ระบบควบคุมทิศทางสายอากาศ

DIRECTION ANTENNA CONTROLLER

จัดทำโดย

นาย ณรรฐพงษ์	พงษ์พานิช	34162152
นาย ทศพร	สาธรวิศิษฐ์	34162156
นาย นนทรัตน์	ศิริรัตน์	34162159
นาย ราชศักดิ์	รัฐธรรม	34162167
นาย สกล	พึ่งและ	34162173

อาจารย์ที่ปรึกษา

อ.เกษตร์ ศิริสันติสัมฤทธิ์

ปริญญานิพนธ์สำหรับปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม
ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะ วิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง ระบบควบคุมทิศทางเสาอากาศ

ผู้จัดทำ

นายณรรฐพงษ์	พงษ์พานิช	34162152
นายทศพร	สาธรวิศิษฐ์	34162156
นายนนทรัตน์	ศิริรัตน์	34162159
นายราชศักดิ์	รัฐธรรม	34162167
นายสกล	พึ่งและ	34162173

.....

(เกษตร์ ศิริสันติสัมฤทธิ์)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	ชุดควบคุมทิศทางของสายอากาศ	
นักศึกษา	ณรรฐพงษ์	พงษ์พานิช
	ทศพร	สาธรวิศิษฐ์
	นนทรัตน์	ศิริรัตน์
	ราชศักดิ์	รัฐธรรม
	สกล	พึ่งและ
ระดับการศึกษา	อุตสาหกรรมศาสตร์บัณฑิตทางเทคโนโลยีคอมพิวเตอร์-อุตสาหกรรม	
อาจารย์ที่ปรึกษา	เกษตร	ศิริสันติสัมฤทธิ์
ปีการศึกษา	2535	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ถูกออกแบบโดยการ นำเอาระบบไมโครคอนโทรลเลอร์ (microcontroller)มาออกแบบและพัฒนาใช้สำหรับการจัดเก็บทิศทางของช่องคู่สถานีและป้องกันความเสียหายของตัวสายอากาศจากลมพายุอัตตโนมัติโดยการนำเอาไมโครคอนโทรลเลอร์เป็นตัวประมวลผลควบคุมการทำงานทั้งหมด ซึ่งจะใช้แทนระบบเก่าที่ผู้ใช้ต้องควบคุมด้วยตัวเองตลอดเวลาซึ่งความสามารถของระบบนี้สามารถพัฒนาให้มากขึ้นด้วยคำสั่งทางซอฟแวร์

คุณสมบัติของงาน

โหมดการทำงานปกติ

1. สามารถควบคุมทิศทางของตัวสายอากาศผ่านทางkey boradโดยผู้ใช้ได้
2. สามารถเก็บข้อมูลทิศทางของคู่สถานีไว้ได้ 100 ช่องสถานี
3. สามารถใช้ข้อมูลของช่องสถานีที่โปรแกรมไว้ ควบคุมทิศทางได้ทันที
4. สามารถใช้งานทั้ง manaul และ autometic

โหมดการทำงานระบบป้องกัน

1. สามารถหันทิศทางelementของสายอากาศตามแรงลมอย่างอัตโนมัติ
2. สามารถวัดความเร็วและทิศทางของลมได้ทุกขณะ
3. ส่งสัญญาณเตือนalarmเมื่อมีลมพายุ

THESIS TITLE	Direction Antenna Controller		
STUDENT NAME	Nattapong	Pongpanich	34.162152
	Totsaphorn	Sathornvisit	34.162156
	Nontarat	Sirirat	34.162159
	Ratchasak	Ratathum	34.162167
	Sakol	Pounglahe	34.162173
LEVEL OF STUDY	Bachelor's Degree of Industrial Computor-Technology.		
ADVISOR	Kaset Sirisantisamrid		
ACADEMIC	1992		

ABSTRACT

This thesis is design by using Micro Controller System to design and develop for keeping the direction of channel of station and protecting the immage of antennna from wine strom automatically. By using the Micro Controller† to be the processer to control all working with will be used instead of the old system. Which that the user much control by themself all the time. The ability of this suustem can be deviloped further more by using the software other.

Charecteristic of Project

Normal Mode

1. It can be control the direction of the antenna by passing on the keyboard.
2. It can be keep the data of the direction of the antenna up to 40 channel.
3. It can be use the data of the programed channel and can order the antenna to move to the design the direction.
4. It can be set the reference direction.

Protect Mode

1. It can be move the element of the antenna to be direction according to the direction of the wind strom, for protecting the immage automatically.
2. It can be masure the speed and the direction of the wind by showing on the display channel.
3. It can be sent the warning sign (alarm) when the wind strom is occured.

สารบัญ

-FLOWCHART
-PROGRAM

กิตติกรรมประกาศ

ภาคผนวก

บรรณานุกรม

บทนำ

การติดต่อสื่อสารทางวิทยุในปัจจุบันมีความเจริญทางเทคโนโลยีขึ้นมาอย่างรวดเร็วได้มีการนำคอมพิวเตอร์เข้ามาใช้ในรูปแบบต่าง ๆ ในการติดต่อสื่อสาร โดยทั่วไปจะมีอยู่ 2 ลักษณะ คือ

1. การสื่อสารข้อมูลระหว่างกันโดยใช้สายเชื่อมต่อระหว่างกัน ซึ่งความสามารถในการส่งข้อมูลขึ้นอยู่กับคุณสมบัติของสายส่ง จึงไม่สะดวกในการบำรุงรักษาหรือตรวจสอบ

2. การสื่อสารข้อมูลโดยผ่านทางคลื่นวิทยุซึ่งสามารถทำการติดต่อกันได้ในระยะทางไกลหรือที่ต่าง ๆ ไม่สามารถเดินสายส่งเข้าไปถึงได้ ทำให้ไม่มีปัญหาในเรื่องสายส่ง

การสื่อสารโดยใช้คลื่นวิทยุ เช่น วิทยุสื่อสาร (วิทยุรับส่ง) จะต้องใช้เสาอากาศในการรับส่งคลื่นวิทยุในคลื่นความถี่ต่าง ๆ ในการติดต่อให้ได้ช่องคลื่นความถี่ที่แรง หรือช่องความถี่ที่ต้องการ จะต้องทำการหันเสาอากาศ (YAKI) ไปในทิศทางที่ต้องการ จึงต้องใช้มอเตอร์ (ROTOR) ในการหันและจากการใช้ ROTOR ในการควบคุม จะต้องมีตัวควบคุม (CONTROL UNIT) ที่ออกแบบโดยวงจรอิเลคทรอนิคส์ จากที่กล่าวมา

ปริญญานิพนธ์นี้ได้นำเอาไมโครคอนโทรลเลอร์ เข้ามาควบคุมแทนตัวควบคุมเดิมทำให้ มีประสิทธิภาพดีขึ้น จากการใช้ไมโครคอนโทรลเลอร์มาใช้ในการควบคุม จะมีความสามารถในการแสดงผล และเก็บข้อมูลในรูปของหน่วยความจำก็คือเราสามารถจดจำช่องความถี่ต่าง ๆ โดยการกำหนดองศาของสถานีที่เราต้องการติดต่อ โดยการบันทึกจากผู้ใช้ เก็บไว้ในหน่วยความจำ

วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาระบบควบคุมทิศทางการหมุนของ ROTOR สายอากาศ ซึ่งเป็น MOTOR ที่ใช้ไฟฟ้ากระแสกลับ
2. ศึกษาการประยุกต์ใช้ไมโครคอนโครลเลอร์
3. เพื่อศึกษาการควบคุมอัตโนมัติโดยใช้คอมพิวเตอร์
4. เพื่อนำเอาโครงงานนี้ไปใช้ประโยชน์ได้จริง
5. สามารถนำเอาความรู้ที่ได้จากปริญญานิพนธ์นี้นำไปแก้ปัญหากับงานอื่น ๆ ต่อไป

ขอบเขตของปริญญานิพนธ์

เนื้อหาของปริญญานิพนธ์ฉบับนี้จะมีเนื้อเรื่อง ซึ่งแบ่งตามบทต่าง ๆ ดังนี้

บทที่ 1 กล่าวถึงหลักการออกแบบการใช้งาน จะกล่าวถึงการออกแบบโดยรวมของเครื่องโดยจะแบ่งออกเป็นบล๊อกไดอะแกรม และการใช้งานในรูปแบบต่าง ๆ

บทที่ 2 กล่าวถึงหลักการออกแบบในภาค CONTROLLER ซึ่งถือเป็นหัวใจของระบบและ

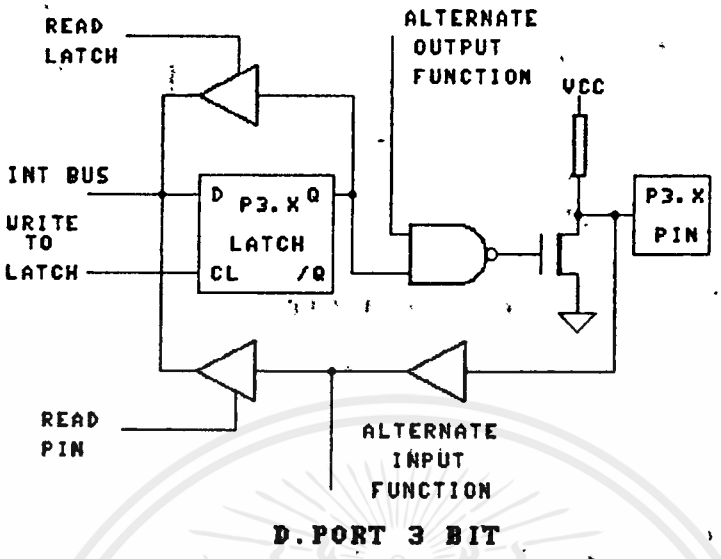
ส่วนประกอบที่สำคัญๆ

บทที่ 3 เป็นการอธิบายถึงการทำงานของภาค DRIVER ROTOR โดยจะกล่าวถึงพื้นฐานการควบคุม AC มอเตอร์ วงจร Interface กับภาค CONTROLLER และชิ้นส่วนประ-กอบของตัว ROTER

บทที่ 4 เป็นเนื้อหาของการออกแบบ WIND DIRECTION และ WIND SPEED โดยจะกล่าวถึงทั้ง machanic และ electronic รวมถึงข้อกำหนดต่าง ๆ

บทที่ 5 เป็นเนื้อหาการออกแบบ GRAPRIC LCD และ KEY BOARD

บทที่ 6 แสดงการทำงานของ PROGRAM และ SOURSE CODE



รูปที่ 1 8051 Bit Latch และ 1/0 Buffer

รูปที่ 1 แสดงไดอแกรมของ Bit Latch และ 1/0 Buffer ของ Port ทั้ง 4 Bit Latch (เป็น 1 Bit ใน SFR ของ Port) ก็คือ D flip-flop ซึ่งค่าจาก Internal Bus จะถูก Latch เมื่อมีสัญญาณclock ป้อนเข้ามาซึ่งก็คือสัญญาณ ' write to Latch ' จาก CPU Output ขา Q ของ flip-flop จะไปปรากฏที่ Internal Bus เมื่อมีสัญญาณ ' read Latch ' จาก CPU และ4 Level ที่ขา Portจะปรากฏที่ Internal Bus เมื่อมีสัญญาณ ' read pin ' จาก CPU มีบางคำสั่ง เช่นคำสั่งอ่าน Port จะถูกกระตุ้นด้วยสัญญาณ ' read Latch ' ซึ่งเป็นการนำค่า Output ที่ขา Q ให้มาปรากฏที่ Internal Bus นั้นเอง ส่วนคำสั่งอื่นๆ จะมีการกระตุ้นด้วยสัญญาณ ' read pin ' ดังได้แสดง ในรูปที่ 1A และ 1C Output Driver ของ Port 0 และ Port 2 สามารถ Switch ระหว่าง Internal ADDR และ ADDR/DATA Bus ได้โดยสัญญาณควบคุมภายในซึ่งจะใช้สำหรับการติดต่อกับหน่วยความจำภายนอกในขณะที่กำลังติดต่อกับหน่วยความจำภายนอก ค่า P2SFR ยังคงค่าเดิมไม่เปลี่ยนแปลง แต่ค่า P0SFRจะถูกเขียน ด้วยค่า '1'จากรูปที่ 1Dถ้า BitLatchของ P3มีค่าเป็น '1'แล้ว Levelที่ขา Output จะถูกควบคุมด้วยสัญญาณ ' Alternate output function ' Level ที่ขา P3x จริงจะปรากฏที่ขา Alternate input function Port 1, Port 2 และ Port 3 มี Internal pullup แต่ Port 0 มี Output แบบ

Open drain แต่ละ Bit 1/0 สามารถเป็น Inputหรือ Output ได้อย่างอิสระ (Port 0 และ Port 2อาจไม่สามารถใช้เป็น 1/0ทั่วไปได้เมื่อมีการต่อCPU กับหน่วยความจำภายนอกเพราะจะถูกใช้เป็น ADDR/DATA Bus) ในการใช้งาน Port เป็น Input Bit Latch ของ Port ต้องมีค่าเป็น '1' ซึ่งมันจะทำการ off

ลักษณะหน้าที่การทำงานของแต่ละบล็อกไดอะแกรม

-CPU (CONTROLLER) มีหน้าที่ควบคุมการทำงานทั้งหมดของวงจร และเป็นส่วนเก็บ หน่วยความจำ และ PROGRAM MONITER ซึ่งมีความสำคัญมาก ในส่วน CONTROLLER นี้จะเป็นตัวรับรู้สภาวะต่างๆ เพื่อสามารถปฏิบัติงานได้จริงตามความต้องการ ส่วนหลักการออกแบบและการทำงานจะกล่าวในบทที่ 2 ต่อไป

-WIND DIRECTION มีหน้าที่รับรู้สภาวะทิศทางของลม เพื่อนำไปเปรียบเทียบกับตำแหน่งของ ROTER ใน MODE 0 และเป็นตัวบอกทิศทางของลม และความแรงของลม ซึ่งจะอยู่ในส่วนของ WIND SPEED เพื่อนำไปใช้งานใน MODE 1 ในส่วนของ WIND DIRECTION และ WIND SPEED จะกล่าวถึงรายละเอียดต่างๆในบทที่ 4

-DRIVER ROTER มีหน้าที่ควบคุมเฟสทั้ง2ของ ROTER ในการทำงานตามทิศทางที่ ภาค CONTROLLER สั่ง และจะเป็นตัวบอกตำแหน่งของ ROTER ในสถานะปัจจุบัน โดยส่งกลับในรูปของสัญญาณ DIGITAL ไปที่ส่วน CONTROLLER รายละเอียดจะกล่าวในบทที่ 3

-GRAGHIC LCD มีหน้าที่แสดงผลการทำงานต่างๆ จาก KEYBOARD ,DRIVER , WIND DIRECTION โดยผ่าน CONTROLLER คุณสมบัติของ LCD จะกล่าวในบทที่ 5

-KEYBOARD มีหน้าที่เป็น INPUT จากผู้ใช้งาน ที่จะควบคุมการทำงานและเก็บข้อมูล ซึ่งจะมี FUNTION ต่างๆตามการใช้งานใน MODE ต่างๆ

จากที่กล่าวมาเป็นหน้าที่หลักการทำงานใน BLOCK ต่างๆ ซึ่งจะกล่าวอย่างคร่าวๆ ส่วนรายละเอียด การออกแบบ การทำงานหลักจะกล่าวถึงในบทต่อๆไป ต่อไปจะกล่าวถึง การใช้งาน และการทำงานของเครื่อง

การใช้งานของเครื่อง

เราสามารถแบ่งการทำงานออกเป็นส่วนสำคัญ 2 MODE การทำงานดังนี้

1. MODE การทำงานเกี่ยวกับระบบควบคุมทิศทางของผู้ใช้ ซึ่งจะอาศัยการผ่านงานทาง MICROCONTROLLER เพื่อควบคุมส่วนของวงจร DRIVE ซึ่งใน MODE นี้จะรวมการทำงานในระบบข้อมูล หน่วยความของทิศทาง ซึ่งผู้ใช้สามารถ PROGRAM ได้

2. MODE การทำงานเกี่ยวกับระบบการป้องกัน ความเสียหายของตัวสายอากาศอันเนื่องจากลมพายุใน MODE นี้จะเป็นการติดต่อโดยตรงระหว่างตัวรับรู้สภาวะทิศทาง (WIND DIRECTION) และตัวรับรู้ความเร็วลม (WIND SPEED) กับ MICROCONTROLLER โดยตรง ซึ่งผู้ใช้สามารถสั่งให้เครื่องเข้าสู่ MODE นี้ได้โดยตรง

การติดตั้งและการใช้งาน

ROTOR ที่นำมาใช้งาน จะต้องเป็น ROTOR แบบ TWIN DIRECTION โดยที่ ต้องSET ให้จุดสิ้นสุดของการหมุนของแต่ละรอบ (0 และ 360 องศา)ไว้ที่ทิศเหนือ หรือทิศที่ต้องการอ้างอิงเสมอ เพื่อเป็นการสร้างทิศทางอ้างอิง

การควบคุมทิศทางที่ต้องการสามารถส่งผ่านทาง KEYBOARD เป็นจำนวนองศาตามต้องการ

หรือสามารถสั่งให้ ROTOR หมุนอย่างต่อเนื่อง ซึ่งควบคุมโดยผู้ใช้โดยตรง (MANUAL)จากปุ่ม D+ , D- เพื่อหาทิศทางที่ดีที่สุดในการรับคลื่น จากนั้นผู้ใช้สามารถ MERORY ช่องสถานีที่ต้องการได้โดยใช้ปุ่ม MW โดยมีรูปแบบการดำเนินคำสั่งดังนี้

เริ่มแรกผู้ใช้จะต้อง SCAN สถานีที่ต้องการ โดยกดปุ่ม D+ , D- ROTOR จะหมุนไปในทิศทางที่เราต้องการ ในการเปรียบเทียบสถานีที่ต้องการ เราก็ดูจาก เครื่องวิทยุรับส่งเช่น เราต้องการติดต่อกับสถานีที่ต้องการ เริ่มแรกเราจะหันสายอากาศโดยกดปุ่ม D+ , D- สายอากาศจะหันสู่เป้าหมายและจะมีการแสดงองศาอยู่ตลอดเวลาที่ LCD DISPLAY เมื่อเราสามารถหันสายอากาศจนสามารถติดต่อกับสถานี ตามที่ต้องการเป็นทิศทางจำนวนกี่องศา

***ในสภาวะเริ่มแรกก่อนที่มีการ MEMORY จะต้อง SCAN สถานีก่อนทุกครั้ง

การสแกน (SCANING)

การสแกนแบ่งออกเป็น 2 ลักษณะ คือ 1. สแกนตามความแรงของช่องสถานี 2. สแกนตามความต้องการของผู้ใช้เพื่อจะติดต่อกับสถานีนั้น และนำไป MEMORY

การสแกนโดยทั่วไปจะสแกนได้โดยผู้ใช้ และจะหยุดสแกนเมื่อพบช่องสถานีที่ต้องการ

วิธีการสแกน (BAND SCAN)

1. ทำการเลือกสถานีที่ต้องการติดต่อ โดยกดปุ่ม D+ , D- โดย CHECK ได้จากวิทยุรับส่งที่ติดตั้งประจำสถานีของตน

2. เมื่อทำการปรับ ROTOR จะหันไปทางทิศที่สามารถรับสัญญาณ (ความถี่) จากสถานีที่ต้องการ DISPLAY จะแสดงค่าองศาที่ได้จากการหันสายอากาศตามที่ได้ติดต่อกับสถานีนั้น

3. ค่าองศาที่ได้จะแสดงและถูกเก็บมาไว้ในส่วนของ NUMBER OF DIRECTION ของ LCD

การบันทึกหน่วยความจำ (MEMORY ENTRY)

หน่วยความจำ (MEMORY) ของเครื่องสามารถบันทึกช่องสถานี ได้ทั้งหมด 00-99 ช่อง ในขณะที่ผู้ใช้ต้องการบันทึกช่องสถานีใหม่ทุกๆครั้งจะต้องทำการ SCAN ก่อนทุกครั้ง ในบางครั้งผู้ใช้เครื่องมีความต้องการจะลบโปรแกรมข้อมูลที่เคยบันทึกไว้จะต้องใส่ข้อมูลใหม่เข้าไป ลักษณะก็คล้ายๆกันกับการบันทึกช่องสถานี

วิธีการบันทึกช่องสถานี

1. เมื่อได้ค่าองศาตามวิธีการ SCAN ซึ่งแสดงในส่วน NUMBER OF DIRECTION ของ LCD เมื่อเราต้องการจะบันทึกช่องสถานีนั้น เราต้องทำการ SCAN ก่อนเสมอ

2. กดปุ่ม MW เพื่อจะทำการเลือกช่องสถานีโดย DISPLAY จะแสดงส่วน MEMORY เพื่อรอรับการ KEY จากผู้ใช้ ซึ่งจะต้อง KEY ช่องสถานีที่ต้องการจะเก็บ โดยมีจำนวน 100 ช่อง

3. เมื่อทำการป้อนค่าช่องสถานีที่ต้องการแล้ว เช่น ต้องการบันทึกช่องสถานีในช่องที่ 5 ให้กดปุ่ม 5 DISPLAY จะแสดง Memory 005

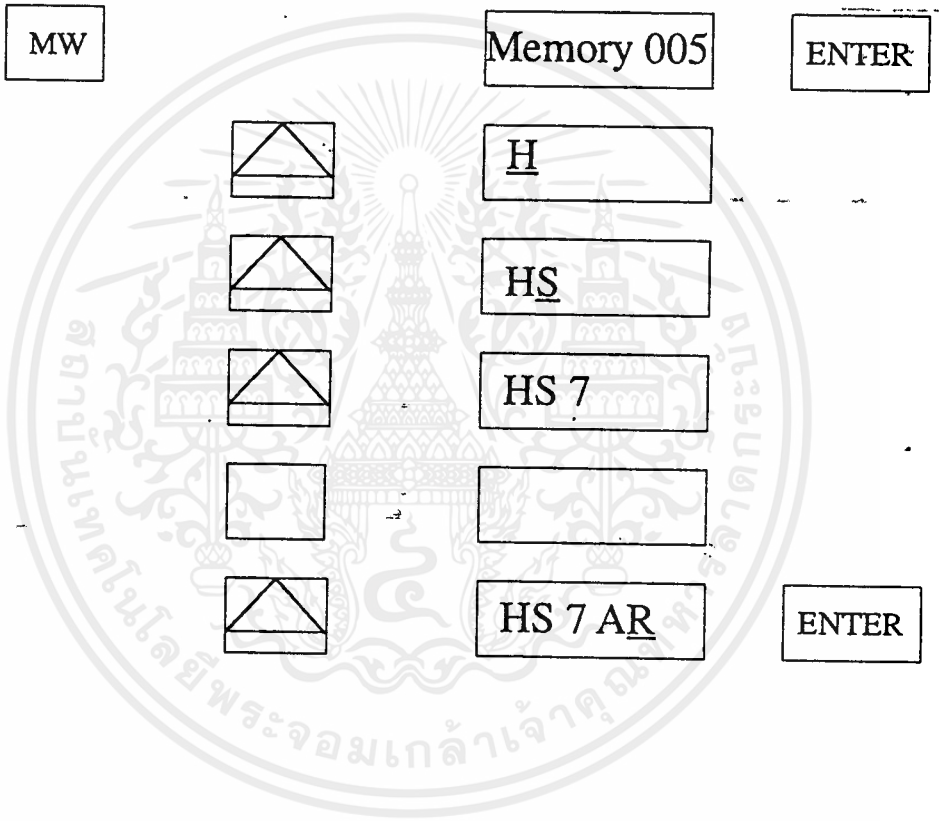
4. ช่องสถานีจะปรากฏที่ DISPLAY ในส่วนของ STATION โดยเครื่องจะเรียกข้อมูลช่องนี้จากหน่วยความจำขึ้นมาแสดง

5. ค่าที่ถูกเรียกจากช่องนี้ถ้ามีการบันทึกอยู่แล้วและต้องการจะบันทึกทับลงไปให้กดปุ่ม ENTER แต่ถ้าไม่ ให้กดปุ่ม ESC เครื่องจะกลับสู่สภาวะ MAIN

6. เครื่องจะรอรับการป้อนชื่อสถานีที่ต้องการติดต่อ ที่ช่อง STATION ข้อมูลโดยจะบอกช่องสถานีและตามด้วยเคอร์เซอร์ ดังนี้ _ เคอร์เซอร์จะรอการป้อน

7. ทำการตั้งชื่อสถานีที่ต้องการ โดยเลือกตัวอักษร (ALPHABET) กับตัวเลข (NUMERIC) โดยใช้ปุ่ม ARROW KEY และ ชุด NUMBER KEY ตามลำดับ

8. เมื่อใส่ชื่อสถานีตามต้องการแล้วกดปุ่ม ENTER เครื่องจะกลับสู่สภาวะ MAIN เช่น ต้องการป้อนชื่อสถานี ชื่อ HS 7 AR ในส่วนของ STATION



การเรียกใช้งาน

1. กดปุ่ม MR DISPLAY จะแสดง เพื่อรอการเรียกช่องสถานีจากผู้ใช้
2. ทำการป้อนช่องสถานีที่ต้องการติดต่อ ซึ่งมีทั้งหมด 100 ช่อง
3. กดปุ่ม MR และตามด้วยปุ่ม ENTER เครื่องจะรับรู้การติดต่อจากผู้ใช้
4. เครื่องจะ DRIVE ROTER ให้หันไปในทิศทางองศาที่บันทึกไว้ในช่องสถานีที่ถูกเรียกใช้

บทที่ 2

หลักการออกแบบภาค controller

จากที่กล่าวมาในบทนำ วิทยานิพนธ์นี้ได้ใช้ microcontroller ตระกูล MCS-51 เป็น CPU ของระบบ เป็นหลักในการออกแบบภาค controller นี้

โดยที่ส่วนของ CONTROLLER ถูกสร้างขึ้นเพื่อใช้ในการควบคุมให้ตรงกับหน้าที่ๆ ต้องการ หลักการของ CPU ในตระกูล MCS-51 คือเป็น microcontroller โครงสร้างทางกายภาพดังนี้

ลักษณะของ board

CPU - 8031

MEMORY - สามารถใส่หน่วยความจำได้สูงสุด 96KB คือ EPROM 64 KB และ RAM 32 KB

I/O - 3 x 8 บิท INPUT/OUTPUT (8255)
- 1 x 8 บิท INPUT/OUTPUT (PORT 1 8031)
- 1 serial port (RS232)

ในการเลือกใช้หน่วยความจำ นั้น จะใช้ EPROM 27512 เป็น PROGRAM MONITER และ RAM 62256 เป็น DATA MEMORY โดยจะมีการ interface กับภาคต่างๆ ดังนี้ คือ

-ภาค DRIVER ROTOR จะ interface แบบ DIGITAL TO ANALOG และแบบ ANALOG TO DIGITAL โดยจะใช้ 8255 PROGRAMMABLE I/O PORT เป็นตัว interface ซึ่ง 8255 จะเป็นตัวขยาย PORT จาก 8031 ให้มีการใช้ PORT ได้มากขึ้น

-ภาค แสดงผล DISPLAY GRAPHIC LCD 12864 ถูกออกแบบให้ต่อเข้ากับ PORT ของ 8255

-KEYBOARD จะถูกออกแบบให้ต่อกับ PORT ของ 8255 และจะมีการต่อเข้ากับ INTERRUPT

-ภาค WINDDIRECTIONจะออกแบบให้ต่อเข้ากับ PORT 1ของ 8255

-ภาค WIND SPEED จะออกแบบให้ต่อเข้ากับ TIMER/COUNTER 0

จากการที่ภาค CONTROLLER จะต้องเป็นส่วนสำคัญในการควบคุม ภาคการทำงานต่างๆ จึงมีความจำเป็นที่จะต้องเข้าใจ หลักการทำงาน ของ MCS-51 อย่างละเอียด

สิ่งสำคัญที่จะต้องพิจารณาในการออกแบบ ภาค CONTROLLER มีดังนี้

- I/O PORT (8031)
- EXPANDING I/O PORT (8255)

- TIMER/COUNER
- EXTERNAL MEMORY (MEMORY MAP I/O)
- INTERRUPT

MICROCONTROLLER DESIGN

การออกแบบ MICROCONTROLLER ตามรูป 1 เป็นวงจรที่ออกแบบโดยสมบูรณ์ในส่วนของหน่วยความจำ EXTERNAL MEMORY (หน่วยความจำภายนอก)

หน่วยความจำภายนอกจะใช้ port 0 เป็น DATA และเป็น ADDRESS LOW-ORDER และ PORT2 เป็น HIGH-ORDER ADDRESS DATA และ LOW-ORDER ADDRESS ใน PORT 0 จะใชเวลาในการ MULTIPLEX และใช้ IC373 เป็นตัว ADDRESS LATCH โดยต่อเข้ากับ PORT 0 เพื่อจัดเก็บ LOW ADDRESS BYTE โดยใช้สัญญาณ ALE ในการเลือกโดยรับจาก 8031

PORT 0 จะเป็น DATA BUS แบบ BI-DIRECTION ภายใต้ MACHINE CYCLE READ และ WRITE

RAM และ ROM จะใช้ ADDRESS ที่แตกต่างกันโดยการควบคุมจาก 8031 โดยสัญญาณ PSEN สำหรับ ROM และ WR และ RD สำหรับ RAM ซึ่งจะทำให้มีตัวใดตัวหนึ่ง สามารถมีหน่วยความจำได้ถึง 64 KBYTES

การนำ Microprosser มาต่อใช้งาน ส่วนที่มีความสำคัญมากส่วนหนึ่งก็คือ Portซึ่ง Portเป็นอุปกรณ์สำคัญที่ CPU จะใช้ติดต่อกับโลกภายนอก ซึ่งมีทั้ง คีย์บอร์ด ,อุปกรณ์เซ็นนิ่งต่างๆ และอุปกรณ์เอาท์พุทต่างๆ เช่น จอแสดงผล ,เครื่องพิมพ์ การใช้งาน Portสำหรับ CPU ที่เป็น Microprocesser ส่วนใหญ่จะต้องต่อเพิ่มขึ้นเอง แต่สำหรับ CPU ที่เป็น Microcontroller จะมี Port อยู่ภายใน ซึ่งสะดวกมากกับระบบขนาดเล็กที่ใช้ Port ไม่มากนัก และก็ยังสามารถต่อขยายได้ โดยเฉพาะอย่างยิ่ง 8051 Microcontroller ที่แต่ละ Bit สามารถใช้เป็น อินพุท หรือ เอาท์พุทได้ และบาง Bitในบาง Port ยังนำมาใช้ทำ Function พิเศษได้อีก

8051 มี Port ทั้งหมด 4 Port ซึ่งเป็นแบบ 2 ทิศทาง(bidirectional) แต่ละBit ของ Port จะประกอบไปด้วย วงจร Latch (ค่าของ Port ทั้ง 4 จะอยู่ในSpecial Function Register P0-P3) , Output Driver และ Input Buffer Output Driver ของ Port 0 และ Port 2 และ Input Buffer ของPort 0 จะถูกใช้ในการติดต่อกับหน่วยความจำภายนอก (External Memory) การใช้งานใน ลักษณะนี้ Port 0 จะเป็น Output ให้ค่า Address Byte ต่ำ (A0-A7) กับหน่วย ความจำภายนอก และในช่วงเวลาถัดไป Port 0 จะถูกใช้เป็น Data Bus สำหรับการเขียนหรืออ่านข้อมูล (D0-D7) โดยจะเป็น Output เมื่อทำการเขียน และเป็น Inputเมื่อทำการอ่านหน่วยความจำภายนอก (การทำงานของ Port 0ในลักษณะนี้จะเป็นแบบ Multiplex Address/Data) Port 2 จะทำหน้าที่เป็น Output ให้ค่า Address Byte สูง (A8-A15) กับหน่วยความจำภายนอก ซึ่งจะทำให้อ้าง Address ได้ครบทั้ง 16 Bit ส่วนการใช้งาน Port 2 ในลักษณะอื่นค่าที่ขาของ Port 2 จะปรากฏตามค่าที่อยู่ใน P2 SFR

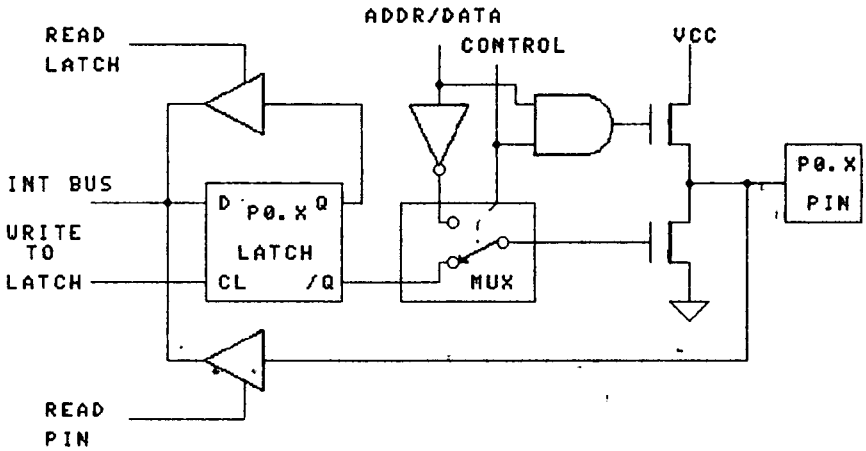
ขาทั้งหมดของ Port 3 และ 2 ขาของ Port 1 (ใน 8051) ถูกจัดไว้สำหรับหน้าที่พิเศษ แสดงในตารางที่ 1

Alternate Function จะสามารถใช้งานได้ก็ต่อเมื่อได้ทำการ Set Bit Latch ใน Port SFR ให้มีค่าเป็น '1' เท่านั้น ส่วนการใช้งานในลักษณะอื่นๆ ให้ทำการ Clear ขา Port นั้นให้เป็น '0'

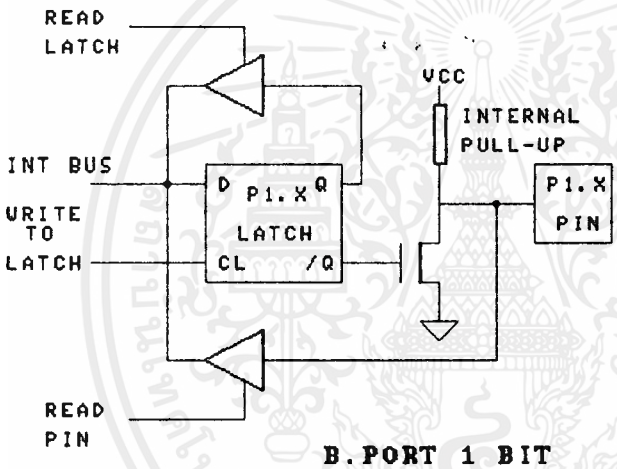
Port Pin	Alternate Function
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 Capture/Reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

ตารางที่ 1 Function พิเศษของ Port 3 และ Port 1

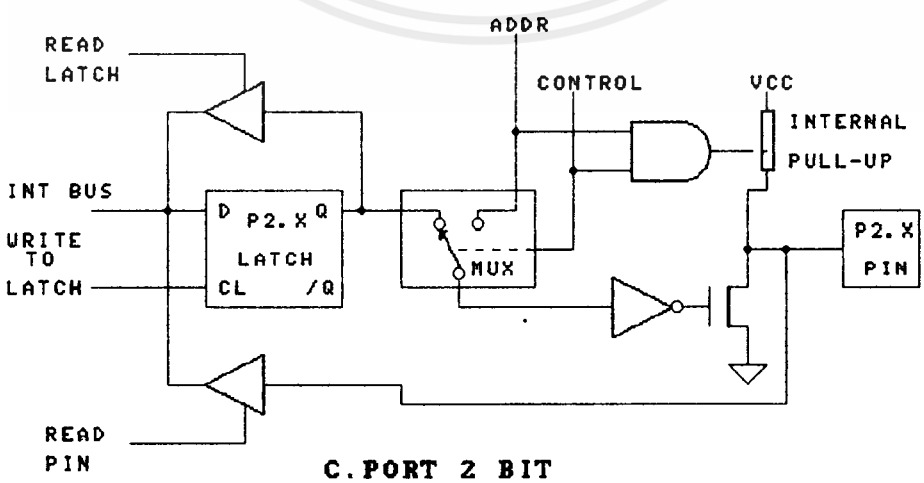
ขา P1.1 และ P1.0 จะถูกใช้ใน Alternate Function เมื่อใช้ 8052



A. PORT 0 BIT



B. PORT 1 BIT



C. PORT 2 BIT

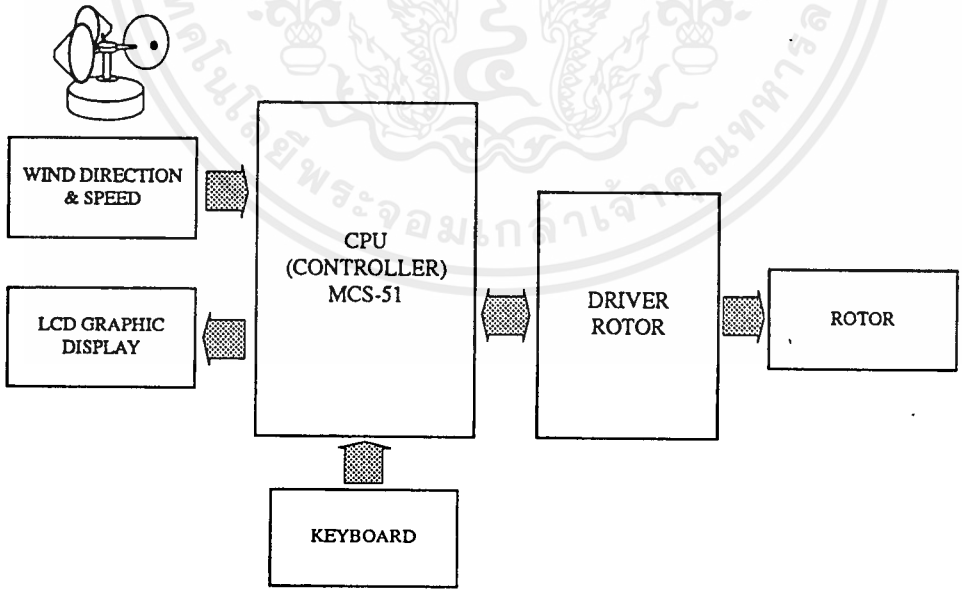
บทที่ 1
หลักการออกแบบและการใช้งาน

การควบคุมสายอากาศแบบทิศทาง เดิมนั้น เป็นการควบคุมทาง ANALOG ซึ่งผู้ใช้ต้องควบคุมทิศทางของสายอากาศอยู่ตลอดเวลา ทำให้เกิดความยุ่งยากเมื่อต้องการเปลี่ยนแปลงทิศทางในการติดต่อกับสถานีแต่ละสถานีที่แตกต่างกัน อีกทั้งสายอากาศนั้นจะเกิดความเสียหายหากทิศทางของสายอากาศมีทิศทางขวางกับทิศทางของลมพายุ

จากเทคโนโลยีในปัจจุบัน ได้มีการนำระบบคอมพิวเตอร์มาใช้ในการควบคุมอัตโนมัติกันอย่างกว้างขวาง ซึ่งทำให้ลดความยุ่งยากและซับซ้อนในระบบควบคุม อีกทั้งเรายังสามารถบันทึกการทำงานและประมวลผลข้อมูลอย่างมีประสิทธิภาพ

ซึ่งจากคุณสมบัตินี้เอง เราจึงนำเอาไมโครคอนโทรลเลอร์ (MCS-51) และจากคุณสมบัติของ MCS-51 นั้นจะประกอบด้วย พอร์ท I/O ถึง 4 พอร์ท มีหน่วยความจำภายในให้ใช้และมีความสามารถในการทำงานชุดคำสั่งระดับ BIT ดังนั้นจากคุณสมบัตินี้ MCS-51 จึงเหมาะสำหรับใช้ในการควบคุมโดยเฉพาะ เราจึงนำมาใช้ในการควบคุมทิศทางการหมุนของสายอากาศ

จากความสามารถและความต้องการที่ได้กล่าวมาแล้วนั้น เราจึงนำขอบเขตการใช้งานของเครื่องที่ควรจะออกแบบ โดยจะมีหลักการออกแบบ และสามารถแบ่งส่วนการทำงานดังแสดงตามบล็อกไดอะแกรมข้างล่างได้ดังนี้



BLOCK DIAGRAM:DIRECTION ANTENNA CONTROLING WITH MICROCONTROLLER SYSTEM

output driver FET แล้วสำหรับ Port 1 , Port 2 และ Port 3 ที่ขา Port จะถูกpulled high ด้วย Internal pullup แต่มันสามารถ pulled low ด้วยแหล่งจ่ายภายนอก (External source)Port 0ต่างจาก Port 1 , Port 2และ Port 3ตรงที่ไม่มี Internal pullup FETใน Output Driver ของ P0 (ดูรูปที่ 1A) จะถูกใช้เมื่อขา Port นั้นมีค่าเป็น '1' ขณะที่กำลังติดต่อกับหน่วยความจำภายนอกเท่านั้น ส่วนในกรณีอื่นๆ pullup FET จะอยู่ในสภาวะ off ดังนั้น Port P0 จะถูกใช้เป็น Port Output ในลักษณะ open drain การเขียนค่า '1' ให้กับ BitLatchทำให้ FETทั้ง 2 ตัวทางด้าน Output off จะมีผลทำให้ขานั้นอยู่ในสภาวะลอยตัว (float) ในสภาวะเช่นนี้ มันสามารถใช้เป็น Input แบบ high impedance ได้และเพราะว่า Port 1 , Port 2และ Port 3มีInternalpullup ค่าคงที่ การต่อวงจรแบบนี้อาจเรียกได้ว่า เป็น Port แบบ quasi-bidirectional เมื่อใช้งานเป็น Input Port เหล่านี้จะต้องถูก pull high และจะเปรียบเสมือนเป็น source current เมื่อมีการ pull low จากภายนอก Port latch ทั้งหมดใน 8051 จะถูกเขียนด้วยค่า '1' ทุก Bit เมื่อ CPU ถูก Reset ถ้าค่า 0 ถูกเขียนไปยัง Port Latch แล้ว Port Latch นั้นสามารถเปลี่ยนเป็นInput ได้โดยการเขียนค่า '1' ให้กับมัน

การต่อใช้งาน

การเชื่อมต่อกับอุปกรณ์ภายนอก เป็นสิ่งที่จะต้องพิจารณาให้รอบคอบ เพราะถ้าต่อใช้งานมากเกินไป อาจทำให้เกิดการ loading ได้ ซึ่งจะทำให้เสียเสถียรภาพของระบบได้สำหรับ Port ของ 8051 Output Buffer Port 1 , Port 2 และ port 3 สามารถขับ Input ของ ไอซี TTL แบบ LS ได้ 4 Input Port ของ8051 HMOS version สามารถ ขับโหลดในสภาวะปกติกับ ไอซี TTL หรือ วงจร NMOS ได้ ทั้งHMOS และ CHMOS version ขาของ Port จัดวงจรด้าน Output เป็นแบบopen-collector และ open-drain ในการใช้งานโหมด Bus ภายนอก (ExternalBusMode) Output Buffer ของ Port 0 แต่ละขาสามารถขับ Input ของ ไอซี TTL แบบ LS ได้ 8 Input แต่จำเป็น จะต้องต่อ resistor เพื่อ pullup ภายนอก

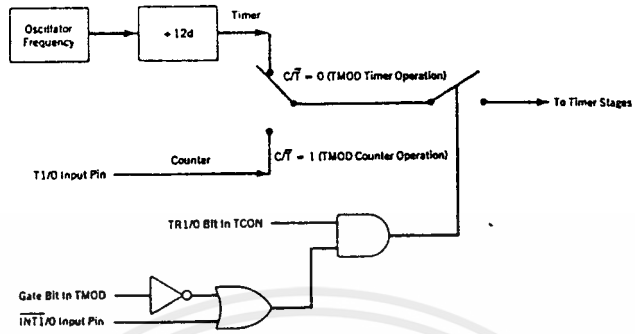
8051 TIMER/COUNTER

การทำงานแต่ละคำสั่งของCPUสามารถทราบเวลาที่แน่นอนได้ แต่ในการทำงานจริงๆ จะมีเงื่อนไงมากมายที่ CPU จะต้องเปลี่ยนแปลงการทำงาน เช่น ถ้าใช้ CPU ทำงาน อย่างหนึ่งในระยะเวลาที่แน่นอน ในขณะที่ต้องทำงานอย่างอื่นด้วย การเขียนโปรแกรมกับงานแบบนี้จะยากมาก เช่นการกำหนดอัตราการรับ/ส่ง ข้อมูล แบบอนุกรม(baud rate) , การสร้างสัญญาณฐานเวลา หรือการนับจำนวนสิ่งของในสายงานการผลิต

ในระบบการใช้งานจริง อาจใช้ support chip ประเภทtimer/counter เพื่อการทำงานแบบนี้ และเป็นการแบ่งเบาภาระของ CPU แต่เมื่อจำนวนชิพเพิ่มขึ้น ระบบก็จะใหญ่ตามไปด้วย วิธีที่ดีที่สุดก็คือการใช้ CPU ประเภทที่มี timer/counter อยู่ภายใน(on chip timer/counter) 8051 เป็น CPU ในตระกูล MCS-51 ตัวหนึ่งที่มี timer/counter อยู่ภายในนอกจากนี้ยังมี การส่งข้อมูลแบบอนุกรม และการทำงานในระดับบิทรวมอยู่ด้วย

8051 มีรีจีสเตอร์ที่เป็นทั้ง timer และ counter อยู่ภายใน 2 ตัว คือ timer0 และ timer1 (8032/8052 มีเพิ่มอีก 1 ตัว คือ timer2) รีจีสเตอร์นี้มีขนาดตัวละ 16 บิท และสามารถแยกเป็นขนาด 8 บิทได้เป็น TH0, TL0, TH1 และ TL1 ทำหน้าที่ในการเก็บค่าของ timer หรือ counter และยังสามารถเคลื่อนย้ายข้อมูล , กระทำทางคณิตศาสตร์ และ ลอจิก กับรีจีสเตอร์อื่นๆ ได้ TIMER0 และ TIMER1 จะทำงานอิสระ คือตัวไหนจะเป็น timer หรือ counter ก็ได้ แต่ในเวลาหนึ่งจะเป็นได้เพียงอย่างเดียว

FIGURE 2.11 Timer/Counter Control Logic



ไดอะแกรม รูปที่ 1

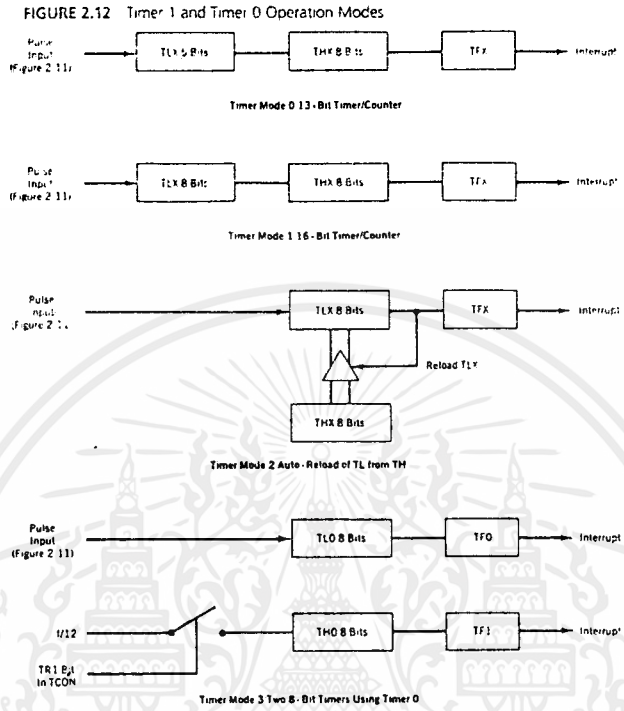
TIMER ทำหน้าที่ในการนับสัญญาณนาฬิกาภายใน CPU (ดูไดอะแกรมรูปที่ 1) สัญญาณนาฬิกาซึ่งมีความถี่แน่นอนจะถูกหารด้วย 12 แล้วให้ timer นับ เช่น ถ้า CPU ใช้ความถี่ 11.0592 MHz timer จะนับที่ความถี่ 11.0592MHz/12 = 921.6 KHz timerจึงนับทุกๆ 1/921.6KHz = 1.085 uSEC timer จะนับขึ้นแบบ binary เท่านั้น (binary up counter) การนับของ timer จะนับขึ้นเรื่อยๆ จนกระทั่งถึงค่าสูงสุด แล้วลงเป็น 0 (FF -> 0 หรือ 0->FF) แล้วเกิด interrupt ภายในตัว CPU ได้ (จะต้อง enable interrupt ด้วย) และตัวแปรที่สำคัญ ที่ทำให้ช่วงเวลาในการ interrupt ต่างกันคือ การกำหนดค่าเริ่มต้นให้กับ timer รีจีสเตอร์ (TH0, TL0, TH1 หรือ TL1) เช่นถ้ากำหนดค่าให้กับ timer รีจีสเตอร์สูง เวลาที่จะนับไปถึง 0 ก็เร็วขึ้น COUNTER ทำหน้าที่ในการนับสัญญาณจาก Input จากขา T0 หรือ T1 (ขา P3.4 และขา P3.5 ตามลำดับ) โดยสัญญาณจากขา Input นี้อาจจะมีคาบเวลาที่ไม่แน่นอนสัญญาณที่ขา T0 ใช้ counter 0 นับ สัญญาณที่ขา t1 ใช้ counter 1 นับเท่านั้น การนับของ counter ก็เหมือนกับ timer คือนับขึ้นแบบ binaryเท่านั้น และค่าเริ่มต้นในการนับ ก็อยู่ใน timer รีจีสเตอร์เช่นกัน มีข้อจำกัดของความถี่สูงสุดที่ป้อนให้กับขา T0 หรือ T1
คือ จะต้องไม่มากกว่าความถี่ของระบบหารด้วย 24 เช่นถ้าระบบใช้ความถี่ 11.0592 MHz ความถี่สูงสุดที่ป้อนให้กับขา T0 หรือ T1 ได้คือ 11.0592MHz/24 = 460.8KHz ความแตกต่างระหว่าง timer กับ counter คือคาบเวลาของสัญญาณ และแหล่งที่มาของสัญญาณ ที่ใช้ในการนับ และก่อนการใช้งาน timer หรือ counter จะต้องกำหนดการทำงานของมันเสียก่อน โดยมีรีจีสเตอร์ที่กำหนดการทำงานและการ interrupt ของtimer และ counter คือ

TMOD Timer/Counter Mode Register

TCON Timer/counter Control Register

IE Interrupt Enable Register

IP Interrupt Priority Register



ไดอะแกรมรูปที่ 2

การใช้งาน Timer

timer ภายใน 8051 ใช้รีจีสเตอร์ขนาด 16 บิท แต่ในการใช้ อาจจะใช้แบบ 8 บิท13 บิท หรือ 16บิท ก็ได้ (ดูไดอะแกรมรูปที่ 2) เพื่อให้ขั้นในการกำหนดการทำงานของtimer ง่ายและชัดเจนขึ้น จึงขอกล่าวเป็นขั้นตอนดังนี้

1. เลือกแบบการทำงานของ timer จากไดอะแกรมรูปที่ 2

2. จาก โหมดของ timer เลือกค่า control code จากตารางรูปที่ 3 ถ้า timer ถูกควบคุมการทำงาน/หยุดด้วยโปรแกรมภายในCPU ให้เลือกค่าในช่อง internal control และถ้าควบคุมการทำงานหยุด ด้วยขา INT0/1 จากภายนอก CPU ให้เลือกค่าในช่อง external control ค่าที่ได้จะถูกกำหนดให้กับรีจีสเตอร์ TMOD

MODE	TIMER 0 FUNCTION	TMOD	
		INTERNAL CONTROL	EXTERNAL CONTROL
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

MODE	TIMER 1 FUNCTION	TMOD	
		INTERNAL CONTROL	EXTERNAL CONTROL
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

ตารางรูปที่ 3

MODE	COUNTER 0 FUNCTION	TMOD	
		INTERNAL CONTROL	EXTERNAL CONTROL
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	one 8-bit Counter	07H	0FH

MODE	COUNTER 1 FUNCTION	TMOD	
		INTERNAL CONTROL	EXTERNAL CONTROL
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	-	-

ตารางรูปที่ 4

รีจีสเตอร์ TMOD นี้ใช้กับ timer 0 และ timer 1 ด้วย รายละเอียดของแต่ละบิทของ รีจีสเตอร์ TMOD มีดังนี้

TMOD : TIMER/COUNTER MODECONTROL REGISTER .
NOT BIT ADDRESSABLE :

GATE จะควบคุมให้ timer ทำงานหรือหยุด (ให้ดูไดอะแกรมรูปที่ 1ประกอบ)จะเห็นว่าการควบคุมทำได้ 2 แบบคือ การทำให้ บิท GATE=0 เป็นการควบคุม CPU โดยโปรแกรม หรือ การควบคุมจากภายนอกด้ายขา INT0/1 = 1 ในขณะที่ บิท GATE = 1 การควบคุมทั้ง 2 แบบจะมีผลเมื่อบิท TR0/1 ในรีจีสเตอร์ TCON เป็น 1ด้วย

C/T เลือกเป็น counter หรือ timer

M1,M0 โหมดการทำงานของ timer หรือ counter

3. กำหนดการ interrupt ในรีจีสเตอร์ IE โดยถ้าให้มีการ interruptบิท EA (IE.7) จะต้องเป็น 1 เสมอ และ บิทอื่นๆ จะเป็น 1 เสมอ เมื่อ ต้องการให้ interrupt แต่ละตัวไป

IE : INTERRUPT ENABLE REGISTER.
BIT ADDRESSABLE:

EA (IE.7) disable all interrupt ถ้าบิทนี้เป็น 0 และ enable interrupt ได้เมื่อบิทนี้เป็น 1 และบิท อื่นๆ (ตามข้างล่าง)เป็น 1 ด้วย

-- (IE.6) ไม่ใช้

ET2 (IE.5) TIMER2 overflow enable/disable interrupt (8032 /8052 เท่านั้น)

ES (IE.4) .....

ET1 (IE.3) TIMER1overflow enable/disable interrupt

EX1 (IE.2) ....

ET0 (IE.1) TIMER0 overflow enable/disable interrupt

EX0 (IE.0) ....

หมายเหตุ บทที่ไม่เกี่ยวกับ timer/counter จะไม่กล่าวถึง

4.ถ้าใช้ timer/counterมากกว่า 1ตัว จะต้องกำหนด ความสำคัญของการinterruptในรีจีสเตอร์ IPด้วย ถ้าเกิดการ interrupt พร้อมกัน ตัวที่มีความสำคัญ สูงกว่า จะได้การบริการก่อน ปกติแล้ว TIMER 0 จะมีความสำคัญกว่า TIMER 1 แต่ก็สามารถกำหนดให้TIMER 1 มีความสำคัญสูงกว่า TIMER 0 ได้โดยให้ บิท IP.3 = 1 และ IP.1 = 0

IP : INTERRUPT PRIORITY REGISTER.

BIT ADDRESSABLE:

(IP.7) ไม่ใช้

(IP.6) ไม่ใช้

PT2 (IP.5) TIMER2 priority level (8032/8052 เท่านั้น)

PS (IP.4) ....

PT1 (IP.3) TIMER1 priority level

PX1 (IP.2) ....

PTO (IP.1) TIMER0 priority level

PX0 (IP.0) ....

5.กำหนดค่าเริ่มต้นให้กับ timer/counter(TH0 , TL0 , TH1หรือ TL1)

6. เซ็ทบิท TCON.4 (หรือ TCON.6) เพื่อให้ TIMER/COUNTER เริ่มทำงาน

TCON : TIMER/COUNTER CONTROL REGISTER.

BIT ADDRESSABLE.

TF1 (TCON.7) เป็น 1 เมื่อเกิดการ overflow จาก TIMER1 และจะกลับเป็น 0 เมื่อ CPU ทำโปรแกรมใน interrupt service routine

TR1 (TCON.6) เซ็ทให้เป็น 1 เพื่อให้ TIMER1 ทำงาน

TF0 (TCON.5) เป็น 1 เมื่อเกิดการ overflow จาก TIMER0 และจะกลับเป็น 0 เมื่อ CPU ทำโปรแกรมใน interrupt service routine

TR0 (TCON.4) เซ็ทให้เป็น 1 เพื่อให้ TIMER0 ทำงาน

IE1 (TCON.3) ....

IT1 (TCOM.2) ....

IE0 (TCON.1) ....

IT0 (TCON.0) ....

การใช้งาน Counter

counter เป็น รีจีสเตอร์ตัวเดียวกับ timer แต่เรียกชื่อต่างกันตามลักษณะการใช้งาน ก่อนการใช้งานจึงต้องกำหนดการทำงาน เช่นเดียวกับ timer (ดูการกำหนดการทำ งานจาก timer) แต่ให้เลือกค่า control code จากตารางรูปที่ 4 แทน

EXPANDING I/O PORT (การขยาย PORT I/O)

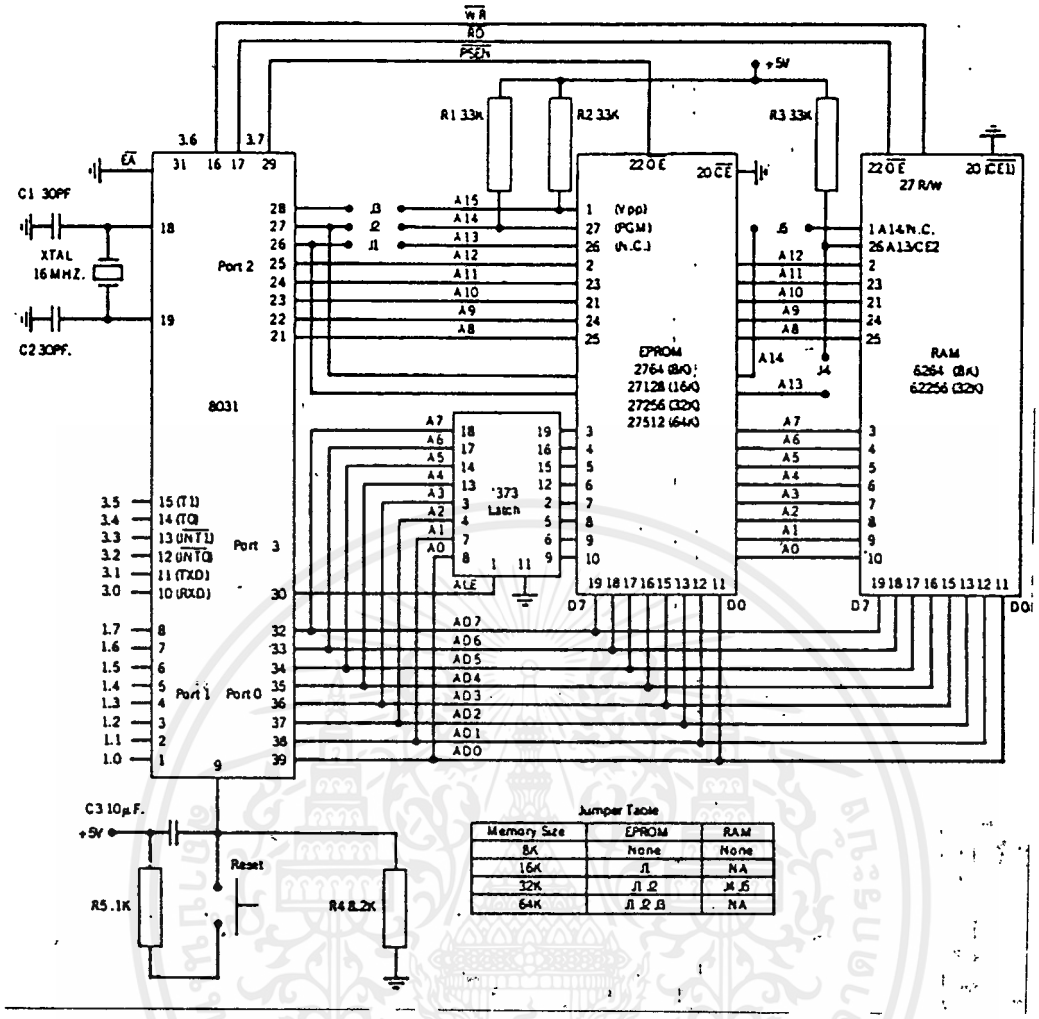
8031 จะมี PORT 0 และ PORT 3 ในการนำไปใช้ในการควบคุมที่มีขนาดเล็ก DATA BUS สามารถ INTERFACE เพื่อใช้ในการขยายวงจร ในส่วนของการขยาย PORT I/O ให้มากขึ้น PROGRAMABLE PORT CHIP ที่นิยมมากที่สุดคือ 8255 ซึ่งรายละเอียดของ 8255 จะกล่าวต่อไป 8255จะมี REGISTER MODE ภายในเพื่อเป็นตัวควบคุมซึ่งสามารถเขียนได้โดย 8031 CONTROLLER WORDS จะมีรายละเอียดของหน้าที่ของ 8255 PORT คือ A B และ C โดยสามารถเลือกว่าจะใช้เป็นINPUT หรือ OUTPUT หรือจะใช้เป็นแบบทั้ง 2 เลยก็ได้ การขยาย PORT สามารถออกแบบได้ดังนี้

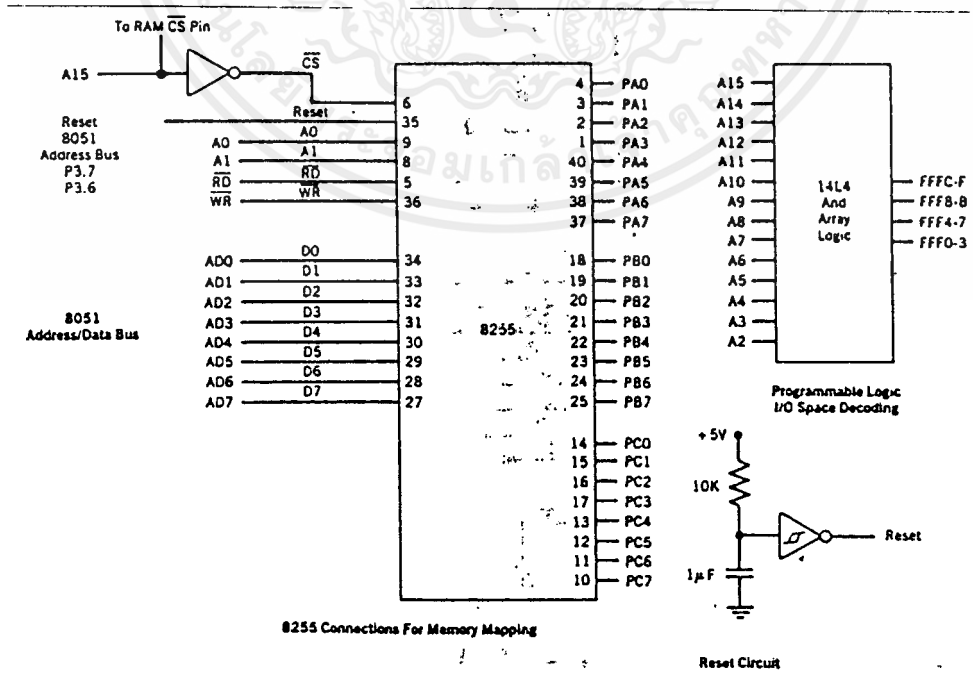
MEMORY MAPPED I/O

8255 PROGRAMABLE I/O PORT CHIP ใช้สำหรับขยาย PORT โดยสามารถใช้ร่วมกับ RAMโดยการออกแบบซึ่งแสดงดังรูป 2 โดยการออกแบบให้ RAM มีหน่วยความจำ 32 KBYTES จาก 64 KBYTES 8255 PORT สามารถใช้ขา ADDRESS A15 เป็น ADDRESS สูง (8000H) และ 32 KBYTES RAM สามารถแสดงที่ขา A15 เป็น ADDRESS ต่ำ (7FFFH และ ต่ำกว่า) การ DECODE จะต้องใช้ขา A15 รวมกันดังนั้นจึงต้องใส่ INVERTER เพื่อการ DECODE หน่วยความจำสำหรับ RAM และ I/O

รูปที่ 2 เป็นการออกแบบโดยจะรวม MEMORY MAPPED PORT I/O ทั้ง 3 คือ ADDRESS FFF0H-FFF3H ; FFF4H-FFF7H ; FFF8H-FFFBH ; และ FFFCH-FFFFH โดย ADDRESS ของ RAM คือ 0000H-FFEFH

ซึ่งทั้งหมดนี้เป็นการออกแบบ EXPANDING I/O แบบ MEMORY MAPPED I/O โดยใช้





8255 Connections For Memory Mapping

Reset Circuit

บทที่ 3

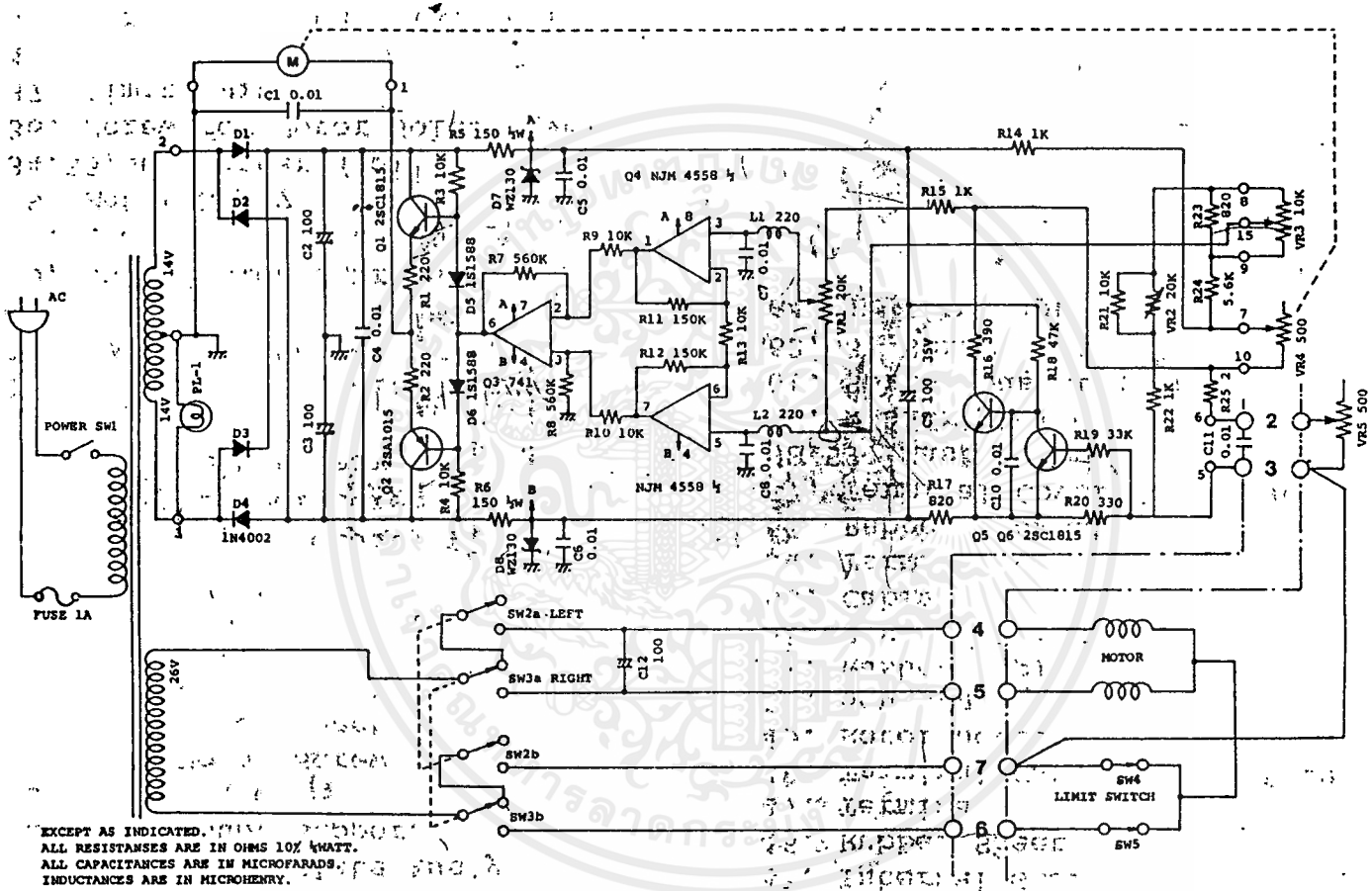
ภาค DRIVER

ROTOR เป็นตัวหมุนทิศเสาอากาศ ซึ่งนักวิทยุสมัครเล่นนิยมใช้กัน และที่จะกล่าวต่อไปนี้เป็น rotor รุ่น KR-600 ซึ่งถูกออกแบบมาเพื่อหมุนสายอากาศขนาดใหญ่ (ดูจากรูปที่)

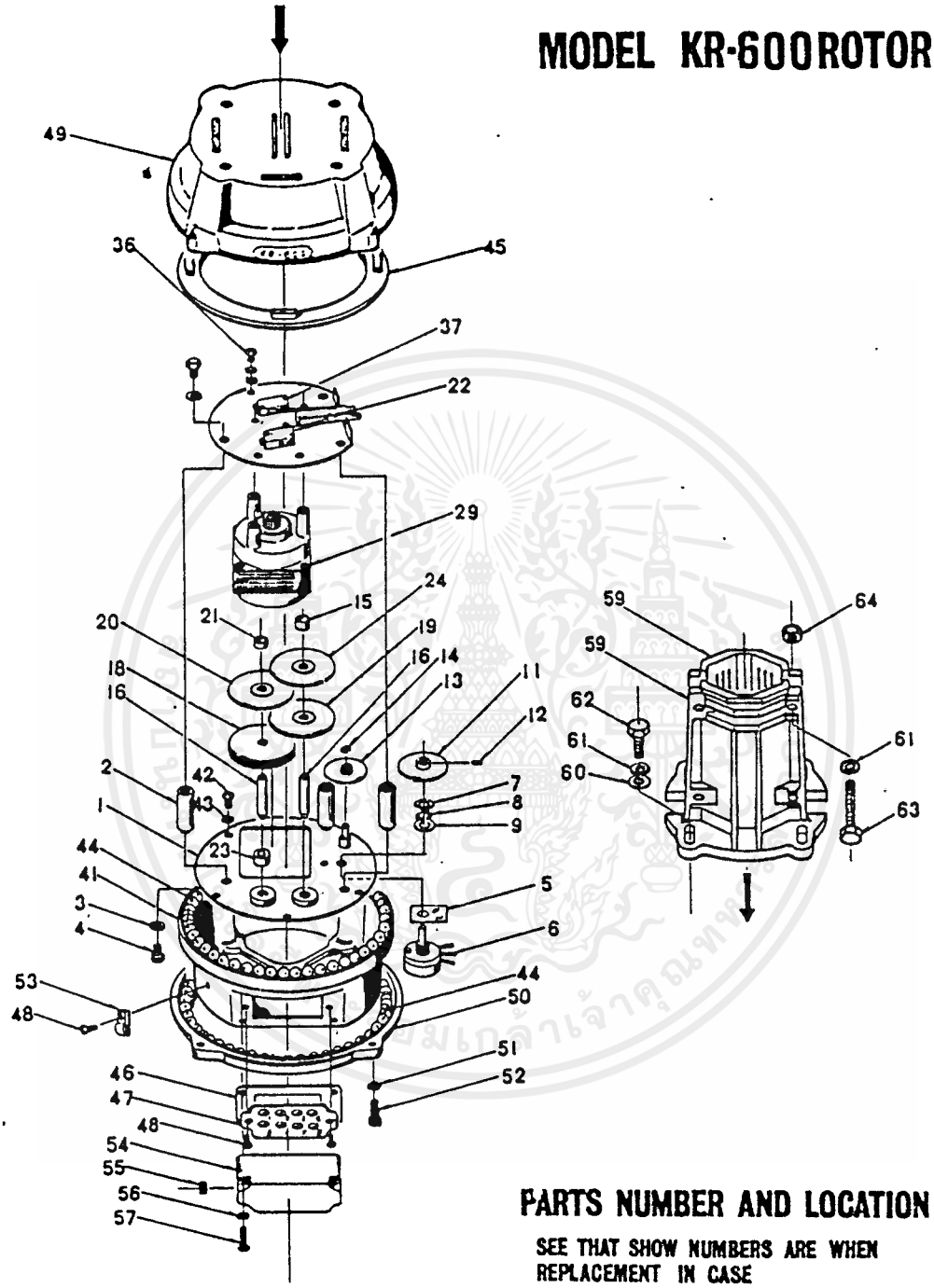
CHARECTERISTIC	
INPUT VOLTAGE	115/230 VOLT AC 50/60 Hz
POWER CONSUUMPTION	40 VA
MOTOR	24 VOLT SPLITPHASE
ROTATION TIME	APPROX 53 sec/60 Hz
	63 sec/50 Hz
END OF ROTATION STOP	ELECTRICAL AND MACHANICAL
ROTATION TORQUE	600 Kg-cm
STATIONARY BREAKING TORQUE	4,000 Kg-cm
VERTICAL LOAD	2,00 Kg
PERMISSIBLE MAST SIZE	38-63 m/m diameter
CABLE TOBE USED	6 conductor

ที่กล่าวมาข้างต้นเป็นลักษณะทั่วไป ต่อไปมาดูลักษณะถึงส่วนประกอบภายในของ KR-600 ดังรูปที่ 2

KR-600RC SCHEMATIC DIAGRAM



MODEL KR-600ROTOR



PARTS NUMBER AND LOCATION

SEE THAT SHOW NUMBERS ARE WHEN REPLACEMENT IN CASE

หลักการทำงานของวงจร DRIVER

จากวงจรเพื่อจุดประสงค์ในการแยกวงจรกำลัง (POWER) ออกจากวงจรควบคุมซึ่งเป็นส่วนที่ INTERFACE กับ microcontroller และเพื่อป้องกัน HIGH VOLTAGE จึงมีการนำ MOC3010 มาทำการ ISOLATE แยกวงจรออกจากกันและจากการที่ ROTER เป็น A.C. MOTER จึงนำ TRIAC มาใช้ร่วมกับ MOC3010 โดย MOC3010 จะเป็นตัว TRIC กระแส GATE จะอยู่ในช่วง 15 mA < IGT < 50 mA โดยให้ TRIAC เป็นเพียง SWITCHตัวหนึ่งซึ่งทำงานตามคุณลักษณะของไฟA.C. และเพื่อป้องกันไม่ให้ขดที่หมุนตามเข็ม (CW. WINDING) ทำงานพร้อมกับขดที่หมุนทวนเข็ม (CCW. WINDING) จึงใช้วงจร COMBINATION ป้องกันส่วนนี้ไว้

CW	CCW	FUNCTION
0	0	NO USE
0	1	หมุนทวนเข็ม
1	0	หมุนตามเข็ม
1	1	NO USE

ตารางแสดง LOGIC ทางด้าน I/P ของวงจร

และในส่วนของ J-K FLIP-FLOP ที่ดัดแปลงทำเป็น T FLIP-FLOP ก็เพื่อสะดวกในการควบคุมโดย MICROCONTROLLER และ R-C ที่ต่อร่วมกับ TRIAC เป็นลักษณะของวงจร R-C SNUBBER

การทำงาน

การทำงานของ TRIAC จะทำงานครั้งละ 2 ตัว สำหรับการหมุนไม่ว่าจะเป็นการหมุนทวนเข็มหรือตามเข็มนาฬิกา ดังต่อไปนี้ P2-P4 เป็น CW ROTATION และ P1-P3 เป็น CCW ROTATION และจะทำงานตามเงื่อนไขของ Truth Table

(ROTOR)

1. GEAR MOUNT PLATE ASS'Y
2. GEAR MOUNT SUPPORT
3. WASHER (6)
4. GEAR MOUNT SCREW
5. INSULATOR SHEET
6. POTENTIOMETER
7. NUT (9)
8. SPRING WASEHER (9)
9. WASHER (9)
11. POT DRIVER GEAR
12. GEAR STOPPER SCREW
13. PLASTIC POT GEAR
14. E-RING (2.5)
15, 21. STUD SUPPORT SLEEVE
16, 17. GEAR SHAFT
18. GEAR
19, 20. PINION/GEAR ASS'Y
22. GEAR MOTOR MOUNT ASS'Y
29. MOTOR ASS'Y
34, 35, 56. WASHER (4)
36 SCREW FOR MOTOR HOLDER (4)
37 LIMIT SWITCH
41 CASE
42 SCREW FOR MOUNT PLATE HILDER
43 WASHER(5)
44 BALL BEARING
45 INTERNAL GEAR
46 RUBBER SHEET
47 TERMINAL
48 TERMINAL/CABLE HOLDER SCREW
49 ROTOR HOUSING
50 HOUSING
51 WASHER (6)
52 HOUSING SCREW

53 CABLE HOLDER

54 TERMINAL COVER

55 RUBBER GROVERMENT

56 SPRING WASHER(4)

57 TERMINAL COVER SCREW (4)

58,59 MAST CLAMP

60 WASHER (8)

61 SPRING WASHER (8)

62,63 SCREW (8)

64 HEX NUT M8

การนำ ROTOR ประยุกต์ใช้

จากที่กล่าวถึงลักษณะทางไฟฟ้าและทางกลของ KR-600 เขียนเป็นวงจรได้ดังรูปที่ 3 ซึ่งมีส่วนประกอบดังต่อไปนี

1. MOTER ซึ่งประกอบด้วยขดลวดสองชุด ชุดแรกเป็นขดที่ไว้หมุนแบบทวนเข็มและขดที่สองเป็นขดที่ไว้หมุนแบบตามเข็ม หรือเรียกว่า split phase เป็นมอเตอร์ประเภท single phase แบบ induction ตัว rotor ของตัวมอเตอร์เป็นแบบกรงกระรอก
2. LIMIT SWITCH เป็นแบบปกติปิด
3. PONTENTIOMETER

จากวงจรที่ 1 ตัวมอเตอร์ซึ่งเป็นแบบ Spit Phase มีขดลวดอยู่ 2 ขดด้วยกัน ขดแรกจะทำการหมุนแบบตามเข็มนาฬิกา และขดที่ 2 จะหมุนแบบทวนเข็มนาฬิกา เมื่อจ่ายไฟให้กับขดลวด โดยที่ตัว Rotor จะมี terminal ดังรูป

เมื่อขดแรกทำงานตัว Rotor จะหมุนแบบทวนเข็มนาฬิกา จะเริ่มหมุนไปจนครบ 360 องศา ตามคุณลักษณะของของ Rotor โดยตัวที่ตัดการทำงานก็คือ Limit Switch ซึ่งจะเป็นแบบ ปกติปิดลักษณะการทำงานของ Limit Switch จะตัดได้โดย เมื่อ Rotor หมุนมาที่ 360 องศา ลักษณะทางกลจะทำให้ Limit Switch Open ดังนั้นการทำงานของ Rotor จะหยุดลง ในทำนองเดียวกันนี้การหมุนแบบทวนเข็มก็เช่นเดียวกัน และในขณะที่ Rotor หมุน Potentiometer หมุนไปตามการหมุนของ Rotor ด้วย ซึ่งสามารถตรวจจับได้จากขั้วที่ 2 เมื่อมี Vref ป้อนให้ ดังนั้นการหมุนของ Rotor จึงสามารถแสดงผลได้และสามารถที่จะควบคุมการทำงานของ Rotor ได้เช่นเดียวกัน

การติดตั้ง KR-600

ในการติดตั้ง KR-600 ถ้าเป็นเสาอากาศที่มีขนาดใหญ่ น้ำหนักมาก ควรจะมีการตรวจสอบความ

เร็วของลม ณ. จุดที่ต้องการจะติดตั้งก่อน เพราะเมื่อเกิดลมพัดแรงแล้ว ความแรงของลมอาจทำให้ช่วงตัว Mast clamp เกิดการเสียหายได้ และท่อที่ต่อเชื่อมระหว่างเสาอากาศ กับ mast clamp งอได้ ซึ่งจะมีการทำเหล็กปิดครอบตัว Rotor และระยะจากตัว mast clamp ถึงเสาอากาศจะให้ได้ไม่เกิน 3 ฟุต ดูรูปประกอบ ส่วนสายที่จะใช้ต่อระหว่าง Rotor กับ Controller จะใช้ได้ดีในระยะ 100 ฟุต ต้องเป็นสายเบอร์ #22 ถ้าสูงกว่าที่กล่าวมาก็ใช้สาย #20

บทที่ 4

SPEED & DIRECT OF WIND

ลมพายุ เป็นภัยธรรมชาติชนิดหนึ่งที่มนุษย์เราได้พบเจอมาทั้งในอดีต ปัจจุบัน และยังคงจะต้องเจอในอนาคตอีกแน่นอน และมันเป็นภัยธรรมชาติที่มนุษย์เราไม่สามารถที่จะหลีกเลี่ยงได้ ซึ่งอำนาจของมันอาจจะก่อให้เกิดความเสียหายแก่ทรัพย์สินโดยทั่วๆ ไป ส่วนจะก่อให้เกิดความเสียหายมากน้อยแค่ไหนนั้นขึ้นอยู่กับความแรงหรือความเร็วของลมพายุนั้นจากข้อมูลของกรมอุตุนิยมวิทยาของประเทศเรานั้นได้แบ่งลักษณะของลมไว้ตามความรุนแรงหรือความเร็วของมันได้ดังนี้

ตารางที่ 1

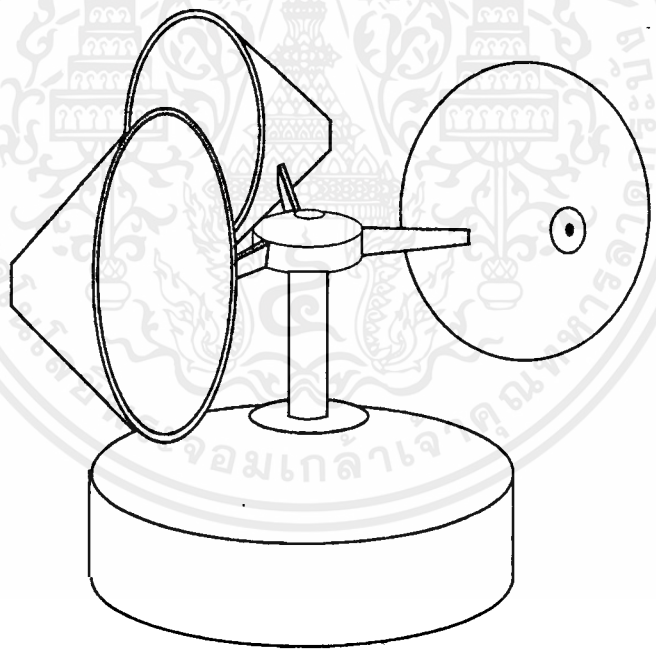
ขนาดของลม	สัญญลักษณ์ที่แสดงบนบก	ความเร็วที่ระดับสูง
ลมสงบ	ลมเงียบ ควันลอยขึ้นตรง ๆ	< 1
ลมเบา	ควันลอยไปตามลมแต่ศรลมไม่หันไปตามทิศลม	15
ลมอ่อน	ลมปะทะหน้า ใบไม้ไหวดังกรอบแกรบ ศรลมหันไปตามทิศลม	6 - 11
ลมโชย	ใบไม้และกิ่งไม้เล็ก ๆ กระดิก ธงปลิวไสว	12 - 19
ลมปานกลาง	ฝุ่นผัดตลบ กระดาษปลิว กิ่งไม้เล็กขยับเขยื้อน	20 - 28
ลมแรง	ต้นไม้เล็ก ๆ โยก น้ำในลำคลองเป็นระลอก	29 - 38
ลมจัด	กิ่งไม้ใหญ่ขยับเขยื้อน ได้ยินเสียงหวิวตามสายโทรเลข ใช้ร่มลำบาก	39 - 49
พายุเกลอ่อน	ต้นไม้ใหญ่ทั้งต้นแกว่งไกว เดินทวนลมไม่สะดวก	50 - 61
พายุเกล	กิ่งไม้หัก การเดินละบากขึ้น	62 - 74
พายุเกลแรง	อาคารที่ไม่มั่นคงหักพัง ป้ายและหลังคาปลิว	75 - 88
พายุ	ต้นไม้ถอนรากถอนโคน สิ่งก่อสร้างพังทลาย เกิดความเสียหายมาก (ไม่ค่อยปรากฏบ่อยนัก)	89 - 102
พายุใหญ่	เกิดความเสียหายทั่วไป (ไม่ค่อยปรากฏ)	103 - 117
ได้ฝุ่น หรือ เฮอร์ริเคน		118 หรือมากกว่า

ในปัจจุบันเสาอากาศเป็นอุปกรณ์ชนิดหนึ่งที่ถูกนำมาใช้กันอย่างแพร่หลาย ร่วมกับอุปกรณ์ต่างๆ ในการติดต่อสื่อสาร ซึ่งลักษณะการใช้งานโดยทั่วไปของเสาอากาศ จะถูกติดตั้งไว้อยู่กับที่ และมีทิศทางในการติดตั้งที่แน่นอน จากลักษณะเช่นนี้จึงอาจจะก่อให้เกิดความเสียหายอันเนื่องมาจากลมพายุได้ เมื่อทิศทางของเสาอากาศอยู่ในตำแหน่งที่ต้านกับทิศทางของกระแสลม การแก้ไขง่าย ๆ ที่อาจจะทำได้โดยไม่ยุ่งยากจากที่อาจจะต้องเก็บเสาอากาศลงมาจากเสา) ทางหนึ่งก็คือ หันทิศทางของเสาอากาศให้มีทิศทางเดียวกันกับกระแสลมแต่เราจะทราบได้อย่างไรว่าเมื่อไหร่กระแสลมจะมีอันตรายต่อเสาอากาศของเรา และทิศทางไหนที่เป็นทิศทางที่ปลอดภัยที่สุด

โครงสร้างของ Wind Speed

Wind Speed จะเป็นตัวตรวจจับหรือคอยวัดความเร็วของลมโดยอาศัยตัว Sensor ที่มีลักษณะการทำงานหมุนตามความเร็วลม หรืออาจจะพูดได้ว่าจะหมุนด้วยความเร็วที่ใกล้เคียงหรือเท่ากับความเร็วของกระแสลมโดยที่ตัว Sensor จะถูกติดตั้งไว้ที่ตำแหน่งที่สัมพันธ์กับตำแหน่งของเสาอากาศ ดังนั้นความเร็วลมตรงจุดที่ Sensor ตรวจวัดได้ก็คือ ความเร็วลมที่กำลังกระทำกับเสาอากาศนั้นเอง

ตัว Sensor จะมีลักษณะที่อาจจะเรียกได้ว่า **เป็นตัวรับลม** คือจะมีลักษณะรูปร่างดังรูปที่ 1



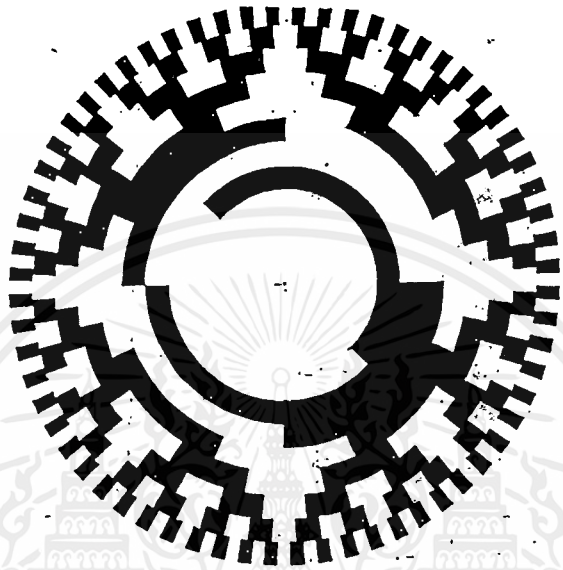
รูปที่ 1

ชุดเคลื่อนที่ จะมีส่วนที่คอยรับลม (ส่วนที่มีลักษณะเหมือนกรวย) เมื่อมีลมมาปะทะ ส่วนนี้ก็จะทำการหมุน ส่วนจะหมุนเร็วหรือช้าก็ขึ้นอยู่กับควมแรงของลมซึ่งจะสัมพันธ์กับความเร็วลมด้วยแล้วจะส่งการหมุนที่รับได้ลงมายังชุดอยู่กับที่ ชุดอยู่กับที่จะแปลงการหมุนนี้ออกมาเป็นรหัส (ความถี่) ข้อมูลส่งออกมาเป็นพัลซ์ เพื่อส่งต่อไปยังส่วนกลางการประมวลผลเพื่อทำการประมวลผลต่อไป

ภายในชุดที่อยู่กับที่ (FIXED PART) จะแบ่งเป็น 2 ส่วนคือ

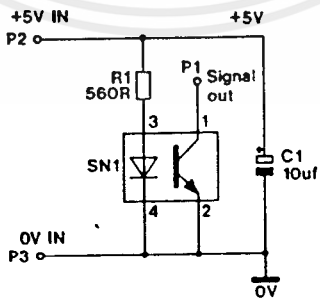
- CODE DISK
- CIRCUIT PART

CODE DISK จะต่ออยู่กับแกนที่ส่งมาจากชุดเคลื่อนที่เมื่อชุดเคลื่อนที่หมุน ตัว CODE DISK ก็จะหมุนด้วยด้วยความเร็วที่เท่ากับชุดเคลื่อนที่ CODE DISK จะมีลักษณะดังรูปที่ 2



รูปที่ 2 CODE DISK

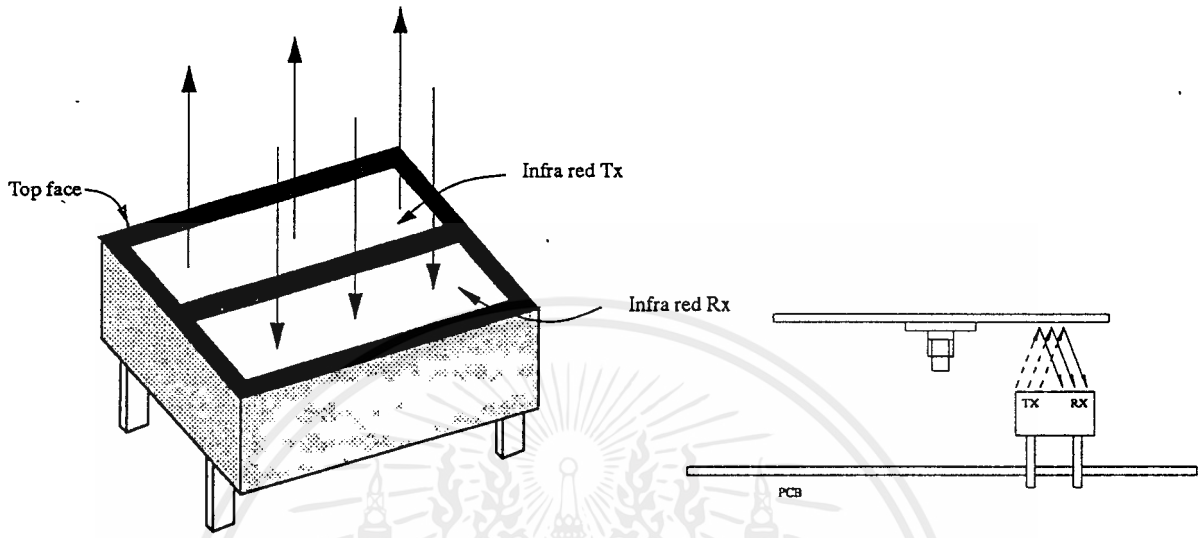
CIRCUIT PART เป็นส่วนที่จะแปลงให้ค่าการหมุนที่ส่งมาเปลี่ยนเป็น ข้อมูลในลักษณะที่เราสามารถนำไปประมวลผลได้ ในที่นี้เราจะแปลงการหมุนให้ออกมาเป็น พัลซ์ที่มีระยะห่างระหว่างระหว่างพัลซ์ต่าง ๆ กัน ซึ่งระยะห่างระหว่างพัลซ์นี้จะขึ้นอยู่กับความเร็วของการหมุนของ CODE DISK วงจรของ CIRCUIT PART นี้ อาจแสดงได้ดังรูปที่ 3



รูปที่ 3 CIRCUIT PART ของ WIND SPEED

การทำงานของวงจรนี้เมื่อเราจ่ายไฟเลี้ยงให้กับวงจร LED ใน Phototransistor จะปล่อยคลื่นแสงอินฟาเรดออกมาแสงอินฟาเรดเป็นคลื่นแม่เหล็กไฟฟ้าชนิดหนึ่งซึ่งจะมีคุณสมบัติในการสะท้อนของคลื่น ดังนั้นถ้าแสงอินฟาเรดนี้วิ่งไปชนกับวัสดุ (ที่ไม่ซับคลื่น) ใด ๆ ก็จะทำให้เกิดการสะท้อนขึ้น และถ้า

การสะท้อนนั้นอยู่ในตำแหน่งที่ถูกต้อง ลำแสงอินฟาเรดนี้ก็จะสะท้อนมาที่ตัวรับของ Phototransistor (ดังรูปที่ 4) ซึ่งตัวรับแสงอินฟาเรดนี้จะเป็นขา Baseของ Transistor ภายใน Phototransistor ทำให้ Transistor นำกระแส จึงมีกระแสไหลจากขา 1 ลงมาขา 2 (เปรียบเสมือนสภาวะวงจรปิด หรือสภาวะ าดำทางโลจิก) ซึ่งปกติระหว่างขา 1 และขา 2 จะไม่มีแรงดันตกคร่อมอยู่เลย เนื่องจากทรานซิสเตอร์ยังไม่นำกระแส (สภาวะ "1")



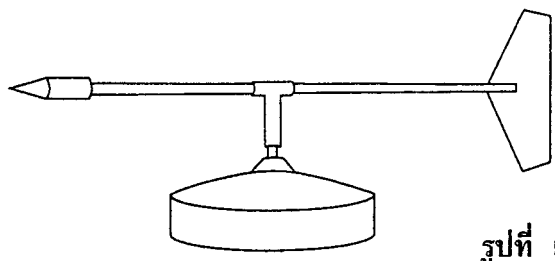
จากลักษณะการทำงานของ CIRCUIT PART เราจะนำส่วน CODE DISK เข้ามาร่วมทำงานด้วย ถ้าเราติดตั้งให้ตำแหน่งโครงสร้างของทั้งสองส่วนนี้สัมพันธ์กัน เมื่อ CODE DISK หมุน หรือเคลื่อนที่ไปเรื่อยๆ ตามกระแสลม โดยปกติระดับสัญญาณที่ขา 1 จะเป็น " 0"

แต่เมื่อตำแหน่งสีขาวของ CODE DISK (ดังในรูปที่ 1) หมุนมาตัดกับแนวแสงอินฟาเรด คือจะมีการสะท้อนของลำแสงอินฟาเรดกลับมาที่ขา Base ของ Transistor ทำให้ Transistor มีสภาวะเป็น าONำ ซึ่งจะส่งผลให้ระดับสัญญาณที่ขา 1 เป็น "0"

ดังนั้นที่ขา 1 นี้เราสามารถต่อหรือนำออกมาเป็นข้อมูลให้กับส่วนประมวลผล เพื่อนำข้อมูลที่ได้นี้ไปคำนวณหาความเร็วที่วัดออกมาได้

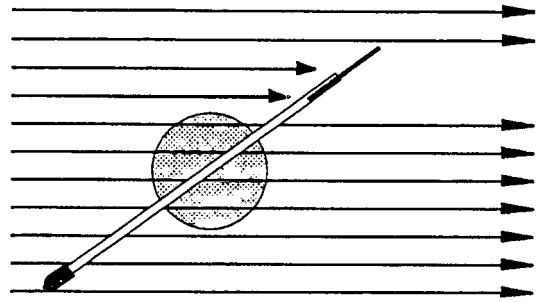
โครงสร้างของ WIND DIRECTION

WIND DIRECTION เป็นตัวที่จะบอกว่าขณะนี้กระแสลมมีทิศทางอยู่ที่ทิศทางใดโดยที่การติดตั้งเริ่มแรก เราจะต้องตั้งจุดอ้างอิงให้แก่ WIND DIRECTION ก่อน (เช่น ให้รหัส -00000000 ตรงกับทิศเหนือ ส่วนประกอบที่สำคัญส่วนหนึ่งของ WIND DIRECTION จะมีลักษณะคล้าย ๆ กับหางปลาเจ้าตัวนี้เองที่เราจะ อาศัยมันในการบอกทิศทางลม รูปร่างของส่วนนี้จะแสดงได้ดังรูปที่ 5

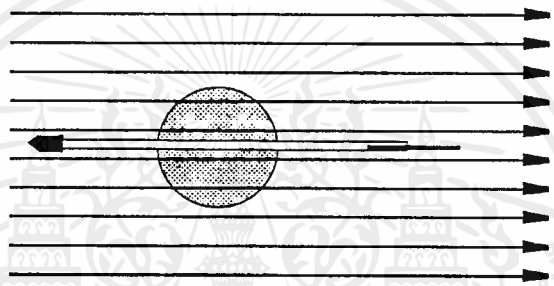


รูปที่ 5 WIND DIRECTER

เมื่อใดก็ตามที่ส่วนหางปลา(ซึ่งเป็นส่วนที่มีพื้นที่ที่จะปะทะกับกระแสลมมากที่สุดเมื่อเทียบกับส่วนอื่น ๆ) อยู่ในตำแหน่งที่ต้านกับกระแสลม ดังรูปที่ 6(ก)



(ก)



(ข)

รูปที่ 6

ตัวหางปลานี้จะถูกผลักกระแสลมให้เคลื่อนที่ จนกระทั่งมันเคลื่อนที่มาอยู่ในทิศทางตามลม หรืออยู่ในตำ แหน่งที่กระแสลมกระทำต่อชุดหางปลานี้น้อยที่สุด ณ ตำแหน่งนี้ทิศทางที่ลูกศรชี้นี้ก็เปรียบเ เสมือนกับ ทิศทางของกระแสลมนั่นเอง (รูปที่ 6(ข))

การที่จะรู้ได้ว่าขณะนี้ลูกศรชี้ทางไหน ก็จะใช้หลักการ Sensor โดยมีหลักการคล้าย ๆ กับ WIND STAR คือจะมีชุดที่เคลื่อนที่ (ลูกศรและหางปลา) ซึ่งจะส่งต่อการหมุนลงมาปลายข้างล่างจะมี CODE DISK สำหรับ WIND DIRECT ติดอยู่ และใกล้ ๆ กับ CODE DISK ก็จะมี CIRCUIT PART ติดตั้งอยู่ ซึ่งตำแหน่งของ CODE DISK และ CICUIT PART จะต้องสัมพันธ์กัน

แต่สิ่งที่ WIND DIRECT ต่างจาก WIND SPEED คือ OUTPUT ของ WIND SPEED จะส่งออกมาในลักษณะ pulse เป็นจังหวะ ๆ คือเราจะได้ค่าที่ต้องการจากความห่างของ pulse แต่ WIND DIRECT ข้อมูลที่ส่งออกมาจะเป็นข้อมูลที่เป็นรหัส และมีความจะเป็นที่จะต้องมีสายข้อมูลหลายบิทเพื่อความละเอียดของทิศทางที่จะอ่านออกมาจากลักษณะรูปแบบของข้อมูลที่มีสภาวะอยู่ 2 สภาวะ คือ สภาวะที่ Transistor "ON"กับ Transistor "OFF" หรือค่าที่ได้รับออกมาจาก OUTPUT
ก็จะเป็น +Vcc กับ GND ตามลำดับ ดังนั้นเลขฐาน 2 จึงเหมาะที่จะทำมาใช้แทนค่าที่ได้ ค่าที่ได้เราสมารถที่จะนำมาสร้างเป็นรหัสรูปแบบต่าง ๆ ได้หลายรูปแบบ ในที่นี้มีรหัสที่น่าสนใจอยู่ 2 รูป แบบ คือ

- รหัสไบนารี (Binary Code)

- รหัสเกรย์ (Gray Code)

รหัสไบนารี่ เป็นรหัสที่ให้แทนอะไรก็ได้ที่แตกต่างกัน โดยที่ถ้ามีเลข Binary n ตัวก็จะแทนสิ่งของที่ต่างกันได้ 2 n สิ่ง ถ้าอะไรก็ได้นี้เป็นตัวเลขฐานสิบ ก็สามารถแทนตัวเลขได้ตั้งแต่ 0 ถึง 2 n -1 โดยที่รหัสไบนารีนี้เกิดจากการแปลงฐานเลขจากฐานสิบเป็นฐาน 2 "โดยตรง "

รหัสเกรย์ ก็เป็นรหัสแบบไบนารีชนิดหนึ่ง ซึ่งรหัสแบบเกย์นี้จะสามารถเขียนเป็นความสัมพันธ์กับรหัสแบบไบนารีได้ดังนี้ (ให้ N = 4)

g0 = b0xor b1

g1 = b1xor b2

g2 = b2 xorb3

g3 = b3

เครื่องหมาย เรียกว่าเครื่องหมาย Eeclusive OR หรือ Modulo - 2 Sum (คล้าย ๆ กับการบวกเลขไบนารี แต่ไม่คิดตัวทด)

0 xor0 = 0 0 xor1 = 1

1 xor0 = 1 1 xor 1 = 0

ข้อดีและข้อแตกต่างในการนำรหัสแบบ Binary และแบบ Gray มาใช้

- Binary Code จะมีน้ำหนักประจำตำแหน่งแน่นอน และเป็นรูปแบบข้อมูล ซึ่งไมโครคอนโทลเลอร์ใช้ได้ทันที

- Gray Code รหัสแบบนี้เป็นแบบที่ไม่มีน้ำหนักประจำตำแหน่ง แต่มีลักษณะพิเศษ คือ รหัสของเลข 2 ตัวที่ติดกัน จะมีบิทที่ต่างกันเพียงบิทเดียวเท่านั้น ฉะนั้นถ้านำรหัสแบบ Gray Code มาใช้จะสามารถป้องกันการ error ของข้อมูบได้ดีกว่าแบบ Binary Code แต่ถ้าใช้ Gray Code เวลาส่งข้อมูลมาที่ Micro controllor จะต้องแปลงข้อมูลให้กลายเป็น Binary Code เสียก่อนจึงจะนำมาประมวลผลได้

ฉะนั้นทั้งสองแบบจึงมีทั้งข้อดีและข้อเสียที่แตกต่างกันการนำมาใช้จึงขึ้นอยู่กับความต้องการของผู้ใช้ เมื่อเรานำรูปแบบรหัสที่เราเลือกแล้วมาทำ Code disk ในที่นี้ได้ออกแบบ Code Disk ทั้งแบบ Binary Code และแบบ Gray Code (ซึ่ง Code Disk ทั้ง 2 รูปแบบนี้สามารถนำมาใช้กับ Circuit Part ได้ทั้งสองรูปแบบ) ไว้ดังรูปที่ 7 และ 8

จากรูปที่ 7 และรูปที่ 8 จะเห็นว่า CODE DISK จะมีลักษณะเป็นวง ซึ่งแต่ละวงจะแทนแต่ละบิท จากจุดนี้เราจะเห็นว่า ถ้าการตั้ง CODE DISK และ CIRCUIT PART ไม่สัมพันธ์กัน ข้อมูลที่ออกมาจะไม่ใช่ข้อมูลที่แท้จริง ค่าที่ได้จะเป็นค่าที่ไม่แน่ชัด

ตารางที่ 2 Gray Code และ Binary Code ของเลข 0 ถึง 15

N	g3 g2 g1 g0 Gray Code	b3 b2 b1 b0 Binary Code
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Circuit Partจะมีลักษณะการทำงานที่คล้ายๆ กับ Wind Speed คือจะใช้ Phototransistor มาเป็นตัว Sensor เพื่อใช้ร่วมกับ Code Disk แต่จะแตกต่างกันที่รูป"แบบของข้อมูลดังตามที่ได้กล่าวมาแล้ว วงจรสามารถแสดงได้ดังรูปที่ 9

การออกแบบตำแหน่งของPhototransistorได้จัดวางไว้ตามรูปแบบดังรูป เพื่อป้องกันการรบกวนกันของคลื่นอินฟาเรดในแตละบิท และการที่เราใช้ Binary Code ที่มีถึง 8-Bit ก็จะทำให้เราสามารถแบ่งเป็นทิศทางได้ถึง 256 ทิศทางซึ่งจะมีความละเอียดที่สูงพอสมควร และเป็นความละเอียดที่เพียงพอที่จะใช้ในโครงงานนี้

ลักษณะการทำงานของ Wind Direct เมื่อนำมาใช้ร่วมกัน MicroController

เมื่อ CPU ต้องการที่จะทราบตำแหน่งของทิศทางลมก็จะส่งสัญญาณมาบอกกับ Wind Direction ว่าต้องการข้อมูลจาก Wind Direction สัญญาณที่ส่งมาบอกความจริงก็คือ ส่งสัญญาณมาเพื่อที่จะแลทเอาข้อมูลในขณะนั้นให้ปรากฏออกมาที่ Output ของ Wind Direction (Output 74HC175) เสร็จแล้ว CPU ก็จะรับเอาค่าที่แลทไว้นี้เข้ามาแทนที่ CPU เพื่อนำข้อมูลที่ได้นี้มาประมวลผล ซึ่งกระบวนการต่อจากนั้นก็จะเป็นเรื่องของทางด้าน Software ที่จะจัดการอย่างไรต่อไป

บทที่ 5

GRAPHIC LCD และ KEYBOARD

GRAPHIC LCD.

อุปกรณ์ในปัจจุบันนี้ในส่วนแสดงผล มักใช้ LCD (Liquid Crystal Display) เสียเป็นส่วนมาก สามารถแบ่ง LCD โมดูล ชนิดดอทเมตริกซ์ ออกได้เป็นพวกๆ คือ

ชุด LCD โมดูลแสดงผล แบบอักขระ (character LCD module)

ชุด LCD โมดูลแสดงผล แบบกราฟิก (graphic LCD module)

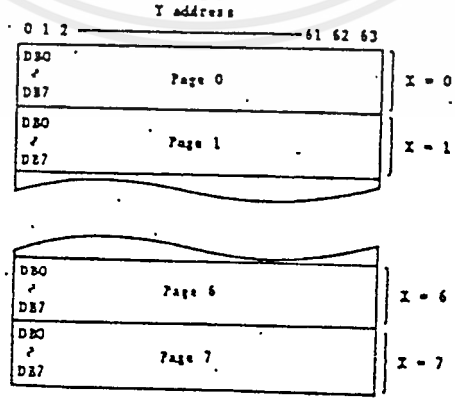
ชุด LCD โมดูลแสดงผล แบบเซกเมนต์ (segment LCd module)

DV-12864 เป็น LCD graphic ขนาด 128x64 dot ซึ่งมี controller ภายใน (HD61202,HD61203) โดยการทำงานของ controller จะมีการแบ่งการควบคุมไว้ดังนี้

Line คือการอ้างถึงบรรทัดข้อมูล ภายในจะแบ่งเป็น 64 แถว (แถว1-แถว64)

Page (A-Address) เป็นการอ้างถึงหน้าต่างแสดงผล ภายในหนึ่ง Page จะ ประกอบไปด้วย 8 Line ซึ่งจะเป็นการอ้างถึงข้อมูลด้วย Data-Bus โดยตารางการทำงานภายในของ LCD จะประกอบด้วย 8 Page ซึ่งถูกชี้โดย X-Registor โดยเมื่อต้องการให้ LCD แสดงผลที่หน้าต่างใดของจอ จะต้องตั้งค่า X ให้กับ LCD ซึ่งเมื่อตั้งค่า X ให้กับ LCD แล้วค่า X นั้นจะไม่มีการเปลี่ยนแปลง จนกว่าจะมีการตั้งค่าใหม่ให้กับ LCD

Segment (Y-Address) เป็นค่าพอยท์เตอร์ ในการชี้ที่อยู่ของข้อมูลซึ่งภายใน LCD จะถูกควบคุมการชี้ข้อมูลโดย HD161202 ซึ่ง HD161202 สามารถชี้ที่อยู่ของข้อมูลได้64 Segment ซึ่ง HD161202 ทั้งสองตัวจะสามารถอ้าง Segment ได้ถึง128 Segment



รูปที่ 1 แสดงการแบ่งการควบคุม

โดยการใช้งาน เมื่อทำการตั้งค่า Y แล้ว ค่าจะถูกเพิ่มขึ้นเสมอ เมื่อมีการอ่านหรือเขียนข้อมูลบน LCD เมื่อค่า ถูกเพิ่มขึ้นมากกว่า 63 แล้ว ค่า Y จะยังไม่เป็นการอ้างข้อมูลในเซกเมนต์ที่ 64 (เซกเมนท์ 0 ของ CS2) ดังนั้นตัวโปรแกรมจะต้องช่วยจัดการในส่วนนี้

คำสั่งควบคุมของ LCD

1. DISPLAY ON/OFF

R/W D/I DB7 ---------- DB0

Code 0 0 0 0 1 1 1 1 1 D

high order bit - - low order bit -

เป็นคำสั่งควบคุมการแสดงผล โดยการแสดงผลจะขึ้นอยู่กับค่า D (DB0) เมื่อค่า D เป็น 1 LCD จะทำการแสดงผล และเมื่อค่า D เป็น 0 LCD จะไม่ทำการแสดงผลข้อมูลภายใน LCD จะไม่มีการเปลี่ยนแปลงเนื่องจากคำสั่งนี้

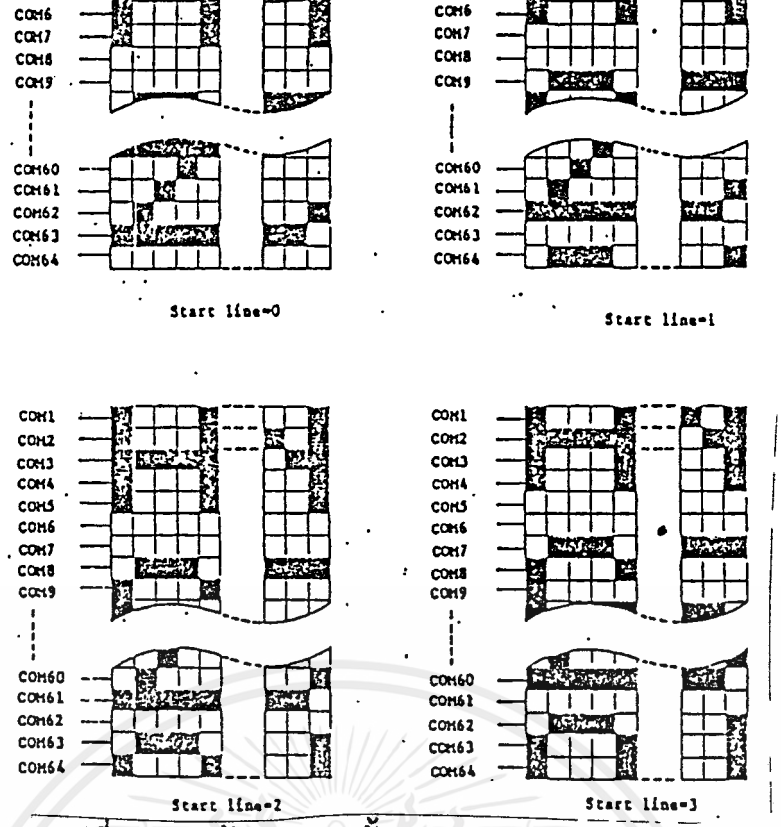
2. DISPLAY START LINE

R/W D/I DB7 ---------- DB0

Code 0 0 1 1 A A A A A A

high order bit - - low order bit -

ค่า A จะเป็นค่าหมายเลขบรรทัด ที่จะให้ LCD แสดงผลเป็นบรรทัดแรกของจอภาพในรูป เป็นตัวอย่างการเลือกค่า LINE จาก 0-3 ซึ่งจะทำให้การแสดงผลแตกต่างกันออกไป



รูปที่ 2 แสดงตัวอย่างการตั้งค่า DISPLAY START LINE

3. SET PAGE (X-ADDRESS)

R/W D/I DB7 ---------- DB0

Code 0 0 1 0 1 1 1 A A A

high order bit - - low order bit -

ค่า AAA ของคำสั่ง จะเป็นการตั้งค่า X-ADDRESS ซึ่งหลังจากทำคำสั่งนี้แล้ว ข้อมูลจาก DB0-DB7 จะเป็นการติดต่อกับ RAM ที่ PAGE นี้ตลอด จนกว่าจะมีการตั้งค่าใหม่ให้กับ LCD

4. SET Y-ADDRESS

R/W D/I DB7 ---------- DB0

Code 0 0 0 1 A A A A A A

high order bit - - low order bit -

ค่า A จะเป็นการตั้งค่าของ Y-ADDRESS (ค่า Y จะมีค่าอยู่ระหว่าง 0-63) และค่าY จะเพิ่มขึ้นครั้งละ 1 เมื่อมีการอ่านหรือเขียนข้อมูล จาก CPU

5. STATUS READ

R/W D/1 DB7 ---------- DB0

ON/ RE
Code 1 0 BUSY 0 OFF SET 0 0 0 0

high order bit - - low order bit -

เป็นการอ่านค่าสถานะของ LCD โดยถ้าค่า BUSY เป็น 1 LCD จะทำงานในส่วนภายในซึ่งจะทำให้ไม่สามารถควบคุม LCD ในขณะนี้ได้ เพื่อความแน่ใจ ในการควบคุมครั้งต่อไปจะต้องตรวจค่า BUSY ให้ได้ค่าเป็น 0 เสียก่อน

6. WRITE DISPLAY DATA

R/W D/1 DB7 ---------- DB0

Code 0 1 D D D D D D D D
high order bit - - low order bit -

เป็นการเขียนข้อมูลเข้าไปใน LCD ซึ่งข้อมูล DDDDDDD จะถูกจัดเก็บอยู่ใน LCD RAM และค่า Y จะถูกเพิ่มขึ้น 1

7. READ DISPLAY DATA

R/W D/1 DB7 ---------- DB0

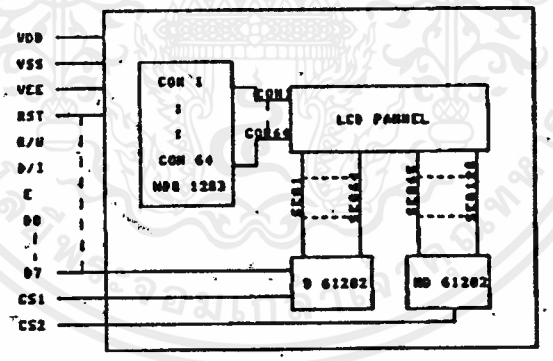
Code 1 1 D D D D D D D D

high order bit - - low order bit -

เป็นการอ่านค่าข้อมูลที่แสดงผล โดย LCD จะให้ค่าของข้อมูลออกมาที่ DATA BUS ค่า Y จะถูกเพิ่มขึ้น 1 เช่นเดียวกับการเขียนข้อมูล

การใช้งาน LCD DV - 12864

โครงสร้างภายในของ LCD จะประกอบด้วย ส่วนของ Controller โดย HD61203จะควบคุมกาอ้างถึง Page 0 ของข้อมูล และ HD61202 จะควบคุมในการอ้าง Segment ซึ่งในการใช้งาน จำเป็นจะต้อง Control ส่วนเหล่านี้ โดยการส่งรหัสควบคุมไปที่ขาของ LCD



รูปที่ 3 แสดงโครงสร้างภายในและขาควบคุม

ขา RST เป็นขาที่ใช้ RESET การทำงานของ LCD

ขา E เป็นขา ENABLE การรับส่งข้อมูล จะทำงานที่ LOGIC HIGH ขอบขาลง

ขา R/W ใช้กำหนด การอ่านหรือเขียนข้อมูล

ขา D/I ใช้ระบุว่า ข้อมูลใน DATA BUS เป็นรหัสควบคุมหรือเป็นข้อมูล

ขา CS1 CHIP SELECT ของ HD61202 ตัวแรก

ขา CS2 CHIP SELECT ของ HD61202 ตัวที่สอง

ขา D0-D7 เป็นขาที่ใช้รับส่งข้อมูลและรหัสควบคุม

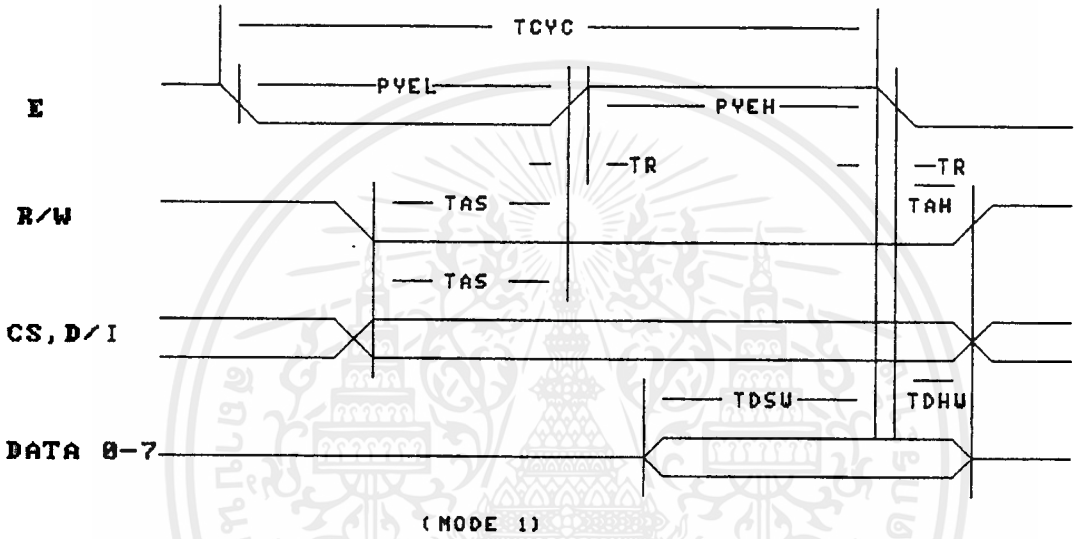
เมื่อขา CS1 เป็น HIGH และ CS2 เป็น LOW จะเป็นการอ้างอิงถึง Segment ที่0-63 และเมื่อขา CS1 เป็น LOW และขา CS2 เป็น HIGH แล้ว จะเป็นการอ้างอิงที่ 64-127 สภาวะการทำงานดูได้จากรูปที่ 3

ITEM	SYMBOL	LIMIT		NOTE
		min	max	
E cycle time	TCYC	1000	-	1,2
E high level width	PYEH	450	-	1,2
E low level width	PYEL	450	-	1,2
E rise time	TR	-	25	1,2
E fall time	TF	-	25	1,2
Adress setup	TAS	140	-	1,2
Adress hold time	TAH	10	-	1,2
data setup time	TDSW	200	-	1
data delay time	TTDR	-	320	2
data hold time (write)	TDHV	10	-	1
data hole time (read)	TDHR	20	-	2

รูปที่ 4 แสดงค่าเวลาในการทำงาน

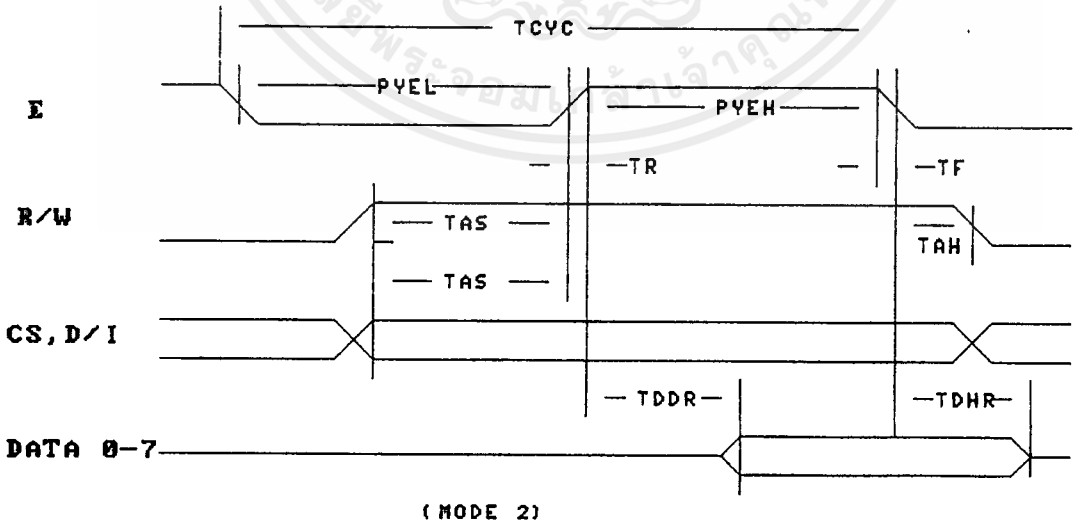
timmingdiagram3-90

1 write operation



(MODE 1)

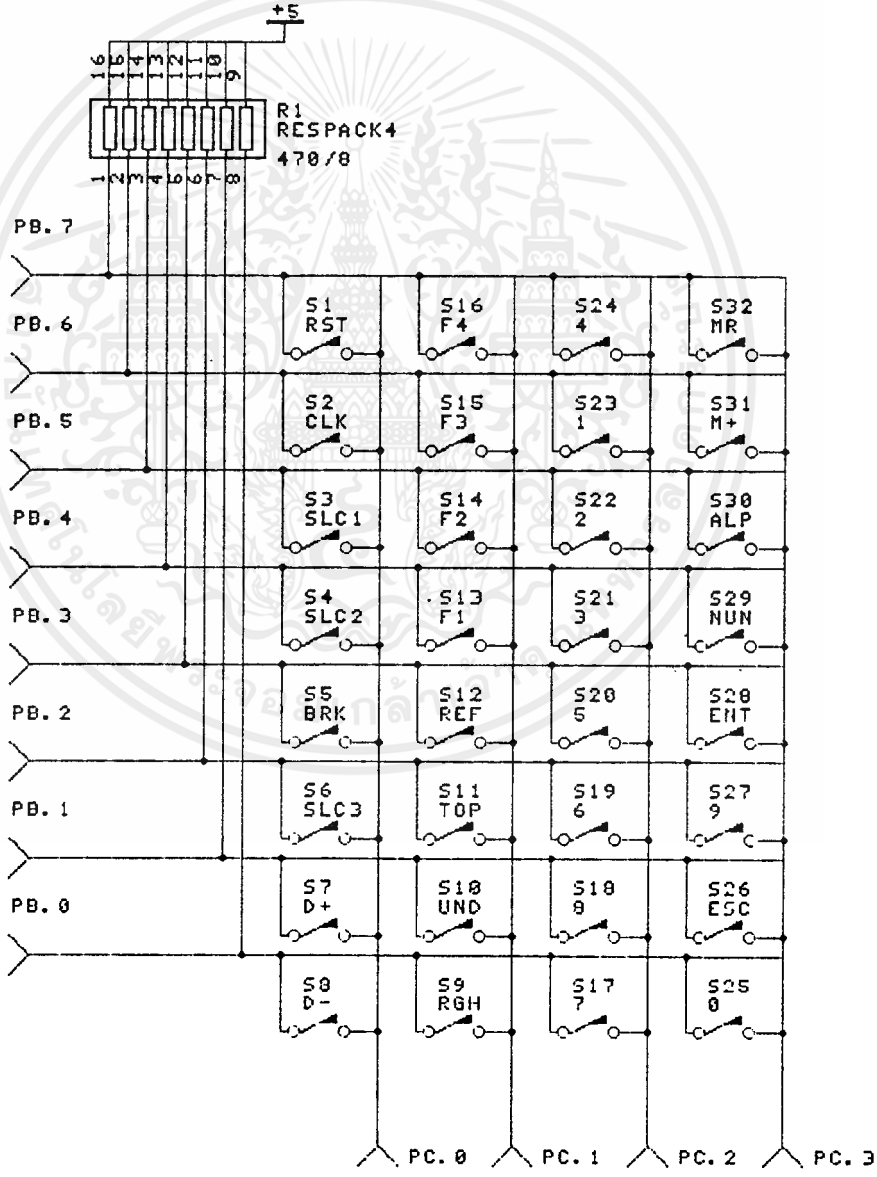
2 READ OPERATION



(MODE 2)

KEYBORD

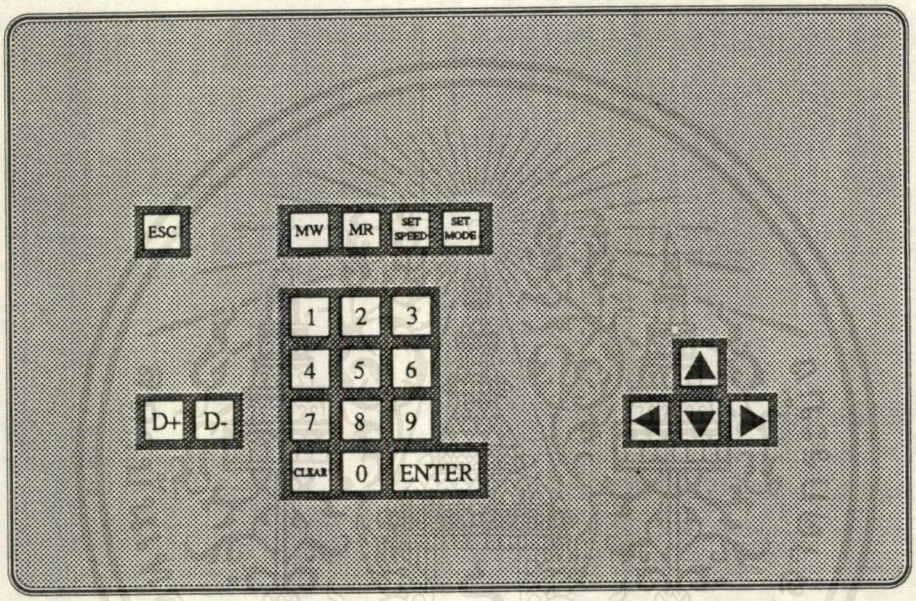
จะทำหน้าที่รับคำสั่งการทำงานจากผู้ใช้ ในกรณีที่ผู้ใช้ต้องการติดต่อกับ CPU โดยที่ CPU จะคอยเช็ค KEYBORD อยู่ตลอดเวลา ว่าในเวลาใดบ้างที่ผู้ใช้ KEY เข้ามา การเช็คนี้จะทำได้โดยการวน LOOP ซึ่งภายในส่วนของ KEYBORD นี้จะต้องมี REGISTER PULL UP ต่ออยู่ด้วย ซึ่ง RESISTER ตัวนี้จะทำหน้าที่รักษาเสถียรภาพของแรงดันให้คงที่อยู่ตลอดเวลา หรือให้ชาชอง SW ที่ต่ออยู่มี LOGIC เป็น "1"



รูปลักษณะของ KEY BORD ในโครงงาน

จากรูปลักษณะของ KEYBORD ที่ใช้ในโครงงานนี้บน KEY BORD นี้จะใช้สำหรับป้อนค่าต่าง ๆ เช่น ทิศทาง ตำแหน่ง ชื่อสถานีและเป็นการเซตสถานะให้กับ CPU เป็นต้น

ส่วนARROWKEYจะทำหน้าที่ในการเลือกใช้เมนูต่าง ๆที่กำหนดโดยเมนูที่ต้องการจะปรากฏบนจอ LCD



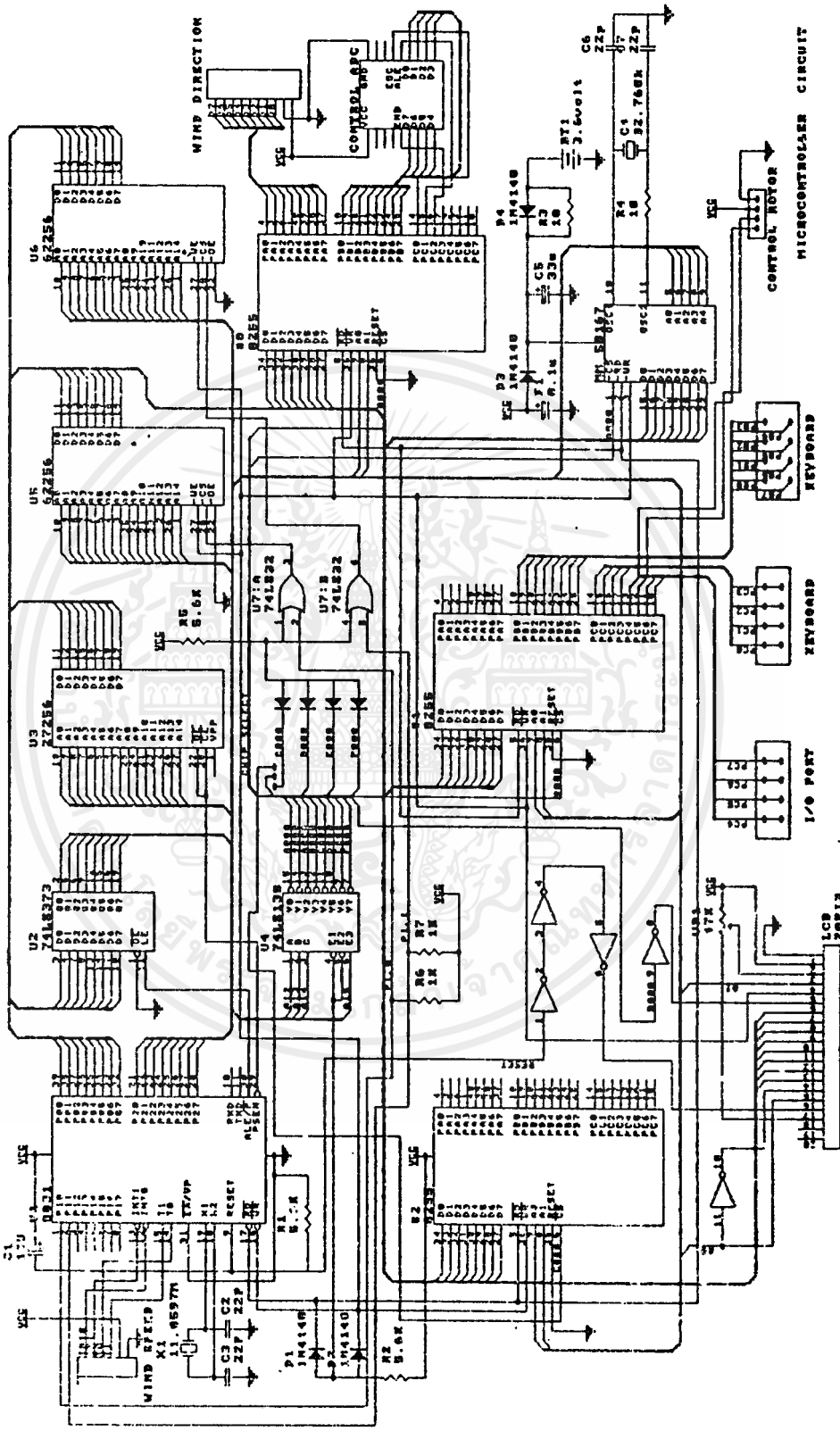
บรรณานุกรม

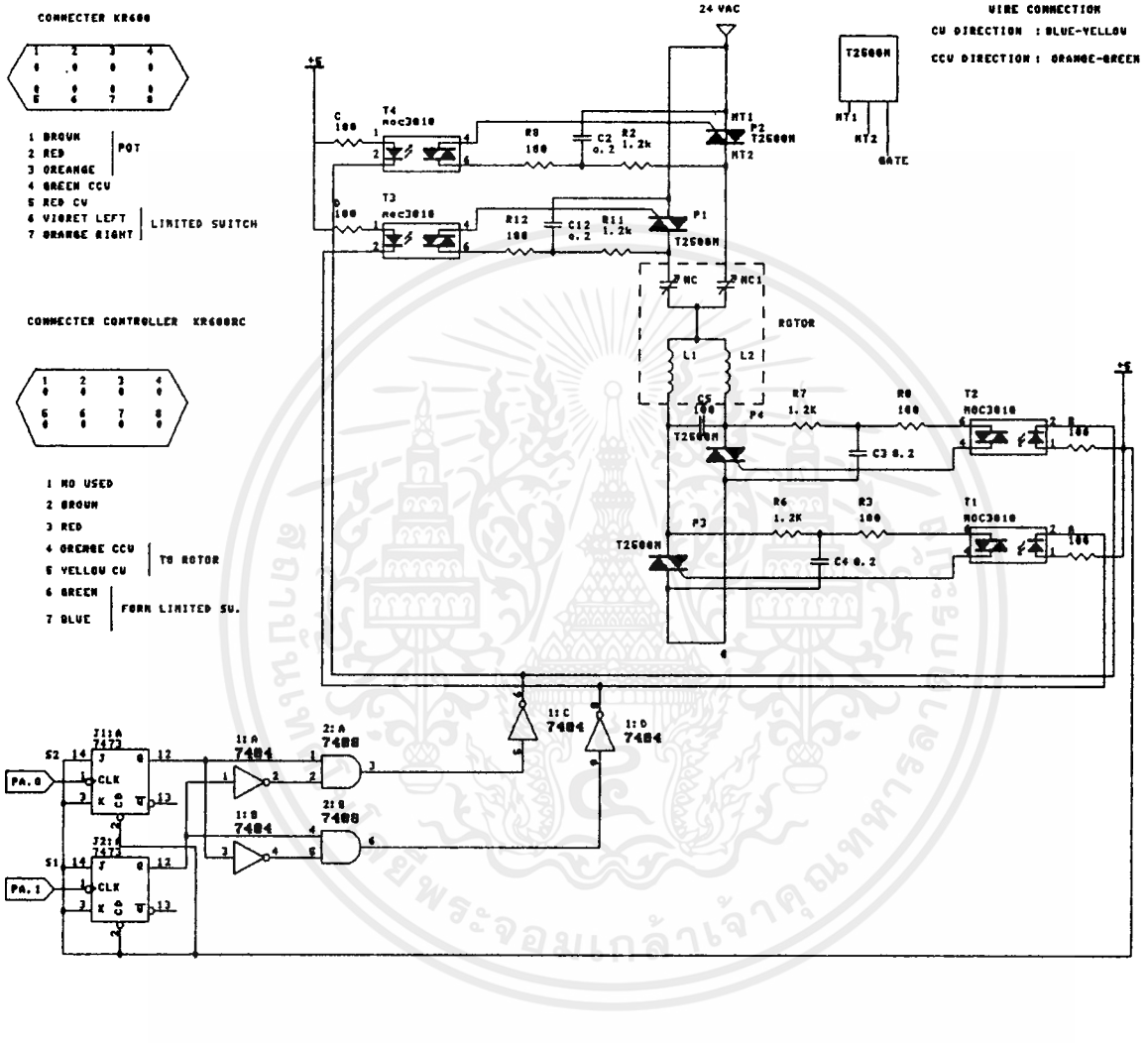
1. Avocet System,Inc. "AvCase,Avedit,and AVMAKE "
By J.W.Bray, PatrickJ.Adam,Peter Hart.
2. Borland International,Inc " Turbo Debuger "
3. Peter Norton computing,Inc. " The Norton Guides "
By Peter Norton.
4. Nelson Johnson ," Advance Graphics in C Programming an Techniqes " ,
Mcgraw-Hill ,U.S.A 1989.
5. Brain W.Kernighan and Dannis M.Ritchie ," The C Programming
Language " ,
Prentice-Hill ,New Jersy 1987.
6. Connelly J.A. ,"Analog Integrated Curcuits " ,
Willy ,New York 1975.
7. Herbeut Schidit ," Turbo C second Edition " ,
Mcgraw-hill ,U.S.A 1989.
8. Shiengold D.H. ,"Analog - Digital Conversion Hand Book " ,
Anal Devices,Norwood ,Mass 1972.
9. Intel the Microcomputer Company ," 8051 Microcontroller " ,
Intel the Microcomputer Company.
10. บุญเลิศ เอี่ยมทัศนา, ยืน ภู่วรวรรณ และ สมนึก คีรีโต " โปรแกรมคอมพิวเตอร์ภาษา " ,
บ.ซีเอ็ดยูเคชั่น จำกัด กรุงเทพฯ 2521.
11.บ.ซีเอ็ดยูเคชั่น จำกัด "เซมิคอนดักเตอร์อิเลคทรอนิคส์ " ฉบับที่ 95-97,103 และ 104 ,
บ.ซีเอ็ดยูเคชั่น จำกัด กรุงเทพฯ 2532-2533.

กิตติกรรมประกาศ

โครงงานชิ้นนี้สามารถสำเร็จลุล่วงไปด้วยดี ก็เนื่องจากได้รับความช่วยเหลือในด้านต่างๆ ทุกๆ ด้าน จากหลายๆฝ่ายด้วยกัน ทางด้านให้คำปรึกษานั้นขอขอบพระคุณท่านอาจารย์ ภากร หุตะสังกาส และอาจารย์ทุกท่านของภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม และต้องขอบคุณอาจารย์ วรวิทย์ สมหา ภาควิชาครุศาสตร์วิศวกรรม เป็นอย่างยิ่งสำหรับคำปรึกษาทางด้าน SOFTWARE และคอยให้กำลังใจมาโดยตลอด ส่วนทางด้านสถานที่ปฏิบัติงาน ขอขอบคุณภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรมและพี่ศาสที่ได้เอื้อเฟื้อสถานที่และเครื่องมือเครื่องใช้ต่างๆในการปฏิบัติงาน ขอขอบคุณสถาบันส่งเสริมการศึกษาและวิจัยเกี่ยวกับอิสลาม สำหรับความเอื้อเฟื้อเรียงพิมพ์คู่มือปริญญานิพนธ์ฉบับนี้ นอกจากนี้ขอขอบคุณทุกๆท่านที่มีส่วนเกี่ยวข้องและเพื่อนๆทุกคนที่คอยช่วยเหลือและให้กำลังใจตลอดมาจนโครงงานชิ้นนี้สำเร็จลุล่วงไปด้วยดี

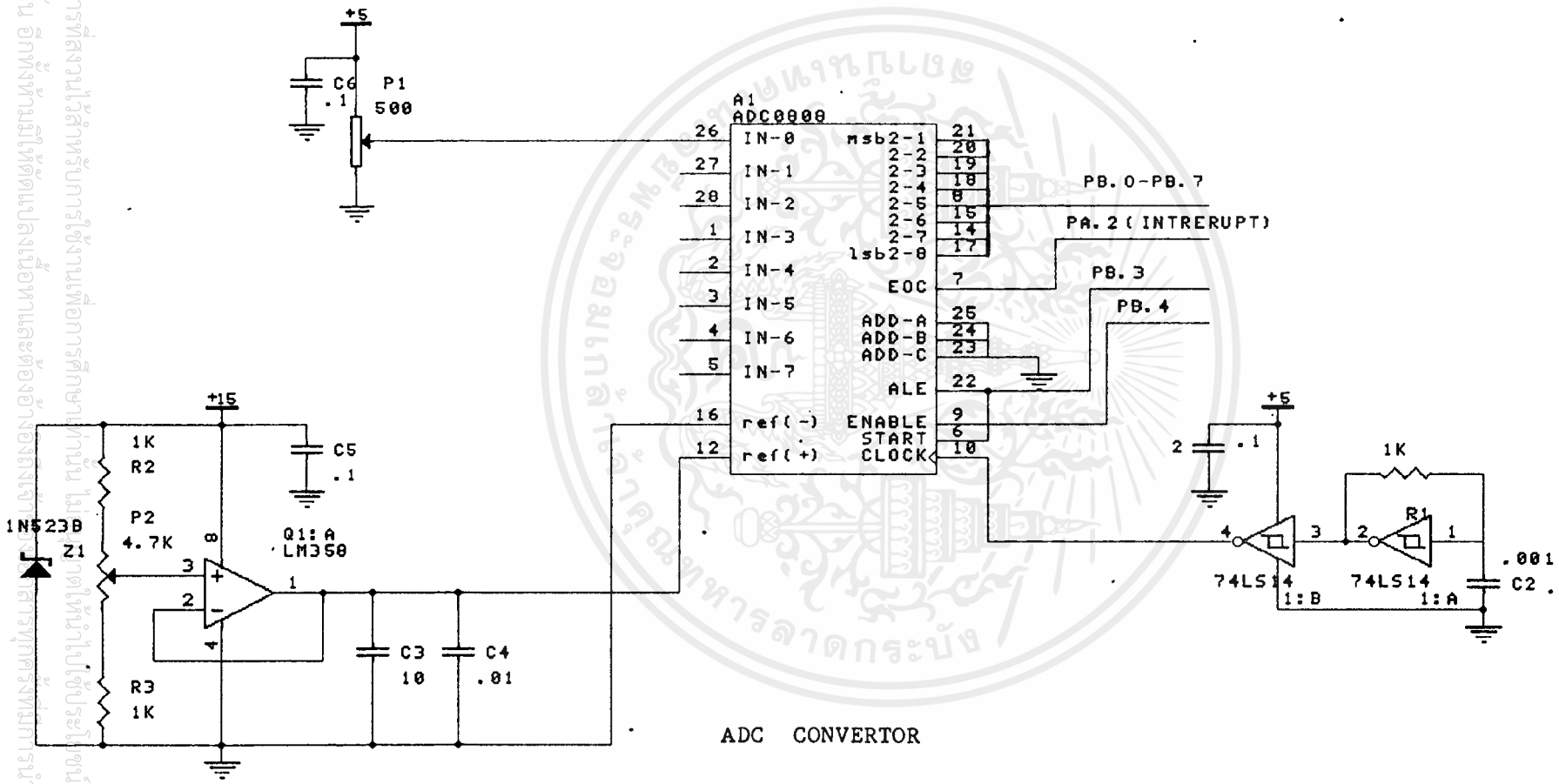
สุดท้ายนี้ขอขอบคุณทุกอย่างที่เป็นสถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง



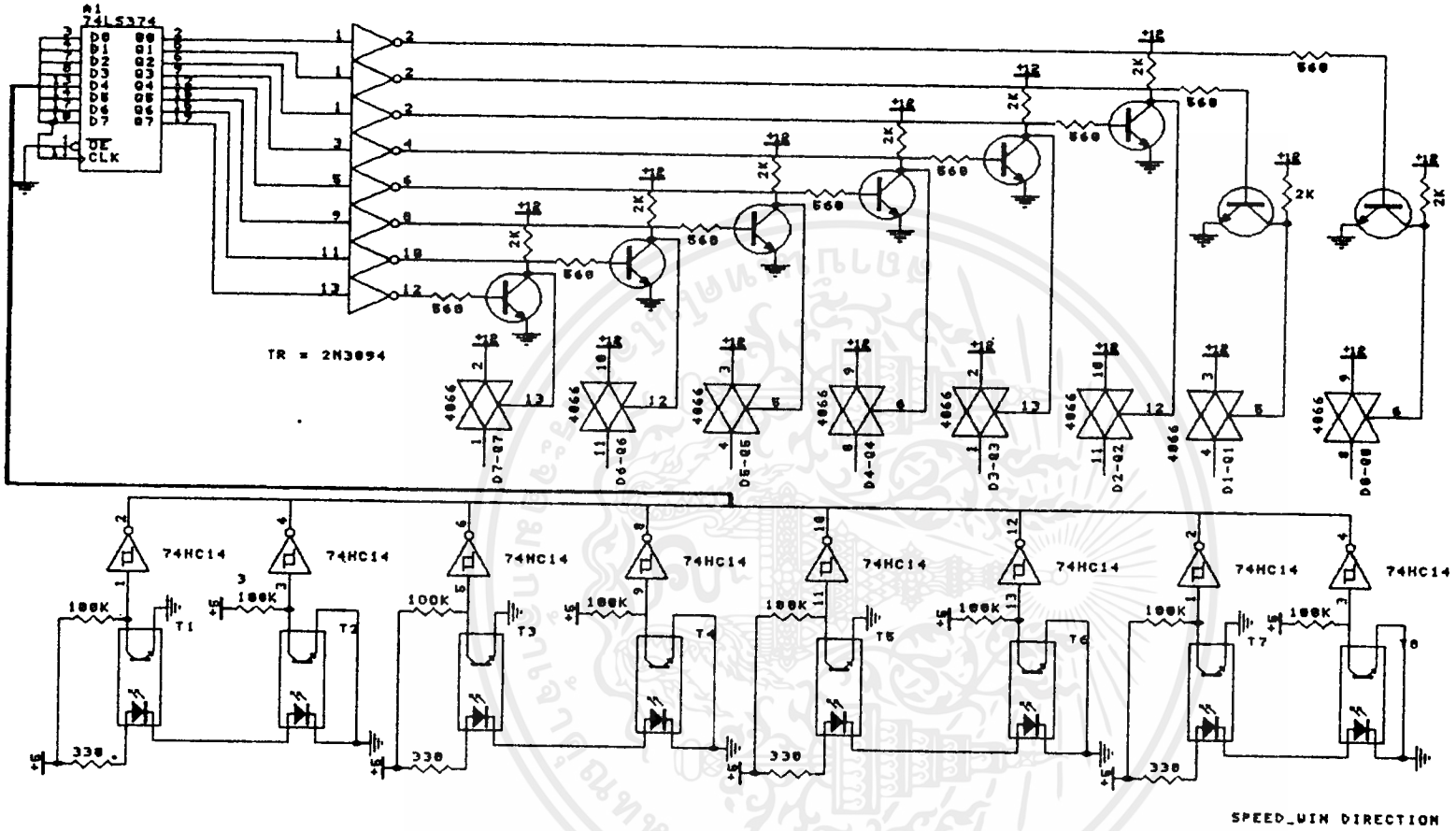


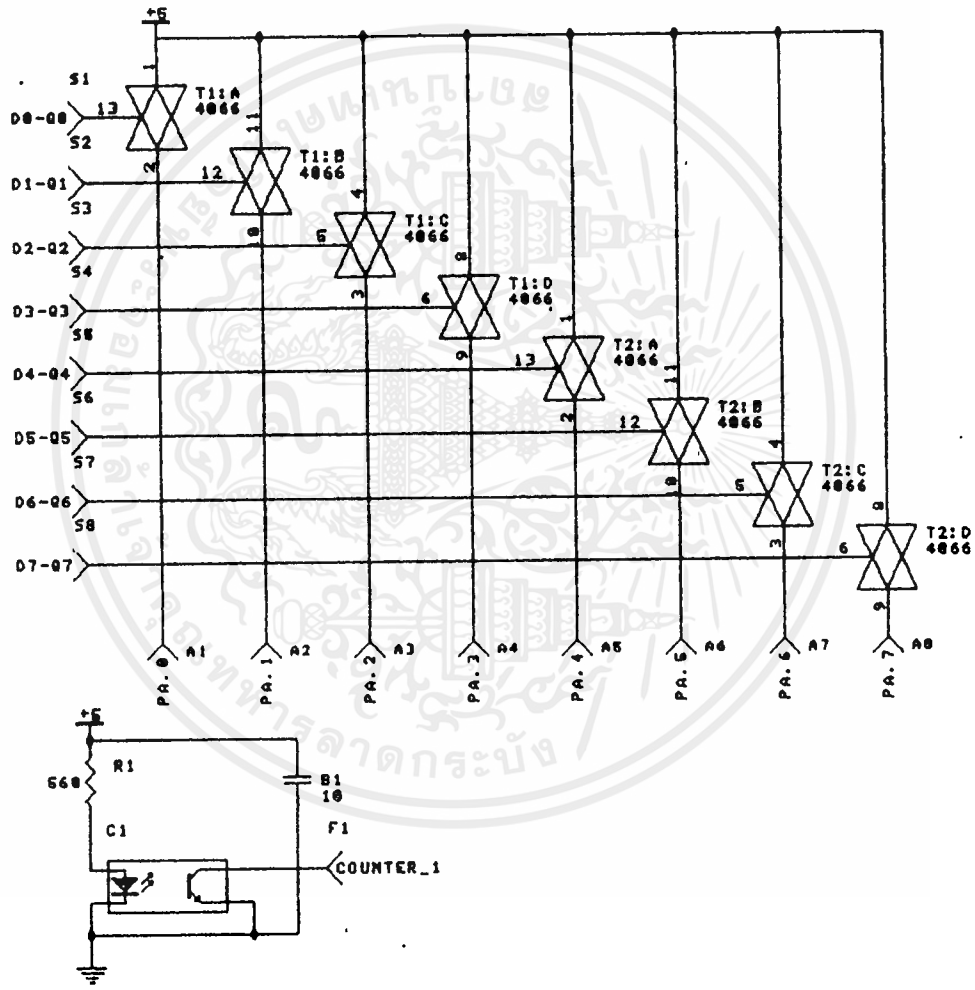
DRIVER ROTOR

PROPOTIONAL VOLTAGE FORM POTENTIOMETER



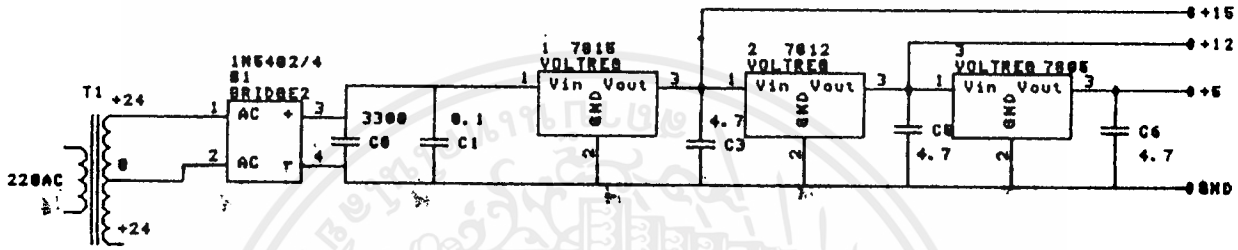
ADC CONVERTOR

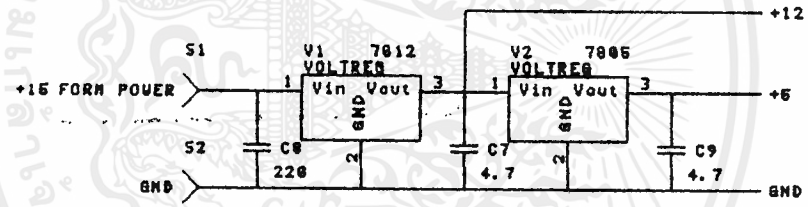

#1
74LS374
D0
D1
D2
D3
D4
D5
D6
D7
OE
CLK
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
TR = 2N3094
2K
560
4066
D7-07
D6-06
D5-05
D4-04
D3-03
D2-02
D1-01
D0-00
74HC14
100K
330
T1
T2
T3
T4
T5
T6
T7
T8
SPEED_WIN DIRECTION


+5
S1
D0-Q0
S2
T1:A
4066
D1-Q1
S3
T1:B
4066
D2-Q2
S4
T1:C
4066
D3-Q3
S5
T1:D
4066
D4-Q4
S6
T2:A
4066
D5-Q5
S7
T2:B
4066
D6-Q6
S8
T2:C
4066
D7-Q7
T2:D
4066
PA.0
A1
PA.1
A2
PA.2
A3
PA.3
A4
PA.4
A5
PA.5
A6
PA.6
A7
PA.7
A8
+5
R1
560
C1
B1
10
F1
COUNTER_1

WIN SPEED

MAIN POWER





VIN POWER

```
#include   <conio.h>
#include   <stdio.h>
#include   <8051.h>
#include   <stdlib.h>
#include   <intrpt.h>
#include   <math.h>

#define  ilcd_cs2 ((char *) 0xE000) /* LCD #2 */
#define  ilcd_cs1 ((char *) 0xE001) /* LCD #1 */
#define  dlcd_cs2 ((char *) 0xE002) /* DATA LCD #2 */
#define  dlcd_cs1 ((char *) 0xE003) /* DATA LCD #1 */
#define  port_a ((char *) 0xA000)   /* I/P KEYBOARD#8 */
#define  port_b ((char *) 0xA001)   /* I/P KEYBOARD#8 */
#define  port_c ((char *) 0xA002)   /* O/P KEYBOARD#
                                       DRIVE ROTOR#3 */
#define  port_i ((char *) 0xA003)   /* CONTROL PORT */
#define  port_a2 ((char *) 0x8000)  /* I/P WIND DIRECT */
#define  port_b2 ((char *) 0x8001)  /* I/P ADC */
#define  port_c2 ((char *) 0x8002)  /* O/P CONTROLL ADC */
#define  port_i2 ((char *) 0x8003)  /* CONTROL PORT */

#define  pi 3.141592654
#define  kms 8

static char A[]={0,0x3E,0x09,0x09,0x09,0x3E,0};
static char B[]={0,0x3F,0x25,0x25,0x25,0x1A,0};
static char C[]={0,0x1E,0x21,0x21,0x21,0x12,0};
static char D[]={0,0x3F,0x21,0x21,0x21,0x1E,0};
static char E[]={0,0x3F,0x25,0x25,0x25,0x21,0};
static char F[]={0,0x3F,0x05,0x05,0x05,0x01,0};
static char G[]={0,0x1E,0x21,0x29,0x29,0x3A,0};
static char H[]={0,0x3F,0x04,0x04,0x04,0x3F,0};
static char I[]={0,0,0x21,0x3F,0x21,0,0};
static char J[]={0,0x10,0x20,0x21,0x1F,0x01,0};
static char K[]={0,0x3F,0x08,0x0C,0x12,0x21,0};
static char L[]={0,0x3F,0x20,0x20,0x20,0,0};
static char M[]={0,0x3F,0x02,0x0C,0x02,0x3F,0};
static char N[]={0,0x3F,0x02,0x04,0x08,0x3F,0};
```

```
static char O[]={0,0x1E,0x21,0x21,0x21,0x1E,0};
static char P[]={0,0x3F,0x09,0x09,0x09,0x06,0};
static char Q[]={0,0x1E,0x21,0x29,0x11,0x2E,0};
static char R[]={0,0x3F,0x09,0x09,0x19,0x26,0};
static char S[]={0,0x12,0x25,0x25,0x25,0x19,0};
static char T[]={0,0x01,0x01,0x3F,0x01,0x01,0};
static char U[]={0,0x1F,0x20,0x20,0x20,0x1F,0};
static char V[]={0,0x0F,0x10,0x20,0x10,0x0F,0};
static char W[]={0,0x3F,0x10,0x0C,0x10,0x3F,0};
static char X[]={0,0x31,0x0A,0x04,0x0A,0x31,0};
static char Y[]={0,0x03,0x04,0x38,0x04,0x03,0};
static char Z[]={0,0x21,0x31,0x29,0x25,0x23,0};

static char Aa[]={0,0x10,0x2A,0x2A,0x2A,0x3C,0};
static char Bb[]={0,0x3F,0x28,0x24,0x24,0x18,0};
static char Cc[]={0,0x1C,0x22,0x22,0x22,0x10,0};
static char Dd[]={0,0x18,0x24,0x24,0x28,0x3F,0};
static char Ee[]={0,0x1C,0x2A,0x2A,0x2A,0x0C,0};
static char Ff[]={0,0,0x08,0x3E,0x09,0x02,0};
static char Gg[]={0,0x04,0x2A,0x2A,0x2A,0x1E,0};
static char Hh[]={0,0x3F,0x08,0x04,0x04,0x38,0};
static char Ii[]={0,0,0x24,0x3D,0x20,0,0};
static char Jj[]={0,0,0x10,0x20,0x20,0x1D,0};
static char Kk[]={0,0x3F,0x08,0x14,0x22,0,0};
static char Ll[]={0,0,0x21,0x3F,0x20,0,0};
static char Mm[]={0,0x3E,0x02,0x3C,0x02,0x3C,0};
static char Nn[]={0,0x3C,0x08,0x04,0x04,0x38,0};
static char Oo[]={0,0x1C,0x22,0x22,0x1C,0,0};
static char Pp[]={0,0,0x3E,0x0A,0x0A,0x04,0};
static char Qq[]={0,0x04,0x0A,0x0A,0x0A,0x3E,0};
static char Rr[]={0,0x3E,0x04,0x02,0x02,0,0};
static char Ss[]={0,0x24,0x2A,0x2A,0x12,0,0};
static char Tt[]={0,0,0x22,0x1F,0x22,0x10,0};
static char Uu[]={0,0x1E,0x20,0x20,0x10,0x3E,0};
static char Vv[]={0,0x06,0x18,0x20,0x18,0x06,0};
static char Ww[]={0,0x1E,0x20,0x18,0x20,0x1E,0};
static char Xx[]={0,0x22,0x12,0x1C,0x24,0x22,0};
static char Yy[]={0,0x22,0x24,0x18,0x08,0x06,0};
static char Zz[]={0,0x22,0x32,0x2A,0x26,0x22,0};
```

```
static char O0[]={0,0x1E,0x29,0x25,0x23,0x1E,0xFF};
static char O1[]={0,0,0x22,0x3F,0x20,0,0xFF};
static char O2[]={0,0x22,0x31,0x29,0x25,0x22,0xFF};
static char O3[]={0,0x11,0x21,0x25,0x25,0x1B,0xFF};
static char O4[]={0,0x18,0x14,0x12,0x3F,0x10,0xFF};
static char O5[]={0,0x17,0x25,0x25,0x25,0x19,0xFF};
static char O6[]={0,0x1E,0x25,0x25,0x25,0x18,0xFF};
static char O7[]={0,0x01,0x01,0x39,0x05,0x03,0xFF};
static char O8[]={0,0x1A,0x25,0x25,0x25,0x1A,0xFF};
static char O9[]={0,0x06,0x29,0x29,0x29,0x1E,0xFF};

static char B0[]={0,0,0xC0,0xF0,0xF0,0x1C,0x1C,0x07,0x07,
                  0x03,0x03,0x03,0x03,0x07,0x07,0x1C,0x1C,
                  0xF0,0xF0,0xC0,0,0,0,0,0xFF,0xFF,0xFF,
                  0,0,0,0,0,0,0,0,0,0,0,0,0xFF,0xFF,0xFF,
                  0,0,0,0,0x03,0x0F,0x0F,0x38,0x38,0xE0,
                  0xE0,0xC0,0xC0,0xC0,0xC0,0xE0,0xE0,0x38,
                  0x38,0x0F,0x0F,0x03,0,0};
static char B1[]={0,0,0,0,0x20,0x20,0x30,0x18,0x0C,0x0E,
                  0xFF,0xFF,0xFF,0,0,0,0,0,0,0,0,0,0,0,0,
                  0,0,0,0,0,0,0,0xFF,0xFF,0xFF,0,0,0,0,0,
                  0,0,0,0,0,0,0,0,0xC0,0xC0,0xC0,0xC0,0xC0,
                  0xC0,0xFF,0xFF,0xFF,0xC0,0xC0,0xC0,0xC0,
                  0xC0,0xC0,0xC0,0,0,0};
static char B2[]={0,0,0x38,0x78,0xFC,0xEC,0xE6,0x07,0x03,
                  0x03,0x03,0x03,0x03,0x03,0x03,0x87,0x86,
                  0xFC,0xF8,0xF8,0,0,0,0,0,0,0,0x80,0x80,
                  0xC0,0xC0,0x60,0x30,0x38,0x1C,0x1E,0x0F,
                  0x07,0x03,0x03,0x01,0x01,0,0,0,0,0xF8,
                  0xFE,0xFF,0xEF,0xE3,0xE1,0xE0,0xE0,0xE0,
                  0xE0,0xE0,0xE0,0xE0,0xE0,0xE0,0xE0,0xFC,
                  0xFC,0,0};
static char B3[]={0,0,0x38,0x78,0x7C,0x3C,0x3C,0x03,0x03,
                  0x03,0x03,0x03,0x03,0,0,0,0,0,0,0,0,0x18,
                  0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x24,
                  0x24,0xC3,0xC3,0x81,0,0,0,0,0x1C,0x1E,
                  0x3E,0x3C,0x3C,0xC0,0xC0,0xC0,0xC0,0xC0,
                  0xC0,0xC0,0xC0,0x20,0x20,0x1F,0x1F,0x0F,
```

```
                    0,0};
static char B4[]={0,0,0,0,0,0,0x80,0x80,0x70,0x70,0x0E,0x0E,
                    0x01,0x01,0xFF,0xFF,0xFF,0,0,0,0,0,0,0,
                    0xE0,0xE0,0x1C,0x1C,0x03,0x03,0,0,0,0,0,
                    0,0xFF,0xFF,0xFF,0,0,0,0,0,0,0,0x03,0x03,
                    0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03,
                    0x03,0x03,0xFF,0xFF,0xFF,0x03,0x03,0x03,
                    0,0};
static char B5[]={0,0,0xFF,0xFF,0x07,0x07,0x07,0x07,0x87,
                    0x87,0x87,0x87,0x87,0x87,0x87,0x07,0x07,
                    0x07,0x1F,0x1F,0,0,0,0,0x0F,0x0F,0x86,
                    0x86,0x03,0x03,0x03,0x01,0x01,0x01,0x01,
                    0x03,0x03,0x06,0x0E,0xFC,0xF0,0xE0,0,0,
                    0,0,0x1E,0x3F,0x7F,0x7F,0xE7,0xC2,0xC0,
                    0xC0,0xC0,0xC0,0xC0,0xC0,0xE0,0x70,0x30,
                    0x3F,0x0F,0x03,0,0};
static char B6[]={0,0,0xE0,0xE0,0xF8,0x18,0x1E,0x06,0x07,
                    0x03,0x03,0x03,0x03,0x03,0x06,0x7E,0xFE,
                    0xF8,0xF8,0x60,0,0,0,0,0xFF,0xFF,0xFF,
                    0x60,0x60,0x18,0x18,0x0C,0x0C,0x0C,0x0C,
                    0x0C,0x18,0x18,0x60,0x61,0x80,0x80,0,0,
                    0,0,0x03,0x1F,0x1F,0x18,0x60,0x60,0xE0,
                    0xC0,0xC0,0xC0,0xC0,0xC0,0xC0,0x30,0x30,
                    0x3F,0x0F,0x0F,0,0};
static char B7[]={0,0,0x7F,0x7F,0x07,0x07,0x07,0x07,0x07,
                    0x07,0x07,0x07,0x07,0x07,0xC7,0xF7,0x3F,
                    0x1F,0x1F,0x07,0,0,0,0,0,0,0,0,0,0,0,
                    0xC0,0xF0,0x3C,0x0F,0x03,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0xFF,0xFF,0xFF,0,0,0,
                    0,0,0,0,0,0,0};
static char B8[]={0,0,0,0xF0,0xFC,0x0C,0x0C,0x03,0x03,0x03,
                    0x03,0x03,0x03,0x03,0x03,0x0C,0x0C,0xFC,
                    0xF0,0,0,0,0,0,0x80,0xE1,0xE7,0x7E,0x1E,
                    0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,
                    0x1E,0x7E,0xE7,0xE1,0x80,0,0,0,0,0x07,
                    0x0F,0x3F,0x30,0xE0,0xC0,0xC0,0xC0,0xC0,
                    0xC0,0xC0,0xC0,0xC0,0xE0,0x30,0x3F,0x0F,
                    0x07,0,0};
static char B9[]={0,0,0xF0,0xF0,0xFC,0x0C,0x0C,0x03,0x03,
```

```
                    0x03,0x03,0x03,0x03,0x07,0x06,0x06,0x18,
                    0xF8,0xF8,0xC0,0,0,0,0,0x01,0x01,0x86,
                    0x06,0x18,0x18,0x30,0x30,0x30,0x30,0x30,
                    0x18,0x18,0x06,0x06,0xFF,0xFF,0xFF,0,0,
                    0,0,0x06,0x1F,0x1F,0x7F,0x7E,0x60,0xC0,
                    0xC0,0xC0,0xC0,0xC0,0xE0,0x60,0x78,0x18,
                    0x1F,0x07,0x07,0,0};

static char O_[]={0,0x20,0x20,0x20,0x20,0x20,0};
static char BK[]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0};
static char CO[]={0,0,0,0x05,0x03,0};
static char ARR[]={0x08,0x08,0x08,0x22,0x14,0x08};
static char CUR[]={0x1C,0x1C,0x7F,0x3E,0x1C,0x08};
static char SSS[]={0,0,0,0,0,0};

static char BS[]={0,0x20,0x10,0x08,0x04,0x02};
static char XX1[]={0,0xFE,0x02,0x02,0x02,0x02};
static char XX2[]={0x02,0x02,0x02,0x02,0x02,0x02};
static char XX3[]={0x02,0x02,0x02,0x02,0xFE,0};
static char XX4[]={0,0x7F,0x40,0x40,0x40,0x40};
static char XX5[]={0x40,0x40,0x40,0x40,0x40,0x40};
static char XX6[]={0x40,0x40,0x40,0x40,0x7F,0};
static char XX7[]={0,0xFF,0,0,0,0};
static char XX8[]={0,0,0,0,0xFF,0};
static char XX9[]={0,0,0,0,0x03,0x02};
static char XX0[]={0x02,0x03,0,0,0,0};
static char YY0[]={0x01,0x7E,0x42,0x42,0xFE,0x40};
static char YY1[]={0x40,0xFE,0x42,0x42,0x7E,0x01};
static char YY2[]={0x80,0x7E,0x42,0x42,0x7F,0x02};
static char YY3[]={0x02,0x7F,0x42,0x42,0x7E,0x80};

static char KMIT0[]={0,0,0,0,0,0,0,0,0,0,0,0x40,0x40,0x80,
                     0x88,0x90,0x20,0x40,0,0x3E,0,0x40,
                     0x20,0x90,0x88,0x80,0x40,0x40,0,0,0,
                     0,0,0,0,0,0,0,0,0};
static char KMIT1[]={0,0,0,0,0,0,0,0x10,0x10,0x88,0x48,
                     0x44,0x24,0x12,0x12,0xC8,0x20,0,0xE4,
                     0x1A,0xE4,0,0x20,0xC8,0x12,0x12,0x24,
                     0x44,0x48,0x88,0x10,0x10,0,0,0,0,0,0,
```

```
                    0,0,0,0,0,0,0,0,0};
static char KMIT2[]={0,0,0,0x08,0x04,0x04,0x02,0x01,0x01,
                    0x80,0x40,0x30,0x08,0x06,0x01,0,0,0,
                    0x57,0xA8,0x57,0,0,0,0x01,0x06,0x08,
                    0x30,0x40,0x80,0x01,0x01,0x02,0x04,
                    0x04,0x08,0,0,0,0,0,0,0,0,0,0,0,0,0};
static char KMIT3[]={0,0,0,0,0x80,0x60,0x10,0x0C,0x02,0x01,
                    0,0,0,0,0,0x80,0x40,0xEA,0x75,0xEA,
                    0x75,0xEA,0x40,0x80,0,0,0,0,0,0x01,
                    0x02,0x0C,0x10,0x60,0x80,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,0,0,0};
static char KMIT4[]={0x20,0x18,0x04,0x03,0,0,0,0,0,0,0x80,
                    0x40,0xA0,0x54,0xAA,0x59,0xAF,0x59,
                    0xAF,0x59,0xAF,0x59,0xAF,0x59,0xAA,
                    0x54,0xA0,0x40,0x80,0,0,0,0,0,0,0x03,
                    0x04,0x18,0x20,0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0};
static char KMIT5[]={0,0,0,0,0,0xD0,0x20,0x40,0xA4,0xDA,
                    0x51,0xDB,0x51,0xDB,0x51,0xDB,0x51,
                    0xDB,0x55,0xDF,0x55,0xDB,0x51,0xDB,
                    0x51,0xDB,0x51,0xDB,0x51,0xDA,0xA4,
                    0x40,0x20,0xD0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0};
static char KMIT6[]={0,0,0x20,0x40,0xA0,0x20,0x41,0x82,
                    0x0D,0x70,0x8B,0x55,0xB9,0x05,0xC3,
                    0x31,0x11,0x91,0xC9,0xAB,0x8D,0x81,
                    0x01,0x01,0x03,0x05,0xB9,0x55,0x8B,
                    0x70,0x0D,0x82,0x41,0x20,0xA0,0x40,
                    0x20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                    0,0,0};
static char KMIT7[]={0,0,0,0,0,0x03,0x05,0x05,0x0A,0x04,
                    0x0B,0x04,0x05,0x02,0,0x01,0x01,0,0,
                    0,0,0,0x01,0x01,0,0x02,0x05,0x04,
                    0x0B,0x04,0x0A,0x05,0x05,0x03,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,0,0};
static char OSB0[]={0x04,0x82,0x82,0x02,0x02,0x02,0x02,
                   0x02,0x04,0xF8,0,0,0,0x04,0x02,0x82,
                   0x82,0x82,0x82,0x8A,0x04,0xFB,0,0,0,
                   0x04,0x0A,0x06,0xFC,0,0,0,0,0,0,0,0,
```

```
                    0xFE,0,0,0,0,0,0,0,0,0,0,0};
static char OSB1[]={0x3F,0x41,0x82,0x81,0x80,0x80,0x80,
                    0x80,0x40,0x3F,0,0,0,0x7E,0xC1,0xA0,
                    0x40,0,0,0,0x01,0xFF,0,0,0,0,0,0,0x3F,
                    0x40,0x80,0x80,0x80,0x80,0x80,0x80,
                    0x40,0x3F,0,0x80,0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0};
static char OSB2[]={0xF0,0x0C,0x02,0x82,0x61,0xA1,0x41,
                    0x01,0x02,0x02,0x04,0xF8,0,0,0,0xC2,
                    0x61,0xA1,0x41,0x01,0x01,0x01,0x02,
                    0xFC,0,0,0,0x02,0x05,0x03,0xFE,0,0,
                    0,0,0,0xFF,0,0,0,0,0,0,0,0,0,0,0,0};
static char OSB3[]={0,0x03,0x1E,0x01,0,0,0,0,0,0,0,0x1F,
                    0,0,0,0x07,0x08,0x10,0x10,0x10,0x10,
                    0x10,0x08,0x07,0,0,0,0x0C,0x12,0x12,
                    0x0F,0x02,0x04,0x04,0x08,0x10,0x0F,0,
                    0,0x0C,0x12,0x12,0x0C,0xF8,0,0,0,0,0,
                   0,0,0,0,0,0,0,0};

static char *title0[]={"{,,,,,,,,,,,,,,,,,,,!$",
                       "&   KING'MOUNGKUT   .$",
                       "&     INSTITUTE     .$",
                       "&        OF         .$",
                       "&     TECHNOLOGY    .$",
                       "&     LADGRABANG    .$",
                       "&                   .$",
                       "}}}}}}}}}}}}}}}}}}}~%"};
static char *title1[]={"()))))))))))))))))))*$",
                       ".                    &$",
                       ".     DIRECTION      &$",
                       ".      ANTENNA       &$",
                       ".      'YAKI'        &$",
                       ".     CONTROLOR      &$",
                       ".                    &$",
                       "+,,,,,,,,,,,,,,,,,,,-%"};
static char *title2[]={"      PRODUCT         $",
                       "        BY            $",
                       "NATTAPONG  PONGPANICH$",
                       "NONTARAT   SIRIRAT   $",
```

```
                        "RACHASAK   RATTATHAM $",
                        "SAKOL      POUNGLAHE $",
                        "TOTSAPHORN SATONVISIT$",
                        "                1993 %"};

static char *note1[]={"          &          $",
                      "          &          $",
                      "          &          $",
                      "          :)))Rotor))$",
                      "           )))Wind)))$",
                      "          &Direct    $",
                      "          &speed     $",
                      "          :))))))))))%"};
static char *memory1[]={" &MEMORY. $",
                        " :)WRITE; $",
                        "Memory    $",
                        " POSITION $",
                        "          $",
                        " STATION$",
                        "          %"};
static char *memory2[]={" &MEMORY. $",
                        " :))READ; $",
                        "Memory    $",
                        " POSITION $",
                        "          $",
                        " STATION$",
                        "          %"};
static char *protect1[]={" &PROTECT.$",
                         " :)))))));$",
                         "          $",
                         "  SET     $",
                         "     Km/h$",
                         "          $",
                         "          %"};
static char *manaul1[]= {"          $",
                         "          $",
                         " Protect  $",
                         "    Km/h  $",
                         "          $",
```

```
                        "          $",
                        "        $",
                        "          %"};
static char *MANUAL[]={"MANUALmode%"};
static char *AUTO[]=  {" AUTO mode%"};
int   x,y;
int   channal;
int   *store,*print_big,*read_num,*wind_test;
int   wind_speed,wind_direct;
int   inter1,inter2,count1,count2;
int   alarm,name_busy;
int   mem_busy,protect;
int   on_name,move_tomem;
unsigned char position,position_adc,*mem_position;
char  busy,namem[12],mode;
char  *many,*lcd;
unsigned char *memory_busy;

/* ######################## */
/* ######## MAIN ######## */
/* ######################## */

main()

{
  Variable();
  Rams();
  On_lcd();
  Clr_s();
  Init_port();
  Set_interupt();
  Demo();
  mode=0;

  for(;;)
  {
      On_lcd();
      Write_wind();
      Write_lcd_wind();
```

```
        Control_rotor();
        Print_rotor();
        Write_lcd_rotor();
        Select_mode();
        Write_mode();
        Select_protect();
        Write_lcd();
        Select_memory();
        Write_lcd();
        Memory();
        Write_lcd();
        Esc();
        Write_lcd();
    }
}
/* ######################### */
/* ##### END OF MAIN ##### */
/* ######################### */

/* #### INTERRUPT TIMMER 0  FUNCTION #### */
void interrupt time0(void)
{
    inter1++;          /* #####  1 Sec ##### */
    if(inter1==4)
    {
        inter2=count2;
        if(inter2<*(memory_busy+3))
            Inter2();
        else
            Inter1();
        count2=0;
        count1=0;
        inter1=0;
    }
    TH0=0;
    TL0=0;
    TCON&=0xD0;
}
```

```
/* ###### INTER2 ###### */
Inter2()
{
            alarm=0;
            P1=0;
}

/* ###### INTER1 ###### */
Inter1()
{
            alarm=1;
            if(mode==0)
                P1=0xFF;
              else
                P1=0;
}

/* #### INTERRUPT COUNTER 1 FUNCTION #### */
void interrupt count0(void)
{
    count1++;
    if(count1>=kms) /* ### number of pulse = 1 Km ### */
    {
       count2++;
       count1=0;
    }
    TH1=0xFF;
    TL1=0xFF;
    TCON&=0x70;
}

/* ###### ESCAPE ###### */
Esc()
{
    char m;
    m=Keyb();
    if(m==8)
    {
      Variable();
```

```
            What();
            Clr_key();
        }
    }

    /* ###### Demo ###### */
    Demo()
    {
        Clr_s();
        Set_xy(0,15);
        Writew(KMIT0,40);
        Set_xy(1,15);
        Writew(KMIT1,40);
        Set_xy(2,15);
        Writew(KMIT2,40);
        Set_xy(3,15);
        Writew(KMIT3,40);
        Set_xy(4,15);
        Writew(KMIT4,40);
        Set_xy(5,15);
        Writew(KMIT5,40);
        Set_xy(6,15);
        Writew(KMIT6,40);
        Set_xy(7,15);
        Writew(KMIT7,40);
        Set_xy(2,75);
        Writew(OSB0,45);
        Set_xy(3,75);
        Writew(OSB1,45);
        Set_xy(5,70);
        Writew(OSB2,50);
        Set_xy(6,70);
        Writew(OSB3,50);
        Print_all();
        Delay(30000);
        Clr_s();
        Set_xy(0,0);
        Printw(title0);
        Print_all();
```

```
        Delay(30000);
        Clr_s();
        Set_xy(0,0);
        Printw(title1);
        Print_all();
        Delay(30000);
        Clr_s();
        Set_xy(0,0);
        Printw(title2);
        Print_all();
        Delay(30000);
        Delay(30000);
        Delay(30000);
        busy=0x0F;
        Set_xy(0,0);
        Printw(note1);
        Set_xy(0,0);
        Printw(manaul1);
        Set_xy(3,4);
        Print_number(*(memory_busy+3));
        Set_xy(0,0);
        Printw(AUTO);
        Line(33,64,33,68);
        Line(33,64,39,64);
        Printrotor();
}

/* ###### SET INTERUPT ###### */
Set_interupt()
{
    TMOD=0x51;
    IE=0x8A;
    TH0=0;
    TL0=0;
    TH1=0xFF;
    TL1=0xFF;
    TCON=0x50;
    set_vector (TIMER0,time0);
    set_vector (TIMER1,count0);
```

```
}

/* ###### RAMS ###### */
Rams()
{
    mamy=malloc(1200);
    mem_position=malloc(100);
    lcd=malloc(1024);
    memory_busy=malloc(4);
    store=malloc(11);
    print_big=malloc(5);
    read_num=malloc(10);
    wind_test=malloc(2);
}

/* ###### VARIABLE ###### */
Variable()
{
    P1=0;
    *(read_num+6)=0;
    on_name=0;
    move_tomem=0;
    mem_busy=0;
    *memory_busy=0;
    name_busy=0;
    *(memory_busy+2)=0;
    *(wind_test+1)=12;
    *wind_test=12;
    inter1=0;
    count1=0;
    channal=0;
    alarm=0;
    protect=0;
}

/* ###### PRINT ALL ###### */
Print_all()
{
     busy=0x0F;
```

```
        Write_lcd_wind();
        Write_lcd();
        Write_lcd_rotor();
        Write_mode();
    }


    /* ###### SELECT MODE ##### */
    Select_mode()
    {
        unsigned a,b;
        int w;

        a=Keyb();

        if(a==29)
        {
            mode++;
            w=mode%2;
            if(w==0)
            {
                Set_xy(0,0);
                Printw(AUTO);
            }
            else if(w==1)
            {
                Set_xy(0,0);
                Printw(MANUAL);
            }
            mode=w;
            Clr_key();
            busy!=0x08;
        }
    }


    /* ##### CONTROL ROTOR ##### */
    Control_rotor()
    {
```

```
        unsigned char a,b;
        int c;
        *port_c=0x0E;
        a=*port_b;
        if(a==0x7F||a==0xBF||(position_adc!=wind_direct&&
           mode==0&&alarm==1))
        {
             if(a==0x7F||(position_adc<wind_direct&&
                mode==0&&alarm==1))
             {
                  *port_c=0x80;
                  *port_c=0x10;
             }
             if(a==0xBF||(position_adc>wind_direct&&
                mode==0&&alarm==1))
             {
                  *port_c=0x80;
                  *port_c=0x20;
             }
             if(position_adc<wind_direct&&mode==0&&alarm==1)
             {
                  *port_c=0x80;
                  *port_c=0x10;
             }
             move_tomem=0;
        }
        else if(move_tomem==0)
             *port_c=0x80;
}


/* ###### MEMORY PART ###### */
Memory()
{
     unsigned char a,b;
     int w,m;
     m=Keyb();
     if(m==32||*memory_busy==1)
     {
```

```
            if(m==32)
            {
                Set_xy(1,0);
                Printw(memory1);
                Clr_key();
                busy!=0x01;
                *(read_num+6)=0;
                mem_busy=0;
            }
            on_name=1;
            channal=Read_number(3,42);
            if(channal==222)
                *memory_busy=1;
            else
            {
                *memory_busy=0;
                name_busy=1;
                *(memory_busy+2)=0;
                on_name=0;
            }
        }
        if(name_busy==1)
        {
            Name_station();
            if(*(memory_busy+2)==0)
            {
                What();
                name_busy=0;
            }
        }
    }


    /* ###### What ###### */
    What()
    {
                Set_xy(0,0);
                Printw(manaul1);
                Set_xy(3,6);
```

```
                Print_number(*(memory_busy+3));
                if(mode==0)
                {
                     Set_xy(0,0);
                     Printw(AUTO);
                }
                else
                {
                     Set_xy(0,0);
                     Printw(MANUAL);
                }
                 busy!=0x01;
    }


    /* ##### MEMORY ##### */
    Select_memory()
    {
         unsigned char a,b;
         int w,m;
         m=Keyb();
         if(m==31||mem_busy==1)
         {
              if(m==31)
              {
                   Set_xy(1,0);
                   Printw(memory2);
                   Clr_key();
                   *(read_num+6)=0;
              }
              on_name=1;
              channal=Read_number(3,42);
              Koko();
              busy!=0x01;
         }
         Jeje();
    }

    /* ###### JEJE ###### */
```

```
Jeje()
{
     if(move_tomem==1)
     {
          if(position_adc<position)
               *port_c=0x10;
          else if(position_adc>position)
               *port_c=0x20;
          else /* if(position_adc==position) */
          {
               *port_c=0x80;
               move_tomem=0;
          }
     }
}

/* ###### KOKO ###### */
Koko()
{
          if(channal==222)
            mem_busy=1;
          else
          {
             on_name=0;
             mem_busy=0;
             What();
             move_tomem=1;
             position=*(mem_position+channal);
          }
}


/* ###### SELECT VALUME PROTECT ###### */
Select_protect()
{
     unsigned char a,b;
     int w,q,m;
     m=Keyb();
     if(m==30||protect!=0)
```

```
        {
            if(m==30)
            {
                Set_xy(1,0);
                Printw(protect1);
                Clr_key();
                *(read_num)=0;
            }
            on_name=0;
            q=Read_number(5,10);
            if(q==222)
                protect=1;
            else
            {
                *(memory_busy+3)=q;
                protect=0;
                What();
            }
            busy!=0x01;
        }
}

/* ###### READ NUMBER FROM KEYBOARD ###### */
int Read_number(a,b)
int a,b;
{
    int numb,i,j,k,l,m,o,n,r,s,ug;
    char ma;
    char c,d;
    unsigned char w,q;

    numb=0;
    if(*(read_num+6)==0)
    {
        l=1;
        o=0;
        n=0;
        Set_xy(a,b);
        Print_number(o);
```

```
}
else
{
    l=*read_num;
    o=*(read_num+1);
    n=*(read_num+2);
    r=*(read_num+3);
    s=*(read_num+4);
}
*(read_num+6)=1;
m=Keyb();
if(m<=28&&m>=17)
{
     o*=10;
     if(m==28)
     {
        *(read_num+6)=0;
        o/=10;
        n=0;
     }
     else if(m==25)
     {
        o=0;
        n=0;
     }
     switch(m)
     {
       case 26:n=0;
               break;
       case 23:n=1;
               break;
       case 22:n=2;
               break;
       case 21:n=3;
               break;
       case 24:n=4;
               break;
       case 20:n=5;
               break;
```

```
            case 19:n=6;
                    break;
            case 17:n=7;
                    break;
            case 18:n=8;
                    break;
            case 27:n=9;
                    break;
          }
          l=o+n;
          r=l/1000;
          s=r*1000;
          o=l-s;
          Set_xy(a,b);
          Print_number(o);
          Clr_key();
          busy!=0x01;
          Delay(1000);
    }
    o=Ketts(m,o,a,b);
    if(on_name==1)
    {
          Clr_l(7,1,61);
          Clr_l(5,21,18);
          Set_xy(7,1);
          k=o*12;
          if(*(many+k)=='*')
          {
               for(i=1;i<12;i++)
               {
                    if(*(many+k+i)=='*')
                         break;
                    ma=*(many+k+i);
                    Printc(ma,0);
               }
               c=*(mem_position+o);
               ug=Adc_degree(c);
               Set_xy(5,21);
               Print_number(ug);
```

```
            }
        }
        if(*(read_num+6)==0)
        {
            if(o>100)
            {
                o=100;
                Set_xy(a,b);
                Print_number(o);
            }
            *(read_num+5)=o;
            busy!=0x01;
            on_name=0;
        }
        else if(*(read_num+6)==1)
            *(read_num+5)=222;
        *read_num=1;
        *(read_num+1)=o;
        *(read_num+2)=n;
        *(read_num+3)=r;
        *(read_num+4)=s;
        return(*(read_num+5));
    }

    /* ##### KETTS ##### */
    Ketts(m,o,a,b)
    char m,a,b;
    int o;
    {
        if(m==11)
        {
           o=o+1;
           if(o>100)
              o=100;
           Set_xy(a,b);
           Print_number(o);
           Clr_key();
           busy!=0x01;
        }
```

```
    else if(m==10)
    {
       o=o-1;
       if(o<0)
          o=0;
       Set_xy(a,b);
       Print_number(o);
       Clr_key();
       busy|=0x01;
    }
    return(o);
}

/* ##### WRITING WIND ##### */
Write_wind()
{
     unsigned char a,b;
     int c;
     a=*port_a2;
     b=~Bcd_gray(a);
     c=Adc_degree(b);
     Set_xy(5,109);
     Print_number(c);
     Set_xy(6,109);
     Print_number(inter2);
     wind_direct=b;
     wind_speed=inter2;
     if(wind_direct!=*wind_test||
        wind_speed!=*(wind_test+1))
          busy|=0x04;
     *wind_test=wind_direct;
     *(wind_test+1)=wind_speed;
}

/* ###### PRINT POSITION ROTOR ###### */
Print_rotor()
{
     int g,h,p,n,o,l;
     g=Read_adc();
```

```
        g=Read_adc();
        g=Read_adc();
        h=Adc_degree(g);
        p=h/100;
        l=h%100;
        n=l/10;
        o=l%10;
        if(*(print_big+2)!=o)
        {
             Set_xy(0,65);
             Print_big_number(p);
             Set_xy(0,85);
             Print_big_number(n);
             Set_xy(0,105);
             Print_big_number(o);
             busy!=0x02;
        }
        position_adc=g;
        *print_big=p;
        *(print_big+1)=n;
        *(print_big+2)=o;
}


/* ###### PRINT POSITION ROTOR ###### */
Printrotor()
{
        int h,p,n,o,l;
        unsigned char g;
        g=Read_adc();
        g=Read_adc();
        g=Read_adc();
        h=Adc_degree(g);
        p=h/100;
        l=h%100;
        n=l/10;
        o=l%10;
             Set_xy(0,65);
             Print_big_number(p);
```

```
            busy!=0x02;
            Set_xy(0,85);
            Print_big_number(n);
            busy!=0x02;
            Set_xy(0,105);
            Print_big_number(o);
            busy!=0x02;
      position_adc=g;
      *print_big=p;
      *(print_big+1)=n;
      *(print_big+2)=o;
}

/* ###### PRINT BIG NUMBER ###### */
Print_big_number(g)
int g;
{
      switch(g)
      {
            case 0: Print_big(B0);
                    break;
            case 1: Print_big(B1);
                    break;
            case 2: Print_big(B2);
                    break;
            case 3: Print_big(B3);
                    break;
            case 4: Print_big(B4);
                    break;
            case 5: Print_big(B5);
                    break;
            case 6: Print_big(B6);
                    break;
            case 7: Print_big(B7);
                    break;
            case 8: Print_big(B8);
                    break;
            case 9: Print_big(B9);
                    break;
```

```
    }
}

/* ###### PRINT BIG ###### */
Print_big(ptr)
char *ptr;
{
    unsigned char a,b;
    int i,j,k,l,p;
    p=0;
    k=y;
    l=x;
    for(i=0;i<3;i++)
    {
        p=i*22;
        Set_xy(l+i,k);
        for(j=0;j<22;j++)
        {
            a=*(ptr+p+j);
            Writer(a);
        }
    }
}

/* ##### Name Station ##### */
Name_station()
{
  int  i,j,k,l,chan,m;

    if(*(memory_busy+2)==0)
    {
        namem[0]='#';
        namem[1]='A';
        namem[2]='A';
        namem[3]='A';
        namem[4]='A';
        namem[5]='A';
        namem[6]='A';
        namem[7]='A';
```

```
        namem[8]='A';
        namem[9]='A';
        namem[10]='A';
        namem[11]='A';
        k=1;
        i=1;
        Clr_l(7,1,61);
        Set_xy(7,1);
        Printc(namem[i],0x80);
        busy!=0x01;
        Clr_key();
        m=Keyb();
    }

    else
    {
        y=*store;
        i=*(store+1);
        j=*(store+2);
        k=*(store+3);
        l=*(store+4);
        x=*(store+5);
    }

    *(memory_busy+2)=1;
    m=Keyb();
    if(m==11)
    {
         y-=6;
         namem[i]++;
         if(namem[i]==0x5B)
             namem[i]+=6;
         if(namem[i]==0x7B)
             namem[i]=' ';
         if(namem[i]==0x21)
             namem[i]='A';
         if(namem[i]<0x3A&&namem[i]>0x2F)
             namem[i]='A';
         Set_y(y);
```

```
        Printc(namem[i],0x80);
        busy!=0x01;
        Delay(5000);
    }
    else if(m==10)
    {
        y-=6;
        namem[i]--;
        if(namem[i]==0x40)
            namem[i]=' ';
        if(namem[i]==0x1F)
            namem[i]='z';
        if(namem[i]==0x60)
            namem[i]='Z';
        if(namem[i]<0x3A&&namem[i]>0x2F)
            namem[i]='A';
        Set_y(y);
        Printc(namem[i],0x80);
        busy!=0x01;
        Delay(5000);
    }
    else if(m<=27&&m>=17&&m!=26)
    {
        switch(m)
        {
           case 17: namem[i]='7';
                    break;
           case 18: namem[i]='8';
                    break;
           case 19: namem[i]='6';
                    break;
           case 20: namem[i]='5';
                    break;
           case 21: namem[i]='3';
                    break;
           case 22: namem[i]='2';
                    break;
           case 23: namem[i]='1';
                    break;
```

```
            case 24: namem[i]='4';
                     break;
            case 25: namem[i]='0';
                     break;
            case 27: namem[i]='9';
                     break;
        }
        i++;
        y-=6;
        Set_y(y);
        if(i>10)
        {
           i=10;
           Printc(namem[i],0x80);
        }
        else
        {
           Printc(namem[i-1],0);
           Printc(namem[i],0x80);
        }
        if(i>k)
           k++;
        busy!=0x01;
        Delay(5000);
}
else if(m==9)
{
        i++;
        y-=6;
        Set_y(y);
        if(i>10)
        {
           i=10;
           Printc(namem[i],0x80);
        }
        else
        {
           Printc(namem[i-1],0);
           Printc(namem[i],0x80);
```

```
        }
      if(i>k)
         k++;
        busy|=0x01;
        Delay(5000);
 }
 else if(m==12)
 {
      y-=6;
      i--;
      if(i<1)
      {
         i=1;
         Set_y(y);
         Printc(namem[i],0x80);
      }
      else
      {
        y-=6;
        Set_y(y);
        Printc(namem[i],0x80);
        Printc(namem[i+1],0);
        y-=6;
      }
      busy|=0x01;
      Delay(5000);
}
else if(m==28)
{
      chan=channal*12;
      *(mamy+chan)='*';
      for(j=1;j<=k;j++)
      {
         *(mamy+chan+j)=namem[j];
      }
      *(mamy+chan+j)='*';
      *(mem_position+channal)=position_adc;
      Delay(500);
      Clr_1(0,10,6);
```

```
            *(memory_busy+2)=0;
            busy!=0x01;
            name_busy=0;
        }
        *store=y;
        *(store+1)=i;
        *(store+2)=j;
        *(store+3)=k;
        *(store+4)=l;
        *(store+5)=x;
}


/* ###### LINE ###### */
Line(a,b,c,d)
int a,b,c,d;
{
     int i,j,k;
     static dot[] = {0x01,0x02,0x04,0x08,0x10,0x20,
                     0x40,0x80};
     for(k=a;k<=c;k++)
     {
          x=k/8;
          i=k%8;
          for(j=b;j<=d;j++)
          {
               y=j;
               Write_or(dot[i]);
          }
     }
}


/* ##### WRITE TO RAM ##### */
Writer(w)
```

```
char w;
{
     int i,j,k;
     i=x*128;
     *(lcd+i+y)=w;
     y++;
}


/* ###### WRITE OR ###### */
Write_or(w)
char w;
{
     int i,j,k;
     i=x*128;
     *(lcd+i+y)|=w;
     y++;
}

/* ##### WRITE FROM RAM TO LCD ##### */
Write_lcd()
{
     static char page[]={0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,
                         0xBE,0xBF};
     int i,j,l;
     unsigned char k,n;
     l=0;
     n=busy&0x01;
     if(n==1)
     {
          for(i=1;i<8;i++)
          {
               l=i*128;
               Ready();
               *ilcd_cs2=page[i];
               for(j=0;j<64;j++)
               {
                    k=0x40|j;
                    Ready();
```

```
                    *ilcd_cs2=k;
                    Ready();
                    *dlcd_cs2=*(lcd+l+j);
                }
            }
        busy&=0xFE;
        }
    }

    /* ##### WRITE FROM RAM TO LCD ##### */
    Write_mode()
    {
        int i,j,l;
        unsigned char k,n;
        l=0;
        n=busy&0x08;
        if(n==0x08)
        {
            Ready();
            *ilcd_cs2=0xB8;
            for(j=0;j<64;j++)
            {
                k=0x40|j;
                Ready();
                *ilcd_cs2=k;
                Ready();
                *dlcd_cs2=*(lcd+j);
            }
            busy&=0xF7;
        }
    }

    /* ##### WRITE FROM RAM TO LCD ##### */
    Write_lcd_rotor()
    {
        static char page[]={0xB8,0xB9,0xBA,0xBB};
        int i,j,l;
        char k,n;
        l=0;
```

```
        n=busy&0x02;
        if(n==0x02)
        {
        for(i=0;i<4;i++)
        {
              l=i*128;
              Ready();
              *ilcd_cs1=page[i];
              for(j=64;j<128;j++)
              {
                        k=0x40|(j-64);
                        Ready();
                        *ilcd_cs1=k;
                        Ready();
                        *dlcd_cs1=*(lcd+l+j);
              }
        }
        busy&=0xFD;
        }
}

/* ##### WRITE FROM RAM TO LCD ##### */
Write_lcd_wind()
{
        static char page[]={0,0,0,0,0xBC,0xBD,0xBE,0xBF};
        int i,j,l;
        char k,n;
        l=0;
        n=busy&0x04;
        if(n==0x04)
        {
        for(i=4;i<8;i++)
        {
              l=i*128;
              Ready();
              *ilcd_cs1=page[i];
              for(j=64;j<128;j++)
              {
                        k=0x40|(j-64);
```

```
                    Ready();
                    *ilcd_cs1=k;
                    Ready();
                    *dlcd_cs1=*(lcd+l+j);
                }
            }
            busy&=0xFB;
            }
        }

/* ##### WRITE_WORD ##### */
Writew(g,h)
int  h;
char *g;
{
    int i;
    for(i=0;i<h;i++)
        Writer(*(g+i));
}

/* ##### PRINT A CHARACTER TO LCD ##### */
Printc(prn,cha)     /* prn = ASCII */
char prn,cha;
{   char i;
    prn-=32;
    switch(prn)
    { oase  0: Write_char(SSS,cha);
               break;
      case  6: Write_char(XX8,cha);
               break;
      case  7: Write_char(CO,cha);
               break;
      case  8: Write_char(XX1,cha);
               break;
      case  9: Write_char(XX2,cha);
               break;
      case 10: Write_char(XX3,cha);
               break;
      case 11: Write_char(XX4,cha);
```

```
            break;
 case 12: Write_char(XX5,cha);
            break;
 case 13: Write_char(XX6,cha);
            break;
 case 14: Write_char(XX7,cha);
            break;
 case 15: Write_char(BS,cha);
            break;
 case 16: Write_char(O0,cha);
            break;
 case 17: Write_char(O1,cha);
            break;
 case 18: Write_char(O2,cha);
            break;
 case 19: Write_char(O3,cha);
            break;
 case 20: Write_char(O4,cha);
            break;
 case 21: Write_char(O5,cha);
            break;
 case 22: Write_char(O6,cha);
            break;
 case 23: Write_char(O7,cha);
            break;
 case 24: Write_char(O8,cha);
            break;
 case 25: Write_char(O9,cha);
            break;
 case 26: Write_char(XX9,cha);
            break;
 case 27: Write_char(XX0,cha);
            break;
 case 30: Write_char(ARR,cha);
            break;
 case 33: Write_char(A,cha);
            break;
 case 34: Write_char(B,cha);
            break;
```

```
case 35: Write_char(C,cha);
         break;
case 36: Write_char(D,cha);
         break;
case 37: Write_char(E,cha);
         break;
case 38: Write_char(F,cha);
         break;
case 39: Write_char(G,cha);
         break;
case 40: Write_char(H,cha);
         break;
case 41: Write_char(I,cha);
         break;
case 42: Write_char(J,cha);
         break;
case 43: Write_char(K,cha);
         break;
case 44: Write_char(L,cha);
         break;
case 45: Write_char(M,cha);
         break;
case 46: Write_char(N,cha);
         break;
case 47: Write_char(O,cha);
         break;
case 48: Write_char(P,cha);
         break;
case 49: Write_char(Q,cha);
         break;
case 50: Write_char(R,cha);
         break;
case 51: Write_char(S,cha);
         break;
case 52: Write_char(T,cha);
         break;
case 53: Write_char(U,cha);
         break;
case 54: Write_char(V,cha);
```

```
        break;
case 55: Write_char(W,cha);
        break;
case 56: Write_char(X,cha);
        break;
case 57: Write_char(Y,cha);
        break;
case 58: Write_char(Z,cha);
        break;
case 65: Write_char(Aa,cha);
        break;
case 66: Write_char(Bb,cha);
        break;
case 67: Write_char(Cc,cha);
        break;
case 68: Write_char(Dd,cha);
        break;
case 69: Write_char(Ee,cha);
        break;
case 70: Write_char(Ff,cha);
        break;
case 71: Write_char(Gg,cha);
        break;
case 72: Write_char(Hh,cha);
        break;
case 73: Write_char(Ii,cha);
        break;
case 74: Write_char(Jj,cha);
        break;
case 75: Write_char(Kk,cha);
        break;
case 76: Write_char(Ll,cha);
        break;
case 77: Write_char(Mm,cha);
        break;
case 78: Write_char(Nn,cha);
        break;
case 79: Write_char(Oo,cha);
        break;
```

```
        case 80: Write_char(Pp,cha);
                 break;
        case 81: Write_char(Qq,cha);
                 break;
        case 82: Write_char(Rr,cha);
                 break;
        case 83: Write_char(Ss,cha);
                 break;
        case 84: Write_char(Tt,cha);
                 break;
        case 85: Write_char(Uu,cha);
                 break;
        case 86: Write_char(Vv,cha);
                 break;
        case 87: Write_char(Ww,cha);
                 break;
        case 88: Write_char(Xx,cha);
                 break;
        case 89: Write_char(Yy,cha);
                 break;
        case 90: Write_char(Zz,cha);
                 break;
        case 91: Write_char(YY0,cha);
                 break;
        case 92: Write_char(YY1,cha);
                 break;
        case 93: Write_char(YY2,cha);
                 break;
        case 94: Write_char(YY3,cha);
                 break;
    }
}

/* ##### WRITE_CHAR ##### */
Write_char(ptr,cha)
char *ptr,cha;

{   char t;
    int  i;
```

```
        for(i=0;i(=5;i++)
        {
           t=*(ptr+i)¦cha;
           Writer(t);
        }

}

/* ##### PRINT WORD ##### */
Printw(pr)
char **pr;
{
    int i,j;
    for(j=0;;j++)
    {
      for(i=0;;i++)
      {
         if(*(*(pr+j)+i)=='%')
            break;
         if(*(*(pr+j)+i)=='$')
            break;
         Printc(*(*(pr+j)+i),0);
      }
      if(*(*(pr+j)+i)=='%')
         break;
      x++;
      Set_xy(x,0);
    }
}

/* ##### PRINT NUMBER ##### */
Print_number(n)
int n;
{
    int t,o,p,q,i;
    t=n/100;
    o=n%100;
    p=o/10;
    q=o%10;
```

```
    for(i=0;i<3;i++)
    {
      if(i==1)
        t=p;
      if(i==2)
        t=q;
      switch(t)
      {
        case 0: Write_char(00,0);
                break;
        case 1: Write_char(01,0);
                break;
        case 2: Write_char(02,0);
                break;
        case 3: Write_char(03,0);
                break;
        case 4: Write_char(04,0);
                break;
        case 5: Write_char(05,0);
                break;
        case 6: Write_char(06,0);
                break;
        case 7: Write_char(07,0);
                break;
        case 8: Write_char(08,0);
                break;
        case 9: Write_char(09,0);
                break;
      }
    }
}

/* ##### READ ADC ##### */
Read_adc()
{
    unsigned char z;
    Delay(50);
    *port_c2=0x02;
    Delay(50);
```

```
    *port_c2=0x06;
    z=*port_b2;
    *port_c2=0x01;
    *port_c2=0;
    return((int)z);
}

/* ###### ADC TO DEGREE ###### */
Adc_degree(adc)
unsigned char adc;
{
    int  c;
    c=adc*1.40625;
    return(c);
}


/* ###### CLEAR KEYBOARD ###### */
Clr_key()
{
    int m;
        for(;;)
        {
           m=Keyb();
           if(m==0)
              break;
        }
}

/* ###### BCD TO GRAY CODE ###### */
Bcd_gray(bcd)
unsigned char bcd;
{
    static char sif[]={0x80,0x40,0x20,0x10,0x08,
                       0x04,0x02,0x01};
    unsigned char b,c,d,f;
    int i;
    f=bcd & sif[0];
    for(i=1;i<8;i++)
```

```
    {
        b= f>>1;
        c= b^bcd;
        d= c & sif[i];
        f|= d;
    }
    return(f);
}


/* ###### Initial port ###### */
Init_port()
{
    *port_i=0x92;
    *port_c=0x80;
    *port_i2=0x92;
    *port_c=0;
}

/* ###### Check Ready Of LCD ###### */
Ready()
{
    char    ready;

    for(;;)
    {
        ready=*ilcd_cs2;
        ready&=0x80;
        if(ready!=0x80)
            break;
    }
}

/* #####  DELAY  ##### */
Delay(de)
{
    int  delay,i;
    for(delay=0;delay<de;delay++);
}
```

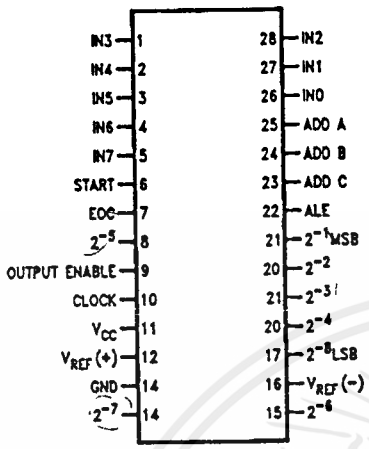
```
/* ##### SET_Y ##### */
Set_y(sy)
int sy;
{
    char   sy1;
    y=sy;
}
/* ##### SET_X ##### */
Set_x(sx)
int sx;
{
    x=sx;
}

/* ##### Set_xy ##### */
Set_xy(a,b)
int  a,b;
{
    Set_x(a);
    Set_y(b);
}

/* ##### CLEAR SCREEN ##### */
Clr_s()
{
     static char page[]={0xB8,0xB9,0xBA,0xBB,
                         0xBC,0xBD,0xBE,0xBF};
     int i,j,l;
     char k;
     l=0;
     for(i=0;i<8;i++)
     {
         l=i*128;
         Ready();
         *ilcd_cs2=page[i];
         Ready();
         *ilcd_cs1=page[i];
         Ready();
```

```
            *ilcd_cs2=0x40;
            Ready();
            *ilcd_cs1=0x40;
            for(j=0;j<64;j++)
            {
                  Ready();
                  *dlcd_cs2=0;
                  Ready();
                  *dlcd_cs1=0;
            }
      }
      Clr_ram();
}



/* ##### CLEAR SCREEN ##### */
Clr_ram()
{
      int i;
      for(i=0;i<1024;i++)
      {
          *(lcd+i)=0;
      }
}


/* ##### CLEAR LINE ##### */
Clr_l(a,b,c)
int a,b,c;
{
      int i;
      Set_xy(a,b);
      for(i=0;i<c;i++)
          Writer(0);
}


/* ##### DISPLAY ON ##### */
```

```
Display_on()
{
    Ready();
    *ilcd_cs2=0x3F;    /*SET DISPLAY 2 ON*/
    Ready();
    *ilcd_cs1=0x3F;
}

/* ##### START LINE ##### */
Start_line()
{
    Ready();
    *ilcd_cs2=0xC0;  /*SET START LINE #0 */
    Ready();
    *ilcd_cs1=0xC0;
}

/* ##### ON LCD GRAPHIC ##### */
On_lcd()
{
    Display_on();
    Start_line();
}

/* #####  SCAN KEYBOARD  ##### */
int Keyb()
{   static  char  scn[] = { 0x0E, 0x0D, 0x0B, 0x07};
    unsigned char    k1;
    int     k2,k3,k4,k;
    k2=0;
    for(k4=0;k4<4;k4++)
    {
       *port_c=scn[k4];
       k1=*port_b;
       if(k1!=0xFF)
         break;
       k2+=8;
    }
    if(k1!=0xFE&&k1!=0xFD&&k1!=0xFB&&k1!=0xF7&&
```

Connection Diagrams

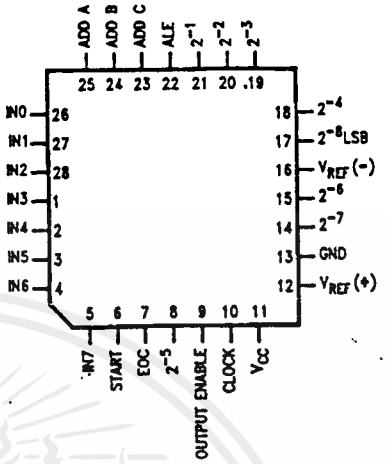
Dual-In-Line Package



TL/H/5672-11

Order Number ADC0808CCN, ADC0809CCN, ADC0808CCJ or ADC0808CJ
See NS Package J28A or N28A

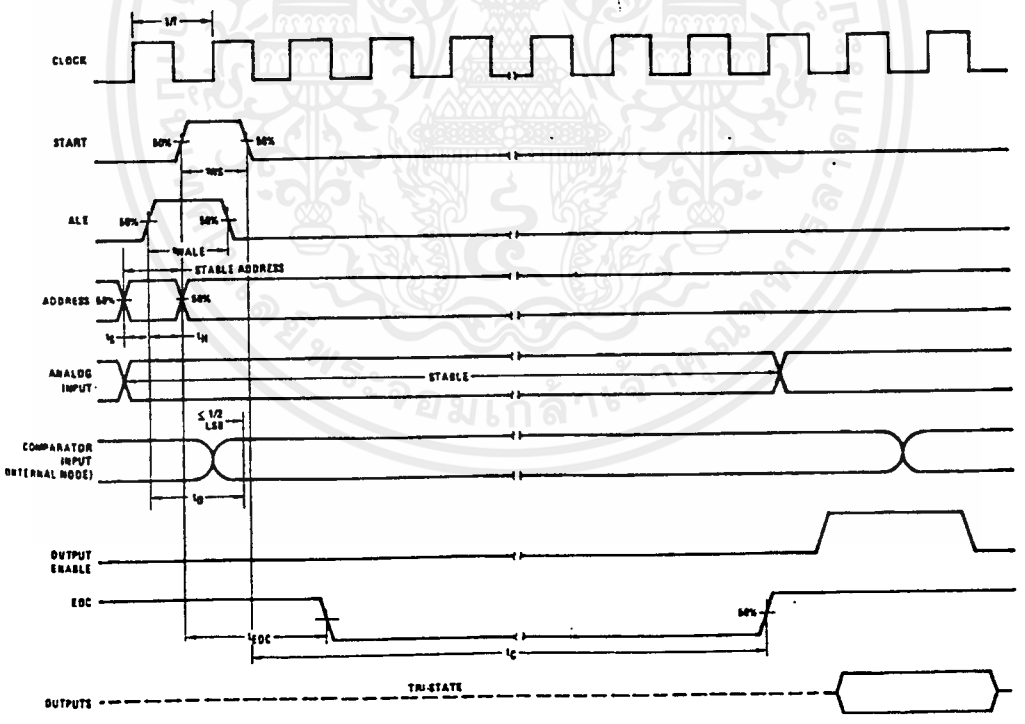
Molded Chip Carrier Package



TL/H/5672-12

Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Timing Diagram



TL/H/5672-4

FIGURE 5

National Semiconductor Corporation

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. (See AN-247 for more information.)

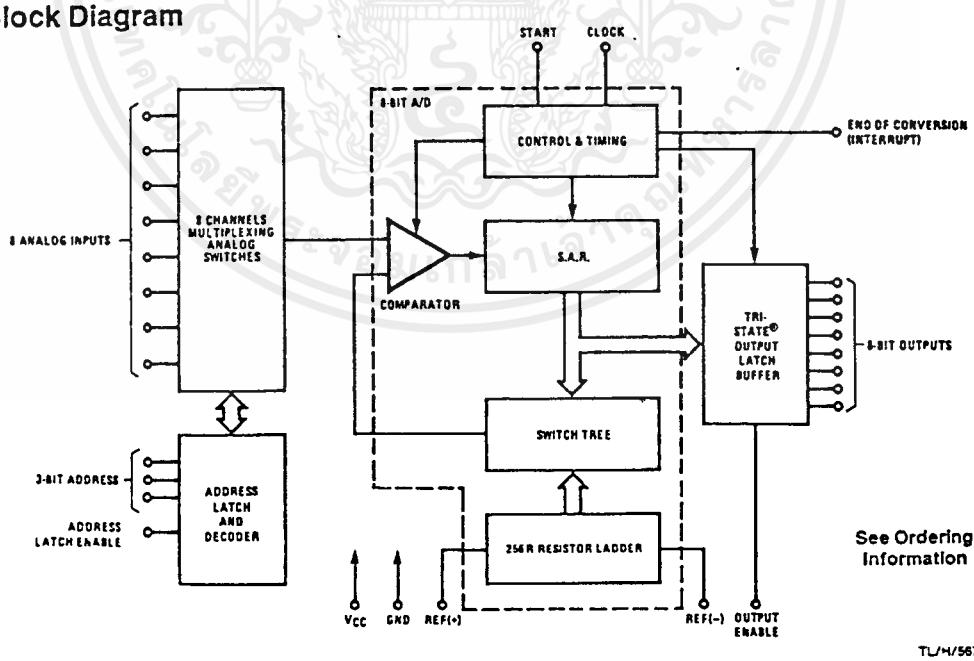
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 μs

Block Diagram



See Ordering Information

TL/H/5672–1

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

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 Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Molded Chip Carrier Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	400V

Operating Conditions (Notes 1 & 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le T_{MAX}$
ADC0808CJ	−55°C≤T_A≤+125°C
ADC0808CCJ, ADC0808CCN, ADC0809CCN	−40°C≤T_A≤+85°C
ADC0808CCV, ADC0809CCV	−40°C ≤ T_A ≤ +85°C
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.0 V_{DC}

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.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	ADC0808 Total Unadjusted Error (Note 5)	25°C T_{MIN} to T_{MAX}			±½ ±¾	LSB LSB
	ADC0809 Total Unadjusted Error (Note 5)	0°C to 70°C T_{MIN} to T_{MAX}			±1 ±1¼	LSB 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_{DC}
$V_{REF(+)}$	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
I_{IN}	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, ADC0808CCV, ADC0809CCN and ADC0809CCV, 4.75≤V_{CC}≤5.25V, −40°C≤T_A≤+85°C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
ANALOG MULTIPLEXER						
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=5V, T_A=25°C T_{MIN} to T_{MAX}		10	200 1.0	nA μA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=0, T_A=25°C T_{MIN} to T_{MAX}	−200 −1.0	−10		nA μA

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, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40°C \le T_A \le +85°C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
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
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	$I_O=-360$ μA	$V_{CC}-0.4$			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	$V_O=5V$ $V_O=0$	 −3		3	μA 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_s	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_{H1}, t_{H0}	OE Control to Q Logic State	$C_L=50$ pF, $R_L=10$k *(Figure 8)*		125	250	ns
t_{1H}, t_{0H}	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 indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

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 $0V_{DC}$ to $5V_{DC}$ 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.

Note 8: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

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 + ½ LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 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.

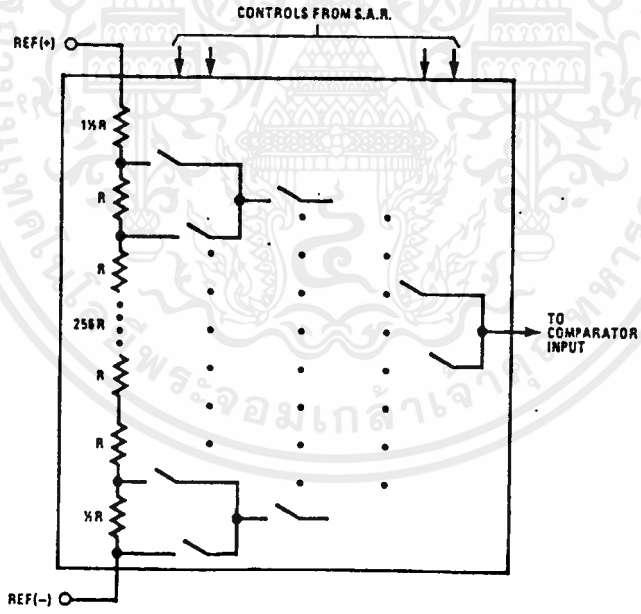


FIGURE 1. Resistor Ladder and Switch Tree

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 throught 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.

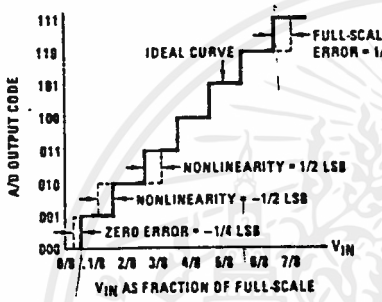


FIGURE 2. 3-Bit A/D Transfer Curve

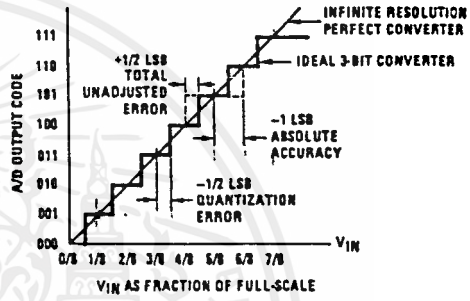


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

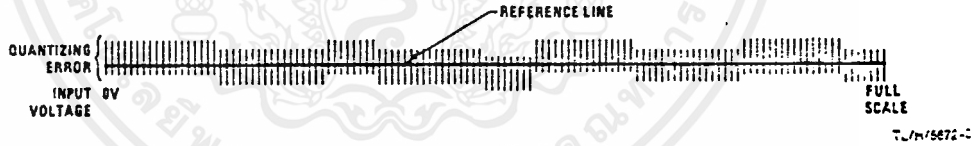


FIGURE 4. Typical Error Curve

```
    k1!=0xEF&&k1!=0xDF&&k1!=0xBF&&k1!=0x7F)
  {
     k2=0;
     k1=0;
  }
  if(k1==0x7F)
     k2+=1;
  if(k1==0xBF)
     k2+=2;
  if(k1==0xDF)
     k2+=3;
  if(k1==0xEF)
     k2+=4;
  if(k1==0xF7)
     k2+=5;
  if(k1==0xFB)
     k2+=6;
  if(k1==0xFD)
     k2+=7;
  if(k1==0xFE)
     k2+=8;
  return(k2);
}
```

Typical Performance Characteristics

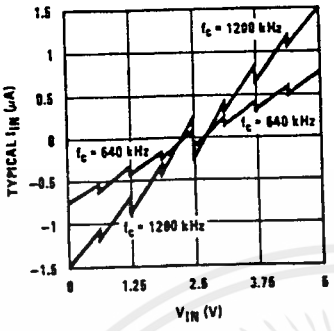
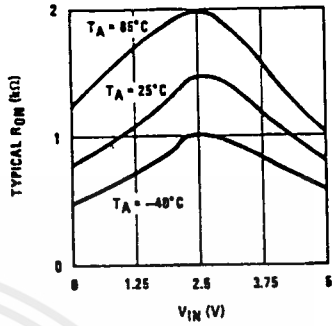


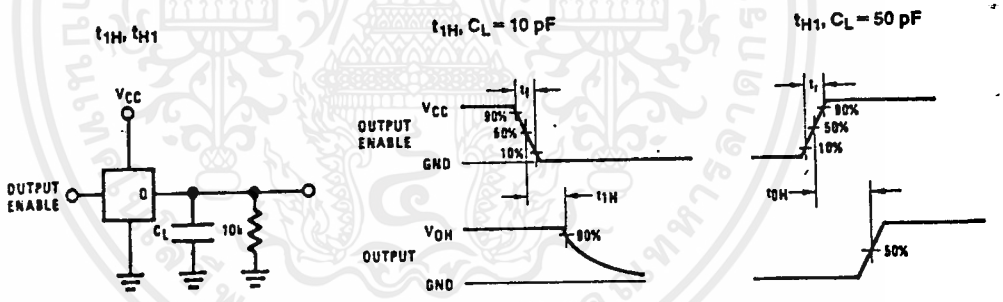
FIGURE 6. Comparator I_{IN} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)



TL/H/5573-6

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

TRI-STATE Test Circuits and Timing Diagrams



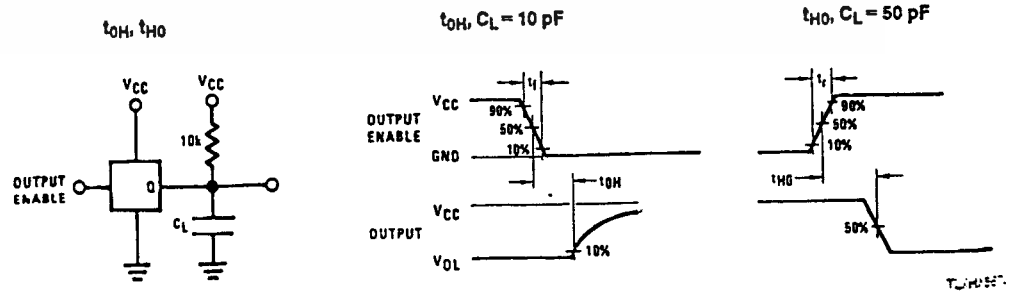


FIGURE 8

Reflective Assemblies

Characteristics			Light Current			Dark Current		LED Foward Voltage		Light Current Rise Time		Collector-Emitter Breakdown	Output
Test Conditions			V_{CE} = 5V I_F as Shown			I_F = 0, H = 0 V_{CE} as Shown		I_F as Shown		V_{CE} = 5V I_C = 1 mA		I_C = 100 µA	
Symbol			I_L			I_D		V_F		t_r		BV_{CEO}	
Units			mA			nA		V		µs		V	
	Fig	Note	Min	I_F(mA)	d inch	Max	V_{CE}(V)	Max	I_F(mA)	Typ	Notes	Min	
HOA149-001	5	7	0.025	40	0.15 *3	100	15	1.6	40	15	*2	30	Transistor
HOA708-001	6	8	0.010	40	0.15 *3	100	10	1.7	40	15	*2	30	Transistor
HOA708-011			0.010	40	0.15 *3	100	10	1.7	40	15	*2	30	Transistor
HOA709-001			1.00	40	0.15 *3	250	10	1.7	40	15	*2	15	Darlington
HOA709-011			1.00	40	0.15 *3	250	10	1.7	40	15	*2	15	Darlington
HLC1395-001	7		0.300	10	0.04 *3	100	10	1.5	20	6	*3	30	Transistor
HLC1395-002			0.600	10	0.04 *3	100	10	1.5	20	6	*3	30	Transistor
HOA1397-001	8		0.20	20	0.05 *3	100	10	1.5	20	6	*2	30	Transistor
HOA1397-002			0.70	20	0.05 *3	100	10	1.5	20	6	*2	30	Transistor
HOA1397-031			2.0	20	0.05 *3	250	10	1.5	20	50	*3	15	Darlington
HOA1397-032			7.0	20	0.05 *3	250	10	1.5	20	50	*3	15	Darlington
HOA1405-001	9	8	0.20	30	0.20 *3	100	10	1.5	20	10	*2	30	Transistor
HOA1405-002			0.80	30	0.20 *3	100	10	1.5	20	15	*2	30	Transistor

OUTLINE DIMENSIONS

Tolerance 3 plc decimals ±.010 (.25)
2 plc decimals ±.030 (.76)
Unless specified

ALL DIMENSIONS IN INCHES (MILLIMETERS)

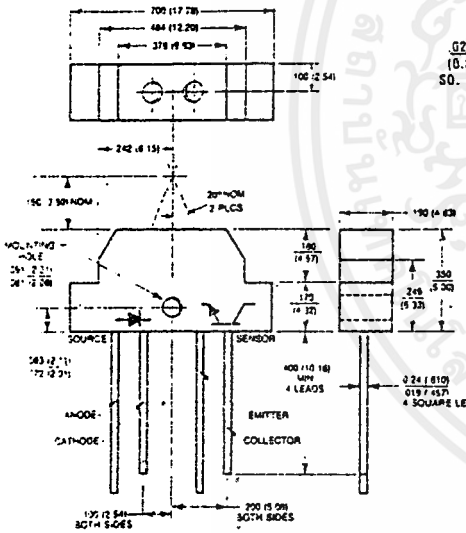


Figure 5

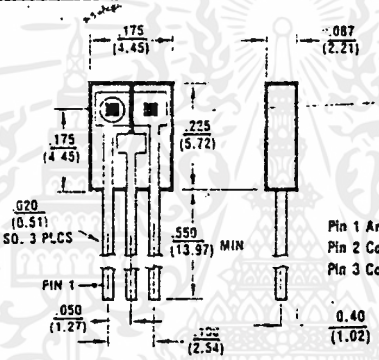


Figure 7

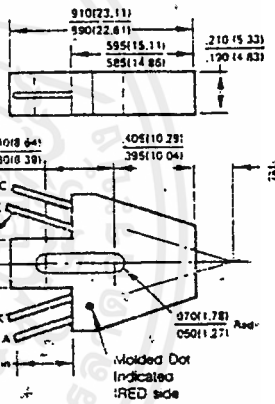


Figure 9

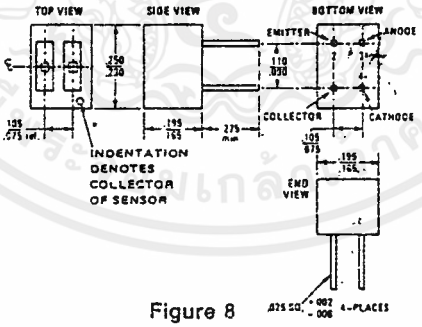


Figure 8

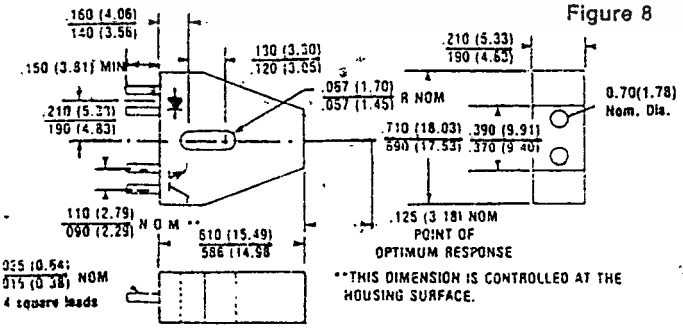


Figure 6

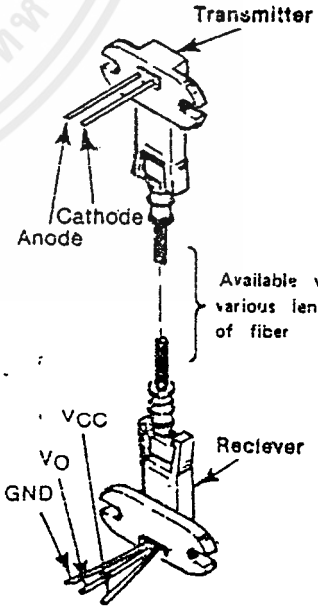


Figure 10

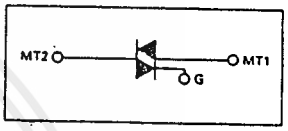
Triacs
Silicon Bidirectional Thyristors

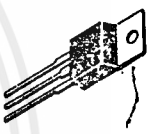
T2500 Series

TRIACs
6 AMPERES RMS
200 thru 800 VOLTS

... designed primarily for full-wave ac control applications, such as light dimmers, motor controls, heating controls and power supplies.

- Blocking Voltage to 800 Volts
- All Diffused and Glass Passivated Junctions for Greater Parameter Uniformity and Stability
- Small, Rugged, Thermowatt Construction for Low Thermal Resistance, High Heat Dissipation and Durability





CASE 221A-04
(TO-220AB)
STYLE 4

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Repetitive Peak Off-State Voltage, Note 1 (T_J = −40 to +100°C) Gate Open T2500 B D M N	V_{DROM}	200 400 600 800	Volts
On-State Current RMS (Full Cycle Sine Wave 50 to 60 Hz) (T_C = +80°C)	$I_{T(RMS)}$	6	Amps
Peak Non-Repetitive Surge Current (One Full Cycle, 60 Hz, T_C = +80°C)	I_{TSM}	60	Amps
Circuit Fusing Considerations (t = 8.3 ms)	I^2t	15	A^2s
Peak Gate Power (T_C = +80°C, Pulse Width = 1 μs)	P_{GM}	16	Watts
Average Gate Power (T_C = +80°C, t = 8.3 ms)	$P_{G(AV)}$	0.2	Watt
Peak Gate Trigger Current (Pulse Width = 10 μs)	I_{GTM}	4	Amps
Operating Junction Temperature Range	T_J	−40 to +100	°C
Storage Temperature Range	T_{stg}	−40 to +150	°C

Note 1. Ratings apply for open gate conditions. Thyristor devices shall not be tested with a constant current source for blocking capability such that the voltage applied exceeds the rated blocking voltage.

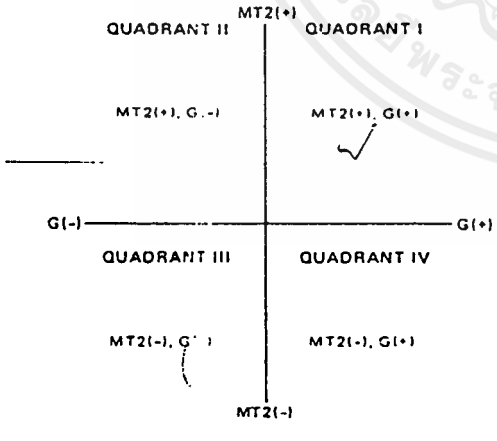
THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	$R_{\theta JC}$	2.7	°C/W

ELECTRICAL CHARACTERISTICS (T_C = 25°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Peak Forward or Reverse Blocking Current (Rated V_{DRM} or V_{RRM}, gate open) T_J = 100°C	I_{DRM}, I_{RRM}	—	—	2	mA
Maximum On-State Voltage (Either Direction) (I_T = 30 A Peak)	V_{TM}	—	—	2	Volts
Gate Trigger Current (Continuous dc) (V_D = 12 Vdc, R_L = 12 Ohms)	I_{GT}				mA
$V_{MT2(+)}$, $V_{G(+)}$		—	10	25	
$V_{MT2(+)}$, $V_{G(-)}$		—	20	60	
$V_{MT2(-)}$, $V_{G(-)}$		—	15	25	
$V_{MT2(-)}$, $V_{G(+)}$		—	30	60	
Gate Trigger Voltage (Continuous dc) (All Quadrants) (V_D = 12 Vdc, R_L = 12 Ohms)	V_{GT}	—	1.25	2.5	Volts
(V_D = V_{DROM}, R_L = 125 Ohms, T_C = 100°C)		0.2	—	—	
Holding Current (Either Direction) (Main Terminal Voltage = 12 Vdc, Gate Open, Initiating Current = 150 mA, T_C = 25°C)	I_{HO}	—	15	30	mA
Gate Controlled Turn-On Time (Rated V_{DROM}, I_T = 10 A, I_{GT} = 160 mA, Rise Time = 0.1 μs)	t_{gt}	—	1.6	—	μs
Critical Rate of Rise of Commutation Voltage (Rated V_{DROM}, $I_{T(RMS)}$ = 6 A, Commutating di/dt = 3.2 A/ms, Gate Unenergized, T_C = 80°C)	dv/dt(c)	—	10	—	V/μs
Critical Rate of Rise of Off-State Voltage (Rated V_{DROM}, Exponential Voltage Rise, Gate Open, T_C = 100°C)	dv/dt				V/μs
T2500B		—	100	—	
T2500D,M,N		—	75	—	

QUADRANT DEFINITIONS



ELECTRICAL CHARACTERISTICS of RECOMMENDED BIDIRECTIONAL SWITCHES

USAGE	General	
PART NUMBER	MBS4991	MBS4992
V_S	6.0 – 10 V	7.5 – 9.0 V
I_S	350 μA Max	120 μA Max
$V_{S1} - V_{S2}$	0.5 V Max	0.2 V Max
Temperature Coefficient	0.02%/°C Typ	

See AN-526 for Theory and Characteristics of Silicon Bidirectional Switches.

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

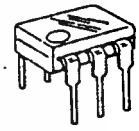
MOC3009
MOC3010
MOC3011
MOC3012

6-Pin DIP Optoisolators
Triac Driver Output

6-PIN DIP
OPTOISOLATORS
TRIAC DRIVER OUTPUT
250 VOLTS

These devices consist of gallium-arsenide infrared emitting diodes, optically coupled to silicon bilateral switch and are designed for applications requiring isolated triac triggering, low-current isolated ac switching, high electrical isolation (to 7500 V peak), high detector standoff voltage, small size, and low cost.

- UL Recognized File Number 54915
- VDE approved per standard 0883/6.80 (Certificate number 41853), with additional approval to DIN IEC380/VDE0806, IEC435/VDE0805, IEC65/VDE0860, VDE110b, covering all other standards with equal or less stringent requirements, including IEC204/VDE0113, VDE0160, VDE0832, VDE0833, etc.
- Special lead form available (add suffix "T" to part number) which satisfies VDE0883/6.80 requirement for 8 mm minimum creepage distance between input and output solder pads.
- Various lead form options available. Consult "Optoisolator Lead Form Options" data sheet for details.



CASE 730A-02
PLASTIC

COUPLER SCHEMATIC

1. ANODE
2. CATHODE
3. NC
4. MAIN TERMINAL
5. SUBSTRATE-
   DO NOT CONNECT
6. MAIN TERMINAL

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INFRARED EMITTING DIODE			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
Total Power Dissipation @ T_A = 25°C Negligible Power in Transistor Derate above 25°C	P_D	100 1.33	mW mW/°C
OUTPUT DRIVER			
Off-State Output Terminal Voltage	V_{DRM}	250	Volts
Peak Repetitive Surge Current (PW = 1 ms, 120 pps)	I_{TSM}	1	A
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	300 4	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage (1) (Peak ac Voltage, 60 Hz, 5 Second Duration)	V_{ISO}	7500	Vac
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	330 4.4	mW mW/°C
Junction Temperature Range	T_J	−40 to +100	°C
Ambient Operating Temperature Range	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−40 to +150	°C
Soldering Temperature (10 s)	—	260	°C

(1) Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating.

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
INPUT LED					
Reverse Leakage Current (V_R = 3 V)	I_R	—	0.05	100	μA
Forward Voltage (I_F = 10 mA)	V_F	—	1.15	1.5	Volts
OUTPUT DETECTOR (I_F = 0 unless otherwise noted)					
Peak Blocking Current, Either Direction (Rated V_{DRM}, Note 1)	I_{DRM}	—	10	100	nA
Peak On-State Voltage, Either Direction (I_{TM} = 100 mA Peak)	V_{TM}	—	1.8	3	Volts
Critical Rate of Rise of Off-State Voltage (Figure 7, Note 2)	dv/dt	—	10	—	V/μs
COUPLED					
LED Trigger Current, Current Required to Latch Output (Main Terminal Voltage = 3 V, Note 3) MOC3009	I_{FT}	—	15	30	mA
MOC3010		—	8	15	
MOC3011		—	5	10	
MOC3012		—	3	5	
Holding Current, Either Direction	I_H	—	100	—	μA

Notes: 1. Test voltage must be applied within dv/dt rating.
2. This is static dv/dt. See Figure 7 for test circuit. Commutating dv/dt is a function of the load-driving thyristor(s) only.
3. All devices are guaranteed to trigger at an I_F value less than or equal to max I_{FT}. Therefore, recommended operating I_F lies between max I_{FT} (30 mA for MOC3009, 15 mA for MOC3010, 10 mA for MOC3011, 5 mA for MOC3012) and absolute max I_F (60 mA).

TYPICAL ELECTRICAL CHARACTERISTICS

T_A = 25°C

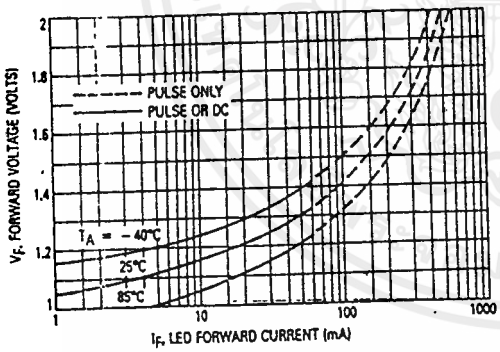


Figure 1. LED Forward Voltage versus Forward Current

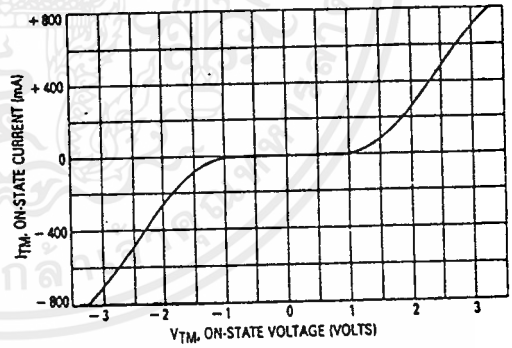


Figure 2. On-State Characteristics

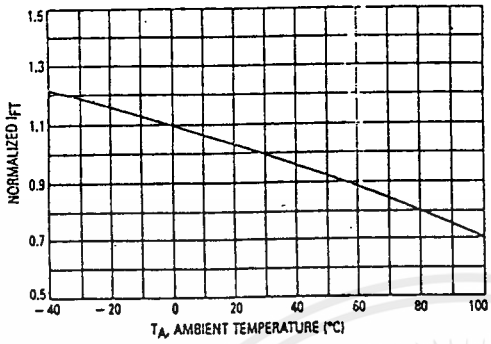


Figure 3. Trigger Current versus Temperature

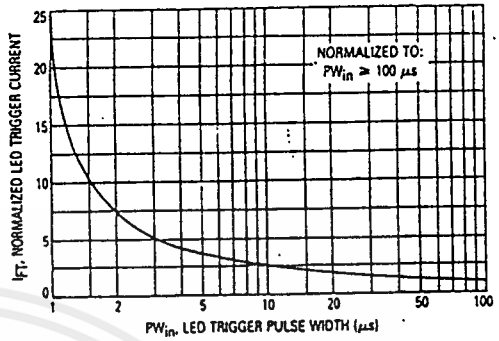


Figure 4. LED Current Required to Trigger versus LED Pulse Width

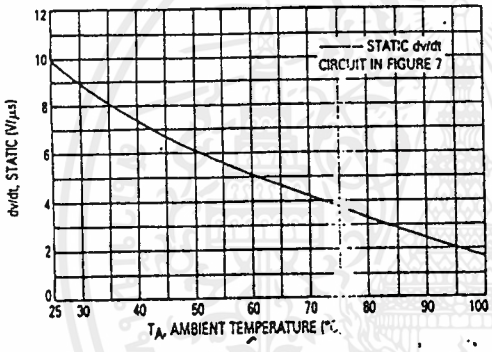


Figure 5. dv/dt versus Temperature

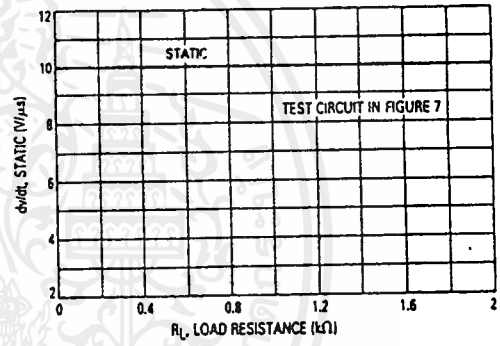
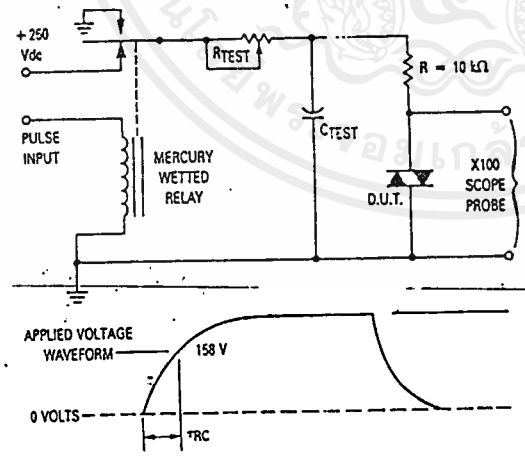


Figure 6. dv/dt versus Load Resistance



1. The mercury wetted relay provides a high speed repeated pulse to the D.U.T.
2. 100x scope probes are used, to allow high speeds and voltages.
3. The worst-case condition for static dv/dt is established by triggering the D.U.T. with a normal LED input current, then removing the current. The variable R_{TEST} allows the dv/dt to be gradually increased until the D.U.T. continues to trigger in response to the applied voltage pulse, even after the LED current has been removed. The dv/dt is then decreased until the D.U.T. stops triggering. t_{RC} is measured at this point and recorded.

Figure 7. Static dv/dt Test Circuit

TYPICAL APPLICATION CIRCUITS

Note: This optoisolator should not be used to drive a load directly. It is intended to be a trigger device only. Additional information on the use of the MOC3009/3010/3011/3012 is available in Application Note AN-780A.

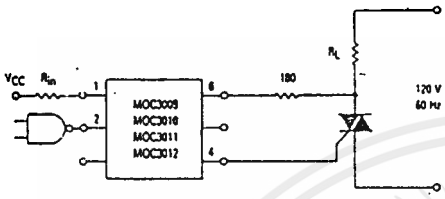


Figure 8. Resistive Load

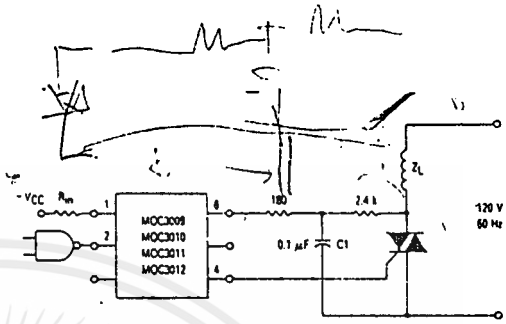


Figure 9. Inductive Load with Sensitive Gate Triac ($I_{GT} \leq 15$ mA)

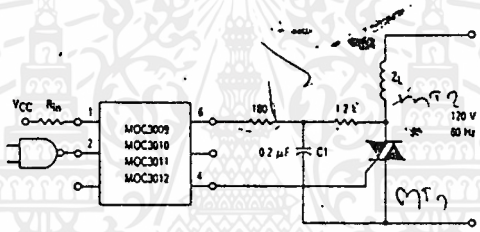
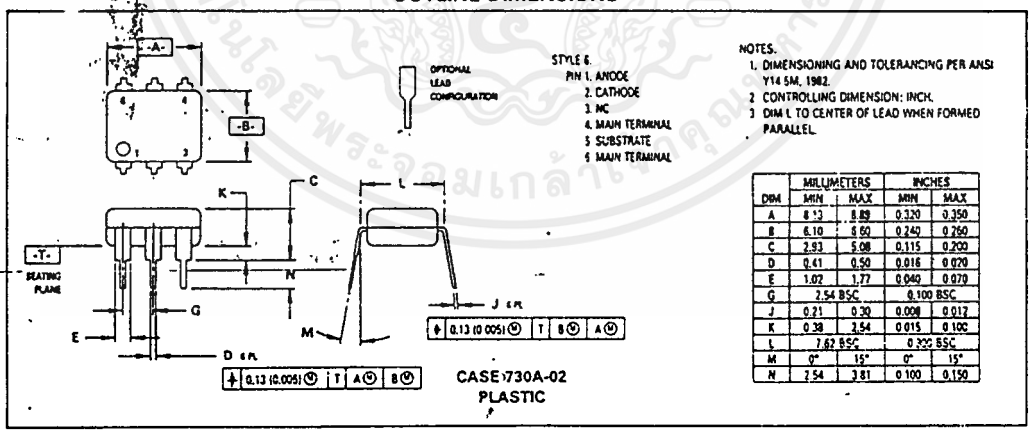


Figure 10. Inductive Load with Non-Sensitive Gate Triac (15 mA < I_{GT} < 50 mA)

OUTLINE DIMENSIONS



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIM L TO CENTER OF LEAD WHEN FORMED PARALLEL.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	8.13	8.89	0.320	0.350
B	6.10	6.60	0.240	0.260
C	2.93	5.08	0.115	0.200
D	0.41	0.50	0.016	0.020
E	1.02	1.77	0.040	0.070
G	2.54 BSC		0.100 BSC	
J	0.21	0.30	0.008	0.012
K	0.38	3.54	0.015	0.100
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	2.54	3.81	0.100	0.150