20 Lead Ceramic DIP
ADC0802LD	±1 LSB		-55°C to +125°C	20 Lead Ceramic DIP
ADC0803LCN	$\pm^1/_2$ LSB	$V_{REF}/2$ Adjusted for Correct Full-Scale Reading	0°C to +70°C	20 Lead Plastic DIP
ADC0803LCD	$\pm^3/_4$ LSB		-40°C to +85°C	20 Lead Ceramic DIP
ADC0802LCWM	±1 LSB		-40°C to +85°C	20 Lead SOIC (W)
ADC0803LD	±1 LSB		-55°C to +125°C	20 Lead Ceramic DIP
ADC0804LCN	±1 LSB	$V_{REF}/2 = 2.500\ V_{DC}$ (No Adjustments)	0°C to +70°C	20 Lead Plastic DIP
ADC0804LCD	±1 LSB		-40°C to +85°C	20 Lead Ceramic DIP

Pinout

ADC0802, ADC0803, ADC0804
(PDIP, CDIP)
TOP VIEW



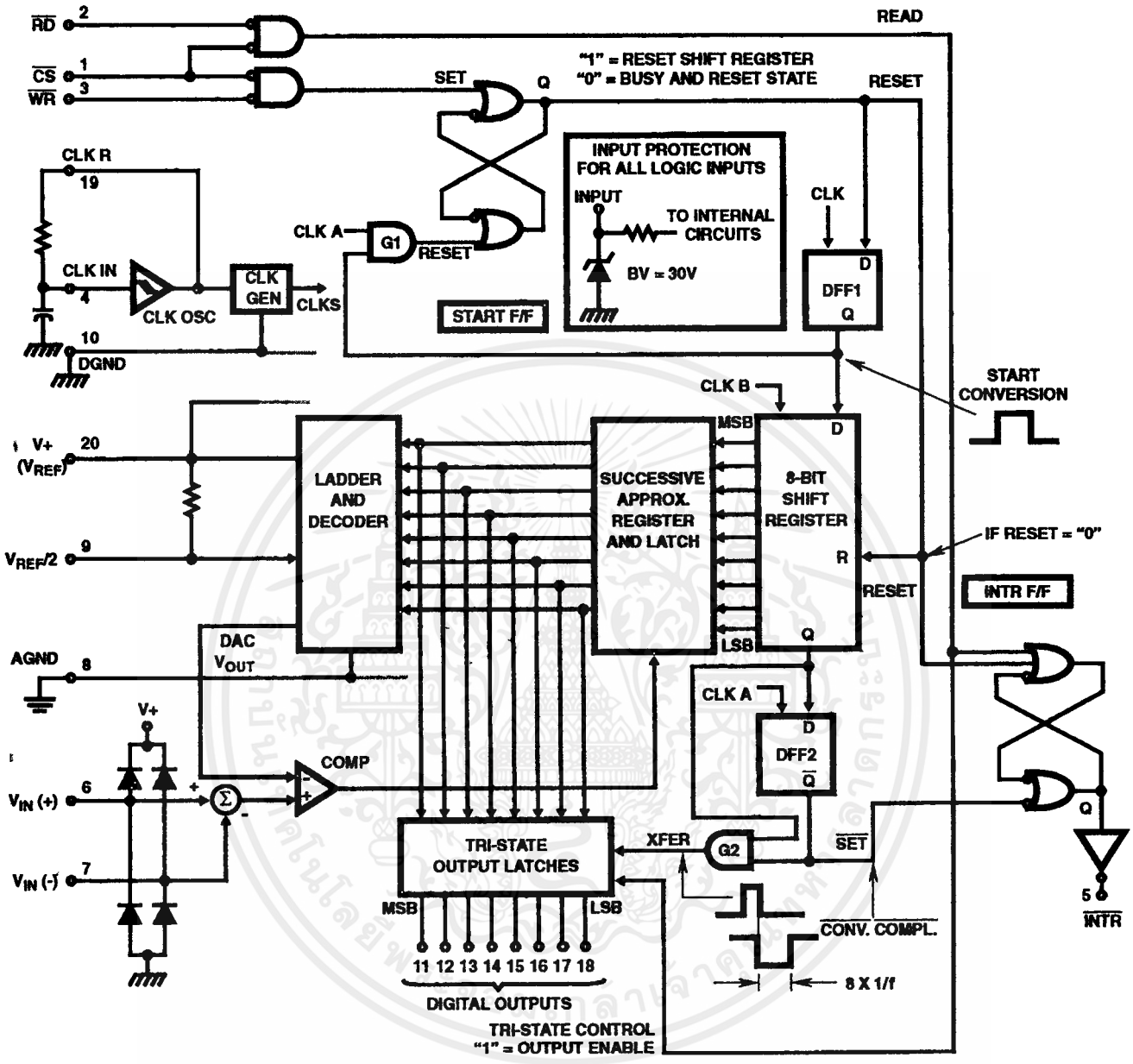
Typical Application Schematic



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper I.C. Handling Procedures.

File Number 3094

Functional Diagram



Absolute Maximum Ratings

Supply Voltage 6.5V
Voltage at Any Input . . . . . -0.3V to (V^+ +0.3V)
Storage Temperature Range . . . . . -65°C to +150°C
Lead Temperature (Soldering, 10s) . . . . . +300°C

Thermal Information

Thermal Resistance	θ_{JA}	θ_{JC}
Plastic DIP Package	125°C/W	-
Ceramic DIP Package	70°C/W	20°C/W
SOIC Package	120°C/W	-

Operating Temperature Range
ADC0802/03LD . . . . . -55°C to +125°C
ADC0802/03/04LCD . . . . . -40°C to +85°C
ADC0802/03/04LCN . . . . . 0°C to +70°C
ADC0803LCWM . . . . . -40°C to +85°C
Junction Temperature
Ceramic Package . . . . . +175°C
Plastic Package . . . . . +150°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Specifications (Notes 1, 7)

PARAMETERS	TEST CONDITIONS	MIN	TYP	MAX	UNITS
CONVERTER SPECIFICATIONS	$V+ = 5V$, $T_A = +25°C$ and $f_{CLK} = 640kHz$, Unless Otherwise Specified				
Total Unadjusted Error					
ADC0802	$V_{REF}/2 = 2.500V$	-	-	$\pm 1/2$	LSB
ADC0803	$V_{REF}/2$ Adjusted for Correct Full-Scale Reading	-	-	$\pm 1/2$	LSB
ADC0804	$V_{REF}/2 = 2.500V$	-	-	±1	LSB
$V_{REF}/2$ Input Resistance	Input Resistance at Pin 9	1.0	1.3	-	kΩ
Analog Input Voltage Range	(Note 2)	GND-0.05	-	(V+) + 0.05	V
DC Common-Mode Rejection	Over Analog Input Voltage Range	-	$\pm 1/16$	$\pm 1/8$	LSB
Power Supply Sensitivity	V+ = 5V ±10% Over Allowed Input Voltage Range	-	$\pm 1/16$	$\pm 1/8$	LSB
CONVERTER SPECIFICATIONS	$V+ = 5V$, $0°C \le T_A \le +70°C$ and $f_{CLK} = 640kHz$, Unless Otherwise Specified				
Total Unadjusted Error					
ADC0802	$V_{REF}/2 = 2.500V$	-	-	$\pm 1/2$	LSB
ADC0803	$V_{REF}/2$ Adjusted for Correct Full-Scale Reading	-	-	$\pm 1/2$	LSB
ADC0804	$V_{REF}/2 = 2.500V$	-	-	±1	LSB
$V_{REF}/2$ Input Resistance	Input Resistance at Pin 9	1.0	1.3	-	kΩ
Analog Input Voltage Range	(Note 2)	GND-0.05	-	(V+) + 0.05	V
DC Common-Mode Rejection	Over Analog Input Voltage Range	-	$\pm 1/8$	$\pm 1/4$	LSB
Power Supply Sensitivity	V+ = 5V ±10% Over Allowed Input Voltage Range	-	$\pm 1/16$	$\pm 1/8$	LSB
CONVERTER SPECIFICATIONS	$V+ = 5V$, $-25°C \le T_A \le +85°C$ and $f_{CLK} = 640kHz$, Unless Otherwise Specified				
Total Unadjusted Error					
ADC0802	$V_{REF}/2 = 2.500V$	-	-	$\pm 3/4$	LSB
ADC0803	$V_{REF}/2$ Adjusted for Correct Full-Scale Reading	-	-	$\pm 3/4$	LSB
ADC0804	$V_{REF}/2 = 2.500V$	-	-	±1	LSB
$V_{REF}/2$ Input Resistance	Input Resistance at Pin 9	1.0	1.3	-	kΩ
Analog Input Voltage Range	(Note 2)	GND-0.05	-	(V+) + 0.05	V
DC Common-Mode Rejection	Over Analog Input Voltage Range	-	$\pm 1/8$	$\pm 1/4$	LSB
Power Supply Sensitivity	V+ = 5V ±10% Over Allowed Input Voltage Range	-	$\pm 1/16$	$\pm 1/8$	LSB

Electrical Specifications (Notes 1, 7) **(Continued)**

PARAMETERS	TEST CONDITIONS	MIN	TYP	MAX	UNITS
CONVERTER SPECIFICATIONS $V+ = 5V$, $-55°C \le T_A \le +125°C$ and $f_{CLK} = 640kHz$, Unless Otherwise Specified					
Total Unadjusted Error					
ADC0802	$V_{REF}/2 = 2.500V$	-	-	±1	LSB
ADC0803	$V_{REF}/2$ Adjusted for Correct Full-Scale Reading	-	-	±1	LSB
$V_{REF}/2$ Input Resistance	Input Resistance at Pin 9	1.0	1.3	-	kΩ
Analog Input Voltage Range	(Note 2)	GND-0.05	-	(V+) + 0.05	V
DC Common-Mode Rejection	Over Analog Input Voltage Range	-	$\pm^1/_8$	$\pm^1/_4$	LSB
Power Supply Sensitivity	V+ = 5V ±10% Over Allowed Input Voltage Range	-	$\pm^1/_8$	$\pm^1/_4$	LSB
AC TIMING SPECIFICATIONS $V+ = 5V$, and $T_A = +25°C$, Unless Otherwise Specified					
Clock Frequency, f_{CLK}	V+ = 6V (Note3)	100	640	1280	kHz
	V+ = 5V	100	640	800	kHz
Clock Periods per Conversion (Note 4), t_{CONV}		62	-	73	clocks/conv
Conversion Rate In Free-Running Mode, CR	$\overline{INTR}$ tied to $\overline{WR}$ with $\overline{CS} = 0V$, $f_{CLK} = 640kHz$	-	-	8888	conv/s
Width of $\overline{WR}$ Input (Start Pulse Width), $t_{W(WR)l}$	$\overline{CS} = 0V$ (Note 5)	100	-	-	ns
Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Data Valid), t_{ACC}	$C_L = 100pF$ (Use Bus Driver IC for larger C_L)	-	135	200	ns
Tri-State Control (Delay from Rising Edge of $\overline{RD}$ to HI-Z State), t_{1H}, t_{0H}	$C_L = 10pF$, $R_L = 10k$ (See Tri-State Test Circuits)	-	125	250	ns
Delay from Falling Edge of $\overline{WR}$ to Reset of $\overline{INTR}$, t_{WI}, t_{RI}		-	300	450	ns
Input Capacitance of Logic Control Inputs, C_{IN}		-	5	-	pF
Tri-State Output Capacitance (Data Buffers), C_{OUT}		-	5	-	pF
DC DIGITAL LEVELS AND DC SPECIFICATIONS $V+ = 5V$, and $T_{MIN} \le T_A \le T_{MAX}$, Unless Otherwise Specified					
CONTROL INPUTS (Note 6)					
Logic "1" Input Voltage (Except Pin 4 CLK IN), V_{INH}	V+ = 5.25V	2.0	-	V+	V
Logic "0" Input Voltage (Except Pin 4 CLK IN), V_{INL}	V+ = 4.75V	-	-	0.8	V
CLK IN (Pin 4) Positive Going Threshold Voltage, $V+_{CLK}$		2.7	3.1	3.5	V
CLK IN (Pin 4) Negative Going Threshold Voltage, $V-_{CLK}$		1.5	1.8	2.1	V
CLK IN (Pin 4) Hysteresis, V_H		0.6	1.3	2.0	V
Logic "1" Input Current (All Inputs), I_{INHI}	$V_{IN} = 5V$	-	0.005	1	µA
Logic "0" Input Current (All Inputs), I_{INLO}	$V_{IN} = 0V$	-1	-0.005	-	µA
Supply Current (Includes Ladder Current), I+	$f_{CLK} = 640kHz$, $T_A = +25°C$ and $\overline{CS}$ = HI	-	1.3	2.5	mA

Electrical Specifications (Notes 1, 7) (Continued)

PARAMETERS	TEST CONDITIONS	MIN	TYP	MAX	UNITS
DC DIGITAL LEVELS AND DC SPECIFICATIONS $V+ = 5V$, and $T_{MIN} \le T_A \le T_{MAX}$, Unless Otherwise Specified (Continued)					
DATA OUTPUTS AND $\overline{INTR}$					
Logic "0" Output Voltage, V_{OL}	$I_O = 1.6mA$ $V+ = 4.75V$	-	-	0.4	V
Logic "1" Output Voltage, V_{OH}	$I_O = -360\mu A$ $V+ = 4.75V$	2.4	-	-	V
Tri-State Disabled Output Leakage (All Data Buffers), I_{LO}	$V_{OUT} = 0V$	-3	-	-	µA
	$V_{OUT} = 5V$	-	-	3	µA
Output Short Circuit Current, I_{SOURCE}	V_{OUT} Short to Gnd $T_A = +25°C$	4.5	6	-	mA
Output Short Circuit Current, I_{SINK}	V_{OUT} Short to V + $T_A = +25°C$	9.0	16	-	mA

NOTES:

1. All voltages are measured with respect to GND, unless otherwise specified. The separate AGND point should always be wired to the DGND, being careful to avoid ground loops.
2. For $V_{IN(-)} \ge V_{IN(+)}$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see Block Diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V+ supply. Be careful, during testing at low V+ levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct-especially at elevated temperatures, and cause errors for analog inputs near full-scale. As long as the analog V_{IN} does not exceed the supply voltage by more than 50mV, the output code will be correct. To achieve an absolute 0V to 5V input voltage range will therefore require a minimum supply voltage of 4.950V over temperature variations, initial tolerance and loading.
3. With V+ = 6V, the digital logic interfaces are no longer TTL compatible.
4. With an asynchronous start pulse, up to 8 clock periods may be required before the internal clock phases are proper to start the conversion process.
5. The $\overline{CS}$ input is assumed to bracket the $\overline{WR}$ strobe input so that timing is dependent on the $\overline{WR}$ pulse width. An arbitrarily wide pulse width will hold the converter in a reset mode and the start of conversion is initiated by the low to high transition of the $\overline{WR}$ pulse (see Timing Diagrams).
6. CLK IN (pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately.
7. None of these A/Ds requires a zero-adjust. However, if an all zero code is desired for an analog input other than 0.0V, or if a narrow full-scale span exists (for example: 0.5V to 4.0V full-scale) the $V_{IN(-)}$ input can be adjusted to achieve this. See Zero Error on page 13 of this data sheet.

Timing Waveforms

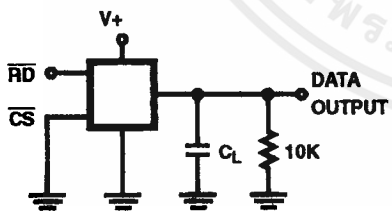


FIGURE 1A. t_{1H}

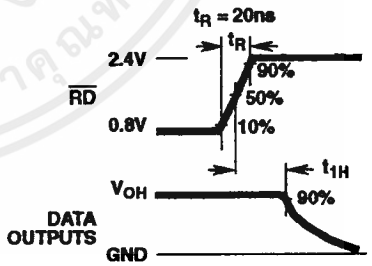


FIGURE 1B. t_{1H}, C_L = 10pF

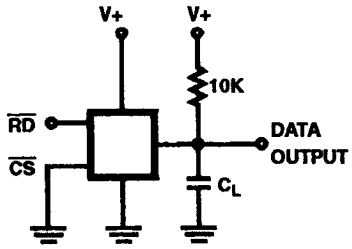


FIGURE 1C. t_{0H}

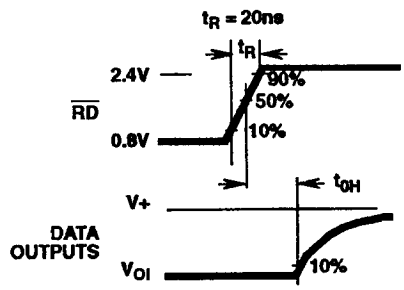


FIGURE 1D. t_{0H}, C_L = 10pF

FIGURE 1. TRI-STATE CIRCUITS AND WAVEFORMS

Typical Performance Curves

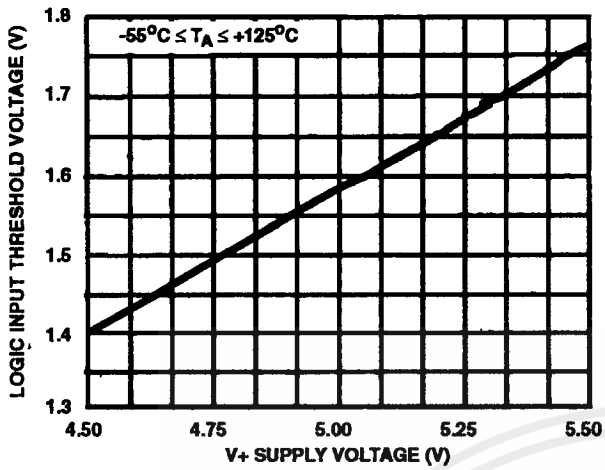


FIGURE 2. LOGIC INPUT THRESHOLD VOLTAGE vs SUPPLY VOLTAGE

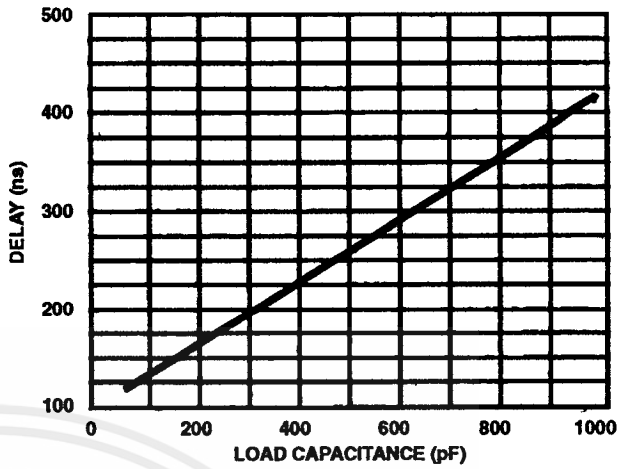


FIGURE 3. DELAY FROM FALLING EDGE OF $\overline{RD}$ TO OUTPUT DATA VALID vs LOAD CAPACITANCE

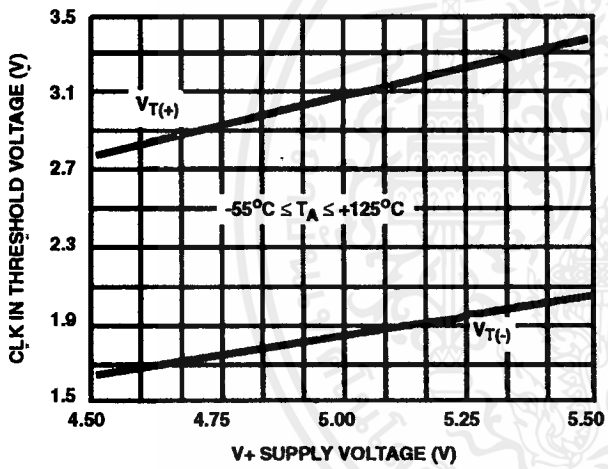


FIGURE 4. CLK IN SCHMITT TRIP LEVELS vs SUPPLY VOLTAGE

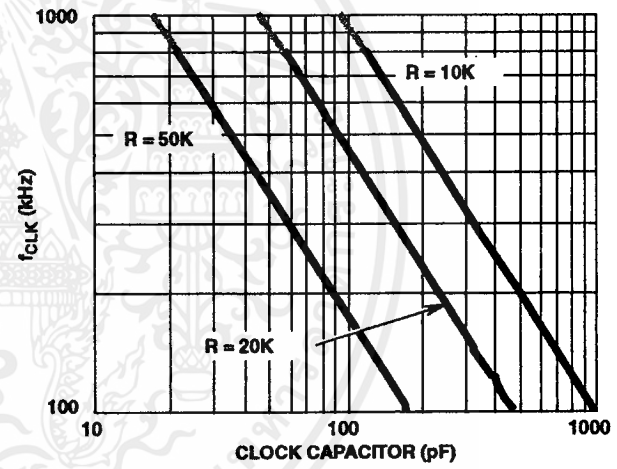


FIGURE 5. f_{CLK} vs CLOCK CAPACITOR

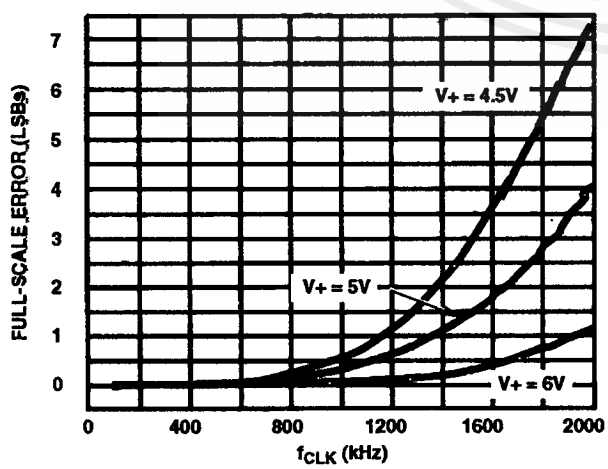


FIGURE 6. FULL SCALE ERROR vs f_{CLK}

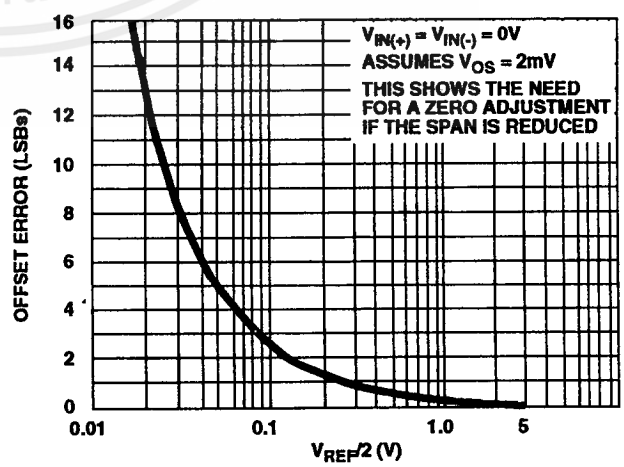


FIGURE 7. EFFECT OF UNADJUSTED OFFSET ERROR

Typical Performance Curves (Continued)

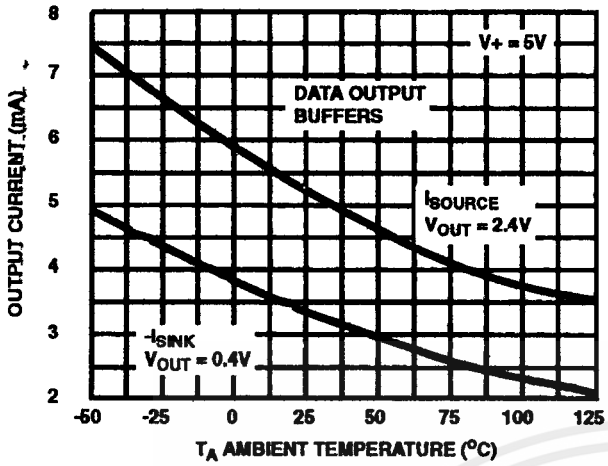


FIGURE 8. OUTPUT CURRENT vs TEMPERATURE

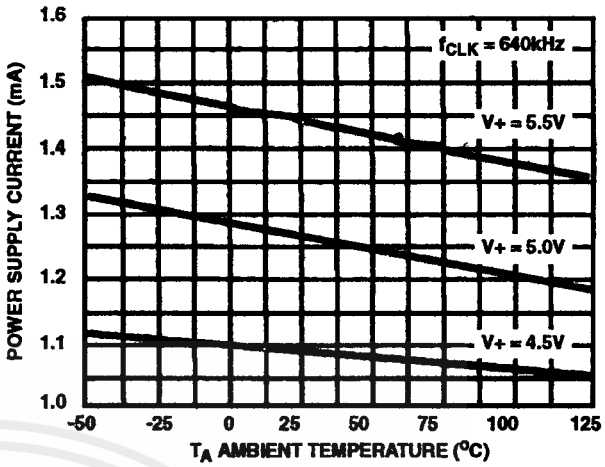


FIGURE 9. POWER SUPPLY CURRENT vs TEMPERATURE

Timing Diagrams

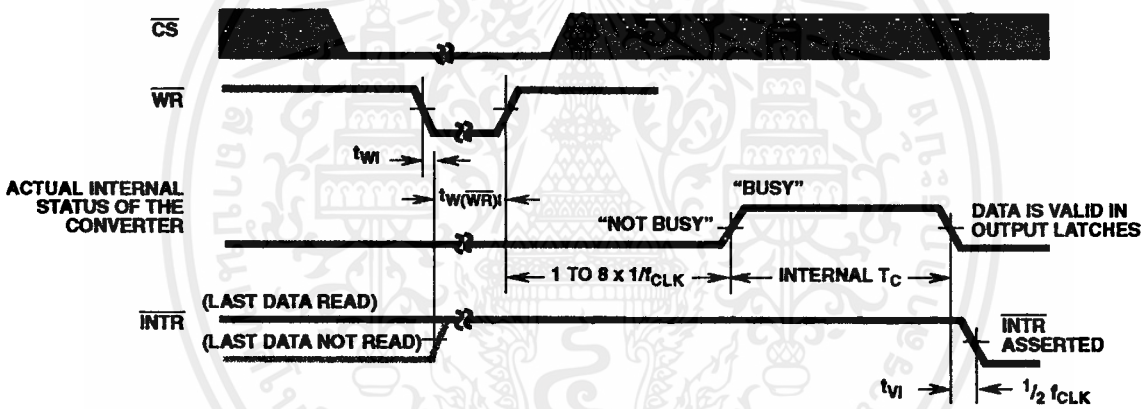


FIGURE 10A. START CONVERSION

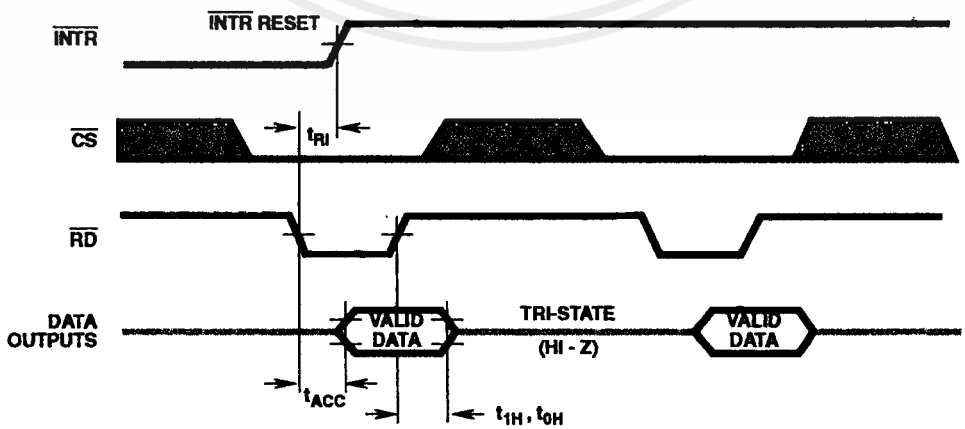


FIGURE 10B. OUTPUT ENABLE AND RESET INTR

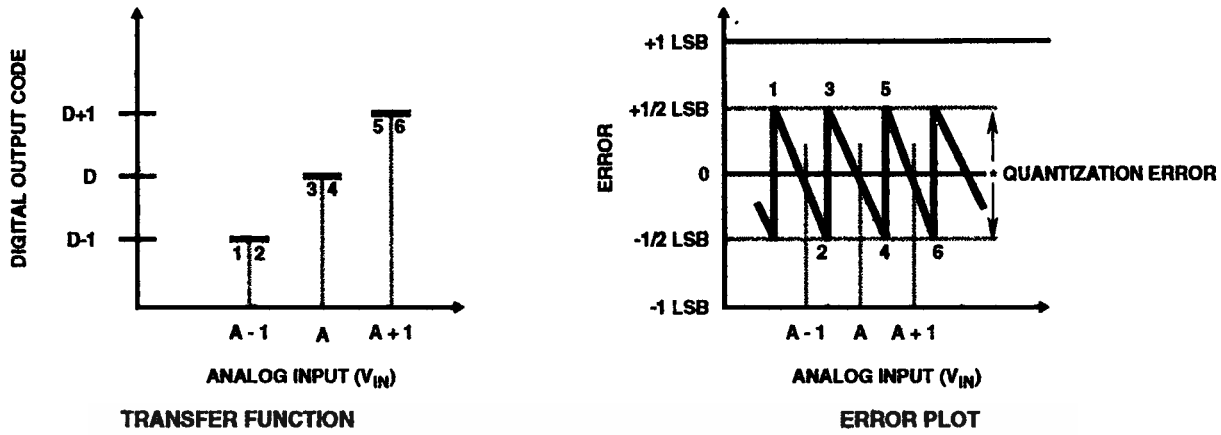


FIGURE 11A. ACCURACY = ±0 LSB; PERFECT A/D

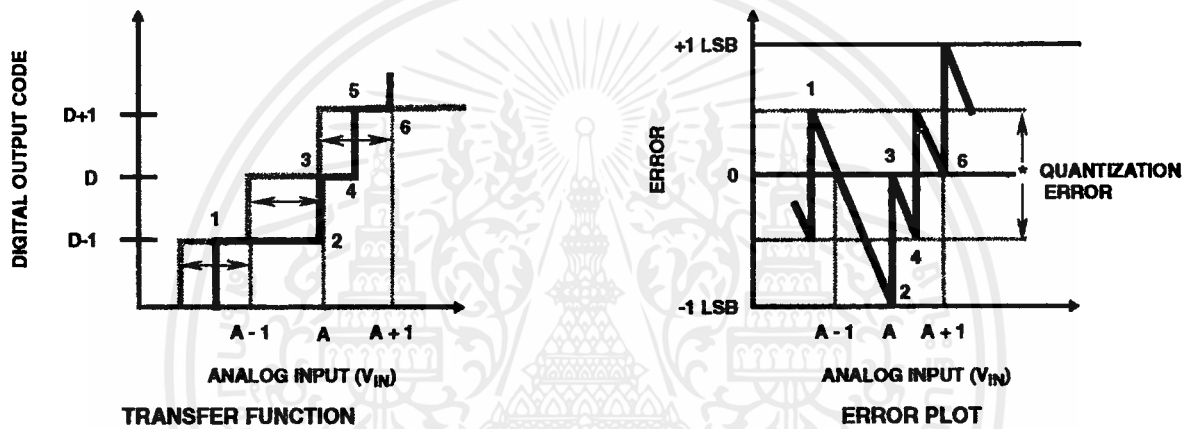


FIGURE 11B. ACCURACY = $\pm^1/_2$ LSB

FIGURE 11. CLARIFYING THE ERROR SPECS OF AN A/D CONVERTER

Understanding A/D Error Specs

A perfect A/D transfer characteristic (staircase wave-form) is shown in Figure 11A. The horizontal scale is analog input voltage and the particular points labeled are in steps of 1 LSB (19.53mV with 2.5V tied to the $V_{REF}/2$ pin). The digital output codes which correspond to these inputs are shown as D-1, D, and D+1. For the perfect A/D, not only will center-value (A - 1, A, A + 1, . . .) analog inputs produce the correct output digital codes, but also each riser (the transitions between adjacent output codes) will be located $\pm^1/_2$ LSB away from each center-value. As shown, the risers are ideal and have no width. Correct digital output codes will be provided for a range of analog input voltages which extend $\pm^1/_2$ LSB from the ideal center-values. Each tread (the range of analog input voltage which provides the same digital output code) is therefore 1 LSB wide.

The error curve of Figure 11B shows the worst case transfer function for the ADC0802. Here the specification guarantees that if we apply an analog input equal to the LSB analog voltage center-value, the A/D will produce the correct digital code.

Next to each transfer function is shown the corresponding error plot. Notice that the error includes the quantization uncertainty of the A/D. For example, the error at point 1 of Figure 11A is $+^1/_2$ LSB because the digital code appeared $^1/_2$ LSB in advance of the center-value of the tread. The error plots always have a constant negative slope and the abrupt upside steps are always 1 LSB in magnitude, unless the device has missing codes.

Detailed Description

The functional diagram of the ADC0802 series of A/D converters operates on the successive approximation principle (see Application Notes AN016 and AN020 for a more detailed description of this principle). Analog switches are closed sequentially by successive-approximation logic until the analog differential input voltage $[V_{IN(+)} - V_{IN(-)}]$ matches a voltage derived from a tapped resistor string across the reference voltage. The most significant bit is tested first and after 8 comparisons (64 clock cycles), an 8-bit binary code (1111 1111 = full-scale) is transferred to an output latch.

The normal operation proceeds as follows. On the high-to-low transition of the $\overline{WR}$ input, the internal SAR latches and the shift-register stages are reset, and the $\overline{INTR}$ output will be set high. As long as the $\overline{CS}$ input and $\overline{WR}$ input remain low, the A/D will remain in a reset state. Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low-to-high transition. After the requisite number of

clock pulses to complete the conversion, the $\overline{INTR}$ pin will make a high-to-low transition. This can be used to interrupt a processor, or otherwise signal the availability of a new conversion. A $\overline{RD}$ operation (with $\overline{CS}$ low) will clear the $\overline{INTR}$ line high again. The device may be operated in the free-running mode by connecting $\overline{INTR}$ to the $\overline{WR}$ input with $\overline{CS} = 0$. To ensure start-up under all possible conditions, an external $\overline{WR}$ pulse is required during the first power-up cycle. A conversion-in-process can be interrupted by issuing a second start command.

Digital Operation

The converter is started by having $\overline{CS}$ and $\overline{WR}$ simultaneously low. This sets the start flip-flop (F/F) and the resulting "1" level resets the 8-bit shift register, resets the Interrupt (INTR) F/F and inputs a "1" to the D flip-flop, DFF1, which is at the input end of the 8-bit shift register. Internal clock signals then transfer this "1" to the Q output of DFF1. The AND gate, G1, combines this "1" output with a clock signal to provide a reset signal to the start F/F. If the set signal is no longer present (either $\overline{WR}$ or $\overline{CS}$ is a "1"), the start F/F is reset and the 8-bit shift register then can have the "1" clocked in, which starts the conversion process. If the set signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F would be at a "1" level) and the 8-bit shift register would continue to be held in the reset mode. This allows for asynchronous or wide $\overline{CS}$ and $\overline{WR}$ signals.

After the "1" is clocked through the 8-bit shift register (which completes the SAR operation) it appears as the input to DFF2. As soon as this "1" is output from the shift register, the AND gate, G2, causes the new digital word to transfer to the Tri-State output latches. When DFF2 is subsequently clocked, the $\overline{Q}$ output makes a high-to-low transition which causes the INTR F/F to set. An inverting buffer then supplies the $\overline{INTR}$ output signal.

When data is to be read, the combination of both $\overline{CS}$ and $\overline{RD}$ being low will cause the INTR F/F to be reset and the tri-state output latches will be enabled to provide the 8-bit digital outputs.

Digital Control Inputs

The digital control inputs ($\overline{CS}$, $\overline{RD}$, and $\overline{WR}$) meet standard TTL logic voltage levels. These signals are essentially equivalent to the standard A/D Start and Output Enable control signals, and are active low to allow an easy interface to microprocessor control busses. For non-microprocessor based applications, the $\overline{CS}$ input (pin 1) can be grounded and the standard A/D Start function obtained by an active low pulse at the $\overline{WR}$ input (pin 3). The Output Enable function is achieved by an active low pulse at the $\overline{RD}$ input (pin 2).

Analog Operation

The analog comparisons are performed by a capacitive charge summing circuit. Three capacitors (with precise ratioed values) share a common node with the input to an auto-zeroed comparator. The input capacitor is switched between $V_{IN(+)}$ and $V_{IN(-)}$, while two ratioed reference capacitors are switched between taps on the reference voltage divider string. The net charge corresponds to the weighted difference between the input and the current total value set by the successive approximation register. A correction is made to offset the comparison by $^1/_2$ LSB (see Figure 11A).

Analog Differential Voltage Inputs and Common-Mode Rejection

This A/D gains considerable applications flexibility from the analog differential voltage input. The $V_{IN(-)}$ input (pin 7) can be used to automatically subtract a fixed voltage value from the input reading (tare correction). This is also useful in 4mA - 20mA current loop conversion. In addition, common-mode noise can be reduced by use of the differential input.

The time interval between sampling $V_{IN(+)}$ and $V_{IN(-)}$ is 4 $^1/_2$ clock periods. The maximum error voltage due to this slight time difference between the input voltage samples is given by:

$$\Delta V_E(MAX) = (V_P)(2\pi f_{CM})\left[\frac{4.5}{f_{CLK}}\right]$$

where:

ΔV_E is the error voltage due to sampling delay

V_P is the peak value of the common-mode voltage

f_{CM} is the common-mode frequency

For example, with a 60Hz common-mode frequency, f_{CM}, and a 640kHz A/D clock, f_{CLK}, keeping this error to $^1/_4$ LSB (~5mV) would allow a common-mode voltage, V_P, given by:

$$V_P = \frac{\left[\Delta V_{E(MAX)}(f_{CLK})\right]}{(2\pi f_{CM})(4.5)}$$

or

$$V_P = \frac{(5\times10^{-3})(640\times10^{3})}{(6.28)(60)(4.5)} \cong 1.9V$$

The allowed range of analog input voltage usually places more severe restrictions on input common-mode voltage levels than this.

An analog input voltage with a reduced span and a relatively large zero offset can be easily handled by making use of the differential input (see Reference Voltage Span Adjust).

Analog Input Current

The internal switching action causes displacement currents to flow at the analog inputs. The voltage on the on-chip capacitance to ground is switched through the analog differential input voltage, resulting in proportional currents entering the $V_{IN(+)}$ input and leaving the $V_{IN(-)}$ input. These current transients occur at the leading edge of the internal clocks. They rapidly decay and do not inherently cause errors as the on-chip comparator is strobed at the end of the clock period.

Input Bypass Capacitors

Bypass capacitors at the inputs will average these charges and cause a DC current to flow through the output resistances of the analog signal sources. This charge pumping action is worse for continuous conversions with the $V_{IN(+)}$ input voltage

at full-scale. For a 640kHz clock frequency with the $V_{IN(+)}$ input at 5V, this DC current is at a maximum of approximately 5μA, Therefore, **bypass capacitors should not be used at the analog inputs or the $V_{REF}/2$** pin for high resistance sources (>1kΩ). If input bypass capacitors are necessary for noise filtering and high source resistance is desirable to minimize capacitor size, the effects of the voltage drop across this input resistance, due to the average value of the input current, can be compensated by a full-scale adjustment while the given source resistor and input bypass capacitor are both in place. This is possible because the average value of the input current is a precise linear function of the differential input voltage at a constant conversion rate.

Input Source Resistance

Large values of source resistance where an input bypass capacitor is not used will not cause errors since the input currents settle out prior to the comparison time. If a low-pass filter is required in the system, use a low-value series resistor (≤ 1kΩ) for a passive RC section or add an op amp RC active low-pass filter. For low-source-resistance applications (≤ 1kΩ), a 0.1μF bypass capacitor at the inputs will minimize EMI due to the series lead inductance of a long wire. A 100Ω series resistor can be used to isolate this capacitor (both the R and C are placed outside the feedback loop) from the output of an op amp, if used.

Stray Pickup

The leads to the analog inputs (pins 6 and 7) should be kept as short as possible to minimize stray signal pickup (EMI). Both EMI and undesired digital-clock coupling to these inputs can cause system errors. The source resistance for these inputs should, in general, be kept below 5kΩ. Larger values of source resistance can cause undesired signal pickup. Input bypass capacitors, placed from the analog inputs to ground, will eliminate this pickup but can create analog scale errors as these capacitors will average the transient input switching currents of the A/D (see Analog Input Current). This scale error depends on both a large source resistance and the use of an input bypass capacitor. This error can be compensated by a full-scale adjustment of the A/D (see Full-Scale Adjustment) with the source resistance and input bypass capacitor in place, and the desired conversion rate.

Reference Voltage Span Adjust

For maximum application flexibility, these A/Ds have been designed to accommodate a 5V, 2.5V or an adjusted voltage reference. This has been achieved in the design of the IC as shown in Figure 12.

Notice that the reference voltage for the IC is either $^1/_2$ of the voltage which is applied to the V+ supply pin, or is equal to the voltage which is externally forced at the $V_{REF}/2$ pin. This allows for a pseudo-ratiometric voltage reference using, for the V+ supply, a 5V reference voltage. Alternatively, a voltage less than 2.5V can be applied to the $V_{REF}/2$ input. The internal gain to the $V_{REF}/2$ input is 2 to allow this factor of 2 reduction in the reference voltage.

Such an adjusted reference voltage can accommodate a reduced span or dynamic voltage range of the analog input voltage. If the analog input voltage were to range from 0.5V to 3.5V, instead of 0V to 5V, the span would be 3V. With 0.5V applied to the $V_{IN(-)}$ pin to absorb the offset, the reference voltage can be made equal to $^1/_2$ of the 3V span or 1.5V. The A/D now will encode the $V_{IN(+)}$ signal from 0.5V to 3.5V with the 0.5V input corresponding to zero and the 3.5V input corresponding to full-scale. The full 8 bits of resolution are therefore applied over this reduced analog input voltage range. The requisite connections are shown in Figure 13. For expanded scale inputs, the circuits of Figures 14 and 15 can be used.

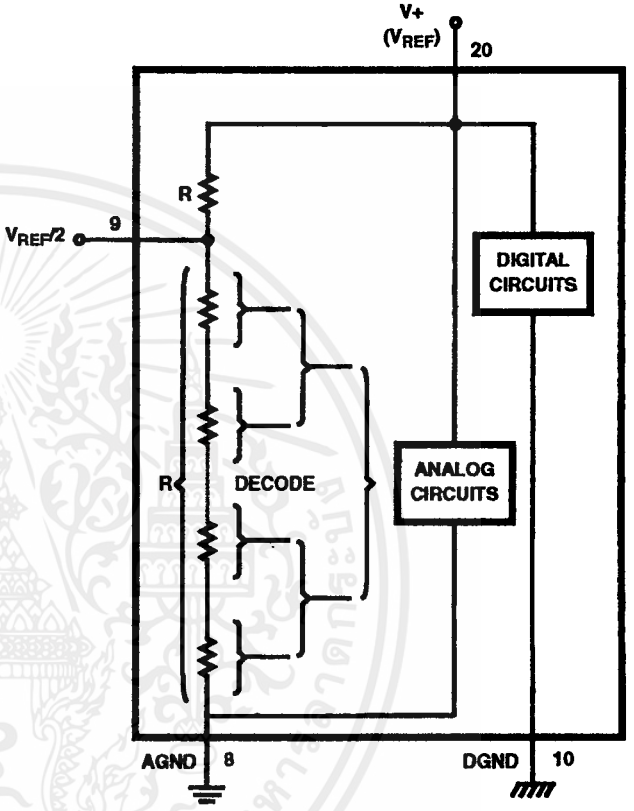


FIGURE 12. THE $V_{REFERENCE}$ DESIGN ON THE IC

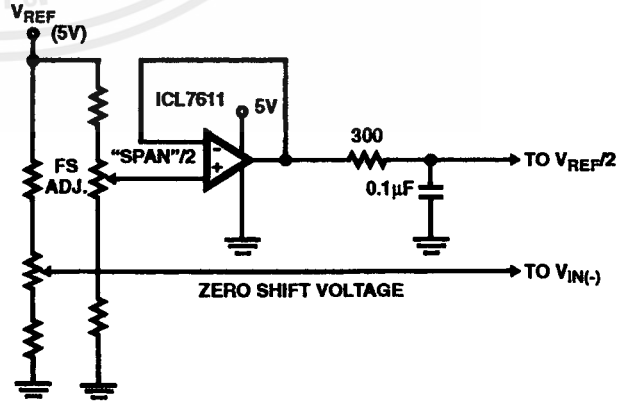


FIGURE 13. OFFSETTING THE ZERO OF THE ADC0802 AND PERFORMING AN INPUT RANGE (SPAN) ADJUSTMENT

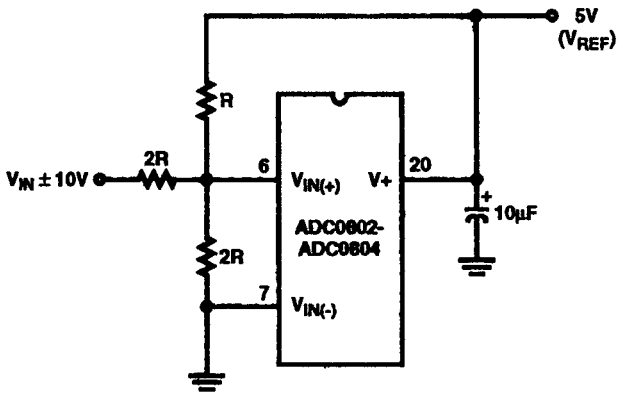


FIGURE 14. HANDLING ±10V ANALOG INPUT RANGE

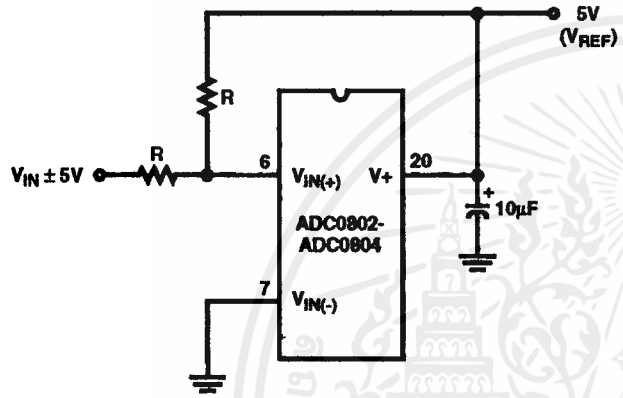


FIGURE 15. HANDLING ±5V ANALOG INPUT RANGE

Reference Accuracy Requirements

The converter can be operated in a pseudo-ratiometric mode or an absolute mode. In ratiometric converter applications, the magnitude of the reference voltage is a factor in both the output of the source transducer and the output of the A/D converter and therefore cancels out in the final digital output code. In absolute conversion applications, both the initial value and the temperature stability of the reference voltage are important accuracy factors in the operation of the A/D converter. For $V_{REF}/2$ voltages of 2.5V nominal value, initial errors of ±10mV will cause conversion errors of ±1 LSB due to the gain of 2 of the $V_{REF}/2$ input. In reduced span applications, the initial value and the stability of the $V_{REF}/2$ input voltage become even more important. For example, if the span is reduced to 2.5V, the analog input LSB voltage value is correspondingly reduced from 20mV (5V span) to 10mV and 1 LSB at the $V_{REF}/2$ input becomes 5mV. As can be seen, this reduces the allowed initial tolerance of the reference voltage and requires correspondingly less absolute change with temperature variations. Note that spans smaller than 2.5V place even tighter requirements on the initial accuracy and stability of the reference source.

In general, the reference voltage will require an initial adjustment. Errors due to an improper value of reference voltage appear as full-scale errors in the A/D transfer function. IC voltage regulators may be used for references if the ambient temperature changes are not excessive.

Zero Error

The zero of the A/D does not require adjustment. If the minimum analog input voltage value, $V_{IN(MIN)}$, is not ground, a zero offset can be done. The converter can be made to output 0000 0000 digital code for this minimum input voltage by biasing the A/D $V_{IN(-)}$ input at this $V_{IN(MIN)}$ value (see Applications section). This utilizes the differential mode operation of the A/D.

The zero error of the A/D converter relates to the location of the first riser of the transfer function and can be measured by grounding the $V_{IN(-)}$ input and applying a small magnitude positive voltage to the $V_{IN(+)}$ input. Zero error is the difference between the actual DC input voltage which is necessary to just cause an output digital code transition from 0000 0000 to 0000 0001 and the ideal $^1/_2$ LSB value ($^1/_2$ LSB = 9.8mV for $V_{REF}/2$ = 2.500V).

Full-Scale Adjust

The full-scale adjustment can be made by applying a differential input voltage which is $1^1/_2$ LSB down from the desired analog full-scale voltage range and then adjusting the magnitude of the $V_{REF}/2$ input (pin 9) for a digital output code which is just changing from 1111 1110 to 1111 1111. When offsetting the zero and using a span-adjusted $V_{REF}/2$ voltage, the full-scale adjustment is made by inputting V_{MIN} to the $V_{IN(-)}$ input of the A/D and applying a voltage to the $V_{IN(+)}$ input which is given by:

$$V_{IN(+)fSADJ} = V_{MAX} - 1.5\left[\frac{(V_{MAX} - V_{MIN})}{256}\right],$$

where:

V_{MAX} = the high end of the analog input range

and

V_{MIN} = the low end (the offset zero) of the analog range. (Both are ground referenced.)

Clocking Option

The clock for the A/D can be derived from an external source such as the CPU clock or an external RC network can be added to provide self-clocking. The CLK IN (pin 4) makes use of a Schmitt trigger as shown in Figure 16.

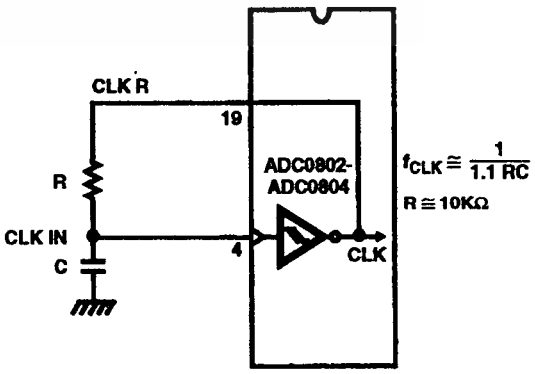


FIGURE 16. SELF-CLOCKING THE A/D

Heavy capacitive or DC loading of the CLK R pin should be avoided as this will disturb normal converter operation. Loads less than 50pF, such as driving up to 7 A/D converter clock inputs from a single CLK R pin of 1 converter, are allowed. For larger clock line loading, a CMOS or low power TTL buffer or PNP input logic should be used to minimize the loading on the CLK R pin (do not use a standard TTL buffer).

Restart During a Conversion

If the A/D is restarted ($\overline{CS}$ and $\overline{WR}$ go low and return high) during a conversion, the converter is reset and a new conversion is started. The output data latch is not updated if the conversion in progress is not completed. The data from the previous conversion remain in this latch.

Continuous Conversions

In this application, the $\overline{CS}$ input is grounded and the $\overline{WR}$ input is tied to the $\overline{INTR}$ output. This $\overline{WR}$ and $\overline{INTR}$ node should be momentarily forced to logic low following a power-up cycle to insure circuit operation. See Figure 17 for details.

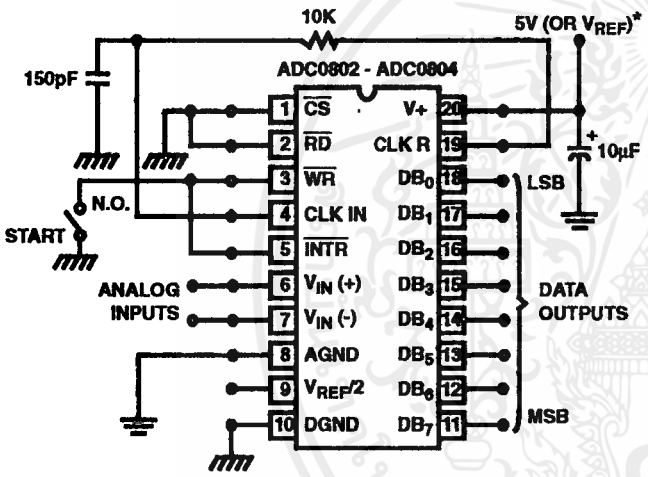


FIGURE 17. FREE-RUNNING CONNECTION

Driving the Data Bus

This CMOS A/D, like MOS microprocessors and memories, will require a bus driver when the total capacitance of the data bus gets large. Other circuitry, which is tied to the data bus, will add to the total capacitive loading, even in tri-state (high-impedance mode). Back plane bussing also greatly adds to the stray capacitance of the data bus.

There are some alternatives available to the designer to handle this problem. Basically, the capacitive loading of the data bus slows down the response time, even though DC specifications are still met. For systems operating with a relatively slow CPU clock frequency, more time is available in which to establish proper logic levels on the bus and therefore higher capacitive loads can be driven (see Typical Performance Curves).

At higher CPU clock frequencies time can be extended for I/O reads (and/or writes) by inserting wait states (8080) or using clock-extending circuits (6800).

Finally, if time is short and capacitive loading is high, external bus drivers must be used. These can be tri-state buffers (low power Schottky is recommended, such as the 74LS240 series) or special higher-drive-current products which are designed as bus drivers. High-current bipolar bus drivers with PNP inputs are recommended.

Power Supplies

Noise spikes on the V+ supply line can cause conversion errors as the comparator will respond to this noise. A low-inductance tantalum filter capacitor should be used close to the converter V+ pin, and values of 1µF or greater are recommended. If an unregulated voltage is available in the system, a separate 5V voltage regulator for the converter (and other analog circuitry) will greatly reduce digital noise on the V+ supply. An ICL7663 can be used to regulate such a supply from an input as low as 5.2V.

Wiring and Hook-Up Precautions

Standard digital wire-wrap sockets are not satisfactory for breadboarding with this A/D converter. Sockets on PC boards can be used. All logic signal wires and leads should be grouped and kept as far away as possible from the analog signal leads. Exposed leads to the analog inputs can cause undesired digital noise and hum pickup; therefore, shielded leads may be necessary in many applications.

A single-point analog ground should be used which is separate from the logic ground points. The power supply bypass capacitor and the self-clocking capacitor (if used) should both be returned to digital ground. Any $V_{REF}/2$ bypass capacitors, analog input filter capacitors, or input signal shielding should be returned to the analog ground point. A test for proper grounding is to measure the zero error of the A/D converter. Zero errors in excess of $^1/_4$ LSB can usually be traced to improper board layout and wiring (see Zero Error for measurement). Further information can be found in Application Note AN018.

Testing the A/D Converter

There are many degrees of complexity associated with testing an A/D converter. One of the simplest tests is to apply a known analog input voltage to the converter and use LEDs to display the resulting digital output code as shown in Figure 18.

For ease of testing, the $V_{REF}/2$ (pin 9) should be supplied with 2.560V and a V+ supply voltage of 5.12V should be used. This provides an LSB value of 20mV.

If a full-scale adjustment is to be made, an analog input voltage of 5.090V (5.120 - $1^1/_2$ LSB) should be applied to the $V_{IN(+)}$ pin with the $V_{IN(-)}$ pin grounded. The value of the $V_{REF}/2$ input voltage should be adjusted until the digital output code is just changing from 1111 1110 to 1111 1111. This value of $V_{REF}/2$ should then be used for all the tests.

The digital-output LED display can be decoded by dividing the 8 bits into 2 hex characters, one with the 4 most-significant bits (MS) and one with the 4 least-significant bits (LS). The output is then interpreted as a sum of fractions times the full-scale voltage:

$$V_{OUT} = \left(\frac{MS}{16} + \frac{LS}{256}\right)(5.12)V.$$

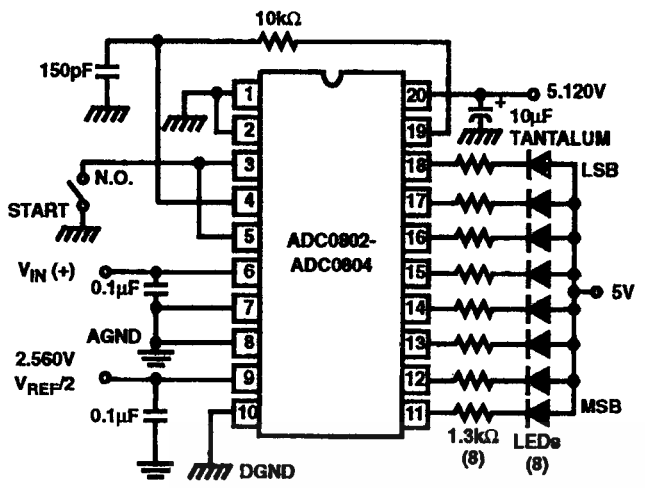


FIGURE 18. BASIC TESTER FOR THE A/D

For example, for an output LED display of 1011 0110, the MS character is hex B (decimal 11) and the LS character is hex (and decimal) 6, so

$$V_{OUT} = \left(\frac{11}{16} + \frac{6}{256}\right)(5.12) = 3.64V$$

Figures 19 and 20 show more sophisticated test circuits.

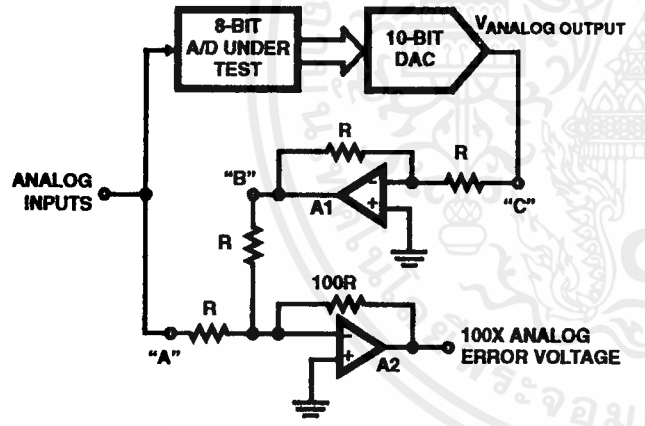


FIGURE 19. A/D TESTER WITH ANALOG ERROR OUTPUT. THIS CIRCUIT CAN BE USED TO GENERATE "ERROR PLOTS" OF FIGURE 11.

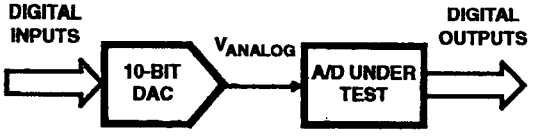


FIGURE 20. BASIC "DIGITAL" A/D TESTER

Typical Applications

Interfacing 8080/85 or Z-80 Microprocessors

This converter has been designed to directly interface with 8080/85 or Z-80 Microprocessors. The tri-state output capability of the A/D eliminates the need for a peripheral interface device, although address decoding is still required to generate the appropriate $\overline{CS}$ for the converter. The A/D can be mapped into memory space (using standard memory-address decoding for $\overline{CS}$ and the $\overline{MEMR}$ and $\overline{MEMW}$ strobes) or it can be controlled as an I/O device by using the $\overline{I/OR}$ and $\overline{I/OW}$ strobes and decoding the address bits A0 → A7 (or address bits A8 → A15, since they will contain the same 8-bit address information) to obtain the $\overline{CS}$ input. Using the I/O space provides 256 additional addresses and may allow a simpler 8-bit address decoder, but the data can only be input to the accumulator. To make use of the additional memory reference instructions, the A/D should be mapped into memory space. See AN020 for more discussion of memory-mapped vs I/O-mapped interfaces. An example of an A/D in I/O space is shown in Figure 21.

The standard control-bus signals of the 8080 ($\overline{CS}$, $\overline{RD}$ and $\overline{WR}$) can be directly wired to the digital control inputs of the A/D, since the bus timing requirements, to allow both starting the converter, and outputting the data onto the data bus, are met. A bus driver should be used for larger microprocessor systems where the data bus leaves the PC board and/or must drive capacitive loads larger than 100pF.

It is useful to note that in systems where the A/D converter is 1 of 8 or fewer I/O-mapped devices, no address-decoding circuitry is necessary. Each of the 8 address bits (A0 to A7) can be directly used as $\overline{CS}$ inputs, one for each I/O device.

Interfacing the Z-80 and 8085

The Z-80 and 8085 control buses are slightly different from that of the 8080. General $\overline{RD}$ and $\overline{WR}$ strobes are provided and separate memory request, $\overline{MREQ}$, and I/O request, $\overline{IORQ}$, signals have to be combined with the generalized strobes to provide the appropriate signals. An advantage of operating the A/D in I/O space with the Z-80 is that the CPU will automatically insert one wait state (the $\overline{RD}$ and $\overline{WR}$ strobes are extended one clock period) to allow more time for the I/O devices to respond. Logic to map the A/D in I/O space is shown in Figure 22. By using $\overline{MREQ}$ in place of $\overline{IORQ}$, a memory-mapped configuration results.

Additional I/O advantages exist as software DMA routines are available and use can be made of the output data transfer which exists on the upper 8 address lines (A8 to A15) during I/O input instructions. For example, MUX channel selection for the A/D can be accomplished with this operating mode.

The 8085 also provides a generalized $\overline{RD}$ and $\overline{WR}$ strobe, with an IO/$\overline{M}$ line to distinguish I/O and memory requests. The circuit of Figure 22 can again be used, with IO/$\overline{M}$ in place of $\overline{IORQ}$ for a memory-mapped interface, and an extra inverter (or the logic equivalent) to provide $\overline{IO}$/M for an I/O-mapped connection.

Interfacing 6800 Microprocessor Derivatives (6502, etc.)

The control bus for the 6800 microprocessor derivatives does not use the $\overline{RD}$ and $\overline{WR}$ strobe signals. Instead it employs a single R/$\overline{W}$ line and additional timing, if needed, can be derived from the $\phi 2$ clock. All I/O devices are memory-mapped in the 6800 system, and a special signal, VMA, indicates that the current address is valid. Figure 23 shows an interface schematic where the A/D is memory-mapped in the 6800 system. For simplicity, the $\overline{CS}$ decoding is shown using $^1/_2$ DM8092. Note that in many 6800 systems, an already decoded $\overline{4/5}$ line is brought out to the common bus at pin 21. This can be tied directly to the $\overline{CS}$ pin of the A/D, provided that no other devices are addressed at HEX ADDR: 4XXX or 5XXX.

In Figure 24 the ADC0802 series is interfaced to the MC6800 microprocessor through (the arbitrarily chosen) Port B of the MC6820 or MC6821 Peripheral Interface Adapter (PIA). Here the $\overline{CS}$ pin of the A/D is grounded since the PIA is already memory-mapped in the MC6800 system and no $\overline{CS}$ decoding is necessary. Also notice that the A/D output data lines are connected to the microprocessor bus under program control through the PIA and therefore the A/D $\overline{RD}$ pin can be grounded.

Application Notes

Some applications bulletins that may be found useful are listed here:

AN016 "Selecting A/D Converters"

AN018 "Do's and Don'ts of Applying A/D Converters"

AN020 "A Cookbook Approach to High Speed Data Acquisition and Microprocessor Interfacing"

AN030 "The ICL7104 - A Binary Output A/D Converter for Microprocessors"

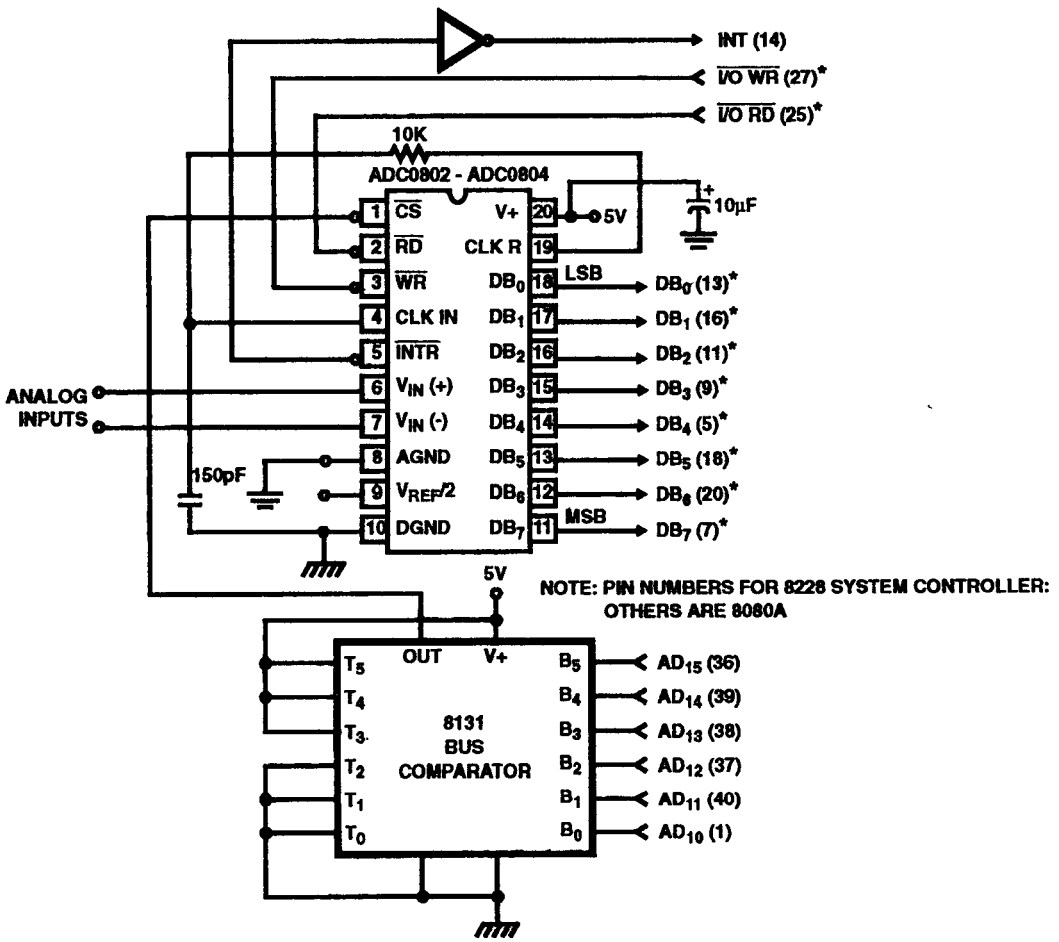


FIGURE 21. ADC0802 TO 8080A CPU INTERFACE

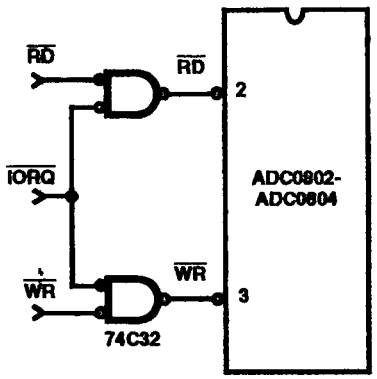
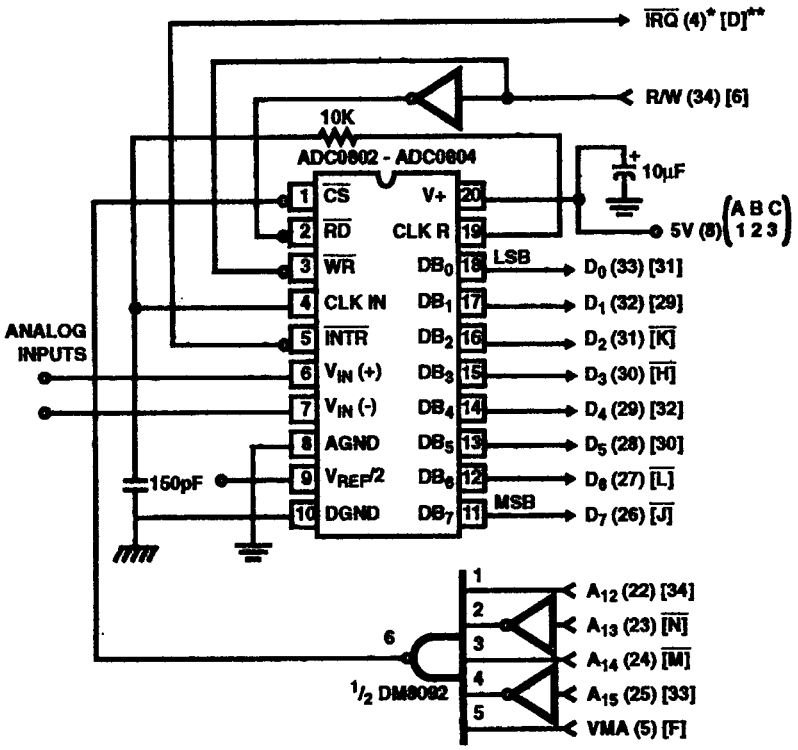


FIGURE 22. MAPPING THE A/D AS AN I/O DEVICE FOR USE WITH THE Z-80 CPU



* NUMBERS IN PARENTHESES REFER TO MC6800 CPU PINOUT.
** NUMBERS OR LETTERS IN BRACKETS REFER TO STANDARD MC6800 SYSTEM COMMON BUS CODE.

FIGURE 23. ADC0802 TO MC6800 CPU INTERFACE

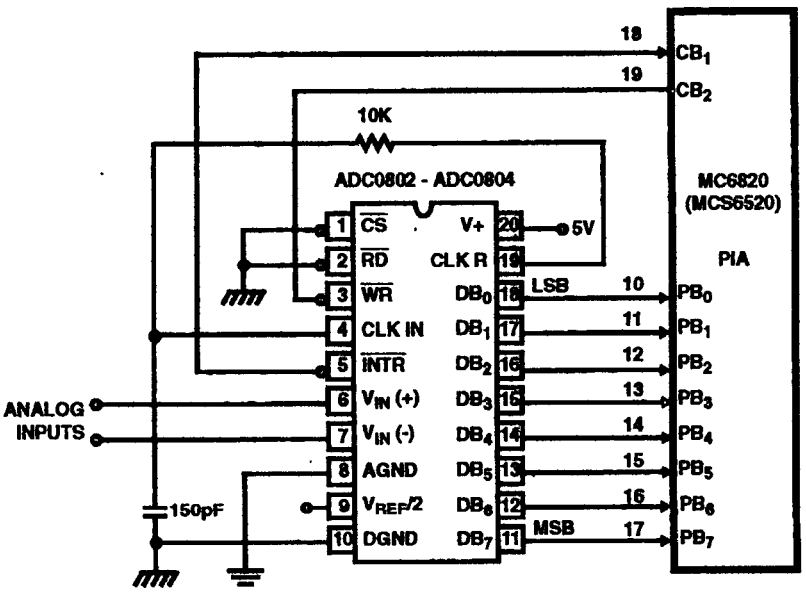


FIGURE 24. ADC0802 TO MC6820 PIA INTERFACE

Die Characteristics

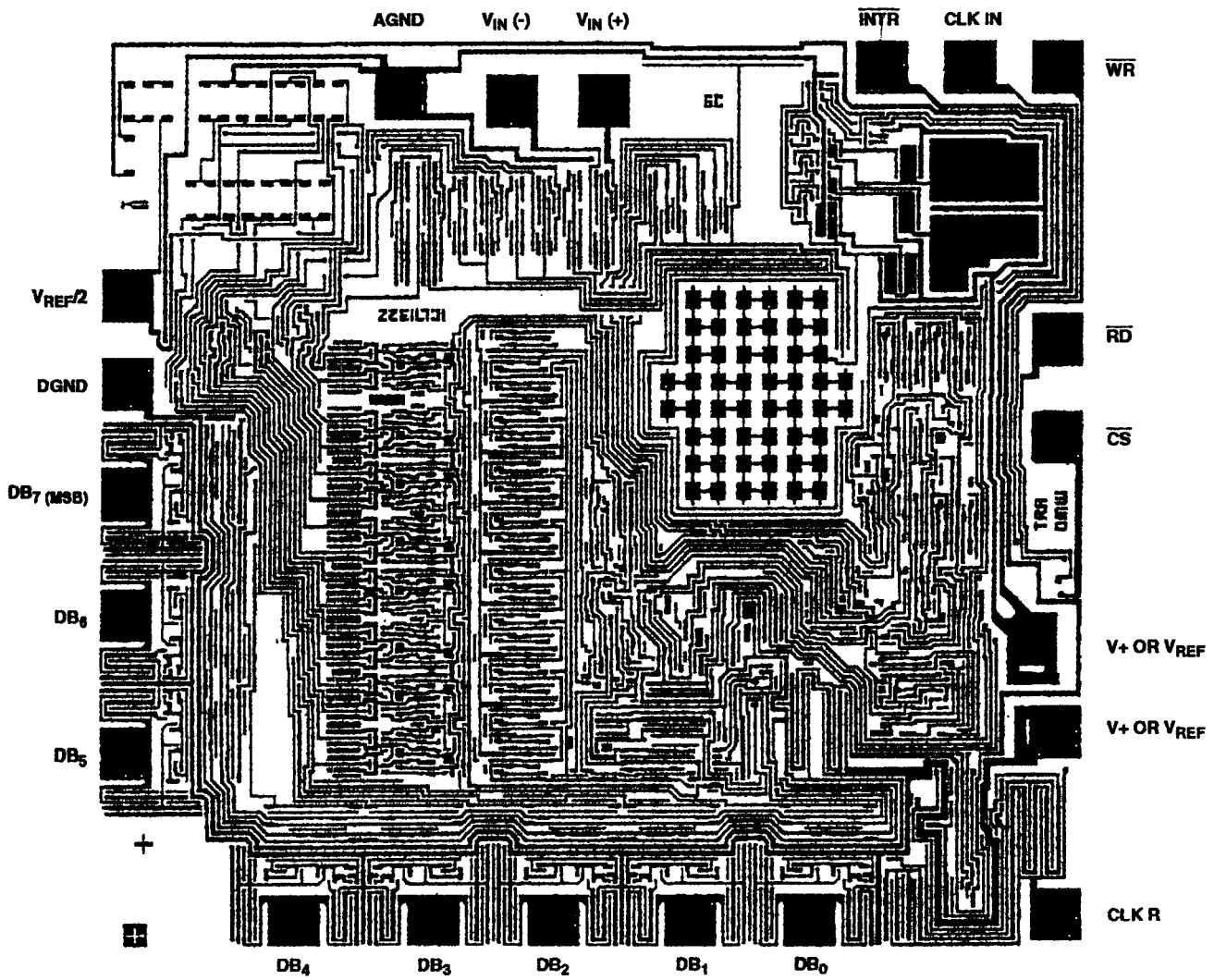
DIE DIMENSIONS:
(101 x 93mils) x 525 x 25µm

METALLIZATION:
Type: Al
Thickness: 10kÅ ± 1kÅ

GLASSIVATION:
Type: Nitride over Silox
Nitride Thickness: 8kÅ
Silox Thickness: 7kÅ

Metallization Mask Layout

ADC0802, ADC0803, ADC0804



กิตติกรรมประกาศ

- ขอขอบคุณอาจารย์นิภา ลีลารุจิ และอาจารย์ณรงค์ เหมกรณ์ ในความช่วยเหลือเอื้อเฟื้ออุปกรณ์ในการทำงาน และคำแนะนำปรึกษาต่าง ๆ
- เพื่อน ๆ ทุกคน ขอขอบคุณที่ให้กำลังใจ
- ขอขอบคุณพี่ ๆ ที่อาคารสำนักวิจัย และบริการคอมพิวเตอร์ที่ช่วยอำนวยความสะดวกในการทดสอบสายอากาศ
- ขอขอบคุณเป็นพิเศษสำหรับคุณวุฒิพล พงษ์พัฒนกิจโชติ ผู้ให้คำแนะนำ และคำปรึกษาอันมีค่ายิ่งในการเขียนโปรแกรมภาษาซี

หนังสืออ้างอิง

[1] บุญชัด เนติศักดิ์, "ทฤษฎี และปฏิบัติเครื่องรับวิทยุ AM/FM," ซีเอ็ดยูเคชั่น, กรุงเทพฯ, 2540

[2] ธานินทร์ ถาวรศาสนวงศ์, ทินกร ดุ๊ก, "การอินเตอร์เฟส IBM/PC," ฟิสิกส์เซ็นเตอร์, กรุงเทพฯ, 2536

[3] Takashi Kenjo, "Stepping motor and their microprocessor controls," Oxford University Press, Oxford, 1986

[4] Constantine A. Balanis, "Antenna Theory Analysis and Design," John Wiley & Sons, INC., New York, 1997