เครื่องทดสอบแพทเทิร์นของสายอากาศด้วยคอมพิวเตอร์

ANTENNA PATTERN TESTER WITH PERSONAL COMPUTER

โดย
นายนุวัฒน์ ศิลาพัฒน์
นายวีระ อีชะโรจน์
นายสุเทพ รุ่งเรือง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2535

ปริญญานิพนธ์ปีการศึกษา 2535
สาขาวิชา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม
ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง เครื่องทดสอบแพทเทิร์นของสายอากาศด้วยคอมพิวเตอร์
(Antenna Pattern Tester with Personal Computer)

รหัส Project : PPT 03

ผู้จัดทำ

1. นายนุวัฒน์	ศิลาพัฒน์	34162209
2. นายวีระ	อิชะโรจน์	34162227
3. นายสุเทพ	รุ่งเรือง	34162236

..... อาจารย์ที่ปรึกษา
(ผศ. พิพัฒน์ เลาหสงคราม)

บทคัดย่อ

ในปัจจุบันเครื่องมือที่ใช้วัดและทดสอบสายอากาศมีราคาสูงมาก ทำให้การใช้งานจำกัดอยู่เฉพาะในองค์กร,บริษัทหรือ สถานการศึกษาที่ใหญ่ๆเท่านั้น กอรปกับเครื่องมือทดสอบบางประเภทมีขีดความสามารถจำกัดหรือฟังก์ชั่นการใช้งานต่างๆมีจำกัด ทำให้ต้องนำเอาเครื่องมือหลายๆประเภทมาประกอบกันจึงจะทำการทดสอบได้ โครงการนี้เป็นการนำเอาไมโครคอมพิวเตอร์มาประยุกต์ใช้งานร่วมกับ field strength มา plot pattern บนไมโครคอมพิวเตอร์ หรือออกทาง printerซึ่งสามารถนำไปประยุกต์ใช้งานในการหา pattern ของสายอากาศ วัดอัตราการขยายของสายอากาศ นอกจากนี้ ยังสามารถนำไปประยุกต์ใช้งานอื่นๆได้อีกเช่น ทดสอบความเข้มแต่ละ cell site ที่จุดต่างๆของวิทยุโทรศัพท์ระบบ เซลลูล่าร์ เป็นต้น

สายอากาศมีอยู่หลายชนิด ดังนั้นการเลือกใช้สายอากาศจึงมีความสำคัญมาก เพราะสายอากาศ คือ อุปกรณ์ที่มีหน้าที่รับ-ส่งคลื่นวิทยุการใช้สายอากาศที่มีอัตราการขยายสูงเพื่อให้ได้กำลังคลื่นสูงขึ้น คือกว่าการใช้แอมปลิไฟเออร์ที่ใช้วงจรอิเล็กทรอนิคส์ทั่วไป เพราะแอมปลิไฟเออร์ส่วนใหญ่จะสร้างสัญญานรบกวนออกมาด้วยจึงทำให้คุณภาพของสัญญานที่ได้ลดลง

เราสามารถวิเคราะห์คุณสมบัติของสายอากาศได้โดยดูแพทเทอร์นของการกระจายคลื่นนั้น แพทเทอร์นของการกระจายคลื่น คือ รูปแพทเทอร์นที่แสดงความสามารถในการกระจาย คลื่นออกไปหรือในการรับคลื่นเข้ามาของสายอากาศ จึงทำให้ทราบค่าบีมวิดท์ซึ่งจะเกี่ยวข้องกับค่าอัตราขยายโดยตรงในลักษณะที่ ถ้าค่าบีมวิดท์แคบ อัตราการขยายของสายอากาศก็จะสูง

ABTRACT

At present, instrument to measure and test an antenna cost very high. Thus, it was to limit using only in some organization, company or college. Consisted of some tester is limited their abilities or function. It caused us to have various tester using a long for a test.

This project bring personnel computer to apply with field strength meter. run, plot pattern on microcomputer or print out to printer which is able to take it on looking for pattern of antenna measuring antenna gain. Furthermore, we can bring it to apply with another job such as intensity testing of each cell site area of mobile telephone on cellular system etc.

There are plenty of antenna, so choosing of use antenna is very important. This is because antenna is the equipment for receive and transmit radio wave. Using of high antenna gain increasing better than using general amplifier of electronics. Because many amplifier produce noise and reduced efficiency of signal.

We can analyze the characteristics of an antenna by seeing pattern of radiation. Pattern of radiation is a pattern to show ability in antenna radiation out or in. So, it caused value of beam width which concern gain of antenna directly value of beam width is shallow, gain of antenna will grow high.

คำนำ

ในปัจจุบันนี้ถือว่าเป็นยุคของการสื่อสารและโทรคมนาคม การติดต่อสื่อสารโดยใช้คลื่นวิทยุคมนาคมนั้นนับว่าสะดวกและรวดเร็วมาก เพราะ เป็นการติดต่อโดยใช้คลื่นแม่เหล็กไฟฟ้าอุปกรณ์ที่มีหน้าที่รับ-ส่งคลื่นวิทยุนั้นก็คือ สายอากาศนั่นเอง ดังนั้นการเลือกใช้สายอากาศจึงมีความ สำคัญมาก

เครื่องทดสอบแพทเทอร์นของสายอากาศด้วยคอมพิวเตอร์ ถูกสร้างขึ้นมาเพื่อช่วยในการวิเคราะห์คุณสมบัติของสายอากาศ และวิเคราะห์ความเข้มของสนามไฟฟ้าเชิงมุม ซึ่งเป็นการนำเอาไมโครคอมพิวเตอร์มาประยุกต์ใช้งานร่วมกับ FIELD STRENGTH METER ทำให้สามารถวัดอัตราขยายของสายอากาศ และวาดแพทเทอร์นเชิงมุมเพื่อวิเคราะห์คุณสมบัติต่างๆ จึงหวังว่าโครงงานชุดนี้คงจะเป็นประโยชน์ต่อทุกท่านที่สนใจ นำไปใช้ในงานที่เกี่ยวข้องกับการสื่อสารโดยใช้วิทยุคมนาคมต่างๆ ได้ดี และยังเป็นการช่วยพัฒนาเทคโนโลยีของประเทศไทยเราด้วยอีกทางหนึ่ง

นายนุวัฒน์ ศิลาพัฒน์

นายวีระ อิชะโรจน์

นายสุเทพ รุ่งเรือง

สารบัญ

บทที่ 1 หลักการเบื้องต้นของการทดสอบการแพร่กระจายคลื่นของสายอากาศ.....1
บทที่ 2 ความรู้ทั่วๆไปเกี่ยวกับ อีจีเอและวีจีเอ.....4
2.1 ลักษณะโดยทั่วๆไปของอีจีเอ/วีจีเอ.....4
2.2 หน่วยความจำแสดงผล.....4
2.3 โหมดตัวอักษร.....4
2.4 โหมดกราฟฟิก.....5
2.5 รีจิสเตอร์ควบคุม.....5
2.6 โหมดการแสดงผล.....6
2.7 ความละเอียดของภาพ.....6
2.8 ความละเอียดของสี.....6
2.9 พาแล็ตส์และรีจิสเตอร์สี.....8
2.10 การประมวลฟผลทางตัวอักษร.....10
2.11 การประมวลผลทางกราฟฟิก.....14
บทที่ 3 การทำงานของโปรแกรมต่างๆ.....22
3.1 ฟังก์ชัน Inititialize.....22
3.2 การเก็บภาพบางส่วนจากจอมอนิเตอร์.....23
3.3 การเก็บภาพกราฟฟิกและการดึงข้อมูล.....24
3.4 การใช้เครื่องพิมพ์ EPSON DOT- MATRIX.....28
3.5 โปรแกรม PRINT_GRAPH.....31
3.6 มาตรฐาน RS-232.....36
3.7 การเตรียมสถานะเริ่มต้นของพอร์ต.....37
3.8 การส่งข้อมูลผ่านพอร์ตอนุกรม 1 ไบต์.....38
3.9 การตรวจสอบสถานะของพอร์ตอนุกรม.....39
3.10 การรับข้อมูลผ่านพอร์ตอนุกรม 1 ไบต์.....40
3.11 ฟังก์ชัน Xmitt.....41
3.12 การสร้างภาพ Background.....42
3.13 การหาตำแหน่ง X,Y เพื่อทำการพล็อตจุด.....43
3.14 ฟังก์ชัน Automatic Test.....46
3.15 ฟังก์ชัน Test.....49
3.16 Main Program.....52
3.17 โปรแกรม Serial Port Communication and Control Rotor.....54
บทที่ 4
การใช้โปรแกรม Ant Test.....55
4.1 การเข้าสู่โปรแกรม Ant Test.....55
4.2 คำสั่ง Load.....56
4.3 คำสั่ง Save.....56
4.4 คำสั่ง Print.....57

4.5 คำสั่ง Auto Test....56
4.6 คำสั่ง Test....58
4.7 คำสั่ง Directory....59
4.8 คำสั่ง Help....59
4.9 การติดตั้งอุปกรณ์....60
4.10 ลำดับขั้นการทดสอบ....60
บทที่ 5
สรุปและวิจารณ์....61
ภาคผนวก
ภาคผนวก ก. 63
โปรแกรม Ant Test....64
โปรแกรม Serial Port Communication & Control Rotor....77
โปรแกรม Gprint . I 80
Circuit....83
ภาคผนวก ข. 91
Rotor Specifications....92
ภาคผนวก ค. 96
Data Sheet....97
Block Diagram Field Strenght Meter....122
กิตติกรรมประกาศ....124
หนังสืออ้างอิง....125

บทที่ 1

หลักการเบื้องต้นของการทดสอบการแพร่กระจายคลื่นของสายอากาศ

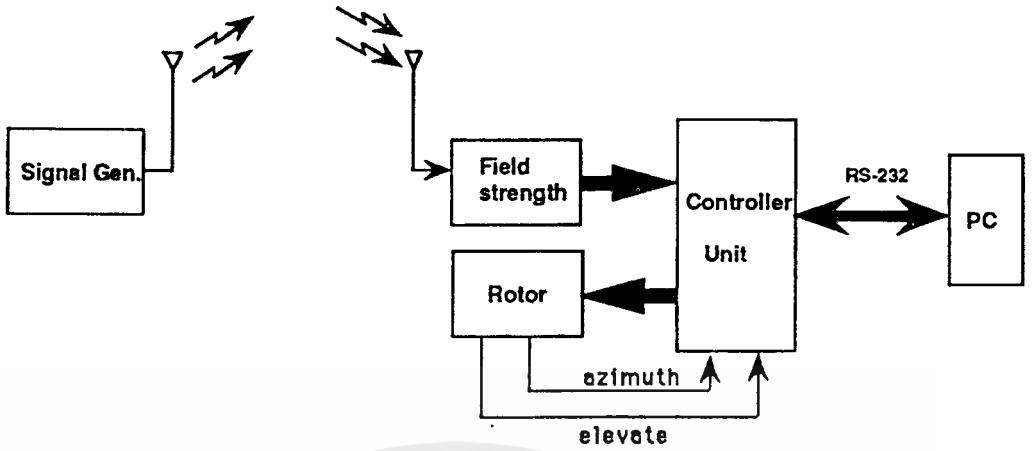
จาก Block Diagram รูปที่ 1 Signal Generator เป็นเครื่องมือกำเนิดความถี่ ตาม frequency response ของสายอากาศที่จะต้องการทดสอบ และ field strength meter เป็นเครื่องมือวัดความเข้มของสนามแม่เหล็กไฟฟ้าที่รับเข้ามาทางสายอากาศโดยวัดเป็นค่า dB เมื่อหมุนทิศทางของสายอากาศไป 360 องศา ก็จะได้แพทเทิร์นการแพร่กระจายคลื่นของสายอากาศเป็นแพทเทิร์นเชิงมุมแนวนอน

ค่าความเข้มของสนามไฟฟ้าซึ่งเป็น Analog Signal จะถูกแปลงเป็นสัญญาน Digital โดยชุด Controller Unit แล้วส่งข้อมูลไปที่ PersonnelComputerเพื่อทำการพล็อตแพทเทิร์นตามค่า dB ที่มุมต่างๆ การควบคุมทิศทางของสายอากาศ จะถูกควบคุมโดย PC โดยจะส่งรหัสคำสั่งผ่าน RS-232 Controller Unit จะรับรหัสคำสั่งผ่านทาง Resial Port แล้วทำการแปลคำสั่งแล้วปฏิบัติตาม ส่วน Controller จะควบคุม Rotor ให้หมุนไปตามทิศทางที่ต้องการ

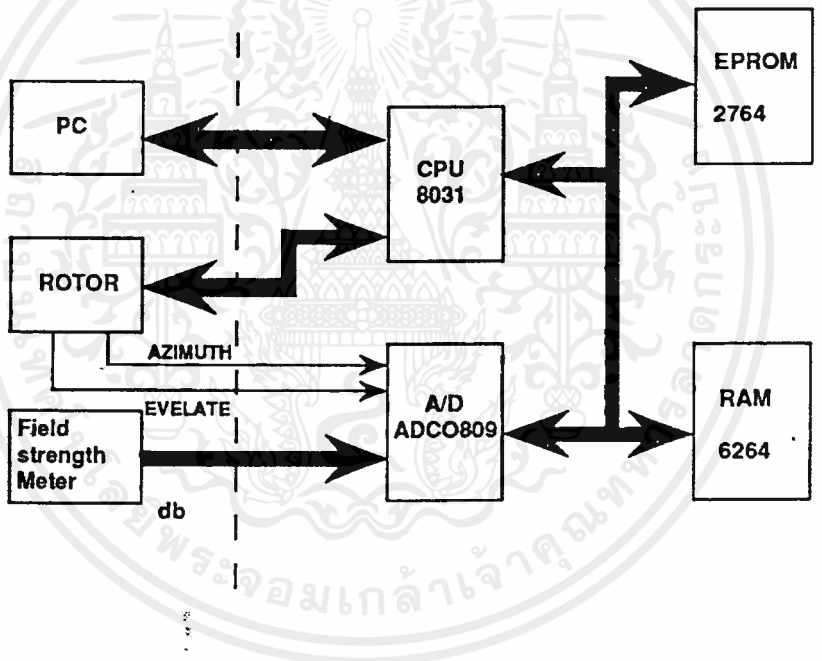
Controller Unit

Hardware

ใช้ไมโครโปรเซสเซอร์ตระกูล MCS - 51 เบอร์ 8031 ใช้ EPROM เบอร์ 2764 และ RAM เบอร์ 6264 โดยได้ออกแบบ Memory Map ให้สามารถทำการขยายได้โดยใช้ Jumper RAM สามารถเลือกขนาดได้ตั้งแต่ 8 k , 16 k , 32 k และ ROM เลือกได้ 8 k, 16 k , 32 k , 64 k A/D ใช้เบอร์ ADC 0809 8 channel Analog input multiplex วงจรเป็นดังรูป สำหรับ PCB ได้ออกแบบโดยใช้โปรแกรม EASYPC



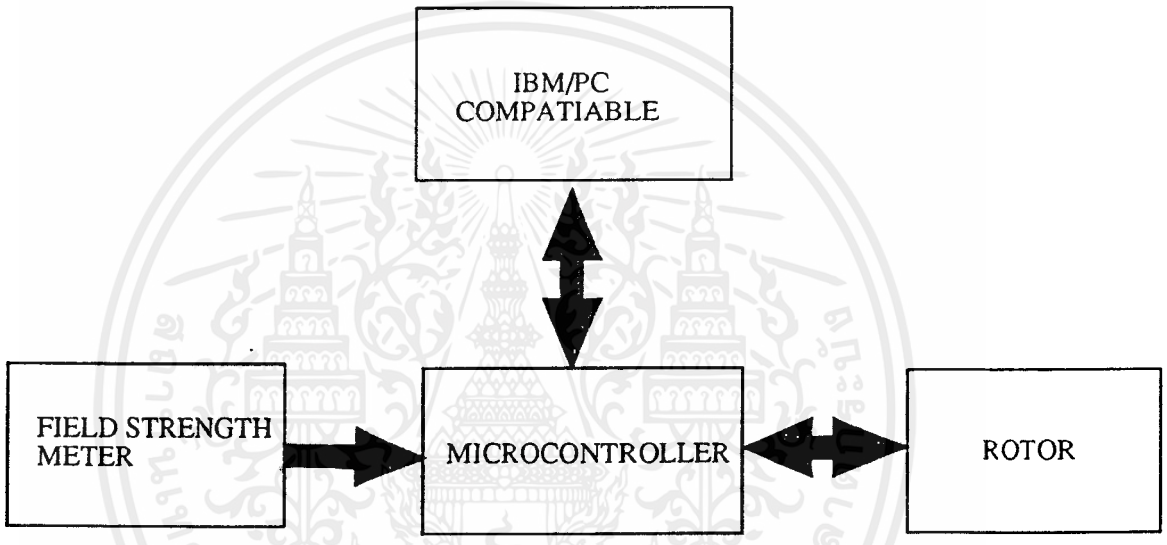
รูป 1. หลักการเบื้องต้นของ Antenna Pattern Tester



รูป 2. Block Diagram Controller Unit

Block Diagram ของระบบ

ชุด INTERFACE ชุดนี้ใช้อุปกรณ์ MICROCONTROLLER เป็นตัวเชื่อมโยงระหว่างคอมพิวเตอร์กับ ROTOR และ FIELD STRENGTH METER ตามบล็อกไดอะแกรมรูปที่ 1



รูปแสดง BLOCK DIAGRAM

อธิบาย

IBM/PC ใช้เป็นตัวประมวลผลเก็บข้อมูล พล็อตแพทเทิร์นการแพร่กระจายคลื่นและควบคุมระบบทั้งหมด

MICROCONTROLLER เป็นชุดที่ออกแบบสร้างขึ้นมาเพื่อเป็นตัวเชื่อมโยงระหว่างคอมพิวเตอร์กับ ROTOR และ FIELD STRENGTH METER

ROTOR คือ ชุดควบคุมการหมุนหาทิศทาง การรับสัญญาณของสายอากาศ

FIELD STRENGTH METER คือ เครื่องมือวัดความเข้มของสนามไฟฟ้าที่รับเข้ามาทางสายอากาศ

บทที่ 2

ความรู้ทั่วๆไปเกี่ยวกับอีจีเอ/วีจีเอ

2.1 ลักษณะทั่วๆไปของEGA/VGA

ในปัจจุบันนี้ อีจีเอ และ วีจีเอ ได้เริ่มเป็นที่นิยมกันมากขึ้น ทั้งนี้เนื่องจากว่ามีราคาไม่สูงนัก, มีความเร็วสูง, มีหน่วยความจำความจุสูง, กะทัดรัดและง่ายต่อการรักษา

คำว่า "EGA/VGA boards" จะหมายความถึงไบออส (BIOS) ของมันด้วย ซึ่งทำให้สามารถใช้ได้กับซอฟท์แวร์ (software) อื่น ๆ ได้ ทั้งนี้จะรวมไปถึง เวิร์ดโปรเซสเซอร์ (word processors), การจัดการฐานข้อมูล (data base managers), สเปรดชีท (spreadsheet) และ ยูทิลิตี้ (UTILITY) อื่น ๆ

ในปัจจุบันนี้ การ์ดอีจีเอ/วีจีเอ (EGA/VGA cards) ส่วนใหญ่จะมีมาตรฐานต่างๆดีขึ้นมามาก เช่น ความละเอียด, สี, หน่วยความจำ, ตัวอักษร (character fonts) ฯลฯ แต่มีข้อเสียคือ ไม่ได้ถูกควบคุมโดยมาตรฐานใดๆ ทำให้ต้องใช้ตัวขับ (drivers) ในการใช้ลักษณะเด่นที่เพิ่มขึ้นเหล่านี้กับซอฟท์แวร์ ที่ได้กล่าวมาแล้ว ซึ่งก่อให้เกิดความไม่สะดวกแก่ผู้ใช้ในกรณีที่ต้องการใช้ซอฟท์แวร์ หลายๆตัว เพราะว่า แต่ละตัวอาจไม่ใช้ตัวขับตัวเดียวกันก็ได้

2.2 หน่วยความจำแสดงผล (display memory)

ใน อีจีเอ/วีจีเอ นั้น ผู้ใช้สามารถอ่านหรือเขียนไปยัง หน่วยความจำแสดงผล โดยไม่ต้องรอช่วงเวลาของการ รีเทรส แนวนอนหรือแนวดิ่ง ซึ่งจะทำให้กระบวนการทาง กราฟิกส์ (graphics) ของ อีจีเอ/วีจีเอ มีความเร็วสูงกว่าอแด็ปเตอร์แสดงผล (display adapters) อื่นๆ นอกจากนี้ หน่วยความจำยังถูกจัดในลักษณะตรงไปตรงมา ซึ่งหากผู้ใช้เข้าใจในการอ้างแอดเดรส (addressing) แล้ว ก็จะสามารถวาดเส้นและรูปร่างๆไปยังจอได้โดยตรง

หน่วยความจำแสดงผล ประกอบด้วย หน่วยความจำ 256 เคไบต์ (Kbytes) ติดต่อกับโปรเซสเซอร์ (processor) ผ่านทาง 8 บิทบัสข้อมูล (bit data bus) สามารถแสดงความละเอียดได้หลายอย่าง ขึ้นอยู่กับ โหมดการแสดงผล (display mode) ที่ใช้ และสามารถใช้แพคฟอร์แมท (packed format) หรือ บิทเพลนฟอร์แมท (bit-plane format) ก็ได้ ขึ้นอยู่กับ โหมดการแสดงผล ที่ใช้งานในขณะนั้น

2.3 โหมดตัวอักษร (Alphanumeric Modes)

หน่วยความจำแสดงผล สามารถใช้คอมแพททิเบิล (compatible) กับ เอ็มดีเอ (MDA) และ ซีจีเอ (CGA) ได้ ตัวอักษรแต่ละตัวถูกแทนใน หน่วยความจำแสดงผล เพียง 2 ไบต์ ไบต์แรกคือ รหัสแอสกี (ASCII character code) ซึ่งเป็นตัวบอกให้ทราบว่า ตัวอักษร (character) ตัวใดที่ต้องการจะแสดง ไบต์ที่สองคือ แอททริบิวต์ (attribute) ซึ่งเป็นตัวบอกสีของ ตัวอักษร

ในหน่วยความจำ ไบต์ที่แสดงตัวอักษร และ แอททริบิวต์ ถูกเก็บในลักษณะเรียงติดกันไป การอ้างแอดเดรส ที่เจาะจงลงไป จะทำให้ ตัวอักษร และ แอททริบิวต์ ถูกเคลื่อนย้ายไปยัง

หน่วยความจำแสดงผล ในลักษณะที่ยังเรียงติดกันด้วย ตัวอักษร สามารถแสดงโดยโมโนโครม (monochrome) หรือ แอททริบิวต์สี ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แอททริบิวต์ของตัวอักษร

โหมด	แอททริบิวต์
Mono	Normal, Intensified, Reversed, Blinking, Underlined
Color	16 foreground, 16 background colors
Color	16 foreground, 8 background, blinking
Color	8 foreground, 8 background, 512 characters, blinking

2.4 โหมดกราฟิคส์ (Graphics Modes)

เนื่องจาก หน่วยความจำแสดงผล ถูกจัดในลักษณะที่บางโหมดสามารถใช้ร่วมกับซีจีเอ (CGA) ได้ ดังนั้นจึงทำให้สามารถใช้กราฟิคส์ของซีจีเอได้ด้วย โหมดกราฟิกส์ แบ่งได้เป็น 2 โหมดใหญ่ๆ คือ โหมดโมโนโครม และ โหมดสี

ในโหมดโมโนโครม มี 2 แบบ คือ 2 บิทต่อ 1 จุดภาพ และ 1 บิทต่อ 1 จุดภาพ แบบแรกใช้กับโปรแกรมในเฮอร์คิวลิส (Hercules) แบบหลังเรียกว่า โหมด 2 สี (two-color modes) เพราะว่าโหมดนี้สามารถแสดงได้เพียง 2 สี จึงยังถูกจัดให้อยู่ในโหมดโมโนโครม สำหรับ 2 สีที่แสดงนี้ไม่จำเป็นต้องเป็นสีขาวและสีดำ

ในโหมดสี แบ่งเป็น 3 แบบ คือ 2,4,และ 8 บิทต่อ 1 จุดภาพ ในโหมด 2 บิทต่อ 1จุดภาพ เลียนแบบซีจีเอ แสดงได้ 4สีต่อภาพในเวลาเดียวกัน ในโหมด 4 บิทต่อ 1จุดภาพ แสดงได้ 16 สีพร้อมกัน และโหมด 8 บิทต่อ 1จุดภาพ แสดงได้ 256 สีพร้อมกัน

2.5 รีจิสเตอร์ควบคุม (Control Registers)

รีจิสเตอร์ ถูกแบ่งออกเป็น 5 กลุ่ม ดัง ตาราง ที่ 2.2

รีจิสเตอร์ แต่ละตัวมีขนาด 1 ไบต์ รีจิสเตอร์บางตัวถูกแบ่งย่อยเป็นฟิล์ด (field) ดังนั้น บ่อยครั้งที่บิทแต่ละบิทภายใน รีจิสเตอร์ เดียวกันจะควบคุมฟังก์ชันที่ต่างกัน และบางฟังก์ชันก็ถูกควบคุมโดย รีจิสเตอร์ ที่มากกว่า 1 ตัว

รีจิสเตอร์ ถูกเข้าถึงโดยผ่านการแอดเดรสไอโอพอร์ท (I/O Port) แต่ละกลุ่มของรีจิสเตอร์ ที่แสดงในตารางที่2.2 ถูกแทนด้วย รีจิสเตอร์ 2 ตัวในไอโอพอร์ท รีจิสเตอร์ของ อีจีเอ/วีจีเอ ถูกชี้โดยการแอดเดรสผ่านอินเด็กซ์รีจิสเตอร์ (index register หรือ address register) และรีจิสเตอร์ข้อมูล (data register)

ตาราง 2.2 กลุ่มรีจิสเตอร์ของ อีจีเอ/วีจีเอ

จำนวนรีจิสเตอร์	ชื่อกลุ่ม
4	General registers
5	Sequencer registers
28	CRT controller registers
9	Graphics controller registers
22	Attribute controller
5	Color registers

2.6 โหมดการแสดงผล (Display Modes)

โหมดการแสดงผล แบ่งได้เป็น 2 โหมด คือ โหมดตัวอักษรและโหมดกราฟิค (all-points-addressable) ซึ่งทั้ง 2 โหมดมีความแตกต่างกันดังที่ได้อธิบายมาแล้ว นอกจากนี้ผู้ใช้ยังสามารถออกแบบโหมดการทำงานขึ้นมาเองได้ โดยการใช้ รีจิสเตอร์ แต่วิธีนี้มีข้อเสียคือ ความซับซ้อนและความยุ่งยาก ซึ่งหากเกิดข้อผิดพลาดขึ้น อาจทำความเสียหายให้กับจอ หรือ ตัวปรับ (adapter) ได้ วิธีที่สะดวกและมีประสิทธิภาพ คือ การใช้ไบออส

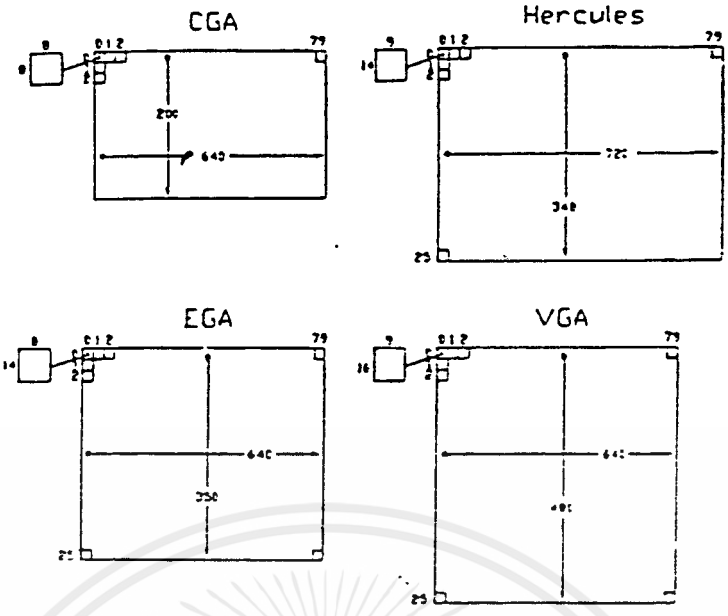
2.7 ความละเอียดของอีจีเอและวีจีเอ (EGA & VGA Resolution)

ความละเอียด ของ อีจีเอ/วีจีเอ ในโหมดความละเอียดต่ำ คือ 320 x 200 จุดภาพ (pixels) ในโหมดความละเอียดสูง คือ 640 x 350 จุดภาพ และวีจีเอ ยังสามารถให้ความละเอียด ที่เพิ่มขึ้นถึง 640 x 480 จุดภาพ ความละเอียดของตัวปรับการแสดงผล ที่เป็นที่นิยมกันทั้ง 4 ชนิด แสดงในรูปที่ 2.1

2.8 ความละเอียดของสี (Color Resolution)

โหมดการแสดงผลหลายโหมดใน อีจีเอ/วีจีเอ สามารถแสดงได้ 16 สีพร้อมกัน ซึ่งเลือกได้จาก 64 สีใน อีจีเอ และ 256K สีใน วีจีเอ นอกจากนี้ โหมด 13Hex ใน วีจีเอ สามารถแสดงได้ถึง 256 สีพร้อมกันจาก 256K สี(262,144สี)

จำนวนของสีที่แสดงได้พร้อมกัน มีความสัมพันธ์กับจำนวนบิทที่เกี่ยวข้องกับแต่ละจุดภาพ ใน หน่วยความจำแสดงผล บิทยิ่งมาก สียิ่งมาก ดังแสดงในตารางที่ 2.3



รูป 2.1 ความละเอียดของตัวปรับแสดงผล 4 ชนิด

ตาราง 2.3 จำนวนของสีที่แสดงได้พร้อมกัน

บิท/จุดภาพ	จำนวนสี
1	2
2	4
4	16
8	256

ค่าของจำนวนสีจะเป็นค่าที่ได้จาก 2 ยกกำลังด้วยค่าจำนวนบิท/จุดภาพ. หน่วยความจำแสดงผล ถูกจัดเป็นไบต์ๆ เพื่อที่จะใช้หน่วยความจำให้เกิดประโยชน์ได้เต็มที่ จึงควรบรรจุ จุดภาพ เป็นคู่ลงไปในไบต์ ความละเอียดซึ่งเกี่ยวข้องกับจำนวนบิท/จุดภาพ จะกำหนดจำนวนของ หน่วยความจำแสดงผล ทั้งหมด ดังแสดงในตารางที่ 2.4

ตาราง 2.4 จำนวนหน่วยความจำแสดงผล

ความละเอียด	บิท/จุดภาพ	จำนวนหน่วยความจำ
320 x 200	1	8,000 ไบต์
320 x 200	2	16,000 ไบต์
320 x 200	8	64,000 ไบต์
640 x 200	2	32,000 ไบต์
640 x 350	1	28,000 ไบต์
640 x 350	2	56,000 ไบต์
640 x 350	4	112,000 ไบต์
640 x 480	1	38,400 ไบต์
640 x 480	4	153,600 ไบต์

2.9 พาเล็ทสีและรีจิสเตอร์สี (COLOR PALETTE AND COLOR REGISTER)

พาเล็ทสี (Color Palette)

พาเล็ทสี เป็นตัวที่ทำหน้าที่เปลี่ยนข้อมูลใน หน่วยความจำแสดงผล ให้เป็นสี ใน วีจีเอ พาเล็ทสี จะมี รีจิสเตอร์สี เพิ่มเข้ามาเพื่อเพิ่มประสิทธิภาพ

พาเล็ทสี เปลี่ยนสีได้อย่างรวดเร็ว เพราะจะทำการเปลี่ยนเฉพาะข้อมูลที่ต้องการแก้ไขเท่านั้น เช่น ภาพใน อีจีเอ/วีจีเอ กินเนื้อที่ใน หน่วยความจำแสดงผล 256Kไบต์ ในโหมด 16 สี จะแสดงสีได้พร้อมกัน 16 สี ถ้าไม่มีพาเล็ทสีแล้ว ข้อมูลทั้ง 256Kไบต์ ใน หน่วยความจำแสดงผล จะถูกแก้ไขทั้งหมด แต่เมื่อมี พาเล็ทสี แล้ว ข้อมูลเพียง 16 ไบต์เท่านั้นที่ถูกแก้ไข

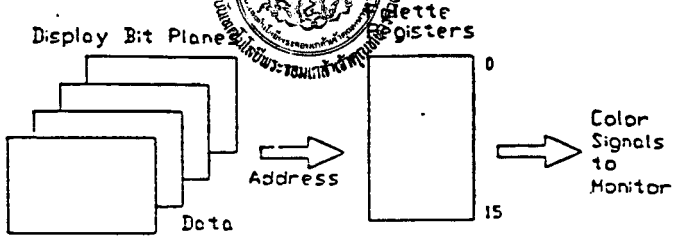
ประโยชน์อีกอย่างของการใช้พาเล็ทสี คือ จำนวนสีที่สามารถแสดงได้จะมากกว่าจำนวนสีที่สามารถแสดงได้พร้อมกัน เช่น อีจีเอ สามารถแสดงได้ 16 สีพร้อมกันจากทั้งหมด 64 สี แต่ละสีในพาเล็ทรีจิสเตอร์ ถูกแทนด้วยคำขนาด 6 บิท ใน วีจีเอ มีโหมด 256 สี ซึ่งแสดงได้ 256 สีในเวลาเดียวกัน แต่ละสีถูกแทนด้วยคำขนาด 18 บิทภายในรีจิสเตอร์สี ซึ่งมีถึง 256K สี ให้เลือก

พาเล็ทรีจิสเตอร์ (Palette registers)

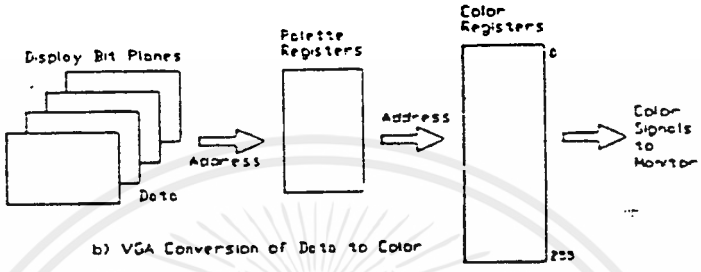
พาเล็ทสี ประกอบด้วย พาเล็ทรีจิสเตอร์ 16 ตัว ข้อมูลใน บิทเพลน ซึ่งตรงกับแต่ละจุดภาพ จะเป็นตัวชี้พาเล็ทรีจิสเตอร์.

ใน อีจีเอ :พาเล็ทรีจิสเตอร์ ที่ถูกชี้จะส่งรหัสสีไปยัง มอนิเตอร์ ดังรูป 2.2

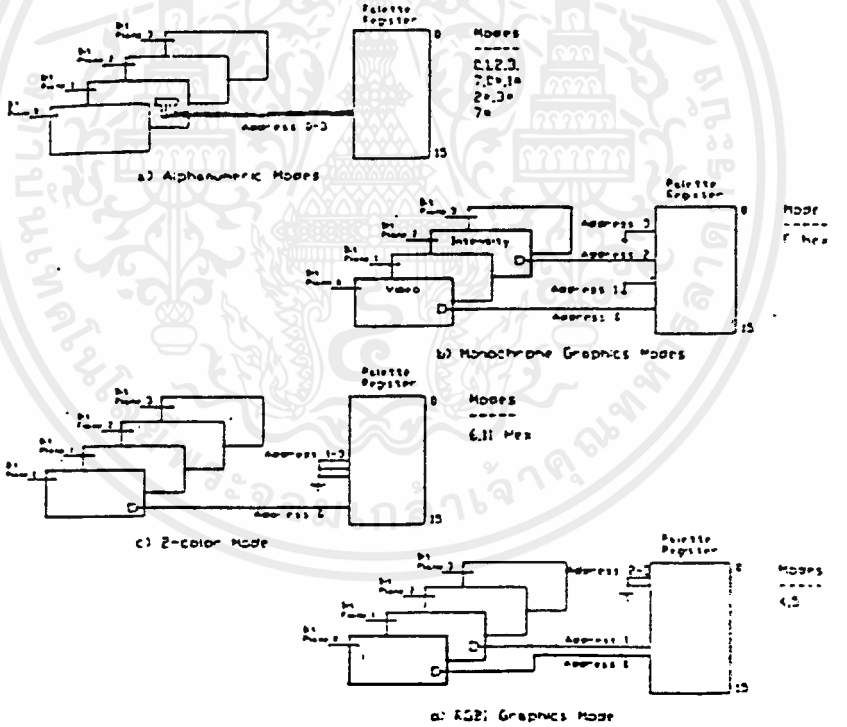
ใน วีจีเอ :พาเล็ทรีจิสเตอร์ ที่ถูกเลือกจะสร้างแอดเดรสซึ่งจะชี้ไปยังรีจิสเตอร์สี ซึ่งรีจิสเตอร์สี จะส่งรหัสสีไปยังมอนิเตอร์ ดังรูป 2.3



รูป 2.2



รูป 2.3

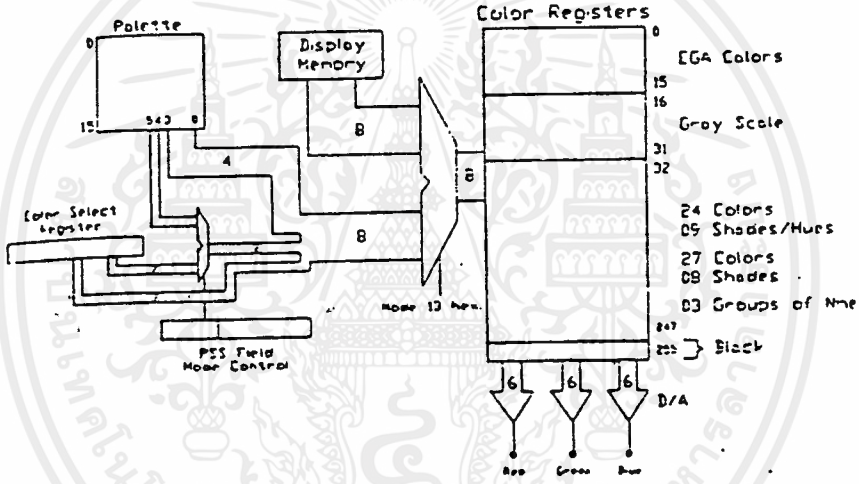


รูป 2.4

รีจิสเตอร์สี (Color Register)

ใน วีจีเอ จะมี 256 รีจิสเตอร์สี แต่ละจุดภาพหรือ ตัวอักษร ประกอบด้วยข้อมูล 8 บิท ซึ่งใช้แอดเดรส 16 พาเล็ทรีจิสเตอร์. เอาท์พุทของ พาเล็ทรีจิสเตอร์ ร่วมกับ ข้อมูลจาก หน่วยความจำแสดงผล จะใช้แอดเดรสรีจิสเตอร์สี ซึ่ง รีจิสเตอร์สี จะส่งรหัสสีไปยัง ดิจิตอลทู อนาล็อก คอนเวิตเตอร์ (digital-to-analog converters หรือ DAC) โดยที่เอาท์พุตของ DAC จะถูกส่งไปยังจอต่อไป รีจิสเตอร์สี แต่ละตัวจะประกอบด้วย 3 ส่วน คือ แดง, เขียว และ น้ำเงิน แต่ละส่วนกว้าง 6 บิท

สำหรับ พาเล็ทรีจิสเตอร์ในวีจีเอนั้นมีเพียงเพื่อให้สามารถคอมแพ็ททิเบิลกับ อีจีเอ เมื่ออยู่ในโหมด 13 Hex แล้ว รีจิสเตอร์สี เพียงอย่างเดียวก็เพียงพอ ซึ่งการแอดเดรส รีจิสเตอร์สี แสดงดังรูป 2.5



รูป 2.5

2.10 การประมวลผลทางตัวอักษร (ALPHANUMERIC PROCESSING)

ในโปรแกรมจำพวกโปรแกรมประยุกต์ (APPLICATION PROGRAM) ต่างๆเช่น เวิร์ดโปรเซสเซอร์ (WORD PROCESSOR), สเปรดชีท (SPREADSEET), ฐานข้อมูล(DATABASE), หรือแม้แต่ดอส (DOS) เองก็ตาม มักนิยมใช้โหมดตัวอักษรในการแสดงผลตัวอักษรมากกว่าโหมดกราฟฟิคส์ เหตุผลที่สำคัญก็คือ ในโหมดตัวอักษร จะแสดงตัวอักษรได้เร็วกว่า โหมดกราฟฟิคส์มาก โดยในโหมดกราฟฟิคส์ หากเราจะเขียนตัวอักษรสักตัวหนึ่ง เราจะต้องกำหนดจุดทุกจุดบนจอให้ได้รูป ตามแบบตัวอักษร (ดูเรื่อง GRAPHIC PROCESSING) แต่ในที่นี้จะแสดงคร่าวๆว่าการจะเขียนตัวอักษรขนาด 8*14 นั้นจะใช้จำนวนไบต์ มากกว่าในโหมดตัวอักษรเท่าใด

โหมดกราฟฟิคส์ (ใน โหมด 16 สี) จะใช้ 4 บิท ในการบอก แอททริบิวต์ ดังนั้น ฟอนต์ แบบ 8*14 จะใช้ 14 ไบต์*4 = 56 ไบต์

โหมดตัวอักษร ใช้ 2 ไบต์ คือ รหัสตัวอักษร และ แอททริบิวต์ จะเห็นว่าใน โหมดกราฟฟิคส์ นั้นต้องเขียนข้อมูลมากกว่าในโหมดตัวอักษร 28 เท่า

เหตุที่ในโหมดตัวอักษรเร็วกว่า โหมดกราฟฟิคส์ (ในการแสดงตัวอักษร)ก็คือในโหมดตัวอักษร จะโหลดชุดของตัวอักษร (CHARACTER FONT)ลงบนหน่วยความจำแสดงผล บิทเพลน 2 หรืออาจเกินมาถึง บิทเพลน 3 เวลาเราเขียนรหัสตัวอักษร ซึ่งก็คือ รหัสแอสกีนั่นเอง จะมีกลไกทางฮาร์ดแวร์ ทำให้รหัสนั้นลงไปเก็บที่ หน่วยความจำแสดงผล บิทเพลน 0 และเก็บ แอททริบิวต์ไบต์ไว้ที่ บิทเพลน 1 จากนั้นก็จะนำข้อมูลจาก บิทเพลน 0 ไปค้นหาว่าเป็นตัวอักษรใดที่บิทเพลน 2,3 แล้วนำมารวมกับ แอททริบิวต์ไบต์ ซึ่งเป็นตัวบอกสีของฟอร์กราวด์ (FOREGROUND) และ แบ็คกราวด์ (BACKGROUND) แล้วนำมาแสดงผลบนจอภาพ เช่น สมมติว่าเราต้องการเขียนตัวอักษร "A" ซึ่งมี รหัสแอสกี 65 ดังนั้นเมื่อเราป้อนค่า 65 ลงบน บิทเพลน 0 มันจะนำไปค้นหาที่ บิทเพลน 2,3 ฟอนต์ ของตัว "A" จะอยู่แอดเดรส ที่

อักษรตัวแรกของ ฟอนต์ + (65 * จำนวนแถว)

ตาราง 2.5 โหมดการแสดงผลแบบตัวอักษร(ALPHANUMERIC DISPLAY MODE)

โหมด(HEX)	แถว*คอลัมน์	ขนาด	จำนวนสี	#เพจ	ความละเอียด
0,1	40*25	8*8	16	8	320*200
0*,1*	40*25	8*14	16	8	320*350
0+,1+	40*25	9*16	16	8	360*400
2,3	80*25	8*8	16	8	640*200
2*,3*	80*25	8*14	16	8	640*350
2+,3+	80*25	9*16	16	8	720*400
7	80*25	9*14	bw	8	720*350
7+	80*25	9*16	bw	8	720*400

ข้อสังเกตุ เครื่องหมาย * และ + หมายถึง การ์ด (CARD) ที่ต่ออยู่กับ มอนิเตอร์ที่มีความละเอียดสูง

โหมดตัวอักษรแบบสี (RGBI Alphanumeric Mode)

โหมด 0 และ โหมด 1

เป็นโหมดการแสดงผลตัวอักษรขนาด 40 คอลัมน์ 16 สี มีความละเอียด 320 จุดภาพ/เส้นสแกน โดยมี ตัวอักษรที่กำหนดไว้ขนาด 8*8 เมื่อใช้ 200 เส้นสแกนแนวดิ่ง และมี ตัวอักษรที่กำหนดไว้ขนาด 8*8 เมื่อใช้ 350 เส้นสแกนแนวดิ่ง และมีความละเอียด 360 จุดภาพ/เส้นสแกน โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16 เมื่อใช้ 400 เส้นสแกนแนวดิ่ง

มี แอดเดรส เริ่มต้นที่ B8000 HEX

โหมด 2 และ โหมด 3

เป็นโหมดการแสดงผลตัวอักษร 80 คอลัมน์ 16 สี มีความละเอียด 640 จุดภาพ/เส้นสแกน

โดยที่ 200 เส้นสแกนแนวดิ่ง จะมี ตัวอักษรที่กำหนดไว้ขนาด 8*8

ที่ 350 เส้นสแกนแนวดิ่ง จะมี ตัวอักษรที่กำหนดไว้ขนาด 8*14 และความละเอียด 720*400 จุดภาพ โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16

มี แอดเดรส เริ่มต้นที่ B8000 HEX

โหมดตัวอักษรแบบขาวดำ (Monochrome Alphanumeric Mode) โหมด 7)

ในโหมด7นี้ รหัสของตัวอักษรแต่ละตัวจะอยู่ใน แอททริบิวต์ไบต์ ของ ตัวอักษร ซึ่ง แอททริบิวต์ ไบต์ อยู่ใน บิทเพลน2 ในโหมดนี้ แอททริบิวต์ ไบต์ สามารถเข้าถึง นาเล็กรีจิสเตอร์ ได้ทั้ง 16 ตัว แต่มีค่าเพียง 4 ค่าเท่านั้น ทั้งนี้เนื่องจากว่ารหัสจะอยู่ใน 2 บิทเท่านั้นคือ บิท 3 และ บิท 4 บิท 0-2, 5-7 ไม่มีผล บิท 3 จะเป็นบิทแสดงสัญญาณวิดีโอ (video) ส่วน บิท 4 เป็นบิทแสดงความเข้มของสัญญาณ

โหมด 7 เป็นโหมดการแสดงผลตัวอักษร 80 คอลัมน์ สี ขาว ดำ ซึ่ง คอมแพททิเบิล กับ โมโนโครม (MDA) มีความละเอียด 720*350 โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*14 และมีความละเอียด 720*400 โดยมี ตัวอักษรที่กำหนดไว้ขนาด 9*16

มี แอดเดรส เริ่มต้นที่ B8000 HEX

ฟอนต์ของตัวอักษร (CHARACTER FONT)

ขนาดของตัวอักษรใน ฟอนต์ นั้นมีตั้งแต่ 1-32 แถว ซึ่งสามารถกำหนดได้โดยการกำหนดจำนวนเส้นสแกนสูงสุด ในซีอาร์ทีคอนโทรลเลอร์รีจิสเตอร์ (CRT CONTROLLER REGISTER) แต่ฟอนต์ที่พบในรอมไบออส (ROM BIOS) นั้นมีอยู่ 5 แบบคือ 8*8,8*14,8*16,9*14,9*16 ซึ่งมีลักษณะดังรูป 2.6 การคำนวณหาตำแหน่งของ ฟอนต์ สามารถคำนวณได้จากสูตร

CHAR PATTERN ADDR = CHAR BASE ADDR + (ASCII CODE * #BYTE/CHAR)

เช่นตัวอักษร "A" ในตัวอย่างที่แล้ว สมมติว่าอยู่ใน ฟอนต์ 8*8 ซึ่งอยู่ที่หน่วยความจำแสดงผล บิทเพลน 2 ที่ ออฟเซทแอดเดรส 2000 HEX

ดังนั้นที่ แอดเดรส = (2000HEX)+(65*8DEC)

= 2000HEX + 520DEC = 2000HEX + 208HEX

= 2208HEX

จะมีค่าเหล่านี้อยู่

แอดเดรส 2208 -----> 30HEX

แอดเดรส 2209 -----> 48HEX

แอดเดรส 2210 -----> 84HEX

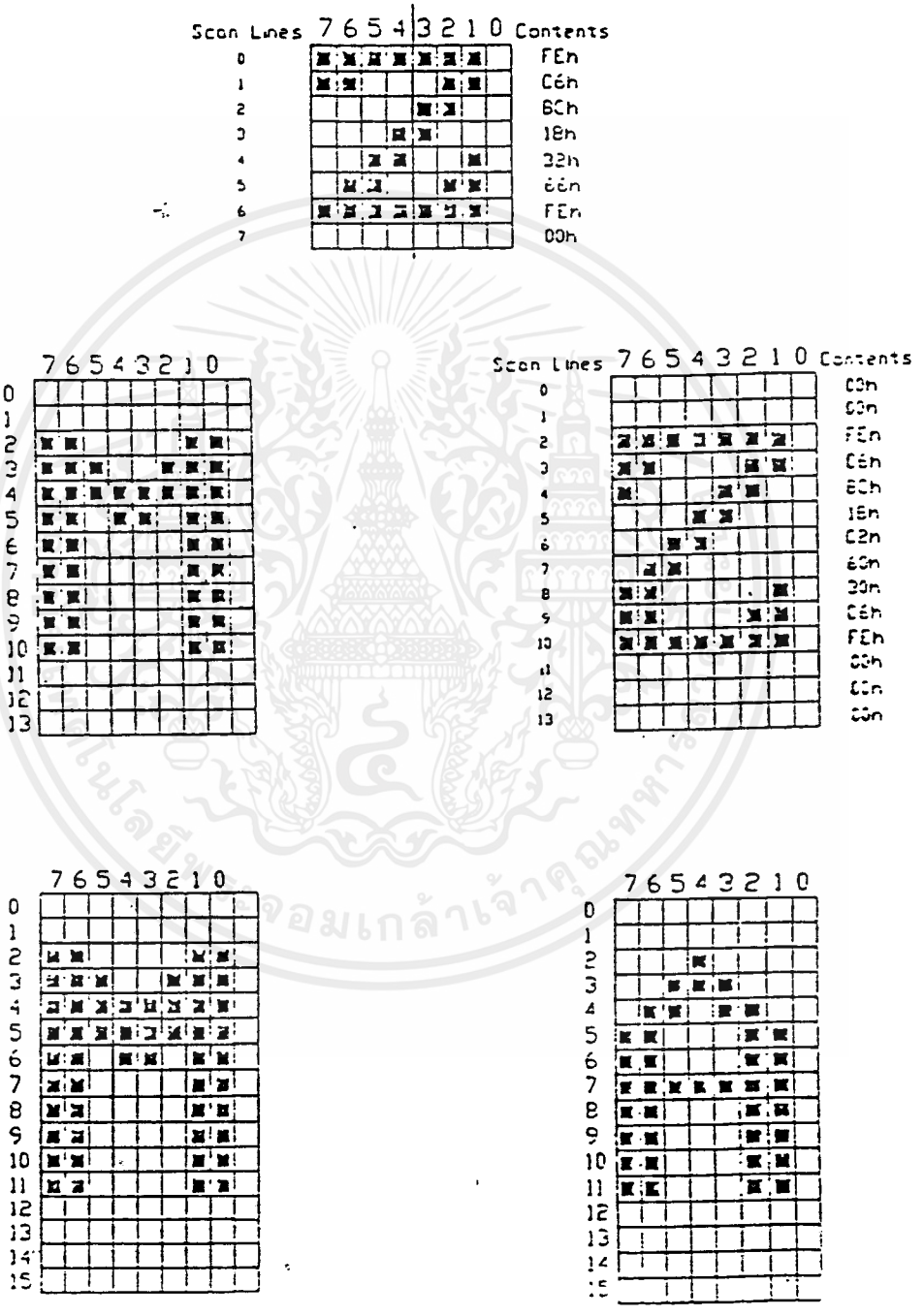
แอดเดรส 2211 -----> 84HEX

แอดเดรส 2212 -----> FCHEX

แอดเดรส 2213 -----> 84HEX

แอดเดรส 2214 -----> 84HEX

แอดเดรส 2215 -----> 00HEX



รูป 2.6 แสดงฟอนต์ตัวอักษร

2.11 การประมวลผลทางกราฟิคส์ (GRAPHICS PROCESSING)

ในปัจจุบันคอมพิวเตอร์กราฟิคส์ (COMPUTER GRAPHIC) ได้มีการพัฒนาไปเป็นอย่างมาก เนื่องจาก รูปภาพสามารถสื่อความเข้าใจได้อย่างรวดเร็ว เช่น ถ้าต้องพิจารณาข้อมูลที่เป็นตัวเลขมากมายก็จะทำให้ยากต่อการเข้าใจ หากนำมาพล็อตเป็นกราฟแล้ว จะสามารถหาข้อสรุปเกี่ยวกับจำนวนเหล่านั้นได้อย่างรวดเร็ว เช่น อัตราการเพิ่ม,ลด นอกจากนี้ยังง่ายต่อการนำเสนออีกด้วย

ทางด้าน ฮาร์ดแวร์ของการแสดงผลก็ได้มีการพัฒนาเช่นเดียวกับ ซอฟท์แวร์ ในปัจจุบัน ภาคแสดงผลของคอมพิวเตอร์ มีขีดความสามารถใกล้เคียงกับเวิร์คสเตชั่น (WORKSTATION) เช่น วีจีเอ(VIDEO GRAPHIC ARRAY) นั้นสามารถแสดงสีได้คราวละ 256 สี จากสีที่ให้เลือกทั้งหมด 256K สีที่ความละเอียด 320*200 จุดภาพ

ในโหมดตัวอักษรนั้น ใช้ 2 ไบต์ ในการแทนตัวอักษร 1 ตัวคือ รหัสแอสกี และ แอททริบิวต์ไบต์ ส่วนโหมดกราฟิคส์นั้น ตำแหน่งในหน่วยความจำทุกเพลน จะแทนจุดบนจอภาพ โดยอาจเป็น 1 บิท/จุดภาพ, 2 บิท/จุดภาพ, 4 บิท/จุดภาพ, 8 บิท/จุดภาพ ขึ้นอยู่กับจำนวนสีที่ต้องการแสดงผล ถ้าหากต้องการให้มีสีหลายๆสี ก็ต้องใช้จำนวน บิท/จุดภาพ มาก ซึ่งจะทำให้ความละเอียดของภาพลดลง

หน่วยความจำแสดงผลของวีจีเอ ประกอบด้วย หน่วยความจำอ่านเขียนแบบไดนามิค (DYNAMIC READ/WRITE MEMORY) ซึ่งแบ่งออกเป็น 4 ส่วน(4 บิทเพลน หรือ 4 bit map) แต่ละส่วนจะมีขนาดตั้งแต่ 16K ไบต์ ไปจนถึง 64 Kไบต์ ขึ้นอยู่กับโหมดการแสดงผล โหมดที่ใช้หน่วยความจำจะสามารถมีได้หลายเพจ ในตาราง2.6 แสดงจำนวนเพจ และ แอดเดรสเริ่มต้น ของแต่ละโหมด

ตาราง 2.6

เพจ	โหมด4,5H	โหมด D	โหมด E	โหมด 10H	โหมด 11H	โหมด 12H	โหมด13H
0	B8000	A0000	A0000	A0000	A0000	A0000	A0000
1	-	A2000	A4000	A8000	-	-	-
2	-	A4000	A8000	-	-	-	-
3	-	A6000	AC000	-	-	-	-
4	-	A8000	-	-	-	-	-
5	-	AA000	-	-	-	-	-
6	-	AC000	-	-	-	-	-
7	-	AE000	-	-	-	-	-

มีเทคนิค 2 ประการ ในการจัดเก็บข้อมูลลง หน่วยความจำแสดงผล คือ บิทเพลน และ แพคดิสเพลย์ฟอร์แมท (Packed Display Format)

-บิทเพลน เป็นการเก็บข้อมูลของ จุดภาพ โดยแบ่งเป็นเพลน เช่น ในโหมด 16สี ใช้ 4 บิท/จุดภาพ แต่ละบิท นี้ จะถูกจัดเก็บให้อยู่บนเพลนต่างๆกัน คือ บิท 0 อยู่บน เพลน 0 ,บิท 1,2,3 อยู่บน เพลน 1, 2, 3 ตามลำดับ

-แพคดิสเพลย์ฟอร์แมท เป็นการเก็บข้อมูลของ จุดภาพ ลงในเพลน เดียวกัน

ตาราง 2.7 แสดงการใช้วิธีการจัดเก็บข้อมูลของจุดภาพในแต่ละโหมด

โหมด (HEX)	รูปแบบ	จำนวนคอลัมน์	จุดภาพ/ไบต์
4,5	แพค	4	4
6	แพค	2	8
F	บิทเพลน	2	8
11	แพค	2	8
D	บิทเพลน	16	8
E,10	บิทเพลน	16	8
12	บิทเพลน	16	8
13	แพค	256	1

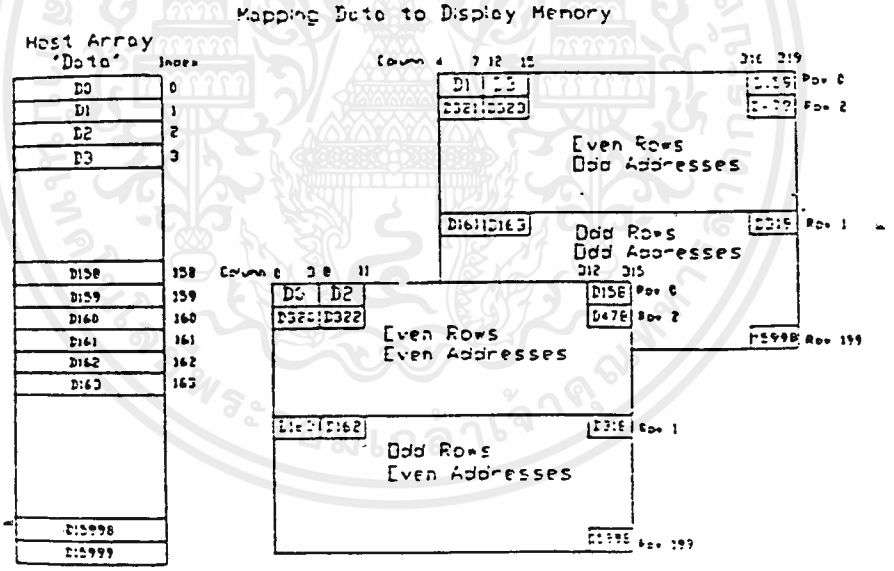
โหมดกราฟฟิคส์สี (RGBI Graphics Modes]โหมด 4,5)

โหมดกราฟฟิคส์สี คือ โหมด 4 และ 5 แต่ละ จุดภาพ ถูกแทนด้วย 2 บิทแอททริบิวต์สี โหมดนี้สามารถใช้ได้กับซีจีเอ ด้วย

ในโหมด 4 และ 5 แอททริบิวต์บิท ทั้ง 2 บิทอยู่ติดกันในข้อมูลไบต์เดียวกัน อุปกรณ์ทางฮาร์ดแวร์ จะควบคุมทางแฟลกคู่/คี่ (odd/even flag) คือ ข้อมูลของไบต์คู่จะถูกส่งไปยัง บิทเพลน 0 และข้อมูลไบต์คี่จะถูกส่งไปยัง บิทเพลน 1 ดังนั้นข้อมูลที่ถูกส่งเข้าไปยัง อีจีเอ/วีจีเอ เมื่ออยู่ในโหมด 4 หรือ 5 จะถูกแบ่งแยกไปที่ บิทเพลน 0 และ 1 ทั้งสองโหมดนี้เป็นการแสดงผลขนาด 320*200 จุดภาพ 4 สี ดังนั้นใน 1 จุดภาพ จึงต้องการ 2 บิท จึงจะแสดงได้ 4 สี โดยจะใช้ บิทเพลน 0,1 ในการเก็บข้อมูล ดังรูป 2.7

ตาราง 2.8 แสดงโหมดต่างๆ ในโหมดกราฟฟิคส์

โหมด(HEX)	จำนวนหน้า	แอดเดรส	จำนวนสี	ความละเอียด
4	1	B8000	4/256K	320*200
5	1	B8000	4/256K	320*200
6	1	B8000	2/256K	640*200
D	8	A0000	16/256K	320*200
E	4	A0000	16/256K	640*200
F	2	A0000	bw	640*350
10	1	A0000	16/256K	640*350
11	1	A0000	2/256K	640*480
12	1	A0000	16/256K	640*480
13	1	A0000	256/256K	320*200



รูป 2.7

โหมด 2 สี (Two-Color Modes)

สามารถแบ่งย่อยได้เป็น 2 โหมด ดังนี้

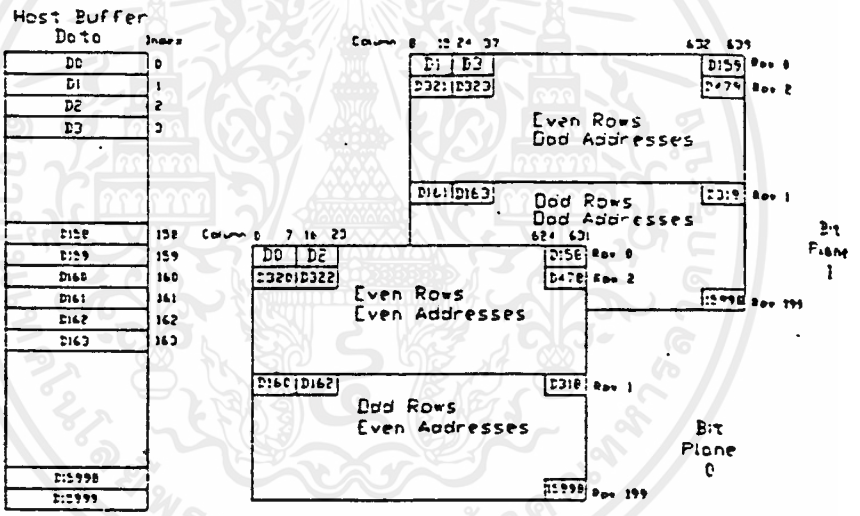
ซีจีเอกูลัลเลอร์อีมูเลชั่น (CGA Two-Color Emulation)โหมด6)

ในโหมด 6 ของ ซีจีเอ/วีจีเอ 1 บิทต่อ 1 จุดภาพ จะเลือก 1 ใน 2ของพาเล็ท ค่าที่โหลดลงไปใน พาเล็ทรีจิสเตอร์ แสดงในตาราง 2.9

ตาราง 2.9 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 6 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17

ใน หน่วยความจำแสดงผล แต่ละ ไบต์ จะประกอบด้วย 8 จุดภาพที่ต่อเนื่องกัน วิธีการต่างๆจะเหมือนกับในโหมดกราฟิคส์สี (โหมด 4 และ 5) ยกเว้นแต่ว่า แสดงได้ 2 สี แทนที่จะเป็น 4 สี ในโหมด 6 นี้ บิทเพลน 0 และ 1 ถูกใช้เก็บข้อมูลไบต์คู่และคี่ตามลำดับ เมื่อ พาเล็ทสี ถูกแอดเดรส แอททริบิวต์บิท 1 บิท (สำหรับแต่ละจุดภาพ) จะอยู่ใน พาเล็ทรีจิสเตอร์ ที่แอดเดรสบิท 0 ดังรูป 2.8



รูป 2.8

วีจีเอ 2 สี (VGA Two-color Emulation โหมด 11)

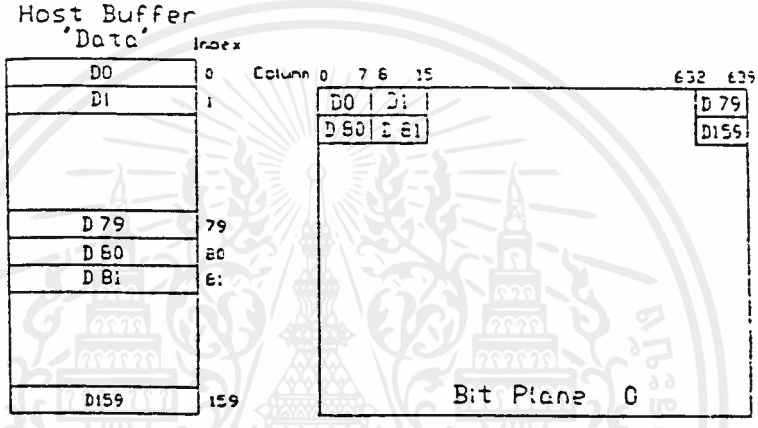
ในโหมด 11 ของ วีจีเอ 1บิทต่อ 1จุดภาพ จะเลือก 1 ใน 2 ของพาเล็ท ค่าที่โหลดลงใน พาเล็ทรีจิสเตอร์ แสดงในตารางที่ 2.10

ตาราง 2.10 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 11 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37

ใน หน่วยความจำแสดงผล แต่ละ ไบต์ ประกอบด้วย 8 จุดภาพ หน่วยความจำจะแมป (map) อย่างตรงๆ คือ ใช้เพียง บิทเพลน 0 ในการเก็บข้อมูล เมื่อพาเล็ทสี ถูก แอดเดรส แอททริบิวต์ บิท 1 บิท (สำหรับแต่ละจุดภาพ) จะอยู่ใน พาเล็ท รีจิสเตอร์ ที่ แอดเดรส บิท 0

เป็นการแสดงผลความละเอียด 640*480 จุดภาพ 2สี ดังนั้นจึงต้องการ 1 บิท/จุดภาพ มี แอดเดรส เริ่มต้นที่ A0000 และใช้ หน่วยความจำแสดงผลเพียงบิทเพลน 0 เท่านั้น ดังในรูป 2.9

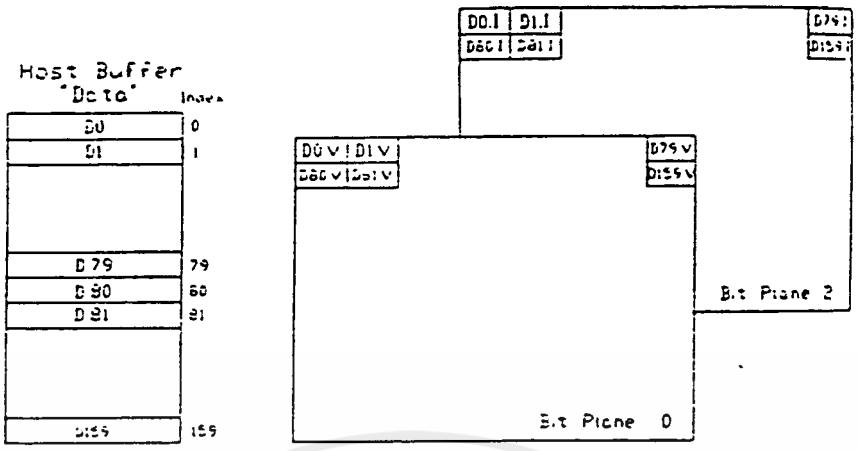


รูป 2.9

โมโนโครมกราฟิคส์โหมด (Monochrome Graphics Mode)Mode F hex)

ใน โหมด F hex มีรหัสขนาด 2 บิท ซึ่งอ้างได้ถึงทุกๆจุดภาพ 2 บิทนี้แทนบิทสัญญาณวิดีโอ และ บิทความเข้มของโมโนโครมแอททริบิวต์ แอททริบิวต์ ขนาด 2 บิท จำนวน 4 แอททริบิวต์ ถูกบรรจุรวมกันเป็น 1 ไบต์ เพื่อโหลดเข้าไปในหน่วยความจำแสดงผล เมื่อ โหมด F hex แอคทีฟ ฮาร์ดแวร์จะโหลด วิดีโอแอททริบิวต์บิตทั้งหมดลงใน บิทเพลน 0 และ แอททริบิวต์บิทที่แสดงความเข้มทั้งหมดลงในบิทเพลน 2

โหมดนี้ จะแสดงผลขนาด 640*350 จุดภาพ 2 สี แต่มีความเข้มด้วย ดังนั้นจึงแสดงความแตกต่างได้ 4 ระดับ ใช้ 2 บิท/จุดภาพ ใช้หน่วยความจำแสดงผลบิทเพลน 0,2 ในการเก็บข้อมูล ดังรูป 2.10



รูป 2.10

โหมด 16 สี (Mode 16 color) โหมด D,E,10H,12H)

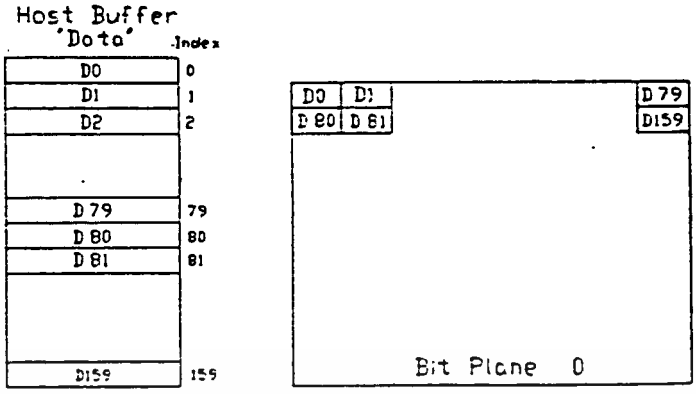
โหมดนี้ประกอบด้วย D,E,10 และ 12 Hex 1จุดภาพ ถูกแทนด้วย 4 บิทใน หน่วยความจำแสดงผล 4บิทนี้อยู่ใน บิทเพลน 0-3 (1 บิท อยู่ใน 1 บิทเพลน) ซึ่งใช้เป็น แอดเดรส ขนาด 4 บิท เพื่อเลือก พาเล็ทรีจิสเตอร์

แต่ละ 2 บิทใน 1 พาเล็ทรีจิสเตอร์ จะแทน สีแดง, เขียว, น้ำเงิน (2บิท/สี) ดังนั้น แต่ละสี จึงมีความเข้มได้ 4 ระดับ ค่าที่เป็นไปได้ของสีเหล่านี้ แสดงในตารางที่ 2.11

ตาราง 2.11 ค่าในพาเล็ทที่กำหนดไว้ในโหมด D,E,10 และ 12 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	1	2	3	4	5	14	7	38	39	3A	3B	3C	3D	3E	3F

โหมดการแสดงผล 16 สีในโหมดนี้ มีความละเอียดต่างกันตามโหมด คือ 320*200 (โหมด D), 640*200 (โหมด E), 640*350 (โหมด 10), 640*480 (โหมด 12) โดยมีลักษณะการเก็บข้อมูลดังรูป 2.11



รูป 2.11

โหมด 256 สี (256-color mode)

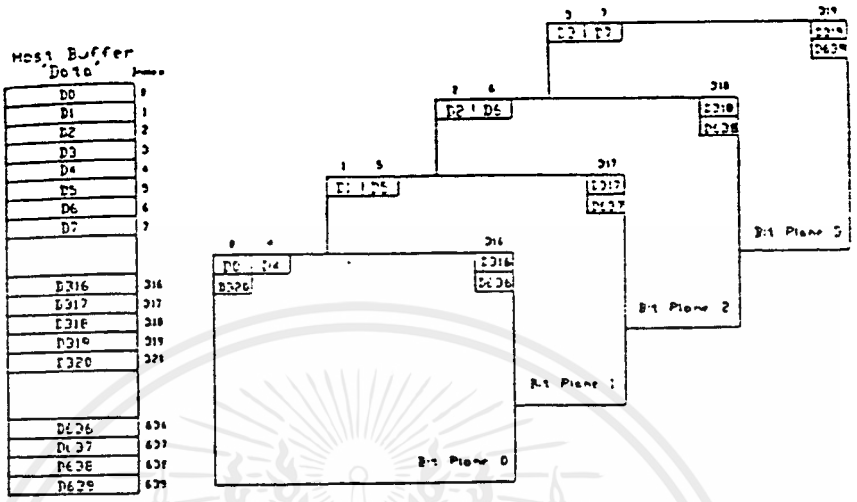
โหมดนี้คือ โหมด 13Hex ซึ่งแสดงได้ถึง 256 สี แต่ละจุดภาพถูกแทนด้วยข้อมูลขนาด 8 บิท ข้อมูลนี้เมื่อถูกโหลดเข้าไปยัง หน่วยความจำแสดงผล จะถูกส่งผ่าน 4 บิท เพลน และจะถูกแยกเป็น 2 ส่วนๆละ 4 บิท ส่วนแรกซึ่งเป็น 4 บิทต่ำ(บิท 0-3) จะ แอดเดรส พาเล็ทรีจิสเตอร์ และ เอาท์พุท 4 บิทต่ำของ พาเล็ทรีจิสเตอร์ ถูกใช้เป็น 4 บิทต่ำ (บิท0-3) สำหรับการแอดเดรสรีจิสเตอร์สี ส่วนที่ 2 คือ บิท 4-7 จาก หน่วยความจำแสดงผล จะถูกใช้เป็น 4 บิทสูง (บิท4-7) ในการแอดเดรสรีจิสเตอร์สี

พาเล็ทรีจิสเตอร์ ถูกจำกัดให้เปลี่ยนแอดเดรส เป็นข้อมูลแบบ หนึ่งต่อหนึ่ง (one-to-one) ดังนั้น ค่าในพาเล็ท จึงเหมือนกับแอดเดรสของมัน ดังแสดงใน ตารางที่ 2.12

ตาราง 2.12 ค่าในพาเล็ทที่กำหนดไว้ในโหมด 13 Hex

รีจิสเตอร์	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ค่า	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

ลักษณะการจัดเก็บข้อมูลของโหมด 13 เป็นดังรูป 2.12



รูป 2.12

การทำงานของโปรแกรมต่างๆ

3.1 ฟังก์ชัน Initialize ( )

เป็นฟังก์ชันที่ใช้เพื่อต้องการเข้าสู่สภาวะกราฟฟิก โดยฟังก์ชันนี้จะทำหน้าที่โหลดแฟ้มข้อมูลที่เป็น Graphics driver ที่อยู่ในดิสก์มาเก็บไว้ในหน่วยความจำ

initgrap(&GraphFriver,& GraphMode,"c:\tc\bgi");

GraphDriver เป็นตำแหน่งของตัวแปรจำนวนเต็มที่ ทำหน้าที่กำหนดชนิดของแฟ้มข้อมูลที่เป็น Graphics Driver การกำหนดชนิดของแฟ้มข้อมูลนี้กำหนดเป็นชื่อแมคโคร Detect จะทำการตรวจสอบชนิดของวงจรที่แสดงผลให้เองพร้อมกับจะโหลดแฟ้มข้อมูล Graphic Driverสำหรับวงจรนั้นเข้ามาเองโดยอัตโนมัติ GraphicMode เป็นการกำหนดขนาดของความละเอียด ในการแสดงผลของแต่ละ Graphic Driver ซึ่งสามารถกำหนดเป็นชื่อแมคโคร หรือค่าคงที่

Graphic Driver	Mode	ค่าคงที่	ความละเอียด
CGA	CGA0	0	320 x 200
	CGA1	1	320 x 200
	CGA 2	2	320 x 200
	CGA3	3	320 x 200
	CGAHI	4	640 x200
MCGA	MCGAC0	0	320 x 200
	MCGAC1	1	320 x 200
	MCGAC2	2	320 x 200
	MCGAC3	3	320 x 200
	MCGAMED	4	640 x 200
	MCGAHI	5	640 x 280
EGA	EGALO	0	640 x 200
	EGAHI	1	640 x 350
EGA64	EGA64LO	0	640 x 200
	EGA64HI	1	640 x 350
EGAMONO	EGAMONOHI	3	640 x 350
HERC	HERCMONOHI	0	720 x 348
ATT400	ATT400C0	0	320 x 200
	ATT400C1	1	320 x 200
	ATT400C2	2	320 x200
	ATT400C3Q	3	320 x 200
	ATT400CMED	4	640 x 200
	ATT400CHI	5	640 x 400
VGA	VGALO	0	640 x 200
	VGAMED	1	640 x 350
	VGAHI	2	640 x 480
PC3270	PC3270HI	0	720 x 350
IBM8514	IBM8514 LO	0	720 x 350
	IBM8514HI	1	1024 x 768

c:\tc\bgi กำหนดเส้นทางเพื่อค้นหาแฟ้มข้อมูล Graphic Driver (แฟ้มข้อมูลที่มีขนาดเป็น .BGI)

ถ้าเกิดการ Error จะพิมพ์ข้อมความ " Graphics system Error :<Error Code>แล้วออกจากโปรแกรม

```
void Initialize()
 {
    GraphicDriver=DETECT;
    initGraph(&GraphDriver, &GraphMode,"c:\tc\bgi);
    Error Code=Graphresult();
    if(ErrorCode!=grok)
    {
        printf("Graphic System Error :%s\n",grapherrormsg(ErrorCode));
        exit(1);
    }
    MaxColors=getmaxcolor()+1;
   getaspectratio(&xasp,&yasp);
   AspectRatio=(double)xasp/(double)yasp;
```

3.2 การเก็บภาพบางส่วนจากจอมอนิเตอร์ลงหน่วยความจำชั่วคราวโดยใช้ฟังก์ชัน Save Image ซึ่งจะต้องผ่านค่าให้กับฟังก์ชันคือ Left,Top,Right,Bottom,Size

รูปแบบ

SaveImage(int left,int right,int bottom,unsigned x size)

โดยกำหนดตัวแปรพอยเตอร์ส่งค่ากลับชื่อ Image การหาขนาดของภาพตามค่าที่ผ่านให้ฟังก์ชัน ใช้ฟังก์ชัน ImageSize

```
* Size = imagesize(left,top,right,bottom)
```

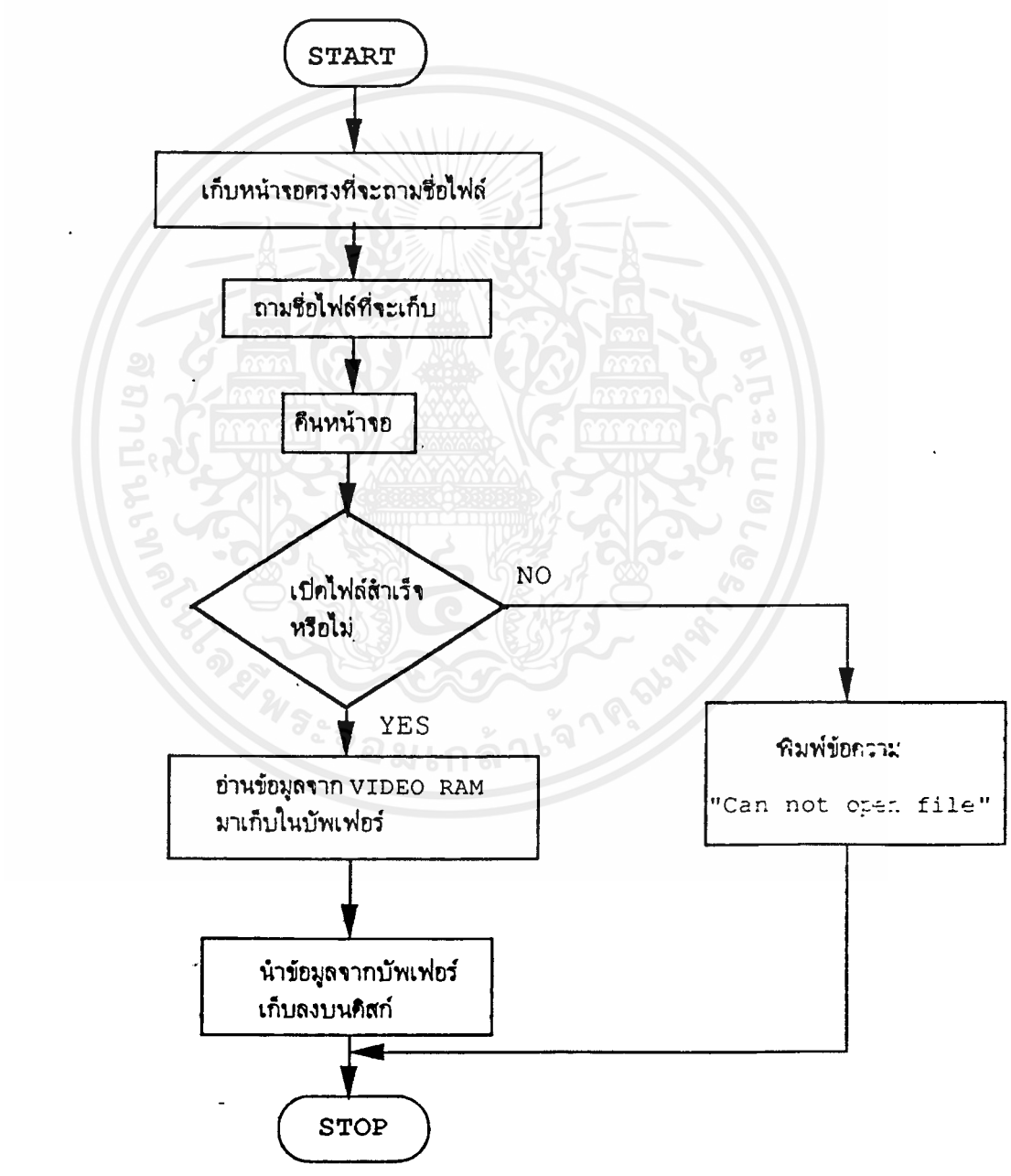
การเก็บภาพโดยใช้ฟังก์ชัน Sav Image นี้ ขนาดของจอภาพจะเก็บได้ครั้งละไม่เกิน 64 k

```
void *SaveImage(int left,int top,int right,int bottom,unsigned *size)
{
    void *image;
            *size=imagesize(left,top,right,bottom);
    image=malloc(*size);
   getimage(left,top,right,bottom,image);
   putimage(left,top,image,XOR_PUT);
   return(image);
```

3.3 การเก็บภาพกราฟฟิกและการดึงข้อมูล

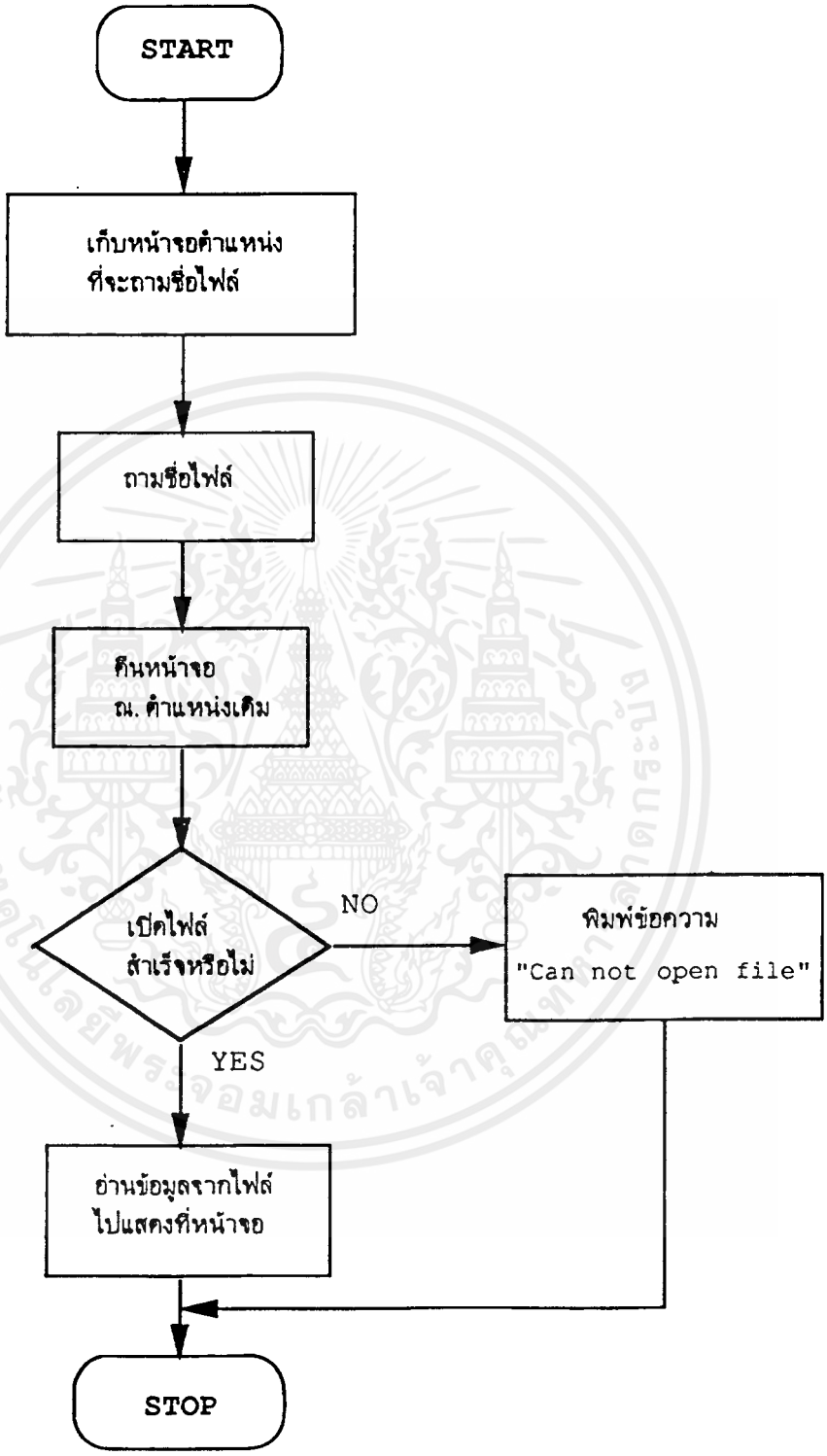
โดยหลักการแล้วการเก็บข้อมูลและการดึงข้อมูลรูปภาพทางกราฟฟิกนั้นออกจะไม่เป็นเรื่องยากเลย เพราะว่าสิ่งที่ปรากฏอยู่บนจอภาพนั้นก็คือข้อมูลที่อยู่ใน Video Ram และการคัดลอก (Copy) ข้อมูลจาก Video Ram ลงในดิสก์ หรือในทางกลับกันก็ไม่มีปัญหา แต่ปัญหาที่ต้องเผชิญก็คือ การป้อนชื่อไฟล์ของข้อมูลที่ต้องการจะเก็บอย่างไร เพื่อหลีกเลี่ยงการทำลายข้อมูลใน Video Ram ในการที่จะหลีกเลี่ยงปัญหานี้ ฟังก์ชัน Save Image จะถูกนำมาใช้ นั่นคือ จะเก็บข้อมูลบางส่วนบนจอมอนิเตอร์ลงไว้ในหน่วยความจำชั่วคราวเสียก่อนแล้วทำการถามชื่อไฟล์ เมื่อป้อนชื่อไฟล์เสร็จแล้ว จึงนำข้อมูลที่เก็บไว้ในหน่วยความจำชั่วคราวออกมาแสดงผลใหม่ ณ.ตำแหน่งเดิม

การเก็บภาพกราฟฟิก



```
/*---------- PROGRAM  SAVE  IMAGE ----------*/

void Save()
 {
  char fname[15];
  FILE *fp;
  register int i,j;
  char far *ptr = (char far *)0xA0000000;        /* พอย์เตอร์ชี้ไปที่ EGA memory */
  unsigned char buff[200][200];
  Flash2=SaveImage(200,220,458,260,0);          /* เก็บภาพตำแหน่งที่จะถามชื่อไฟล์ */
  rectangle(200,220,458,260);
  goto_xy(15,27);
  printf("FILE NAME : ");                       /* ถามชื่อไฟล์ */
  gets(fname);
  putimage(200,220,Flash2,COPY_PUT);            /* คืนหน้าจอ */
  free(Flash2);
  Rescreen();
 if(!(fp=fopen(fname,"wb")))                    /* เปิดไฟล์ */
   { Flash2=SaveImage(200,220,458,260,0);
     rectangle(200,220,458,260);
     outtextxy(210,240,"cannot open file...");  /* เปิดไฟล์ไม่สำเร็จพิมพ์ข้อความ */
     getch();
     putimage(200,220,Flash2,COPY_PUT);
     free(Flash2);
     return;
   }
 for(i=0;i<200;i++)                             /* เก็บภาพลงใน Buffer */
  for(j=0;j<192;j++)
   { buff[i][j]=*ptr;                           /* เก็บข้อมูลจาก Buffer ลงดิสก์*/
     ptr++;
   }
 for(i=0;i<200;i++)
   for(j=0;j<192;j++)
    { putc(buff[i][j],fp);
    }
  fclose(fp);
}
```

การโหลดภาพจากไฟล์ในดิสก์



```
/*---------------- PROGRAM     LOAD    IMAGE ----------------*/

void Load()
 {
   char fname[80]:
   FILE *fp:
   register int i,j:
   char far *ptr = (char far *)0xA0000000;        /*  พอย์เตอร์ชี้ไปที่ EGA memory  */
   Flash2=SaveImage(200.220,458.260.0):          /*  เก็บภาพตำแหน่งที่จะถามชื่อไฟล์ */
   rectangle(200,220,458,260);
   goto_xy(15,27);
   printf("FILE NAME : "):                          /*  ถามชื่อไฟล์  */
   gets(fname):
   putimage(200,220,Flash2,COPY_PUT);          /*  คืนหน้าจอ  */
   free(Flash2);
  if(!(fp=fopen(fname."rb")))                        /* เปิดไฟล์  */
   { Flash2=SaveImage(200.220.458.260.0):
     rectangle(200.220.458.260):
     outtextxy(210,240,"cannot open file...");     /*  เปิดไฟล์ไม่สำเร็จพิมพ์ข้อความ  */
     getch();
     putimage(200.220,Flash2,COPY_PUT);
     free(Flash2):
     return:
   }
        for(i=0;i<200;i++)                          /*  อ่านข้อมูลจากไฟล์ไปแสดงที่หน้าจอ  */
            for(j=0;j<192;j++)
        {
            *ptr=getc(fp);
           ptr++;
        }
     fclose(fp);
}
```

3.4 การใช้เครื่องพิมพ์ Epson Dot - Matrix

ใช้ ASCII Code เครื่องพิมพ์ Epson FX-85 เป็นมาตรฐาน ทำงานในโหมดกราฟฟิกมี 8 โหมดดังแสดงในตาราง

Graphic FX-85 Graphics Mode

MODE	DENSITY	DESCRIPTION	SPEED
0	Single	60 dpi	16 in/sec
1	Low speed double	120 dpi	8 in/sec
2	High speed double	120 dpi	16 in/sec(1)
3	Quadruple	240 dpi	8 in/sec(1)
4	CRT I	80 dpi	8 in/sec(2)
5	One-to-One	72 dpi	12 in/sec(2)
6	CRT II	90 dpi	8 in/sec
7	Dual-Density	144 dpi	3 in/sec(3)

Portriat และ Landscape

ในโปรแกรมนี้สามารถเลือกรูปแบบการพิมพ์ 2 อย่างคือ Lanscape หรือ Portrait ถ้าเลือกพิมพ์แบบ Portrait แกน X ของจอภาพจะสัมพันธ์กันกับความกว้างของกระดาษ และแกน Y จะสัมพันธ์กับความยาวของกระดาษ ถ้าเลือกแบบ Landscape แกน X ของจอภาพ จะสัมพันธ์กับความยาวของกระดาษและแกน Y จะสัมพันธ์กับความกว้างของกระดาษ

การเลือกพิมพ์แต่ละโหมดจะเข้ากันได้ดีกับโหมดจอภาพตามตาราง

Prefered Dot- Matrix Modes For Graphics

Direction	CGA	EGA	VGA
Portrait	3/7	1/7	1/7
Landscape	0	0	5

การเลือกโหมดพิมภาพกราฟฟิกพิจารณาหลัก 3 ประการดังนี้

ประการแรก จำนวน dots per inch (dpi) ต้องสูงเพียงพอ ใช้ความละเอียด 60 dpi (mode 0) สำหรับ Portrait และความกว้างของกระดาษต้องใหญ่กว่า 8 นิ้ว ถ้าจุดภาพทางแนวนอนเท่ากับ 640 ต้องใช้กระดาษกว้าง 13 นิ้ว และ 480 จุดภาพใช้กระดาษกว้าง 8 นิ้ว

ประการที่สอง dots per inch ทางแนวนอน และบรรทัดต่อนิ้วทางแนวตั้งต้องให้สมดุย์กับจอภาพ dots per inch ทางแนวนอนสามารถเปลี่ยนแปลงได้ แต่ทางแนวตั้งจะเปลี่ยนแปลงได้ยาก

ประการที่สาม การพล็อตจุด จะพล็อตหนึ่งจุดภาพต่อหนึ่ง dot ถ้าความเข้มในการพิมพ์สูงมาก ภาพจะเล็กลง เช่น โหมด 3 จะพล็อตภาพใน Portrait โหมดกว้างเพียง 2.7 นิ้ว

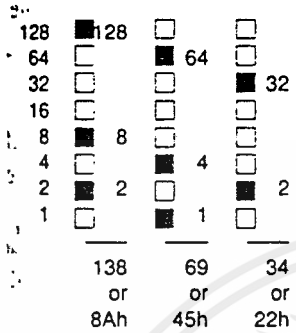
ข้อควรจำ ถ้าจำนวน dot per inch (dpi) สูงจะทำให้ภาพทางแนวแกนนอนเล็กลง ถ้าต้องการให้ภาพมีความเข้มสูง และภาพใหญ่ จะต้องพิมพ์ 2 dot ต่อจุดภาพ

Caculating The Dot-Matrix Graphics Character

เมื่อใช้เครื่องพิมพ์ dot-matrix พิมพ์ตัวอักษร 8 บิท Character Code เลือกรูปแบบ 9 x 9 จาก ROM Character sets ของเครื่องพิมพ์

ในกราฟฟิกโหมดจะส่ง Charater แต่ละตำแหน่งทางแนวนอน ครั้งละ 8 จุด แต่ละจุดจะควบคุมให้เข็มพิมพ์ทำการพิมพ์ตามจุดภาพที่ปรากฏบนจอ โดยตำแน่งบิทใด ON ก็จะควบคุมให้เข็มพิมพ์พิมพ์จุด

Graphic Character Code รูปแบบเป็นดังรูป โดยบิท 7 จะอยู่บนสุด บิทล่างสุดจะเป็นบิท 0

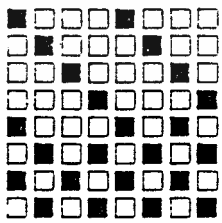


The graphics character code for the dot-matrix print head is calculated as an 8-bit character with bit 7 for the top pin (value 128) and bit 0 for the bottom pin (value 1). The character code sent is the sum of the values for the pins desired to print.

Graphics results are shown here in both decimal and hexadecimal codes.

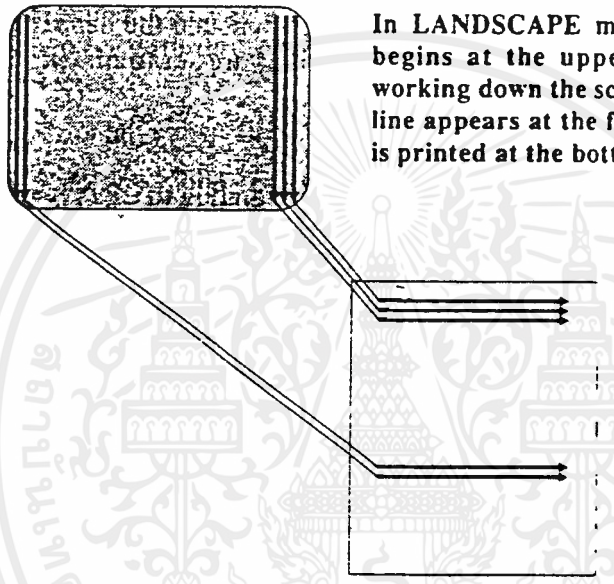
จากรูปเป็น Graphic Character Code เรียง 8 แถวตามแนวนอน การพิมพ์จะพิมพ์ครั้งละ 8 จุด ดังนั้นในโหมด Portrait แต่ละบรรทัดจะนับจากจุดบนลงไป 8 จุด พิมพ์จากซ้ายสุดของจอภาพ ไปจนถึงขวาสุด แล้วกลับไปขึ้นบรรทัดใหม่นับลงไปอีก 8 จุด และแต่ละ Character Code เกิดจากจุดตามแนวตั้ง

ในโหมด Landscape ทิศทางการพิมพ์จะพิมพ์จากมุมบนขวาสุดของจอภาพ แถวละ 8 จุดภาพนับจากขวาไปซ้าย พิมพ์ลงไปครั้งละแถวจนถึงล่างสุดของจอภาพ และจะพิมพ์ลงบนกระดาษตามด้านกว้างจากบนซ้ายไปขวา บรรทัดละ 8 จุดภาพ แล้วจะกลับไปตำแหน่งบนสุดของจอภาพ column ถัดไป เช่นนี้ไปเรื่อยๆจนถึง column ซ้ายสุดของจอภาพลงไปจนถึงล่างสุด ลักษณะBit image character ของแต่ละโหมดเป็นดังรูป



In PORTRAIT mode, the pixels would be read as eight vertical sets (left to right) as 8Ah, 45h, 22h,15h, 8Ah, 45h, 2Ah and 15h and transmitted in this order.

In LANDSCAPE mode, the pixels would be read as eight horizontal sets (top to bottom) as 11h, 22h, 44h, 88h, 51h, AAh, 55h and AAh and tranmitted in this order. Remember, the high order bit is to the right, the low order bit to the left.



In LANDSCAPE mode, the first scan line begins at the upper right of the screen, working down the screen while the final scan line appears at the far left of the screen and is printed at the bottom of the paper image.

This rotation matches the long axis of the screen with the long axis of the paper, providing a better appearance and allowing a larger image to be printed.

3.5 โปรแกรม Print_Graph

ก่อนอื่นกำหนดค่าคงที่ให้สำหรับแต่ละโหมด

```
# define PORTRAIT 0
# define LANSCAPE 1
```

ฟังก์ชันต้องการผ่านค่าพารามิเตอร์ Mode เพื่อเซ็ทกราฟฟิกโหมด

```
void Print_Graph(int Mode)
{
        char m;
        int i,j,k,Msb,Lsb,
                Max X = getmaxx(),
                Max Y = getmaxy(),
```

ตัวแปร MaxX และ MaxY จะเป็นที่เก็บค่าความละเอียดของจุดภาพทางแกน X และแกน Y ซึ่งได้จากฟังก์ชัน getmaxx() และ getmaxy()

```
        fprintf (stdprn,"\x1B\x2B%c",50);
```

String \x1B\x2Bn เป็นการสั่งให้เครื่องพิมพ์เซ็ทบรรทัดเป็น 50/360 (ค่า n มีค่าอยู่ระหว่าง 0-225) การเลือกพิมพ์โหมด Protrait หรือ Landscape รับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปร Direction

```
        switch (Direction)
        {
                case PORTRIAT:
                {
```

Mode Portriat จอภาพ EGA ในโหมดละเอียดสูง 640 x 480 ทางแนวนอน 640 จุด แต่เครื่องพิมพ์ Epson FX-85 รับข้อมูลพิมพ์ครั้งละ 8 บิทดังนั้นจะต้องหาค่า Lsb และ Msb ได้ตามสมการ

```
                        Lsb = (MaxX+1)& 0x00FF;
                        Msb = (MaxX+1)>>8 ;
```

และลูปเริ่มต้นจากบนสุดของจอภาพ ในแต่ละลูปจะพล็อตจุด 8 จุดทางแนวตั้ง

```
                        for(j=0;j<=MaxY/8;j++)
                        {
```

Mode,Lsb และMsb จะผ่านค่า Char (unsigned 8 bit Integers) ไปให้กับ \x1B* เป็นการเลือกกราฟฟิกโหมด

```
        fprintf(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
```

ลูปจะเริ่มต้นจากซ้ายสุดของจอภาพไปจนถึงขวาสุดของจอภาพที่จุด MaxX

```
        for(i=0;i<=MaxX;i++)
         {
```

m จะถูกเซ็ทให้เป็น 0 ในตอนเริ่มต้นของแต่ละสเต็ปภายในลูปเริ่มจาก 0 ถึง 7 เป็นการอ่านค่า 8 จุดบนจอภาพทางแนวตั้ง

```
        m=0;
        for(k=0;k<8;k++)
        {
```

การทำงานในแต่ละลูป m จะshift ซ้ายไป 1 บิต ถ้าตำแหน่งจุดในขณะนั้น ไม่เป็น 0 แล้ว (ไม่เป็นสีดำหรือสีพื้น) แล้ว m จะเพิ่มค่าขึ้นหนึ่ง (ขวาสุดหรือบิต 0 จะถูกเซ็ท) ลูปต่อไป m จะถูก shift ซ้าย แต่ละบิทจะเลื่อนขึ้น จนลูปครบ 8 ครั้งนั่นคือ shift m ไปทางซ้าย 1 บิต ถ้าจุดภาพใด on จะเพิ่มขึ้น 1 หลังจากลูปครบ 8 ครั้งจะได้ charater code ส่งไปให้เครื่องพิมพ์

```
fprintf(stdprn,"%c",m);
}
```

เมื่อจบบรรทัด CR/LF Code จะถูกส่งไปบอกเครื่องพิมพ์ให้ไปเริ่มต้นที่บรรทัดต่อไป

```
fprintf(stdprn,"\x0D\x0A");
        }
}
```

การทำงานจะต่อเนื่องไปจนกระทั่ง ครบจำนวนบรรทัด (MaxY/8) นั่นคือ ภาพหน้าจอถูกส่งไปทำการพิมพ์หมดแล้ว

Mode Landscape

การทำงานคล้ายๆกันแต่ต่างกันที่การคำนวนความยาวของ String คำนวนจาก MaxY แทนที่จะใช้ MaxX และการอ่านจากจอภาพ จะอ่านจากบนขวาเลื่อนลงและอ่าน column ต่อๆไปทางซ้ายครั้งละ 8 จุดภาพทางแนวนอน

```
Case LANDSCAPE:
{
        Lsb = MaxY&0x00FF;
        Msb = MaxY>>8 ;
        for(j=0;j<MaxX;+=8)
    {
        fprint(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
```

ไม่เหมือนกับ Portrait Mode ในที่นี้การอ่านจะเริ่มต้นจากล่างสุดและเพิ่มขึ้นเรื่อยๆ

```
for(i=MaxY;i>=0;i--)
{
    m=0
  for(k=0;k<8;k++)
  {
        m<<=1;
        if(getpixel(j+k,i))m++;
  }
        fprintf(stdprn,"%c",m);
}
  fprintf(stdprn,"\x0D\x0A);
 }
}
}
```

สุดท้ายส่งค่า Character Code จากจอภาพไปให้เครื่องพิมพ์หมดแล้ว ส่งรหัสเพื่อเลื่อนกระดาษไปให้เครื่องพิมพ์

```
        fprintf(stdprn,"\f");
}
```

```
/*----------------FUNCTION PRINT GRAPHIC IMAGE----------*/

void Print_Graph(int Mode)
{
 char m,chr;
 int i,j,k,Msb,Lsb,
        MaxX=getmaxx(),
        MaxY=getmaxy(),
        Direction;
 do
    {
      MenuPrint=SaveImage(200,180,458,300,0);
      rectangle(200,180,458,300);
      rectangle(202,182,456,298);
     outtextxy(240,210,"SELECT MODE (0 OR 1)");
     outtextxy(270,240,"0. PORTRAIT");
     outtextxy(270,260,"1. LANDSCAPE");
     chr=getch();
     switch(chr)
        {
         case '0': Direction=0;
                   break;
         case '1': Direction=1;
                  break;
        }
}
while(chr<'0'||chr>'1');
 setviewport(0,0,MaxX,MaxY,0);
 putimage(200,180,MenuPrint,COPY_PUT);
 free(MenuPrint);
 fprintf(stdprn,"\x1B\x2B%c",50);          /*sets line spacing to 50/360 inch*/
 switch(Direction)
 {
    case PORTRAIT:
      {
       Lsb=MaxX+1 & 0x00FF;                /*MaxX modulo 256*/
       Msb=MaxX+1 >> 8;                    /*(int)MaxX/256*/
       for(j=0;j<=MaxY/8;j++)
        {
          fprintf(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
          for(i=0;i<=MaxX;i++)
            {
              m=0;
              for(k=0;k<8;k++)
                  {                          /*shift m left one bit*/
                     m<<=1;
                     if(getpixel(i,j*8+k))m++;   /*if pixel on,bit on*/
                  }
              fprintf(stdprn,"%c",m);
            }
          fprintf(stdprn,"\x0D\x0A");        /*use CR/LF codes vs\n flag*/
        }
      }
   case LANDSCAPE:
{
 Lsb=MaxY+1 & 0x00FF;                       /*MaxY modulo 256*/
 Msb=MaxY+1>>8;                             /*(int)MaxY/256*/
```

```
    for(j=0;j<MaxX;j+=8)
        {
          fprintf(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
          for(i=MaxY;i>=0;i--)
            {
                m=0;
                for(k=0;k<8;k++)
                  {
                      m<<=1;                          /*shift m left one bit*/
                      if(getpixel(j+k,i))m++;         /*if pixel on,set bit*/
                  }
                fprintf(stdprn,"%c",m);
            }
          fprintf(stdprn,"\x0D\x0A");                 /*use CR/LF codes vs\n flag*/
        }
    }
  }
fprintf(stdprn,"\f");                                 /*form feed to advance paper*/
}
```

3.6 มาตรฐาน RS-232

พอร์ทแบบอนุกรมส่วนใหญ่ จะมีรูปร่างขึ้นอยู่กับมาตรฐานของ RS-232 คือมีขา 25 ขา ที่คอนเน็กเตอร์แต่ละปลายสายส่ง แต่เครื่องคอมพิวเตอร์รุ่นใหม่มีทั้ง 25 และ 9 ขา

ลักษณะสัญญาณพื้นฐานของ RS-232

สัญญาณ	ชื่อย่อ	หมายเลขขา
REQUEST TO SEND	RTS	4
CLEAR TO SEND	CTS	5
DATA SET READY	DSR	6
DATA TERMINAL READY	DTR	20
TRANSMIT DATA	TxD	2
RECEIVE DATA	RxD	3
GROUND	GRD	7

ความผิดพลาดของกรอบข้อมูล

ความผิดพลาดของกรอบข้อมูล (framimg error) คือ ความผิดพลาดของการส่งข้อมูลที่เกิดจากการสัญญาณนาฬิกา (Clock) ที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านมีค่าไม่เท่ากัน เพราะว่าจากการทำงานของพอร์ทอนุกรม เมื่อพอร์ทได้รับบิตเริ่มต้นก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบ เพื่ออ่านบิตต่อไป ซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบกำหนดได้จาก Baud rate ถ้าหากคอมพิวเตอร์ทั้ง 2 เครื่อง มีสัญญาณนาฬิกาไม่ตรงกัน คอมพิวเตอร์ด้านรับก็จะอ่านข้อมูลจากส่วนรับข้อมูลของตน ช้าเกินไปหรือเร็วเกินไป ก่อนที่ข้อมูลจะถูกส่งมาจากคอมพิวเตอร์ ด้านส่งทำให้เกิด Framing error ขึ้น

ฮาร์ดแวร์แฮนด์เชคกิ้ง (Hardware handshaking) คือ วิธีที่ใช้ในการรับส่งข้อมูลแบบอนุกรมผ่านพอร์ทอนุกรม โดยจะต้องตรวจดูสถานะของคอมพิวเตอร์ด้านของการรับข้อมูลว่าพร้อมจะรับข้อมูลหรือไม่ เมื่อคอมพิวเตอร์ด้านส่งพร้อมจะส่งข้อมูลให้ ดังนั้นข้อมูลจะไม่ถูกส่งออกไปจนกว่าสัญญาณพร้อมรับข้อมูลจะถูกส่งกลับมาจากคอมพิวเตอร์ด้านรับ สัญญาณพร้อมรับข้อมูลของคอมพิวเตอร์ด้านรับคือ clear to send (CTS)

โปรแกรมจำลองการรับส่งข้อมูล จะมีรูปแบบดังนี้คือ

```
do {
    while (not CTS) wait;
    send (byte) ;
}  while  (bytes to send );
```

หมายความว่า จะมีการตรวจสถานะ CTS ตลอดเวลาว่า พร้อมจะรับข้อมูลหรือไม่ ถ้าไม่พร้อม ก็จะรอไปเรื่อยๆ แต่ถ้าพร้อมก็จะส่งข้อมูลไปให้ จะทำเช่นนี้ไปเรื่อยๆ ตลอดที่ยังมีข้อมูลที่จะต้องส่งอยู่

ปัญหาของการสื่อสาร

เพื่อที่จะให้การสื่อสารเป็นไปอย่างถูกต้อง สัญญาณหลายๆ สัญญาณถูกใช้เพื่อตรวจดูว่าข้อมูลจะพร้อมเมื่อใดหรือข้อมูลไบท์ต่อไปจะถูกส่งมาเมื่อใด แต่ต่อมาเมื่อมีการสื่อสารผ่านคอมพิวเตอร์ สัญญาณบางสัญญาณได้ถูกตัดทิ้งไป เพื่อว่าสัญญาณจะได้ลดน้อยลง และลดค่าใช้จ่า

เกี่ยวกับสายส่ง สัญญาณจึงเหลือเพียง GRD, TxD และ RxD ซึ่งในทางทฤษฎีแล้ว เมื่อคอมพิวเตอร์เครื่องหนึ่งพร้อมที่จะส่งข้อมูล แต่เมื่อลดสายสัญญาณลงแล้ว จะทำให้เกิดปัญหาตามมามากมายที่ยุ่งยากปัญหาหนึ่งก็คือ ปัญหาของข้อมูลถูกเขียนทับ (overrun error)

ปัญหาข้อมูลถูกเขียนทับ

เมื่อการติดต่อผ่านพอร์ทอนุกรมใช้สายส่งเพียง 2 สายนั้นจะต้องใช้กรรมวิธีพิเศษเล็กน้อย เพื่อให้พอร์ทตัวส่งเข้าใจว่าพอร์ทด้านรับพร้อมที่จะรับข้อมูลเสมอ โดยการต่อขา 6 , 8 และขา 20 ของคอนเน็กเตอร์เข้าด้วยกัน วิธีการเช่นนี้เป็นการตัดฮาร์ดแวร์แฮนด์เชคกิ้งไปนั่นเอง

แต่การทำเช่นนี้จะทำให้เกิดปัญหาข้อมูลถูกเขียนทับได้ง่าย ซึ่งก็คือความผิดพลาดในการรับส่งข้อมูลอย่างหนึ่ง ที่เกิดขึ้นจากการที่คอมพิวเตอร์เครื่องส่ง ส่งข้อมูลใหม่มาให้คอมพิวเตอร์ด้านรับ ในขณะที่คอมพิวเตอร์ด้านรับยังไม่พร้อมจะรับข้อมูล เนื่องจากในขณะนั้นข้อมูลเดิมยังไม่ได้ถูกอ่านเข้าไปเก็บ แต่ข้อมูลใหม่ก็ถูกส่งเข้ามาแล้ว ข้อมูลเดิมจึงถูกเขียนทับไป จึงเกิดความผิดพลาดของข้อมูลขึ้น

การใช้พอร์ทอนุกรมผ่าน BIOS

การใช้งานพอร์ทอนุกรมสามารถทำได้ 3 วิธีคือ ผ่านทาง DOS ผ่านทาง BIOS และสุดท้ายคือ การเขียนโปรแกรมควบคุมพอร์ทอนุกรมโดยตรง การเรียกใช้พอร์ทอนุกรมนั้น ไม่เหมาะสมเท่าใดนัก เพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะของการรับส่งและตัวพอร์ทอนุกรมว่าการรับส่งข้อมูลถูกต้องหรือไม่เพียงใด การเขียนโปรแกรมควบคุมการทำงานของพอร์ทอนุกรมโดยตรงนั้นคงไม่จำเป็นเพราะมีวิธีที่ดีกว่าและง่ายกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพต์หมายเลข 14

3.7 การเตรียมสถานะเริ่มต้นของพอร์ท

ในการเตรียมสถานะเริ่มต้นของพอร์ทอนุกรมเราสามารถทำได้โดยผ่านอินเตอร์รัพท์หมายเลข 14 Function หมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังชั่น รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ทอนุกรม โดยมีความหมายของแต่ละบิตดังตาราง

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตราส่งข้อมูล
	000 = 110 baud
	001 = 150 baud
	010 = 300 baud
	011 = 600 baud
	100 = 1200 boud
	110 = 9600 baud
4,3	พาริตี้
	00 หรือ 10 = ไม่มีพาริตี้
	01 = พาริตี้คี่
	11 = พาริตี้คู่
2	จำนวนของบิตสิ้นสุด
	0 = 1 บิตสิ้นสุด
	1 = 2 บิตสิ้นสุด
1,0	จำนวนบิตในข้อมูล : ไบต์
	10 = 7 บิต
	11 = 8 บิต

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ทอนุกรมมากถึง 7 พอร์ทโดยหมายเลขที่จะสามารถกำหนดผ่าน รีจิสเตอร์ DX พอร์ทแรกมีหมายเลข 0 พอร์ทต่อไปหมายเลข 1 เช่น นี้ต่อไปเรื่อยๆ ฟังก์ชั่นต่อไปนี้มีชื่อว่า init_port() มีหน้าที่เตรียมสถานะของพอร์ทในระบบ

```
/* ฟังก์ชั่นนี้ทำหน้าที่เตรียมสถานะเริ่มต้นของพอร์ท */
void port_init(port,code)
int port;
unsigned char code;
{
        union REGS r;

        r.x.dx = port;   /* หมายเลขพอร์ทอนุกรม */
        r.h.ah = 0;      /* เรียกฟังก์ชั่นหมายเลข 0 ทำหน้าที่เตรียมสถานะ */
        r.h.al = code; /* รหัสตั้งสถานะเริ่มต้น */
        int86 (0x14,&r,&r);
}
```

3.8 การส่งข้อมูลผ่านพอร์ทอนุกรม 1 ไบต์

อินเตอร์รัพท์หมายเลข 14 ฟังก์ชั่นหมายเลข 1 ของ BIOS ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ทอนุกรม หมายเลขของพอร์ทอนุกรมที่จะทำการส่งข้อมูล สามารถกำหนดผ่านรีจิสเตอร์DXและข้อมูลที่ต้องการส่งจะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเสร็จเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่า การส่งข้อมูลถูกต้องหรือไม่

รายละเอียดของฟังก์ชั่น sport( ) มีดังนี้

```
/* ฟังก์ชั่น sport() ทำหน้าที่ส่งตัวอักษรออกทางพอร์ทอนุกรม */
void sport (port,c)
int port;       /* หมายเลขของพอร์ท */
char c;         /* ตัวอักษรที่ต้องการส่ง */
{
        union REGS r;

        r.x.dx = port;  /* กำหนดหมายเลขพอร์ท */
        r.h.al = c;     /* ตัวอักษรที่ต้องการส่ง */
        r.h.ah = 1;     /* ฟังก์ชั่นหมายเลข 1 ทำหน้าที่ส่งข้อมูล 1 ไบต์ */
        int86 (0x14,&r&r);
        if(r.h.ah & 128) {   /* ตรวจดูบิทที่ 7 */
```

```
            printf("send error detected in serial port");
            exit(1);
        }
}
```

ถ้าบิทที่ 7 ของรีจิสเตอร์ AH มีค่า 1 เมื่อส่งข้อมูลเสร็จสิ้น แสดงว่าเกิดความผิดพลาดในการส่งข้อมูลขึ้น จะต้องตรวจดูสถานะของพอร์ทเพื่อดูว่าเกิดความผิดพลาดอะไร ดังในหัวข้อต่อไป

3.9 การตรวจสอบสถานะของพอร์ทอนุกรม

ฟังก์ชั่นหมายเลข 3 ของอินเตอร์รัพต์หมายเลข 14 ของ BIOSใช้ตรวจสอบดูสถานะของพอร์ทอนุกรม รีจิสเตอร์ DXใช้สำหรับหมายเลขพอร์ท ที่จะตรวจสอบ สถานะของพอร์ทสามารถแปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL ดังตาราง

สถานะของสายส่ง (AH)

ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data Ready	0
Overrun error	1
Parity error	2
Framming error	3
Break-Detect Error	4
Transfer hold-register empty	5
Transfer shift-register empty	6
Time out error	7

สถานะของโมเด็ม (AL)

ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in-data-set-ready	1
Trialing-edge ring detector	2
Change in line signal	3
Clear-to-send	4
Clear-set-ready	5
Ring indicator	6
Line signal detector	7

จะเห็นว่าสัญญาณสถานะต่างๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดังนั้นเมื่อนำมาใช้กับอุปกรณ์อื่นจึงลดความสำคัญลง แต่ก็ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสามารถมากก็คือ Data Ready ซึ่งจะเป็นตัวบอกว่า เมื่อไหร่ที่ข้อมูลถูกรับเข้ามา และพร้อมที่จะถูกอ่านเข้าไปเก็บ ฟังก์ชั่นในหัวข้อต่อไปนี้มีชื่อว่า rport( ) มีหน้าที่อ่านข้อมูลจากพอร์ท โดยจะใช้สถานะ Data Ready นี้เป็นตัวบอกว่า ข้อมูลพร้อมที่จะถูกอ่านหรือยัง

3.10 การรับข้อมูลผ่านพอร์ทอนุกรม 1 ไบต์

การรับข้อมูลจากพอร์ทอนุกรม สามารถทำได้โดย เรียกผ่าน BIOS อินเตอร์รัพต์หมายเลข 14 ฟังห์ชั่นหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ท ข้อมูลที่อ่านได้จะเก็บในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ทจะสามารถตรวจดูได้ที่บิต 7 ของรีจิสเตอร์ AH

ต่อไปนี้คือฟังก์ชั่น rport () มีหน้าที่อ่านข้อมูลเข้า 1 ไบต์

```
/* ฟังก์ชั่นนี้ทำหน้าที่อ่านตัวอักษรจากพอร์ทอนุกรม */
rport (port)
int port;    /* หมายเลขพอร์ท */
{
        union REGS r;
        /* รอจนกว่าจะได้รับตัวอักษร */
        while(!check_stat(port)&256))
            if(kbhit())  {   /* ยกเลิกเมื่อมีการกดคีย์ */
                r.h.ch = getch;
                if(r.h.ch == 'q' )
            }

        r.x.dx = port; /* กำหนดหมายเลขพอร์ท */
        r.h.ah = 2 ;   /* ฟังก์ชั่นหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
        int86(0x14,&r&r);
        if(r.h.ah & 128)
           printf("read error detected in serial port");
        return r.h.al;
```

การทำงานของฟังก์ชั่นนี้คือ จะรอจนกระทั่งข้อมูลถูกรับมาผ่านพอร์ทอนุกรม แล้วส่งค่าอักษรกลับมา แต่ว่าการทำงานเช่นนี้อาจทำให้โปรแกรมไม่หลุดจากลูปการทำงาน เช่น เมื่อสายสงของพอร์ทเกิดเสีย ดังนั้นจึงต้องตรวจสถานะของพอร์ทอนุกรมเสียก่อน ดังที่ได้อธิบายในหัวข้อที่ผ่านมา ฟังก์ชั่น kbhit( ) ใช้ตรวจสอบการกดคีย์ ถ้าหากไม่มีข้อมูลผู้ใช้ก็สามารถกดคีย์ ' q ' เพื่อออกจากลูปได้ แต่ถ้ามีข้อมูลรับเข้ามา ฟังก์ชั่นก็จะผ่านไปเรียกอินเตอร์รัพเพื่ออ่านข้อมูลเข้ามา และเช่นเดียวกับการส่งข้อมูล บิต 7 ของรีจิสเตอร์ AH ใช้บอกว่า การอ่านข้อมูลมีข้อผิดพลาดหรือไม่

```
/* ---------- การเลื่อน CURSOR ---------- */

void goto_xy(int x,int y)
{
 union REGS r;
 r.h.ah=2:        /* cursor function */
 r.h.dl=y:        /* column coordinate */
 r.h.dh=x:        /* row coordinate *
 r.h.bh=0:        /* video page */
 int86(0x10.&r,&r):
}
```

3.11 ฟังก์ชั่น XMITT

ฟังก์ชั่นนี้จะส่งตัวอักษรออกทาง RS-232 แล้วรับค่า Azimuth , Elevate , dB และรับตัวอักษรที่ส่งออกไปกลับมาตรวจสอบ ถ้าไม่ตรงกันแสดงว่าเกิดการผิดพลาดในระหว่างรับ-ส่ง จะต้องส่งใหม่จนกว่าอักษรที่ส่งออกไปและรับ Echo กลับมาจะตรงกัน

```
xmit(unsigned code)
{
  do
    {
      sport(po,code);              /* ส่งตัวอักษรออกทาง RS-232 */
      azimuth = rport(po);         /* รับค่า Azimuth */
      elevate = rport(po);         /* รับค่า Elevate */
      dB = rport(po):              /* รับค่า dB */
    echo = rport(po);              /* รับตัวอักษรที่ส่งออกไปกลับมาตรวจสอบ */
    }
 while(echo != code);  /* ตรวจสอบตัวอักษรที่ส่งไปและรับกลับมาว่าตรงกันหรือไม่
}                      ถ้าไม่ตรงส่งใหม่ */
```

3.12 /* ----------การสร้างภาพ Background ----------*/

```
void Background(int r)
{
 int R;
 int value=0;
 rectangle(0,0,639,479);              /* สร้างกรอบภาพ */
 line(181,0,456,479);                 /* ลากเส้นที่องศาต่าง ๆ ตัดจุดศูนย์กลางภาพ */
 line(0,54,639,424);
 line(0,424,639,54);
 line(181,479,456,0);
 line(0,239,639,239);
 line(319,0,319,479);
 outtextxy(310,5,"0");                /* แสดงค่ามุมที่ตำแหน่งต่าง ๆ */
 circle(316,2,1);
 outtextxy(167,7,"30");
 circle(181,4,1);
 outtextxy(5,70,"60");
 circle(20,67,1);
 outtextxy(5,228,"90");
 circle(20,225,1);
 outtextxy(5,425,"120");
 circle(27,422,1);
 outtextxy(155,470,"150");
 circle(178,467,1);
 outtextxy(293,470,"180");
 circle(316,467,1);
 outtextxy(458,470,"210");
 circle(481,467,1);
 outtextxy(600,421,"240");
 circle(623,418,1);
 outtextxy(610,228,"270");
 circle(636,225,1);
 outtextxy(600,79,"300");
 circle(623,76,1);
 outtextxy(458,7,"330");
 circle(481,4,1);
for(R=r*zoom;R<400;R=R+r*zoom)        /* ทำวงกลมตามค่ารัศมีที่ผ่านให้ฟังก์ชั่น */
 {
  value=value+r;
  circle(319,239,R);
  outtextxy(322,(239-(R+10)),"dB");
 if(R<240){
  if(value<100)
  gprintxy(300,(239-(R+10)),"%d",value);
  else gprintxy(293,(239-(R+10)),"%d",value);
  }
 }
}
```

3.13 การหาตำแหน่ง X,Y เพื่อทำการพล็อตจุด

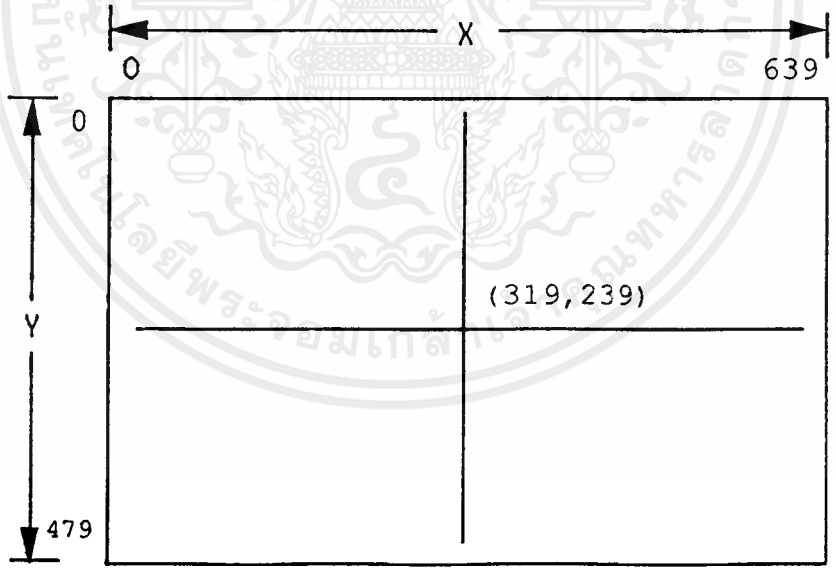
ค่าความเข้มของสนามไฟฟ้าที่อ่านได้จาก field strength meter และค่า Azimuth จะต้องนำมาคำนวณหาตำแหน่งที่จะพล็อตจุดบนมอนิเตอร์ เพื่อให้สัมพันธ์กันตามค่ามุมและความเข้มของสนามไฟฟ้า เป็น dB

Degree = Azimuth * Step

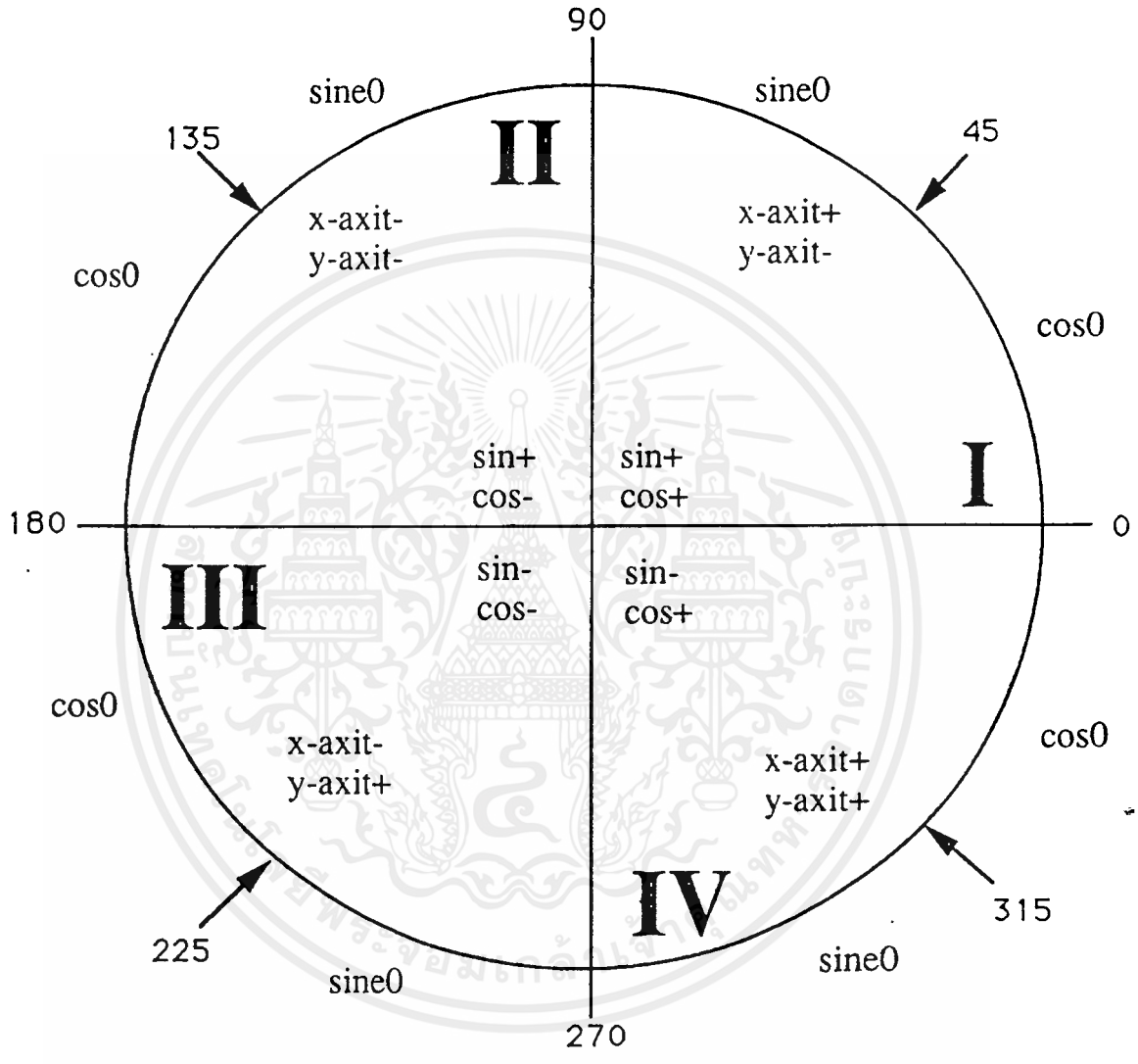
Degree เป็นค่ามุมที่ผ่านการปรับแล้วเพราะว่าค่ามุมที่ A / D อ่านได้จากโรเตอร์มีค่าเป็นไปได้ตั้งแต่ ' 0 - 255 (A / D ขนาด 8 bit) เพราะฉะนั้นจะต้องทำการปรับให้ได้เป็นมุมตั้งแต่ 0' - 360 ' โดยคูณด้วยค่าคงที่ 1.40626 จะได้ความละเอียดของมุมที่ทำการพล็อต 1.4026 องศา ค่าของมุมหน่วยเป็นองศาจะต้องทำการเปลี่ยนเป็นเรเดียน เพื่อเรียกใช้ฟังก์ชัน sin กับ cos ซึ่งอยู่ในแฟ้มข้อมูล math.h

Rad = (Pi * Degree/180);

การคำนวณหาตำแหน่ง X,Y



Sine / Cosine Angle Values



คำนวณหาตำแหน่ง X,Y ที่จะพล็อตจุดได้ดังสมการต่อไปนี้ โดยคูณกับค่า Zoom ด้วยเมื่อต้องการย่อส่วนหรือขยายภาพ

```
a = (dB * Zoom) * sin (rad);b=(dB * Zoom) * cos(rad);
```

ค่าที่ได้เป็นค่าจำนวนจริง มีจุดทศนิยมฉะนั้นจะต้องทำให้เป็นจำนวนเต็มเพื่อให้ถูกต้องตรงกับตำแหน่ง X,Y บนจอมอนิเตอร์

```
x=a;a=a-x;z=a*z;x=x+z;
y=b;b=b-y;b=z;y=y+z;
```

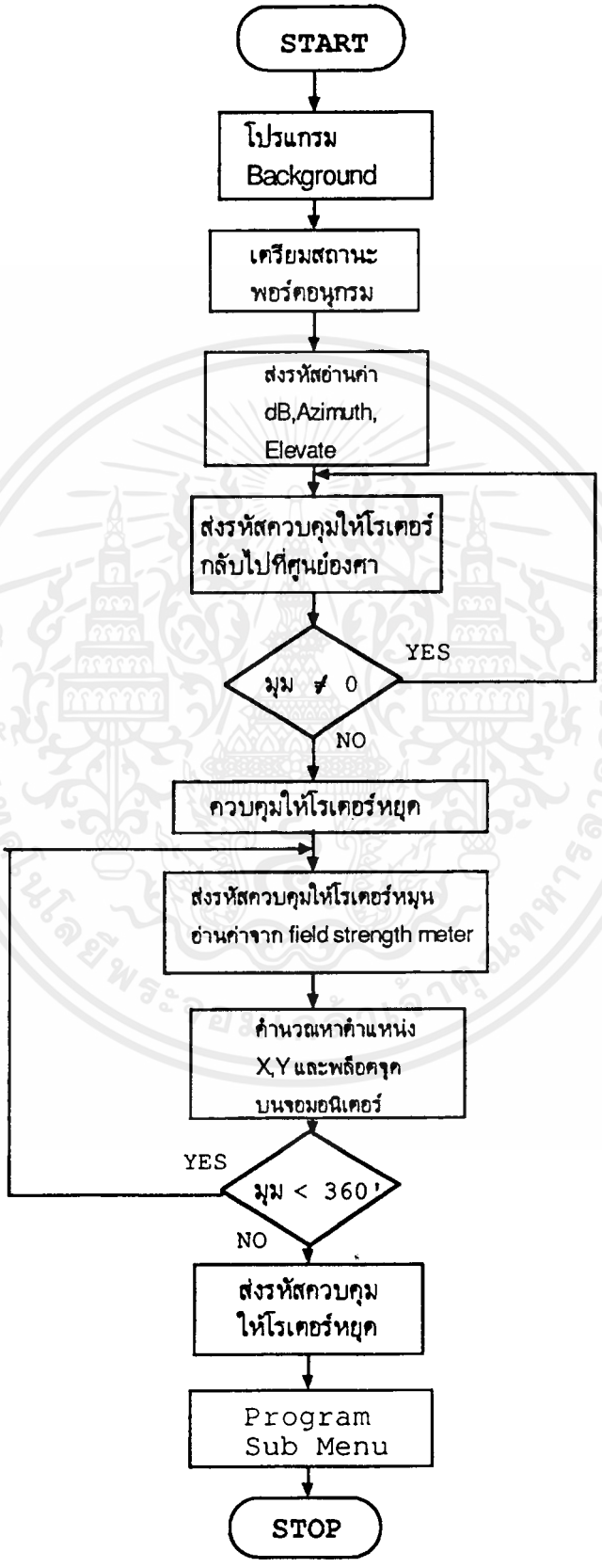
หลังจากนั้นทำการพล็อตจุดตามค่า X,Y ที่คำนวณได้

```
putpixel (319-x,239-y,white);
```

จุดศูนย์กลางอยู่ที่ตำแหน่ง (319,239) เมื่อขนาดของจอ 640 X 480 จุดที่พล็อตนั้นต้องการให้หางจากจุดศูนย์กลาง ออกไปตามค่า X,Y ฉะนั้นจุดทางแนวนอน (แกน X) ได้จาก 319-x และจุดทางแนวตั้งได้จาก 239-y

3.14 ฟังก์ชัน AUTOMATIC TEST

ฟังก์ชันนี้ทำหน้าที่ ทำการทดสอบแพทเทิร์นของสายอากาศโดยอัตโนมัติ โปรแกรมจะทำการเซ็ทให้โรเตอร์ไปเริ่มต้นที่มุมศูนย์องศาแล้วทำการควบคุมให้โรเตอร์หมุนไปจนครบ 360 องศา พร้อมทั้งอ่านค่าจาก field strength meter แล้วทำการพล็อตแพทเทิร์น จนได้เป็นแพทเทิร์นการแพร่กระจายคลื่นของสายอากาศเป็น Horizontal Patterns



```
/*----------FUNCTION AUTOMATIC TEST----------*/

AutoTest()
{
 int r,x,y,Size=0;
 register int i,j;
 char ch,ch1;
 char filename[15];
 double a,b,z,rad,Degree,DB;
 cleardevice();
 zoom=4;
 Background(10);                          /* โปรแกรม Background */
 free(screen1);
 free(screen2);
 free(screen3);
 free(screen4);
 chowtext();                              /* แสดงอักษรหน้าจอ */
 port_init(po,195);                       /* เตรียมสถานะพอร์ตอนุกรม */
 ch='a';ch1=1;                  /* ส่งรหัสอ่านค่า dB, Azimuth , Elevate */
 xmit(ch);
 xmit(ch1);
 do
   {
    ch=0;
    ch1=75;
    xmit(ch);                   /* ควบคุมให้ Rotor กลับไปที่ศูนย์องศา */
    xmit(ch1);
 }
while(azimuth!=0);
 OldAzimuth=azimuth;
 do
   {
    ch1=77;
    xmit(ch);          /* ควบคุมให้ Rotor หมุนอ่านค่า dB, Azimuth , Elevate */
    xmit(ch1);
   DB=dB*1.171875;                          /* ปรับ Scalling */
   Degree = azimuth*step;                   /* ทำองศาเป็นเรเดียน */
 rad=(pi*Degree/180);                       /* คำนวณหาตำแหน่ง X, Y */
 a=(DB*zoom)*sin(rad);b=(DB*zoom)*cos(rad);

                                  /*ถ้ามุมเปลี่ยนแปลงทำการพล็อตจุดบนจอ */
 x=a;a=a-x;z=a*2;x=x+z;
 y=b;b=b-y;z=b*2;y=y+z;
if(OldAzimuth!=azimuth)
 .circle(319-x,239-y,1);
                                  /* putpixel(319-x,239-y,WHITE);*/
 OldAzimuth=azimuth;
                                  /* Level();*/
 }
   while (azimuth<255);           /* ตรวจสอบมุมครบ 360 องศาหรือยัง */
   ch='a';ch1=1;
```

```
    xmit(ch);                          /* ส่งรหัสคำสั่งให้ Rotor หยุดหมุน */
    xmit(ch1);
    SubMenu();                         /* เข้าสู่โปรแกรม SubMenu */
}
  SubMenu()                            /* โปรแกรมเมนย่อย */
  {
    char ch;
    do
       {
         ch=getch();                   /* ตรวจสอบ Key */
         switch(tolower(ch))
            {
             case 'l': Load();         /* เรียกฟังก์ชั่น Load */
                       break;
             case 's': Save();         /* เรียกฟังก์ชั่น Save */
                       break;
             case 'd': SaveScreen();
                       system("DIR/W");          /* ดู Directory */
                       getch();
                      ReSaveScreen();
                      getch();
                      break;
            case 'm': break;
            case 'p': Rescreen();
                      Print_Graph(0);          /* เรียกฟังก์ชั่น Print Graph */
                      break;
            case 'q': closegraph();
                      exit(0);

            default : break;
            }
       }
while(ch!='m');                        /* กด M เข้าสู่ Main Menu */
}
```

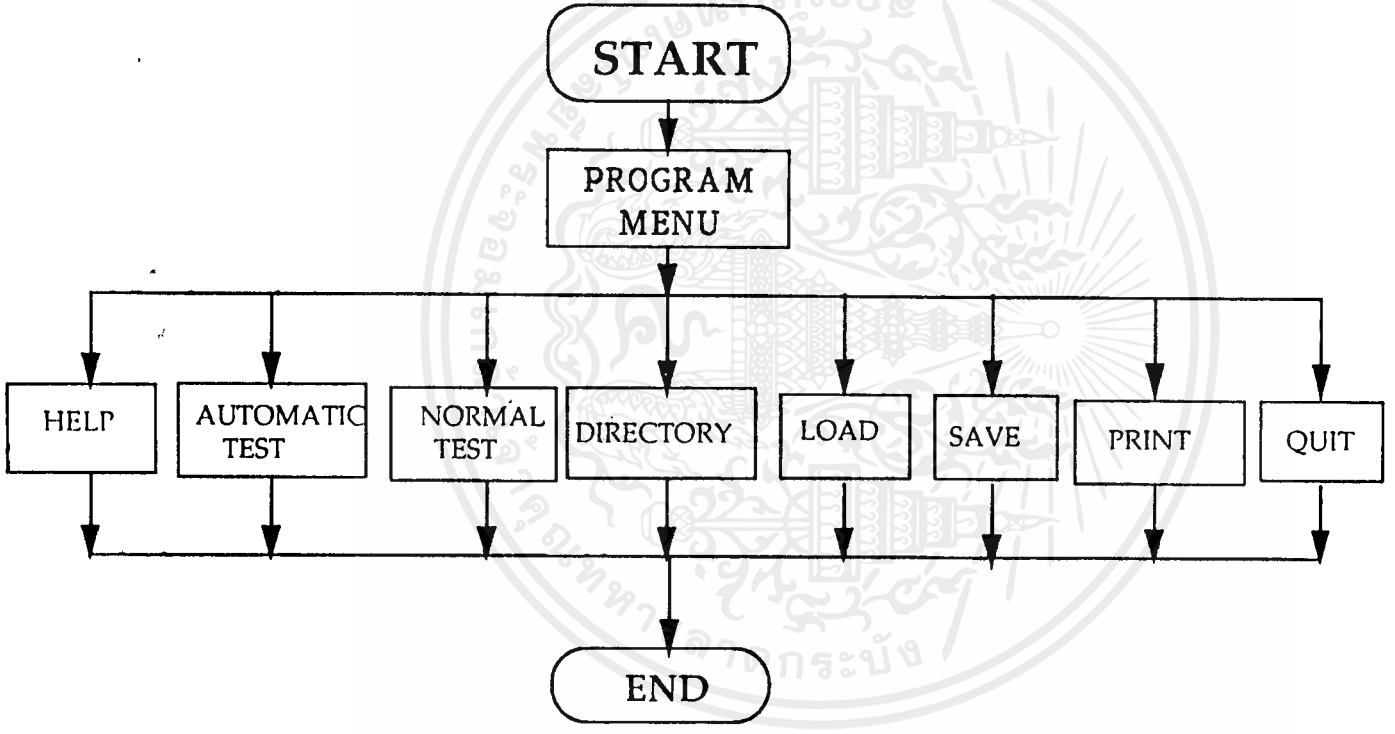
3.15 FUNCTION TEST

ฟังก์ชันนี้ทำหน้าที่ทดสอบแพทเทิร์นของสายอากาศ โดยอ่านค่าความเข้มของสนามไฟฟ้าจาก Field Strength Meter ทำการคำนวนหาตำแหน่ง X,Y และพล็อตจุดบนจอมอนิเตอร์ การควบคุมให้โรเตอร์หมุนเพื่ออ่านค่าความเข้มของสนามไฟฟ้าที่มุมต่างๆสามารถควบคุมที่ Keyboard ของเครื่องPersonal Computer โดยกด Key ← เมื่อต้องการให้โรเตอร์หมุนซ้าย กด Key → เมื่อต้องการให้โรเตอร์หมุนขวา กด Key ↑ เมื่อต้องการให้โรเตอร์ยกขึ้นกด Key ↓ เมื่อต้องการให้โรเตอร์ก้มลง การควบคุมให้หมุนซ้ายหรือหมุนขวาสามารถควบคุมให้หมุนได้ 0 - 360 การควบคุมให้ยกขึ้น หรือ ก้มลงสามารถควบคุมได้ 0 - 180 ขณะที่โปรแกรมทำงานอยู่บนฟังก์ชันนี้สามารถสั่ง Load,Save,หรือ Print ได้

```
/*----------FUNCTION TEST----------*/

Test()
{
   int r,x,y;
   char filename[15];
   double a,b,z,rad,Degree,DB;
   union k{
   int i;
  char c[2];
 }
 key;
 cleardevice();
 zoom=4;
 Background(10);                          /* สร้าง  Background  */
 free(screen1);
 free(screen2);
 free(screen3);
 free(screen4);
 chowtext();
 port_init(po,195);                       /*  เตรียมสถานะพอร์ตอนุกรม */

 key.c[0]='a';key.c[1]=1;
 xmit(key.c[0]);                  /* ส่งรหัสอ่านค่า dB,  Azimuth ,  Elevate */
 xmit(key.c[1]);
 do
  {
                                  /*Level();*/
    DB=dB*1.171875;
    Degree = azimuth*step;                    /* ทำองศาเป็นเรเดียน */
    rad=(pi*Degree/180);
    a=(DB*zoom)*sin(rad);b=(DB*zoom)*cos(rad);     /* คำนวณหาตำแหน่ง X, Y */
    x=a;a=a-x;z=a*2;x=x+z;
   y=b;b=b-y;z=b*2;y=y+z;                     /* ทำ X, Y ให้เป็นจำนวนเต็ม */
   if(OldAzimuth!=azimuth)
    {
      circle(319-x,239-y,1);
      putpixel(319-x,239-y,WHITE);
                                 /* ถ้ามุมไม่แปลี่ยนแปลงทำการพล็อตจุดบนจอ */
      OldAzimuth=azimuth;
    }
 if (kbhit())
 key.i=bioskey(0);                        /*  ตรวจสอบ Key */
 xmit(key.c[0]);                           /* ส่งรหัส Key ที่กด */
 xmit(key.c[1]);
 switch(key.c[1])                         /*  ตรวจสอบ Key  ลูกศร */
  { case 75:                               /* LEFT */
            if(azimuth==0)
                 {key.c[0]='a';
                  key.c[1]='1';}
                  break;
   case 77:                               /* RIGHT */
```

```
                {key.c[0]='a';
                 key.c[1]='1';}
                 break;
    case 77:                    /* RIGHT */
            if(azimuth==255)
                {key.c[0]='a';
                 key.c[1]='1';}
                 break;
    case 72:                    /* UP */
            if(elevate==0)
                {key.c[0]='a';
                 key.c[1]='1';}
                 break;
    case 80:                    /* DOWN */
            if(elevate==255)
                {key.c[0]='a';
                 key.c[1]='1';}
                 break;
    default: break;
   }
 switch(tolower(key.c[0]))
  { case 'l':  SaveScreen();
               Load();
               getch();
               ReSaveScreen();
               break;


    case 's':  Save();
               getch();
               break;
    case 'd':  SaveScreen();
               system("DIR/W");
          getch();
               ReSaveScreen();
               getch();
               break;

    case 'p':  Rescreen();
               Print_Graph(0);
               break;

    case 'm':  goto end:

    case 'q':  closegraph();
               exit(0);
    default :  break;
   }


}

while (azimuth<255):
end:xmit('a');
getch();

}
```

3.16 การทำงานของ Main Program

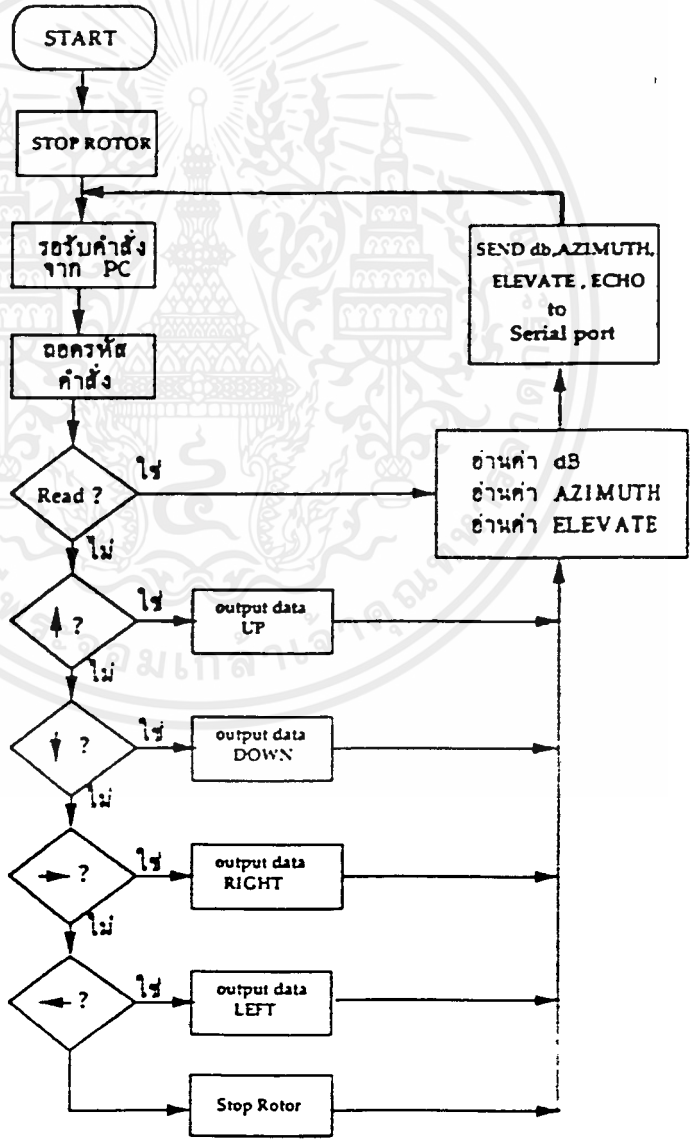


FLOW CHART MAINPROGRAM

```
/*---------------------- MAIN PROGRAM ----------------------*/

main()
{
 union k
    {
     char c[2];
     int i;
    }
 key;
 Initialize();                                  /* เข้าสู่ภาวะกราฟฟิกโหมด */
 SelectPort();                                  /* เลือกพอร์ตอนุกรม */
 Background(20);                                /* โปรแกรม Background */
 do
   {
    menu();                                     /* Menu โปรแกรม */
    key.i=bioskey(0);                           /* รอจนกว่าผู้ใช้กด Key */
    putimage(200,150,MemMenu,COPY_PUT);         /* คืนหน้าจอ */
    free(MemMenu);

          closegraph();
          system("DIR/W");                      /* ดู Directory */
           getch();
          Initialize();
          ReSaveScreen();
          break;
  case 'q': closegraph();                       /* จบโปรแกรม */
          exit(0);
  default : break;
 }
 }
 while(key.c[0]!='q');                          /* กด Q จบโปรแกรม */
 closegraph();
}
```

3.17 Program Serial Port Communication and Control Rotor

โปรแกรมนี้เป็นโปรแกรมในส่วนของชุดController ซึ่งโปรแกรมส่วนนี้จะอยู่ในEPROM โดยโปรแกรมนี้จะรับข้อมูลเข้ามาทางพอร์ตอนุกรม แล้วทำการตรวจสอบการกด Key ที่ Keyboardของ PC ถ้ากด Key ⬆ ให้ส่ง 0F1H ไปที่ พอร์ท 1 ซึ่งพอร์ท 1.1 จะควบคุม ROTOR ให้ UP ถ้ากด ⬇ ให้ส่ง OF2 Hไปที่ พอร์ท 1ซึ่ง พอร์ท 1.2 จะควบคุมให้ DOWN ถ้ากด Key ⬅ ให้ส่ง 0 F4H ไปที่ P_1 ซึ่ง $P_{1.2}$ จะควบคุมให้ ROTOR หมุนซ้าย ถ้ากด Key ➡ ให้ส่ง 0F8H ไปที่ P_1 ซึ่ง $P_{1.1}$ จะควบคุมให้ ROTOR หมุนขวา ถ้ากด Key อื่นๆ ให้ส่ง 0F0H ไปที่ Port 1 แล้วอ่านข้อมูลและ ECHO กลับส่งไปให้ PC ตรวจสอบความถูกต้องทุกครั้ง ซึ่งการส่งข้อมูลอนุกรมตั้งให้ทำงานใน Mode 1 โดย Baud Rate เท่ากับ 4800 bps

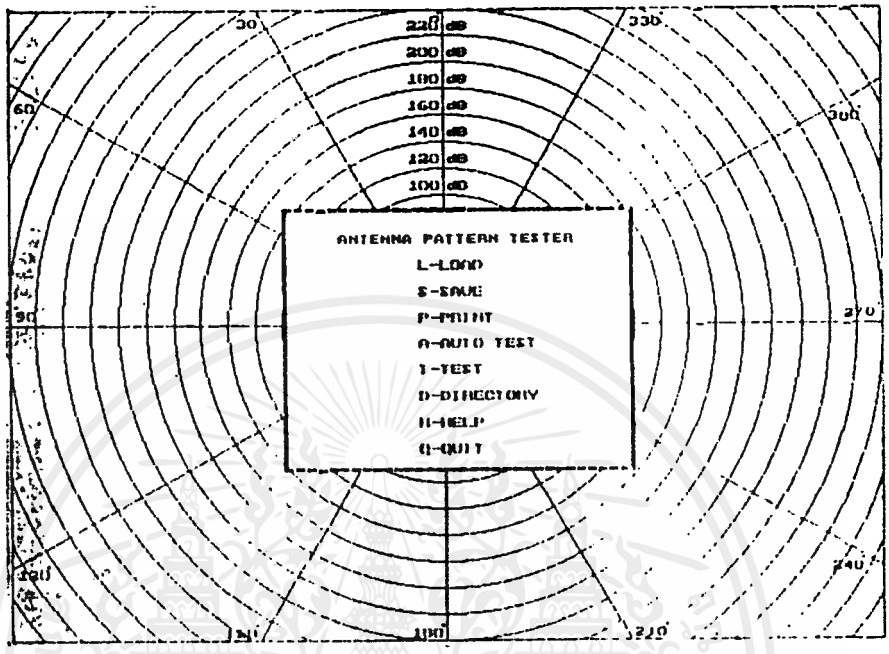


FLOW CHART OF CONTROLLER UNIT

การใช้โปรแกรม AntTest

4.1 การเข้าสู่โปรแกรม AntTest

A:>AntTest Enter



รูป MENU

เมนูประกอบด้วยคำสั่งต่างๆดังนี้

L - Load

S - Save

P - Print

A - Auto test

T - Test

D - Direction

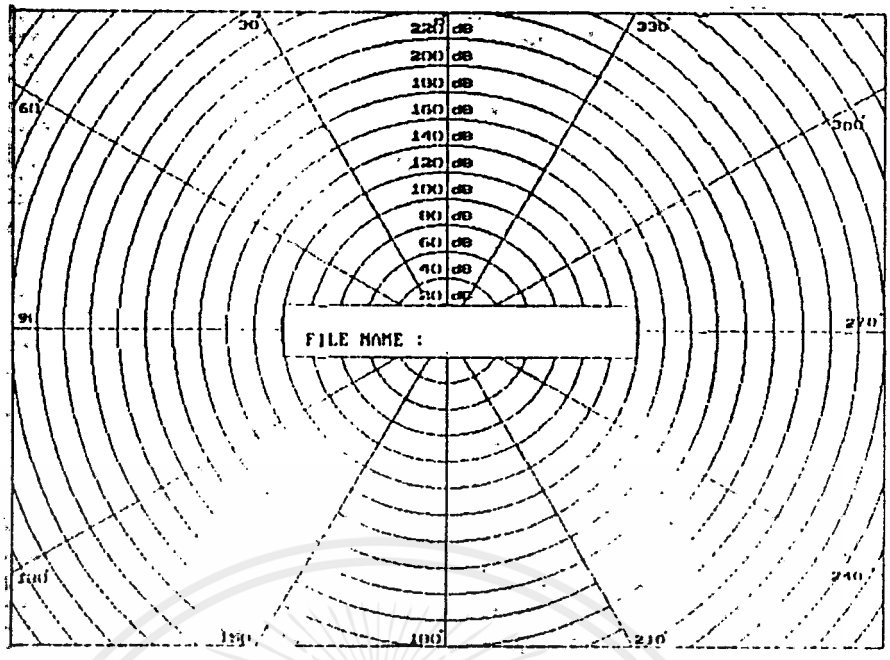
H - Help

Q - Quit

การเข้าสู่เมนูต่างๆทำได้โดยการกดอักษรนำหน้าโปรแกรมก็จะเข้าสู่การทำงานในแต่ละคำสั่ง

4.2 <u>คำสั่ง Load</u>

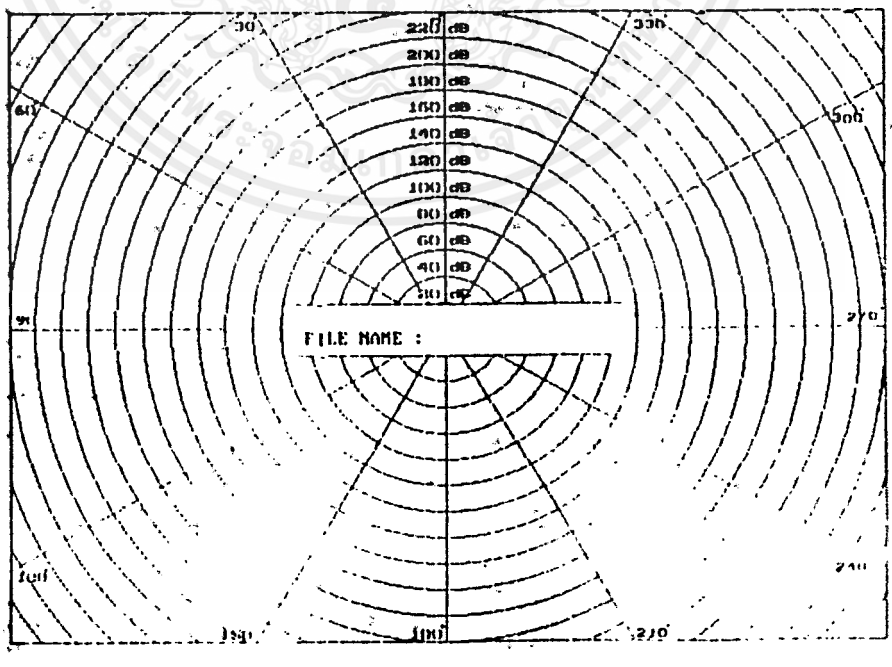
เป็นคำสั่งให้ดึงแฟ้มข้อมูลภาพที่มีอยู่แล้วมาปรากฏบนจอมอนิเตอร์ ขณะอยู่ในเมนูหลัก:มีกด L จะปรากฏข้อความถามชื่อไฟล์



ให้ผู้ใช้ป้อนชื่อไฟล์ที่ต้องการ หรือ ถ้าต้องการดึงแฟ้มข้อมูลภาพจากไดร์ฟไหนก็ให้พิมพ์ไดร์ฟตามด้วยชื่อแฟ้มข้อมูลภาพ แล้วกด Enter เมื่อต้องการเข้าสู่เมนูหลักก็ให้กด Key อะไรก็ได้

4.3 คำสั่ง Save

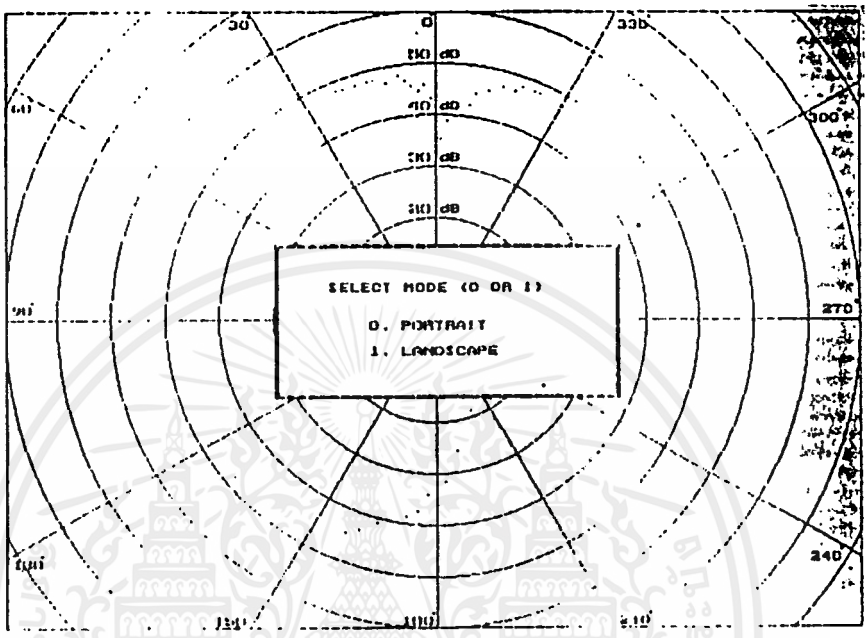
เป็นคำสั่งให้นำข้อมูลภาพที่ปรากฏอยู่บนจอมอนิเตอร์ เข้าไปเก็บในแฟ้มข้อมูลบนดิสก์ ขณะอยู่ในเมนูหลักเมื่อกด S จะปรากฏข้อความถามชื่อไฟล์



ให้ผู้ใช้ป้อนชื่อไฟล์ตามต้องการ หรือต้องการเก็บภาพกราฟฟิกลงในไดร์ฟไหนก็ให้พิมพ์ตามด้วยชื่อไฟล์ แล้วกด Enter ภาพที่ปรากฏบนจอมอนิเตอร์จะถูกเก็บลงในดิสก์

4.4 คำสั่ง Print

เป็นคำสั่งให้พิมพ์ภาพกราฟฟิกที่ปรากฏบนจอมอนิเตอร์ ขณะอยู่ในเมนูหลัก เมื่อกด P จะประกฏข้อความดังนี้



ให้กด 0 เมื่อต้องการพิมพ์ตามขวาง และกด 1 เมื่อต้องการพิมพ์ตามยาว่ของกระดาษ

4.5 คำสั่ง Auto Test

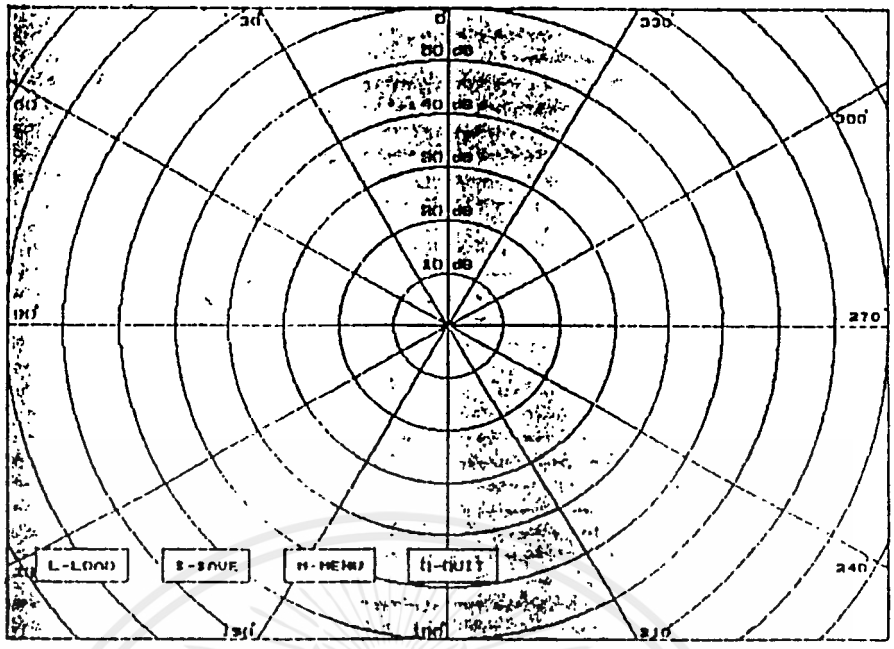
เป็นคำสั่งให้ทำการทดสอบแพทเทิร์นของสายอากาศแบบ Automatic ขณะอยู่ในเมนูหลัก เมื่อกด A โปรแกรมจะเข้าสู่โหมด AutoTest

กด S เมื่อต้องการเก็บภาพลงบนดิสก์

กด P เมื่อต้องการพิมพ์ภาพกราฟฟิก

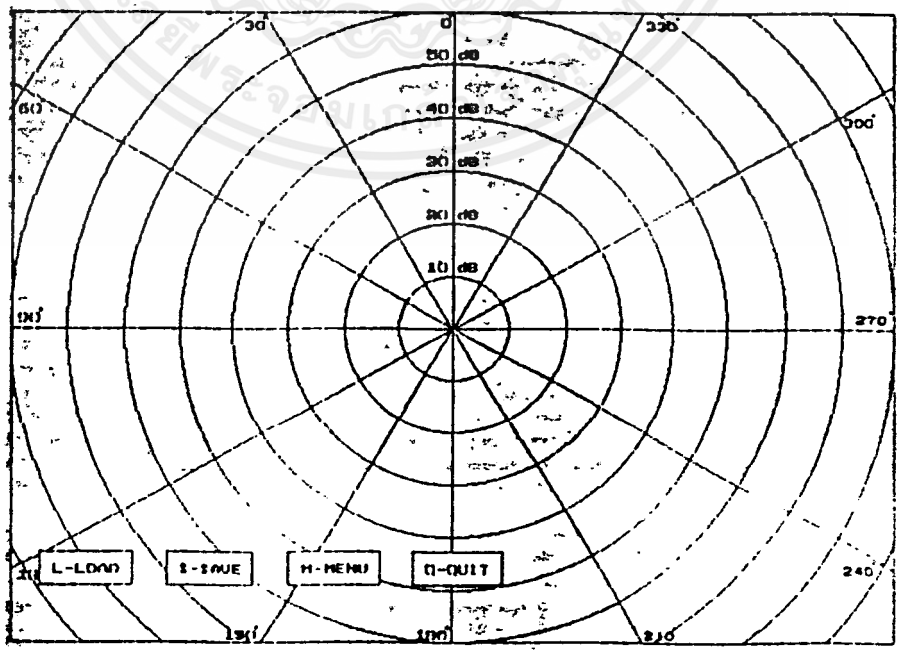
กด L เมื่อต้องการดึงข้อมูลภาพแสดงบนจอมอนิเตอร์

กด M เมื่อต้องการเข้าสู่เมนูหลัก



4.6 **คำสั่ง Test**

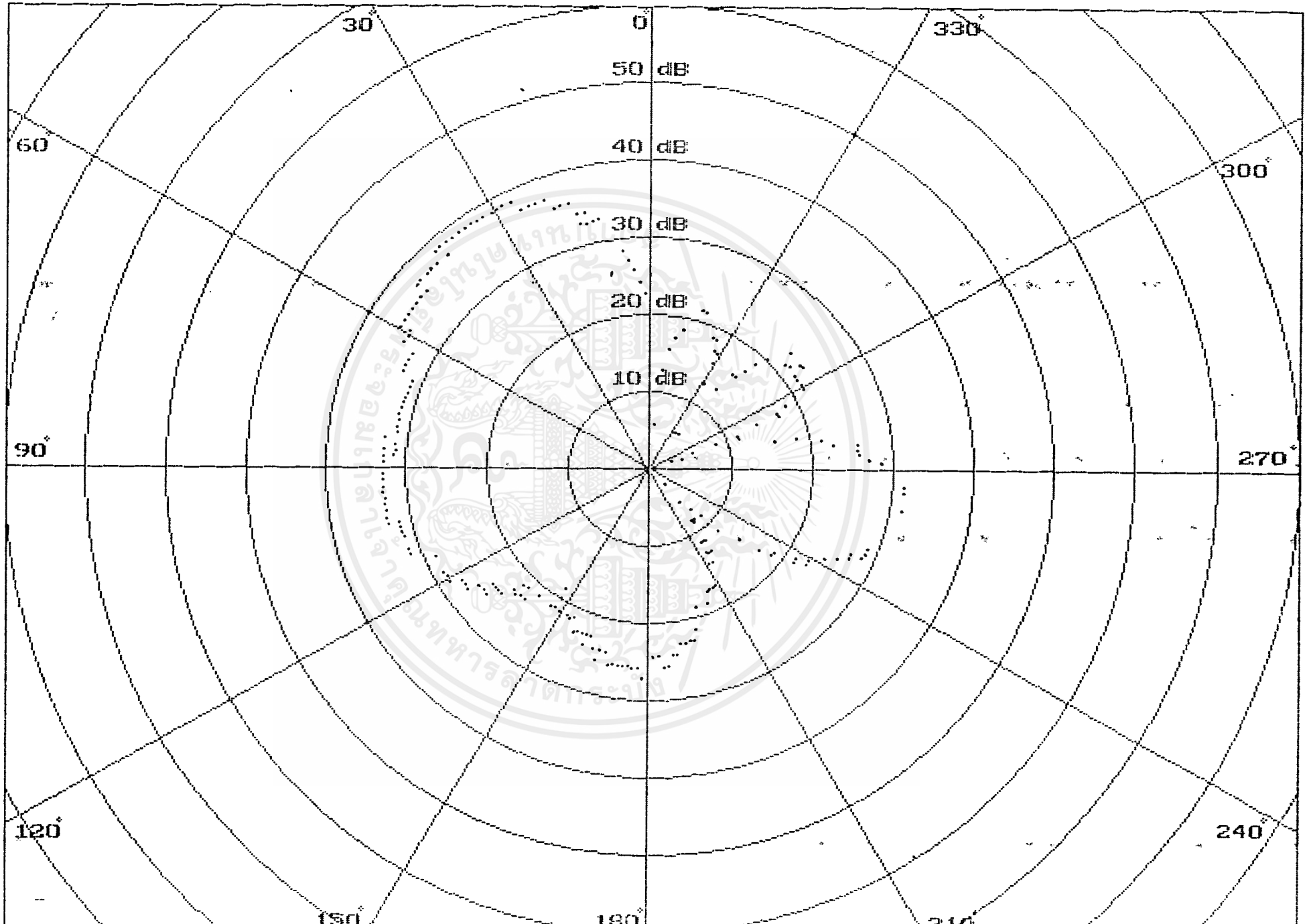
เป็นคำสั่งให้ทำการทดสอบแพทเทิร์นของสายอากาศควบคุมการหมุนหาทิศทางของ Rotor โดย Key ลูกศรให้หมุนซ้าย ขวา ขึ้น ลง ตามต้องการขณะอยู่ในเมนูหลัก เมื่อกด T โปรแกรมจะเข้าสู่โหมด Test

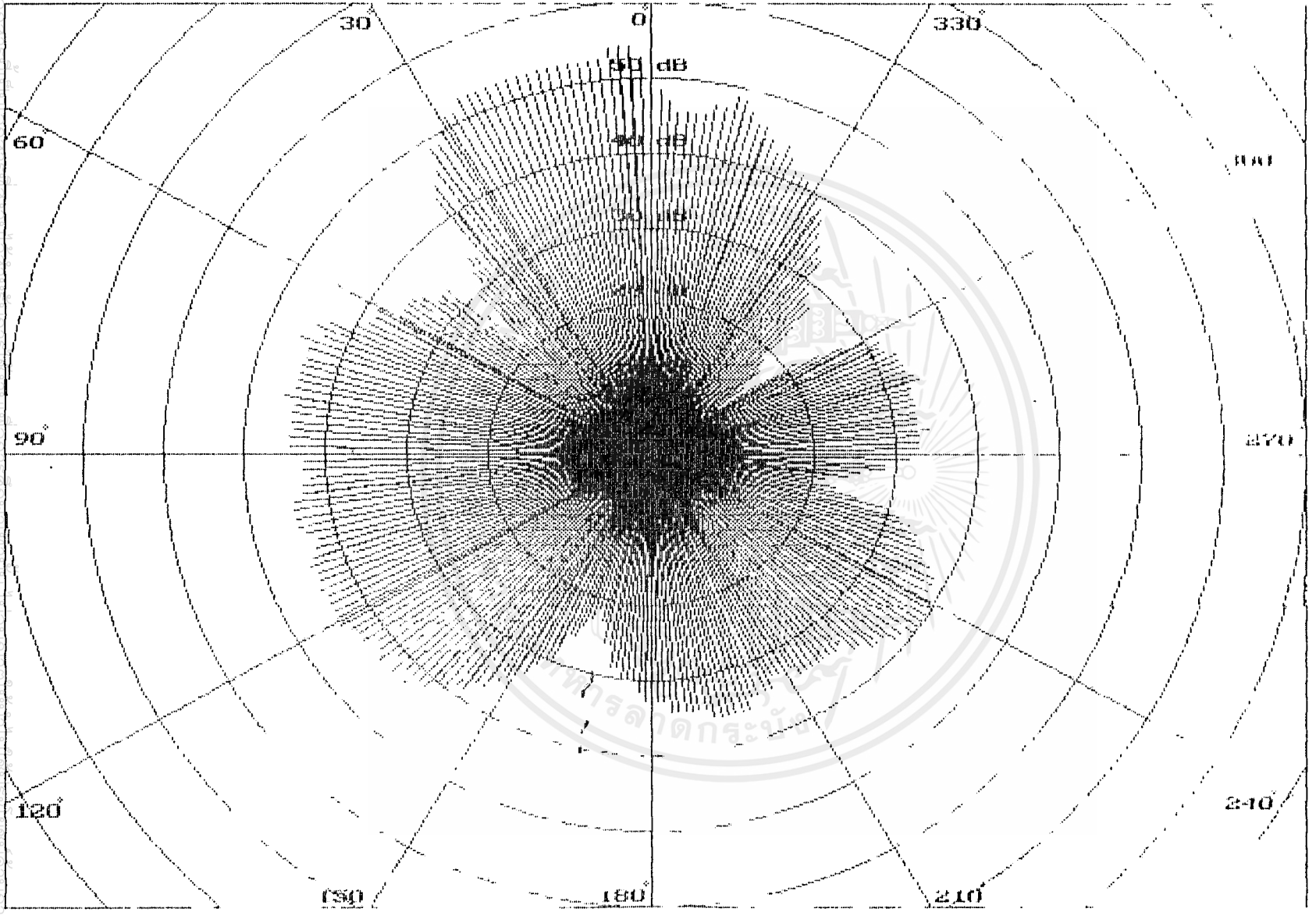


กด ← เมื่อต้องการให้ Rotor หมุนซ้าย
กด → เมื่อต้องการให้ Rotor หมุนขวา
กด ↑ เมื่อต้องการให้ Rotor ยกขึ้น
กด ↓ เมื่อต้องการให้ Rotor ก้มลง
กด S เมื่อต้องการเก็บภาพลงบนดิสก์
กด P เมื่อต้องการพิมพ์ภาพกราฟฟิก
กด L เมื่อต้องการดึงภาพจากไฟล์ในดิสก์แสดงบนจอมอนิเตอร์

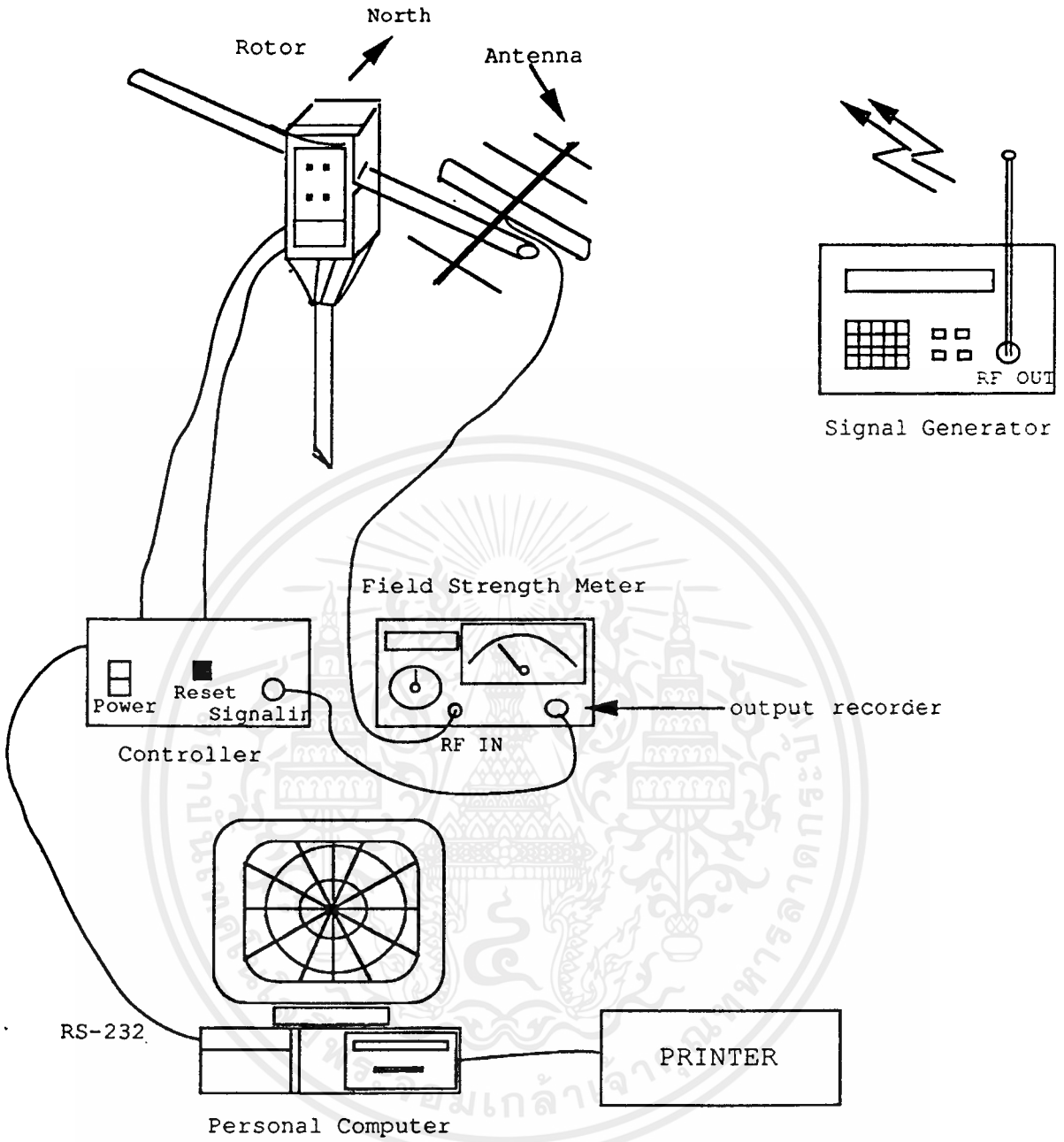
4.7 **คำสั่ง Directory** เป็นการขอดูแฟ้มข้อมูล

4.8 **คำสั่ง Help** เป็นคำสั่งขอดูคำอธิบายการใช้ในโปรแกรม


0°
30°
330°
50 dB
40 dB
60°
300°
30 dB
20 dB
10 dB
90°
270°
120°
240°
150°
180°
210°


0°
30°
330°
60°
90°
120°
150°
180°
210°
240°
270°
dB

4.9 การติดตั้งอุปกรณ์



4.10 ลำดับขั้นการทดสอบ

1. ติดตั้งอุปกรณ์ตามรูป
2. ตั้งความถี่ที่ Signal Gen. ให้ตรงกับ Frequency response ของสายอากาศที่ต้องการทดสอบแพทเทิร์น
3. ตั้ง Rotor ให้หันไปทาง Signal Gen.
4. ตั้งความถี่ของ Field strength meter ให้ตรงกับความถี่ Signal Gen.
5. Run โปรแกรม AntTest
6. Gen.ความถี่ให้มี level แรงพอประมาณ สังเกตดูที่หน้าปัดของ field strength meter ไม่ให้เข็มชี้เกินสเกล (ถ้าเข็มชี้เกินสเกลให้ Atten ลงก็ได้แต่ต้องเซ็ตค่า Atten ให้กับโปรแกรมด้วย)
7. เลือกโหมด Test ตามต้องการ

บทที่ 5

สรุปและวิจารณ์

เครื่องทดสอบแพทเทิร์นของสายอากาศด้วยคอมพิวเตอร์ (Antenna Pattern Tester with Personal Computer) เป็นเครื่องมือทดสอบที่ใช้วิเคราะห์คุณสมบัติของสายอากาศ และวิเคราะห์หาความเข้มของสนามไฟฟ้าเชิงมุม โดยใช้ Personal Computer เป็นตัวควบคุม เครื่องทดสอบนี้สามารถวัดอัตราการขยายของสายอากาศ และ สามารถแสดงลักษณะของ แพทเทิร์น ออกมาทาง Monitor ของ Personal Computer ได้ และยังสามารถ Print out ออกมาทาง Printer ได้

เครื่องทดสอบชุดนี้ประกอบด้วยส่วนสำคัญ 4 ส่วนด้วยกันคือ

1. คอมพิวเตอร์ IBM/PC
2. ชุด Field Strength meter
3. ชุด Driver และ Rotor
4. ชุด Microcontroller (มี A/D,I/O Port รวมอยู่ด้วย)

ในการออกแบบโครงงานชุดนี้มีหลักใหญ่ดังนี้

1. ต้องทำงานได้ดี มีประสิทธิภาพเชื่อถือได้
2. ง่ายต่อการซ่อมบำรุง
3. อุปกรณ์ที่ใช้ต้องหาง่ายและมีราคาถูก
4. สามารถจะ Expand ใช้กับระบบต่างๆ ตามความเหมาะสมได้
5. มีความคล่องตัวในการใช้งาน

Project นี้ประกอบด้วยทั้ง Hardware และ Software ซึ่งมีความสำคัญพอๆกันเพราะงานชิ้นนี้เป็นงานขนาดใหญ่สามารถนำไปประยุกต์ใช้งานได้มากมายโดยแก้ไขที่ตัวโปรแกรม (Software) ดังนั้นจะต้องศึกษาส่วนของ Software ให้เข้าใจเพราะจะสามารถนำไปดัดแปลงตลอดจนประยุกต์ใช้งานอื่นๆได้อย่างมีประสิทธิภาพยิ่งขึ้น

ภาคผนวก

ภาคผนวก ก

```
#define pi 3.141592654
#define step 1.40626
#define PORTRAIT 0
#define LANDSCAPE 1
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<alloc.h>
#ifdef_TINY_
#error Graphics will noy run in the tiny model
#endif
#include<stdlib.h>
#include<stdarg.h>
#include<graphics.h>
#include<gprint.i>
int       GraphDriver;
int       GraphMode;
double    AspectRatio;
int       xasp,yasp;
int       MaxColors;
int       ErrorCode=0;
int    azimuth,OldAzimuth,elevate,dB,echo;
int    po.zoom=1;
void      *Flash,*Flash1,*Flash2,*Flash3,
          *screen1,*screen2,*screen3,*screen4,
          *save1,*save2,*save3,*save4;
          *MemMenu,*Mem,*MenuPrint;
void Initialize()
{
  GraphDriver=DETECT;
  initgraph(&GraphDriver,&GraphMode,"c:\tc\bgi");
  ErrorCode=graphresult();
  if(ErrorCode!=grOk)
  {
    printf("Graphics System Error:%s\n",grapherrormsg(ErrorCode));
    exit(1);
  }
  MaxColors=getmaxcolor()+1;
  getaspectratio(&xasp,&yasp);
  AspectRatio=(double)xasp/(double)yasp;
}

void *SaveImage(int left,int top,int right,int bottom,unsigned *size)
{
  void *image;

  *size=imagesize(left,top,right,bottom);
  image=malloc(*size);
  getimage(left,top,right,bottom,image);
  putimage(left,top,image,XOR_PUT);
  return(image);
}

void FileImage(void *image,unsigned size,char *filename)
{
  FILE *fl=fopen(filename,"w");
```

```
  fwrite(image,size,1,f1);
  fflush(f1);
  fclose(f1);
}

void *ReadImage(char *filename)
{
  unsigned xsize,ysize,size;
  FILE *f1=fopen(filename,"r");
  void *tempimage;
  xsize=fgetc(f1)|(fgetc(f1)<<8);
  ysize=fgetc(f1)|(fgetc(f1)<<8);
  size=imagesize(0,0,xsize,ysize);
  tempimage=malloc(size);
  rewind(f1);
  fread(tempimage,size,1,f1);
  fclose(f1);
  return(tempimage);
}
/* ----------การสร้างภาพ Background ----------*/

void Background(int r)
{
 int R;
 int value=0;
 rectangle(0,0,639,479);          /* สร้างกรอบภาพ */
 line(181,0,456,479);             /* ลากเส้นที่องศาต่าง ๆ ตัดจุดศูนย์กลางภาพ */
 line(0,54,639,424);
 line(0,424,639,54);
 line(181,479,456,0);
 line(0,239,639,239);
 line(319,0,319,479);
 outtextxy(310,5,"0");            /* แสดงค่ามุมที่ตำแหน่งต่าง ๆ */
 circle(316,2,1);
 outtextxy(167,7,"30");
 circle(181,4,1);
 outtextxy(5,70,"60");
 circle(20,67,1);
 outtextxy(5,228,"90");
 circle(20,225,1);
 outtextxy(5,425,"120");
 circle(27,422,1);
 outtextxy(155,470,"150");
 circle(178,467,1);
 outtextxy(293,470,"180");
 circle(316,467,1);
 outtextxy(458,470,"210");
 circle(481,467,1);
 outtextxy(600,421,"240");
 circle(623,418,1);
 outtextxy(610,228,"270");
 circle(636,225,1);
 outtextxy(600,79,"300");
 circle(623,76,1);
```

```
 outtextxy(458,7,"330");
 circle(481,4.1);
for(R=r*zoom;R<400;R=R+r*zoom)        /*  ทำวงกลมตามค่ารัศมีที่ผ่านให้ฟังก์ชั่น  */
 {
  value=value+r;
  circle(319,239,R);
  outtextxy(322,(239-(R+10)),"dB");
 if(R<240){
  if(value<100)
  gprintxy(300,(239-(R+10)),"%d",value);
  else gprintxy(293,(239-(R+10)),"%d",value);
  }
 }
}
void port_init(port,code)
int port;
unsigned char code;
{
 union REGS r;
 r.x.dx = port;
 r.h.ah = 0;
 r.h.al = code;
 int86(0x14,&r,&r);
 }
void sport(port,c)
int port;
char c;
{
 union REGS r;
 r.x.dx = port;
 r.h.al = c;
 r.h.ah = 1;
 int86(0x14,&r,&r);
 if(r.h.ah & 128) {
    printf("send error detected in serial port");
    exit(1);
    }
  }
rport(port)
int port;
{
 union REGS r;
 while(!(check_stat(po)&256))
   if(kbhit()) {
      r.h.ch=getch();
     if(r.h.ch=='q')
      exit(1);
      }
 r.x.dx = port;
 r.h.ah = 2;
 int86(0x14,&r,&r);
 if(r.h.ah & 128)
    printf("\n read error detected in serial port \n");
 return r.h.al;
 }
 check_stat(port)
 int port;
```

```
{
 union REGS r;

 r.x.dx = port;
 r.h.ah = 3;
 int86(0x14,&r,&r);
 return r.x.ax;
}
/* ---------- การเลื่อน CURSOR ---------- */

void goto_xy(int x,int y)
{
 union REGS r;
 r.h.ah=2;          /* cursor function */
 r.h.dl=y;          /* column coordinate */
 r.h.dh=x;          /* row coordinate *
 r.h.bh=0;          /* video page */
 int86(0x10,&r,&r);
}

 xmit(unsigned code)
 {
  do
    {
      sport(po,code);              /* ส่งตัวอักษรออกทาง RS-232 */
      azimuth = rport(po);         /* รับค่า Azimuth */
      elevate = rport(po);         /* รับค่า Elevate */
      dB = rport(po);              /* รับค่า dB */
    echo = rport(po);              /* รับตัวอักษรที่ส่งออกไปกลับมาตรวจสอบ */
    }
 while(echo != code);  /* ตรวจสอบตัวอักษรที่ส่งไปและรับกลับมาว่าตรงกันหรือไม่
 }                     ถ้าไม่ตรงส่งใหม่ */

/*---------- PROGRAM SAVE IMAGE ----------*/

void Save()
{
 char fname[15];
 FILE *fp;
 register int i,j;
 char far *ptr = (char far *)0xA0000000;      /* พอย์เตอร์ชี้ไปที่ EGA memory */
 unsigned char buff[200][200];
 Flash2=SaveImage(200,220,458,260,0);         /* เก็บภาพตำแหน่งที่จะถามชื่อไฟล์ */
 rectangle(200,220,458,260);
 goto_xy(15,27);
 printf("FILE NAME : ");                      /* ถามชื่อไฟล์ */
 gets(fname);
 putimage(200,220,Flash2,COPY_PUT);           /* คืนหน้าจอ */
 free(Flash2);
 Rescreen();
```

```
if(!(fp=fopen(fname,"wb")))                    /* เปิดไฟล์ */
  { Flash2=SaveImage(200,220,458,260,0);
    rectangle(200,220,458,260);
    outtextxy(210,240,"cannot open file..."):   /* เปิดไฟล์ไม่สำเร็จพิมพ์ข้อความ */
    getch();
    putimage(200,220,Flash2,COPY_PUT);
    free(Flash2);
    return;
  }
for(i=0;i<200;i++)                              /* เก็บภาพลงใน Buffer */
 for(j=0;j<192;j++)
  { buff[i][j]=*ptr;                            /* เก็บข้อมูลจาก Buffer ลงดิสก์*/
    ptr++;
  }
for(i=0;i<200;i++)
  for(j=0;j<192;j++)
   { putc(buff[i][j],fp);
   }
 fclose(fp);
}
/*------------------ PROGRAM LOAD IMAGE ------------------*/

void Load()
{
  char fname[80];
  FILE *fp;
  register int i,j;
  char far *ptr = (char far *)0xA0000000;       /* พอย์เตอร์ชี้ไปที่ EGA memory */
  Flash2=SaveImage(200,220,458,260,0);          /* เก็บภาพตำแหน่งที่จะถามชื่อไฟล์ */
  rectangle(200,220,458,260);
  goto_xy(15,27);
  printf("FILE NAME : ");                       /* ถามชื่อไฟล์ */
  gets(fname);
  putimage(200,220,Flash2,COPY_PUT);            /* คืนหน้าจอ */
  free(Flash2);
 if(!(fp=fopen(fname,"rb")))                    /* เปิดไฟล์ */
  { Flash2=SaveImage(200,220,458,260,0);
    rectangle(200,220,458,260);
    outtextxy(210,240,"cannot open file...");   /* เปิดไฟล์ไม่สำเร็จพิมพ์ข้อความ */
    getch();
    putimage(200,220,Flash2,COPY_PUT);
    free(Flash2);
    return;
  }
      for(i=0;i<200;i++)                        /* อ่านข้อมูลจากไฟล์ไปแสดงที่หน้าจอ */
          for(j=0;j<192;j++)
      {
          *ptr=getc(fp);
          ptr++;
       }
    fclose(fp);
```

```
}
SelectPort()
{
 char port;
 Flash1=SaveImage(200,220,458,260,0);
 Flash3=SaveImage(200,220,458,260,0);
  do
      {
        rectangle(200,220,458,260);
        goto_xy(15,27);
        printf("SELECT COM PORT (0-7): ");
        port=getchar();
        if(port<'0'|port>'7')

          {
            putimage(200,220,Flash3,COPY_PUT);
             rectangle(200,220,458,260);
            goto_xy(14,27);
            printf("ENTER NUMBER 0 - 7");
            goto_xy(15,27);
            printf("PRESS ANY KEY TO CONTINUE..");
            getch();
             putimage(200,220,Flash3,COPY_PUT);
          }
      }
while(port<'0'||port>'7');
putimage(200,220,Flash1,COPY_PUT);
free(Flash3);
free(Flash1);
 switch(port)
      {
        case '0':po=0;break;
        case '1':po=1;break;
        case '2':po=2;break;
        case '3':po=3;break;
        case '4':po=4;break;
        case '5':po=5;break;
        case '6':po=6;break;
        case '7':po=7;break;
      }
}

void SaveScreen()
  {
     FILE *fp;
     register int i,j;
    char far *temp=(char far*)0xA0000000;
    unsigned char buf[200][200];                    /* Screen Buffer */
    for(i=0;i<200;i++)
    for(j=0;j<194;j++)
      {
        buf[i][j]=*temp;
        *temp=0;                                    /* Delete Top Line */
        temp++;
      }
  }

void ReSaveScreen()
```

```
    {
      FILE *fp;
      register int i,j;
     char far *temp=(char far*)0xA0000000;
     unsigned char buf[200][200];
     for(i=0;i<200;i++)
     for(j=0;j<194;j++)
       {
           *temp=buf[i][j];
          temp++;
        }
/*------------------ PROGRAM    MENU ----------------------*/

void menu()
 {
   MemMenu=SaveImage(200,150,458,350,0);
   rectangle(200,150,458,350);
   rectangle(202,152,456,348);
   outtextxy(240,170,"ANTENNA PATTERN TESTER");
   outtextxy(300,190,"L-LOAD");
   outtextxy(300,210,"S-SAVE");
   outtextxy(300,230,"P-PRINT");
   outtextxy(300,250,"A-AUTO TEST");
   outtextxy(300,270,"T-TEST");
   outtextxy(300,290,"D-DIRECTORY");
   outtextxy(300,310,"Q-QUIT");
 }
void chowtext()
  {
    screen1=SaveImage(20,410,85,435,0);
    rectangle(20,410,85,435);
    outtextxy(30,420,"L-LOAD");
    screen2=SaveImage(110,410,175,435,0);
    rectangle(110,410,175,435);
   outtextxy(120,420,"S-SAVE");
   screen3=SaveImage(200,410,265,435,0);
   rectangle(200,410,265,435);
   outtextxy(210,420,"M-MENU");
   screen4=SaveImage(290,410,355,435,0);
   rectangle(290,410,355,435);
   outtextxy(300,420,"Q-QUIT");
 }
 Rescreen()
{
   putimage(20,410,screen1,COPY_PUT);
   putimage(110,410,screen2,COPY_PUT);
   putimage(200,410,screen3,COPY_PUT);
  putimage(290,410,screen4,COPY_PUT);
}

Level()
{
  gprintxy(5,440,"Azimuth = %d",azimuth);
  gprintxy(170,440,"Elevate = %d",elevate);
  gprintxy(370,440,"dB = %d",dB);
  gprintxy(570,440,"echo = %d",echo);
}
```

```
xmit(unsigned code)
{
 do
   {
     sport(po,code);              /* ส่งตัวอักษรออกทาง RS-232 */
     azimuth = rport(po):         /* รับค่า Azimuth */
     elevate = rport(po);         /* รับค่า Elevate */
     dB = rport(po):              /* รับค่า dB */
    echo = rport(po);             /* รับตัวอักษรที่ส่งออกไปกลับมาตรวจสอบ */
   }
 while(echo != code);  /* ตรวจสอบตัวอักษรที่ส่งไปและรับกลับมาว่าตรงกันหรือไม่
}                      ถ้าไม่ตรงส่งใหม่ */

/*---------- FUNCTION PRINT GRAPHIC IMAGE ----------*/

void Print_Graph(int Mode)
{
 char m,chr;
 int i,j,k,Msb,Lsb,
        MaxX=getmaxx(),
        MaxY=getmaxy(),
        Direction;
 do
    {
     MenuPrint=SaveImage(200,180,458,300,0);
     rectangle(200,180,458,300);
     rectangle(202,182,456,298);
     outtextxy(240,210,"SELECT MODE (0 OR 1)");
     outtextxy(270,240,"0. PORTRAIT");
     outtextxy(270,260,"1. LANDSCAPE");
     chr=getch();
     switch(chr)
        {
          case '0': Direction=0;
                    break;
          case '1': Direction=1;
                    break;
        }
 }
while(chr<'0'||chr>'1');
 setviewport(0,0,MaxX,MaxY,0);
 putimage(200,180,MenuPrint,COPY_PUT);
 free(MenuPrint);
 fprintf(stdprn,"\x1B\x2B%c",50);              /*sets line spacing to 50/360 inch*/
 switch(Direction)
 {
    case PORTRAIT:
      {
       Lsb=MaxX+1 & 0x00FF;                    /*MaxX modulo 256*/
       Msb=MaxX+1 >> 8;                        /*(int)MaxX/256*/
       for(j=0;j<=MaxY/8;j++)
        {
          fprintf(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
```

```
                for(i=0;i<=MaxX;i++)
                  {
                    m=0;
                    for(k=0;k<8;k++)
                        {                                   /*shift m left one bit*/
                            m<<=1;
                            if(getpixel(i,j*8+k))m++;       /*if pixel on,bit on*/
                        }
                    fprintf(stdprn,"%c",m);
                  }
                fprintf(stdprn,"\x0D\x0A");                 /*use CR/LF codes vs\n flag*/
              }
          }
        case LANDSCAPE:
    {
     Lsb=MaxY+1 & 0x00FF;                                   /*MaxY modulo 256*/
     Msb=MaxY+1>>8;                                         /*(int)MaxY/256*/
     for(j=0;j<MaxX;j+=8)
         {
          fprintf(stdprn,"\x1B*%c%c%c",Mode,Lsb,Msb);
          for(i=MaxY;i>=0;i--)
            {
              m=0;
              for(k=0;k<8;k++)
                {
                  m<<=1;                                    /*shift m left one bit*/
                  if(getpixel(j+k,i))m++;                   /*if pixel on,set bit*/
                }
              fprintf(stdprn,"%c",m);
            }
          fprintf(stdprn,"\x0D\x0A");                       /*use CR/LF codes vs\n flag*/
         }
    }
  }
fprintf(stdprn,"\f");                                       /*form feed to advance paper*/
}

/*------------------------- FUNCTION TEST -------------------------*/

Test()
{
   int r,x,y;
   char filename[15];
   double a,b,z,rad,Degree,DB;
   union k{
   int i;
  char c[2];
  }
 key;
 cleardevice();
 zoom=4;
 Background(10);                                    /* สร้าง Background */
 free(screen1);
 free(screen2);
 free(screen3);
 free(screen4);
 chowtext();
```

```
port_init(po,195);                          /*  เตรียมสถานะพอร์ตอนุกรม */

key.c[0]='a';key.c[1]=1;
xmit(key.c[0]);                        /*  ส่งรหัสอ่านค่า dB,  Azimuth ,  Elevate */
xmit(key.c[1]);
do
 {
                                        /*Level();*/
   DB=dB*1.171875;
   Degree = azimuth*step;                              /* ทำองศาเป็นเรเดียน */
   rad=(pi*Degree/180);
   a=(DB*zoom)*sin(rad):b=(DB*zoom)*cos(rad):     /* คำนวณหาตำแหน่ง X, Y */
   x=a;a=a-x;z=a*2;x=x+z;
  y=b;b=b-y;z=b*2;y=y+z;                        /* ทำ X, Y ให้เป็นจำนวนเต็ม */
  if(OldAzimuth!=azimuth)
   {
     circle(319-x,239-y,1);
      putpixel(319-x,239-y,WHITE);
                                  /* ถ้ามุมไม่แปลี่ยนแปลงทำการพล็อตจุดบนจอ */
     OldAzimuth=azimuth;
   }
 if (kbhit())
key.i=bioskey(0);                          /*  ตรวจสอบ Key */
xmit(key.c[0]);                            /*  ส่งรหัส Key ที่กด */
xmit(key.c[1]);
switch(key.c[1])                           /*  ตรวจสอบ Key ลูกศร */
 { case 75:                                /* LEFT */
          if(azimuth==0)
               {key.c[0]='a';
                key.c[1]='1';}
               break;
  case 77:                                 /* RIGHT */
          if(azimuth==255)
               {key.c[0]='a';
               key.c[1]='1';}
               break;
  case 72:                                 /* UP */
          if(elevate==0)
               {key.c[0]='a';
               key.c[1]='1';}
               break;
  case 80:                                 /* DOWN */
          if(elevate==255)
               {key.c[0]='a';
               key.c[1]='1';}
               break;
  default: break;
 }
switch(tolower(key.c[0]))                  /*  ตรวจสอบ Key */

 { case 'l':  SaveScreen();
```

```
                Load();                 /* กด L เข้าสู่ฟังก์ชั่น Load */
                getch():
                ReSaveScreen():
                break:
    case 's':  Save();                  /* กด S เข้าสู่ฟังก์ชั่น Save */
                getch():
                break:
    case 'd':  SaveScreen():
                system("DIR/W");        /* กด D ดู Directory */
                  getch():
                ReSaveScreen():
                getch():
                break:
    case 'p':  Rescreen():
                Print_Graph(0);         /* กด P เข้าสู่ฟังก์ชั่น Print Graph */
                break:
    case 'm':  goto end:                /* กด M เข้าสู่ Main Menu */

    case 'q':  closegraph();            /* กด Q จบโปรแกรม */
                exit(0);
    default :  break;
   }
}
while (azimuth<255):                    /* ตรวจสอบมุม Rotor 360 องศา */
end:xmit('a');                          /* ส่งรหัสเพื่อหยุด Rotor */
getch():
}

/*---------- FUNCTION AUTOMATIC TEST ----------*/

AutoTest()
{
 int r.x.y.Size=0;
 register int i.j;
 char ch.ch1:
 char filename[15]:
 double a.b.z.rad.Degree.DB;
 cleardevice():
 zoom=4:
 Background(10);                        /* โปรแกรม Background */
 free(screen1);
 free(screen2);
 free(screen3):
 free(screen4);
 chowtext();                            /* แสดงอักษรหน้าจอ */
 port_init(po.195):                     /* เตรียมสถานะพอร์ตอนุกรม */
 ch='a';ch1=1;                  /* ส่งรหัสอ่านค่า dB, Azimuth , Elevate */
 xmit(ch);
 xmit(ch1);
 do
```

```
    {
      ch=0;
      ch1=75;
      xmit(ch);                          /* ควบคุมให้ Rotor กลับไปที่ศูนย์องศา */
      xmit(ch1);
  }
while(azimuth!=0);
  OldAzimuth=azimuth;
  do
    {
      ch1=77;
      xmit(ch);              /* ควบคุมให้ Rotor หมุนอ่านค่า dB, Azimuth, Elevate */
      xmit(ch1);
      DB=dB*1.171875;                                    /* ปรับ Scalling */
      Degree = azimuth*step;                             /* ทำองศาเป็นเรเดียน */
  rad=(pi*Degree/180);                                   /* คำนวณหาตำแหน่ง X, Y */
  a=(DB*zoom)*sin(rad);b=(DB*zoom)*cos(rad);

                                        /*ถ้ามุมเปลี่ยนแปลงทำการพล็อตจุดบนจอ */
  x=a;a=a-x;z=a*2;x=x+z;
  y=b;b=b-y;z=b*2;y=y+z;
  if(OldAzimuth!=azimuth)
    circle(319-x,239-y,1);
                                        /* putpixel(319-x,239-y,WHITE);*/
  OldAzimuth=azimuth;
                                        /* Level();*/
  }
    while (azimuth<255);                /* ตรวจสอบมุมครบ 360 องศาหรือยัง */
    ch='a';ch1=1;
    xmit(ch);                           /* ส่งรหัสคำสั่งให้ Rotor หยุดหมุน */
    xmit(ch1);
    SubMenu();                          /* เข้าสู่โปรแกรม SubMenu */
  }
    SubMenu()                           /* โปรแกรมเมนย่อย */
    {
      char ch;
      do
          {
            ch=getch();                 /* ตรวจสอบ Key */
            switch(tolower(ch))
                {
                  case 'l': Load();     /* เรียกฟังก์ชั่น Load */
                            break;
                  case 's': Save();     /* เรียกฟังก์ชั่น Save */
                            break;
                  case 'd': SaveScreen();
                            system("DIR/W");          /* ดู Directory */
                            getch();
                           ReSaveScreen();
```

```
                    getch();
                    break;
                case 'm': break;
                case 'p': Rescreen();
                          Print_Graph(0);          /* เรียกฟังก์ชั่น Print Graph */
                          break;
                case 'q': closegraph();
                          exit(0);

                default : break;
                }
            }
while(ch!='m');                                   /* กด M เข้าสู่ Main Menu */
}
/*------------------------ MAIN PROGRAM ------------------------*/

main()
 {
  union k
     {
       char c[2];
       int i;
     }
  key;
  Initialize();                                   /* เข้าสู่ภาวะกราฟฟิกโหมด */
  SelectPort();                                   /* เลือกพอร์ตอนุกรม */
  Background(20);                                 /* โปรแกรม Background */
  do
     {
       menu();                                    /* Menu โปรแกรม */
       key.i=bioskey(0);                          /* รอจนกว่าผู้ใช้กด Key */
       putimage(200,150,MemMenu,COPY_PUT);        /* คืนหน้าจอ */
       free(MemMenu);

              closegraph();
              system("DIR/W");                    /* ดู Directory */
               getch();
              Initialize();
              ReSaveScreen();
              break;
    case 'q': closegraph();                       /* จบโปรแกรม */
              exit(0);
    default : break;
    }
   }
  while(key.c[0]!='q');                           /* กด Q จบโปรแกรม */
  closegraph();
 }
```

8051 CROSS ASSEMBLER - VERSION 3.00g

INPUT FILENAME : CONT.ASM
OUTPUT FILENAME : CONT.HEX

```
1
2                                   :Program  Serial port communication for controll rotor
3       00 48           UP          EQU  48H
4       00 50           DOWN        EQU  50H
5       00 4B           LEFT        EQU  4BH
6       00 4D           RIGHT       EQU  4DH
7  0000                             ORG  0000H
8  0000  02 00 92       MODEONE: LJMP  OVER          :jump over serial interrupt address
9  000B                             ORG  000BH
10 000B  02 00 50                     LJMP  CLOCK          :clock A to D
11 0050                             ORG  50H
12 0050  C0 E0          CLOCK:      PUSH  ACC
13 0052  C0 D0                      PUSH  PSW
14 0054  C3                         CLR  C
15 0055  EC                         MOV  A.R4
16 0056  23                         RL  A
17 0057  FC                         MOV  R4.A
18 0058  33                         RLC  A
19 0059  92 B4                      MOV  P3.4.C
20
21 005B  D0 D0          GO:         POP  PSW
22 005D  D0 E0                        POP  ACC
23 005F  32                         RETI
24 0060  C2 98          NODATA:  CLR  RI
25 0062  C2 99                      CLR  TI
26 0064  30 98 FD                     JNB  RI.$
27 0067  AA 99                      MOV  R2.SBUF
28 0069  90 80 00                   MOV  DPTR.#8000H
29 006C  7B 03                      MOV  R3.#3
30 006E  F0             RDATA:      MOVX  @DPTR.A        ;start A to D
31 006F  7E 9B                       MOV  R6.#9BH          :conversion time delay
32 0071  DE FE                      DJNZ  R6.$
33 0073  E0                          MOVX  A.@DPTR        :read A to D
34 0074  C2 98          XMIT:  CLR  RI
35 0076  C2 99                      CLR  TI
36 0078  F5 99                      MOV  SBUF.A
37 007A  30 99 FD                   JNB  TI.$
38 007D  C2 99                      CLR  TI
39 007F  A3                         INC  DPTR
40 0080  DB EC                      DJNZ  R3.RDATA
41
42
43 0082  C2 98          ECHO:    CLR  RI                 ;echo character back to terminal
44 0084  C2 99                   CLR  TI
45 0086  8A 99                   MOV  SBUF.R2
46 0088  30 99 FD                JNB  TI.$
47 008B  C2 99                   CLR  TI
48 008D  12 00 AD                LCALL CHKEY             :calls subroutines chkey
```

```
49  0090  80 CE              SJMP  NODATA
50
51  0092  75 90 F0           OVER:   MOV  P1.#0F0H      :stops rotor
52  0095  7C 55              MOV  R4.#55H
53  0097  75 89 22           MOV  TMOD.#22H
54  009A  75 98 50           SETSCON: MOV  SCON.#50H     :set serial port mode 1 and
enable receiver
55  009D  75 8D FA           MOV  TH1.#0FAH                :baud rate 4800
56  00A0  D2 AF              SETB  IE.7                     :enable interrupt
57  00A2  D2 8E                 SETB  TCON.6              :start T1 to generatebaud clock
58  00A4  D2 A9              SETB  IE 1                 :enable T0 Overflow Interrupt
59  00A6  75 8C FE              MOV  TH0.#0FEH
60  00A9  D2 8C              SETB  TCON.4           :start T0 generate clock for A to D
61  00AB  80 B3              SJMP  NODATA
62
63
64
65
66
67  00AD  EA          CHKEY:  MOV  A.R2              :check key
68  00AE  B8 01 2A    CJNE  R0.#1.ORTHER
69  00B1  B4 48 06       CJNE  A.#UP.ADD1            :UP ?
70  00B4  75 90 F1       MOV  P1.#0F1H                :out data UP to rotor
71  00B7  78 00       MOV  R0.#0
72  00B9  22          RET
73  00BA  B4 50 06          ADD1:   CJNE  A.#DOWN.ADD2   :DOWN ?
74  00BD  75 90 F2          MOV  P1.#0F2H                :out data DOWN to rotor
75  00C0  78 00          MOV  R0.#0
76  00C2  22             RET
77  00C3  B4 4B 06          ADD2:   CJNE  A.#LEFT.ADD3     :LEFT ?
78  00C6  75 90 F4          MOV  P1.#0F4H                :out data LEFT to rotor
79  00C9  78 00          MOV  R0.#0
80  00CB  22             RET
81  00CC  B4 4D 06          ADD3:   CJNE  A.#RIGHT.ADD4    :RIGHT ?
82  00CF  75 90 F8          MOV  P1.#0F8H                  :out data RIGHT to rotor
83  00D2  78 00          MOV  R0.#0
84  00D4  22             RET
85  00D5  75 90 F0          ADD4:   MOV  P1.#0F0H
86  00D8  78 00          MOV  R0.#0
87  00DA  22             RET
88  00DB  B4 00 F7          ORTHER:  CJNE  A.#0.ADD4
89  00DE  78 01          MOV  R0.#1
90  00E0  22             RET
91
92  00E1                    END
```

********** CROSS REFERENCE TABLE **********

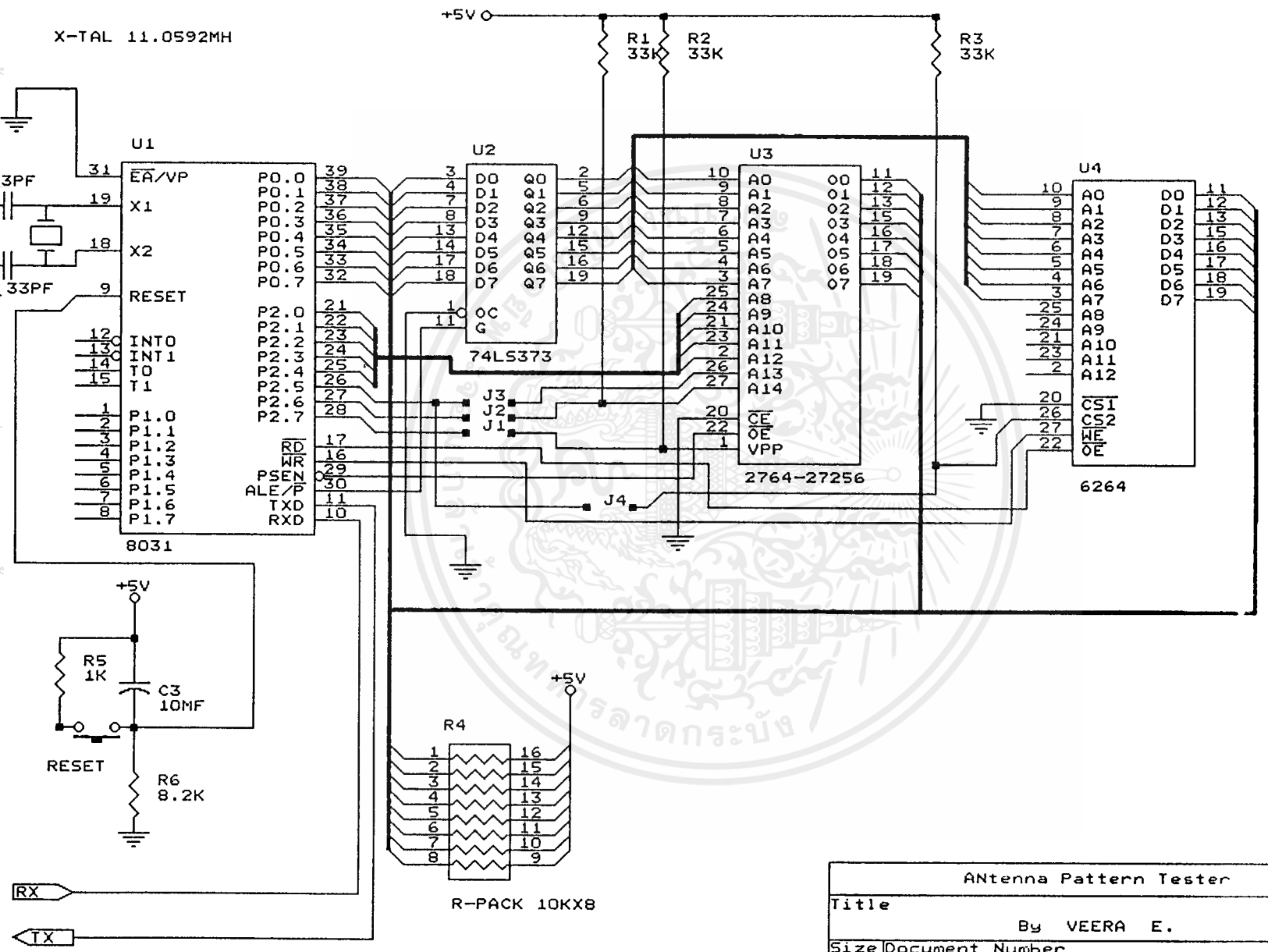
ADD1	00BA	: 69	
ADD2	00C3	: 77	
ADD4	00D5	: 81	88
CHKEY	00AD	: 48	
CLOCK	00500	: 73	
ECHO	0082	:	
LEFT	= 004B		
NODATA	0060	: 49	61
ORTHER	00DB	: 68	
OVER	0092	: 8	
RDATA	006E	: 40	
RIGHT	=004D	: 81	
SETSCON	009A	:	
UP	= 0048	: 69	
XMIT	0074	:	

LINES ASSEMBLED : 92 ASSEMBLY ERRORS : 0

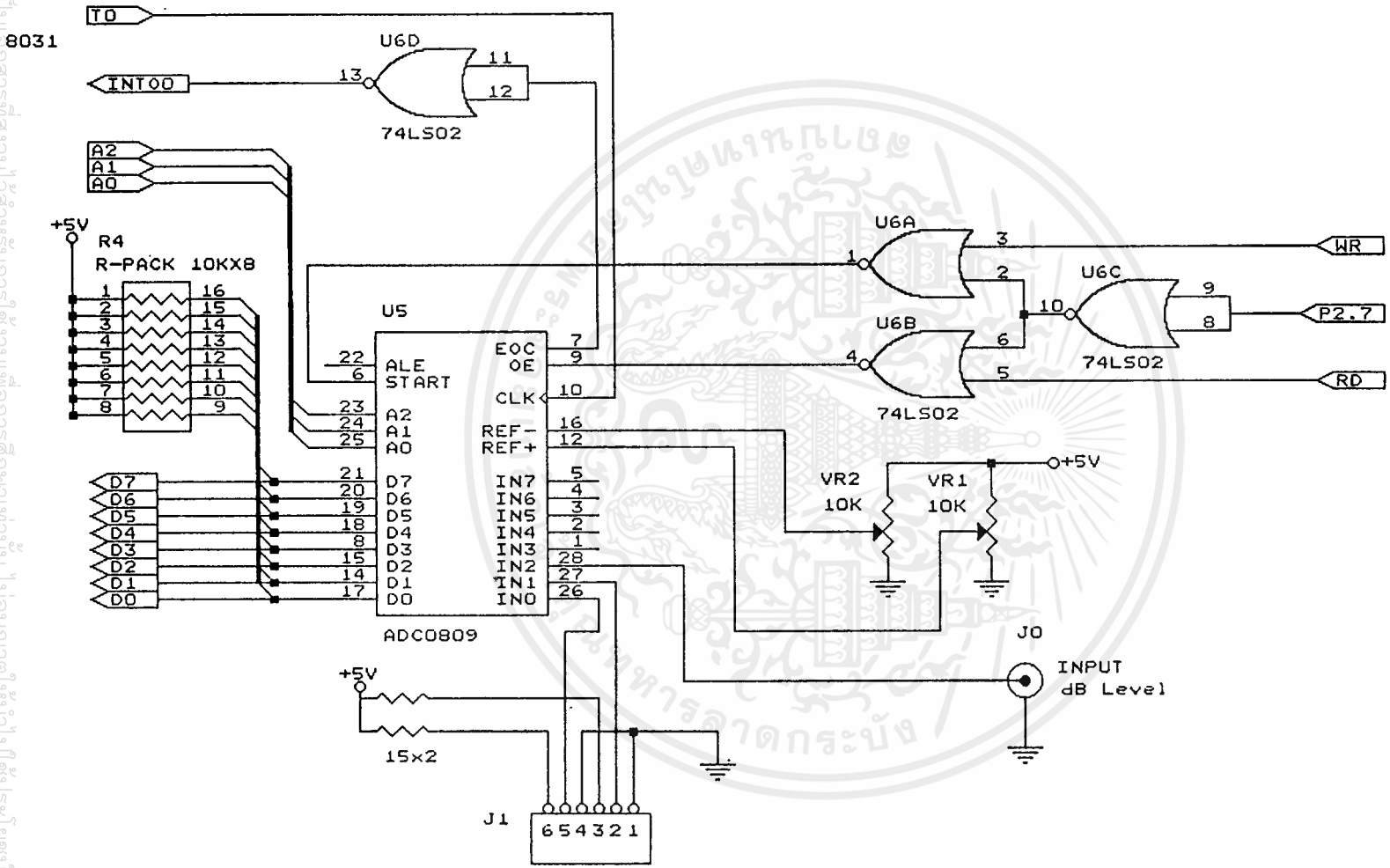
```
void erasestr(int xloc,int yloc,char str)
{
   struct textsettingstype textinfo;
   int xdim,ydim;
   void textimage;
   gettextsettings(&textinfo);
   switch(textinfo.direction)
   {
        case HORIZ_DIR : xdim=textwidth(str);
                         ydim=textheigh(str);
                         xloc--;break;
        case VERT_DIR  : ydim=textwidth(str);
                         xdim=textheigh(str);
                         yloc++;break;
}
switch(textinfo.horiz)
{
   case   LEFT_TEXT   :                    break;
   case   CENTER_TEXT :  xloc-=xdim/2      break:
   case   RIGHT_TEXT  :  xloc-=xdim;       break;
}
switch(textinfo.vert)
{
   case   BOTTOM _TEXT :  yloc-=xdim;      break;
   case   CENTER_TEXT  :  yloc-=xdim/2     break:
   case   TOP_TEXT     :                   break;
}
while(xloc<0){xloc++;xdim--;}
while(yloc<0){yloc++;xdim--;}
textimage=malloc(imagesize(xloc,yloc,xdim,ydim));
getimage(xloc,yloc,xloc+xdim,yloc+ydim,textimage);
    putimage(xloc,yloc,textimage,XOR_PUT);
    free(textimage);
}
void gprintf
{
   va_list argptr:
   char    str[140];
   structtextsettingstype textinfo;
   int pos_adj= xloc;
   va_star(argptr, format);
   vsprintf(str,fmt,argptr);
   gettextsettings(&textinfo);
   #if(_TURBOC_==0x150)
   {
        xloc+=textheight(str);
        if(textinfo.horiz==CENTER_TEXT)
           xloc-=textheight(str)/8;
   }
```

```
}
#endif
outtextxy(pos_adj,yloc,str);
switch(textinfo.direction)
{
   case   HORIZ_DIR   :   yloc+=textheight(str)+2;
break;
   case   VERT_DIR    :   xloc+=textheight(str)+2;
break;
};
va_end(argptr);
}
void gprintc(int xloc,int yloc,char fmt,...)
{
    va_list argptr;
    char    str[140];
    struct  textsettingstype textinfo;
    int     pos_adj= xloc;

    va_start(argptr,format);
    vsprintf(str,fmt,atgptr);
    gettextsettings(&textinfo);
    #if(_TURBO_==0x150)
    {
        if(textinfo.direction && textinfo.font)
        {
            xloc += texthight(str);
            if(textinfo.horiz == CENTER_TEXT)
                xloc-=texthight(str)/8;
        }
    }
    #endif
    erasestr(xloc,yloc,str);
    outtextxy(pos_adj,yloc,str);
    switch (textinfo.direction)
    {
        case HORIZ_DIR:YLOC+=textheight(str)+2;break;
        case VERT_DIR:YLOC+=textheight(str)+2;break;
    };
    va_end(argptr);
}
void gprintxy(in xloc,int yloc,char fmt,...)
{
   va_list argprt;
   char str[140];
   struct textsettingstype textinfo;
   va_start(argptr,format);
   vsprintf(str,fmt,argptr);
   gettextsettings(&textinfo);
```

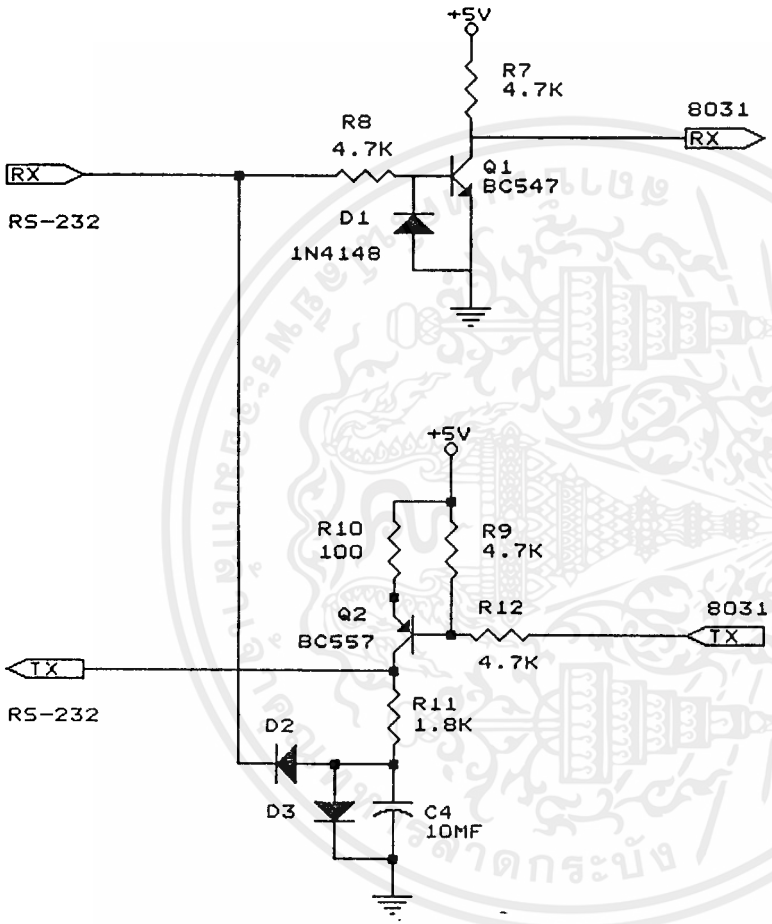
```
    #if(_TURBO_==0x150)
    {
      if(textinfo.direction&&textinfo.font)
      {
           xloc+=textheight(str);
           if(textinfo.horiz==CENTER_TEXT)
              xloc-=textheight(str)/8;
      }
    }
    #endif
    erasestr(xloc,yloc,str);
    outtextxy(xloc,yloc,str);
    va_end(argptr);
}
```



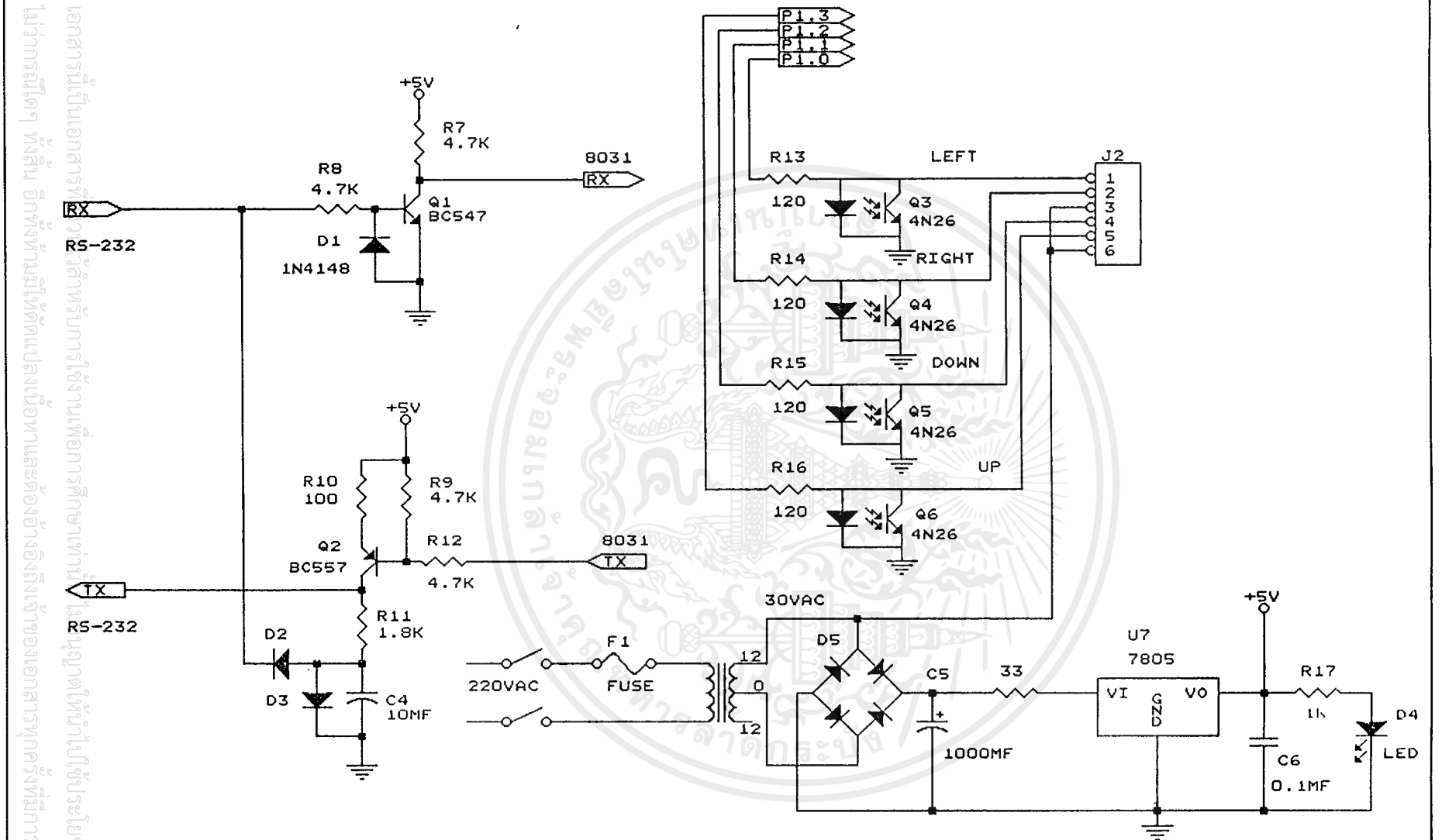
ANtenna Pattern Tester		
Title By VEERA E.		
Size A	Document Number	REV
Date: March 16, 1987	Sheet of	



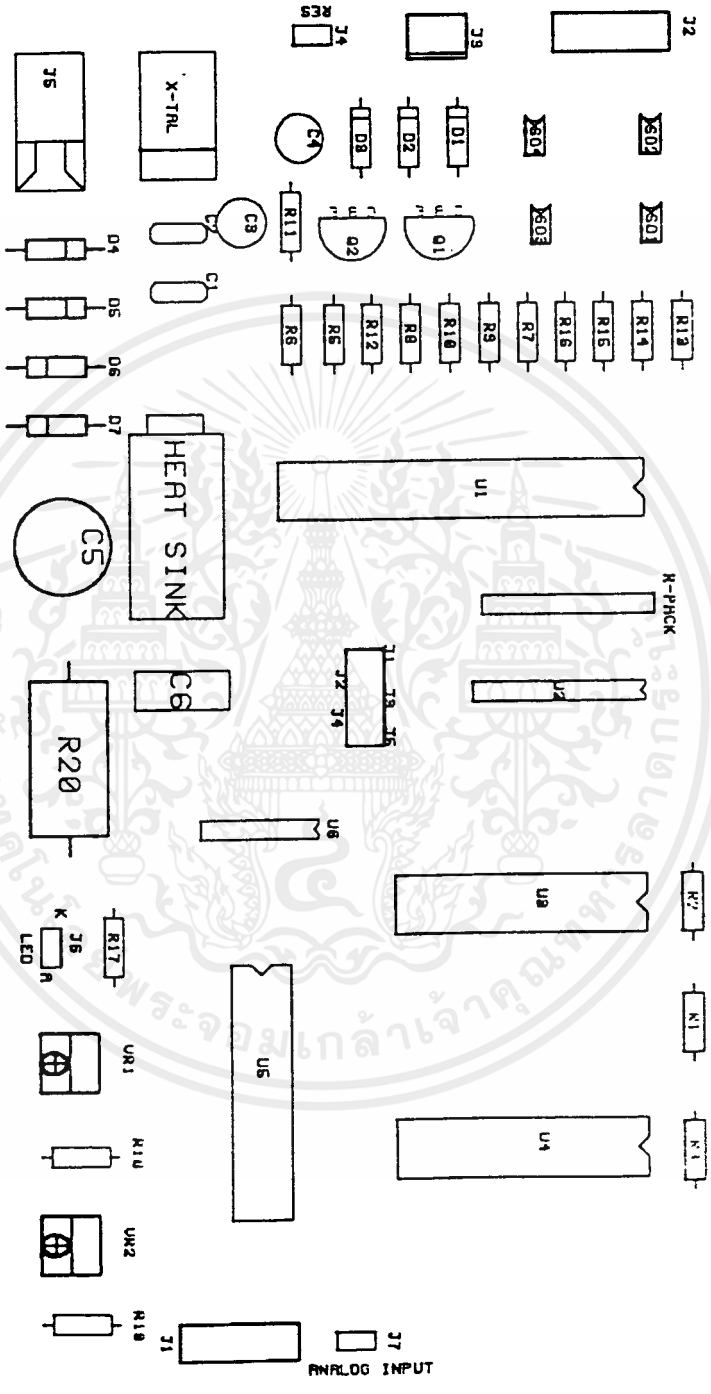
Antenna Pattern Tester		
Title	By VEERA E.	
Size A	Document Number	REV 1/A
Date: March 15, 1993	Sheet 2 of 4	

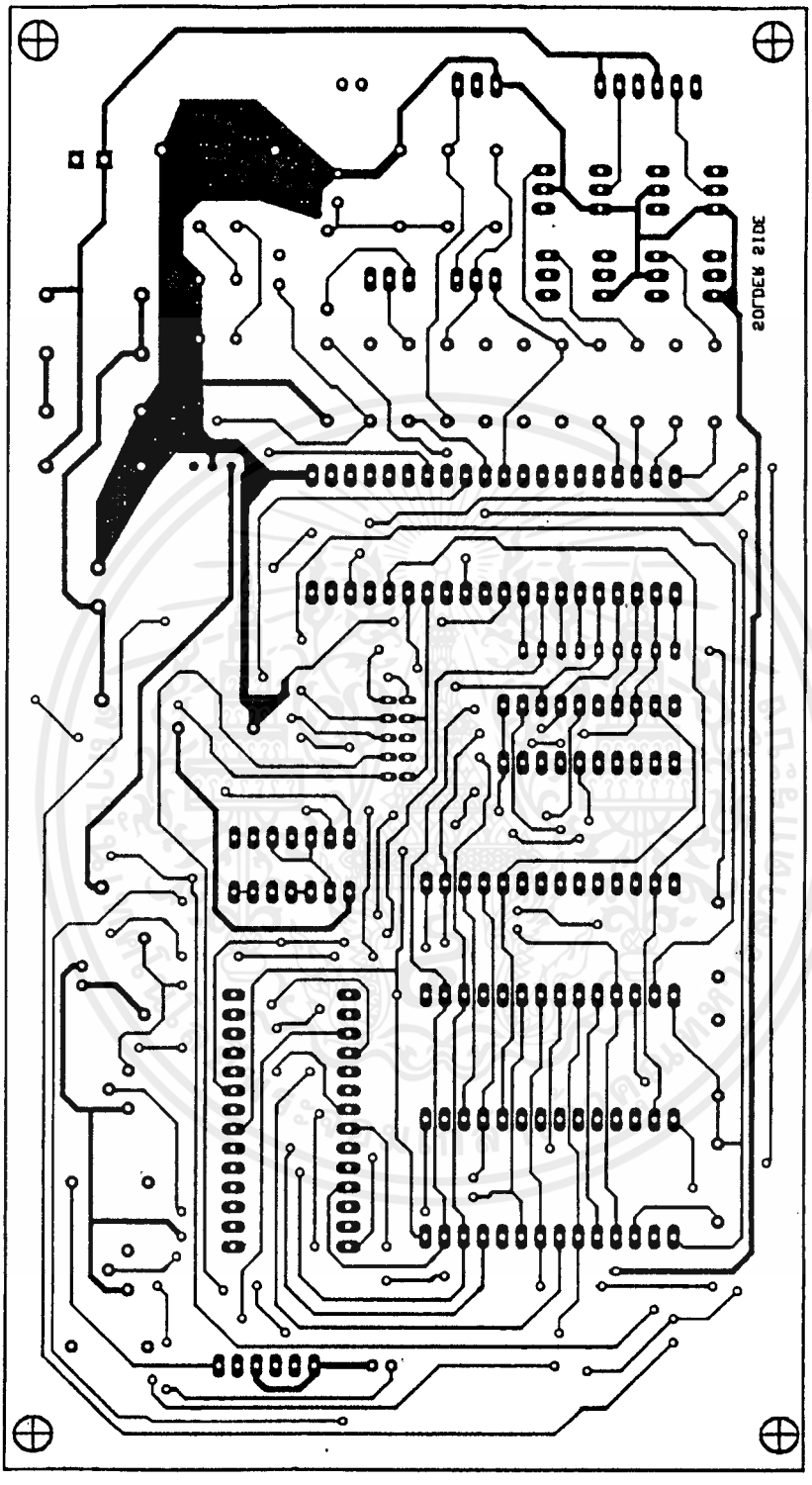


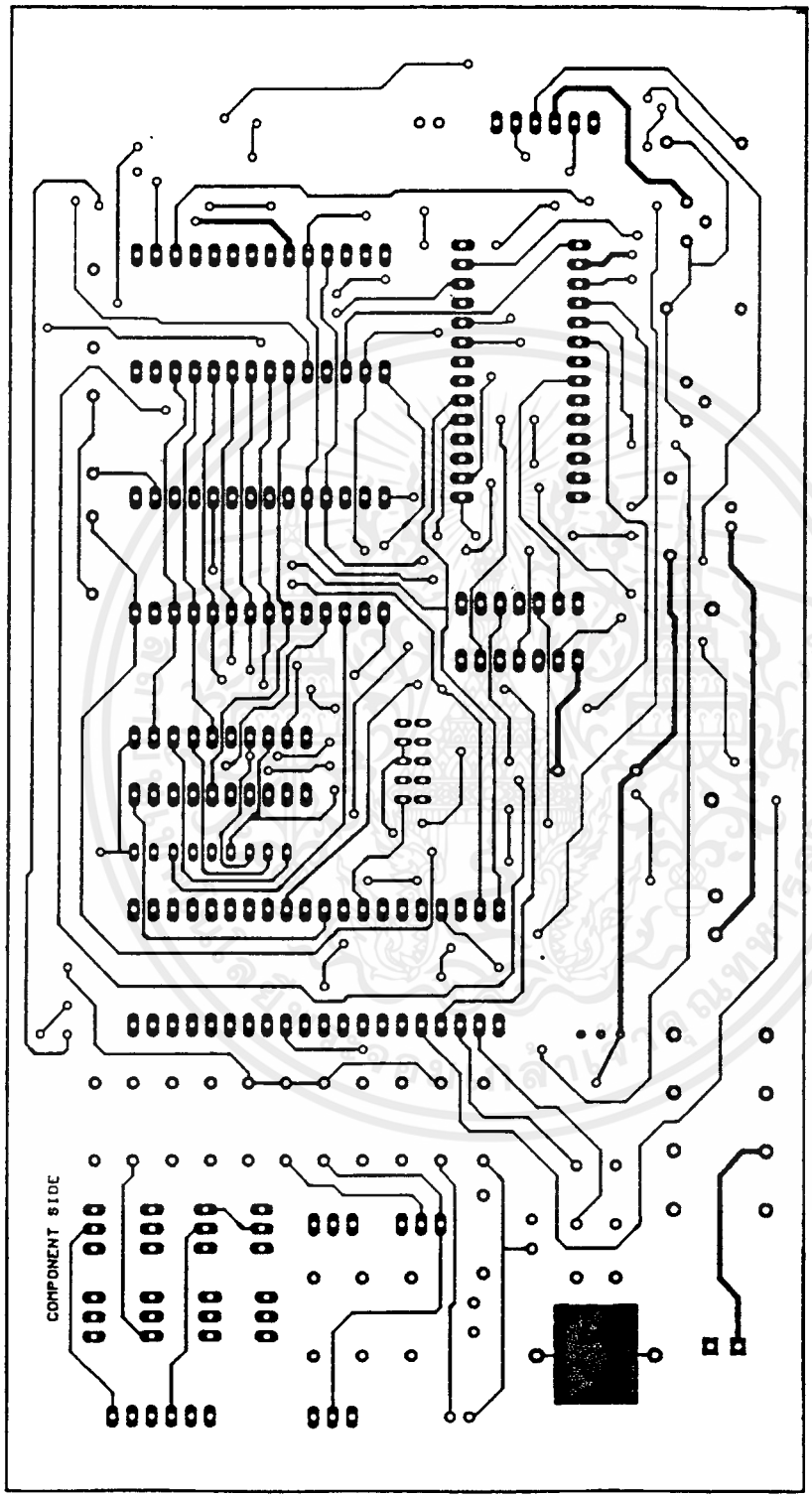
Serial Port		
Title		
By VEERA E.		
Size A	Document Number	REV 1/A
Date: March 16, 1993	Sheet 4 of	4



Antenna Pattern Tester		
Title	By VEERA E.	
Size A	Document Number	REV 1/A
Date: March 16, 1993	Sheet 3 of 4	







PARTS LIST

U1	8031
U2	74LS373
U3	27256
U4	6264
U5	ADC0809
U6	74LS02
U7	7805

Q1,Q2	TR #BC547
Q3,Q4,Q5,Q6	OPTO ISOLATOR TR #4N26
X1	X-TAL 11.0592MHz
D1,D2,D3	1N4148
D4	LED
D5	BRIDGE RECTIFIER
C1,C2	22 PF
C3,C4	10 MF
C5	1000 MF
C6	0.1 MF

VR1,VR2	10 Kohm
R1,R2,R3	33 Kohm
R4	R-PACK X8 10Kohm
R5	1 Kohm
R6	8.2 Kohm
R7,R8,R9	4.7 Kohm
R10	100 ohm
R11	1.8 Kohm
R12	4.7 Kohm
R13,R14,R15,R16	120 ohm
R17	1 Kohm
R18,R19	15 ohm

ภาคผนวก ข

SPECIFICATIONS

A. Model:KR-5400A

§ AZIMUTH (KR-400)

Rotation Torque	:	600 Kg.- Cm.(520 Lbs.- In.)
Stationary Brake Torque	:	2000 Kg.- Cm.(1730 Lbs.- In.)
Vertical Load	:	200 Kg. (440 Lbs.)
End-of-Rotation Stopper	:	Mechanical
Rotation Time	:	60sec./50Hz, 50sec./60Hz

Thermostat is installed.

§ ELEVATION (KR-500)

Rotation Torque	:	1000 Kg. Cm.(866 Lbs.-In.)
Stationary Brake Torque	:	2000 Kg.- Cm.(1730 Lbs.- In.)
End-of-Rotation Stopper	:	Mechanical
Rotation	:	0° to 180° (+5°/-0°)

Thermostat is installed.

Permissible Mast Size	:	Ø38 - Ø63
Permissible Boom Size	:	Ø32 - Ø43
Continuous Operation Time	:	Max. 5 Minutes
Antenna Wind Load Area	:	Less than 0.8M²
Control Cable	:	6 Conductor
Input Voltage	:	AC 115/230V, 50Hz/60Hz
Motor (Rotor Unit)	:	AC24V
Meter Indication Difference	:	±4°
Weight (Incl. Rotor & Clamps)	:	12 Kgs.

B. Model:KR-5600A

§ AZIMUTH (KR-600X)

Rotation Torque	:	700 Kg.- Cm.(606 Lbs.-In.)
Stationary Brake Torque	:	4000 Kg.- Cm.(3460 Lbs.- In.)
Vertical Load	:	200 Kg. (440 Lbs.)
End-of-Rotation Stopper	:	Electrical/Mechanical
Rotation Time	:	60sec./50Hz, 50sec./60Hz

Thermostat is installed.

§ ELEVATION (KR-500)

Rotation Torque	:	1000 Kg.- Cm.(866 Lbs.- In.)
Stationary Brake Torque	:	2000 Kg.- Cm.(1730 Lbs.- In.)
End-of-Rotation Stopper	:	Mechanical
Rotation	:	0° - 180° (+5°/-0°)

Permissible Mast Size	:	Ø38 - Ø63
Permissible Boom Size	:	Ø32 - Ø43
Continuous Operation Time	:	Max. 5 Minutes
Antenna Wind Load Area	:	Less than 0.8M²
Control Cable	:	6 Conductor
Input Voltage	:	AC 115/230V, 50Hz/60Hz
Motor (Rotor)	:	AC24V
Meter Indication Difference	:	±4°
Weight (Incl. Rotor & Clamps)	:	12 Kgs.

¶ A ROTOR UNIT OF ORDINARY KR-600RC and KR-600 DOES NOT USE WITH THE CONTROL BOX OF MODEL:KR-5600.

P A R T S L I S T

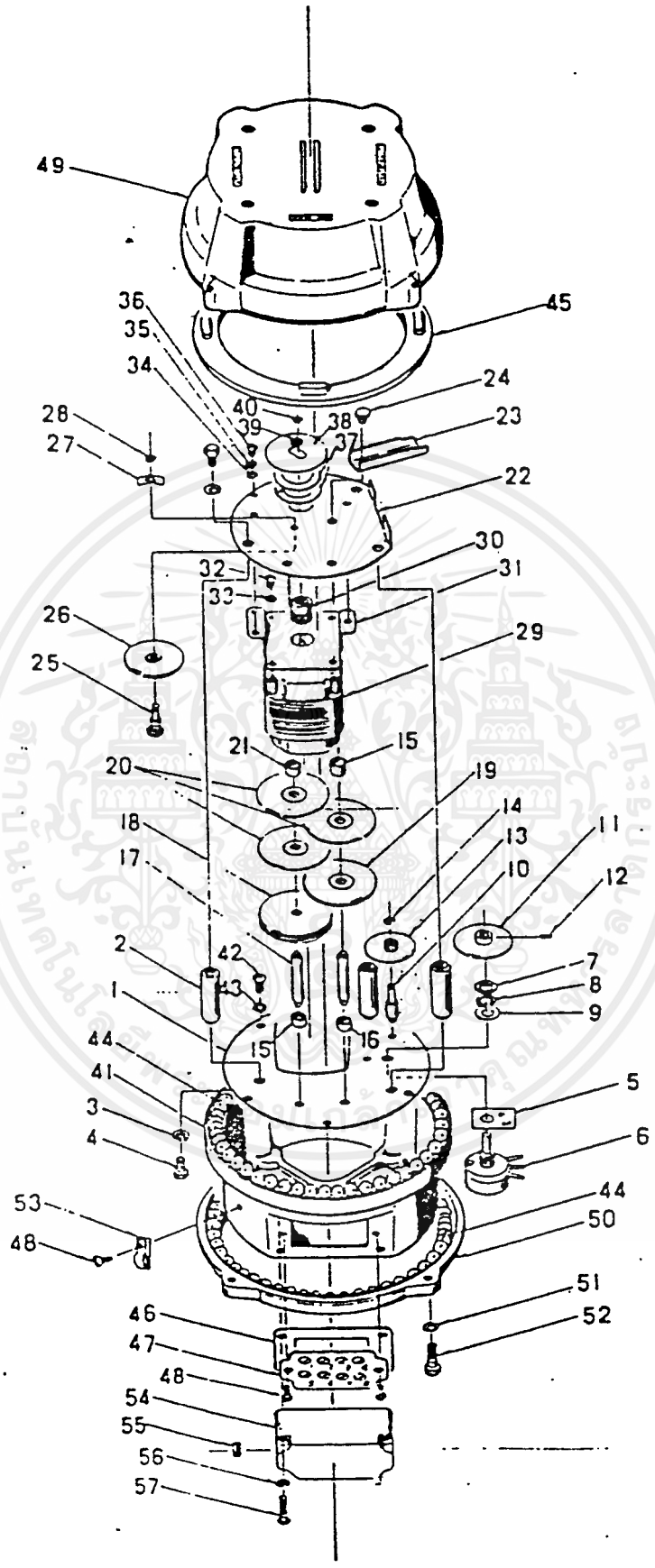
¶ AZIMUTH Rotor Unit (KR-400)

1. Gear Mount Plate
2. Gear Mount Support
3. Washer (Ø6)
4. Gear Mount Screw
5. Insulator Sheet
6. Potentiometer
10. Gear Pot Shaft
11. Pot. Devider Gear
12. Gear Stopper Screw
13. Plastic Pot. Gear
14. 'E' Ring
15.16.21. Stud Support Sleeve
17. Gear Shaft
18. Gear
19.20. Pinion/Gear Assmb.
22 Gear/Motor Mount Plate
23. Revolution Stopper
24. Fixing Pin
25. Gear Pot. Shaft
26. Plastic Gear
29. AC 24V Motor
30. Motor Pinion
31. Motor Mount Plate
32. Motor Mount Screw
33. Washer
34. Washer
35. Washer
36. Motor Holder Screw
37. Disc Pad
38. Brake Plate
39. Washer
40. Ø2.5 'E' Ring
41. Case
42. Gear Mount Plate Holder Screw
43. Washer
44. Ball Bearing
45. Internal Gear
46. Rubber Terminal Sheet
47. Terminal
48. Terminal/Cable Holder Screw
49. Rotor Housing
50. Housing
51. Washer
52. Housing Screw
53. Cable Holder
54. Terminal Cover
55. Rubber Grommet
56. Washer
57. Terminal Cover Screw

¶ AZIMUTH Rotor Unit (KR-600X)

1. Gear Mount Plate Assmb.
2. Gear Mount Support
3. Washer
4. Gear Mount Screw
5. Insulator Sheet
6. Potentiometer
7. Nut
8. Spring Washer
9. Washer
10. Gear Pot. Shaft
11. Pot. Devider Gear
12. Gear Stopper Screw
13. Plastic Pot. Gear
14. 'E' Ring
15.21. Stud Support Sleeve
16.17. Gear Shaft
18. Gear
19.20. Pinion/Gear Assmb.
22 Gear/Motor Mount Assmb.
29. Motor Assmb.
34. Washer
35.56. Washer
36. Screw for Motor Holder
37. Limit Switch
41. Case
42. Mount Plate Holder Screw
43. Washer
46. Rubber Sheet
47. Terminal
48. Terminal/Cable Holder Screw
49. Rotor Housing
50. Housing
51. Washer
52. Housing Screw
53. Cable Holder
54. Terminal Cover
55. Rubber Grommet
56. Spring Washer
57. Terminal Cover Screw
58. 59. Mast Clamp
60. Washer
61. Spring Washer
62.63. Screw
64. Nut
65. Condenser Bracket
66. Electrolytic Condenser (100mfd/60
67. Screw M3 x 6

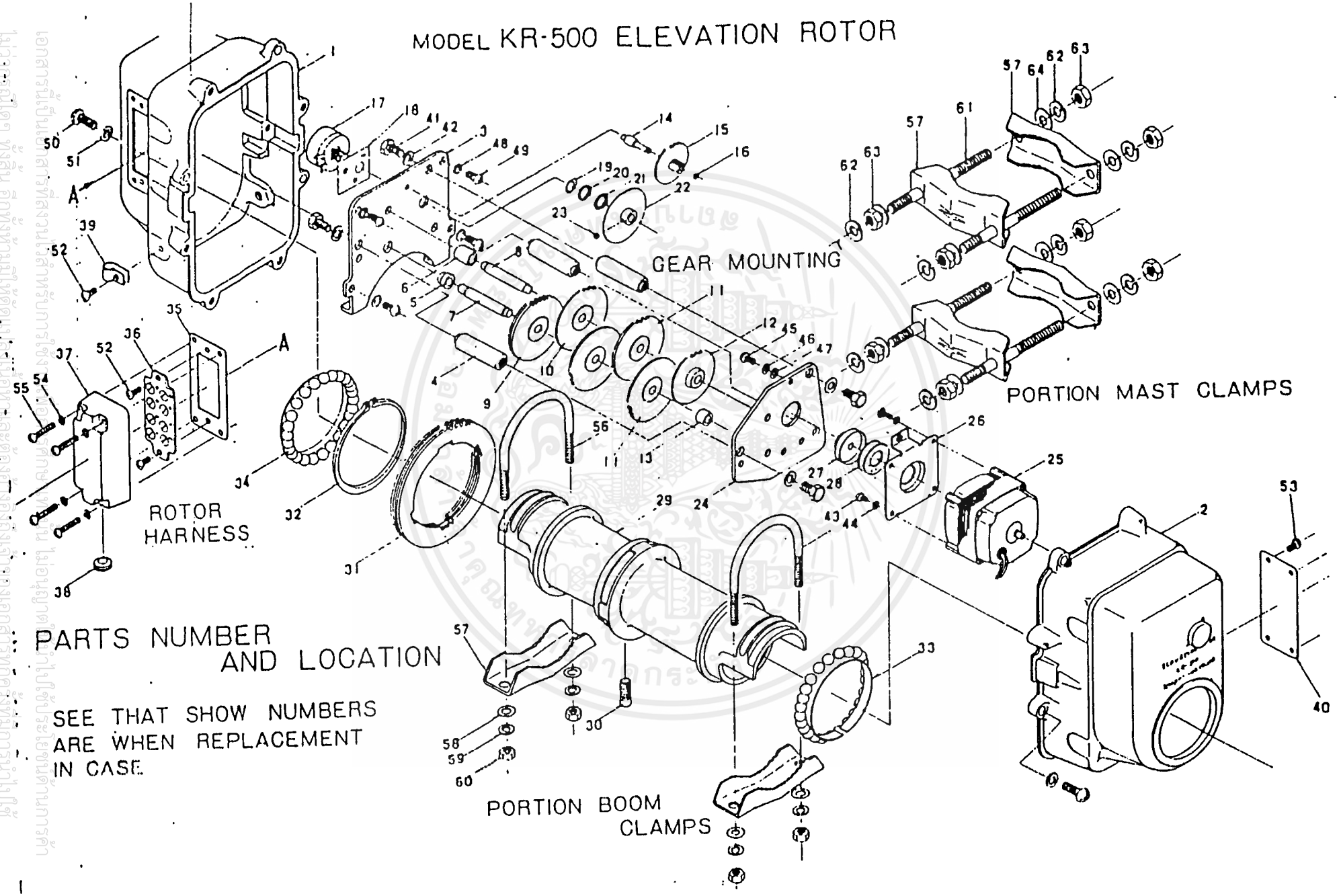
MODEL KR-400 ROTOR



PARTS NUMBER AND LOCATION

SEE THAT SHOW NUMBERS ARE WHEN
REPLACEMENT. IN CASE

MODEL KR-500 ELEVATION ROTOR



PARTS NUMBER AND LOCATION

SEE THAT SHOW NUMBERS ARE WHEN REPLACEMENT IN CASE

ภาคผนวก ค.

National Semiconductor

54F/74F373
Octal Transparent Latch with TRI-STATE® Outputs

General Description

The 'F373 consists of eight latches with TRI-STATE outputs for bus organized system applications. The flip-flops appear transparent to the data when Latch Enable (LE) is HIGH. When LE is LOW, the data that meets the setup times is latched. Data appears on the bus when the Output Enable ($\overline{OE}$) is LOW. When $\overline{OE}$ is HIGH the bus output is in the high impedance state.

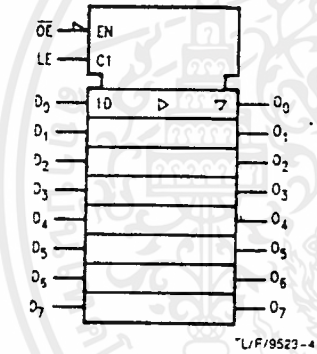
Features

- Eight latches in a single package
- TRI-STATE outputs for bus interfacing
- Guaranteed 4000V minimum ESD protection

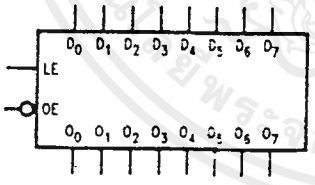
Ordering Code: See Section 5

Logic Symbols

IEEE/IEC



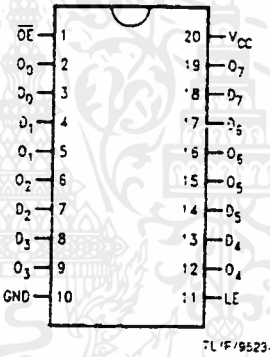
TL/F/9523-4



TL/F/9523-

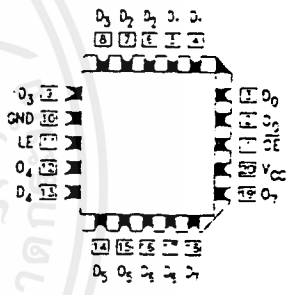
Connection Diagrams

Pin Assignment for DIP, SOIC and Flatpak



TL/F/9523-2

Pin Assignment for LCC



TL/F/9523-3

Unit Loading/Fan Out: See Section 2 for U.L. Definitions

Pin Names	Description	54F/74F	
		U.L. HIGH/LOW	Input I_{IH}/I_{IL} Output I_{OH}/I_{OL}
D_0–D_7	Data Inputs	1.0/1.0	20 μA/−0.6 mA
LE	Latch Enable Input (Active HIGH)	1.0/1.0	20 μA/−0.6 mA
$\overline{OE}$	Output Enable Input (Active LOW)	1.0/1.0	20 μA/−0.6 mA
O_0–O_7	TRI-STATE Latch Outputs	150/40 (33.3)	−3 mA/24 mA (20 mA)

Functional Description

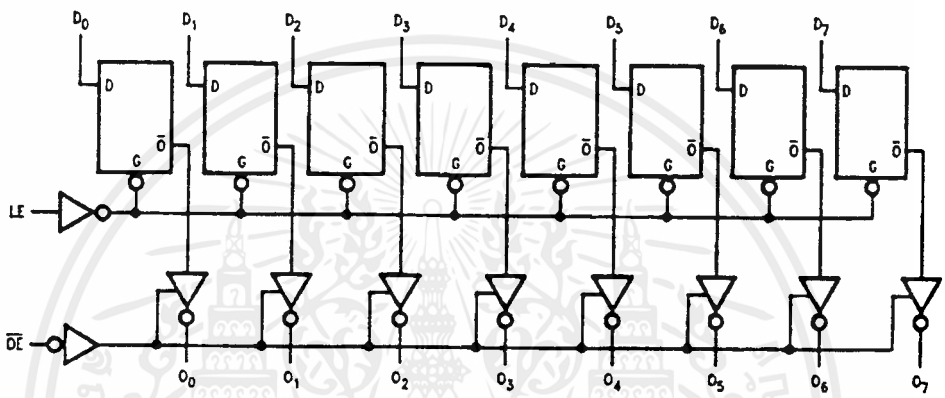
The 'F373 contains eight D-type latches with TRI-STATE output buffers. When the Latch Enable (LE) input is HIGH, data on the D_n inputs enters the latches. In this condition the latches are transparent, i.e., a latch output will change state each time its D input changes. When LE is LOW, the latches store the information that was present on the D inputs a setup time preceding the HIGH-to-LOW transition of LE. The TRI-STATE buffers are controlled by the Output Enable ($\overline{OE}$) input. When $\overline{OE}$ is LOW, the buffers are in the bi-state mode. When $\overline{OE}$ is HIGH the buffers are in the high impedance mode but this does not interfere with entering new data into the latches.

Truth Table

Inputs			Output
LE	$\overline{OE}$	D_n	O_n
H	L	H	H
H	L	L	L
L	L	X	O_n (no change)
X	H	X	Z

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial
Z = High Impedance State

Logic Diagram



TL/F/9523–5

Please note that this diagram is provided only for the understanding of logic operations and should not be used to estimate propagation delays.

National Semiconductor

Analog-to-Digital Converters

ADC0808, ADC0809 8-Bit μP Compatible A/D Converters With 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

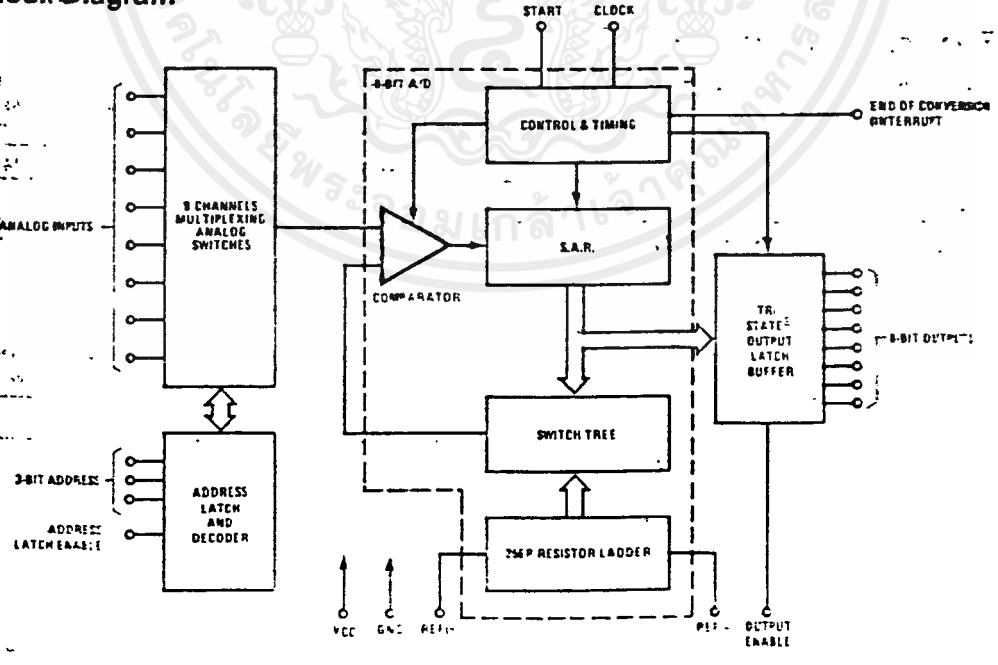
The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet.

Features

- Resolution — 8-bits
- Total unadjusted error — ± 1/2 LSB and ± 1 LSB
- No missing codes
- Conversion time — 100 μs
- Single supply — 5 V_{DC}
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- 8-channel multiplexer with latched control logic
- Easy interface to all microprocessors, or operates "stand alone"
- Outputs meet T^2L voltage level specifications
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Standard hermetic or molded 28-pin DIP package
- Temperature range −40°C to +85°C or −55°C to +125°C
- Low power consumption — 15 mW
- Latched TRI-STATE® output

Block Diagram



Absolute Maximum Ratings (Notes 1 and 2)

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage at Any Pin Except Control Inputs	−0.3V to (V_{CC} + 0.3V)
Voltage at Control Inputs (START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)	−0.3V to +15V
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A = 25°C	875 mW
Lead Temperature (Soldering, 10 seconds)	300°C

Operating Ratings (Notes 1 and 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le$
ADC0808CJ	−55°C ≤ T_A ≤ +
ADC0808CCJ, ADC0808CCN, ADC0809CCN	−40°C ≤ T_A ≤ +
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.

Electrical Characteristics

Converter Specifications: V_{CC} = 5 V_{DC} = $V_{REF(+)}$, $V_{REF(-)}$ = GND, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK} = 640 kHz unless otherwise stated.

	Parameter	Conditions	Min	Typ	Max	Units
	ADC0808					
	Total Unadjusted Error	25°C			± 1/2	LSB
	(Note 5)	T_{MIN} to T_{MAX}			± 3/4	LSB
	ADC0809					
	Total Unadjusted Error	0°C to 70°C			± 1	LSB
	(Note 5)	T_{MIN} to T_{MAX}			± 1 1/4	LSB
	Input Resistance	From Ref(+) to Ref(−)	1.0	2.5		kΩ
	Analog Input Voltage Range	(Note 4) V(+) or V(−)	GND−0.10		V_{CC}+0.10	V
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	V_{CC}+0.1	V
$\frac{V_{REF(+)}+V_{REF(-)}}{2}$	Voltage, Center of Ladder		V_{CC}/2−0.1	V_{CC}/2	V_{CC}/2+0.1	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V
	Comparator Input Current	f_c = 640 kHz, (Note 6)	−2	±0.5	2	μA

Electrical Characteristics

Digital Levels and DC Specifications: ADC0808CJ 4.5V ≤ V_{CC} ≤ 5.5V, −55°C ≤ T_A ≤ +125°C unless otherwise noted
ADC0808CCJ, ADC0808CCN, and ADC0809CCN 4.75 ≤ V_{CC} ≤ 5.25V, −40°C ≤ T_A ≤ +85°C unless otherwise noted

	Parameter	Conditions	Min	Typ	Max	Units
ANALOG MULTIPLEXER						
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 5V,				
		T_A = 25°C		10	200	nA
		T_{MIN} to T_{MAX}			1.0	μA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 0,				
		T_A = 25°C	−200	−10		nA
		T_{MIN} to T_{MAX}	−1.0			μA
CONTROL INPUTS						
$V_{IN(1)}$	Logical "1" Input Voltage		V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage				1.5	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN} = 15V			1.0	μA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN} = 0	−1.0			μA
I_{CC}	Supply Current	f_{CLK} = 640 kHz		0.3	3.0	mA

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0808CJ $4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_A \le +125°C$ unless otherwise noted; ADC0808CCJ, ADC0808CCN, and ADC0809CCN $4.75 \le V_{CC} \le 5.25V$, $-40°C \le T_A \le +85°C$ unless otherwise noted.

Parameter	Conditions	Min	Typ	Max	Units
DATA OUTPUTS AND EOC (INTERRUPT)					
$V_{OUT(1)}$ Logical "1" Output Voltage	$I_O = -360\ \mu A$	$V_{CC}-0.4$			V
$V_{OUT(0)}$ Logical "0" Output Voltage	$I_O = 1.6$ mA			0.45	V
$V_{OUT(0)}$ Logical "0" Output Voltage EOC	$I_O = 1.2$ mA			0.45	V
I_{OUT} TRI-STATE Output Current	$V_O = 5V$			3	μA
	$V_O = 0$	-3			μA

Electrical Characteristics

Timing Specifications: $V_{CC} = V_{REF(+)} = 5V$, $V_{REF(-)} = GND$, $t_r = t_f = 20$ ns and $T_A = 25°C$ unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	*(Figure 5)*		100	200	ns
t_{WALE}	Minimum ALE Pulse Width	*(Figure 5)*		100	200	ns
t_{SET-UP}	Minimum Address Set-Up Time	*(Figure 5)*		25	50	ns
t_H	Minimum Address Hold Time	*(Figure 5)*		25	50	ns
t_D	Analog MUX Delay Time From ALE	$R_S = 0\Omega$ *(Figure 5)*		1	2.5	μs
t_{1H}, t_{H0}	OE Control to Q Logic State	$C_L = 50$ pF, $R_L = 10$k *(Figure 8)*		125	250	ns
t_{1Z}, t_{0Z}	OE Control to Hi-Z	$C_L = 10$ pF, $R_L = 10$k *(Figure 8)*		125	250	ns
t_C	Conversion Time	$f_c = 640$ kHz, *(Figure 5)* (Note 7)	90	100	116	μs
f_C	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	*(Figure 5)*	0		8 + 2 μs	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE® Output Capacitance	At TRI-STATE® Outputs, (Note 12)		10	15	pF

Note 1: Absolute maximum ratings are those values beyond which the life of the device may be impaired.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: A zener diode exists, internally, from V_{CC} to GND and has a typical breakdown voltage of 7 V_{DC}.

Note 4: Two on-chip diodes are tied to each analog input which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. The spec allows 100 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 100 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.900 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors. See *Figure 3*. None of these A/Ds requires a zero or full-scale 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 4.5V full-scale) the reference voltages can be adjusted to achieve this. See *Figure 13*.

Note 6: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence *(Figure 6)*. See paragraph 4.0.

Note 7: The outputs of the data register are updated one clock cycle before the rising edge of EOC.

Functional Description

Multiplexer: The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table I shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE I

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach *(Figure 1)* was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +1/2 LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs n iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.

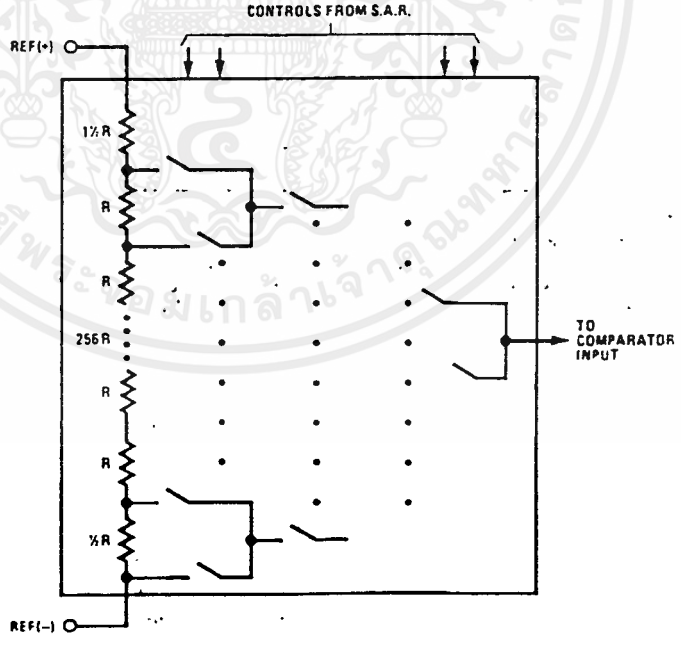


FIGURE 1. Resistor Ladder and Switch Tree

Functional Description (Continued)

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0808 as measured using the procedures outlined in AN-179.

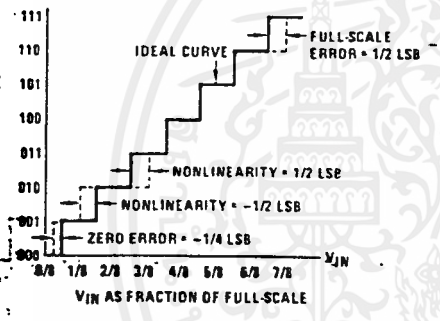


FIGURE 2. 3-Bit A/D Transfer Curve

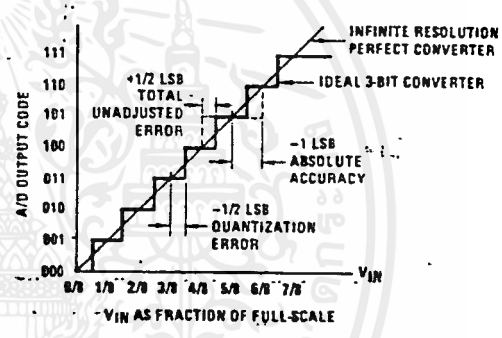


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

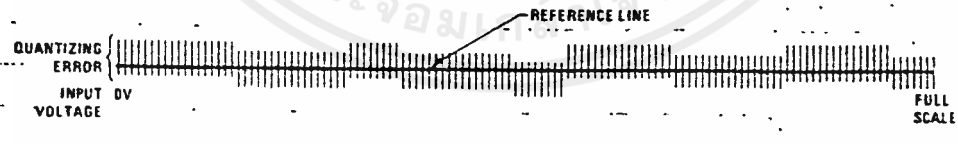
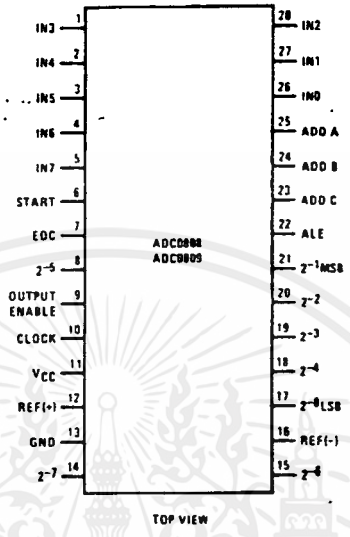


FIGURE 4. Typical Error Curve

Connection Diagram

Dual-In-Line Package



TOP VIEW

Timing Diagram

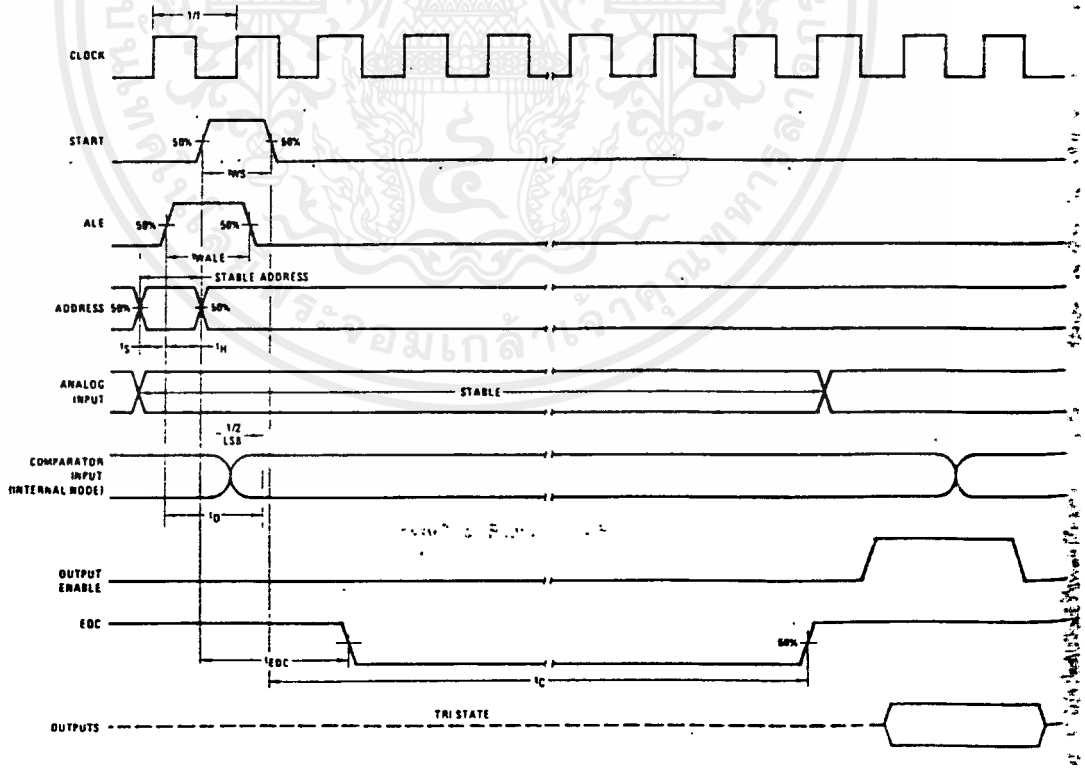


FIGURE 5

Typical Performance Characteristics

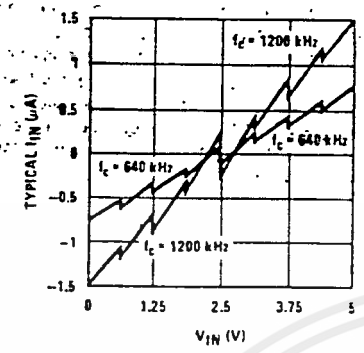


FIGURE 6. Comparator I_{IN} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)

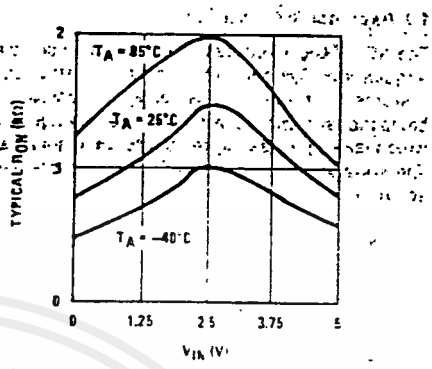


FIGURE 7. Multiplexer R_{ON} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)

TRI-STATE® Test Circuits and Timing Diagrams

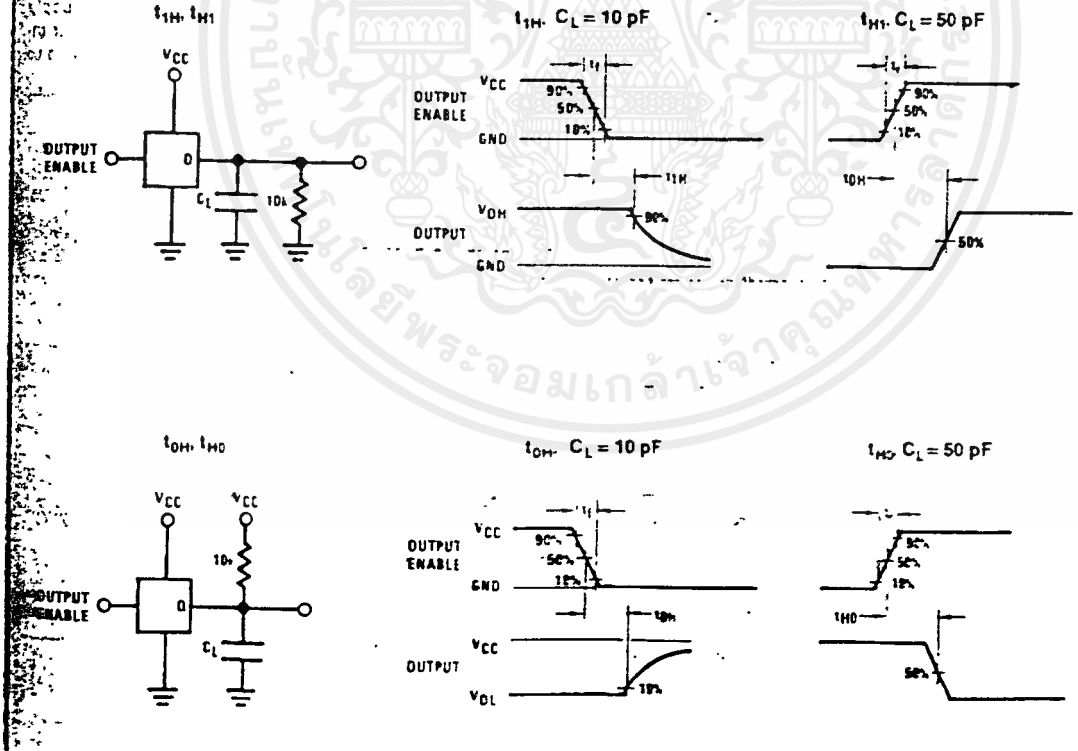


FIGURE 8

Applications Information

OPERATION

1.0 Ratiometric Conversion

The ADC0808, ADC0809 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0808 is expressed by the equation

$$\frac{V_{IN}}{V_{fs} - V_Z} = \frac{D_X}{D_{MAX} - D_{MIN}} \quad (1)$$

V_{IN} = Input voltage into the ADC0808
V_{fs} = Full-scale voltage
V_Z = Zero voltage
D_X = Data point being measured
D_{MAX} = Maximum data limit
D_{MIN} = Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0808, ADC0809 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs, *(Figure 9)*.

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc. are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a system reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC} = V_{REF} = 5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

2.0 Resistor Ladder Limitations

The voltages from the resistor ladder are compared to the selected input 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder, Ref(−), should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 10 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V is used, the supply should be adjusted to the same voltage within 0.1V.

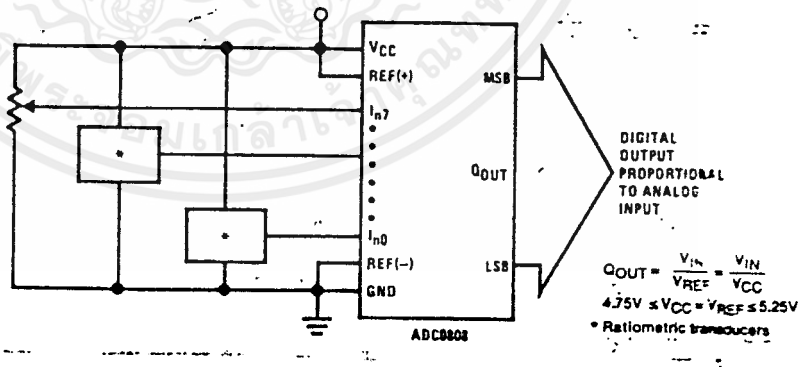


FIGURE 9. Ratiometric Conversion System

Applications Information (Continued)

The ADC0808 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 11* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 12*. The LM301 is overcompensated to insure stability when loaded by the 10 μF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 13*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB bit to be half the size of a 5V reference system.

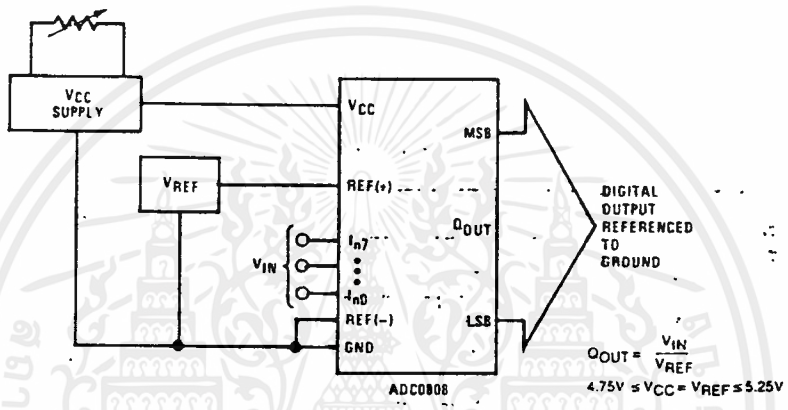


FIGURE 10. Ground Referenced Conversion System Using Trimmed Supply

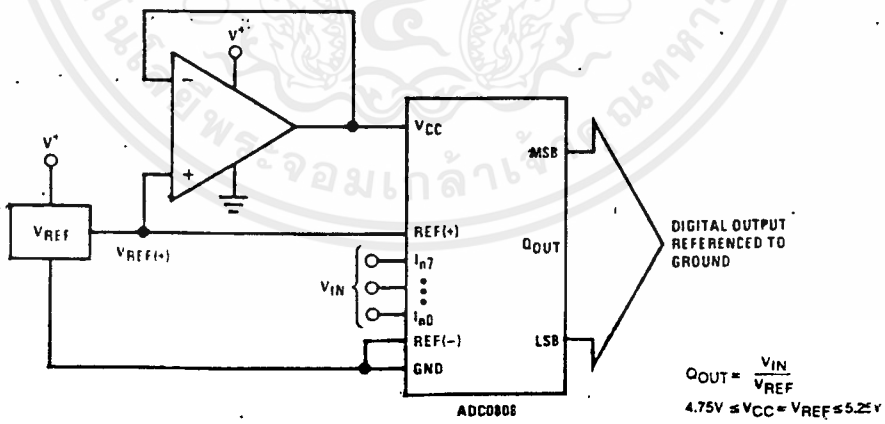


FIGURE 11. Ground Referenced Conversion System with Reference Generating V_{CC} Supply

Applications Information (Continued)

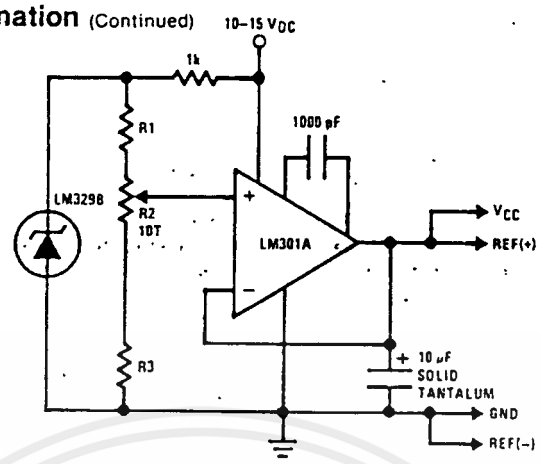


FIGURE 12. Typical Reference and Supply Circuit

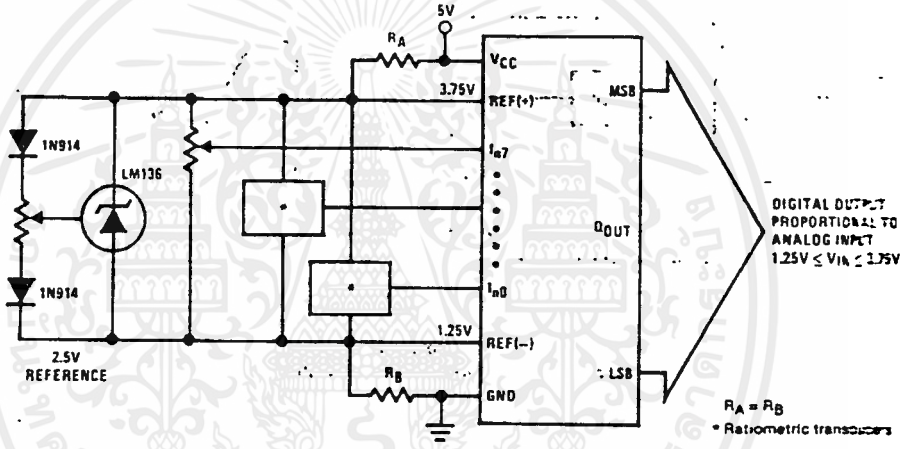


FIGURE 13. Symmetrically Centered Reference

3.0 Converter Equations

The transition between adjacent codes N and N+1 is given by:

$$V_{IN} = \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} + \frac{1}{512} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} = \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

where: V_{IN} = Voltage at comparator input
$V_{REF(+)}$ = Voltage at Ref(+)
$V_{REF(-)}$ = Voltage at Ref(−)
V_{TUE} = Total unadjusted error voltage (typically $V_{REF(+)} \div 512$)

4.0 Analog Comparator Inputs

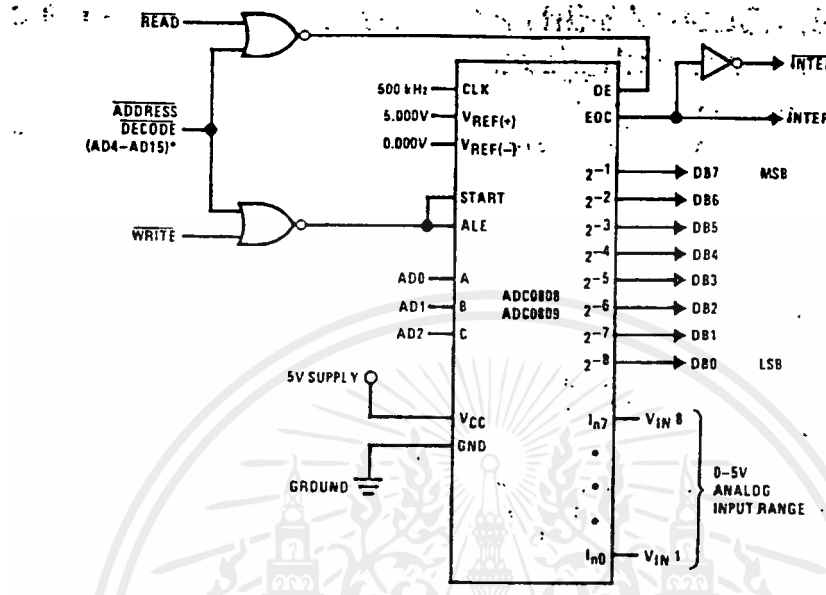
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6*.

If no filter capacitors are used at the analog inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally.

Typical Application



* Address latches needed for 8085 and SC/MP interfacing the ADC0808 to a microprocessor

MICROPROCESSOR INTERFACE TABLE

PROCESSOR	READ	WRITE	INTERRUPT (COMMENT)
8080	MEMR	MEMW	INTR (Thru RST Circuit)
8085	RD	WR	INTR (Thru RST Circuit)
Z-80	RD	WR	INT (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA•Φ2•R/W	VMA•Φ2•R/W	IRQA or IRQB (Thru PIA)

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C		−55°C to +125°C
Error	± 1/2 Bit Unadjusted	ADC0808CCN	ADC0808CCJ	ADC0808CJ
	± 1 Bit Unadjusted	ADC0809CCN		
Package Outline		N28A Molded DIP	J28A Hermetic DIP	J28A Hermetic DIP

Signetics

Microprocessor Division

SC80C31B/SC80C51B

CMOS Single-Chip 8-Bit Microcontroller

Product Specification

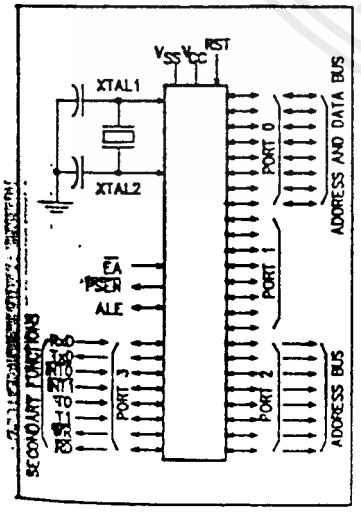
DESCRIPTION

The Signetics SC80C31B/SC80C51B is a high-performance microcontroller fabricated with Signetics high-density CMOS technology. The CMOS SC80C31B/SC80C51B is functionally compatible with the NMOS SCN8031/SCN8051 microcontrollers. The Signetics CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS. Signetics' epitaxial substrate minimizes latch-up sensitivity.

The SC80C31B/SC80C51B contains a 4K x 8 ROM, a 128 x 8 RAM, 32 I/O lines, two 16-bit counter/timers, a five-source, two priority level nested interrupt structure, a serial I/O port for either multi-processor communications, I/O expansion or full duplex UART, and on-chip oscillator and clock circuits.

In addition, the SC80C31B/SC80C51B has two software selectable modes of power reduction – idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

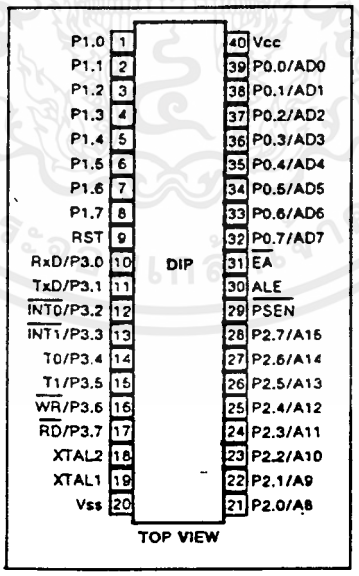
LOGIC SYMBOL



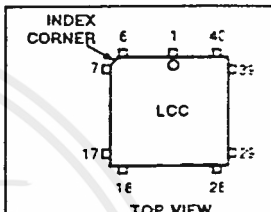
FEATURES

- SCN8031/SCN8051/SC80C51 compatible
  - 4K x 8 ROM
  - 128 x 8 RAM
  - Two 16-bit counter/timers
  - Full duplex serial channel
  - Boolean processor
- Memory addressing capability
  - 64K ROM and 64K RAM
- Power control modes:
  - Idle mode
  - Power-down mode
- CMOS and TTL compatible
- Three speed ranges at V_{CC} = 5V ±20%
  - 3.5 to 12MHz
  - 3.5 to 16MHz
  - 0.5 to 12MHz
- Three package styles

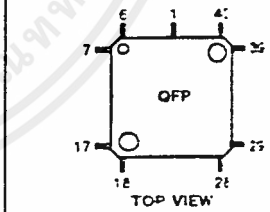
PIN CONFIGURATION



PIN CONFIGURATION (Cont)



Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	P2.0/A8
3	P1.1	25	P2.1/A9
4	P1.2	26	P2.2/A10
5	P1.3	27	P2.3/A11
6	P1.4	28	P2.4/A12
7	P1.5	29	P2.5/A13
8	P1.6	30	P2.6/A14
9	P1.7	31	P2.7/A15
10	RST	32	PSEN
11	RxD/P3.0	33	ALE
12	NC	34	NC
13	TxD/P3.1	35	EA
14	INT0/P3.2	36	P0.7/AD7
15	INT1/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	WR/P3.6	40	P0.3/AD3
19	RD/P3.7	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	Vss	44	VCC



Pin	Function	Pin	Function
1	NC	23	VSS
2	P1.0	24	P0.0/AD0
3	P1.1	25	P0.1/AD1
4	P1.2	26	P0.2/AD2
5	P1.3	27	P0.3/AD3
6	P1.4	28	P0.4/AD4
7	P1.5	29	P0.5/AD5
8	P1.6	30	P0.6/AD6
9	P1.7	31	P0.7/AD7
10	RST	32	EA
11	RxD/P3.0	33	ALE
12	NC	34	NC
13	TxD/P3.1	35	PSEN
14	INT0/P3.2	36	P2.7/A15
15	INT1/P3.3	37	P2.6/A14
16	T0/P3.4	38	P2.5/A13
17	T1/P3.5	39	P2.4/A12
18	WR/P3.6	40	P2.3/A11
19	RD/P3.7	41	P2.2/A10
20	XTAL2	42	P2.1/A9
21	XTAL1	43	P2.0/A8
22	Vss	44	VCC

CMOS Single-Chip 8-Bit Microcontroller SC80C31B/SC80C51B

ORDERING INFORMATION

SC80C □1B□□□□□ (CPxxxx)

ROMless/ROM
3 = ROMless
5 = ROM

Custom ROM Pattern No.
Applies to masked ROM versions only. Number will be assigned by Signetics. Contact Signetics sales office for ROM pattern submission requirements.

Pins
40 = 40-pin
44 = 44-pin

Package
A = Plastic LCC
N = Plastic DIP
B = Plastic QFP

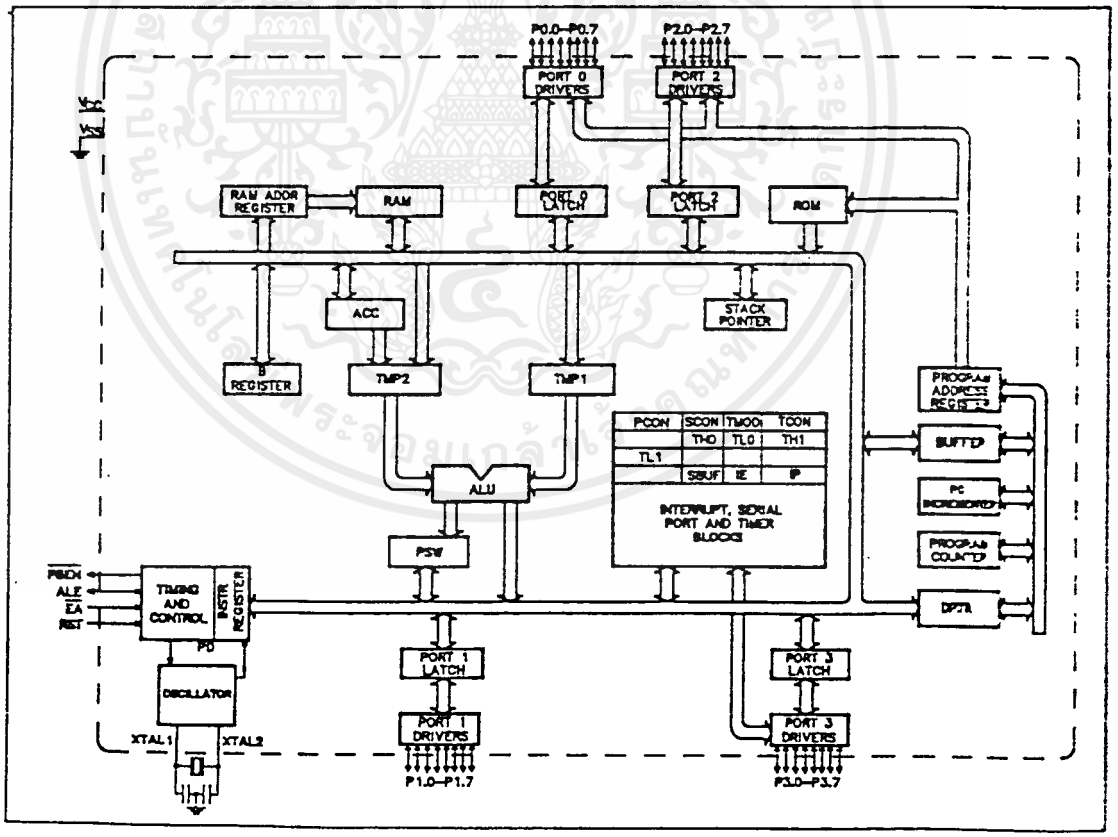
Speed
B = 0.5 to 12MHz
C = 3.5 to 12MHz
G = 3.5 to 16MHz

Operating Temperature Range
A = -40°C to +85°C
C = 0°C to +70°C

PART NUMBER SELECTION

ROMless	ROM	Temperature and Package	Frequency
SC80C31BCCN40	SC80C51BCCN40	0 to +70°C plastic DIP	3.5 to 12MHz
SC80C31BCGN40	SC80C51BCGN40	0 to +70°C plastic DIP	3.5 to 16MHz
SC80C31BCBN40	SC80C51BCBN40	0 to +70°C plastic DIP	0.5 to 12MHz
SC80C31BCCA44	SC80C51BCCA44	0 to +70°C plastic LCC	3.5 to 12MHz
SC80C31BCGA44	SC80C51BCGA44	0 to +70°C plastic LCC	3.5 to 16MHz
SC80C31BCBA44	SC80C51BCBA44	0 to +70°C plastic LCC	0.5 to 12MHz
SC80C31BACN40	SC80C51BACN40	-40 to +85°C plastic DIP	3.5 to 12MHz
SC80C31BAGN40	SC80C51BAGN40	-40 to +85°C plastic DIP	3.5 to 16MHz
SC80C31BACA44	SC80C51BACA44	-40 to +85°C plastic LCC	3.5 to 12MHz
SC80C31BAGA44	SC80C51BAGA44	-40 to +85°C plastic LCC	3.5 to 16MHz
SC80C31BCCB44	SC80C51BCCB44	0 to +70°C plastic QFP	3.5 to 12MHz
SC80C31BCGB44	SC80C51BCGB44	0 to +70°C plastic QFP	3.5 to 16MHz

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC/QFP	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	**Ground:** 0V reference.
		23	I	**Ground:** 0V reference. (QFP only)
V_{CC}	40	44	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–P0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pullups when emitting 1s. Port 0 also outputs the code bytes during program verification in the SC80C31B/SC80C51B. External pull-ups are required during program verification.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pullups. (See DC Electrical Characteristics: I_{IL}). Port 1 also receives the low-order address byte during program memory verification.
P2.0–P2.7	21–28	24–31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pullups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), port 2 emits the contents of the P2 special function register.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the SC80C51 family, as listed below:
	10	11	I	RxD (P3.0): Serial input port
	11	13	O	TxD (P3.1): Serial output port
	12	14	I	INT0 (P3.2): External interrupt
	13	15	I	INT1 (P3.3): External interrupt
	14	16	I	T0 (P3.4): Timer 0 external input
	15	17	I	T1 (P3.5): Timer 1 external input
	16	18	O	WR (P3.6): External data memory write strobe
	17	19	O	RD (P3.7): External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	I/O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** The read strobe to external program memory. When the SC80C31B/SC80C51B is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}$	31	35	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H and 0FFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFH.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	O	**Crystal 2:** Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 1.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide-by-two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

POWER-DOWN MODE

In the power-down mode the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON.

Table 1 shows the state of I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

Mode	Program Memory	ALE	PSEN	Port 0	Port 1	Port 2	Port 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Storage temperature range	−65 to +150	°C
Voltage on any other pin to V_{SS}	−0.5 to + 6.5	V
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C or T_A = −40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V

Symbol	Parameter	Test Conditions	Limits			Unit
			Min	Typical[1]	Max	
V_{IL}	Input low voltage, except $\overline{EA}$		−0.5		$0.2V_{CC}-0.1$	V
V_{IL1}	Input low voltage to $\overline{EA}$		0		$0.2V_{CC}-0.3$	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{CC}+.9$		$V_{CC}+0.5$	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{CC}$		$V_{CC}+0.5$	V
V_{OL}	Output low voltage, ports 1, 2, 3	I_{OL} = 1.6mA[2]			0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$	I_{OL} = 3.2mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3	V_{CC} = 5V±10%, I_{OH} = −60µA	2.4			V
		I_{OH} = −25µA	$0.75V_{CC}$			V
		I_{OH} = −10µA	$0.9V_{CC}$			V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$)[3]	V_{CC} = 5V±10%, I_{OH} = −800µA	2.4			V
		I_{OH} = −300µA	$0.75V_{CC}$			V
		I_{OH} = −80µA	$0.9V_{CC}$			V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.45V			−50	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3	See note 4			−650	µA
I_{LI}	Input leakage current, port 0	$V_{IN} = V_{IL}$ or V_{IH}			±10	µA
I_{CC}	Power supply current: Active mode @ 12MHz[5] Idle mode @ 12MHz[5] Power down mode	See note 6		11.5 1.3 3	25 4 50	 mA mA µA
R_{RST}	Internal reset pulldown resistor		50		300	kΩ
C_{IO}	Pin capacitance				10	pF

NOTES:
1. Typical ratings are based on a limited number of samples taken from early manufacturing lots and are not guaranteed. The values listed are at room temperature, 5V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100pF), the noise pulse on the ALE pin may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the $0.9V_{CC}$ specification when the address bits are stabilizing.
4. Pins of ports 1, 2, and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.
5. I_{CCMAX} at other frequencies is given by:
   Active mode: I_{CCMAX} = 0.94 X FREQ + 13.7
   Idle mode: I_{CCMAX} = 0.14 X FREQ + 2.3
   where FREQ is the external oscillator frequency in MHz. I_{CCMAX} is given in mA. See Figure 5.
6. See Figures 9 through 12 for I_{CC} test conditions.

AC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C or T_A = -40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V[1, 2]

SYMBOL	FIGURE	PARAMETER	12MHz CLOCK		VARIABLE CLOCK		UNIT
			Min	Max	Min	Max	
Program Memory							
$1/t_{CLCL}$	1	Oscillator frequency: **Speed Versions** SC80C31B/SC80C51B B SC80C31B/SC80C51B C SC80C31B/SC80C51B G			 0.5 3.5	 12 12	 MHz MHz
					3.5	16	MHz
t_{LHLL}	1	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	1	Address valid to ALE low	28		$t_{CLCL}-55$		ns
t_{LLAX}	1	Address hold after ALE low	48		$t_{CLCL}-35$		ns
t_{LLIV}	1	ALE low to valid instruction in		234		$4t_{CLCL}-100$	ns
t_{LLPL}	1	ALE low to $\overline{PSEN}$ low	43		$t_{CLCL}-40$		ns
t_{PLPH}	1	$\overline{PSEN}$ pulse width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	1	$\overline{PSEN}$ low to valid instruction in		145		$3t_{CLCL}-105$	ns
t_{PXIX}	1	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	1	Input instruction float after $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{AVIV}	1	Address to valid instruction in		312		$5t_{CLCL}-105$	ns
t_{PLAZ}	1	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	2, 3	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	2, 3	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	2, 3	$\overline{RD}$ low to valid data in		252		$5t_{CLCL}-165$	ns
t_{RHDX}	2, 3	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	2, 3	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	2, 3	ALE low to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	2, 3	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	2, 3	ALE low to $\overline{RD}$ or $\overline{WR}$ low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	2, 3	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	203		$4t_{CLCL}-130$		ns
t_{QVWX}	2, 3	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{WHQX}	2, 3	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	2, 3	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	2, 3	$\overline{RD}$ or $\overline{WR}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
External Clock							
t_{CHCX}	5	High time	20		20		ns
t_{CLCX}	5	Low time	20		20		ns
t_{CLCH}	5	Rise time		20		20	ns
t_{CHCL}	5	Fall time		20		20	ns
Shift Register							
t_{XLXL}	4	Serial port clock cycle time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	4	Output data setup to clock rising edge	700		$10t_{CLCL}-133$		ns
t_{XHQX}	4	Output data hold after clock rising edge	50		$2t_{CLCL}-117$		ns
t_{XHDX}	4	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	4	Clock rising edge to input data valid		700		$10t_{CLCL}-133$	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

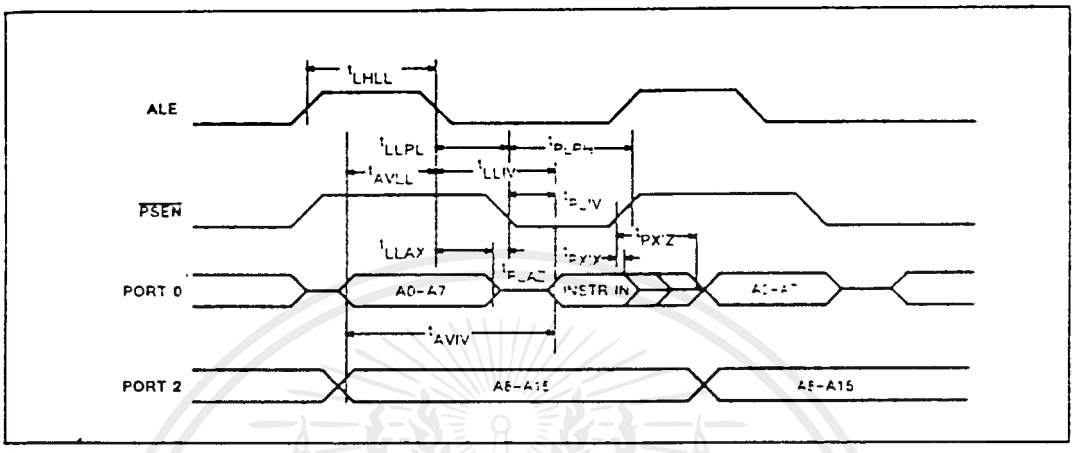


Figure 1. External Program Memory Read Cycle

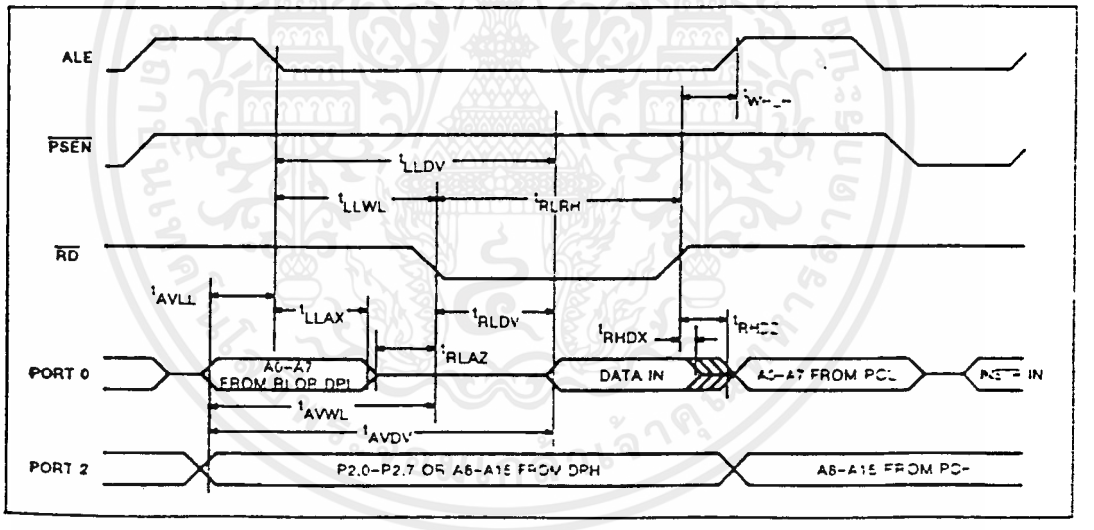


Figure 2. External Data Memory Read Cycle

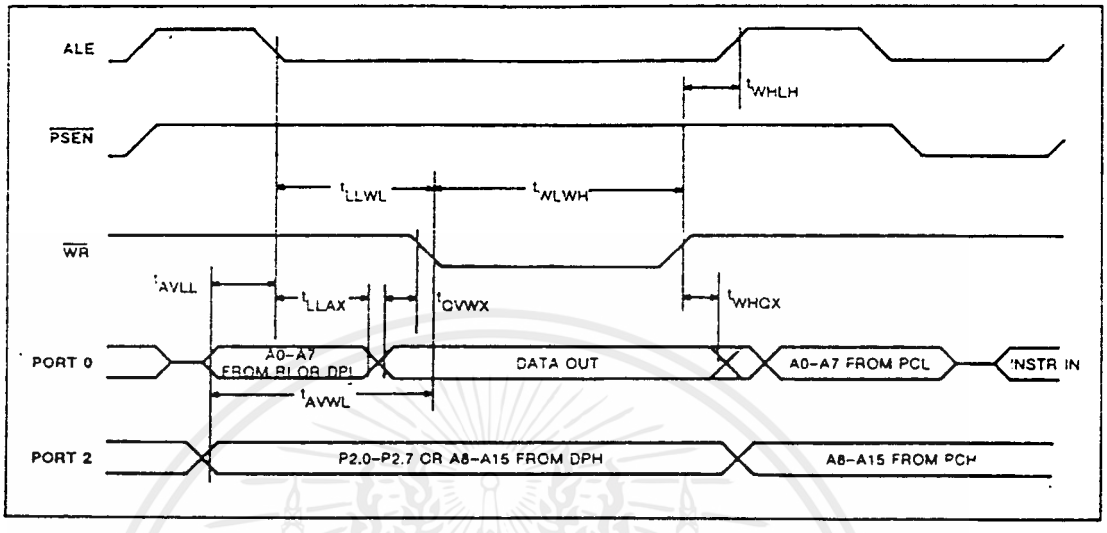


Figure 3. External Data Memory Write Cycle

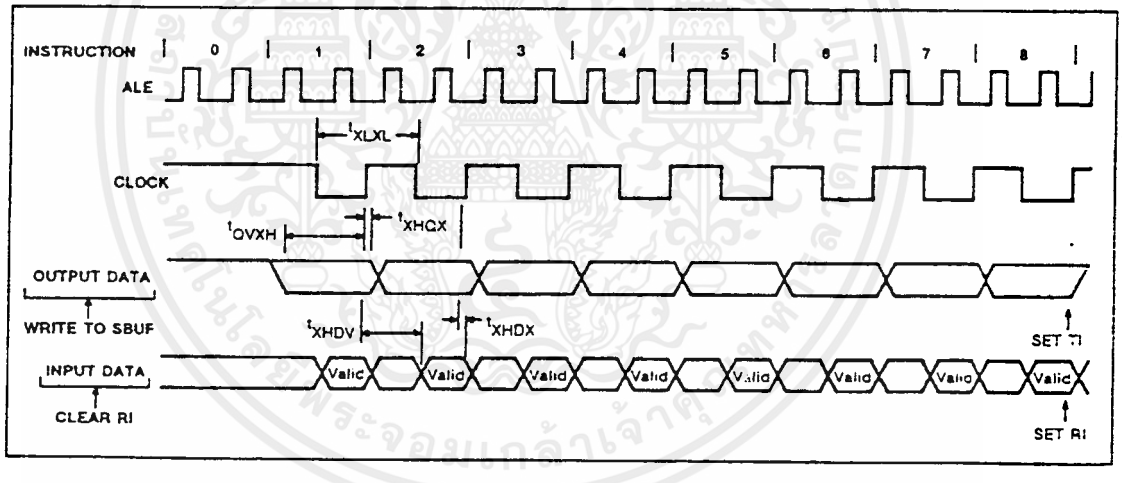


Figure 4. Shift Register Mode Timing

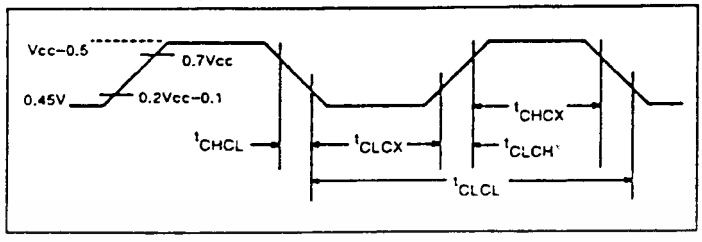


Figure 5. External Clock Drive

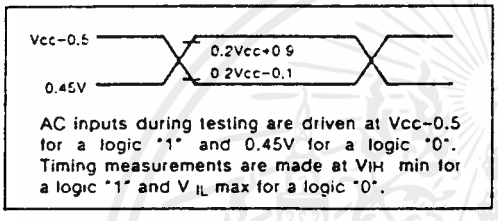


Figure 6. AC Testing Input/Output

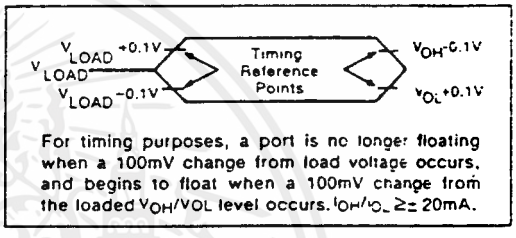


Figure 7. Float Waveform

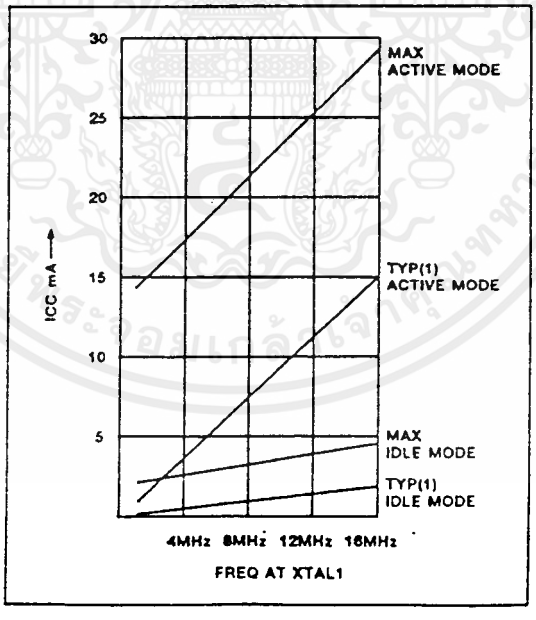


Figure 8. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test

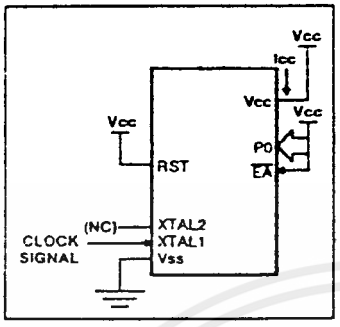


Figure 9. I_{CC} Test Condition, Active Mode
All other pins are disconnected

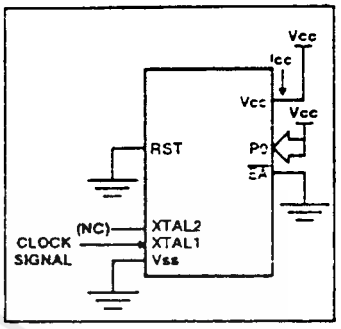


Figure 10. I_{CC} Test Condition, Idle Mode
All other pins are disconnected

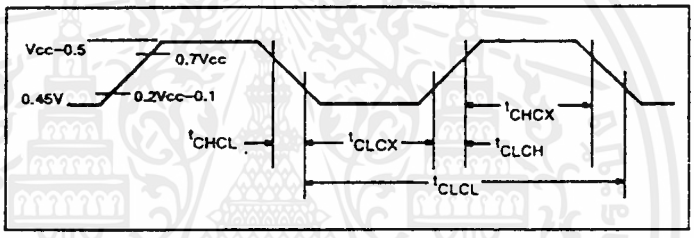


Figure 11. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
$t_{CLCH} = t_{CHCL} = 5ns$

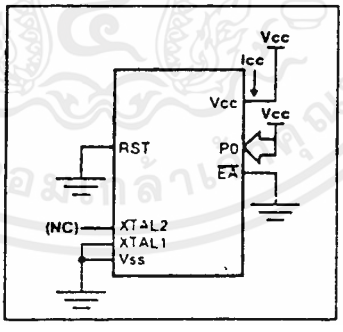


Figure 12. I_{CC} Test Conditions, Power Down Mode
All other pins are disconnected. V_{CC} = 2V to 5.5V

54374/74374 Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF
TI	SN54S374	J									SN54LS374	J													
	SN74S374	J	N								SN74LS374	J	N												
FAIRCHILD																									
MOTOROLA																									
NSC																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS374/SN74LS374

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommend operating conditions

		SN54LS374 MIN	NOM	MAX	SN74LS374 MIN	NOM	MAX	UNIT
Supply voltage V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output voltage V_{OH}				−400			−400	μA
High-level output current, I_{OH}				15			15	mA
Width of clock enabling pulse t_w	High	15			15			ns
	Low	15			15			
Data hold time, t_{hold}		0↑			0↑			ns
Setup time, t_{setup}		20↑			20↑			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

	PARAMETER	TEST CONDITIONS‡		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC}=MIN, I_I=−18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC}=MIN, V_{IH}=2V, V_{IL}=V_{IL} max, I_{OH}=MAX		2.4	3.1		V
V_{OL}	Low-level output voltage	V_{CC}=MIN, V_{IH}=2V, V_{IL}=V_{IL} max	I_{OL}=24mA		0.35	0.5	V
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC}=MAX, V_{IH}=2V, V_O=2.7V				20	μA
I_{OZL}	Off-state output current, low-level voltage applied	V_{CC}=MAX, V_{IH}=2V, V_O=0.4V				−20	μA
I_I	Input current at maximum input voltage	V_{CC}=MAX, V_I=7V				0.1	mA
I_{IH}	High-level input current	V_{CC}=MAX, V_I=2.7V				20	μA
I_{IL}	Low-level input current	V_{CC}=MAX, V_I=0.4V				−0.4	mA
I_{OS}	Short-circuit output current §	V_{CC}=MAX		−30		−130	mA
I_{CC}	Supply current	V_{CC}=MAX, Output control at 4.5V	LS374		27	40	mA

switching characteristics, V_{CC}=5V, T_A=25°C

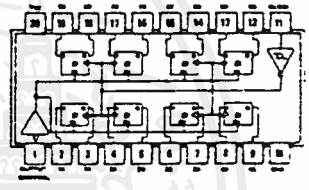
PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{max}			C_L=45pF, R_L=667Ω, See Notes 2 and 3	35	50		MHz
t_{PLH} / t_{PHL}	Data	Any Q					ns
t_{PLH}	Clock or enable	Any Q			15	23	ns
t_{PHL}					18	23	
t_{PZH}	Output Control	Any Q			20	23	ns
t_{PZL}					21	23	
t_{PHZ}	Output Control	Any Q	C_L=5pF, R_L=667Ω, See Note 3		17	25	ns
t_{PLZ}					14	25	

‡ For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
§ All typical values are at V_{CC}=5V, T_A=25°C.
¶ Not more than one output should be shorted at a time and duration of the short-circuit should not exceed one second.

NOTES 2. Maximum clock frequency is tested with all outputs loaded.
3. See load circuits and waveforms on page 3-11.
f_{max} = maximum clock frequency
t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
t_{PZH} = output enable time to high level
t_{PZL} = output enable time to low level
t_{PHZ} = output disable time from high level
t_{PLZ} = output disable time from low level

Pin Assignment (Top View)

①



SN54LS374 (J) SN74LS374 (J, N)
SN54S374 (J) SN74S374 (J, N)

'LS374, 'S374 FUNCTION TABLE

OUTPUT CONTROL	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

54373/74373 Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S373		J								SN54LS373		J												
	SN74S373		J								SN74LS373		J												
FAIRCHILD																									
MOTOROLA																									
N.S.C.																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS373/SN74LS373

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommend operating conditions

		SN54LS373			SN74LS373			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−1			−2.6	mA
High-level output voltage, V_{OH}				5.5			5.5	V
Pulse width t_w	Clock enable high	15			15			ns
	Clock enable high	15			15			
Setup time t_{SU}		0			0			ns
Hold time, t_{hold}		10			10			ns
Operating free-air temperature T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS‡		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = V_{IL} max, I_{OH} = MAX		2.4	3.1		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = V_{IL} max	I_{OL} = 24mA		0.35	0.5	V
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, V_{IH} = 2V, V_O = 2.7V				20	μA
I_{OZL}	Off-state output current, low-level voltage applied	V_{CC} = MAX, V_{IH} = 2V, V_O = 0.4V				−20	μA
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4V				−0.4	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−30		−130	mA
I_{CC}	Supply current	V_{CC} = MAX, Output control at 4.5V	'LS373		24	40	mA

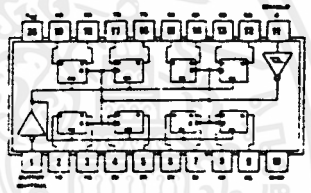
switching characteristics, V_{CC} = 5V, T_A = 25°C

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{max}							MHz
t_{PLH}	Data	Any Q	C_L = 45pF, R_L = 667Ω, See Notes 2 and 3		12	18	ns
t_{PHL}					12	18	
t_{PLH}	Clock or enable	Any Q			20	30	ns
t_{PHL}					18	30	
t_{PZH}	Output Control	Any Q			15	28	ns
t_{PZL}					25	36	
t_{PHZ}	Output Control	Any Q	C_L = 5pF, R_L = 667Ω, See Note 3		12	20	ns
t_{PLZ}					15	25	

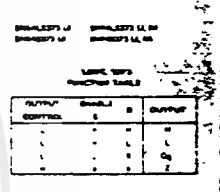
† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
§ Not more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

NOTES 2. Maximum clock frequency is tested with all outputs loaded.
3. See load circuits and waveforms on page 3-41.
f_{max} = maximum clock frequency
t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
t_{PZH} = output enable time to high level
t_{PZL} = output enable time to low level
t_{PHZ} = output disable time from high level
t_{PLZ} = output disable time from low level

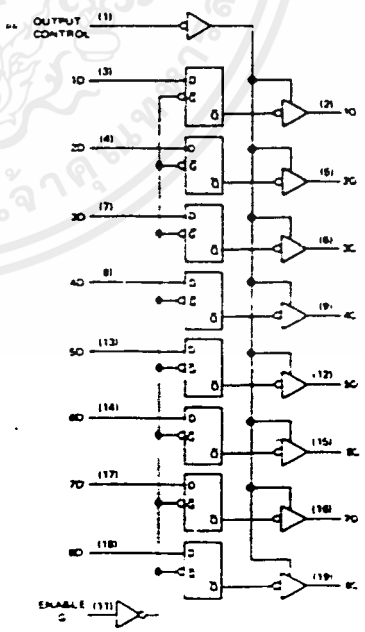
Pin Assignments (Top View)



SN54LS373 (J)
SN54S373 (J)
SN74LS373 (J, N)
SN74S373 (J, N)



'LS373 'S373
TRANSPARENT LATCHES



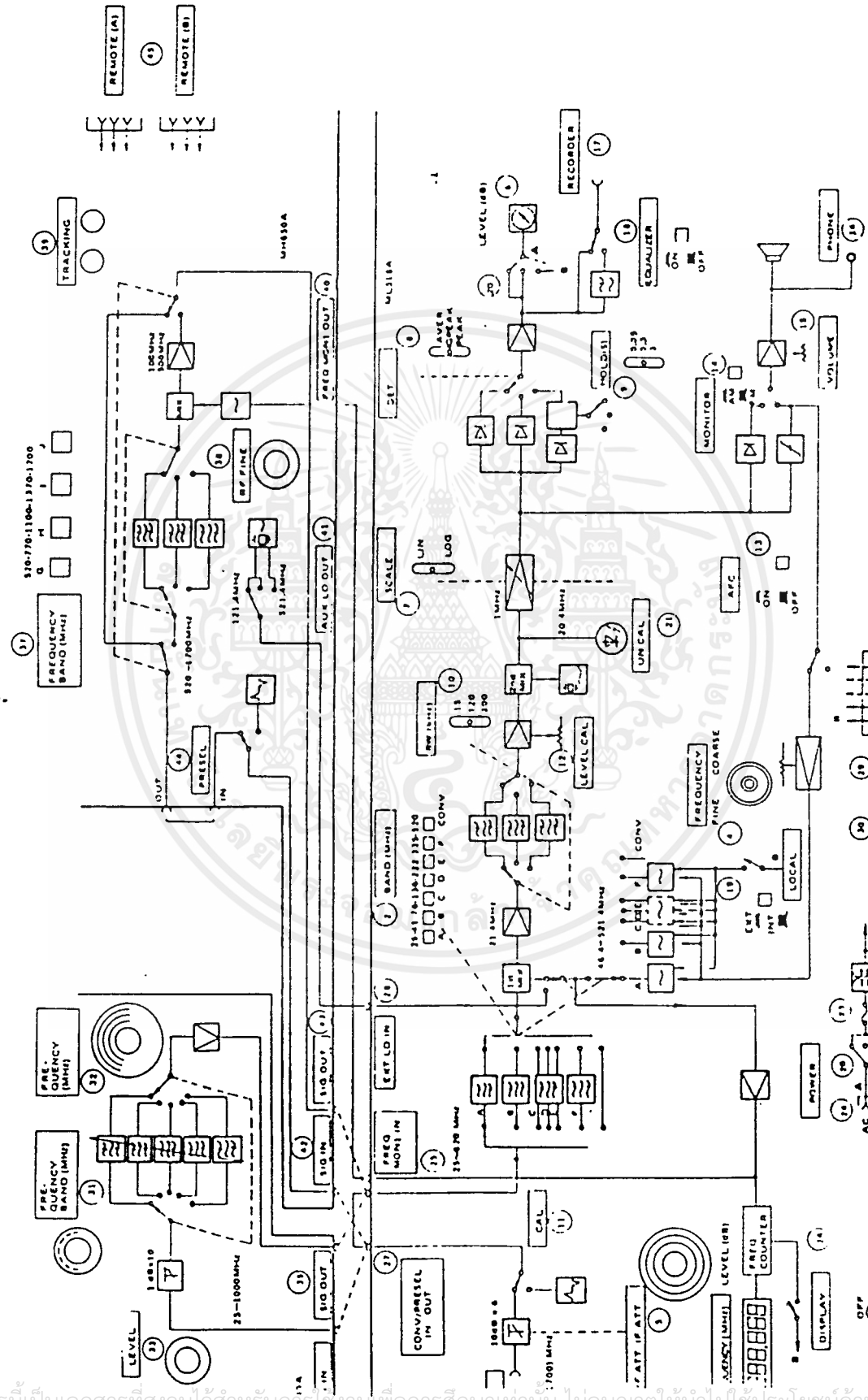


Fig. 4-3 Block Diagram

5.3.6 Recorder Output Check

a) Receive the signal at the operation of para. 5.3.1 and adjust the SG output level for maximum deflection of t indicator pointer.

b) Connect a voltmeter having an internal resistance of 1 or greater to the recorder output terminal and measur output voltage. This voltage must be 1 V ±20 %.

NOTE

The recorder output corresponds to the indicator scale and shows the characteristic shown in Fig. 5.2. This characteristics can be measured by connecting a voltmeter to the recorder output when checking the indicator scale as described in para. 5.3.5.

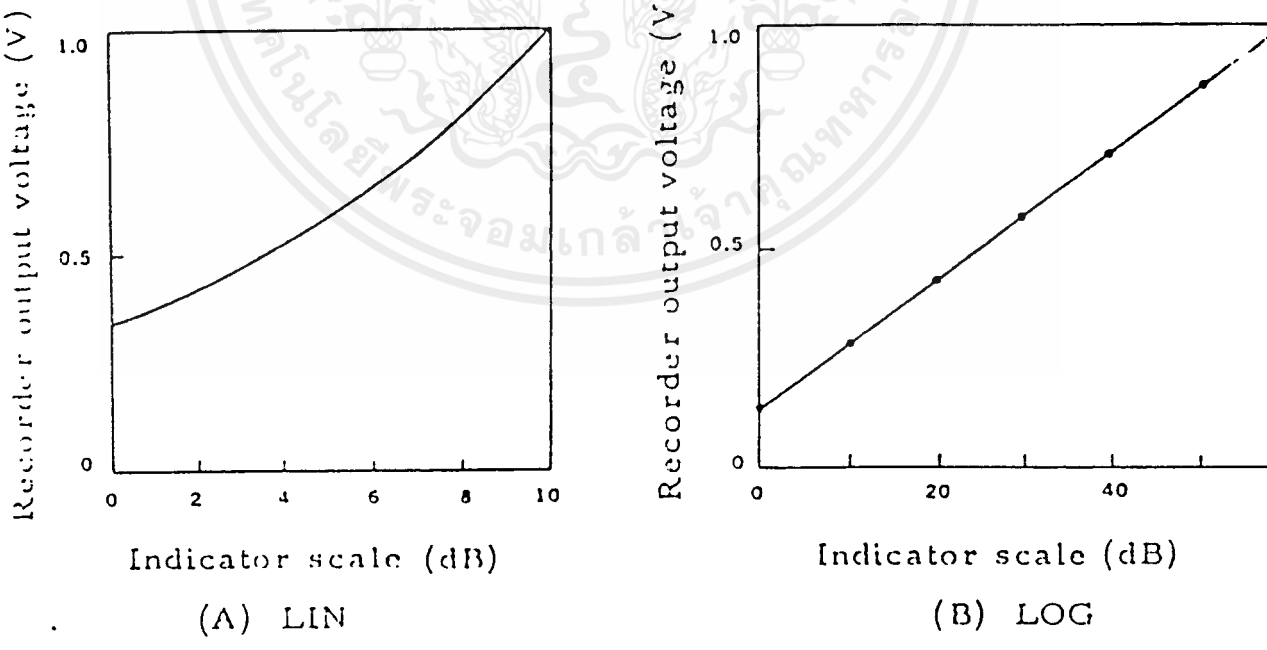


Fig. 5.2 Recorder Output Characteristics

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่องเครื่องทดสอบแพทเทิร์นของสายอากาศด้วยคอมพิวเตอร์นี้สำเร็จลุล่วงไปด้วยดี เพราะได้รับคำแนะนำ และ ช่วยเหลือจาก ผศ. พิพัฒน์ เลาหสงคราม . พี่โกมล,พี่สมศักดิ์ (กสท.) เพื่อนๆที่บ้านรุ่งอรุณ ตลอดจนการสื่อสารแห่งประเทศไทยและบริษัทซีเกตเทคโนโลยี (ประเทศไทย) จำกัด ที่ให้การเอื้อเฟื้อสถานที่และเครื่องมืออุปกรณ์ จึงขอแสดงความขอบคุณมา ณ.ที่นี้ด้วย

หนังสืออ้างอิง

1. โมไนย ไกรฤกษ์ และ วิวัฒน์ กิรานนท์ [1989], "ทฤษฎีสายอากาศ" คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า ลาดกระบัง

2. Richard F.Frerraro [1988], "Programmer's Guide to the EGA and VGA Card", Addison-wesley Publishing Company, Inc.,

3. บุญเลิศ เอี่ยมทัศนา [2531], "โปรแกรมคอมพิวเตอร์ ภาษาซี", บริษัท ซีเอ็ดยูเคชั่น จำกัด

4. Breadley Dyck Klier [1988]," EGA/VGA A Programmer's Referance Guide",McGraw-Hill Book Company.

5. Ben Ezzell , "Graphics Programming in Turbo C 2.0"

6. Steve Oualline [1992], "Advanced C Programming", A Division of Simon & Schuster,Inc.

7. รศ. มัณฑนา ปราการสมุทร [2534], "การเขียนชุดคำสั่งภาษาซี", บริษัท ควงกมลสมัย จำกัด.