ระบบบริหารจัดการพลังงานในอาคารพร้อมตรวจจับความเคลื่อนไหว

Energy Management with Movement Monitoring System

โดย

นางสาวพิมพ์อรุณ วัจฉลพงษ์

นายเขตต์ ธนะสีลังกูร

นายอรรถพล จันพันเอก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ผ่านการตรวจรูปเล่มแล้ว
(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจชิ้นงานแล้ว
(ลงชื่อ).....ผู้ตรวจ

ระบบบริหารจัดการพลังงานในอาคารพร้อมตรวจจับความเคลื่อนไหว

Energy Management with Movement Monitoring System

โดย

นางสาวพิมพ์อรุณ วัจฉลพงษ์ 46010057

นายเขตต์ ธนะสีลังกูร 46010078

นายอรรถพล จันพันเอก 46010937

อาจารย์ที่ปรึกษา

ผศ.ดร.พิพัฒน์ พรหมมี

อาจารย์ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบบริหารจัดการพลังงานในอาคารพร้อมตรวจจับความเคลื่อนไหว**

Energy Management with Movement Monitoring System

ผู้จัดทำ

1. นางสาวพิมพ์อรุณ วัจฉลพงษ์ 46010057

2. นายเขตต์ ธนะสีลังกูร 46010078

3. นายอรรถพล จันพันเอก 46010937

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.พิพัฒน์ พรหมมี)

..... อาจารย์ที่ปรึกษา

(อ.ศรวัฒน์ ชิวปรีชา)

ระบบบริหารจัดการพลังงานในอาคารพร้อมตรวจจับความเคลื่อนไหว

Energy Management with Movement Monitoring System

โดย นางสาวพิมพ์อรุณ วัจฉลพงษ์ 46010057
นายเขตต์ ธนะสีลังกูร 46010078
นายอรรถพล จันพันเอก 46010937

อาจารย์ที่ปรึกษา ผศ.ดร.พิพัฒน์ พรหมมี
อ.ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้มีจุดประสงค์เพื่อออกแบบระบบประหยัดพลังงาน ชนิดโปรแกรมได้ด้วยคอมพิวเตอร์ พร้อมกับอุปกรณ์เซนเซอร์ในการตรวจจับความเคลื่อนไหวของมนุษย์ ระบบจะทำงานโดยการตั้งเวลาเปิด/ปิด เครื่องใช้ไฟฟ้าในห้องต่างๆในเวลาที่กำหนด ในขณะเดียวกันระบบจะตรวจสอบความเคลื่อนไหวของห้องนั้นๆด้วย ในกรณีที่มีบุคคลใดอยู่ในห้องระบบก็จะทำงานตามที่ตั้งเวลาไว้ หากช่วงเวลานั้นไม่มีบุคคลใดอยู่ในห้องระบบจะทำการปิดระบบไฟฟ้าในห้องนั้น เพื่อเป็นการประหยัดพลังงานที่สูญเสียไปโดยไม่จำเป็น พร้อมกันนั้นระบบจะทำการเก็บข้อมูลการทำงานของแต่ละห้องไว้เพื่อใช้ในการทำรายงานสรุปค่าใช้จ่ายที่ลดลงไปอีกด้วย

Abstract

This project propose a design of Energy Management with Movement Monitoring System which is programmable and connected with sensors for detect the movement . The system is able to setting time to turn on/off the appliances in the several rooms. Meanwhile system also can be detected a human movement. The system works properly if there are humans movement in the room . Otherwise the system will turn off the appliances for saving energy. Additonally the system is able to make reports of reducing of electrical payments.

สารบัญ

บทที่ 1 บทนำ

1.1 แนวความคิดและที่มาของโครงงาน 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 องค์ประกอบหลักของโครงงาน 2

บทที่ 2 ทฤษฎีและหลักการ

2.1 อุปกรณ์ตรวจจับความเคลื่อนไหว 3

2.1.1 รังสีอินฟราเรด 3

2.1.2 ไพโรอิเล็กทริกเซนเซอร์ 3

2.1.3 สัญญาณรบกวนอุปกรณ์ตรวจจับ 6

2.1.4 ความถี่และแอมปลิจูดของสัญญาณเอาท์พุท 9

2.2 เลนส์เฟรสเนล 10

2.3 ไมโครคอนโทรเลอร์ 11

2.3.1 คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 11

2.3.2 การจัดการขาของไมโครกอนโทรลเลอร์ MCS-51 12

2.3.3 โครงสร้างและการทำงานของพอร์ต 12

2.3.4 จังหวะการทำงานของไมโครคอนโทรลเลอร์ 12

2.3.5 การจัดการหน่วยความจำ 13

2.4 การสื่อสารผ่านพอร์ตอนุกรม 19

2.4.1 รูปแบบการเชื่อมต่อผ่านพอร์ตสื่อสาร 19

2.4.2 รูปแบบการติดต่อสื่อสาร 20

2.4.3 การส่งข้อมูลในการสื่อสารแบบอนุกรม 21

2.4.4 การส่งข้อมูลแบบเข้าจังหวะเวลา (synchronous) 21

2.4.5 มาตรฐาน RS-232 22

2.4.6 การเชื่อมต่อ MCS-51 กับ RS-232 23

2.4.7 อัตราบอดในการสื่อสาร 23

2.4.8 การแปลงรูปแบบข้อมูล 24

บทที่ 3 การคำนวณและการสร้าง

3.1 โครงสร้างด้านฮาร์ดแวร์ 26

3.1.1 วงจรตรวจจับความเคลื่อนไหว โดยใช้อุปกรณ์ตรวจจับไพโรอิเล็กทริก 26

3.1.2 วงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์ 27

3.1.3 วงจรจ่ายไฟเลี้ยง 28

3.1.4 วงจรรับส่งข้อมูลอนุกรม 29

3.1.5 โครงสร้างการเชื่อมต่ออุปกรณ์ภายในระบบ 30

3.2 โครงสร้างด้านซอร์ฟแวร์ 31
3.2.1 โฟลว์ชาร์ท 31
3.2.1.1 โปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ 31
3.2.1.2 โปรแกรมควบคุมการทำงานของหลอดไฟโดยอัตโนมัติ 34
3.2.2 UML diagram 35
3.2.3 โครงสร้างของคลาส 35
3.2.3.1 คลาส FileManager 35
3.2.3.2 คลาส Calculation 36
3.2.3.3 คลาส TableReport 37
3.2.3.4 คลาส StatusChecker 37
3.2.3.5 คลาส SerialPort 37
3.2.3.6 คลาส Admin 38
3.2.4 โปรแกรม ESystem V3.0 (Engergy saving System) 38
3.2.4.1 การใช้งานโปรแกรมเบื้องต้น 38
บทที่ 4 การทดลองและผลการทดลอง
4.1 การทดลองวงจรตรวจจับความเคลื่อนไหว 44
4.2 การทดลองวงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์ 45
บทที่ 5 บทสรุปและวิจารณ์
5.1 สรุปผลการทดลอง 50
5.2 ปัญหาที่พบในการทดลอง 50
5.3 แนวทางพัฒนาต่อไป 51
หนังสืออ้างอิง 52
ภาคผนวก 53

สารบัญรูปภาพ

รูปที่ 2.1 แสดงต้นแบบของอุปกรณ์ตรวจจับ 4
รูปที่ 2.2 แสดงการขยายในขั้นตอนแรก 5
รูปที่ 2.3 แสดงการขยายในขั้นตอนที่สอง 5
รูปที่ 2.4 แสดงสัญญาณที่ได้รับการขยายแล้ว 6
รูปที่ 2.5 แสดงสัญญาณรบกวนของอุปกรณ์ตรวจจับความเคลื่อนไหว 7
รูปที่ 2.6 แสดงการแปลงสัญญาณกระแสสลับให้เป็นสัญญาณพัลส์ 8
รูปที่ 2.7 แสดงสัญญาณเอาท์พุทในรูปพัลส์ 8
รูปที่ 2.8 แสดงตัวอย่างรูปกราฟค่าการดอบสนองของ LHi98 9
รูปที่ 2.9 แสดงสัญญาณเอาท์พุทที่มีรูปเป็นสัญญาณแรงดันกระแสสลับ 10
รูปที่ 2.10 แสดงลักษณะของเลนส์เฟรสเนล 10
รูปที่ 2.11 โครงสร้างหน่วยความจำ 14
รูปที่ 2.12 แสดงการสื่อสารแบบขนาน 19
รูปที่ 2.13 แสดงการสื่อสารแบบอนุกรม 20
รูปที่ 2.14 แสดงการสื่อสารแบบซิมเพล็กซ์ 20
รูปที่ 2.15 แสดงการสื่อสารแบบฮาล์ฟดูเพล็กซ์ 20
รูปที่ 2.16 แสดงการสื่อสารแบบฟูลล์ดูเพล็กซ์ 21
รูปที่ 2.17 แสดงลักษณะสัญญาณที่ใช้ในการส่งรหัส ASCII ของตัว “A” 22
รูปที่ 2.18 แสดงพอร์ตสื่อสารสำหรับคอนเนคเตอร์แบบ DB-9 23
รูปที่ 2.19 แสดงการแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม 24
รูปที่ 3.1 แสดงวงจรตรวจจับความเคลื่อนไหว โดยใช้อุปกรณ์ตรวจจับไพโรอิเล็กทริก 26
รูปที่ 3.2 แสดงวงจรตรวจจับความเคลื่อนไหวที่ใช้งานจริง 27
รูปที่ 3.3 แสดงวงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์ 27
รูปที่ 3.4 แสดงวงจรจ่ายไฟเลี้ยง 28
รูปที่ 3.5 แสดงวงจรรับส่งข้อมูลอนุกรม 29
รูปที่ 3.6 แสดงวงจรควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าที่ใช้งานจริง 29
รูปที่ 3.7 แสดงโครงสร้างการเชื่อมต่ออุปกรณ์ภายในระบบ 30
รูปที่ 3.8 แสดง การทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (1) 31
รูปที่ 3.9 แสดง การทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (2) 32
รูปที่ 3.10 แสดง การทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (3) 33
รูปที่ 3.11 แสดง การทำงานของโปรแกรมควบคุมการทำงานของหลอดไฟโดยอัตโนมัติ 34
รูปที่ 3.12 UML diagram 35

รูปที่ 3.13 แสดงหน้าต่างหน้าหลักของระบบ 39
รูปที่ 3.14 แสดงหน้าต่างตารางเวลาควบคุม 40
รูปที่ 3.15 แสดงหน้าต่างประวัติการส่งผ่านข้อมูล 41
รูปที่ 3.16 แสดงหน้าต่างสถิติการใช้ไฟฟ้า 42
รูปที่ 3.17 แสดงหน้าต่างกราฟ 43
รูปที่ 4.1 แสดงสัญญาณเอาท์พุท 0 โวลต์ 44
รูปที่ 4.2 แสดงสัญญาณเอาท์พุท 5 โวลต์ 45
รูปที่ 4.3.1 ก่อนทำคำสั่ง "เปิดอุปกรณ์ไฟฟ้าทุกห้อง" วันศุกร์ที่ 19 มกราคม 2550 เวลา 8.00 น. 46
4.3.2 หลังทำคำสั่งแล้วสถานะเครื่องใช้ไฟฟ้าห้องที่ 1 และ 2 จากปิดเปลี่ยนเป็นเปิด 46
รูปที่ 4.4 แสดงสถานะหลังจากเซนเซอร์ไม่สามารถตรวจจับการเคลื่อนไหวได้ภายใน 1.30 วินาที 47
รูปที่ 4.5 แสดงสถานะหลังจากเซนเซอร์สามารถตรวจจับการเคลื่อนไหวภายในห้องที่ 1 ได้ 48
รูปที่ 4.6 แสดงสถานะหลังจากทำคำสั่ง (จากข้อที่ 3) "ปิดอุปกรณ์ไฟฟ้าทุกห้อง"
ณ เวลา 12.00 น. วันศุกร์ 19 มกราคม 2550 49

สารบัญตาราง

ตารางที่ 2.1 แสดงชนิดของอุปกรณ์ที่นำไปประยุกต์ใช้งานในแบบต่างๆร่วมกับกระจก
หรือเฟรสเนลเลนส์ หลายหน้าตัด 3
ตารางที่ 2.2 การเลือกแบงค์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงค์ R0-R7 16
ตารางที่ 2.3 แสดงแสดงอัตราบอดของ UART ที่ใช้กันทั่วไปในการโอนถ่ายข้อมูลแบบอนุกรม 23

บทที่ 1
บทนำ

1.1 แนวความคิดและที่มาของโครงงาน

ตั้งแต่อดีตจวบจนถึงปัจจุบัน พลังงานเป็นสิ่งจำเป็นต่อการดำรงชีวิตของมนุษย์ ไม่ว่าจะเป็นพลังงานที่ใช้แล้วหมดไปหรือพลังงานหมุนเวียนก็ตาม " พลังงาน " ถือเป็นสิ่งที่มีค่าและมีความจำเป็นต่อการดำรงชีวิตประจำวันของเราทุกคนไม่ว่าจะเป็นพลังงานน้ำ พลังงานน้ำมัน พลังงานไฟฟ้า ซึ่งความต้องการใช้พลังงานมีอัตราที่เพิ่มมากขึ้นทุก ๆ ปี ในขณะที่พลังงานต่าง ๆ มีอยู่อย่างจำกัด ดังนั้นจึงจำต้องหาวิธีประหยัดพลังงานหรือใช้พลังงานที่มีอยู่อย่างคุ้มค่า

พลังงานไฟฟ้าเป็นพลังงานรูปแบบหนึ่งที่มีความสัมพันธ์กับชีวิตประจำวันของเรามากที่สุด ไม่ว่าจะในบ้าน ในรถยนต์ ในโรงเรียน สถานที่ทำงาน ตามถนนหนทาง หรือโรงงานต่างๆล้วนแต่ใช้พลังงานไฟฟ้าทั้งสิ้นและในปัจจุบันเทคโนโลยีต่างๆได้มีการพัฒนาอย่างไม่หยุดยั้งเพื่อตอบสนองความต้องการของมนุษย์ เทคโนโลยีเหล่านั้นสามารถนำมาประยุกต์ใช้ในการประหยัดพลังงานได้ ยกตัวอย่าง เช่น ระบบการจัดการพลังงาน (Energy Management System)

ตาม พ.ร.บ. การส่งเสริมการอนุรักษ์พลังงาน พ.ศ. 2535 มาตรา 3 **ระบบการจัดการพลังงาน (Energy Management System)** หมายถึง ระบบภายใน สถานประกอบการ ซึ่งประกอบด้วย บุคลากร ทรัพยากร นโยบายและขั้นตอนการดำเนินการโดยมีการทำงานประสานกันอย่างมีระเบียบ และมีแบบแผน เพื่อปฏิบัติงานที่กำหนดไว้ หรือเพื่อให้บรรลุ หรือรักษาเป้าหมายที่กำหนดไว้ ให้มีประสิทธิผลและเกิดประสิทธิภาพ

โครงงานนี้นำเสนอระบบบริหารจัดการพลังงานภายในอาคาร ซึ่งได้นำเอาเทคโนโลยีของอุปกรณ์ตรวจจับความเคลื่อนไหวเพื่อเปิด-ปิดอุปกรณ์ไฟฟ้าอัตโนมัติ สำหรับห้องที่ไม่มีการใช้งานตลอดเวลา เช่น ห้องเรียนหรือห้องประชุม อีกทั้งควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าอัตโนมัติตามเวลาเพื่อป้องกันการลืมปิดไฟในช่วงพักเที่ยงหรือเลิกงาน เพื่อลดปริมาณการใช้พลังงานไฟฟ้า

1.2 วัตถุประสงค์ของโครงงาน

(1) เพื่อศึกษาการทำงานของเซนเซอร์ตรวจจับความเคลื่อนไหว ไพโรอิเล็กทริกเซนเซอร์ (PIR : Pyroelectric infrared sensor) และไมโครคอนโทรเลอร์ (microcontroller)

(2) เพื่อทำการออกแบบวงจรตรวจจับความเคลื่อนไหว เพื่อเปิด-ปิดอุปกรณ์ไฟฟ้าอัตโนมัติ

(3) เพื่อจัดการพลังงานให้เหมาะสมและลดต้นทุนด้านพลังงานของสถานประกอบการให้ต่ำที่สุด

1.3 องค์ประกอบหลักของโครงงาน

(1) ส่วนของการศึกษาค้นคว้า

ในส่วนนี้ประกอบด้วยการศึกษาค้นคว้าทฤษฎีต่างๆที่เกี่ยวข้องกับโครงงาน ไม่ว่าจะเป็นเซนเซอร์ตรวจจับความเคลื่อนไหว ไมโครคอนโทรเลอร์ และพอร์ตสื่อสารอนุกรม เป็นต้น

(2) ส่วนของการเขียนโปรแกรมคอมพิวเตอร์

โครงงานนี้ได้ใช้โปรแกรมภาษาซีในการออกแบบและควบคุมส่วนของอุปกรณ์ไมโครคอนโทรเลอร์และการเชื่อมต่ออุปกรณ์ไมโครคอนโทรเลอร์และคอมพิวเตอร์ผ่านพอร์ตสื่อสารอนุกรม และโครงงานนี้ได้ใช้โปรแกรม Visual C# ในการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าอัตโนมัติตามเวลา เนื่องจากสามารถพัฒนาได้ง่ายและมีความซับซ้อนต่ำ

(3) ส่วนของการออกแบบวงจร

ในส่วนนี้จะเป็นการออกแบบวงจรของอุปกรณ์เปิด-ปิดไฟฟ้าอัตโนมัติ อันประกอบด้วยวงจรตรวจจับความเคลื่อนไหว, วงจรไมโครคอนโทรเลอร์ และวงจรสื่อสารอนุกรม

บทที่ 2
ทฤษฎีและหลักการ

2.1 อุปกรณ์ตรวจจับความเคลื่อนไหว

อุปกรณ์ตรวจจับความเคลื่อนไหวถูกนำไปประยุกต์ใช้อย่างกว้างขวางในด้านการรักษาความปลอดภัย ไม่ว่าจะเป็นอุปกรณ์เตือนภัยจากผู้บุกรุก เครื่องเปิดประตูหรือเปิดไฟอัตโนมัติ เพื่อเพิ่มความสะดวกสบายและความปลอดภัยให้ดียิ่งขึ้น

อุปกรณ์ทุกๆชิ้นที่กล่าวมาแล้วนั้นล้วนแต่มีส่วนการทำงานของอุปกรณ์ตรวจจับไพโรอิเล็กทริก ดังตาราง 2.1

ชนิด	การประยุกต์งาน
LHi 878	สวิตช์ไฟ
LHi 958	สัญญาณเตือนผู้บุกรุกแบบพื้นฐาน
LHi 968	สัญญาณเตือนผู้บุกรุกประสิทธิภาพสูง
LHi 906	อุปกรณ์ตรวจจับติดบนเพดาน

ตารางที่ 2.1 แสดงชนิดของอุปกรณ์ที่นำไปประยุกต์ใช้งานในแบบต่างๆร่วมกับกระจกหรือเฟรสเนลเลนส์ หลายหน้าตัด

2.1.1 รังสีอินฟราเรด

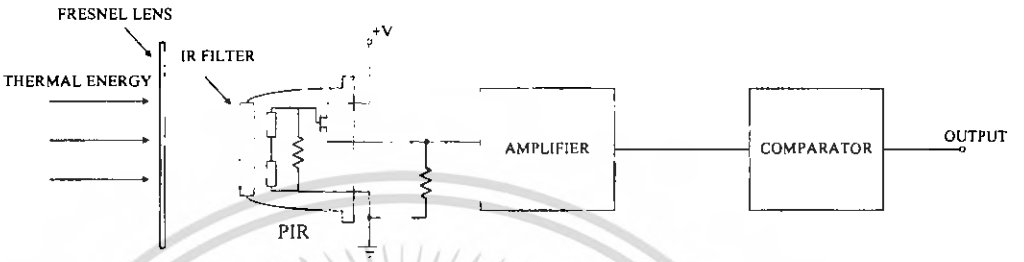
รังสีอินฟราเรดเป็นสเปกตรัมแม่เหล็กไฟฟ้าที่มีความยาวคลื่นยาวกว่าแสงที่ตามองเห็นรังสีชนิดนี้ไม่สามารถมองเห็นได้ด้วยตาเปล่าแต่สามารถตรวจจับได้ วัตถุที่ก่อกำเนิดความร้อนก็จะให้รังสีอินฟราเรดด้วยเช่นกัน สิ่งมีชีวิตต่างๆก็ก่อกำเนิดรังสีอินฟราเรดด้วยเช่นกันรวมไปถึงร่างกายมนุษย์ ซึ่งจะแผ่รังสีอินฟราเรดที่มีความแรงสูงสุดที่ความยาวคลื่น 9.4 ไมโครเมตร

2.1. 2 ไพโรอิเล็กทริกเซนเซอร์

ไพโรอิเล็กทริกเซนเซอร์สร้างขึ้นจากวัสดุจำพวกคริสตอลซึ่งจะสร้างประจุไฟฟ้าบนพื้นผิวเมื่อมีพลังงานความร้อนในรูปรังสีอินฟราเรดตกกระทบ เมื่อปริมาณของรังสีอินฟราเรดมีการเปลี่ยนแปลงจะส่งผลให้คริสตอลเกิดการเปลี่ยนแปลง กล่าวคือปริมาณประจุไฟฟ้าจะเปลี่ยนแปลง อันสามารถตรวจวัดค่า

ได้ด้วยอุปกรณ์ FET ที่มีความไวต่อการเปลี่ยนแปลงสูง ซึ่งถูกสร้างขึ้นมาภายในอุปกรณ์ตรวจจับ โดยเฉพาะ อุปกรณ์ตรวจจับนี้มีความไวสูงในช่วงแถบความถี่ค่อนข้างกว้าง จึงจำต้องมีตัวกรองเพื่อจำกัดให้รังสีที่แผ่ออกมาอยู่ในช่วง 8 ถึง 14 ไมโครเมตร ซึ่งเป็นช่วงความถี่ที่ร่างกายมนุษย์แผ่ออกมา

TYPICAL CONFIGURATION



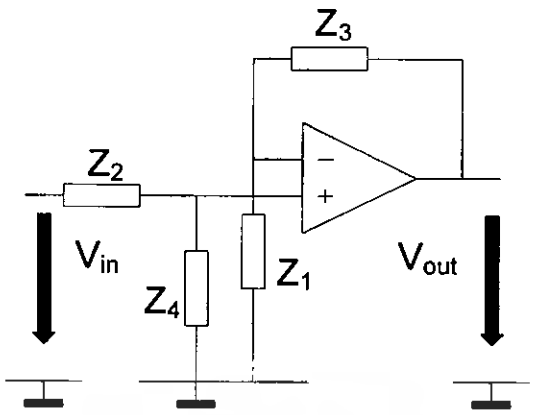
รูปที่ 2.1 แสดงต้นแบบของอุปกรณ์ตรวจจับ

จากรูป 2.1 ที่ปลายแหล่งกำเนิด FET ขา source เชื่อมต่อกับตัวต้านทานพูลดาวน์ประมาณ 100 กิโลโอห์ม ไปยังกราวน์ นำไปป้อนวงจรขยายสองระดับ โดยแต่ละระดับที่ต่อกันนั้นมีอัตราขยายประมาณ 100 เท่า เป็นผลให้อัตราขยายรวมมีค่าประมาณ 10000 เท่า วงจรขยายนั้นโดยทั่วไปแล้วจะทำการขยายสัญญาณในช่วงความถี่จำกัดต่ำกว่า 10 Hz เพื่อขจัดสัญญาณรบกวนที่มีความถี่สูง จากนั้นนำไปเชื่อมต่อกับวงจรเปรียบเทียบสัญญาณ (comparator) ซึ่งตอบสนองสัญญาณเอาท์พุทของอุปกรณ์ทั้งการเปลี่ยนแปลงทางด้านบวกและทางด้านลบ ส่วนขา drain ของ FET นั้นเชื่อมต่อกับแหล่งจ่ายไฟ 3-5 โวลต์

อุปกรณ์ตรวจจับนี้จะสามารถทำงานได้โดยการจัดหาแหล่งกำเนิดแรงดันไฟฟ้า (voltage source) ที่ขา drain และจะได้สัญญาณออกมาทางขา source ซึ่งจำเป็นต้องมีตัวต้านทานขนาด 47 กิโลโอห์มต่ออยู่ด้วย วงจรขยายนี้ให้ประโยชน์ใน 2 ด้านคือสัญญาณได้รับการขยายและยังเป็นวงจรกรองแถบความถี่ซึ่งจะกำจัดสัญญาณในส่วนที่ไม่ต้องการภายนอกย่านความถี่ในการทำงานได้

สัญญาณเอาท์พุทของอุปกรณ์ตรวจจับพื้นฐานจะอยู่ในช่วง 0.5-1 มิลลิโวลต์ ขณะตรวจวัดเป้าหมายและสัญญาณนี้จะถูกนำไปขยายก่อนที่จะนำไปประมวลผล ซึ่งในปัจจุบันนิยมใช้ออบแอมป์แม้แต่ในการใช้งานที่มีค่ากระแสต่ำมากๆซึ่งก็สามารถใช้ออบแอมป์ชนิดพิเศษที่ใช้พลังงานต่ำมากๆได้ ซึ่งในส่วนของการขยายสัญญาณนี้ต้องใช้วงจรขยาย 2 ขั้นตอน

ยกตัวอย่างเช่นถ้าต้องการสัญญาณที่นำไปประมวลผลต่อในระดับแรงดัน TTL สัญญาณต้องได้รับการขยายมากสุดถึง 10000 เท่าแต่ที่นิยมใช้กันจะมีค่าประมาณ 3000 ถึง 5000 เท่า ซึ่งจะต้องทำการขยายใน 2 ขั้นตอน

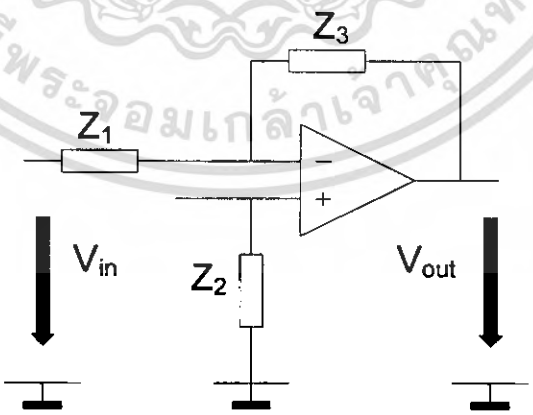


รูปที่ 2.2 แสดงการขยายในขั้นตอนแรก

ออปแอมป์ตัวแรกควรใช้ในโหมด non-inverting ข้อดีของโหมดนี้ก็คือเป็นอิสระต่อความต้านทานภายในตัวออบแอมป์เองจึงไม่ต้องมีการปรับเปลี่ยนรูปแบบใดๆระหว่างตัวตรวจจับและวงจรขยาย ย่านความถี่ที่ต้องการสามารถกำหนดได้จากค่า RC ซึ่งอัตราการขยายของวงจรดังกล่าวเขียนเป็นสมการได้ดังนี้

$$\frac{V_{out}}{V_{in}} = \frac{Z_4(Z_1 + Z_3)}{Z_1(Z_2 + Z_4)} \qquad (2.1)$$

$$Z_{in} = Z_2 + Z_4$$

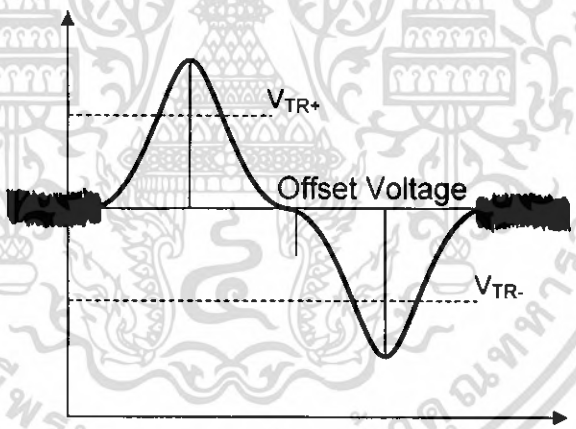


รูปที่ 2.3 แสดงการขยายในขั้นตอนที่สอง

ในการขยายขั้นที่สองนี้อาจออกแบบให้อยู่ในโหมด inverting โดยใช้คับปลิ้ง RC (RC coupling) ในการเชื่อมต่อมายังขั้นตอนนี้เพื่อที่จะแยกส่วนที่เป็น สัญญาณไฟกระแสตรงออกไปและยังคงความถี่ให้อยู่ในช่วงที่ต้องการได้และย่านความถี่ที่ต้องการสามารถกำหนดได้จากค่า RC ซึ่งอัตราการขยายของวงจรดังกล่าวเขียนเป็นสมการได้ดังนี้

$$\frac{V_{out}}{V_{in}} = \frac{Z_3}{Z_1}$$
$$Z_{in} = Z_1 \qquad (2.2)$$

การออกแบบส่วนใหญ่ในปัจจุบันจะเป็นวงจรอนาลอกในกรณีนี้วงจรเปรียบเทียบสัญญาณ (comparator) ถูกนำมาใช้ในการแยกสัญญาณการเคลื่อนไหวที่ต้องการออกจากสัญญาณรบกวนและการเคลื่อนไหวของอากาศ สัญญาณที่ถูกขยายมาแล้วนั้นประกอบด้วยแรงดันไฟฟ้ากระแสสลับขี่อยู่บนแรงดันไฟฟ้ากระแสตรงซึ่งเป็นระดับอ้างอิงระหว่างขนาดของสัญญาณ(amplitude) ทั้งฝั่งบวกและลบ ดังรูปที่ 2.4



รูปที่ 2.4 แสดงสัญญาณที่ได้รับการขยายแล้ว

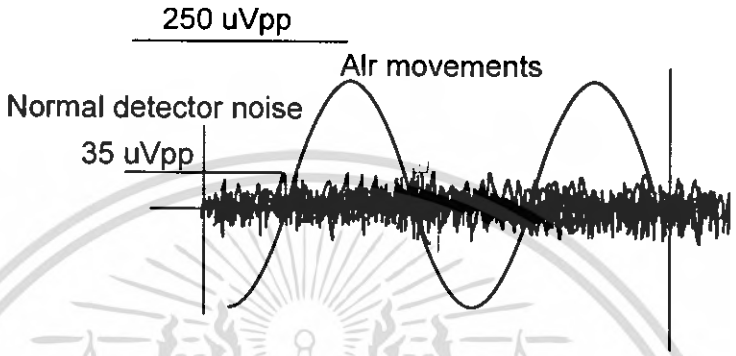
สิ่งที่ต้องพิจารณาในการออกแบบวงจรเปรียบเทียบสัญญาณคือ ระดับทริกเกอร์ (trigger level) ซึ่งมีความเกี่ยวข้องกับสัญญาณรบกวน

2.1.3 สัญญาณรบกวนอุปกรณ์ตรวจจับ

สัญญาณรบกวนอุปกรณ์ตรวจจับได้แก่สัญญาณรบกวนพื้นฐานจากตัวอุปกรณ์เองซึ่งมีค่าประมาณ 250 ไมโครโวลต์ สัญญาณรบกวนที่สำคัญอีกส่วนหนึ่งคือการเคลื่อนไหวของอากาศ

จากการทดลองพบว่าภายนอกอาคารการเคลื่อนไหวของอากาศที่มีอุณหภูมิสูงจะสร้างสัญญาณสูงกว่า 250 ไมโครโวลต์ ส่วนภายในอาคารจะสร้างสัญญาณต่ำกว่า 180 ไมโครโวลต์ สัญญาณรบกวนอื่นๆได้แก่สัญญาณรบกวนคลื่นแม่เหล็กไฟฟ้าและการกระแทกเชิงกลความแรงของ สัญญาณรบกวนดังกล่าวขึ้นอยู่กับความแรงของแรงดันไฟฟ้าและกำลังที่ใช้ ซึ่งอาจมีค่าสูงถึง 100 ไมโครโวลต์

ระดับทริกเกอร์อ้างอิงจากค่าสูงสุดของสัญญาณรบกวนที่พิจารณา ยกตัวอย่างเช่น 250 ไมโครโวลต์ ดังรูป 2.5



รูปที่ 2.5 แสดงสัญญาณรบกวนของอุปกรณ์ตรวจจับความเคลื่อนไหว

ที่ระดับนี้เราจำเป็นต้องทำการขยายสัญญาณเพื่อให้ได้สัญญาณกระแสสลับในการออกแบบวงจรเปรียบเทียบสัญญาณ เมื่อได้ชดเชยระดับไฟตรง กำหนดด้วยซีเนอร์ไดโอดให้มีค่า 4.0 โวลต์ สามารถคำนวณระดับทริกเกอร์ได้ดังสมการ

$$V_{Trigger} = V_{Disturb} \times Amplification \tag{2.3}$$

สมมติให้วงจรขยายมีอัตราขยาย 3100 เท่า จะได้

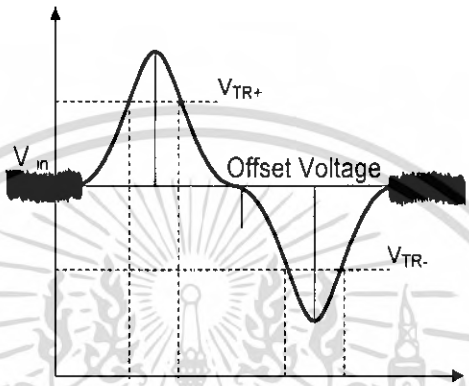
$$\begin{aligned} V_{Trigger} &= V_{Disturb} \times Amplification \\ &= 250\mu\text{V} \times 3100 \\ &= 775\ \text{mV} \end{aligned} \tag{2.4}$$

จากสมการข้างต้น จะได้ระดับแรงดันทริกเกอร์ มีค่าดังนี้

$$\begin{aligned} V_{TrL} &= V_{Offset} +/- V_{Trigger} \\ &= 4\text{V} +/- 775\text{mV} \end{aligned} \tag{2.5}$$

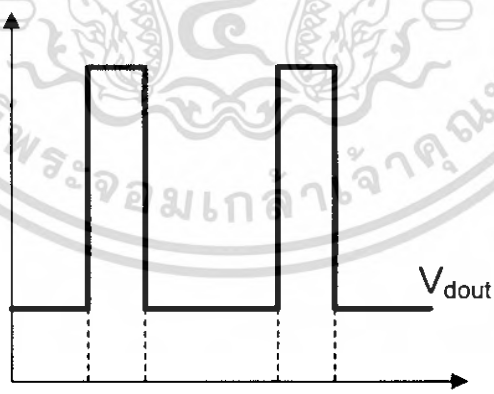
$$V_{TL+} = 4,775\text{V}$$
$$V_{TL-} = 3,225\text{V}$$

จากระดับแรงดันทริกเกอร์ข้างต้น จะนำไปสู่การแปลงสัญญาณกระแสสลับให้เป็นสัญญาณพัลส์ ได้ดังรูป 2.6



รูปที่ 2.6 แสดงการแปลงสัญญาณกระแสสลับให้เป็นสัญญาณพัลส์

จากรูปแสดง สัญญาณที่เข้ามาเปรียบเทียบกับระดับทริกเกอร์และแสดงถึงส่วนของสัญญาณที่มีค่าเกินระดับทริกเกอร์ ซึ่งจะได้สัญญาณเอาท์พุทในรูปพัลส์ ดังรูป 2.7



รูปที่ 2.7 แสดงสัญญาณเอาท์พุทในรูปพัลส์

2.1.4 ความถี่และแอมปลิจูดของสัญญาณเอาท์พุท

สิ่งที่มีผลต่อความถี่และแอมปลิจูดของสัญญาณเอาท์พุท ได้แก่

- ระยะห่างระหว่างร่างกายมนุษย์กับเซนเซอร์ตรวจจับความเคลื่อนไหว
- ความเร็วในการเดิน
- ระยะโฟกัสและรูปแบบของระบบเชิงแสง

สามารถเขียนความสัมพันธ์ดังกล่าวในรูปสมการ ดังนี้

$$f = \frac{\nu_b * f_b}{2\pi * S * L} \qquad (2.6)$$

f_b = focal length (mm)

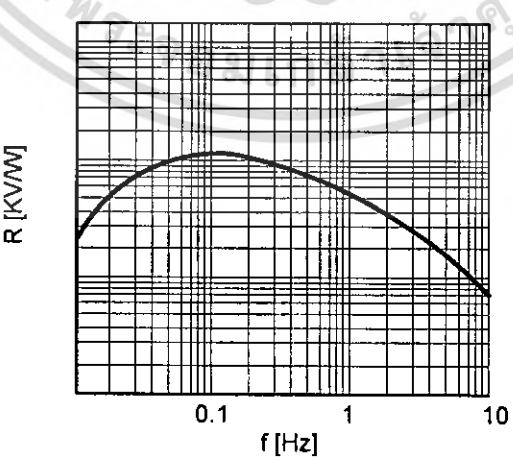
f = freguency (Hz)

ν_b = velocity,walking speed (m/s)

S = size of sensing element

จากความสัมพันธ์ดังกล่าวพบว่าโดยทั่วไปแล้วช่วงความถี่จะมีค่าเท่ากับ 0.08-8 Hz เมื่อใช้งานกับ เลนส์เฟรสเนลที่มีระยะโฟกัส 25 mm ซึ่งช่วงความถี่สูงๆได้จากการวัดระยะใกล้ประชิดอุปกรณ์ และ ช่วงความถี่ต่ำๆได้จากการวัดระยะไกล ระยะไกลนั้นต้องการความไวสูงสุดซึ่งจะให้ความถี่ประมาณ 0.1 Hz

ความไว(sensitivity)เป็นคำหนึ่งในนิยามของอุปกรณ์ตรวจวัดเชิงแสงและถูกนำไปใช้อ้างอิงในรูปของค่าการตอบสนอง(responsibility) อันหมายความถึงค่าความไวต่อพื้นที่ ในเงื่อนไขเฉพาะใดๆ



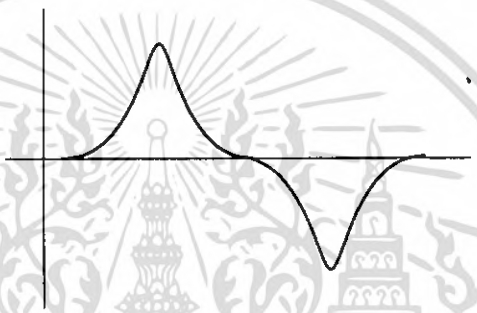
Freguency response Lhi 968

รูปที่ 2.8 แสดงตัวอย่างรูปกราฟค่าการตอบสนองของ LHi98

จากกราฟแสดงให้เห็นว่าค่าความไวสูงสุดอยู่ที่ความถี่ประมาณ 0.1 Hz ซึ่งยินยอมการตรวจจับความเคลื่อนไหวในระยะไกลๆได้ ถ้าการเคลื่อนไหวของร่างกายมนุษย์เข้าใกล้ตัวอุปกรณ์มากขึ้น กำลังงานที่แผ่ออกมายัง PIR จะมีค่าสูงขึ้น สรุปได้ว่าอุปกรณ์ตรวจจับชิ้นนี้เหมาะสมที่จะใช้งานในย่านความถี่ดังกล่าวอย่างไรก็ตามการเลือกอุปกรณ์ตรวจจับที่เหมาะสมที่สุดนั้นต้องพิจารณาจากปัจจัยอื่นนอกเหนือจากค่าความไวด้วย เช่นความปลอดภัยต่อสิ่งรบกวนต่างๆ เป็นต้น

ช่วงการทำงานและระยะโฟกัสของระบบเชิงแสงเป็นส่วนหลักที่กำหนดช่วงความถี่ของเซนเซอร์ตรวจจับความเคลื่อนไหวในทุกการใช้งาน การเลือกระบบเลนส์ที่เหมาะสมจึงเป็นสิ่งสำคัญ

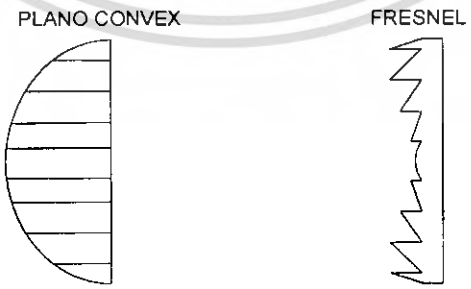
ระบบเลนส์เป็นสิ่งที่กำหนดขอบเขตการทำงานของเซนเซอร์ เมื่อมีคนเดินผ่าน อุปกรณ์ตรวจจับจะสร้างสัญญาณเอาท์พุทที่มีรูปเป็นสัญญาณแรงดันกระแสสลับ ดังรูป 2.9



รูปที่ 2.9 แสดงสัญญาณเอาท์พุทที่มีรูปเป็นสัญญาณแรงดันกระแสสลับ

2.2 เลนส์เฟรสเนล

เลนส์เฟรสเนลมีลักษณะราบข้างหนึ่งและเว้าข้างหนึ่งโดยแต่ละร่องของเลนส์จะเป็นพื้นผิวชิ้นเล็กๆที่เบี่ยงเบนไปจากรูปทรงกลมเล็กน้อยและปราศจากการพร่ามัว ความลาดเอียงของแต่ละพื้นผิวต้องถูกทำให้บางขึ้นจากพื้นผิวแบบเดิมเพื่อชดเชยการดัดแปลงนี้ให้ยังคงคุณลักษณะของอุปกรณ์เชิงแสงและลดความสูญเสียจากการดูดกลืน ดังรูป 2.10



รูปที่ 2.10 แสดงลักษณะของเลนส์เฟรสเนล

FL65 เป็นตัวอย่างหนึ่งของเลนส์เฟรสเนล ที่ถูกสร้างจากวัสดุที่ยินยอมให้รังสีอินฟราเรดผ่านในช่วง 8 ถึง 14 ไมโครเมตร ซึ่งเป็นช่วงที่มีความไวที่สุดของรังสีที่แผ่ออกมาจากร่างกายมนุษย์ อุปกรณ์ชิ้นนี้ถูกออกแบบให้ด้านที่เป็นร่องหันเข้าหาอุปกรณ์ตรวจจับรังสีอินฟราเรด ดังนั้นจะหันด้านที่เรียบเข้าหาวัตถุ

FL65 มีระยะโฟกัส 0.65 นิ้วจากเลนส์ถึงอุปกรณ์ตรวจจับ ตัวเลนส์นั้นมีเส้นผ่านศูนย์กลาง 1 นิ้ว และมีพื้นที่ขอบ 1.5 ตารางนิ้ว ซึ่งขอบนี้ใช้สำหรับค้ำจุนเลนส์ในกรอบที่เหมาะสม สิ่งค้ำจุนที่ดีและง่ายที่สุดคือแถบสกอตช์แทบ ยางซิลิโคนที่มีความเหนียวก็สามารถใช้ในการป้องกันน้ำได้

2.3 ไมโครคอนโทรเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อีเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุท วงจรขับสัญญาณเอาท์พุท หน่วยคำนวณ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอีเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันก็คือ "ไมโคร" (Micro) ซึ่งหมายถึง ไมโครโปรเซสเซอร์ (Microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วยหน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU: Arithmatic Logic Unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า "คอนโทรลเลอร์" (Controller) หมายถึง อุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุมโดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

2.3.1 คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล MCS-51

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียู ขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลช สามารถลบและเขียนใหม่ได้
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความ จำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทางสามารถใช้งานได้ทั้งอินพุทและเอาท์พุท
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ในภายในชิป

- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอทช์ด็อกไทเบอร์ในตัว สำหรับในอนุกรม AT89Sxx

2.3.2 การจัดการขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกันโดยมีรายละเอียดขั้นต้น ดังนี้

- ขา VCC ใช้สำหรับต่อไฟเลี้ยง + 5V
- ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 (P0.0 – P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุทและเอาท์พุท สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุทสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (Float) จึงมีอินพุทอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุทได้

2.3.3 โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรเลอร์ MCS – 51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ พอร์ต 0 ถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลและเอาท์พุทสำหรับส่งสัญญาณข้อมูลออก พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาท์พุทสำหรับงานทั่วไปและใช้ในการติดต่อหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขา นอกจากจะใช้เป็นขาพอร์ตอินพุทเอาท์พุทตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก

2.3.4 จังหวะการทำงานของไมโครคอนโทรลเลอร์

ในการใช้งานไมโครคอนโทรลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะการทำงานของซีพียูและลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของซีพียูจะมีขั้นตอนหลักๆ 2 ขั้นตอนคือ กระบวนการเฟทซ์ (fetch) เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผลขั้นตอนต่อมาคือ กระบวนการเอ็กซิคิวท์ (execute) เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตซ์ขึ้นมาโดยกระบวนการก่อนหน้านี้เมื่อทำการเอ็กซิคิวต์คำสั่งเรียบร้อยแล้วก็จะไปเริ่มกระบวนการเฟทซ์คำสั่งใหม่ต่อไป เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์จะ

เกิดการรีเซตเกิดขึ้นในลักษณะที่เรียกว่า เพาเวอร์ออนรีเซต (Power on Reset) ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม

จังหวะการทำงานของซีพียูจะเป็นไปตามรูปแบบโดยได้รับการกำหนดมาจากรอบการทำงานหรือแมชชีนไซเกิล (Machine Cycle) ใน 1 รอบการทำงานหรือในหนึ่งแมชชีนไซเกิลจะแบ่งย่อยออกเป็น 6 สเตท (State) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตทมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms

2.3.5 การจัดการหน่วยความจำ

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำภายในหลัก ๆ อยู่ 2 ส่วน คือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์

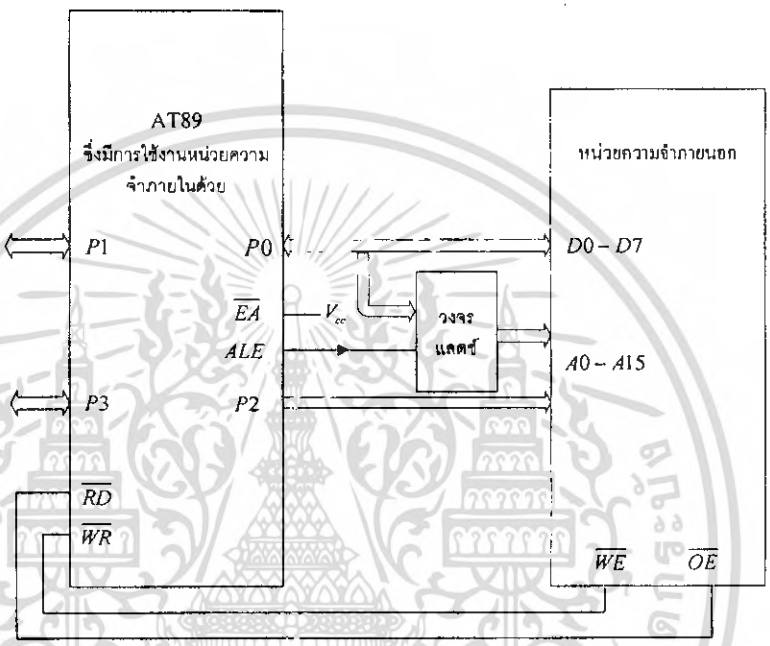
2.3.5.1 หน่วยความจำโปรแกรม (Program Memory)

หน่วยความจำของโปรแกรมที่ใช้ในการเก็บข้อมูลของโปรแกรมควบคุมการทำงานของไม-โครคอนโทรลเลอร์หรือที่เรียกว่าโปรแกรมมอนิเตอร์ (Monitor Program) ถ้าหากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม(EPROM : Erasable Programmable Read-only Memory) ซึ่งสามารถทำการอ่านได้เพียงอย่างเดียว กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51แบบแฟลชที่มีหน่วยความจำโปรแกรมภายในแต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์ ยกตัวอย่างไมโครคอนโทรลเลอร์AT89C51 มีหน่วยความจำโปรแกรมขนาด 4 กิโลไบต์ มีแอดเดรสอยู่ระหว่าง 0000H – 0FFFH เมื่อต่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H-FFFFH

2.3.5.2 หน่วยความจำข้อมูล (Data Memory)

หน่วยความจำข้อมูล มีด้วยกัน 2 แบบ คือ หน่วยความจำข้อมูลภายนอกและหน่วยความจำข้อมูลภายในโดยไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ในอนุกรม AT89 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก การติดต่อกับหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชแสดงดังในรูป 2.11 จะเห็นได้ว่า มีลักษณะคล้ายกับการติดต่อกับหน่วยความจำโปรแกรมภายนอก แตกต่างกันที่มีสัญญาณที่ใช้สำหรับการอ่านและเขียนหน่วยความจำข้อมูลภายนอก นั่นคือ ขา RD และ WR สำหรับไมโครคอนโทรลเลอร์ MCS-

51 แบบแฟลช ในอนุกรม AT89 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม (RAM : Random Access Memory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่าง (Lower)และส่วนบน (Upper)และรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR : Special Function Register) แต่ละส่วนมีขนาด128 ไบต์



รูปที่ 2.11 โครงสร้างหน่วยความจำ

ขนาดของหน่วยความจำข้อมูลของไมโครโทรลเลอร์ MCS-51 แบบแฟลช โดยแท้จริงและมีเพียง 256 ไบต์ แต่ด้วยการจัดการเข้าถึงที่แตกต่างกัน จึงดูเสมือนว่า ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีหน่วยความจำข้อมูลภายในสูงถึง 384 ไบต์ โดยในหน่วยความจำข้อมูลส่วนล่างขนาด 128 ไบต์ มีแอดเดรสอยู่ที่ 00H-7FH สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128 ไบต์เช่นกัน มีแอดเดรสอยู่ที่ 80H-FFH สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H-FFH เช่นเดียวกับหน่วยความจำข้อมูลส่วนบน แต่สำหรับรีจิสเตอร์ SFR ใช้การเข้าถึงแบบโดยตรง ดังนั้นเพื่อความสะดวกและง่าย ตลอดจนป้องกันความสับสนในการเขียนโปรแกรมสำหรับผู้เริ่มต้น จึงควรใช้หน่วยความจำข้อมูลภายในเพียง 128 ไบต์ จากหน่วยความจำข้อมูลส่วนล่างร่วมกับรีจิสเตอร์ SFR หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H-2FH เป็นพื้นที่สำหรับใช้งานทั่วไป

สามารถเข้าถึงได้ในระดับบิต(Stack:ที่พักข้อมูลชั่วคราวในกรณีที่ซีพียูมีการกระทำโดยไปทำงานในโปรแกรมย่อย) การเข้าถึงหน่วยความจำในส่วนนี้ต้องใช้การเข้าถึงระดับไบต์

2.3.5.3 รีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR)

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 มีด้วยกัน 22 ตัว สำหรับเบอร์ AT89C51 รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H-FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงได้โดยตรง (Direct Addressing)

2.3.5.4 รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word : PSW)

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต จึงสามารถกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ มีแอดเดรสอยู่ที่ D0H ทำหน้าที่เก็บสถานะของการทำงานของโปรแกรมในขณะนั้นจะเรียกสถานะต่าง ๆ ของโปรแกรม เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงนั้นจะมาปรากฏที่บิตต่าง ๆ ของรีจิสเตอร์ PSW

2.3.5.5 แอกคิวมูเลเตอร์ (Accumulator : ACC)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์และโดยเฉพาะอย่างยิ่งในการคำนวณทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้ให้แก่ซีพียูเพื่อทำการประมวลผลต่อไปอาจเรียกสั้น ๆ ว่า รีจิสเตอร์ A หรือ ACC รีจิสเตอร์นี้สามารถเข้าถึงระดับบิตได้

2.3.5.6 รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ F0H มีหน้าที่พิเศษ คือ หากต้องการคูณหรือหารทางคณิตศาสตร์ต้องนำข้อมูลที่ต้องการหารหรือคูณมาเก็บไว้ในรีจิสเตอร์ B แล้วจึงกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป

2.3.5.7 โปรแกรมเคาน์เตอร์ (Program Counter : PC)

มีขนาด 16 บิต มีหน้าที่แจ้งแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งถัดไปที่ซีพียูจะต้องไปทำงาน รีจิสเตอร์ PC เป็นรีจิสเตอร์ตัวเดียวที่ไม่ได้จัดสรรไว้รวมกับรีจิสเตอร์ SFR

ตัวอื่น ๆ การเปลี่ยนแปลงค่าของรีจิสเตอร์ PC จะขึ้นอยู่กับผลของการกระทำคำสั่งภายในหน่วยความจำโปรแกรมที่ผู้เขียนโปรแกรมกำหนด

RS1	RS0	แบงค์ของรีจิสเตอร์	ช่วงแอดเดรส
0	0	แบงก์ 0	00H-07H
0	1	แบงก์ 1	08H-0FH
1	0	แบงก์ 2	10H-17H
1	1	แบงก์ 3	18H-1FH

ตารางที่ 2.2 การเลือกแบงค์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงก์ R0-R7

2.3.5.8 สแต็กพอยน์เตอร์ (Stack Pointer : SP)

รีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81 ใช้ในการเก็บค่าตำแหน่งของตัวชี้สแต็กซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อยหรือกระโดดจากโปรแกรมย่อมแล้วกลับมายังโปรแกรมหลัก เมื่อมีการรีเซตเกิดขึ้น (รีเซต : การกระทำที่ส่งผลให้ซีพียูต้องเริ่มต้นการทำงานใหม่ตั้งแต่ต้น) ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H ดังนั้นแอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H)

2.3.5.9 รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data Pointer : DPTR)

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไบต์สูง (DPH) และรีจิสเตอร์ชี้ข้อมูลไบต์ต่ำ(DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H , สำหรับ DPL และ 83H สำหรับ DPH รีจิสเตอร์ DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

2.3.5.10 รีจิสเตอร์พอร์ต (Port Register)

เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 มี 4 ตัวคือ รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิส-เตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H, รีจิสเตอร์พอร์ต 2 หรือ P2 ซึ่งมีแอดเดรสอยู่ที่ A0H และรีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหรือ

เขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ ซึ่งจะต้องมีการกระทำผ่านรีจิสเตอร์นี้ทุกครั้ง

2.3.5.11 รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือรีจิสเตอร์ส่งข้อมูล (Transmit Buffer Register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (Receive Buffer Register) เมื่อมีการเขียนข้อมูล มายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูลเพื่อส่งออกจากเอาท์พุทของไมโครคอน - โทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่การอ่านข้อมูลจาก รีจิสเตอร์SBUFข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งไปยัง ไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ทางอินพุทของตัวไมโครคอนโทรลเลอร์ MCS-51

2.3.5.12 รีจิสเตอร์ไทเมอร์ (Timer Register)

เป็นรีจิสเตอร์ขนาด 16 บิต แบ่งเป็นไบต์สูงและไบต์ต่ำสุดเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ (Counter) ภายในตัวของไมโครคอนโทรลเลอร์ เพื่อใช้ในการสร้างฐานเวลา,จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน บางทีเรียกว่ารีจิสเตอร์ตัวนี้ว่ารีจิสเตอร์ไทเมอร์/เคาน์เตอร์

ในไมโครคอนโทรลเลอร์ เบอร์ AT89C51 มีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ 2 ตัวแบ่งเป็น T0 หรือ Timer 0 และ T1 หรือ Timer 1 ในรีจิสเตอร์ยังแบ่งเป็นรีจิสเตอร์ไทเมอร์ไบต์ต่ำ (TL) และรีจิสเตอร์ไทเมอร์ไบต์สูง (TH) เหมือนกัน โดยรีจิสเตอร์ TLO มีแอดเดรสอยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8BH ในขณะที่ TL1 และ TH1 มีแอดเดรสอยู่ที่ 8CH และ 8DH สำหรับในเบอร์ AT89C52 และในอนุกรม AT89Sxx จะมีรีจิสเตอร์ไทเมอร์ / เคาน์เตอร์ถึง 3 ตัว โดยมีรีจิสเตอร์ TL2 และ TH2 ซึ่งมีแอดเดรสอยู่ที่ 0CCH และ 0CDH เพิ่มเติม

2.3.5.13 รีจิสเตอร์แคปเจอร์ (Capture Register)

เป็นรีจิสเตอร์ขนาด 16 บิต มีเฉพาะไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช เบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น เนื่องจากต้องใช้ร่วมกับไทเมอร์/เคาน์ท-เตอร์ 2 (Timer 2) โดยรีจิสเตอร์แคปเจอร์นี้มีชื่อเรียกอย่างย่อว่า รีจิสเตอร์ RCAP2 ซึ่งสามารถแบ่งออกเป็นไบต์ต่ำ คือ RCAP2L มีแอดเดรสอยู่ที่0CAHและไบต์สูงคือRCAP2H มีแอดเดรสอยู่ที่

0CB รีจิสเตอร์แคปเจอร์จะถูกใช้งานเมื่อกำหนดให้ไทเมอร์ 2 ซึ่งจะให้ทำงานในโหมดแคปเจอร์ ซึ่งเป็นโหมดที่กำหนดให้ไมโครคอนโทรลเลอร์ทำการตรวจจับการเปลี่ยนแปลงสถานะทางลอจิกที่ขา T2EX ทั้งนี้เพื่อใช้ประโยชน์ในการวัดคาบเวลา ความถี่ และการเปลี่ยนแปลงของสัญญาณพัลส์ที่ขา T2EX

2.3.5.14 รีจิสเตอร์ควบคุม (Control Register)

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมคอนโทรล- เลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจร สื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์MCS-51แบบแฟลช

รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของตัวไทเมอร์ / เคาน์ทเตอร์ ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2CON ใช้สำหรับไทเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52และในอนุกรมAT89Sxx

รีจิสเตอร์ TMOD และ T2MOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะในการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดย T2MOD ใช้สำหรับไทเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์AT89C52 และในอนุกรมAT89Sxx

รีจิสเตอร์ IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเตอร์รัปต์ (Interrupt: การขัดจังหวะการทำงานปกติของซีพียู) โดย IE เป็นรีจิสเตอร์สำหรับเอ็นเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP เป็น รีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะให้ซีพียูตอบสนองการเกิดอินเตอร์รัปต์ในลักษณะใดก่อนหรือหลัง

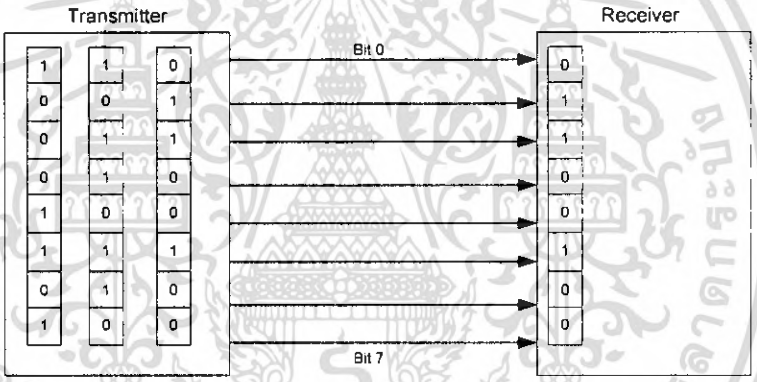
กำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะให้ซีพียูตอบสนองการเกิดอินเตอร์รัปต์ในลักษณะใดก่อนหรือหลัง

2.4 การสื่อสารผ่านพอร์ตอนุกรม

2.4.1 รูปแบบการเชื่อมต่อผ่านพอร์ตสื่อสาร

ในการสื่อสารหรือการส่งข้อมูลในระบบดิจิตอลนั้นมีรูปแบบในการสื่อสารที่สำคัญอยู่ 2 รูปแบบ คือ

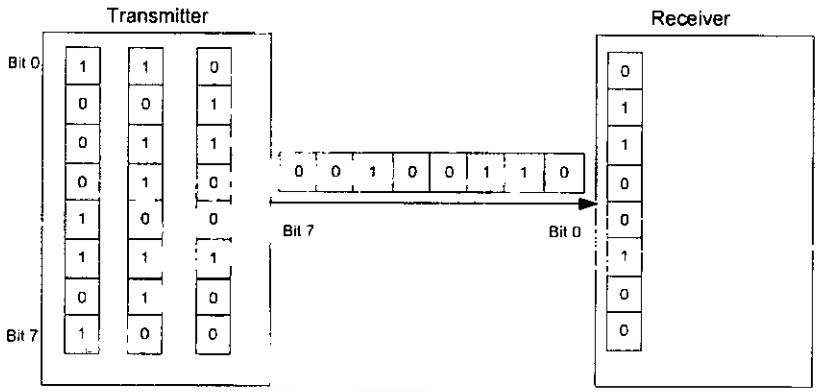
1.การสื่อสารแบบขนาน (Parallel Communication) เป็นการสื่อสารในรูปแบบที่ส่งข้อมูลทุกบิตออกไปพร้อมกัน ดังรูป 2.12



รูปที่ 2.12 แสดงการสื่อสารแบบขนาน

การส่งรูปแบบนี้มีข้อดีคือทำให้สามารถส่งข้อมูลได้ครั้งละจำนวนมากและรวดเร็วกว่าการส่งแบบอนุกรม แต่เนื่องจากการสื่อสารในรูปแบบนี้จะต้องใช้สายนำสัญญาณจำนวนมาก จึงไม่เหมาะกับการส่งข้อมูลในระยะไกลๆได้ โดยมากแล้วจะใช้ในการส่งระยะใกล้เช่น ระหว่างไมโครโพรเซสเซอร์กับฮาร์ดดิสก์ เครื่องพิมพ์ เป็นต้น

2.การสื่อสารแบบอนุกรม (Serial Communication) เป็นการสื่อสารโดยการส่งข้อมูลทีละบิต โดยจะเริ่ส่งบิตต่ำ (LSB) ออกไปก่อน ผ่านสายนำสัญญาณเพียงเส้นเดียว จึงประหยัดค่าใช้จ่ายกว่าแบบขนานแต่จะส่งได้ช้ากว่า ดังรูป 2.13



รูปที่ 2.13 แสดงการสื่อสารแบบอนุกรม

2.4.2 รูปแบบการติดต่อสื่อสาร

การสื่อสารข้อมูลระหว่างตัวรับและตัวส่ง แบ่งตามลักษณะการรับส่งข้อมูล ได้ 3 รูปแบบคือ

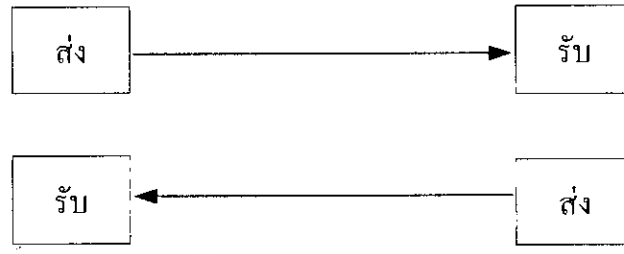
1.การสื่อสารแบบซิมเพล็กซ์ (simplex) สถานีส่งจะทำหน้าที่ส่งอย่างเดียวและสถานีรับทำหน้าที่รับอย่างเดียว เช่น การส่งข้อมูลระหว่างคอมพิวเตอร์กับเครื่องพิมพ์ ดังรูป 2.14

รูปที่ 2.14 แสดงการสื่อสารแบบซิมเพล็กซ์

2.การสื่อสารแบบฮาล์ฟดูเพล็กซ์ (half duplex) ทั้งสองสถานีสามารถส่งและรับข้อมูลได้แต่จะทำในเวลาต่างกัน เช่น การสื่อสารแบบเครื่องอินเตอร์คอมที่มีสายส่งสัญญาณได้ทางเดียว ดังรูป 2.15

รูปที่ 2.15 แสดงการสื่อสารแบบฮาล์ฟดูเพล็กซ์

3.การสื่อสารแบบฟูลล์ดูเพล็กซ์ (full duplex) ทั้งสองสถานีสามารถส่งและรับข้อมูลได้สองทิศทางในเวลาเดียวกัน ดังรูป 2.16



รูปที่ 2.16 แสดงการสื่อสารแบบฟูลล์ดูเพล็กซ์

2.4.3 การส่งข้อมูลในการสื่อสารแบบอนุกรม

การส่งข้อมูลในการสื่อสารแบบอนุกรมมี 2 วิธี คือ

1.การส่งแบบเข้าจังหวะเวลา (synchronous) จะต้องมีการส่งสัญญาณนาฬิการ่วมไปด้วยเพื่อควบคุมการรับส่งข้อมูล

2.การส่งข้อมูลแบบไม่เข้าจังหวะเวลา (asynchronous) ไม่ต้องมีการส่งสัญญาณนาฬิกาแต่จะใช้การกำหนดอัตราเร็วในการรับส่งข้อมูลให้มีค่าเท่ากันทั้งสองสถานีที่เรียกว่า อัตราบอด หรือ บอดเรท (baud rate) มีหน่วยเป็นบิตต่อวินาที(bit per second : bps)

2.4.4 การส่งข้อมูลแบบเข้าจังหวะเวลา (synchronous)

เมื่ออุปกรณ์ทั้งสองตัวจะสื่อสารข้อมูลซึ่งกันและกันจะต้องกำหนดรูปแบบกฎเกณฑ์การสื่อสารซึ่งกันและกันเพื่อให้สามารถเข้าใจกันได้ รูปแบบการรับส่งข้อมูลที่กำหนดขึ้นมานี้เรียกว่าโปรโตคอล (protocol)

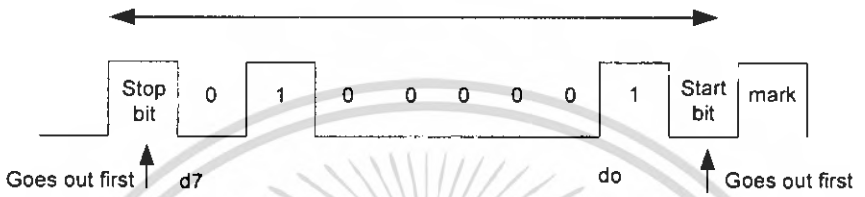
การรับส่งข้อมูลแบบเข้าจังหวะเวลา ประกอบด้วย 4 ส่วน ได้แก่

1. บิตเริ่มต้น(start bit)
2. ข้อมูลอนุกรม
3. บิตตรวจสอบความถูกต้อง
4. บิตสุดท้าย(stop bit)

การส่งข้อมูลแบบเข้าจังหวะเวลา จะทำการส่งข้อมูลออกไปเป็นชุดเรียกว่า เฟรม ภายในเฟรมจะประกอบด้วยข้อมูลหรือรหัส ASCII ก็ได้ ในแต่ละเฟรมจะเริ่มต้นด้วยบิตเริ่มต้น (start bit) ที่จะบอกว่าสิ่งที่ตามมาคือบิตข้อมูลและจะจบด้วยข้อมูลบิตสุดท้าย(stop bit) ที่จะบอกว่าข้อมูลในเฟรมนั้นๆได้สิ้นสุดลงแล้ว

การส่งข้อมูลในลักษณะนี้ถ้าหากยังไม่มีการส่งข้อมูลระดับลอจิกที่สายส่งจะเป็นลอจิก "1"เรียกว่าสภาวะรอ (waiting stage) ถ้าหากมีข้อมูลส่งสัญญาณจะเป็นลอจิก "0" ในช่วงเวลาหนึ่ง บิตข้อมูลบิตนี้เรียกว่าบิตเริ่มต้น จากนั้นจะตามด้วยบิตข้อมูล

ตัวอย่างเช่น ถ้าหากจะส่งรหัส ASCII ของตัว "A" สัญญาณจะมีลักษณะดังรูป 2.17 โดยส่งค่าข้อมูล 8 บิต(01000001) ออกไปจากนั้นจะจบด้วยบิตปิดท้ายหรือบิตหยุดที่มีสภาวะเป็นลอจิก "1"



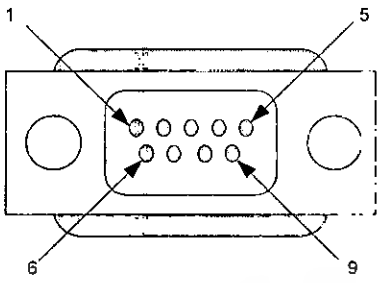
รูปที่ 2.17 แสดงลักษณะสัญญาณที่ใช้ในการส่งรหัส ASCII ของตัว "A"

ข้อมูลที่รับส่งกันนั้นอาจเป็น 7 บิตหรือ 8 บิตก็ได้ ขึ้นอยู่กับการออกแบบ และสามารถทำการตรวจสอบข้อมูลในระดับหนึ่งเรียกว่า การตรวจสอบบิตพาริตี้ (parity bit) ซึ่งเป็นบิตข้อมูลที่เพิ่มเข้าไปก่อนบิตปิดท้าย การสร้างบิตพาริตี้จะถูกสร้างจากภาคส่งของ UART ซึ่งสามารถโปรแกรมได้ว่าการส่งนั้นจะตรวจสอบแบบพาริตี้คี่ (odd parity) หรือพาริตี้คู่ (even parity) หรือไม่มีพาริตี้ก็ได้

2.4.5 มาตรฐาน RS-232

การสื่อสารแบบอนุกรมกุบคอมพิวเตอร์ส่วนบุคคลมักจะใช้รูปแบบมาตรฐาน Recommended Standard-232 หรือ RS-232 ซึ่งเป็นรูปแบบของการส่งข้อมูลแบบเข้าจังหวะเวลากำหนดโดย Electrical Industries Association (EIA) มีความยาวสายสูงสุด 50 ฟุต ระดับแรงดันของลอจิกที่ใช้ในการสื่อสารตามมาตรฐานนี้ ลอจิก "1" แทนด้วยแรงดัน -3 ถึง -12 โวลต์ ส่วนลอจิก "0" จะแทนด้วยแรงดัน +3 ถึง +12 โวลต์ แต่แรงดันในช่วง +3 จะไม่ถูกกำหนดให้ใช้งานเนื่องจากแรงดันดังกล่าวไม่สามารถใช้งานกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้ โดยทั่วไปแล้วถ้าหากต้องการให้ไมโครคอนโทรเลอร์ติดต่อกับคอมพิวเตอร์ ตามมาตรฐาน RS-232 จะต้องออกแบบวงจรอิเล็กทรอนิกส์เพิ่มเติมแต่ในปัจจุบันจะใช้ไอซี MAX232 ทำหน้าที่เปลี่ยนแรงดันทางลอจิกให้อยู่ในมาตรฐาน

ในคอมพิวเตอร์จะมีขั้วต่อ RS-232 หรือที่เรียกว่า คอนเนคเตอร์ (connector) อยู่สองแบบ คือ ขั้วต่อแบบ DB-25 และ DB-9 การเชื่อมต่อกับพอร์ตสื่อสารของคอมพิวเตอร์ส่วนบุคคลจะเลือกใช้พอร์ตสื่อสารแบบ DB-9 ซึ่งสามารถทำการรับส่งข้อมูลได้แบบอนุกรม โดยลักษณะของการเชื่อมต่อของพอร์ตสื่อสารสำหรับคอนเนคเตอร์แบบ DB-9 สามารถแสดงได้ดังรูป 2.18



Pin	Description
1	Data carrier detect (DCD)
2	Receiver data (RXD)
3	Transmitted data (TXD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (DSR)
7	Request to send (RTS)
8	Clear to send (CTS)
9	Ring indicator (RI)

รูปที่ 2.18 แสดงพอร์ตสื่อสารสำหรับคอนเนคเตอร์แบบ DB-9

2.4.6 การเชื่อมต่อ MCS-51 กับ RS-232

การเชื่อมต่อพอร์ตอนุกรมของ MCS-51 กับมาตรฐาน RS-232 นั้นเนื่องจากระดับสัญญาณของการสื่อสาร RS-232 ไม่เป็นไปตามแรงดัน TTL จึงต้องนำชิบ MAX232 มาช่วยปรับแรงดันให้กับ MCS-51

พอร์ตที่ใช้ในการรับส่งข้อมูลแบบอนุกรมของ MCS-51 จะส่งออกมาทางขา RxD และTxD ซึ่งอยู่ในพอร์ต 3 (P3.0 และ P3.1) โดย TxD จะเป็นขาที่ 9 และ RxD จะเป็นขาที่ 10 การรับส่งข้อมูลกับมาตรฐาน RS-232 เราจะต้องคิดต่อผ่านขาทั้งสองนี้โดยนำไอซี MAX232 มาเชื่อมต่อได้ดังวงจรในรูป ไอซีตัวนี้ใช้ไฟเลี้ยง +5 โวลต์เท่ากับไฟเลี้ยงไมโครคอนโทรเลอร์แต่สามารถยกระดับแรงดันตั้งแต่ -25 ถึง+25 โวลต์ได้โดยไม่ต้องใช้ไฟเลี้ยงแบบคู่ (dual power supply)

2.4.7 อัตราบอดในการสื่อสาร

อัตราบอด (Baud rate) คือ การเปลี่ยนแปลงของสัญญาณในเวลา 1 วินาทีซึ่งการเปลี่ยนแปลงของสัญญาณ 1 ครั้งอาจจะแสดงถึง การส่งข้อมูลอนุกรมมากกว่า 1 บิตก็ได้ ดังนั้นอัตราการส่งข้อมูลเป็นจำนวนบิตจึงมีค่าเท่ากับ อัตราบอดคูณกับจำนวนบิตใน 1 บอด ดังตาราง 2.3

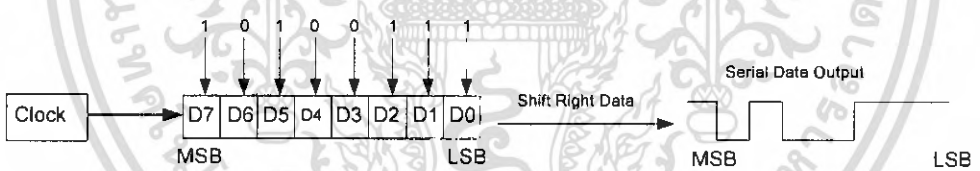
อัตราบอด	ช่วงเวลาของแต่ละบิต(ms)
110	9.91
150	6.67
300	3.33
600	1.67
1200	0.833

อัตราบอด	ช่วงเวลาของแต่ละบิต(ms)
2400	0.417
4800	0.208
9600	0.104
19200	0.052

ตารางที่ 2.3 แสดงแสดงอัตราบอดของ UART ที่ใช้กันทั่วไปในการโอนถ่ายข้อมูลแบบอนุกรม

2.4.8 การแปลงรูปแบบข้อมูล

ใน MCS-51 จะมีพอร์ตอนุกรมอยู่ภายในซึ่งรับส่งข้อมูลแบบฟูลล์ดูเพล็กซ์ เมื่อต้องการติดต่อกับอุปกรณ์ภายนอก MCS-51 จะส่งข้อมูลออกมามีขนาดเป็นไบต์ หรือ 8 บิต เนื่องจาก MCS-51 มีบัสข้อมูลขนาด 8 บิต การโอนถ่ายข้อมูลต่างๆจะทำแบบขนาน ดังนั้นถ้าต้องการส่งข้อมูลออกไปแบบอนุกรมจะต้องเปลี่ยนข้อมูลแบบขนานนี้ให้เป็นข้อมูลอนุกรมเสียก่อนแล้วจึงส่งออกไป ส่วนการรับข้อมูลนั้นจะรับข้อมูลเข้ามาครั้งละ 1 บิตแล้วเปลี่ยนข้อมูลให้เป็นข้อมูลแบบขนานเพื่อส่งให้ MCS-51 ประมวลผลต่อไป ในระบบคอมพิวเตอร์ อุปกรณ์ที่เปลี่ยนข้อมูลอนุกรมเป็นขนานและเปลี่ยนข้อมูลขนานเป็นอนุกรมคือ UART (Universal Asynchronous Receiver-Transmitter) ดังรูป 2.19



รูปที่ 2.19 แสดงการแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม

การแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม เริ่มจากข้อมูลแบบขนานจะถูกนำไปเก็บไว้ใน Shift Register หลังจากนั้นจะใช้สัญญาณนาฬิกาในการเลื่อนค่าในรีจิสเตอร์ออกมาทีละบิต (โดยการเลื่อนค่าปทางขวามือ) โดยบิตแรกที่ถูกเลื่อนออกมาคือบิต LSB ของข้อมูลและบิตที่สองที่ถูกเลื่อนออกมาก็คือบิตที่อยู่ถัดจากบิต LSB และบิตต่อๆไป สำหรับบิตสุดท้ายที่ถูกเลื่อนออกมาก็คือบิต MSB ของข้อมูล

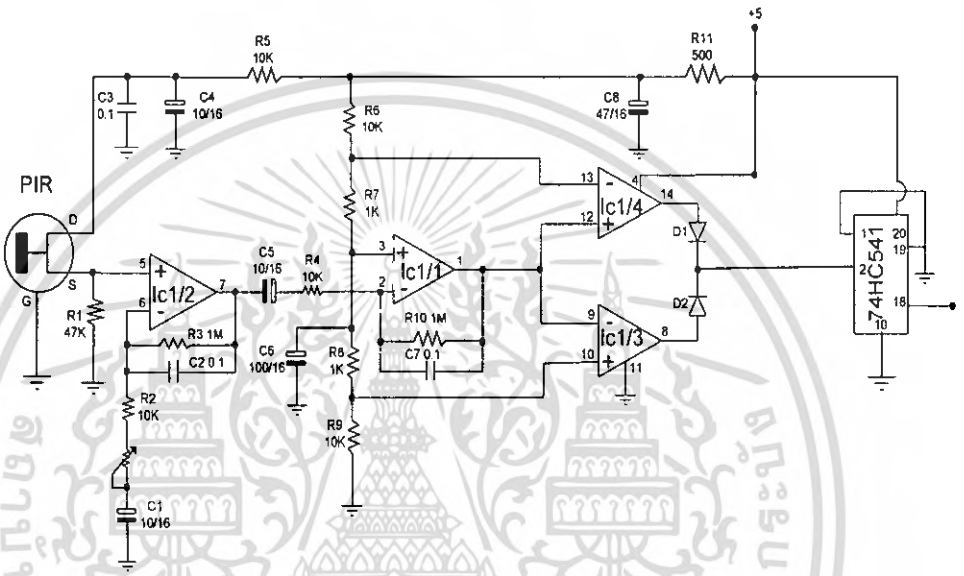
การแปลงข้อมูลแบบอนุกรมไปเป็นข้อมูลแบบขนานนั้นมีขั้นตอนตรงกันข้ามกับที่กล่าวมา นั่นคือข้อมูลแบบอนุกรมจะถูกเลื่อนเข้าไปใน Shift Register โดยใช้สัญญาณนาฬิกาเป็นตัวควบคุมและหลังจากข้อมูลทุกบิตได้เคลื่อนเข้าไปในรีจิสเตอร์ได้ทั้งหมดแล้ว ข้อมูลเหล่านั้นจะถูกนำออกมาแบบขนานเพื่อนำไปใช้งานต่อไป

หน้าที่ของ UART นอกจากจะทำแปลงข้อมูลดังที่ได้กล่าวมาแล้ว ยังมีหน่วยควบคุมและตรวจสอบการทำงานอีกด้วย

บทที่ 3
การคำนวณและการสร้าง

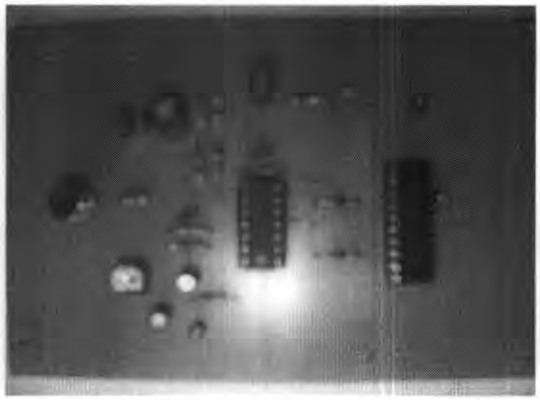
3.1 โครงสร้างด้านฮาร์ดแวร์

3.1.1 วงจรตรวจจับความเคลื่อนไหว โดยใช้อุปกรณ์ตรวจจับไพโรอิเล็กทริก



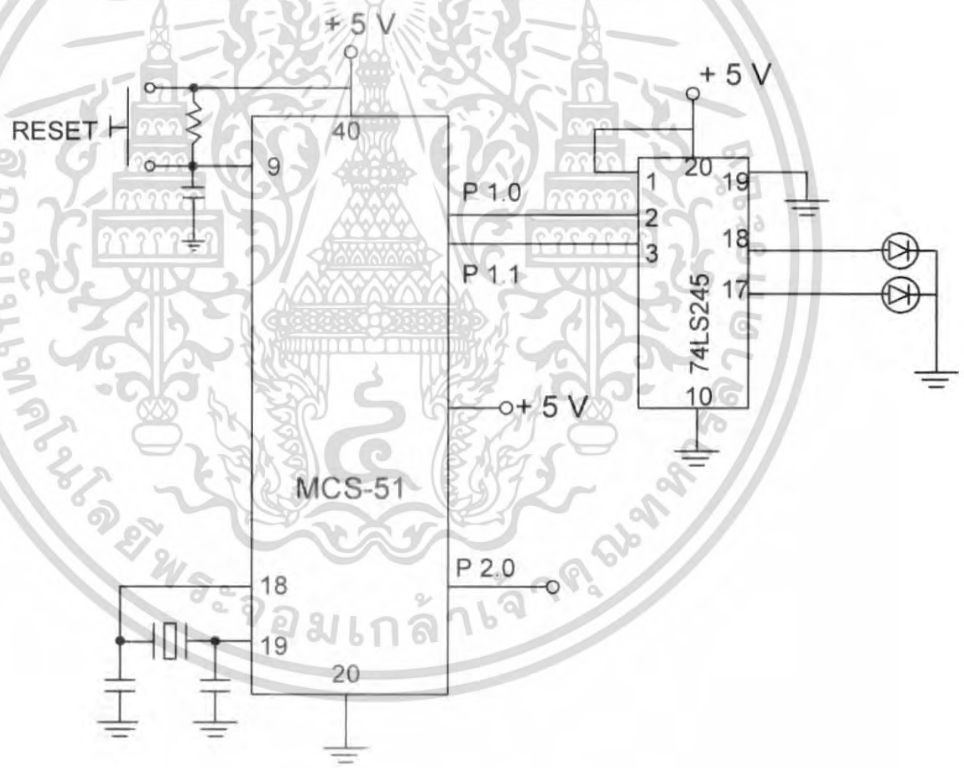
รูปที่ 3.1 แสดงวงจรตรวจจับความเคลื่อนไหว โดยใช้อุปกรณ์ตรวจจับไพโรอิเล็กทริก

เมื่อมีคนหรือสัตว์เดินผ่านหน้า PIR จะทำให้ที่ขา S ของ PIR มีพัลส์ลูกเล็กๆเกิดขึ้น เนื่องมาจากตัว PIR จะการทำการตรวจจับการเปลี่ยนแปลงความร้อนจากการเปลี่ยนแปลงของรังสีอินฟาเรดที่แผ่ออกมาจากตัวของคนหรือสัตว์ ในขณะที่มีการเคลื่อนไหวพัลส์ลูกเล็กๆที่ออกมาจาก PIR นี้จะถูกขยายด้วย IC1/2 ซึ่งทำหน้าที่เป็นวงจรปรีแอมป์ สัญญาณที่ได้นี้จะถูกขยายอีก 100 เท่าด้วย IC1/1 ก่อนส่งไปเข้า IC1/3 และ IC1/4 ซึ่งไอซีทั้งสองนี้จะทำหน้าที่เป็นตัวเปรียบเทียบสัญญาณที่เข้ามา โดย IC1/3 จะเปรียบเทียบในช่วงที่สัญญาณสวิงลง ส่วน IC1/4 จะเปรียบเทียบในช่วงที่สัญญาณสวิงขึ้น



รูปที่ 3.2 แสดงวงจรตรวจจับความเคลื่อนไหวที่ใช้งานจริง

3.1.2 วงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์

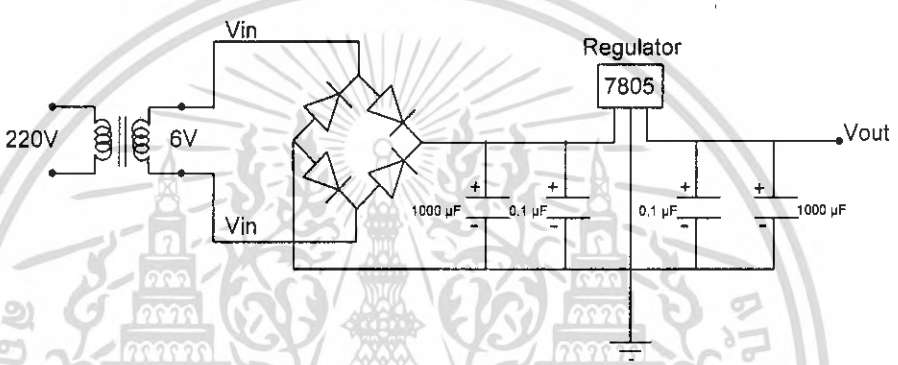


รูปที่ 3.3 แสดงวงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์

ตัวไมโครคอนโทรเลอร์จะทำการรับข้อมูลจากพอร์ตสื่อสารอนุกรมมา แล้วทำการเปรียบเทียบเงื่อนไขในการที่จะทำให้หลอดไฟติดหรือดับ โดยจะส่งข้อมูลไปยังไอซี 74LS245 ซึ่งเป็นบัฟเฟอร์ เพื่อขับให้ LED ติดหรือดับตามที่ได้รับข้อมูลมา ซึ่งการที่จะให้ LED สว่างนั้นจะต้องไบแอสด้วยความต่าง

ศักย์ 1.5-3 โวลต์ และให้กระแสไหลผ่านในช่วง 5 ถึง 25 มิลลิแอมป์ แต่โดยทั่วไปแล้วแต่ละบิทของ MCS-51 จะมีค่ากระแสไฟฟ้าต่ำ จึงต้องมีการนำบัฟเฟอร์มาช่วยในการขับกระแส นอกจากจะทำการสั่งให้ LED ติดหรือดับตามข้อมูลที่ได้รับมาจากพอร์ตจากสื่อสารอนุกรมแล้วไมโครคอนโทรเลอร์ยังรับค่ามาจากวงจรตรวจจับความเคลื่อนไหวว่ามีการเคลื่อนไหวหรือไม่ ถ้ามีการเคลื่อนไหวไมโครคอนโทรเลอร์ก็จะสั่งให้ LED ติด แต่ถ้าไม่มีการเคลื่อนไหวภายในระยะเวลาที่กำหนดไมโครคอนโทรเลอร์จะสั่งให้ LED ดับ

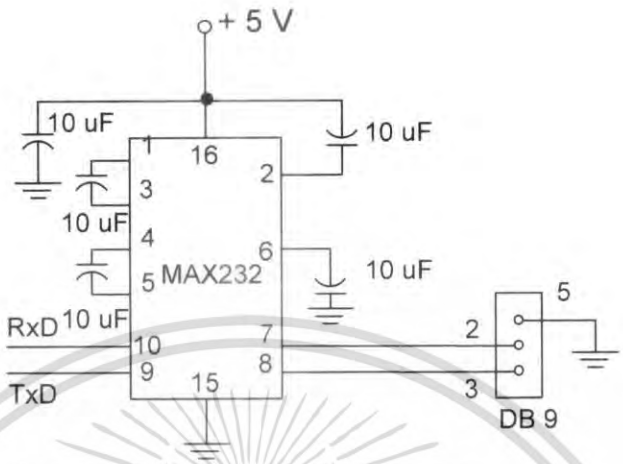
3.1.3 วงจรจ่ายไฟเลี้ยง



รูปที่ 3.4 แสดงวงจรจ่ายไฟเลี้ยง

วงจรจ่ายไฟเลี้ยงเป็นวงจรแปลงไฟแบบเต็มคลื่นแบบหนึ่ง ประกอบไปด้วยไดโอด 4 ตัว ตัวเก็บประจุทำหน้าที่เกลี่ยไฟฟ้ากระแสตรงให้เรียบขึ้น และหม้อแปลงไฟแบบไม่ต้องมี center tapped การทำงานของวงจรภาคจ่ายไฟเริ่มจากหม้อแปลงไฟ 220 โวลต์ ไฟฟ้ากระแสสลับผ่านวงจร bridge diode เพื่อทำการแปลงให้เป็นไฟฟ้ากระแสตรงผ่านตัวเก็บประจุเพื่อเพิ่มประสิทธิภาพของแรงดันไฟฟ้า หลังจากที่ได้ไฟฟ้ากระแสตรงที่มีแรงดัน 5 โวลต์โดยผ่าน ไอซี regulator เบอร์ 7805 เพื่อแปลงแรงดันจาก 6 โวลต์ดีซีเป็น 5โวลต์ ดีซี เพื่อนำไปใช้เลี้ยงไอซีต่อไป

3.1.4 วงจรรับส่งข้อมูลอนุกรม



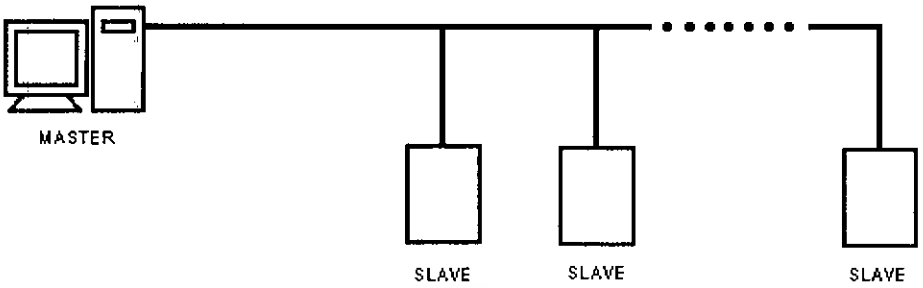
รูปที่ 3.5 แสดงวงจรรับส่งข้อมูลอนุกรม

สัญญาณตามมาตรฐาน RS-232 ซึ่งมีระดับแรงดันอยู่ -12V สำหรับลอจิก "1" และ +12V สำหรับลอจิก "0" เมื่อสัญญาณถูกส่งผ่านมา จะถูกแปลงให้มีระดับแรงดัน +5V สำหรับลอจิก "1" และ 0V สำหรับลอจิก "0" โดยใช้ไอซี MAX232

ไอซี MAX232 1 ตัวจะมีวงจรแปลงอยู่ภายในทั้งหมด 4 ชุดแบ่งออกเป็นวงจรแปลงจาก TTL เป็น RS-232 2 ชุด และวงจรแปลงจาก RS-232 เป็น TTL อีก 2 ชุด

รูปที่ 3.6 แสดงวงจรควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าที่ใช้งานจริง

3.1.5 โครงสร้างการเชื่อมต่ออุปกรณ์ภายในระบบ



MASTER = Main computer

SLAVE = Microcontroller + Sensor cuircuit

รูปที่ 3.7 แสดงโครงสร้างการเชื่อมต่ออุปกรณ์ภายในระบบ

การเชื่อมต่ออุปกรณ์ทั้งหมดเข้าสู่ระบบโดยใช้การเชื่อมต่อแบบ RS232 นั้น นี้ต้องมีการจัดอันดับความสำคัญของอุปกรณ์ที่ต่ออยู่ในระบบ คือ ให้มีอุปกรณ์ทำหน้าที่เป็นตัวแม่ (Master) 1 ตัว สำหรับทำหน้าที่เป็นตัวกลางในการควบคุมการทำงานของอุปกรณ์ตัวลูก ส่วนตัวลูก (Slave) เมื่อได้รับคำสั่งจากอุปกรณ์ตัวแม่ ก็จะปฏิบัติตามคำสั่งนั้นพร้อมทั้งรายงานผลการทำงานกลับไปยังอุปกรณ์ตัวแม่ ซึ่งในทางฮาร์ดแวร์ ในโครงงานนี้เป็นการจำลองระบบอย่างง่ายโดยให้มีตัวลูกเพียง 2 ตัว กล่าวคือสามารถควบคุมการใช้พลังงานใน 2 ห้อง

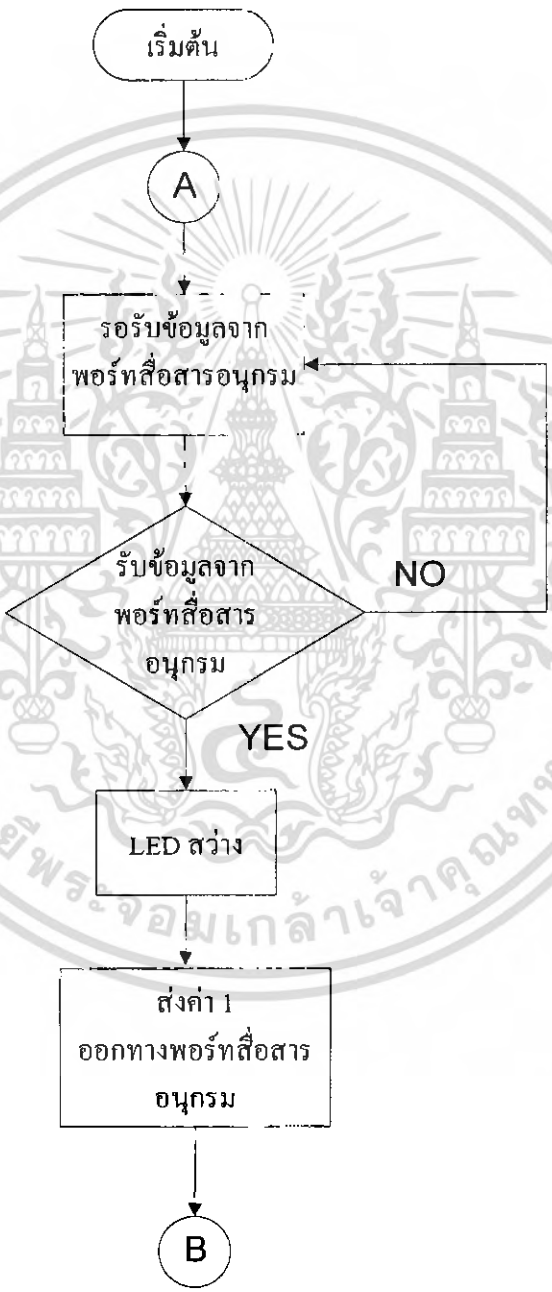
การทำงานของระบบจะถูกควบคุมโดยตัวแม่ เมื่อตัวแม่ต้องการส่งข้อมูลไปยังตัวลูกตัวใด ตัวแม่จะทำการส่งแอดเดรสของตัวลูกที่ต้องการติดต่อด้วยผ่านการเชื่อมต่อ RS232 ตัวลูกตัวใดที่มีแอดเดรสตรงกันกับแอดเดรสที่ตัวแม่ต้องการติดต่อด้วย ก็จะทำการรับข้อมูลจากตัวแม่ และเมื่อตัวลูกตัวใดต้องการส่งข้อมูลกลับไปยังตัวแม่ ตัวลูกตัวนั้นสามารถทำการส่งข้อมูลได้ทันที โดยข้อมูลที่ส่งนั้นต้องประกอบด้วย แอดเดรสของตัวลูกตัวนั้นพร้อมกับข้อมูลที่ต้องการจะส่ง

3.2 โครงสร้างด้านซอร์ฟแวร์

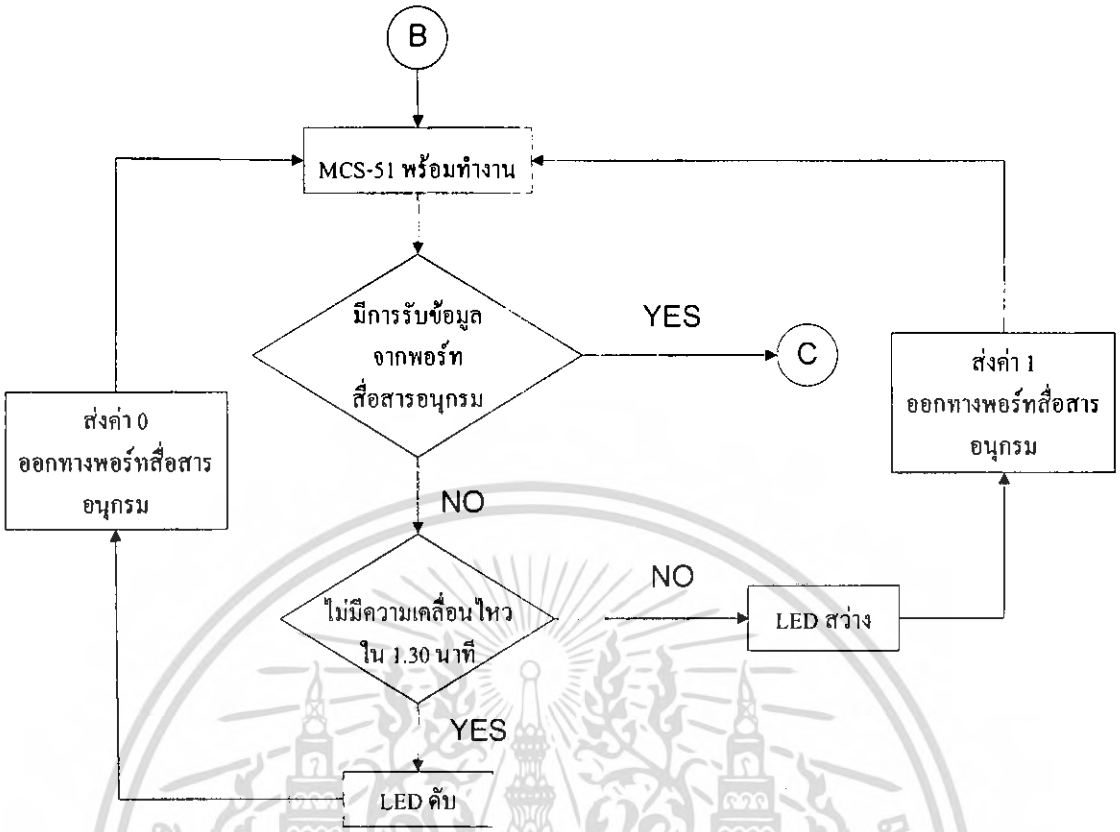
3.2.1 โฟลว์ชาร์ท

ทำการเขียนโปรแกรมควบคุมการทำงานตามโฟลว์ชาร์ทได้ดังนี้

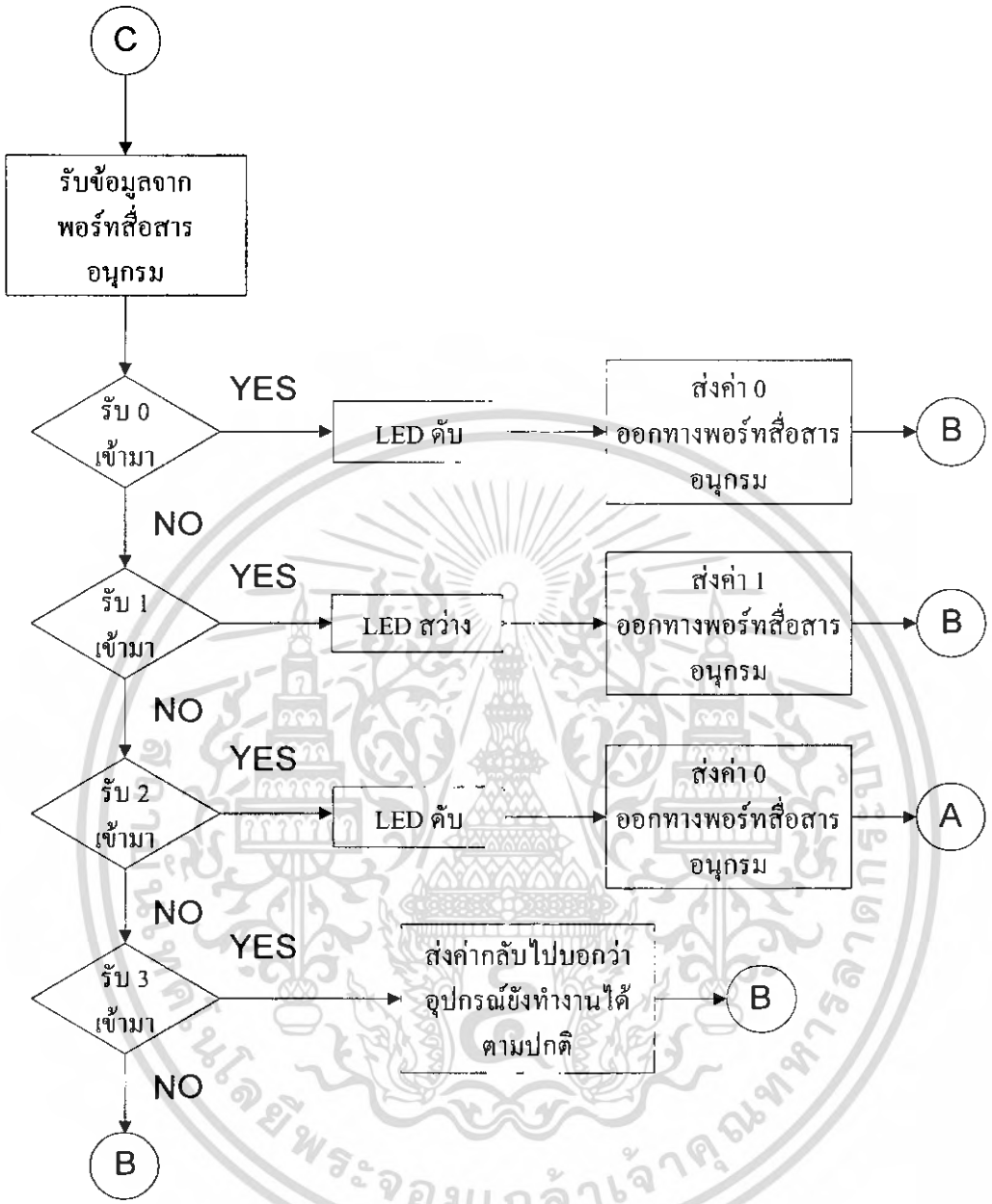
3.2.1.1 โปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์



รูปที่ 3.8 แสดงการทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (1)

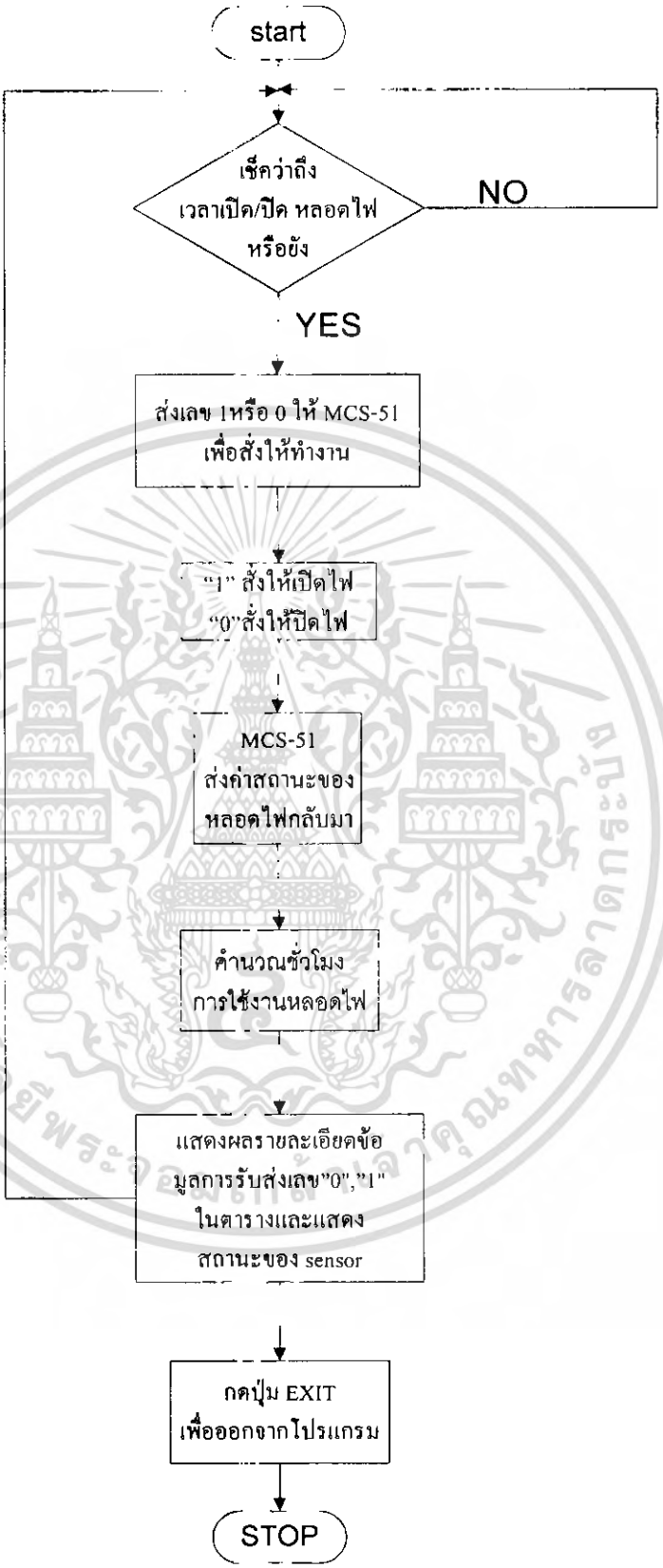


รูปที่ 3.9 แสดงการทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (2)



รูปที่ 3.10 แสดงการทำงานของโปรแกรมหลักควบคุมการทำงานของไมโครคอนโทรเลอร์ (3)

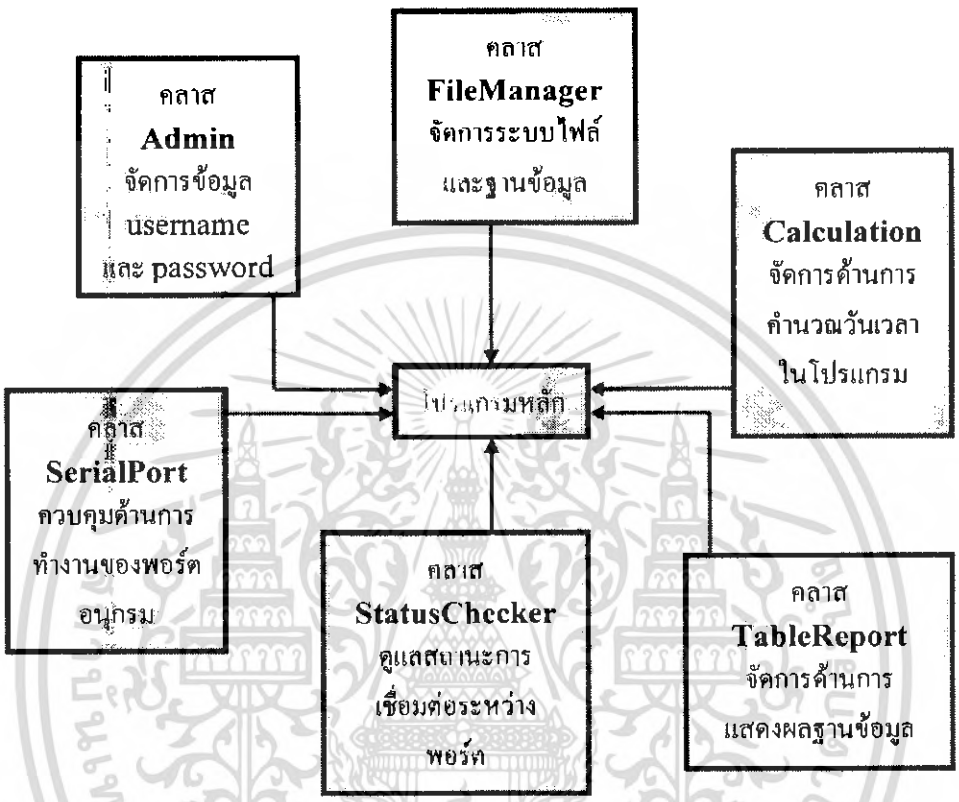
3.2.1.2 โปรแกรมควบคุมการทำงานของหลอดไฟโดยอัตโนมัติ



รูปที่ 3.11 แสดง การทำงานของโปรแกรมควบคุมการทำงานของหลอดไฟโดยอัตโนมัติ

3.2.2 UML diagram

UML ไดอะแกรม แสดงความสัมพันธ์ระหว่างคลาสต่างๆกับโปรแกรม



รูปที่ 3.12 UML diagram

3.2.3 โครงสร้างของคลาส

3.2.3.1 คลาส FileManager (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่จัดการดูแลระบบไฟล์ไม่ว่าจะเป็นการเขียนไฟล์ การเปิดไฟล์ แทรกไฟล์และจัดการระบบฐานข้อมูลทุกอย่างในโปรแกรม นอกจากนี้ยังช่วยในการจัดการรูปแบบข้อมูลอีกด้วย คลาส FileManager ประกอบไปด้วย member function ต่างๆดังนี้

- SetupDatabase

  ทำหน้าที่สร้างฐานข้อมูล ตรวจสอบความพร้อมของฐานข้อมูล และไฟล์ต่างๆ

- ConvertNumtoDay

  ทำหน้าที่แปลงตัวเลขไปเป็นวันที่ในภาษาไทย เช่น 1 ---> วันจันทร์ เป็นต้น

- ConvertDayToNum

ทำหน้าที่แปลงวันที่ในภาษาไทยไปเป็นตัวเลข เช่น วันจันทร์ ---> 1 เป็นต้น

- DeleteTextFromFile :

  ทำหน้าที่ลบข้อความที่กำหนดออกจากไฟล์ ถ้าในไฟล์มีข้อความนั้นอยู่

- WriteTextToFile :

  ทำหน้าที่เขียนข้อความที่กำหนดลงในไฟล์

- WriteDataToStatisticsFile :

  ทำหน้าที่ลบข้อความที่กำหนดออกจากไฟล์ ถ้าในไฟล์มีข้อความนั้นอยู่

- WriteToOverallPeriodFile

  ทำหน้าเขียนข้อมูลเกี่ยวกับสถิติเวลาการใช้ไฟฟ้าลงในไฟล์เฉพาะที่ใช้เก็บข้อมูลนี้

- ReadFileLineToLine

  ทำหน้าอ่านไฟล์แบบบรรทัดต่อบรรทัด

- ReadFile_MainCommand

  ทำหน้าอ่านไฟล์จากไฟล์ฐานข้อมูลที่เก็บคำสั่งหลัก

- ReadFile_UserCommand

  ทำหน้าอ่านไฟล์จากไฟล์ฐานข้อมูลที่เก็บคำสั่งรอง

- GetMainComData :

  คืนค่าเป็นข้อมูลที่ได้จากการอ่านไฟล์จากไฟล์ที่เก็บคำสั่งหลัก

3.2.3.2 คลาส Calculation (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่คำนวณวันและเวลาไม่ว่าจะเป็นหาผลต่างของชั่วโมง นาที วินาที เปรียบเทียบวันและเวลา แปลงหน่วยระหว่างวัน ชั่วโมง นาที วินาที รวมทั้งจัดการรูปแบบข้อมูลที่ใช้แสดงผลวันและเวลา คลาส Calculation ประกอบไปด้วย member function ต่างๆดังนี้

- GetTodayDateTime

  คืนค่าเป็นวัน เดือน ปี และ เวลา ของวันปัจจุบัน

- ConvertHMStoSecond

  แปลงค่าชั่วโมง นาที วินาที ให้เป็นหน่วยวินาทีทั้งหมด

- ConvertSecondToHMS

  แปลงวินาทีให้เป็น ชั่วโมง นาที และ วินาที

- ConvertEnglishDayToThaiDay

  แปลงวันภาษาอังกฤษให้เป็นตัวเลข

- CalculationNextDuty

  คำนวณหาวัน เดือน ปี และเวลา ที่จะต้องทำคำสั่งถัดไป

3.2.3.3 คลาส TableReport (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่จัดการการแสดงผลข้อมูลในฐานข้อมูลให้ออกมาอยู่ในรูปของ ตารางรายงาน เพื่อให้ดูข้อมูลง่ายและข้อมูลมีความเป็นระเบียบ คลาส TableReport ประกอบไปด้วย member function ต่างๆดังนี้

- TransferHistoryTable
  แสดงตารางข้อมูลประวัติการส่งผ่านของข้อมูลระหว่างคอมพิวเตอร์กับไมโครคอนโทรเลอร์
- ExtraUserCommandTable
  แสดงตารางข้อมูลในฐานข้อมูลคำสั่งรอง
- MainCommandTable
  แสดงตารางข้อมูลในฐานข้อมูลคำสั่งหลัก
- StatisticsTable
  แสดงตารางสถิติการใช้ไฟฟ้าทั้งหมด

3.2.3.4 คลาส StatusChecker (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่ตรวจสอบสถานการณ์เชื่อมต่อระหว่างพอร์ตอนุกรมและ ตรวจสอบสถานะการเชื่อมต่อกับ MCS-51 นอกจากนี้ยังตรวจสอบสถานะของเครื่องใช้ไฟฟ้าในแต่ละห้องว่าเปิดหรือปิดอยู่ด้วย คลาส StatusChecker ประกอบไปด้วย member function ต่างๆดังนี้

- StatusCheck
  ทำหน้าที่ตรวจสอบสถานะระหว่างพอร์ตสื่อสารอนุกรม
- RoomStatus
  ทำหน้าที่ตรวจสอบสถานะของเครื่องใช้ไฟฟ้าในแต่ละห้อง

3.2.3.5 คลาส SerialPort (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่เปิด/ปิดพอร์ตสื่อสารอนุกรม และส่งข้อมูลผ่านพอร์ตสื่อสารอนุกรม คลาส SerialPort ประกอบไปด้วย member function ต่างๆดังนี้

- Open
  ทำหน้าที่เปิดพอร์ตสื่อสารอนุกรม
- Close
  ทำหน้าที่ปิดพอร์ตสื่อสารอนุกรม
- Write

ทำหน้าส่งข้อมูลผ่านพอร์ตสื่อสารอนุกรม

3.2.3.6 คลาส Admin (ซอร์สโค้ดของโปรแกรมดูที่ ภาคผนวก...)

ทำหน้าที่ดูแลข้อมูล username และ password ของทั้งผู้ดูแลระบบและ ผู้ใช้ทั่วไป (คลาสนี้จะมีประโยชน์เมื่อมีผู้ใช้งานร่วมกันมากขึ้น)

คลาส Admin ประกอบไปด้วย member function ต่างๆดังนี้

- GetUsername

  คืนค่าเป็น username ของผู้ใช้คนนั้น

- GetPassword

  คืนค่าเป็น password ของผู้ใช้คนนั้น

3.2.4 โปรแกรม ESystem V3.0 (Engergy saving System)

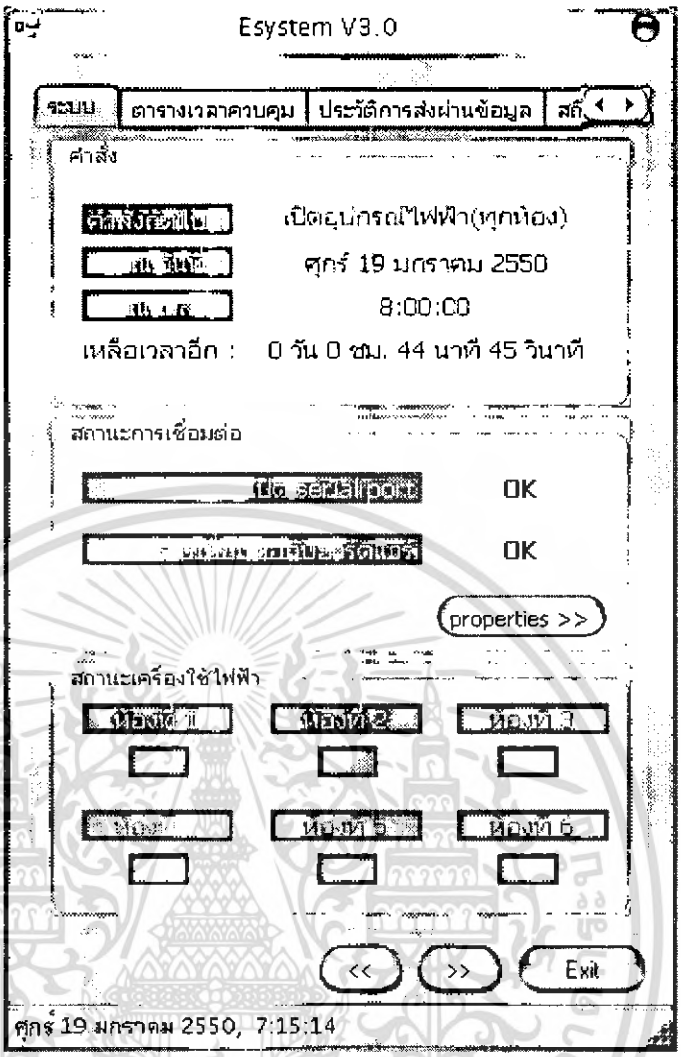
โปรแกรม ESystem เป็นโปรแกรมที่สร้างขึ้นมาตามแผนการที่วางแผนไว้ดังที่กล่าวไปแล้วข้างต้นไม่ว่าจะเป็น โฟลว์ชาร์ท UML ไดอะแกรม และโครงสร้างของคลาสต่างๆ

โปรแกรม ESystem มีหน้าที่หลักต่างๆดังนี้

1. ตรวจสอบการรับส่งข้อมูลอัตโนมัติระหว่างพอร์ตสื่อสารอนุกรม
2. ตรวจสอบคำสั่งในการควบคุม การเปิด/ปิด เครื่องใช้ไฟฟ้า
3. เก็บข้อมูลทางสถิติทุกอย่าง ไม่ว่าจะเป็น ประวัติการรับส่งข้อมูล ช่วงเวลาที่ใช้ไฟฟ้า เป็นต้น
4. แสดงผลและรายงานผลข้อมูลต่างๆแบบเรียลไทม์ (Realtime Report)

3.2.4.1 การใช้งานโปรแกรมเบื้องต้น

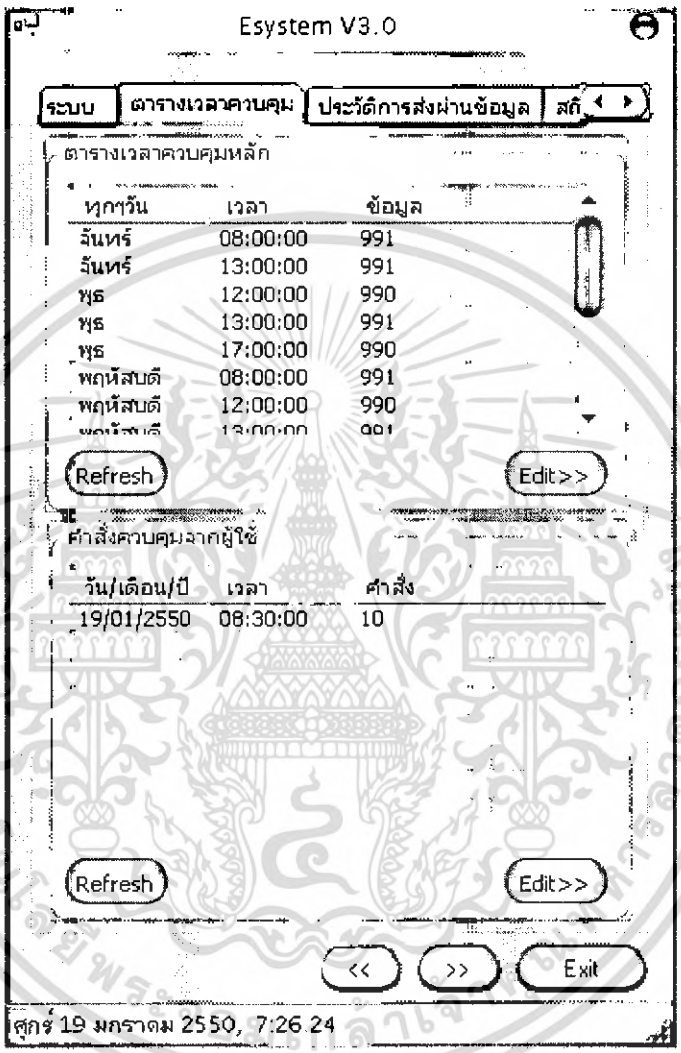
1) หน้าหลักระบบ เมื่อเปิดโปรแกรมขึ้นมาจะเจอหน้าต่างดังรูปข้างล่างโดยเมื่อเปิดโปรแกรมโปรแกรมจะเริ่มทำงานเองโดยอัตโนมัติเมื่อสถานการณ์เชื่อมต่อพร้อม (สังเกตส่วนของสถานการณ์เชื่มต่อ แสดง “OK”)



รูปที่ 3.13 แสดงหน้าต่างหน้าหลักของระบบ

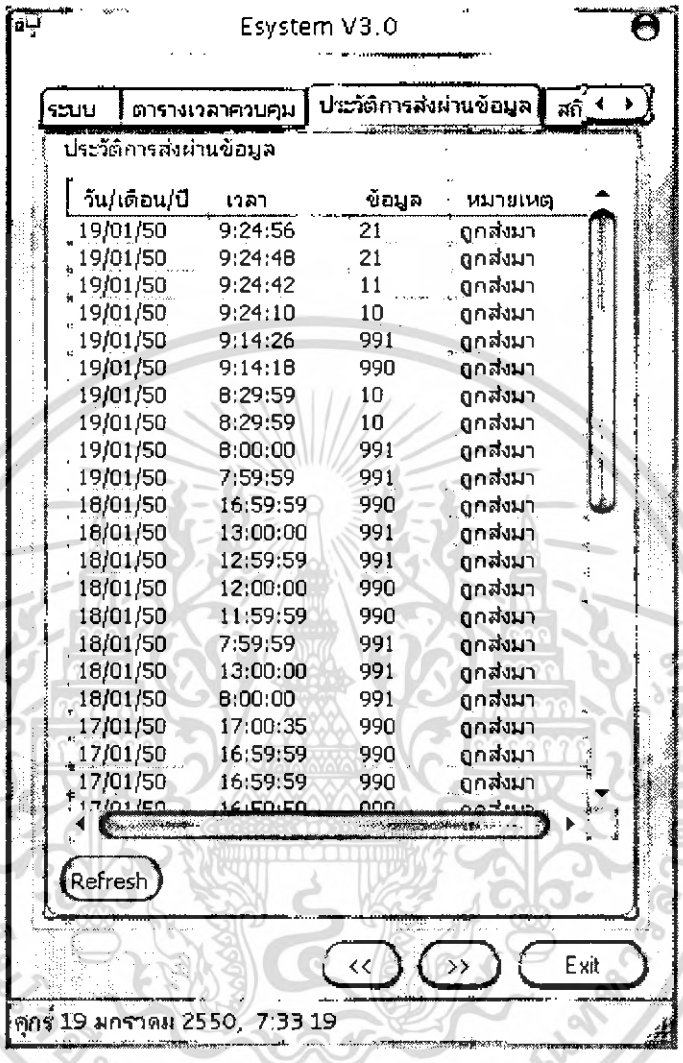
จากรูปในส่วนของ "คำสั่ง" นั้นจะแสดงรายละเอียดคำสั่งที่ต้องทำถัดไป เวลาที่เหลือจะนับถอยหลังในการทำคำสั่งถัดไป และในส่วนของ "สถานะเครื่องใช้ไฟฟ้า" นั้นจะแสดงว่าเครื่องใช้ไฟฟ้าในแต่ละห้องมีสถานะเป็นเช่นไร สังเกตที่แถบสีแดงใต้คำว่าห้องที่ 1 และ 2 เป็นสีแดงหมายความว่าไม่มีการใช้งานเครื่องใช้ไฟฟ้าในห้องนั้นๆ

2) ตารางเวลาควบคุม จะแสดงตารางของคำสั่งที่ต้องทำในวันและเวลาต่างๆ โดยตารางเวลาควบคุมหลักจะแสดงคำสั่งที่ต้องทำเป็นประจำในแต่ละวันที่กำหนด ส่วนคำสั่งควบคุมจากผู้ใช้จะเป็นคำสั่งที่เพิ่มเติมเข้ามาเป็นพิเศษจากตารางเวลาควบคุมหลัก โดยเราสามารถเพิ่ม ลบ คำสั่งต่างๆได้โดยกดปุ่ม Edit



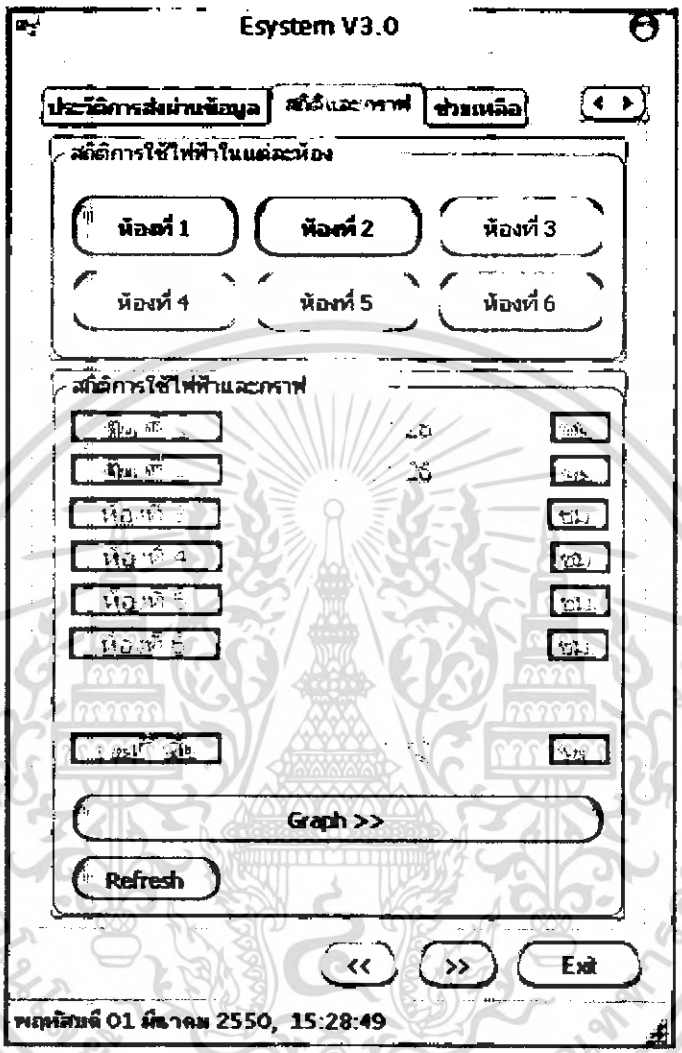
รูปที่ 3.14 แสดงหน้าต่างตารางเวลาควบคุม

3) ประวัติการส่งผ่านข้อมูล จะแสดงตารางประวัติในการส่งและรับข้อมูลทุกครั้งระหว่างพอร์ตสื่อสารอนุกรม

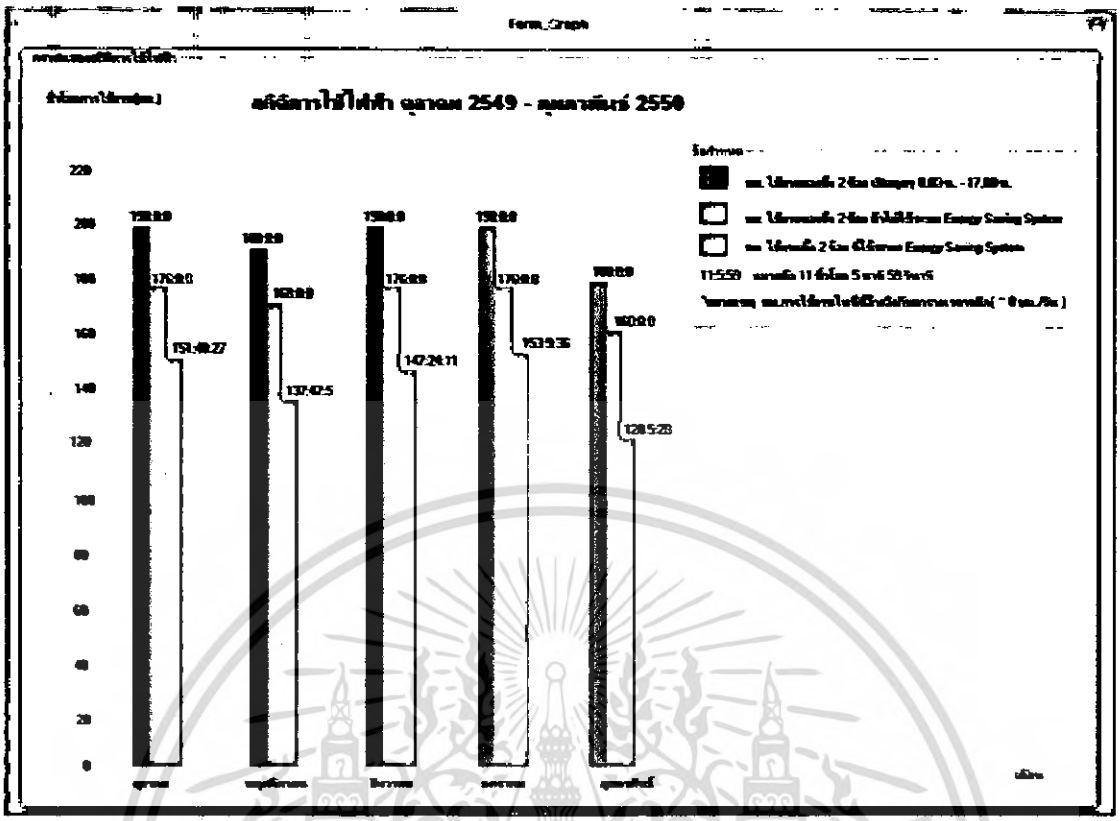


รูปที่ 3.15 แสดงหน้าต่างประวัติการส่งผ่านข้อมูล

4) สถิติและกราฟ แสดงข้อมูลของระยะเวลาในการใช้ไฟฟ้าของอุปกรณ์ไฟฟ้าในแต่ละห้องและสถิติการใช้ไฟฟ้าในแต่ละเดือนในรูปแบบกราฟ



รูปที่ 3.16 แสดงหน้าต่างสถิติการใช้ไฟฟ้า



รูปที่ 3.17 แสดงหน้าต่างกราฟ

จากรูปที่ 3.17 เส้นสีน้ำเงินแสดงจำนวนชั่วโมงการใช้งานหลอดไฟตั้งแต่เวลา 8.00-17.00

เส้นสีแดงคือจำนวนชั่วโมงการใช้งานหลอดไฟโดยที่ไม่ได้ใช้ระบบการจัดการ หลอดไฟจะปิดเวลา 12.00 และเปิดเวลา 13.00 จะเห็นว่าตัวเลขการใช้งานหลอดไฟนั้นมีค่าสูง

เส้นสีเขียวคือจำนวนชั่วโมงการใช้งานหลอดไฟเมื่อใช้ระบบบริหารจัดการและตรวจจับความเคลื่อนไหว จะเห็นว่าตัวเลขจำนวนชั่วโมงการใช้งานหลอดไฟนั้นจะมีค่าน้อยกว่าเส้นสีแดง ทำให้เราประหยัดพลังงานได้

บทที่ 4

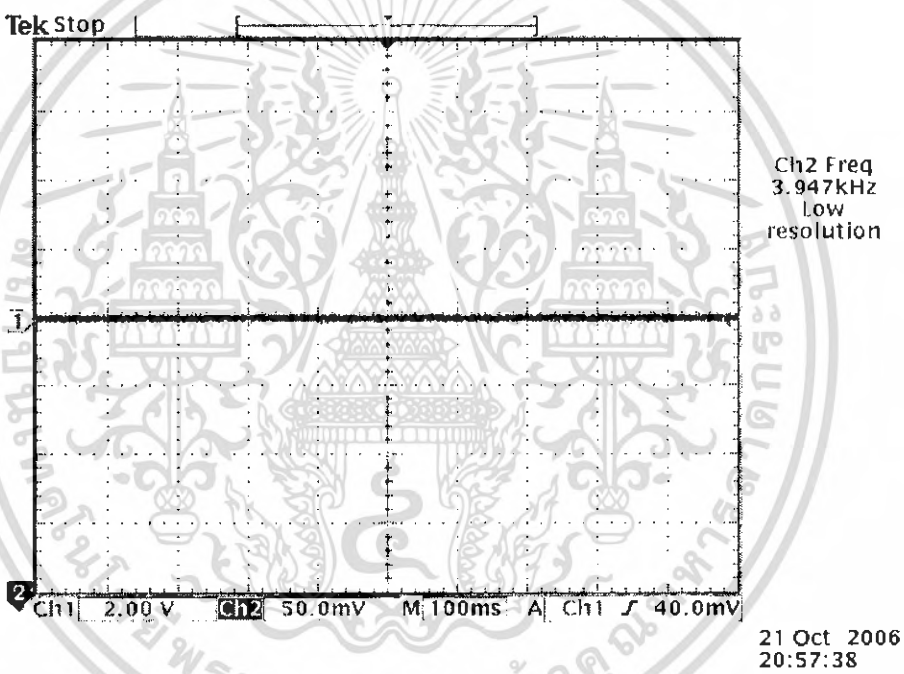
การทดลองและผลการทดลอง

การทดลองนี้จะประกอบด้วย 2 ส่วน คือ ส่วนของการรับส่งข้อมูลผ่านพอร์ตสื่อสารอนุกรม และส่วนของวงจรตรวจจับความเคลื่อนไหว

4.1 การทดลองวงจรตรวจจับความเคลื่อนไหว

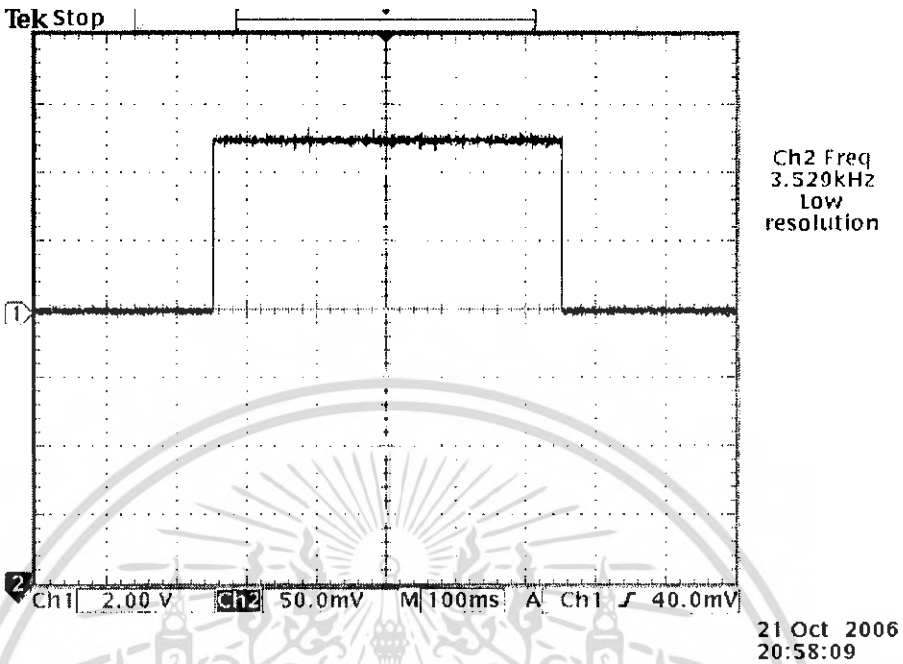
ผลการทดลอง

1. เมื่อไม่มีความเคลื่อนไหว สัญญาณเอาท์พุทของวงจร มีลักษณะดังรูปที่ 4.1



รูปที่ 4.1 แสดงสัญญาณเอาท์พุท 0 โวลต์

2. เมื่อมีความเคลื่อนไหว สัญญาณเอาท์พุทของวงจร มีลักษณะดังรูป 4.2

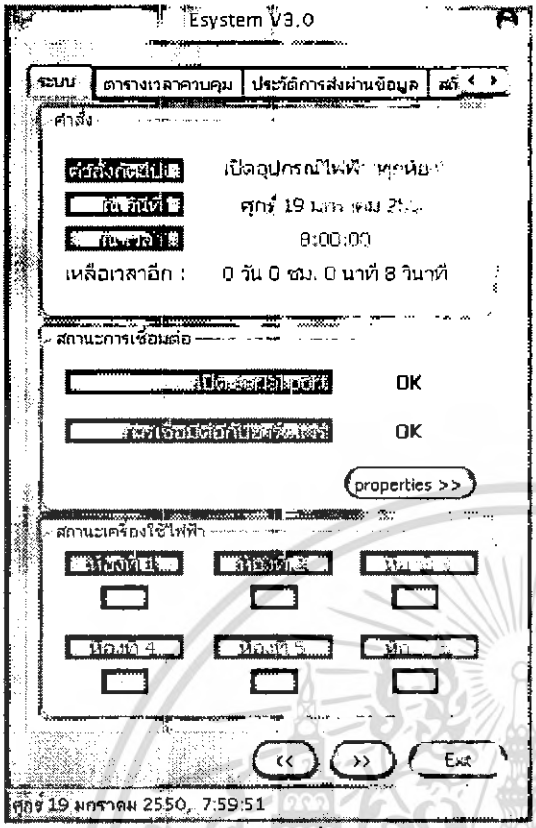


รูปที่ 4.2 แสดงสัญญาณเอาท์พุท 5 โวลต์

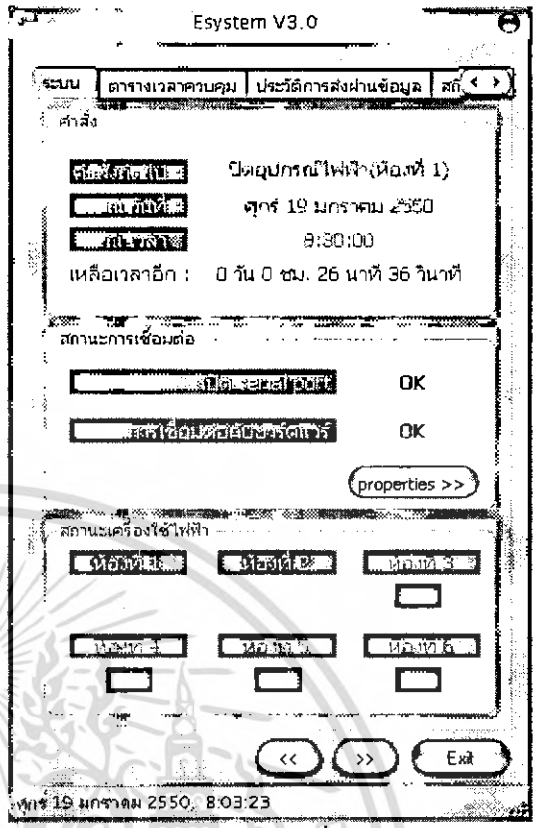
4.2 การทดลองวงจรควบคุมอุปกรณ์ไฟฟ้าโดยใช้ไมโครคอนโทรเลอร์

ผลการทดลอง

1. เมื่อถึงเวลาที่กำหนดไว้ให้ทำการเปิดไดโอดเปล่งแสงโปรแกรมจะทำการส่งคำสั่งควบคุมออกทางพอร์ตสื่อสารอนุกรม ไมโครคอนโทรเลอร์จะรับค่าของคำสั่งนั้น แล้วทำการสั่งเปิดไดโอดเปล่งแสง จากนั้นไมโครคอนโทรเลอร์ส่งค่าสถานะของไดโอดเปล่งแสงกลับมา โปรแกรมจะแสดงเวลาในการปิดหรือเปิดครั้งถัดไปดังรูป สังเกตที่สถานะเครื่องใช้ไฟฟ้าของแต่ละห้องจะเป็นสีเขียว



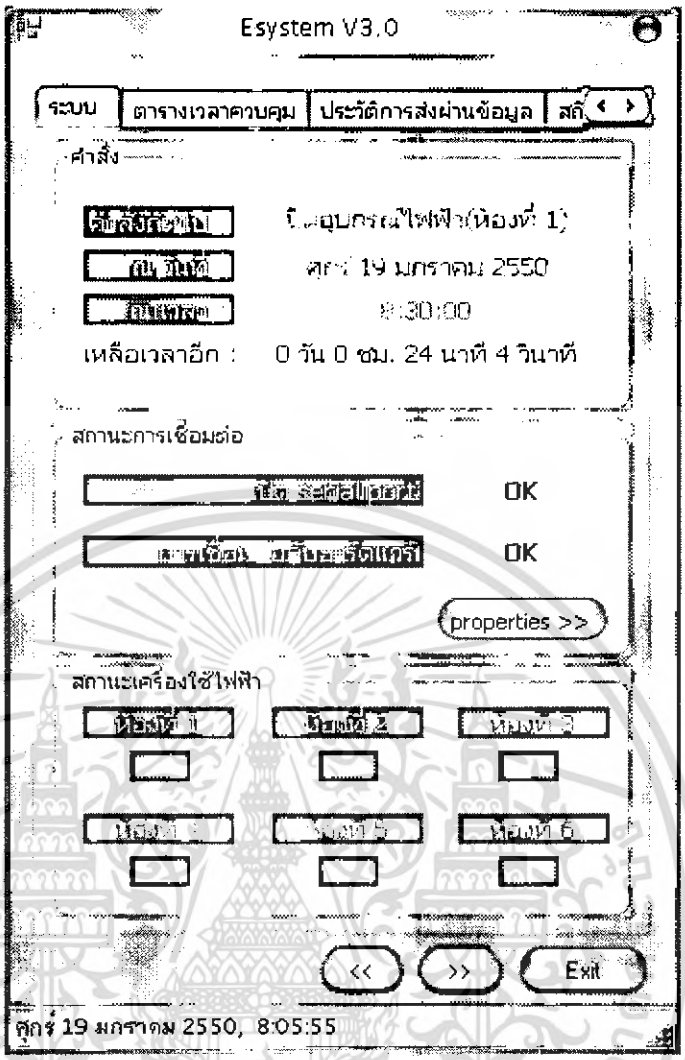
ก่อนทำคำสั่ง



หลังทำคำสั่ง

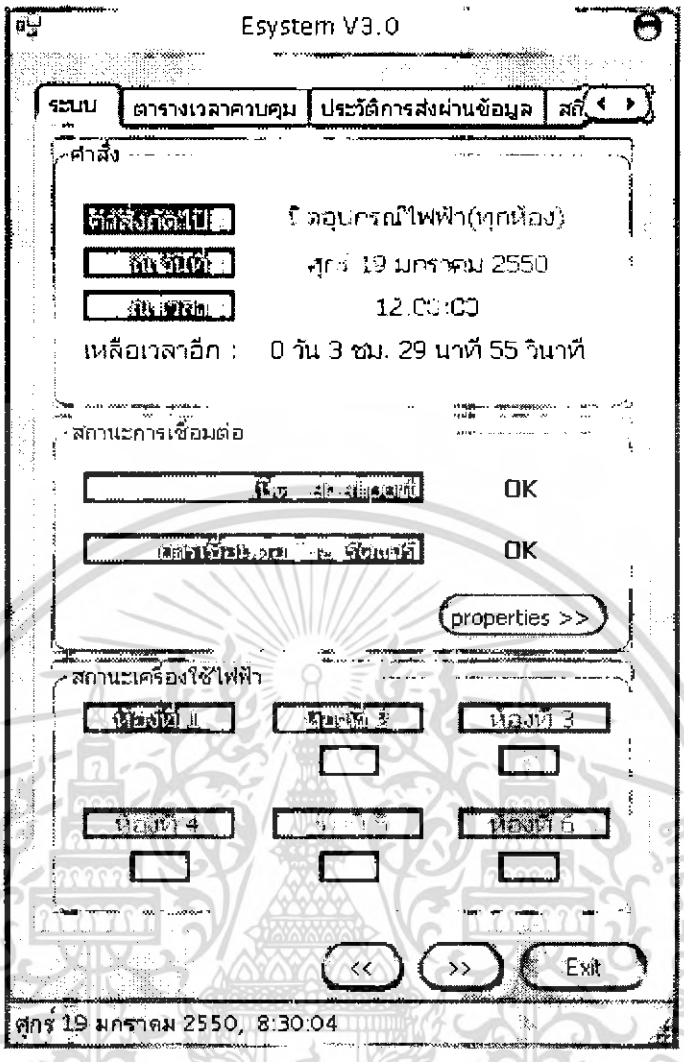
รูปที่ 4.3.1 ก่อนทำคำสั่ง "เปิดอุปกรณ์ไฟฟ้าทุกห้อง" วันศุกร์ที่ 19 มกราคม 2550 เวลา 8.00 น. (รูปซ้าย)
รูปที่ 4.3.2 หลังทำคำสั่งแล้วสถานะเครื่องใช้ไฟฟ้าห้องที่ 1 และ 2 จากปิดเปลี่ยนเป็นเปิด (รูปขวา)

2. ถ้าภายในเวลา 1.30 นาที PIR ไม่สามารถตรวจจับความเคลื่อนไหวได้ไมโครคอนโทรเลอร์จะสั่งให้ ไดโอดเปล่งแสงดับโดยอัตโนมัติแล้วส่งสถานะใหม่ กลับมายังโปรแกรมควบคุมทำให้สถานะเครื่องใช้ไฟฟ้าเปลี่ยนเป็นสีแดง(ปิด) ดังรูป



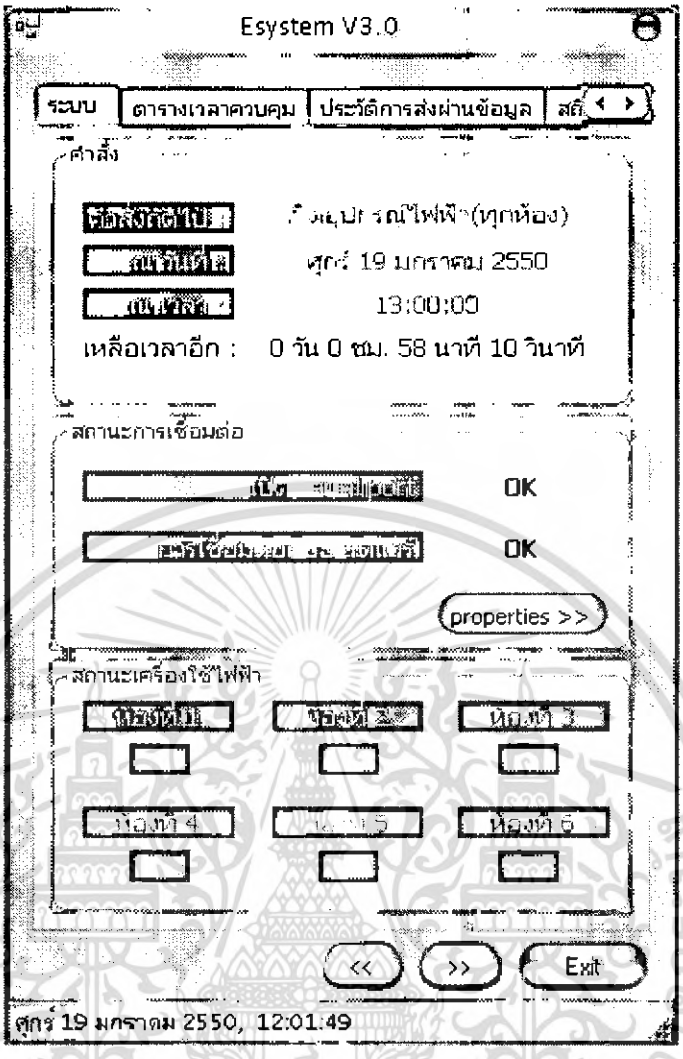
รูปที่ 4.4 แสดงสถานะหลังจากเซนเซอร์ไม่สามารถตรวจจับการเคลื่อนไหวได้ภายใน 1.30 วินาที

3. ถ้า PIR ของห้องใดตรวจจับความเคลื่อนไหวได้แล้วส่งค่ามาให้ไมโครคอนโทรเลอร์จะทำให้ไดโอดเปล่งแสงติด แล้วไมโครคอนโทรเลอร์ค่าสถานะใหม่ กลับมาให้โปรแกรมควบคุม ดังรูปที่ห้องที่ 1 เกิดการตรวจจับได้สถานะจะเปลี่ยนเป็นสีเขียวอีกครั้ง



รูปที่ 4.5 แสดงสถานะหลังจากเซนเซอร์สามารถตรวจจับการเคลื่อนไหวภายในห้องที่ 1 ได้

4. เมื่อถึงเวลาที่จะทำคำสั่งถัดไปดังข้อที่ 3 คือเวลา 12.00 น.โปรแกรมจะส่งคำสั่งปิด ไดโอดเปล่งแสงโดยส่งคำสั่งปิดผ่านพอร์ตสื่อสารอนุกรมไปให้กับไมโครคอนโทรเลอร์ไมโครคอนโทรเลอร์จะรับค่าคำสั่ง แล้วทำการสั่งปิดไดโอดเปล่งแสงแล้ว ส่งค่าสถานะของไดโอดเปล่งแสง กลับมาดังรูป และสถานะจะเปลี่ยนเป็นสีแดง



รูปที่ 4.6 แสดงสถานะหลังจากทำคำสั่ง (จากข้อที่ 3) "ปิดอุปกรณ์ไฟฟ้าทุกห้อง"
ณ เวลา 12.00 น. วันศุกร์ 19 มกราคม 2550

บทที่ 5
บทสรุปและวิจารณ์

5.1 สรุปผลการทดลอง

เมื่ออุปกรณ์ตรวจจับความเคลื่อนไหวได้จะให้สัญญาณพัลส์ลูกเล็กๆออกมา จากนั้นนำสัญญาณที่ได้ไปผ่านวงจรขยาย 2 ขั้นตอน วงจรเปรียบเทียบสัญญาณและวงจรบัฟเฟอร์ตามลำดับ พบว่าเมื่ออุปกรณ์ตรวจจับความเคลื่อนไหวไม่ได้ สัญญาณเอาท์พุทที่ได้จะมีค่า 0 โวลต์ และเมื่อตรวจจับความเคลื่อนไหวได้ สัญญาณเอาท์พุทจากบัฟเฟอร์จะมีค่า 5 โวลต์

สัญญาณที่ได้จากวงจรบัฟเฟอร์นำไปป้อนเข้าสู่พอร์ต 2.0 ของไมโครคอนโทรเลอร์ ไมโครคอนโทรเลอร์จะทำการรับค่าสัญญาณดังกล่าวมาประมวลผล ถ้าระดับแรงดันเป็น 5 โวลต์จะทำการสั่งเปิดไดโอดเปล่งแสง ถ้าระดับแรงดันเป็น 0 โวลต์นานกว่า 0.30 นาที จะทำการสั่งปิดไดโอดเปล่งแสง พร้อมกันนี้ทุกครั้งที่ไดโอดเปล่งแสงติดหรือดับไมโครคอนโทรเลอร์จะทำการส่งค่าสถานะของไดโอดเปล่งแสงกลับไปให้คอมพิวเตอร์

นอกจากนี้ไมโครคอนโทรเลอร์สามารถรับค่าสถานะของไดโอดเปล่งแสงจากโปรแกรมควบคุมบนคอมพิวเตอร์ได้ โดยสามารถโปรแกรมเวลาเปิด-ปิดอัตโนมัติของไดโอดเปล่งแสง เมื่อถึงเวลาที่กำหนดโปรแกรมจะทำการส่งข้อมูลให้ไมโครคอนโทรเลอร์สั่งเปิดหรือปิดไดโอดเปล่งแสง นอกจากนี้โปรแกรมยังสามารถสั่งเปิดหรือปิดไดโอดเปล่งแสงได้ตามต้องการและยังสามารถตรวจสอบการทำงานของไมโครคอนโทรเลอร์ว่าเป็นปกติดีหรือไม่ได้อีกด้วย

การติดต่อสื่อสารระหว่างไมโครคอนโทรเลอร์และคอมพิวเตอร์นั้น จะใช้การเชื่อมต่อแบบ RS232 ในรูปแบบ ตัวแม่/ตัวลูก (Master/Slave) ดังนั้นคอมพิวเตอร์ซึ่งทำหน้าที่เป็นอุปกรณ์ตัวแม่ต้องควบคุมอุปกรณ์ตัวลูกอันได้แก่ไมโครคอนโทรเลอร์และอุปกรณ์ตรวจจับความเคลื่อนไหวทุกๆชุด

ระบบบริหารจัดการพลังงานในโครงงานนี้ สามารถควบคุมและจัดการพลังงานได้หลายๆห้องในเวลาเดียวกันและยังสามารถรายงานผลของการใช้งานพลังงานหรือการประหยัดพลังงานให้ผู้ใช้งานระบบรับทราบได้อีกด้วย

5.2 ปัญหาที่พบในการทดลอง

1. ไม่สามารถตรวจวัดค่าสัญญาณที่ออกมาจากอุปกรณ์ตรวจจับได้โดยตรงเนื่องมาจากสัญญาณที่ออกมามีค่าน้อยมาก

2. ในการตรวจสอบการทำงานของอุปกรณ์ตรวจจับเมื่อไม่มีความเคลื่อนไหว ทำได้ค่อนข้างลำบาก ทั้งนี้เนื่องมาจากผู้ทดลองต้องไม่มีการเคลื่อนไหวใดๆทั้งสิ้น

3. ไม่มีความรู้ทางด้านภาษาซีมาก่อน จึงต้องเสียเวลาในการศึกษาค่อนข้างนาน

5.3 แนวทางพัฒนาต่อไป

1. การส่งข้อมูลระหว่างวงจรตรวจจับความเคลื่อนไหวและไมโครคอนโทรเลอร์สามารถพัฒนาเป็นการส่งข้อมูลแบบไร้สายได้

2. สามารถพัฒนาเป็นเครื่องควบคุมการเปิดปิดอุปกรณ์ต่างๆภายในบ้านหรืออาคารสำนักงานเพื่อประหยัดพลังงานได้และสามารถพัฒนาให้สั่งเปิดหรือปิดอุปกรณ์ไฟฟ้าผ่านหน้าเวปไซต์ได้

หนังสืออ้างอิง

[1] ร.ศ. ธีรวัฒน์ ประกอบผล, "ภาษาแอสเซมบลี สำหรับMCS-51", : สำนักพิมพ์ ส.ส.ท สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

[2] ร.ศ. ธีรวัฒน์ ประกอบผล, "การประยุกต์ใช้งานไมโครคอนโทรเลอร์", : สำนักพิมพ์ ส.ส.ท. สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

[3] อุดม รานอก, "ภาษา C สำหรับงานควบคุมไมโครคอนโทรเลอร์", : สำนักพิมพ์ ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์

[4] Electronics for Pyroelectic Detectors (30 สิงหาคม 2549). Available URL : http://www.perkinelmer.com

[5] Frequency Range for Pyroelectric Detectors (31 สิงหาคม 2549). Available URL : http://www.perkinelmer.com

[6] Infrared parts manual (2 กันยายน 2549). Available URL : http://www.glolab.com/pirparts/pirmanual.PDF

ภาคผนวก

1.โปรแกรมที่ใช้ในการควบคุมไมโครคอนโทรเลอร์ห้องที่ 1

```
#pragma CODE
#include <stdio.h>
#include <reg51.h>
unsigned int data1;
sbit light1=P1^0;
sbit pir=P2^0;
sbit ready=P1^1;
unsigned char chk1,chk2;


void serial_on()
{
      SBUF=0x31;
      while (TI==0)
      {
      }
            TI=0;
            chk1=1;
}
void serial_off()
{
      SBUF=0x30;
      while (TI==0)
      {
      }
            TI=0;
            chk2=1;
}
void serial_check()
{
      SBUF=0x34;
      while (TI==0)
      {
      }
            TI=1;
}

void delay1(unsigned long time)
{
      unsigned long count;
      unsigned char A,num;


      chk1=0;
      chk2=0;
      num=1;
      while(num==1)
      {
            count=0;
            while(count < time)
                  {
                  A=pir;
                        if (A==1)
                        {
                                    count=1;
```

```
                        light1=1;
                        chk2=0;
                        if (chk1==0)
                        {
                              serial_on();
                        }
                  }
                  else
                  {
                  count=count++;
                  }

                  if (RI==1)
                  {
                  RI=0;
                  data1=SBUF;

                        if (data1==0X35)
                        {
                        count=357142;
                        light1=0;
                        serial_off();
                        }
                        else if(data1==0X36)
                        {
                        light1=1;
                        count=1;
                        serial_on();
                        chk2=0;
                        }
                        else if(data1==0X33)
                        {
                        num=0;
                        light1=0;
                        count=time;
                        serial_off();
                        }
                        else if(data1==0X34)
                        {

                        serial_check();
                        }

                  }
            }
            light1=0;
            chk1=0;
            if (chk2==0)
            {
            serial_off();
            }
      }
}


void main ()
{
      PCON=0x00;
```

```
        TMOD=0X20;
        TH1=0XFD;
        SCON=0X50;
        RI=0;
        TI=0;
        TR1=1;
        while(1)
              {
              pir=0;
              light1=0;
              ready=1;
              if (RI==1)
                    {
              RI=0;
                    data1=SBUF;
                    }


              if (data1==0X31)
                    {
                    light1=1;
                    serial_on();
                    delay1(357142);
                    }

              }
}
```

2.โปรแกรมที่ใช้ในการควบคุมไมโครคอนโทรเลอร์ห้องที่ 2

```
#pragma CODE
#include <stdio.h>
#include <reg51.h>
unsigned int data1;
sbit light1=P1^0;
sbit pir=P2^0;
sbit ready=P1^1;
unsigned char chk1,chk2;


void serial_on()
{
      SBUF=0x39;
      while (TI==0)
      {
      }
            TI=0;
            chk1=1;
}
void serial_off()
{
      SBUF=0x38;
      while (TI==0)
      {
      }
            TI=0;
            chk2=1;
```

```
}
void serial_check()
{
      SBUF=0x32;
      while (TI==0)
      {
      }
            TI=1;
}

void delay1(unsigned long time)
{
      unsigned long count;
      unsigned char A,num;

      chk1=0;
      chk2=0;
      num=1;
      while(num==1)
      {
            count=0;
            while(count < time)
                  {
                  A=pir;
                        if (A==1)
                        {
                              count=1;
                              light1=1;
                              chk2=0;
                              if (chk1==0)
                              {
                                    serial_on();
                              }
                        }
                        else
                        {
                        count=count++;
                        }

                        if (RI==1)
                        {
                        RI=0;
                        data1=SBUF;

                              if (data1==0x38)
                              {
                              count=357142;
                              light1=0;
                              serial_off();
                              }
                              else if(data1==0x39)
                              {
                              light1=1;
                              count=1;
                              serial_on();
                              chk2=0;
                              }
                              else if(data1==0x30)
                              {
                              num=0;
```

```
                                light1=0;
                                count=time;
                                serial_off();
                                }
                                else if(data1==0x37)
                                {

                                serial_check();
                                }
                        }
                }
                light1=0;
                chk1=0;
                if (chk2==0)
                {
                serial_off();
                }
        }
}


void main ()
{
        PCON=0x00;
        TMOD=0x20;
        TH1=0xFD;
        SCON=0x50;
        RI=0;
        TI=0;
        TR1=1;
        while(1)
                {
                pir=0;
                light1=0;
                ready=1;
                if (RI==1)
                        {
                RI=0;
                        data1=SBUF;
                        }


                if (data1==0x32)
                        {
                        light1=1;
                        serial_on();
                        delay1(357142);
                        }

                }
}
```

3. โปรแกรมควบคุมการทำงานของหลอดไฟโดยอัตโนมัติ

1) คลาส FileManager

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
using System.IO;

namespace EsystemV3
{

    class FileManager
    {
        //-------------- file address
        public string FileAddr_FixSched =
"C:\\ESystemDatabase\\fixed.aonz"; // ตารางภารกิจหลัก
        public string FileAddr_ExSched =
"C:\\ESystemDatabase\\Extra.aonz";// ตารางภารกิจพิเศษ
        public string FileAddr_SentHistSched =
"C:\\ESystemDatabase\\SentHist.aonz";//ประวัติการส่งข้อมูล
        public string FileAddr_LastestSentCommand =
"C:\\ESystemDatabase\\NowUsing.aonz";//ช่วงเวลาการใช้งาน (เปิด-ปิด) ในแต่ละครั้ง เพื่อนำไปบันทึก
ต่อยอดที่ลงใน SumHistory
        public string FileAddr_SumHistory =
"C:\\ESystemDatabase\\SumHist.aonz"; // ประวัติการใช้งานรวมทั้งหมด รวมทั้งคำนวณ ชม. ใช้งาน
        public string FileAddr_OverallPeriod =
"C:\\ESystemDatabase\\OverallPeriod.aonz"; // ประวัติการใช้ ชม. ใช้งาน แต่ละห้อง
        public string File_Gen = "C:\\ESystemDatabase\\General.aonz";

        //-------------- data variable
        string[,] Data_MainC=new string[40,5];

        public string ConvertNumtoDay(int num)
        {
            string ThaiDay="";
            if (num == 1) { ThaiDay = "จันทร์"; }
            else if (num == 2) { ThaiDay = "อังคาร"; }
            else if (num == 3) { ThaiDay = "พุธ"; }
            else if (num == 4) { ThaiDay = "พฤหัสบดี"; }
            else if (num == 5) { ThaiDay = "ศุกร์"; }
            else if (num == 6) { ThaiDay = "เสาร์"; }
            else if (num == 7) { ThaiDay = "อาทิตย์"; }
            return ThaiDay;

        }
        public int ConvertDayToNum(string ThaiDay)
        {
            int num = 0;
            if (ThaiDay == "จันทร์") { num = 1; }
            else if (ThaiDay == "อังคาร") { num = 2; }
            else if (ThaiDay == "พุธ") { num = 3; }
            else if (ThaiDay == "พฤหัสบดี") { num = 4; }
            else if (ThaiDay == "ศุกร์") { num = 5; }
            else if (ThaiDay == "เสาร์") { num = 6; }
            else if (ThaiDay == "อาทิตย์") { num = 7; }
            return num;

        }
        public void DeleteTextFromFile(string TextToDelete,string
FileAddr)
        {
            FileManager Fmgr = new FileManager();
            string[] ResultData=new string[100];
```

```
            string[] ReadData = Fmgr.ReadFileLineToLine(FileAddr);
            int k=0;
            for (int i = 0; i < ReadData.GetLength(0); i++)
            {
                if (ReadData[i] == null)
                {
                    break;
                }
                else if (TextToDelete != ReadData[i])
                {
                    ResultData[k]=ReadData[i];
                    k++;
                 }


            }
            StreamWriter sw = new StreamWriter(FileAddr);
            int j = 0;
            while (ResultData[j] != null)
            {
                sw.WriteLine(ResultData[j]);
                j++;

            }
            sw.Close();


        }



        public void WriteTextToFile(string InputText,string
FileAddr,int Method)
        {
            //Method
            // = 1 หมายถึงเขียนต่อบรรทัดสุดท้ายจากไฟล์เดิมที่มีอยู่แล้ว
            // = 2 หมายถึงเขียนแทรกไว้บรรทัดบนสุดของไฟล์เดิม
            //
            if (Method == 1)
            {
            }
            else if (Method == 2)
            {
                string sbuffer;
                StreamReader sr = new StreamReader(FileAddr);
                sbuffer = sr.ReadToEnd();
                sr.Close();
                StreamWriter sw = new StreamWriter(FileAddr);
                sw.WriteLine(InputText);
                sw.WriteLine(sbuffer);
                sw.Close();
            }
        }
        public string[] ReadFileLineToLine(string FileAddr)
        {
            /////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);
```

```
            string[] datalist = new string[100];     // ตัวแปรเก็บค่า วัน เวลา
            int i=0;
            char[] charArray = new char[] { ',' };  // split


            //////////Read input data from file.
            string strLine = sr.ReadLine();
            while ((strLine != "") && (strLine != null))
            {
                datalist[i] = strLine;

                strLine = sr.ReadLine();

               i++;
            }
            sr.Close();

            return datalist;
        }
        public void SetupDatabase()
        {
            //////////////
DATABASE////////////////////////////////////
            //ตรวจสอบว่ามีไฟล์ดาต้าเบสในไดเรกตอรี่ที่เราต้องการหรือยังถ้ายังสร้างซะ
            string DirectPath = "C:\\ESystemDatabase";

            if (!(Directory.Exists(DirectPath)))
            {
                Directory.CreateDirectory(DirectPath);

            }

            if (!(File.Exists(FileAddr_FixSched)))
            {
                StreamWriter zz = File.CreateText(FileAddr_FixSched);
                zz.Close();
            }

            if (!(File.Exists(FileAddr_ExSched)))
            {
                StreamWriter aa = File.CreateText(FileAddr_ExSched);
                aa.Close();
            }


            if (!(File.Exists(FileAddr_SentHistSched)))
            {
                StreamWriter bb =
File.CreateText(FileAddr_SentHistSched);
                bb.Close();
            }

            if (!(File.Exists(FileAddr_LastestSentCommand)))
            {
                StreamWriter cc =
File.CreateText(FileAddr_LastestSentCommand);
                cc.Close();
            }

            if (!(File.Exists(FileAddr_SumHistory)))
```

```
            {
                StreamWriter dd =
File.CreateText(FileAddr_SumHistory);
                dd.Close();
            }


            if (!(File.Exists(File_Gen)))
            {
                StreamWriter ee = File.CreateText(File_Gen);
                ee.Close();
            }

////////////////////////////////////////////////////////////
        }



        public void ReadFile_MainCommand()
        {

            ////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr_FixSched,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);

            string[] datalist = new string[3];     // ตัวแปรเก็บค่า วัน เวลา
            string[] datalist_HMS = new string[3];
            char[] charArray = new char[] { ',' , ':' };  // split
            int index1=0,index2=0,numOfDay;
            string StrBuffer;

            //////////Read input data from file.
            string strLine = sr.ReadLine();
            while ((strLine != "") && (strLine != null))
            {
                index2 = 0;
                datalist = strLine.Split(charArray[0]);
                StrBuffer = datalist[0].ToString();
                numOfDay = int.Parse(StrBuffer);
                Data_MainC[index1,index2] =
ConvertNumtoDay(numOfDay);
                index2++;
                datalist_HMS = datalist[1].Split(charArray[1]);
                Data_MainC[index1, index2] = datalist_HMS[0];
                index2++;
                Data_MainC[index1, index2] = datalist_HMS[1];
                index2++;
                Data_MainC[index1, index2] = datalist_HMS[2];
                index2++;
                Data_MainC[index1, index2] = datalist[2];
                index2++;

                strLine = sr.ReadLine();

                index1++;
            }
            sr.Close();
```

```
        }


        public void ReadFile_UserCommand(ref DateTime[] Output,ref
int[] OutCommand,ref int Num)
        {
            ////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr_ExSched,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);

            string[] datalist = new string[7];     // ตัวแปรเก็บค่า วัน เวลา

            char[] charArray = new char[] { ',', ':' ,'/'};  // split


            //////////Read input data from file.
            string strLine = sr.ReadLine();
            int i = 0;


            while ((strLine != "") && (strLine != null))
            {
                datalist = strLine.Split(charArray);
                Output[i] = new DateTime(int.Parse(datalist[2])-543,
int.Parse(datalist[1]), int.Parse(datalist[0]),

int.Parse(datalist[3]), int.Parse(datalist[4]),
int.Parse(datalist[5]));
                OutCommand[i] = int.Parse(datalist[6]);
                strLine = sr.ReadLine();
                i++;
            }
            Num = i;
            sr.Close();

        }
        public void WriteDataToStatisticsFile(string TextToWrite)
        {
            // จะเป็นการเซ็ตช่วงเวลาการใช้ไฟฟ้าแล้วเขียนลงไฟล์
            string[] ReadFile =
ReadFileLineToLine(FileAddr_SumHistory);
            char[] ChArray = new char[] {','};
            string[] InputTextAfterSplit = new string[3];
            string[] ReadTextAfterSplit = new string[3];

            InputTextAfterSplit = TextToWrite.Split(ChArray);
            //ReadTextAfterSplit=TextToWrite..Split(ChArray);
            //เอาเฉพาะส่วนของคำสั่งมาเช็ค
            int Succeed = 0;
            int
CommandFromInputText=int.Parse(InputTextAfterSplit[2]);
             //int
CommandFromReadText=int.Parse(ReadTextAfterSplit[2]);
            int i=0;
            if((ReadFile[i]==null)&&((CommandFromInputText%10)==1))//
ไม่มีข้อมูลอยู่ในฐานข้อมูลอยู่ก่อนเลย
            {
                WriteTextToFile(TextToWrite,FileAddr_SumHistory,2);
```

```
                Succeed = 1;
            }
            else
            {
                int SuffixOfInputText=CommandFromInputText%10;
                while (ReadFile[i] != null)
                {
                    ReadTextAfterSplit=ReadFile[i].Split(ChArray);
                    if (int.Parse(ReadTextAfterSplit[2]) ==
CommandFromInputText)//เท่ากับตัวมันเอง
                    {
                        Succeed = 1;
                        break;
                    }
                    else if(SuffixOfInputText==0)//คำสั่งลงท้ายด้วย 0
                    {
                        if ((CommandFromInputText + 1) ==
int.Parse(ReadTextAfterSplit[2])) //เจอตัวมันเอง +1
                        {
                            //เขียนลงไฟล์
                            WriteTextToFile(TextToWrite,
FileAddr_SumHistory, 2);
                            Succeed = 1;
                            break;
                        }
                    }
                    else if (SuffixOfInputText == 1)//คำสั่งลงท้ายด้วย 0
                    {
                        if ((CommandFromInputText -1) ==
int.Parse(ReadTextAfterSplit[2])) //เจอตัวมันเอง +1
                        {
                            //เขียนลงไฟล์
                            WriteTextToFile(TextToWrite,
FileAddr_SumHistory, 2);
                            Succeed = 1;
                            break;
                        }
                    }
                    i++;
                }

            }

            if ((Succeed != 1)&&((CommandFromInputText%10)==1))
            {
                WriteTextToFile(TextToWrite, FileAddr_SumHistory, 2);
                Succeed = 1;
            }
        }//End of WriteDataToStatisticsFile
        public void WriteToOverallPeriodFile(int RoomNo,string
TextUsageTime)
        {
            string[] buffer =
ReadFileLineToLine(FileAddr_OverallPeriod);
            int DontHaveInFile = 1;
            for(int i=0;i<buffer.GetLength(0);i++)
            {
                if(buffer[i]!=null)
                {
                    string[] datalist = new string[2];
```

```
                    char[] ChArray=new char[]{','};
                    datalist = buffer[i].Split(ChArray);
                    if (datalist[0] == RoomNo.ToString())
                    {
                        buffer[i] = RoomNo.ToString() + "," +
TextUsageTime;
                        DontHaveInFile = 0;
                        break;
                    }

                }
                if (buffer[i] == null)
                    break;
            }

            StreamWriter sw = new
StreamWriter(FileAddr_OverallPeriod);
            if (buffer[0] == null)
            {
                sw.WriteLine(RoomNo.ToString() + "," +
TextUsageTime);

            }
            else
            {
                for (int k = 0; k < buffer.GetLength(0); k++)
                {

                    if (buffer[k] != null)
                    {
                        sw.WriteLine(buffer[k]);

                    }
                }
            }
            if (DontHaveInFile == 1)
            {
                sw.WriteLine(RoomNo.ToString() + "," +
TextUsageTime);
            }
            sw.Close();
        }

        public string[,] GetMainComData()
        {
            return Data_MainC;
        }


    }
}
```

2) คลาส Calculation

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Globalization;
namespace EsystemV3
{
    class Calculation
```

```
    {
        public string GetTodayDateTime()
        {
            return DateTime.Today.ToString("dddd dd MMMM yyyy") + ",
"+ DateTime.Now.ToLongTimeString();
        }
        public int ConvertHMStoSecond(int Hour,int Min,int sec)
        {
            return (Hour * 60 * 60) + (Min * 60) + sec;
        }
        public string ConvertEnglishDayToThaiDay(string EnglishDay)
        {
            string ThaiDay = "";
            if (EnglishDay == "Monday") { ThaiDay = "จันทร์"; }
            else if (EnglishDay == "Tuesday") { ThaiDay = "อังคาร"; }
            else if (EnglishDay == "Wednesday") { ThaiDay = "พุธ"; }
            else if (EnglishDay == "Thursday") { ThaiDay = "พฤหัสบดี"; }
            else if (EnglishDay == "Friday") { ThaiDay = "ศุกร์"; }
            else if (EnglishDay == "Saturday") { ThaiDay = "เสาร์"; }
            else if (EnglishDay == "Sunday") { ThaiDay = "อาทิตย์"; }
            return ThaiDay;
        }
        public void ConvertSecondToHMS(int Sec,ref int OutH,ref int
OutM,ref int OutS)
        {
            OutH = 0;
            OutM = 0;
            OutS = 0;
            while (Sec >= 60)
            {
                if (Sec >= (60 * 60))
                {
                    OutH = Sec / (60 * 60);
                    Sec = Sec - (OutH * 60 * 60);// เหลืออีกกี่วินาที
                }
                else if (Sec >= 60)
                {
                    OutM = Sec / 60;
                    Sec = Sec - (OutM * 60);
                }
                else if(Sec<60)
                {
                    OutS = Sec;
                }
            }

        }

        public DateTime CalculationNextDuty(ref int Command,ref
string error)//ref DateTime OutNextDuty,ref int H,ref int M,ref int
S,ref int Command)
        {
            FileManager FileMgnr = new FileManager();
            string[,] Data_MainCommand = new string[40, 5];
            int IndexOfResult = 0;
            int FoundInFile = 0;
            FileMgnr.ReadFile_MainCommand();
            Data_MainCommand = FileMgnr.GetMainComData();
```

```
            //------------- step 1 . เช็คระหว่างวันเวลาของวันนี้ กับในฐานข้อมูล fixed.apnz

            int[] ResultHMS = new int[3];

            string testTodayThaiDay;
            //----------------
            CultureInfo Thai_culInfo = new CultureInfo("th-TH");
            DateTimeFormatInfo Thai_info =
Thai_culInfo.DateTimeFormat;
            testTodayThaiDay = DateTime.Now.ToString("dddd",
Thai_info);
            //------------------------

            //------------
            int Min = -1, FirstTimeCheckMin = 0;
            int NowInSecond = 0;
            int DatabaseInSecond = 0;
            int SubstractResult = 0;
            //-----------
            DateTime Today = DateTime.Now;
            DateTime Check = Today;

            if (Data_MainCommand[0,0] == null)
            {
                error = "Error   : ไม่มีข้อมูลในฐานข้อมูล";
                return DateTime.Now;
            }
            else
            {
                //เช็คก่อนว่าวันนี้ (เช่น เสาร์) มี คำสั่งในฐานข้อมูลหรือเปล่า
                for (int k = 0; k < Data_MainCommand.GetLength(0);
k++)
                {
                    if (testTodayThaiDay == Data_MainCommand[k, 0])
                    {
                        FoundInFile = 1;
                        break;
                    }
                }

                if (FoundInFile == 1)//มี คำสั่งในฐานข้อมูล
                {
                    for (int i = 0; i <
Data_MainCommand.GetLength(0); i++)
                    {
                        if (testTodayThaiDay == Data_MainCommand[i,
0]) //วันของวันนี้ ตรงกลับวันในฐานข้อมูล
                        {
                            //แปลงเวลาให้อยู่ในหน่วยวินาที
                            NowInSecond =
ConvertHMStoSecond(DateTime.Now.Hour, DateTime.Now.Minute,
DateTime.Now.Second);
                            DatabaseInSecond =
ConvertHMStoSecond(int.Parse(Data_MainCommand[i, 1])

, int.Parse(Data_MainCommand[i, 2])

, int.Parse(Data_MainCommand[i, 3]));
                            SubstractResult = DatabaseInSecond -
NowInSecond;// ลบกัน
```

```
                                if ((FirstTimeCheckMin == 0) &&
(SubstractResult >= 0))//ทำการเช็คครั้งแรก
                                {
                                    Min = SubstractResult;
                                    IndexOfResult = i;
                                    FirstTimeCheckMin = 1;
                                }
                                else if ((FirstTimeCheckMin == 1) &&
(SubstractResult >= 0))//ไม่ใช่ครั้งแรก
                                {
                                    if (Min > SubstractResult)
                                    {
                                        Min = SubstractResult;
                                        IndexOfResult = i;
                                    }

                                }
                            }
                        }
                    }
                    if ((FoundInFile == 0) || (Min < 0)) //ไม่มีคำสั่งของวันนี้ในไฟล์
หรือ มีในไฟล์ แต่วันที่ทำคำสั่งจึงๆเป็นวันถัดไป ค่า Min ไม่เปลี่ยนแปลง
                    {
                        int z = 1, End = 0;
                        FirstTimeCheckMin = 0;
                        while (End != 1)
                        {
                            Check = Check.AddDays(z);
                            //เช็คก่อนว่าวันของ Check (เช่น เสาร์) มี คำสั่งในฐานข้อมูลหรือเปล่า
                            for (int k = 0; k <
Data_MainCommand.GetLength(0); k++)
                            {
                                string TestDay =
ConvertEnglishDayToThaiDay(Check.DayOfWeek.ToString());
                                if (TestDay == Data_MainCommand[k, 0])
                                {
                                    FoundInFile = 1;
                                    break;
                                }
                            }
                            if (FoundInFile == 1)
                            {
                                for (int k = 0; k <
Data_MainCommand.GetLength(0); k++)
                                {
                                    string NewDay =
ConvertEnglishDayToThaiDay(Check.DayOfWeek.ToString());
                                    if (NewDay == Data_MainCommand[k, 0])
                                    {
                                        if (FirstTimeCheckMin == 0)
                                        {
                                            Min =
int.Parse(Data_MainCommand[k, 1]);
                                            FirstTimeCheckMin = 1;
                                            IndexOfResult = k;
                                        }
                                        else if (FirstTimeCheckMin == 1)
                                        {
```

```
                                        if (Min >
int.Parse(Data_MainCommand[k, 1]))
                                        {
                                            Min =
int.Parse(Data_MainCommand[k, 1]);
                                            IndexOfResult = k;
                                        }
                                    }

                                }
                            }//end for
                            if (Min < 0)
                            {
                                z++;
                            }
                            else { End = 1; }

                        }//end if


                    }

                }
                //มาถึงตรงนี้จะได้ว่าตัวแปร Check เป็นค่าวันที่ที่เราต้องการ
                //และได้ค่า  IndexOfResult ซึ่งชี้ว่าเป็นบรรทัดไหนในฐานข้อมูล
                //ต่อไปก็คือทำหน้าที่ใส่ค่าเวลาให้ Check
                DateTime Result = new DateTime(Check.Year,
Check.Month, Check.Day,

int.Parse(Data_MainCommand[IndexOfResult, 1]),

int.Parse(Data_MainCommand[IndexOfResult, 2]),

int.Parse(Data_MainCommand[IndexOfResult, 3]));

                Command = int.Parse(Data_MainCommand[IndexOfResult,
4]);

                //------------------ Step 2.  ตรวจสอบกับฐานข้อมูลรอง
                DateTime[] SecondResult=new DateTime[100];
                int[] OutputCommand=new int[100];
                int numberOfData = 0;
                FileMgnr.ReadFile_UserCommand(ref SecondResult,ref
OutputCommand,ref numberOfData);

                //-----------------เก็บค่าแรกไว้อ้างอิงก่อน
                if (numberOfData != 0)
                {
                    TimeSpan Diff = SecondResult[0].Subtract(Result);
                    if (Diff.TotalSeconds < 0)
                    {
                        Result = SecondResult[0];
                        Command = OutputCommand[0];
                    }

                    for (int k = 1; k < numberOfData; k++)
                    {
                        Diff = SecondResult[k].Subtract(Result);
                        if (Diff.TotalSeconds < 0)
                        {
```

```
                        Result = SecondResult[k];
                        Command = OutputCommand[k];
                    }
                }
            }
            //---------------------------------
            return Result;

        }//end if


    }  //End CalculationNextDuty    FUNCTION

}
}
```

3) คลาส TableReport

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Globalization;

namespace EsystemV3
{
    class TableReport
    {
        //-----
        public int OverallPeriod = 0;


        public void TransferHistoryTable(ref
System.Windows.Forms.ListView ListData,string FileAddr)
        {
            //
            ListData.Clear();
            //กำหนดการแสดง กริด ListData
            ListData.GridLines = true;

            ColumnHeader col1 = ListData.Columns.Add("วัน/เดือน/ปี", 70,
HorizontalAlignment.Left);
            ColumnHeader col2 = ListData.Columns.Add("เวลา", 70,
HorizontalAlignment.Left);
            ColumnHeader col3 = ListData.Columns.Add("ข้อมูล", 50,
HorizontalAlignment.Left);
            ColumnHeader col4 = ListData.Columns.Add("หมายเหตุ", 70,
HorizontalAlignment.Left);

            /////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);

            string[] datalist = new string[4];     // ตัวแปรเก็บค่า วัน เวลา
            char[] charArray = new char[] { ',' };  // split
            //////////Read input data from file.
            string strLine = sr.ReadLine();
```

```
            while ((strLine != "") && (strLine != null))
            {

                datalist = strLine.Split(charArray);

                ListViewItem Litem = new ListViewItem(datalist);
                ListData.Items.Add(Litem);       // เพิ่มข้อมูลลงตาราง
                strLine = sr.ReadLine();

            }

            sr.Close();                                  ////close file
            ////////////////////////////
            ListData.View = View.Details;
            //
        }

        public void ExtraUserCommandTable(ref
System.Windows.Forms.ListView ListData, string FileAddr)
        {
            //
            ListData.Clear();
            //กำหนดการแสดง กริด ListData
            ListData.GridLines = true;

            ColumnHeader col1 = ListData.Columns.Add("วัน/เดือน/ปี", 70,
HorizontalAlignment.Left);
            ColumnHeader col2 = ListData.Columns.Add("เวลา", 70,
HorizontalAlignment.Left);
            ColumnHeader col3 = ListData.Columns.Add("คำสั่ง", 70,
HorizontalAlignment.Left);

            ////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);

            string[] datalist = new string[3];     // ตัวแปรเก็บค่า วัน เวลา
            char[] charArray = new char[] { ',' };  // split
            //////////Read input data from file.
            string strLine = sr.ReadLine();

            while ((strLine != "") && (strLine != null))
            {

                datalist = strLine.Split(charArray);

                ListViewItem Litem = new ListViewItem(datalist);
                ListData.Items.Add(Litem);       // เพิ่มข้อมูลลงตาราง
                strLine = sr.ReadLine();

            }

            sr.Close();                                  ////close file
            ////////////////////////////
            ListData.View = View.Details;
            //
        }
```

```
        public void MainCommandTable(ref
System.Windows.Forms.ListView ListData, string FileAddr)
        {
            //
            ListData.Clear();
            //กำหนดการแสดง กริด ListData
            ListData.GridLines = true;

            ColumnHeader col1 = ListData.Columns.Add("ทุกๆวัน", 70,
HorizontalAlignment.Left);
            ColumnHeader col2 = ListData.Columns.Add("เวลา", 70,
HorizontalAlignment.Left);
            ColumnHeader col3 = ListData.Columns.Add("ข้อมูล", 50,
HorizontalAlignment.Left);



            /////////////// เปิดไฟล์ fixed.aonz
            FileStream aFile = new FileStream(FileAddr,
FileMode.Open);
            StreamReader sr = new StreamReader(aFile);

            string[] datalist = new string[3];     // ตัวแปรเก็บค่า วัน เวลา
            char[] charArray = new char[] { ',' };  // split
            ///////////Read input data from file.
            string strLine = sr.ReadLine();

            while ((strLine != "") && (strLine != null))
            {

                datalist = strLine.Split(charArray);
                FileManager FF=new FileManager();
                datalist[0] =
FF.ConvertNumtoDay(int.Parse(datalist[0]));
                ListViewItem Litem = new ListViewItem(datalist);
                ListData.Items.Add(Litem);       // เพิ่มข้อมูลลงตาราง
                strLine = sr.ReadLine();

            }

            sr.Close();                                  ////close file
            ////////////////////////////
            ListData.View = View.Details;
            //
        }
        public void StatisticsTable(ref System.Windows.Forms.ListView
ListData, string FileAddr,int RoomNo)
        {
            //กำหนดตัวแปรว่าจะให้เช็คห้องไหน
            string RoomCommandTurnOn = "", RoomCommandTurnOff = "";
            RoomCommandTurnOn = RoomNo.ToString() + "1";
            RoomCommandTurnOff = RoomNo.ToString() + "0";
            string EveryRoomTurnOn = "", EveryRoomTurnOff = "";
            EveryRoomTurnOn = "991";
            EveryRoomTurnOff = "990";

            //
            ListData.Clear();
            //กำหนดการแสดง กริด ListData
```

```
            ListData.GridLines = true;

            ColumnHeader col1 = ListData.Columns.Add("เวลาที่เปิด", 90,
HorizontalAlignment.Left);
            ColumnHeader col2 = ListData.Columns.Add("เวลาที่ปิด", 90,
HorizontalAlignment.Left);
            ColumnHeader col3 = ListData.Columns.Add("ระยะเวลาใช้งาน", 100,
HorizontalAlignment.Left);



            string[] datalist = new string[7];     // ตัวแปรเก็บค่า วัน เวลา
            string[] Result = new string[3];     // ตัวแปรเก็บค่า วัน เวลา
            char[] charArray = new char[] { ',' , '/' ,':' };  //
split
            //////////Read input data from file.

            int First=0;
            FileManager FF = new FileManager();
            DateTime D1 = new DateTime();
            DateTime D2 = new DateTime();
            TimeSpan Diff = new TimeSpan();

            string[]
ReadFile=FF.ReadFileLineToLine(FF.FileAddr_SumHistory);
            //หาจำนวนข้อมูลที่มีทั้งหมด
            int num=0;
            while (ReadFile[num] != null)
            {
                num++;
            }
            num--;
            for (int i =num; i >= 0; i--)
            {
                if (ReadFile[i] != null)
                {
                    if (First == 0)
                    {
                        datalist = ReadFile[i].Split(charArray);
                        if ((datalist[6] == RoomCommandTurnOn) ||
(datalist[6] == RoomCommandTurnOff)
                            || (datalist[6] == EveryRoomTurnOn) ||
(datalist[6] == EveryRoomTurnOff))
                        {
                            D1 = new DateTime(int.Parse(datalist[2])
- 543, int.Parse(datalist[1]), int.Parse(datalist[0]),

int.Parse(datalist[3]), int.Parse(datalist[4]),
int.Parse(datalist[5]));

                            First++;
                        }
                    }
                    else
                    {
                        datalist = ReadFile[i].Split(charArray);
                        if ((datalist[6] == RoomCommandTurnOn) ||
(datalist[6] == RoomCommandTurnOff)
```

```
                            || (datalist[6] == EveryRoomTurnOn) ||
(datalist[6] == EveryRoomTurnOff))
                        {
                            D2 = new DateTime(int.Parse(datalist[2])
- 543, int.Parse(datalist[1]), int.Parse(datalist[0]),

int.Parse(datalist[3]), int.Parse(datalist[4]),
int.Parse(datalist[5]));
                            Diff = D2.Subtract(D1);

                            CultureInfo Thai_culInfo = new
CultureInfo("th-TH");
                            DateTimeFormatInfo Thai_info =
Thai_culInfo.DateTimeFormat;
                            Result[0] = D1.ToString();
                            Result[1] = D2.ToString();

                            string UsageTime =
Convert.ToString((Diff.Days * 24) + (Diff.Hours)) + ":" +
Convert.ToString(Diff.Minutes)
                                                                    +
":" + Convert.ToString(Diff.Seconds);
                            Result[2] = UsageTime;//ระยะเวลาใช้งาน

                            OverallPeriod = OverallPeriod +
(int)Diff.TotalSeconds;//ระยะเวลาใช้งานทั้งหมด

                            Main MM = new Main();

                            ListViewItem Litem = new
ListViewItem(Result);
                            ListData.Items.Add(Litem);      // เพิ่มข้อมูลลง
ตาราง
                            //หาค่า D1 ใหม่
                        }
                        i--;
                        if (i >= 0)
                        {
                            datalist = ReadFile[i].Split(charArray);
                            if ((datalist[6] == RoomCommandTurnOn) ||
(datalist[6] == RoomCommandTurnOff)
                                || (datalist[6] == EveryRoomTurnOn) ||
(datalist[6] == EveryRoomTurnOff))
                            {
                                D1 = new
DateTime(int.Parse(datalist[2]) - 543, int.Parse(datalist[1]),
int.Parse(datalist[0]),

int.Parse(datalist[3]), int.Parse(datalist[4]),
int.Parse(datalist[5]));
                            }
                        }
                    }
                }

            }
                ////////////////////////////////
                ListData.View = View.Details;
            //
        }
```

```
    }
}
```

4) คลาส StatusChecker

```
using System;
using System.Collections.Generic;
using System.Text;

namespace EsystemV3
{
    class StatusChecker
    {
        public int StatusCheck(ref System.IO.Ports.SerialPort
SPort,ref int DataReceive,ref string Text)
        {
            SPort.WriteLine("3");
            if (DataReceive == 3) //ส่ง 3 กลับมาแสดงว่ามีการเชื่อมต่อสมบูรณ์
            {
                Text = "OK";
                DataReceive = 999;//เคลียร์ค่า
                return 1;
            }
            else
            {
                Text = "Failed";
                DataReceive = 999;//เคลียร์ค่า
                return 0;
            }

        }
        public int RoomStatus(ref System.IO.Ports.SerialPort SPort,
ref int DataReceive,int RoomNumber)
        {
            SPort.WriteLine("981");
            string CheckCode=RoomNumber.ToString();
            if (DataReceive.ToString() == (CheckCode + "1")) //เปิดอยู่
            {
                DataReceive = 999;//เคลียร์ค่า
                return 1;
            }
            else if (DataReceive.ToString() == (CheckCode + "0"))//ปิด
อยู่
            {
                DataReceive = 999;//เคลียร์ค่า
                return 0;
            }
            else
                return 0;

        }


    }
}
```

5) คลาส SerialPort

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO.Ports;

namespace EsystemV3
{
    class SerialPort
    {
        public int OpenOK = 0;
        public SerialPort()// Constructor
        {

        }
        public int Open(System.IO.Ports.SerialPort SP)
        {
            //#################### Serial port
            ////เปิดการทำงานของพอร์ต
            //open serial port
                SP.Open();
                OpenOK = 1;
                return OpenOK;//เปิดได้

        }
        public void Close(System.IO.Ports.SerialPort SP)
        {
            SP.Close();
        }
        public void Write(System.IO.Ports.SerialPort SP, string data)
        {
            SP.Write(data);
        }


    }
}
```

6) คลาส Admin (กำลังพัฒนา)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace EsystemV3
{
    class Admin
    {
        public string Username = "admin";
        public string Password = "13579";
        public string GetUsername()
        {
            return Username;
        }
        public string GetPassword()
        {
            return Password;
        }
```

Technical Information of Pyro-electric Infra-red Sensors

(红外线感应元)

TYPE : PIS-209S

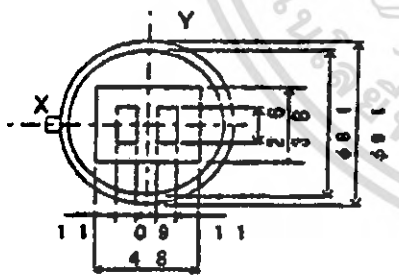
SPEC No : IPI209S-KC100

Equivalent Spec. : IPI209S-KC000

Characteristics 类型		209S	Unit	Test Conditions (20±15°C)
Detector Type		Dual element	-	-
Housing		TO-5	-	-
Responsivity (灵敏度)	min. (最小)	1.19	Vp-p	Chopper, 1Hz Black body, 165°C 40dB Amp Supply Voltage, 5VDC Load Resistance, 47kΩ (负载电阻)
		(130)	(%)	
Noise Voltage (噪音电压)	max (最大)	300	mVp-p	73dB Amp. Supply Voltage, 5VDC Load Resistance, 47kΩ
Source voltage (伏应电压)	min. (最小)	0.3	V	40dB Amp. Supply Voltage, 5VDC Load Resistance, 47kΩ
	typ. (一般)	0.7		
	max (最大)	1.5		
Storage temperature (储存温度)		-20 to +70	℃	-
Operating temperature (工作温度)		-10 to +50	℃	-
Operating voltage (工作电压)	min. (最小)	3	V	DC
	typ. (一般)	5		
	max. (最大)	15		

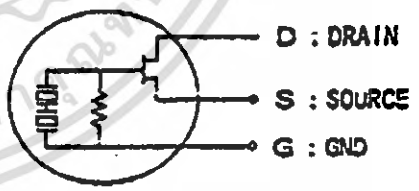
notes : Load Resistance is connected between Source pin and GND.

TOP VIEW (上视)

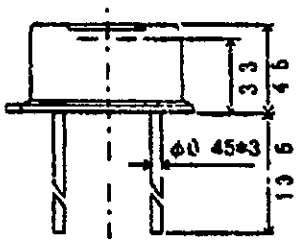


BOTTOM VIEW (下视)

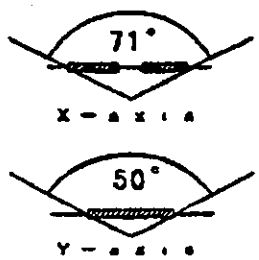
CIRCUIT DIAGRAM (线路图)



SIDE VIEW (侧视)



FIELD OF VIEW

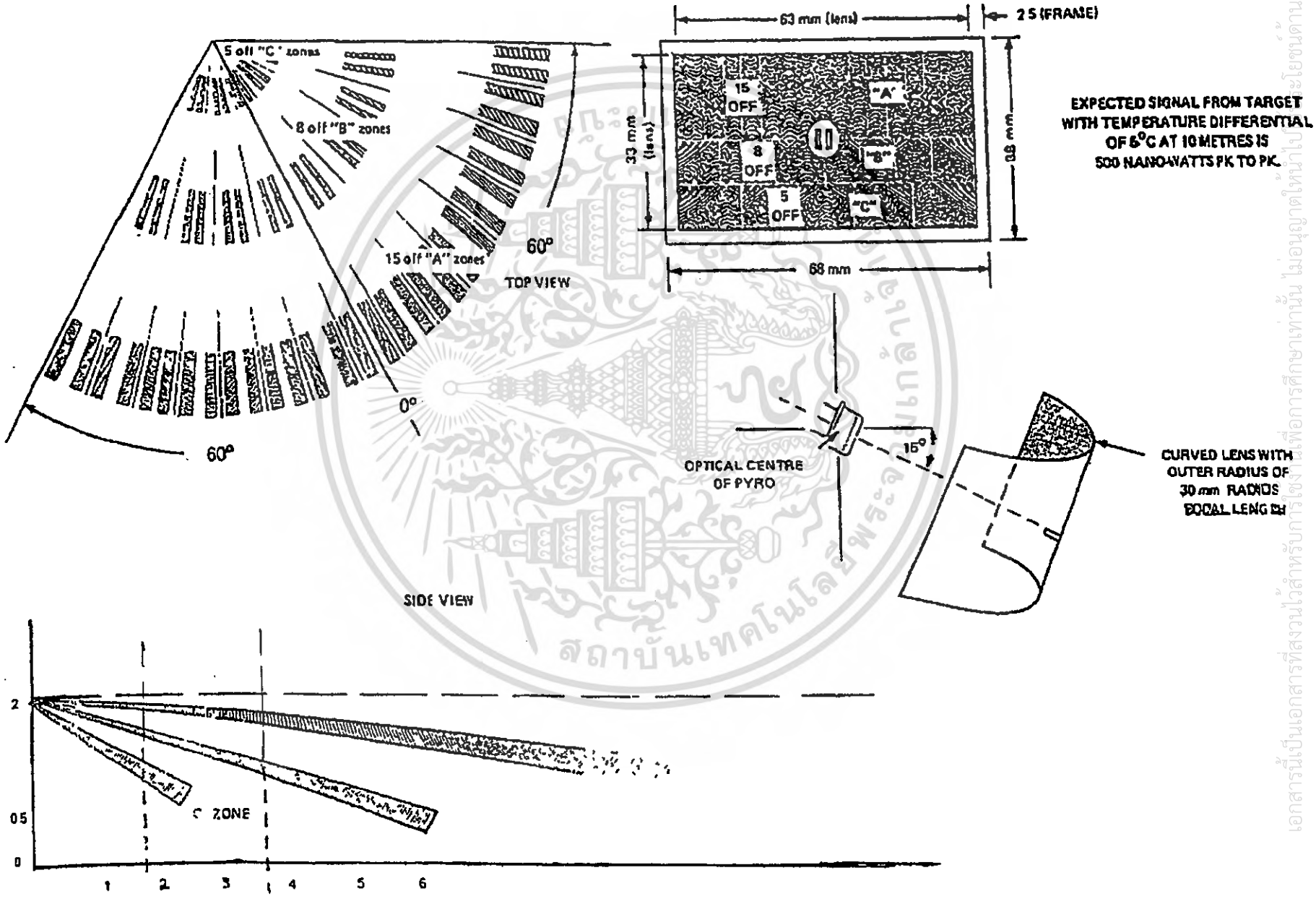


DAISHINKU Corp.

FRESNEL LENS

MULTI-SEGMENT (28)
EXTRA-WIDE ANGLE LENS
IN 3 PLANES.

WILL GIVE 56 DETECTION ZONES OVER 120° WHEN USED WITH A DUAL ELEMENT PYROSENSOR HAVING MINIMUM 120° FIELD OF VIEW.



19-4323; Rev 11; 2/03

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

**Contact factory for dice specifications.*

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ..... ±30V
R_{IN} (MAX220) ..... ±25V
T_{OUT} (Except MAX220) (Note 1) ..... ±15V
T_{OUT} (MAX220) ..... ±13.2V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) 440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) 842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10s) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kbps
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

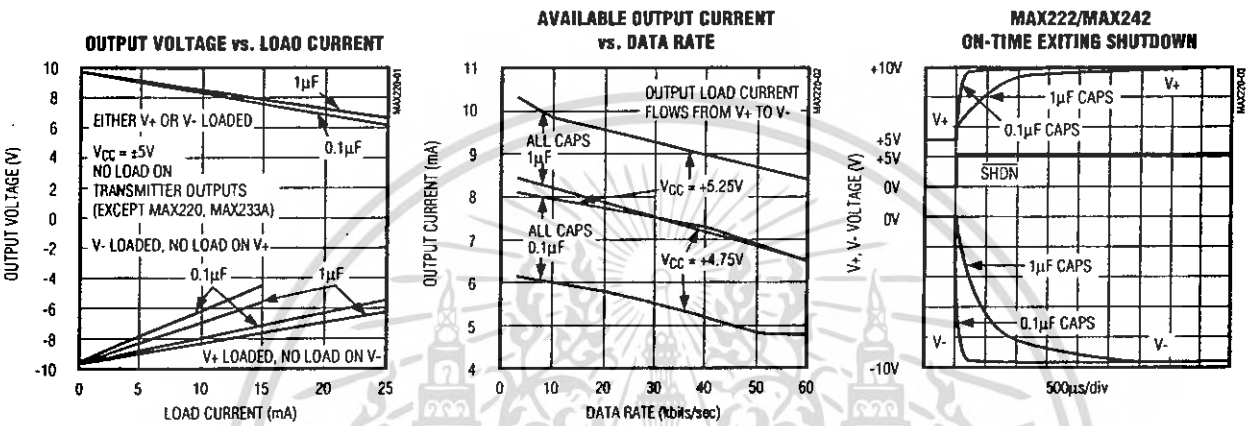
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (Normal Operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (Normal Operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (Shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ Goes High), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ Goes Low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (Normal Operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (Normal Operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

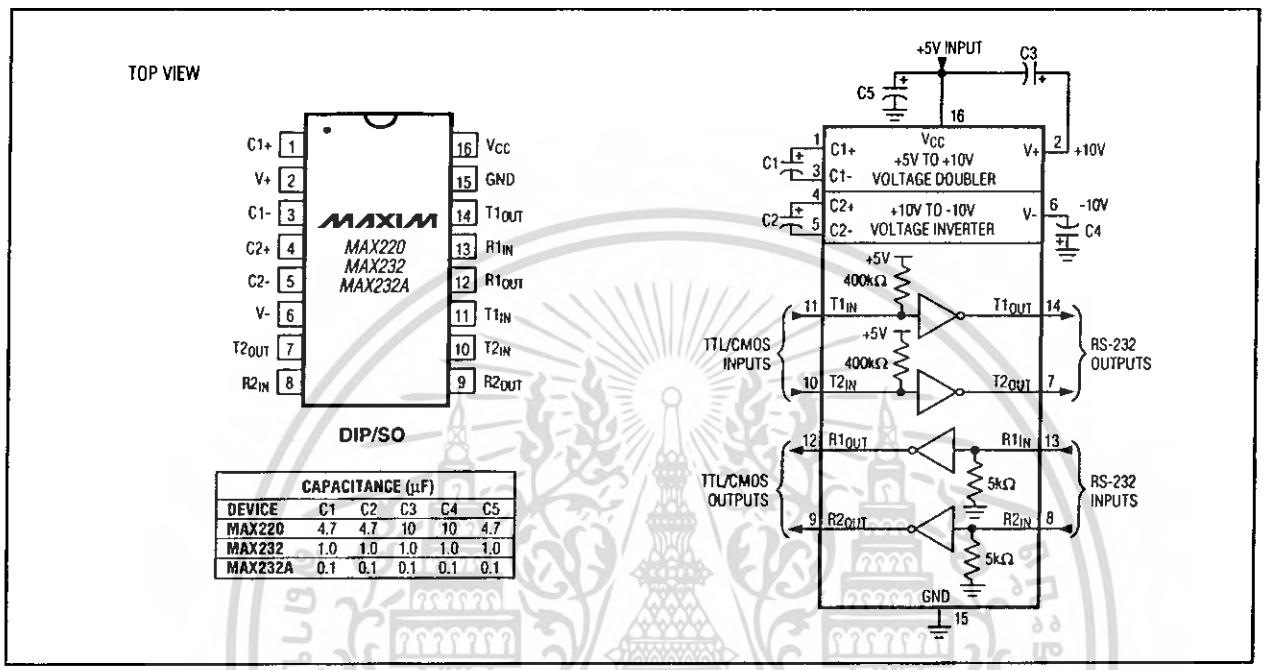
Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243





CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

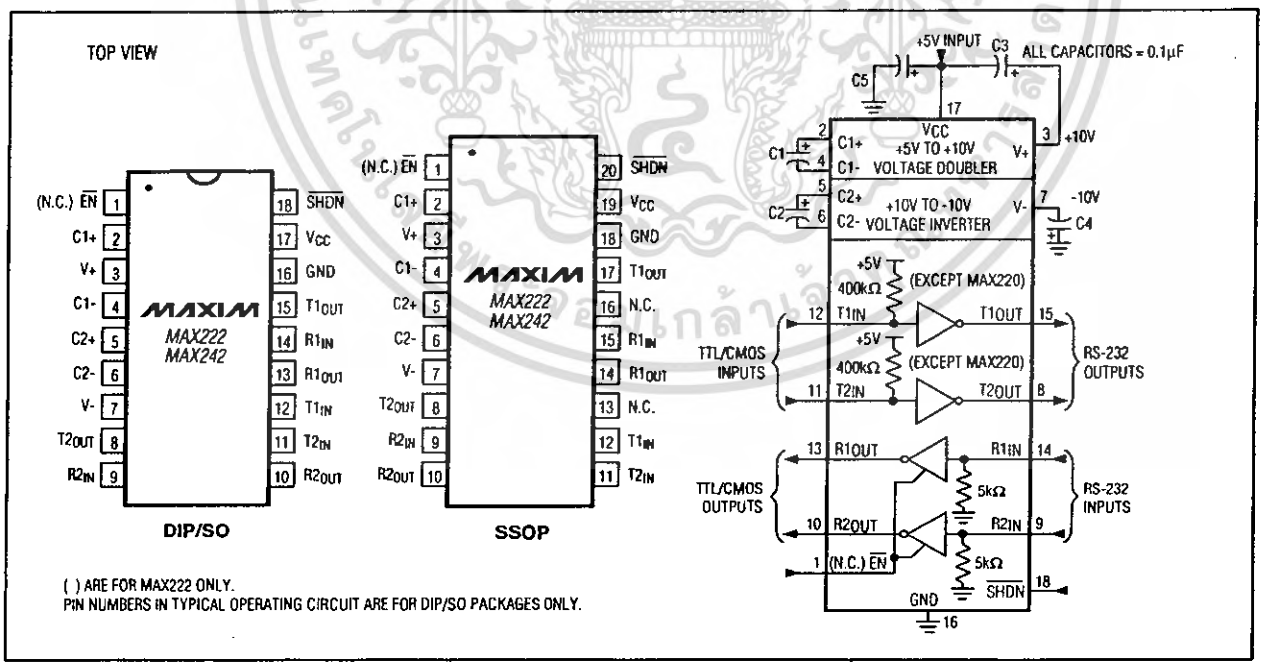


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS189

January 1998

CD74HC540, CD74HCT540, CD74HC541, CD74HCT541

High Speed CMOS Logic Octal Buffer and Line Drivers, Three-State

Features

- CD74HC540, CD74HCT540 . . . . . Inverting
- CD74HC541, CD74HCT541 . . . . . Non-Inverting
- Buffered Inputs
- Three-State Outputs
- Bus Line Driving Capability
- Typical Propagation Delay = 9ns at V_{CC} = 5V, C_L = 15pF, T_A = 25°C
- Fanout (Over Temperature Range)
  - Standard Outputs . . . . . 10 LSTTL Loads
  - Bus Driver Outputs . . . . . 15 LSTTL Loads
- Wide Operating Temperature Range . . . -55°C to 125°C
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- HC Types
  - 2V to 6V Operation
  - High Noise Immunity: N_{IL} = 30%, N_{IH} = 30% of V_{CC} at V_{CC} = 5V
- HCT Types
  - 4.5V to 5.5V Operation
  - Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)
  - CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}

Description

The Harris CD74HC540 and CD74HCT540 are Inverting Octal Buffers and Line Drivers with Three-State Outputs and the capability to drive 15 LSTTL loads. The Harris CD74HC541 and CD74HCT541 are Non-Inverting Octal Buffers and Line Drivers with Three-State Outputs that can drive 15 LSTTL loads. The Output Enables ($\overline{OE1}$) and ($\overline{OE2}$) control the Three-State Outputs. If either $\overline{OE1}$ or $\overline{OE2}$ is HIGH the outputs will be in the high impedance state. For data output $\overline{OE1}$ and $\overline{OE2}$ both must be LOW.

Ordering Information

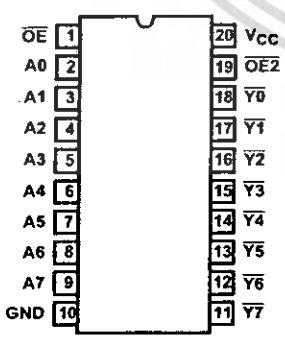
PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CD74HC540E	-55 to 125	20 Ld PDIP	E20.3
CD74HCT540E	-55 to 125	20 Ld PDIP	E20.3
CD74HC541E	-55 to 125	20 Ld PDIP	E20.3
CD74HCT541E	-55 to 125	20 Ld PDIP	E20.3
CD74HC540M	-55 to 125	20 Ld SOIC	M20.3
CD74HCT540M	-55 to 125	20 Ld SOIC	M20.3
CD74HC541M	-55 to 125	20 Ld SOIC	M20.3
CD74HCT541M	-55 to 125	20 Ld SOIC	M20.3

NOTES:

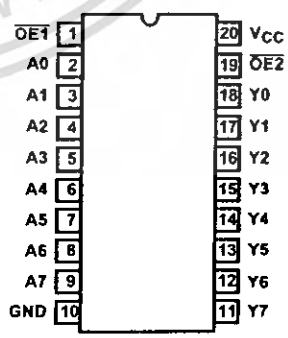
1. When ordering, use the entire part number. Add the suffix 96 to obtain the variant in the tape and reel.
2. Wafer and die for this part number is available which meets all electrical specifications. Please contact your local sales office or Harris customer service for ordering information.

Pinouts

CD74HC540, CD74HCT540
(PDIP, SOIC)
TOP VIEW



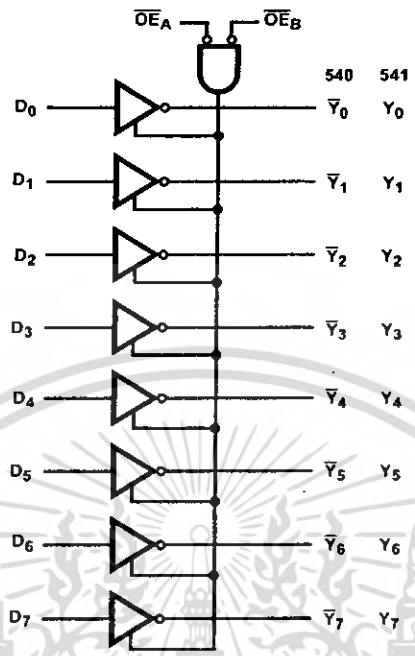
CD74HC541, CD74HCT541
(PDIP, SOIC)
TOP VIEW



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

File Number 1659.2

Functional Diagram



TRUTH TABLE

INPUTS			OUTPUTS	
$\overline{OE1}$	$\overline{OE2}$	An	540	541
L	L	H	L	H
H	X	X	Z	Z
X	H	X	Z	Z
L	L	L	H	L

NOTE:
H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care
Z = High Impedance

Absolute Maximum Ratings

DC Supply Voltage, V_{CC} ..... -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V ..... ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V ..... ±20mA
DC Drain Current, per Output, I_O
For -0.5V < V_O < V_{CC} + 0.5V ..... ±35mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V ..... ±25mA
DC V_{CC} or Ground Current, I_{CC} ..... ±50mA

Operating Conditions

Temperature Range, T_A ..... -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types ..... 2V to 6V
HCT Types ..... 4.5V to 5.5V
DC Input or Output Voltage, V_I, V_O ..... 0V to V_{CC}
Input Rise and Fall Time
2V ..... 1000ns (Max)
4.5V ..... 500ns (Max)
6V ..... 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 3)	θ_{JA} (°C/W)
PDIP Package	125
SOIC Package	120

Maximum Junction Temperature ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 300°C
(SOIC - Lead Tips Only)

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

NOTE:

3. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			-6	4.5	3.98	-	-	3.84	-	3.7	-	V
			-7.8	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			6	4.5	-	-	0.26	-	0.33	-	0.4	V
			7.8	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS		V_{CC} (V)	25°C			-40°C TO 85°C		-55°C TO 125°C		UNITS
		V_I (V)	I_O (mA)		MIN	TYP	MAX	MIN	MAX	MIN	MAX	
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	8	-	80	-	160	µA
Three- State Leakage Current	I_{OZ}	V_{IL} or V_{IH}	V_O = V_{CC} or GND	6	-	-	±0.5	-	±5.0	-	±10	µA
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-6	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			6	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} and GND	0	5.5	-		±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	8	-	80	-	160	µA
Three- State Leakage Current	I_{OZ}	V_{IL} or V_{IH}	V_O = V_{CC} or GND	5.5	-	-	±0.5	-	±5.0	-	±10	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load	ΔI_{CC}	V_{CC} -2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE: For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS	
	HCT540	HCT541
A0 - A7	1	0.4
$\overline{OE2}$	0.75	0.75
$\overline{OE1}$	1.15	1.15

NOTE: Unit load is ΔI_{CC} limit specific in DC Electrical Specifications Table, e.g., 360µA max. at 25°C.

Switching Specifications C_L = 50pF, Input t_r, t_f = 6ns

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C			-40°C TO 85°C		-55°C TO 125°C		UNITS
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
HC TYPES											
Propagation Delay Data to Outputs (540)	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	110	-	140	-	165	ns
			4.5	-	-	22	-	28	-	33	ns
		C_L = 15pF	5	-	9	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	19	-	24	-	28	ns
Data to Outputs (541)	t_{PLZ}, t_{PHZ}	C_L = 50pF	2	-	-	115	-	145	-	175	ns
			4.5	-	-	23	-	29	-	35	ns
		C_L = 15pF	5	-	9	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	20	-	25	-	30	ns
Output Enable and Disable to Outputs (540)	t_{PLZ}, t_{PHZ}	C_L = 50pF	2	-	-	160	-	200	-	240	ns
			4.5	-	-	32	-	40	-	48	ns
		C_L = 15pF	5	-	13	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	27	-	34	-	41	ns
Output Enable and Disable to Outputs (541)	t_{PLZ}, t_{PHZ}	C_L = 50pF	2	-	-	160	-	200	-	240	ns
			4.5	-	-	32	-	40	-	48	ns
		C_L = 15pF	5	-	14	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	23	-	29	-	35	ns
Output Transition Time	t_{THL}, t_{TLH}	C_L = 50pF	2	-	-	60	-	75	-	90	ns
			4.5	-	-	12	-	15	-	18	ns
			6	-	-	10	-	13	-	15	ns
Input Capacitance	C_I	C_L = 50pF	-	10	-	10	-	10	-	10	pF
Three-State Output Capacitance	C_O	-	-	20	-	20	-	20	-	20	pF
Power Dissipation Capacitance (Notes 4, 5) (540)	C_{PD}	C_L = 15pF	5	-	50	-	-	-	-	-	pF
Power Dissipation Capacitance (Notes 4, 5) (541)	C_{PD}	C_L = 15pF	5	-	48	-	-	-	-	-	pF
HCT TYPES											
Propagation Delay Data to Outputs (540)	t_{PHL}, t_{PLH}	C_L = 50pF	4.5	-	-	24	-	30	-	36	ns
		C_L = 15pF	5	-	9	-	-	-	-	-	ns
Data to Outputs (541)	t_{PHL}, t_{PLH}	C_L = 50pF	4.5	-	-	28	-	35	-	42	ns
		C_L = 15pF	5	-	11	-	-	-	-	-	ns
Output Enable and Disable to Outputs (540, 541)	t_{PLZ}, t_{PHZ}	C_L = 50pF	4.5	-	-	35	-	44	-	53	ns
		C_L = 15pF	5	-	14	-	-	-	-	-	ns
Output Transition Time	t_{TLH}, t_{THL}	C_L = 50pF	4.5	-	-	12	-	15	-	18	ns
Input Capacitance	C_I	C_L = 50pF	-	10	-	10	-	10	-	10	pF

Switching Specifications C_L = 50pF, Input t_r, t_f = 6ns **(Continued)**

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C			-40°C TO 85°C		-55°C TO 125°C		UNITS
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
Three-State Output Capacitance	C_O	-	-	20	-	20	-	20	-	20	pF
Power Dissipation Capacitance (Notes 4, 5) (540, 541)	C_{PD}	C_L = 15pF	5	-	55	-	-	-	-	-	pF

NOTES:

4. C_{PD} is used to determine the dynamic power consumption, per channel.
5. $P_D = V_{CC}^2 f_i (C_{PD} + C_L)$ where f_i = Input Frequency, C_L = Output Load Capacitance, V_{CC} = Supply Voltage.

Test Circuits and Waveforms

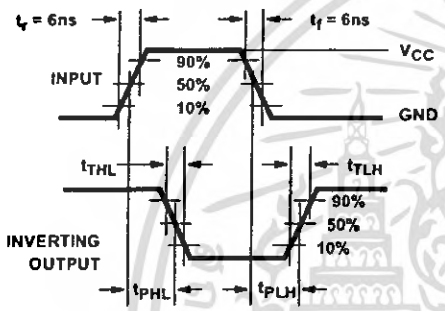


FIGURE 1. HC TRANSITION TIMES AND PROPAGATION DELAY TIMES, COMBINATION LOGIC

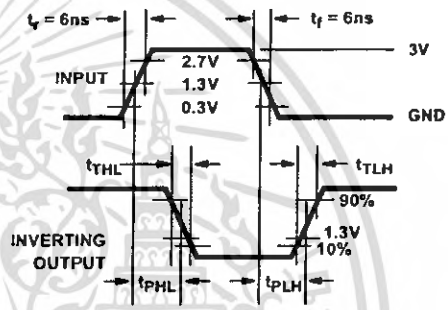


FIGURE 2. HCT TRANSITION TIMES AND PROPAGATION DELAY TIMES, COMBINATION LOGIC

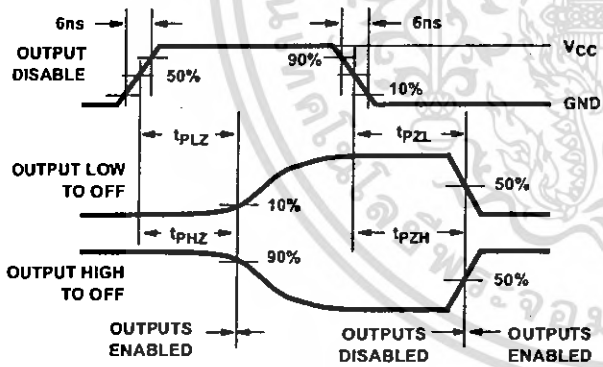


FIGURE 3. HC THREE-STATE PROPAGATION DELAY WAVEFORM

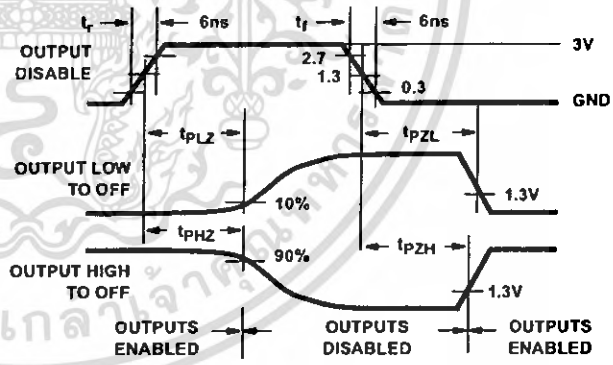
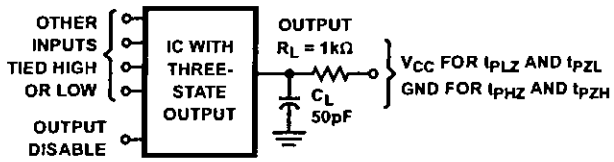


FIGURE 4. HCT THREE-STATE PROPAGATION DELAY WAVEFORM

Test Circuits and Waveforms (Continued)



NOTE: Open drain waveforms t_{PLZ} and t_{PZL} are the same as those for three-state shown on the left. The test circuit is Output R_L = 1kΩ to V_{CC}, C_L = 50pF.

FIGURE 5. HC AND HCT THREE-STATE PROPAGATION DELAY TEST CIRCUIT