ระบบแจ้งเตือนและป้องกันภัย

NOTIFICATION SYSTEM AND PREVENTIVE

โดย

นายนวคุณ จึงเจริญสุขยิ่ง

นายปพน เลิศเศรษฐกานนท์

นายชยกร กนกโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบแจ้งเตือนและป้องกันภัย
Notification System and Preventive

โดย

นาย นวคุณ จึงเจริญสุขยิ่ง

นาย ปพน เลิศเศรษฐกานนท์

นาย ชยกร กนกโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบแจ้งเตือนและป้องกันภัย
NOTIFICATION SYSTEM AND PREVENTIVE

โดย

นาย	นวคุณ	จึงเจริญสุขยิ่ง	53010823
นาย	ปพน	เลิศเศรษฐกานนท์	53010916
นาย	ชยกร	กนกโรจน์	53010953

อาจารย์ที่ปรึกษา

ผศ. สมภพ แก้วมีชัย

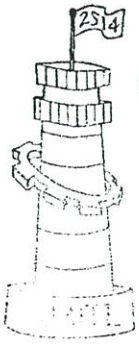
ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

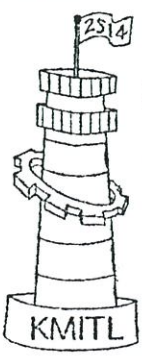


ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

7/พ.ย./57.



ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

14/03/14

ปริญญานิพนธ์ปีการศึกษา 2556
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบแจ้งเตือนและป้องกันภัย
NOTIFICATION SYSTEM AND PREVENTIVE

ผู้จัดทำ
1. นาย นวคุณ จึงเจริญสุขยิ่ง 53010823
2. นาย ปพน เลิศเศรษฐกานนท์ 53010916
3. นาย ชยกร กนกโรจน์ 53010953

..... **อาจารย์ที่ปรึกษา**
(ผศ. สมภพ แก้วมีชัย)

กิตติกรรมประกาศ

ปริญญานิพนธ์จะสำเร็จขึ้นไม่ได้ถ้าปราศจากการสนับสนุนจาก คุณพ่อและคุณแม่ของคณะผู้จัดทำทั้งสามท่าน ที่คอยให้กำลังใจ แนะนำการทำงาน และให้การสนับสนุนในเรื่องทุนทรัพย์สำหรับการศึกษาอีกทั้งเข้าใจถึงปัญหาของการจัดทำปริญญานิพนธ์ในการศึกษาเป็นเวลาต่อเนื่องตลอดมา

ขอขอบคุณคณาจารย์ทุกท่านจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่อบรมสั่งสอน และให้ความรู้แก่คณะผู้จัดทำตลอดมา โดยเฉพาะอย่างยิ่ง ผู้ช่วยศาสตราจารย์สมภพ แก้วมีชัย ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ และ เป็นผู้ช่วยเหลือและให้คำแนะนำทั้งในเรื่องวิชาการ และในด้านประสบการณ์ ตลอดจนข้อคิดในการดำเนินงาน

ขอขอบคุณพี่ๆ ทุกท่านในภาคโทรคมนาคม ที่ให้คำแนะนำ ช่วยเหลือในการทดสอบ และให้คำปรึกษา รวมถึงเทคนิคต่างๆ ที่สามารถนำมาประยุกต์ใช้ในการทำปริญญานิพนธ์

สุดท้ายนี้ขอขอบคุณเพื่อนๆ ร่วมชั้นปีในห้องทุกท่านโดยเฉพาะ คุณพสธร ยิ่งเจริญรัตน์ และ คุณปิยะวัฒน์ ดามพวรรณวงศ์ ที่ให้ความช่วยเหลือในการทดสอบ และช่วยเหลือข้าพเจ้าในเรื่องที่สามารถช่วยได้ ประโยชน์อันใดที่ได้จากปริญญานิพนธ์ฉบับนี้ขอมอบให้แด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

ระบบแจ้งเตือนและป้องกันภัย
NOTIFICATION SYSTEM AND PREVENTIVE

โดย นาย นวคุณ จึงเจริญสุขยิ่ง 53010823
นาย ปพน เลิศเศรษฐกานนท์ 53010916
นาย ชยกร กนกโรจน์ 53010916

อาจารย์ที่ปรึกษา ผศ สมภพ แก้วมีชัย

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการออกแบบระบบแจ้งเตือนและป้องกันโดยการควบคุมอุปกรณ์ผ่านไมโครคอนโทรลเลอร์ ซึ่งอุปกรณ์จะประกอบไปด้วย เซ็นเซอร์อินฟราเรด, สปอร์ตไลท์, ประตูหน้า, ประตูนิรภัย, กำแพง, เสียงแจ้งเตือน, อุปกรณ์เช็คน้ำหนัก โดยผู้ใช้จะสามารถควบคุมอุปกรณ์ และเข้า-ออกห้องนิรภัย จากการเข้ารหัส (Login) โดยจะตรวจสอบจากชื่อผู้ใช้ (Username), รหัสผ่าน (Password) และน้ำหนักของผู้ใช้ (Weight) และสามารถตรวจสอบความพร้อมของอุปกรณ์ซึ่งแสดงผลผ่านหน้าจอ LCD ซึ่งระบบจะทำการตรวจสอบกับระบบฐานข้อมูล (Database) จากนั้นระบบจะสามารถเก็บบันทึกประวัติการใช้งาน (History) ไว้โดยจะมีการทดสอบการใช้งานผ่านแบบจำลอง (Model) ที่สร้างขึ้น

ABSTRACT

This Project is planning on notification and preventive system by control the appliance through microcontroller which appliances include with Infrared sensor, Spotlight, Front-door, Security-door, Wall, Alarm, Load-cell. The user can control appliances and walk in-out to the security room by login with the form of interface program that must checking on username, password, weight of user. In addition, user can check preparedness's working of appliances that the result will be showed by LCD module. Interface program will compare the information of login with database system and keep history of login, checking. And this project will be tested with the simulation model's room.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากปัจจุบันนี้ การติดตามข่าวสารบ้านเมืองทางสังคม จะเห็นได้ว่ามีการเกิดการโจรกรรมเกิดขึ้นกันอย่างมากมาย และหลากหลายรูปแบบไม่เว้นในแต่ละวัน มีความเดือนร้อนอยู่เสมอทั้งในด้านของผู้เสียหายหรือผู้รักษาความปลอดภัยจนไปถึงตำรวจ เช่น การลักทรัพย์, การหลอกลวงรูปแบบต่างๆ, การโจรกรรมสิ่งของและการปล้น-จี้ เป็นต้น ซึ่งก่อให้เกิดความสูญเสียทางด้านบุคคลากร, ทรัพย์สิน และมีผลกระทบตามมาอีกมากมาย รวมถึงแนวทางการป้องกันหรือรักษาความปลอดภัยก็ยังไม่มีมาก และครอบคลุมเท่าที่ควร ซึ่งจะเห็นได้จากการโจรกรรมที่เกิดขึ้นซ้ำไปมา และมีให้พบเห็นกันได้ในทุกๆ วัน ทั้งนี้ผู้จัดทำจึงได้มีแนวคิดที่จะออกแบบระบบป้องกัน และระบบเตือนภัยจากการโจรกรรมขึ้นในรูปแบบต่างๆ เพื่อหวังว่าระบบนี้จะถูกลองทดสอบ หรือนำไปพัฒนาประยุกต์ใช้กับการโจรกรรมที่เกิดขึ้น และสามารถป้องกันหรือลดเหตุความเสียหายต่างๆ ลงได้ ประเภทของการโจรกรรมจะยกตัวอย่างเช่น การโจรกรรมร้านค้า, ร้านทองและ ร้านเพชร เป็นต้น ซึ่งระบบรักษาความปลอดภัยและป้องกันผู้ดูแลร้านเป็นการออกแบบระบบของจำลองห้องเสมือนร้านค้า ซึ่งระบบป้องกันและเตือนภัยนี้จะมาจากการเขียนโปรแกรมโดยใช้ตัวควบคุมคือ ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นหลัก โดยมีระบบป้องกันและเตือนภัยหลายชนิดในปริญญานิพนธ์นี้

1.2 วัตถุประสงค์

1) เพื่อศึกษาความเป็นไปได้ในระบบรักษาความปลอดภัย
2) เพื่อหาแนวทางการป้องกันการโจรกรรม
3) เพื่อศึกการทำงานของไมโครคอนโทรลเลอร์ (Microcontroller) และนำมาประยุกต์ใช้ในชีวิตจริง
4) เพื่อศึกษาภาษาซี (C) และนำไปประยุกต์ใช้ในการเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ (Microcontroller)
5) เพื่อศึกษาการสร้างโปรแกรมอินเตอร์เฟส และฐานข้อมูล รวมถึงวิธีการเชื่อมต่อสัญญาณจากคอมพิวเตอร์ไปสู่อุปกรณ์อื่นๆ
6) เพื่อศึกษาระบบการทำงานของเซ็นเซอร์อินฟราเรด (Infrared Sensor)
7) เพื่อศึกษาระบบการทำงานของ Loadcell

1.3 ขอบเขตของปริญญานิพนธ์

1) เขียนโปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยต่างๆ ประกอบด้วย Spotlight , Alarm , DC Motor (wall - front door - security door) , Load-cell , Infrared Sensor
2) เขียนโปรแกรมอินเตอร์เฟส (Interface) การลงชื่อเข้าใช้ (Login) โดยตรวจสอบจากชื่อผู้ใช้ (Username), รหัสผ่าน (Password) และน้ำหนัก (Weight) ซึ่งจะต้องตรงกับฐานข้อมูลที่ทำการกำหนดไว้ และโปรแกรมอินเตอร์เฟส (Interface) การเข้าเช็คอุปกรณ์
3) สร้างระบบฐานข้อมูล (Database) ของผู้ใช้ (User) และประวัติ (History) การใช้งานการเข้าเช็คอุปกรณ์ และเชื่อมต่อเพื่อใช้กับระบบเข้ารหัส
4) สร้างวงจรเซ็นเซอร์อินฟราเรดทางด้านฝั่งส่งและฝั่งรับ (Transmit and Receiver Infrared Sensor)
5) เขียนโปรแกรมแสดงความพร้อมในการทำงานและตรวจเช็คอุปกรณ์โดยแสดงผลผ่านหน้าจอ LCD
6) สร้างแบบจำลอง (Model) เพื่อจำลองและทดสอบการทำงานของอุปกรณ์

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่บรรจุความสามารถมากมายไม่ว่าจะเป็นหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก (logic) การประมวลผลจะอยู่ภายใต้การเขียนโปรแกรมในการควบคุมหรือออกคำสั่ง นอกจากนี้ในตัวไมโครคอนโทรลเลอร์ยังมีวงจรรับสัญญาณอินพุต วงจรขับสัญญาณออกทางเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาทำให้ไมโครคอนโทรลเลอร์สามารถนำไปใช้งานได้อย่างมากมาย ตามรูปแบบที่เหมาะสมกับงานนั้นๆ

ไมโครคอนโทรลเลอร์มาจาก 2 คำรวมกันคือ ไมโคร (Micro) ซึ่งหมายความถึง ไมโครโปรเซสเซอร์ ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วยหน่วยประมวลผลกลางหรือซีพียู (CPU:Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU:Arithmatic Logic Unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรเชื่อมต่อสัญญาณนาฬิกา และคำว่า คอนโทรลเลอร์ ซึ่งหมายถึง อุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม ภายใต้การควบคุมนั้นจะถูกกำหนดจากการเขียนโปรแกรมควบคุม การเขียนโปรแกรมควบคุมจะสามารถเขียนโปรแกรมได้หลายภาษา เช่น ภาษาซี ภาษาแอสเซมบลี ภาษาอินเตอร์พรีเตอร์ (Interpreter) เป็นต้น

2.2 โครงสร้างของไมโครคอนโทรลเลอร์

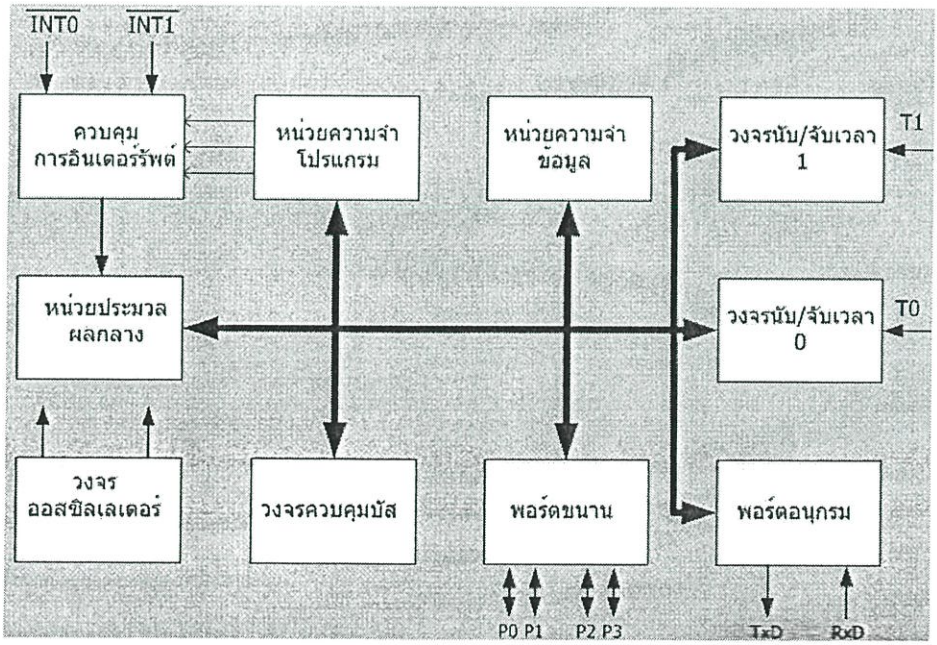
2.2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

1) ต้องการแหล่งจ่ายไฟ +5 ชุดเดียว
2) มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และเบอร์ 8031,8032 จะไม่มีหน่วยความจำโปรแกรม ส่วนเบอร์ 8052 จะมีหน่วยความจำโปรแกรม 8 กิโลไบต์
3) มีหน่วยความจำสำหรับข้อมูล (Data Memory) ขนาด 128 ไบต์ แต่สำหรับเบอร์ 8052 จะมีขนาด 256 ไบต์
4) หน่วยความจำสำหรับโปรแกรมและข้อมูล (Program Memory and Data Memory) แยกจากกันอย่างละ 64 กิโลไบต์ ส่วนเบอร์ 8052 จะแยกจากกันอย่างละ 128 ไบต์

5) คำสั่งที่ใช้เวลาน้อยที่สุดอยู่ที่ประมาณ 1uS เมื่อทำงานที่ความถี่ 12 MHz
6) มี Time/Counter ขนาด 16 บิต 2 ชุด
7) รับอินเตอร์รัปต์ (interrupt) ได้ 6 แหล่ง 5 เวคเตอร์
8) มีพอร์ตสำหรับส่งข้อมูลอนุกรม (UART) 2 พอร์ต ทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
9) มีคำสั่งในการทำ AND,OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิต และ 1 บิต

2.2.2 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์

1) เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
2) ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลช (Flash) สามารถลบและเขียนใหม่ได้หนึ่งพันครั้ง
3) หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
4) ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
5) มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ (Full Duplex)
6) ไทเมอร์/เคาร์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
7) สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
8) สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
9) มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป



รูปที่ 2.1 โครงสร้างพื้นฐานภายในของไมโครคอนโทรลเลอร์ [1]

รูปที่ 2.1 แสดงโครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 8051 ประกอบด้วยอุปกรณ์ต่างๆดังนี้ ส่วนของหน่วยความจำภายในสำหรับเก็บข้อมูลขนาด 128 ไบท์ (Internal Data Memory) ส่วนของหน่วยความจำภายในสำหรับเก็บโปรแกรมที่มีขนาด 4 กิโลไบท์ (Internal Program Memory) อุปกรณ์ควบคุมการอินเตอร์รัพท์ (Interrupt Control Unit) ตัวตั้งเวลาและตัวนับเวลาขนาด 16 บิท 2 ชุด (Timer/Counter 0 and Timer/Counter 1) พอร์ตควบคุมการสื่อสารอนุกรมแบบ Full Duplex ซึ่งสามารถรับส่งข้อมูลพร้อมกันได้ พอร์ตขนาน สำหรับติดต่อกับอุปกรณ์ภายนอกจำนวน 4 พอร์ต พอร์ตละ 8 บิท วงจรผลิตสัญญาณนาฬิกาภายใน

2.3 การจัดขาของไมโครคอนโทรลเลอร์

2.3.1 ขาพอร์ท 0 (P0.0 P0.7)

มี 8 ขา แต่ละขาสามารถกำหนดให้ได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าต้องการให้ขาพอร์ท 0 ขาใดขาหนึ่งเป็นอินพุต เขียนข้อมูล 1 ไปยังขานั้น ๆ ส่งผลให้ขานั้นมีอินพุทอิมพิแดนซ์สูง โดยปกติพอร์ท 0 ใช้ทำหน้าที่ เป็น บัสข้อมูล (DATA BUS D0 D7) และเป็นแอดเดรสบัส (ADDRESS BUS A0-A7)ไบต์ต่ำ โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วยเพื่อสลับการทำงาน คือเป็นได้ทั้งแอดเดรสบัส และดาต้าบัสในพอร์ทเดียวกัน

2.3.2 ขาพอร์ท 1 (P1.0 P1.7)

มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุทและเอ้าท์พุทสำหรับใช้งานทั่วไป ถ้าต้องการให้ขาพอร์ท 1 ขาใด ขาหนึ่งเป็นอินพุท ทำได้ดังนี้ เขียนข้อมูล 1 ไปยังขานั้น ๆ ส่งผลให้ขานั้นมีอินพุทอิมพิแดนซ์สูง

2.3.3 ขาพอร์ท 2 (P2.0 P2.7)

มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุทและเอ้าท์พุทสำหรับใช้งานทั่วไป ถ้าต้องการให้ขาพอร์ท 2 ขาใด ขาหนึ่งเป็นอินพุท ทำได้ดังนี้ เขียนข้อมูล 1 ไปยังขานั้น ๆ ส่งผลให้ขานั้นมีอินพุทอิมพิแดนซ์สูง โดยปกติพอร์ท 2 ใช้ทำหน้าที่เป็นแอดเดรสไบท์สูง 8บิทบน คือ ADDRESS BUS A8 A15 ของหน่วยความจำภายนอก

2.3.4 ขาพอร์ท 3(P3.0- P3.7)

มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุทและเอ้าท์พุทสำหรับใช้งานทั่วไป ถ้าต้องการให้ขาพอร์ท 3 ขาใด ขาหนึ่งเป็นอินพุท ทำได้ดังนี้ เขียนข้อมูล 1 ไปยังขานั้น ๆ ส่งผลให้ขานั้นมีอินพุทอิมพิแดนซ์สูง โดยปกติทำหน้าที่ดังนี้

P3.0 ใช้เป็นขาอินพุทสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุทสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุทรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 0 หรือขา INT0
P3.3 ใช้เป็นขาอินพุทรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 1 หรือขา INT1
P3.4 ใช้เป็นขาอินพุทรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0
P3.5 ใช้เป็นขาอินพุทรับสัญญาณไทเมอร์จากภายนอกช่อง 1 หรือขา T1
P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีติดต่อกับหน่วยความจำภายนอก
P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีติดต่อกับหน่วยความจำภายนอก

2.3.5 ขารีเซต (RESET)

ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซท สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชินไซเคิล

2.3.6 ขา $\overline{\text{ALE}}$/PROG (Address Latch Enable/Program Pulse input)

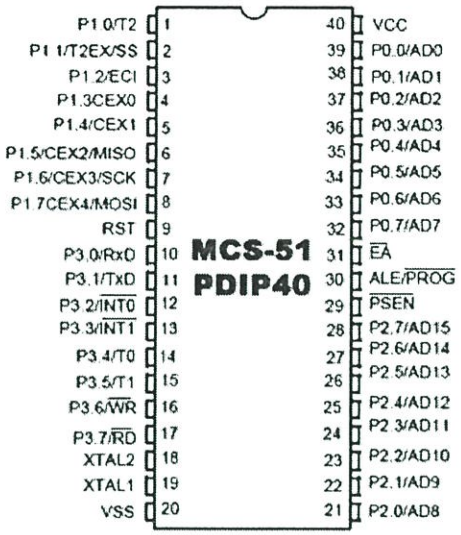
เป็นขาที่ใช้ในการควบคุมการแลตช์ของพอร์ท 0 เมื่อมีการใช้งานหน่วยความจำภายนอก

2.3.7 ขา $\overline{\text{PSEN}}$ (PROGRAM STORE ENABLE)

ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชินไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใด ๆออกมา

2.3.8 ขา $\overline{\text{EA}}$/Vpp (External Access enable /Programming Voltage input)

ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ดังนี้ ถ้าขา $\overline{\text{EA}}$ = 0เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ถ้าขา $\overline{\text{EA}}$ = 1 เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อหน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์ โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วยเพื่อสลับการทำงาน คือเป็นได้ทั้งแอดเดรสบัส และดาต้าบัสในพอร์ทเดียวกัน



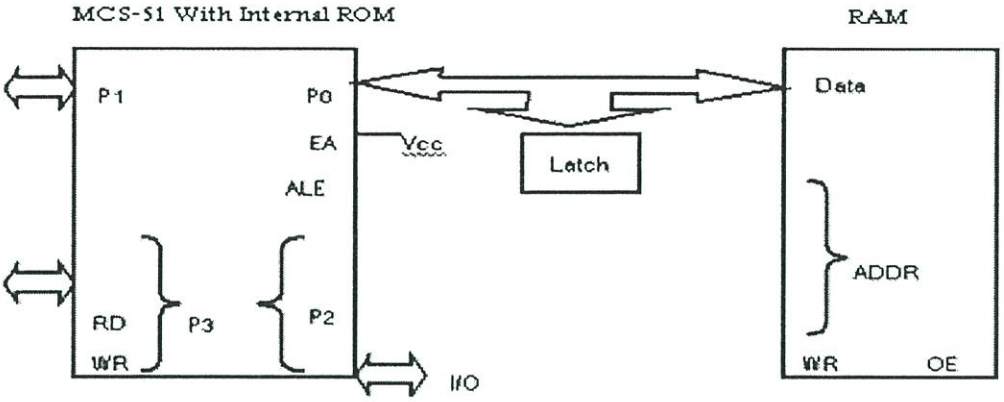
รูป 2.2 ขาของไมโครคอนโทรลเลอร์ MCS-51 [2]

2.4 โครงสร้างหน่วยความจำภายในไมโครคอนโทรลเลอร์

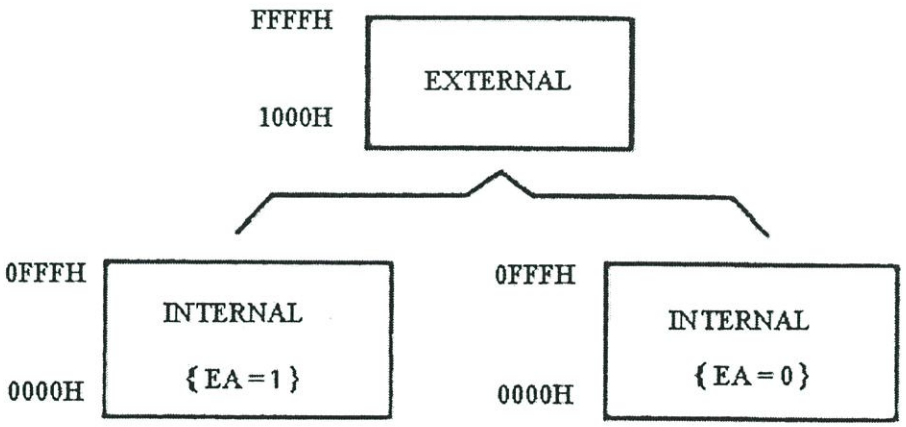
ไมโครคอนโทรลเลอร์ MCS-51 แยกการจัดการหน่วยความจำออกเป็นสองส่วนอย่างชัดเจน คือ หน่วยความจำโปรแกรม (PROGRAM MEMORY) และหน่วยความจำข้อมูล (DATA MEMORY) หน่วยความจำทั้งสองนี้ มีหน้าที่แตกต่างไปจากกัน และใช้วิธีการอ้างแอดเดรสสัญญาณการติดต่อแยกออกจากกัน

2.4.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 เป็นบริเวณหน่วยความจำสำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหายโครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ของหน่วยความจำ ประเภทต่างๆ เช่น หน่วยความจำแบบรอม (READ ONLY MEMORY) หรือ อีพรอม (ERASABLE PROGRAMABLE READ ONLY MEMORY) ในไมโครคอนโทรลเลอร์ MCS-51 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (INTERNAL PROGRAM MEMORY) ซึ่งเป็นหน่วยความจำรอม หรือ อีพรอม ที่อยู่ภายในตัวไอซีของไมโครคอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก (EXTERNAL PROGRAM MEMORY) ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ



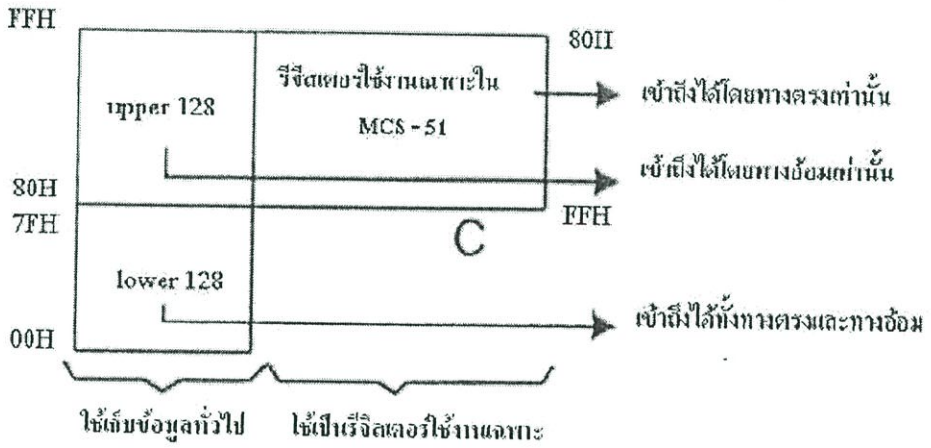
รูปที่ 2.3 การใช้หน่วยความจำสำหรับเก็บโปรแกรม [3]



รูปที่ 2.4 การจัดพื้นที่ของหน่วยความจำโปรแกรมภายในและภายนอก [3]

ไมโครคอนโทรลเลอร์เบอร์ต่างๆ ของตระกูล 8051 นี้สามารถขยายให้ใช้งาน ในหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้นจะต้องทำการควบคุมระดับลอจิกของสัญญาณ ในขณะนั้นด้วย ขนาดหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์เบอร์ต่างๆ ภายในตระกูล 8051 จะแตกต่างกันออกไป เพื่อความเหมาะสมกับการนำไปใช้งานลักษณะต่างๆเช่น

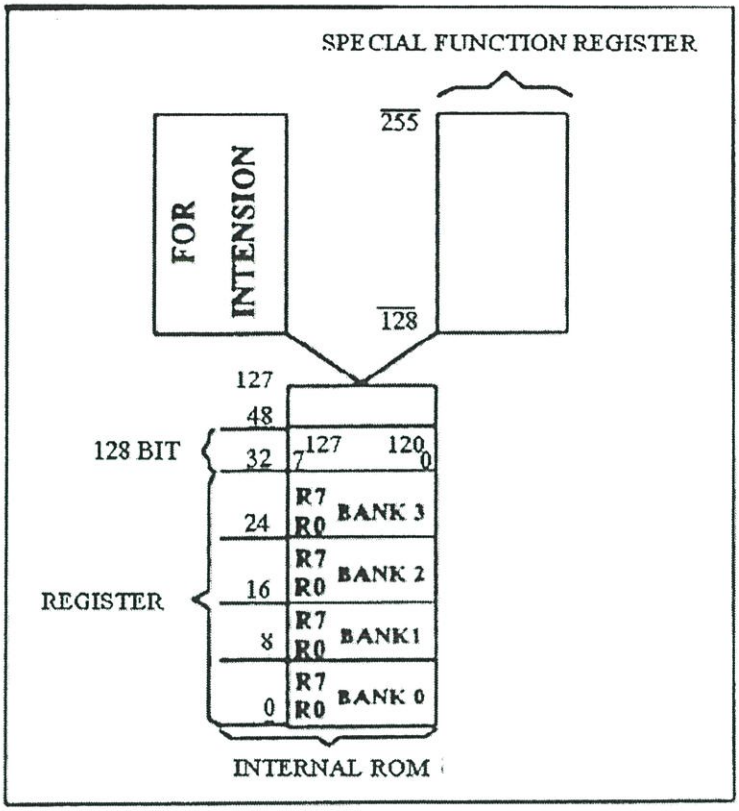
- 8051 และ 8052 มีหน่วยความจำแบบรอม 4 และ 8 กิโลไบต์
- 8751 มีหน่วยความจำแบบ อีพรอม ขนาด 4 กิโลไบต์ ข้อมูลที่จัดเก็บภายในนี้ ซึ่งสามารถใช้แสงอุลตร้าไวโอเลตลบและนำกลับไปบรรจุโปรแกรมใหม่ได้ อีกครั้งหนึ่ง
- 8031 และ 8032 ไม่มีหน่วยความจำโปรแกรมอยู่ภายในตัวไอซี ดังนั้นในการนำไปใช้งานจึงจำเป็นต้องอาศัยหน่วยความจำโปรแกรมภายนอกเสมอ



รูปที่ 2.5 หน่วยความจำสำหรับเก็บข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 [3]

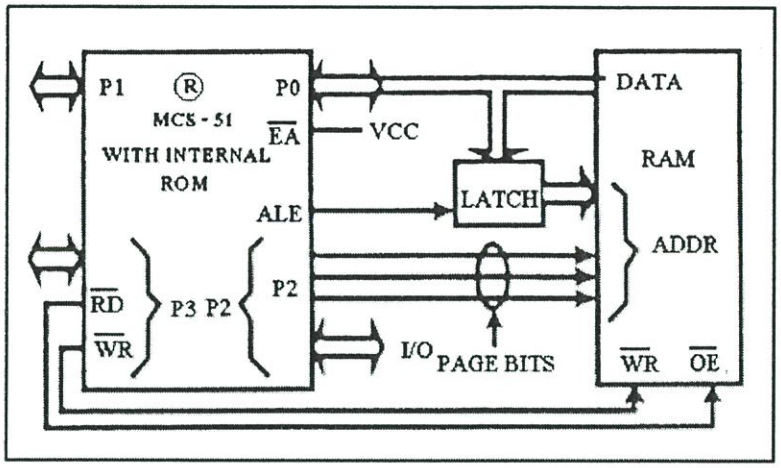
2.4.2 หน่วยความจำข้อมูล (DATA MEMORY)

โดยพื้นฐานแล้วเป็นหน่วยความจำแรมสามารถเขียนหรืออ่านข้อมูลได้ (READ OR WRITE MEMORY) ใช้สำหรับเก็บข้อมูลหรือตัวแปร ที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว ซึ่งโดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำแรมแบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญไป พื้นที่ของหน่วยความจำข้อมูลของไมโคร-คอนโทรลเลอร์ MCS-51 มีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ตามลักษณะของหน่วยความจำโปรแกรมภายในซึ่งก็เป็นแรมที่อยู่ภายในตัวไอซีในตระกูลของไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลภายนอกซึ่งเป็นการใช้ไอซีหน่วยความจำแรมมาเพิ่มเติมเข้าไปในวงจร ลักษณะเดียวกับ การนำไอซีอีพรอมมาใช้งานเป็นหน่วยความจำโปรแกรมนั่นเอง



รูปที่ 2.6 การจัดหน่วยความจำข้อมูล [3]

โดยที่หน่วยความจำสำหรับเก็บข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 นี้สามารถแบ่งออกเป็น 2 ส่วนคือ ในส่วนที่เป็นหน่วยความจำสำหรับเก็บข้อมูลภายในไอซี และหน่วยความจำสำหรับเก็บข้อมูลภายนอกไมโครคอนโทรลเลอร์ MCS-51 ทุกๆ เบอร์จะมีหน่วยความจำเก็บข้อมูลทั่วๆ ไปภายในไอซีอย่างน้อยคือ 128 ไบต์ ไปจนถึง 256 ไบต์ทั้งนี้ ขึ้นกับเบอร์ของไอซี หน่วยความจำสำหรับเก็บ ข้อมูลภายในไอซีในบริเวณ 128 ไบต์เรียกว่า LOWER 128 และในบริเวณ 128 ไบต์หลัง ที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า UPPER 128 แสดงดังรูปที่ 2.7



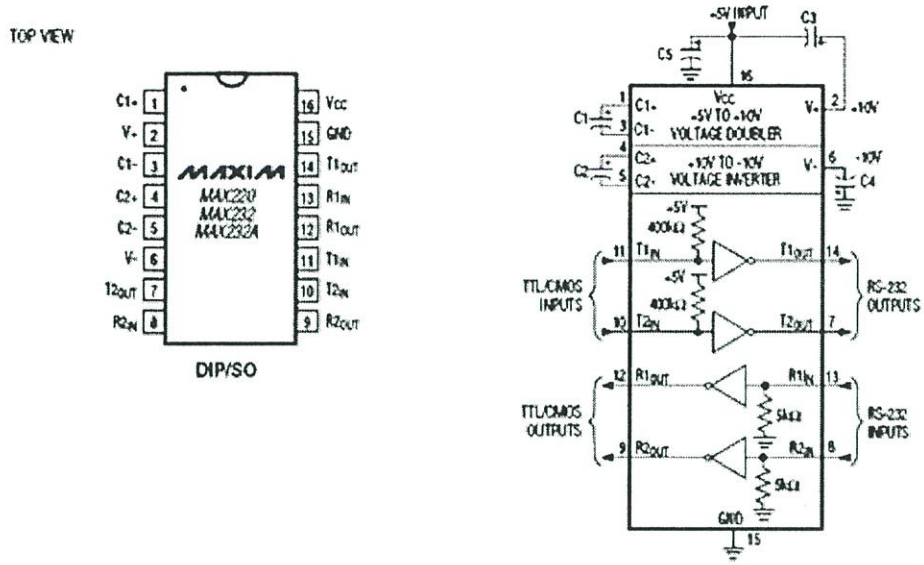
รูปที่ 2.7 การต่อกับหน่วยความจำข้อมูลภายนอกไอซี [3]

2.4.3 การรีเซ็ต

ความหมายของการรีเซตเป็นการบังคับให้มีการเริ่มต้นใหม่อีกครั้งหนึ่ง ซึ่ง มักจะกระทำโดยการกำหนดสภาวะของสัญญาณที่ขารีเซตของไอซี MCS-51 ให้เป็นระดับลอจิก ที่เหมาะสมเท่านั้น การรีเซตด้วยวิธีนี้ถือว่าเป็นการอินเตอร์รัปต์อย่างหนึ่งได้ แต่จะมีลักษณะต่างออกไปจากการอินเตอร์รัปต์ของสัญญาณนี้ได้ ซึ่งมีศัพท์เฉพาะเรียกว่า NON-MASKABLE INTERRUPT นอกจากนี้การดำเนินการของโปรแกรมก็แตกต่างออกไปด้วย โดยจะไม่มีการเก็บค่าของคำสั่งที่กำลังจะไปทำในลำดับต่อไปภายในรีจิสเตอร์ PC เมื่อมีการรีเซตเกิดขึ้นโปรแกรม จะถูกสั่งให้กระโดดไปยังแอดเดรส 0000 ทันที ซึ่งตำแหน่งนี้จะเป็นตำแหน่งเริ่มต้นของการทำงานของไมโคร-คอนโทรลเลอร์ MCS-51 เมื่อเริ่มจ่ายไฟให้กับระบบเมื่อใดก็ตามที่มีการรีเซตเกิดขึ้นค่าสภาวะต่างๆ ภายในไมโครคอนโทรลเลอร์จะถูกกำหนดกลับไปเป็นค่าเริ่มต้นใหม่อีกครั้ง

2.5 การส่งข้อมูลออกพอร์ทอนุกรม (Serial Port)

การส่งข้อมูลแบบอะซิงโครนัส จะเป็นการส่งข้อมูลได้ 2 ทิศทางในเวลาเดียวกัน ซึ่งนำมาใช้กับการติดต่อ RS232 การควบคุมผ่านพอร์ตอนุกรม หรือ ที่มักเรียกว่า RS232 จะใช้การส่งสัญญาณโดยใช้สายสัญญาณเพียง 3 เส้นเท่านั้น และจะใช้ IC MAX232 ซึ่งเป็น IC +5V Powered , Multichannel RS-232 Drivers / Receivers ทำหน้าที่รับส่งข้อมูล โครงสร้างและตำแหน่งขาของ IC MAX232 มีดังรูปที่ 2.8



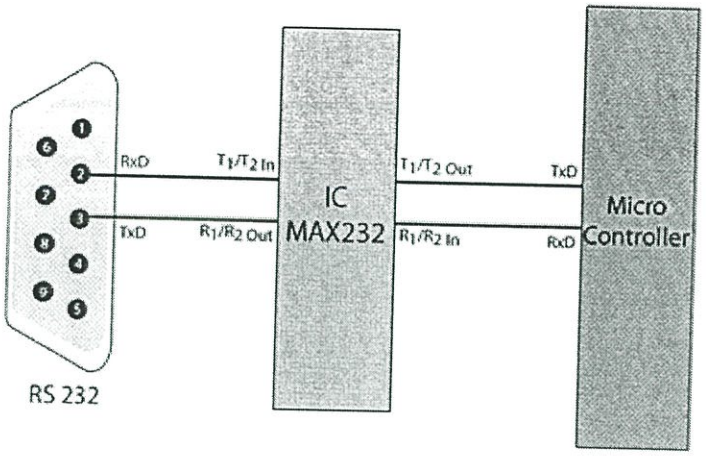
รูปที่ 2.8 IC และ โครงสร้างของ MAX-232 [4]

สำหรับตำแหน่งขา ของ PIC ที่ทำหน้าที่ รับส่งข้อมูล มีใช้งาน ขา TX และ RX และในการใช้งานการส่งข้อมูลนี้ จะมีจะมีเบอร์บางเบอร์เท่านั้น เช่น ขาของ PIC16F677A คือ

ขาพอร์ต RC6 / TX / CK เป็นขาสำหรับส่งข้อมูล - - - >

ขาพอร์ต RC7 / RX / DT เป็นขาสำหรับรับข้อมูล < - - -

การส่งข้อมูลออกพอร์ทอนุกรมจะต้องใช้ไมโครคอนโทรลเลอร์ที่มีพอร์ททีใช้ติดต่อกับพอร์ทอนุกรม นอกจากนี้จะต้องมีตัวแปลงสัญญาณให้เป็นสัญญาณ ทีทีเอล (TTL) ก่อนที่จะส่งเข้าตัวไมโครคอนโทรลเลอร์ หรือแปลงสัญญาณทีทีเอล (TTL) เป็นสัญญาณ RS-232ก่อนที่จะส่งเข้า PC (Personal Computer) โดยผ่าน RS-232 แสดงดังรูปที่ 2.9



รูปที่ 2.9 การส่งข้อมูลออกจากไมโครคอนโทรลเลอร์ผ่าน MAX232 [5]

2.6 รหัสแอสกี (ASCII)

รหัสแอสกีหรือรหัสมาตรฐานของสหรัฐอเมริกาเพื่อการแลกเปลี่ยนสารสนเทศ(ASCII: American Standard Code for Information Interchange) เป็นรหัสอักขระที่ประกอบด้วยอักษรละติน เลขอารบิก เครื่องหมายวรรคตอน และสัญลักษณ์ต่างๆ โดยแต่ละรหัสจะแทนด้วยตัวอักขระหนึ่งตัว เช่น รหัส 65 (เลขฐานสิบ) ใช้แทนอักษรเอ (A) พิมพ์ใหญ่ เป็นต้น

รหัสแอสกีมีใช้ในระบบคอมพิวเตอร์ และเครื่องมือสื่อสารแบบดิจิทัลต่างๆ พัฒนาขึ้นโดยคณะกรรมการ X3 ซึ่งอยู่ภายใต้การดูแลของสมาคมมาตรฐานอเมริกา (American Standards Association) ภายหลังกลายเป็น สถาบันมาตรฐานแห่งชาติอเมริกา (American National Standard Institute : ANSI) ในปี ค.ศ. 1969 โดยเริ่มต้นใช้ครั้งแรกในปี ค.ศ. 1967 ซึ่งมีอักขระทั้งหมด 128 ตัว (7 บิต) โดยจะมี 33 ตัวที่ไม่แสดงผล (Unprintable/Control Character) ซึ่งใช้สำหรับควบคุมการทำงานของคอมพิวเตอร์บางประการ เช่น การขึ้นย่อหน้าใหม่สำหรับการพิมพ์ (CR & LF - Carriage Return and Line Feed) การสิ้นสุดการประมวลผลข้อมูลตัวอักษร (ETX - End of Text) เป็นต้น และ อีก 95 ตัวที่แสดงผลได้ (Printable Character) ดังที่ปรากฏตามผังอักขระ (Character Map) ด้านล่าง

รหัสแอสกีได้รับการปรับปรุงล่าสุดเมื่อ ค.ศ. 1986 ให้มีอักขระทั้งหมด 256 ตัว (8 บิต) และเรียกใหม่ว่าแอสกีแบบขยาย อักขระที่เพิ่มมา 128 ตัวใช้สำหรับแสดงอักขระเพิ่มเติมในภาษาของแต่ละท้องถิ่นที่ใช้ เช่นภาษาเยอรมัน ภาษารัสเซีย ฯลฯ โดยจะมีผังอักขระที่แตกต่างกันไปในแต่ละภาษาซึ่งเรียกว่า โคดเพจ (Codepage) โดยอักขระ 128 ตัวแรกส่วนใหญ่จะยังคงเหมือนกันแทบทุกโคดเพจ มีส่วนน้อยที่เปลี่ยนแค่บางอักขระ

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0010 0000	32	20	(ช่องว่าง)
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

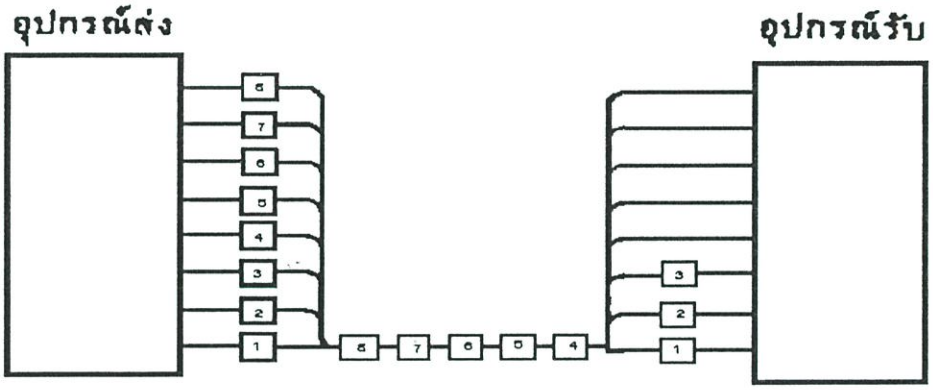
ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

รูปที่ 2.10 ผังอักขระแอสกีที่แสดงผล [6]

2.7 พอร์ทอนุกรม (Serial Port)

การเชื่อมต่อระบบไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกส่วนใหญ่จะใช้การเชื่อมต่อแบบขนาน (Parallel Tranmisson) กับแบบอนุกรม (Serial Transmission) สำหรับการเชื่อมต่อแบบอนุกรมนิยมใช้กันมาก เช่น การเคลื่อนย้ายกันระหว่างไมโครคอมพิวเตอร์ หรือ อุปกรณ์เสริมต่างๆ

ในการถ่ายโอนข้อมูลแบบอนุกรมนั้น ข้อมูลจะได้รับการส่งออกมาครั้งละ 1 บิตระหว่างจุดรับและจุดส่ง จะเห็นว่ามีการส่งข้อมูลแบบอนุกรมนี้จะช้ากว่าการส่งข้อมูลแบบขนาน แต่ยังคงใช้อยู่ก็เพราะ ต้วกลางการสื่อสารต้องการช่องเดียวหรือมีสายเพียงคู่เดียวซึ่งจะประหยัดค่าใช้จ่าย ในการใช้ตัวกลางมากกว่าแบบขนานซึ่งถ้าเป็นระยะทางไกลจะดีเพราะเรามีระบบการสื่อสารทางโทรศัพท์ อยู่แล้ว จึงสามารถนำมาใช้ในการส่งข้อมูลแบบอนุกรมนี้ได้



รูปที่ 2.11 การส่งข้อมูลแบบอนุกรม [7]

การส่งข้อมูลแบบอนุกรม ข้อมูลจะถูกเปลี่ยนให้เป็นแบบอนุกรมเสียก่อนแล้วค่อยทยอยส่งครั้งละ 1 บิต ไปยังที่จะรับ ณ จุดรับจะต้องมีกลไกในการเปลี่ยนข้อมูลที่ส่งมาครั้งละบิตให้เป็นสัญญาณแบบขนาน ซึ่งลงตัวพอดี นั่นคือ บิตที่ 1 ลงที่บัสข้อมูลเส้นที่1พอดีการที่จะทำให้การแปลงสัญญาณจากแบบอนุกรม ครั้งละบิตให้ลงพอดีนั้น จำเป็นต้องมีกลไกที่เหมาะสมเพื่อป้องกันการผิดพลาดจากการรับกลไกที่ว่าแบ่ง ออกเป็น 2 แบบ คือ แบบซิงโครนัส(Synchronous) และแบบอะซิงโครนัส (Asynchronous)

การติดต่อแบบอนุกรมอาจจะแบ่งตามรูปลักษณะการส่งข้อมูลได้ 2 แบบคือ

1) แบบฮาล์ฟเพลกซ์ (Haft Duplex) เป็นการส่งข้อมูลได้ทางเดียวเท่านั้น บางครั้งเรียกว่า การส่งในทิศทางเดียว

2) แบบฟลูดูเพลกซ์ (Full Duplex) ทั้ง 2 สถานีสามารถรับ และส่งได้ในเวลาเดียวกัน

ความเร็วของการถ่ายโอนข้อมูลแบบอนุกรมมีหน่วยวัดเป็น บิตต่อวินาที หรือที่เรียกว่า บีพีเอส (BPS) แต่เรายังมีหน่วยที่นิยมใช้กันมากคือ โบด์เรต หรือ อัตราโบด์ (Baud Rate) ซึ่งหมายถึง การเปลี่ยนแปลงของสัญญาณใน 1 วินาที หลายคนยังเข้าใจสับสนระหว่างหน่วยบีพีเอส กับอัตราโบด์ กล่าวคือ การเปลี่ยนแปลงของสัญญาณของสัญญาณ 1ครั้งอาจจะแสดงถึง การส่งข้อมูลแบบอนุกรมมากกว่า 1 บิต อัตราการส่งข้อมูลเป็นจำนวนบิตจึงเท่ากับ อัตราโบด์คูณกับจำนวนบิตใน 1 โบด์

การสื่อสารแบบอะซิงโครนัส (Asynchronous) ประกอบด้วยบิตเริ่มต้นหรือบิตสตาร์ต (Start Bit) และบิตสิ้นสุดหรือบิตสต็อป (Stop Bit) ขณะที่สถานะของการส่งเป็นแบบว่าง หรือ ไอเดิล (Idle) คือยังไม่มีสัญญาณที่ส่งออกมาแต่จะมีสัญญาณ หรือมีแรงดันตลอดเวลาเพื่อความแน่ใจว่าฝ่ายรับยังติดต่ออยู่กับฝ่ายส่งฝ่ายส่งจะเริ่มส่งข้อมูลบอกจุด เริ่มต้น สัญญาณของอะซิงโครนัสจะเป็น "0" หนึ่งช่วงสัญญาณนาฬิกา บิตนี้เรียกว่าบิตสตาร์ต ข้อมูล 1 ตัวอักษรที่ตามหลังบิตสตาร์ตจะมีขนาดตั้งแต่ 5 บิต จนถึง8 บิต โดยอักขระนี้ส่วนมากจะนิยมใช้ รหัสแอสกี(ASCII CODE)แรกเริ่มที่เดียวของการส่งข้อมูล จะส่งข้อมูลจะส่งรหัสโบดอด (Baudot Code) ซึ่งใช้ 5 บิตในการแทนอักขระ 1 ตัว ส่วนที่ตามหลังข้อมูลก็จะเป็นบิตพาริตี้ ซึ่งจะอาจจะใช้หรือไม่ใช้ก็ได้บิตพาริตี้จะทำหน้าที่เป็นตัวตรวจสอบ ความถูกต้องของสัญญาณที่ได้รับ บิตพาริตี้อาจจะเป็นแบบคู่ (Even) หรือแบบคี่ (Odd) ก็ได้ หมายความว่า ถ้าหากเป็นพาริตี้คู่ จำนวนบิตที่เป็น "1" ในช่วงบิตข้อมูลกับบิตพาริตี้รวมกันแล้วต้องเป็นเลขคู่ผู้ส่งข้อมูล จะทำหน้าที่ตรวจสอบข้อมูลแล้วใส่บิตพาริตี้เอง

ฝ่ายรับ เมื่อรับสัญญาณแล้วก็จะต้องตรวจสอบดูว่าเป็นจริงดังสถานการณ์ที่ตั้งไว้หรือไม่ หากผิดพลาด ก็หมายความ ว่า สัญญาณที่รับนั้นผิดพลาดไปจากสถานีที่ส่งออกมาทั้งนี้ทั้งนั้นจะต้องผิดเป็นจำนวนคี่ เท่านั้นคือ ผิดไป 1 บิต 3 บิต หรือ 5 บิตพร้อมกัน จึงจะตรวจสอบได้ว่าผิดเป็นจำนวนคู่ผลรวมของ จำนวนบิตที่เป็น "1" ก็ยังเป็นคู่อยู่ดี

ทั้งนี้ทั้งนั้นไม่ได้หมายความว่า พาริตี้จะตรวจสอบการผิดพลาดเป็นจำนวนคู่ได้ความจริงแล้ว สามารถตรวจสอบความผิดพลาด ได้เหมือนพาริตี้คู่แต่แทนที่จะตรวจสอบดูว่าสัญญาณ ที่รับเข้ามามีจำนวนคู่ ก็ตรวจสอบดูว่ามีจำนวนคี่หรือเปล่า อย่างไรก็ตาม โอกาสที่จะผิดพลาดเป็น 2,4,6 หรือ 8 บิตพร้อมกันมีน้อยมาก

ย้อนกลับมาดูสัญญาณอะซิงโครนัสใหม่ หลังจากบิตพาริตี้แล้วจะต้องมีบิตสต็อปซึ่งเป็น "1" ความกว้าง ของบิตสต็อปอาจจะเป็น 1,1.5 หรือ 2 พัลส์ของสัญญาณนาฬิกา ซึ่งแล้วแต่ผู้รับและผู้ส่งจะตกลงใช้กันเอง

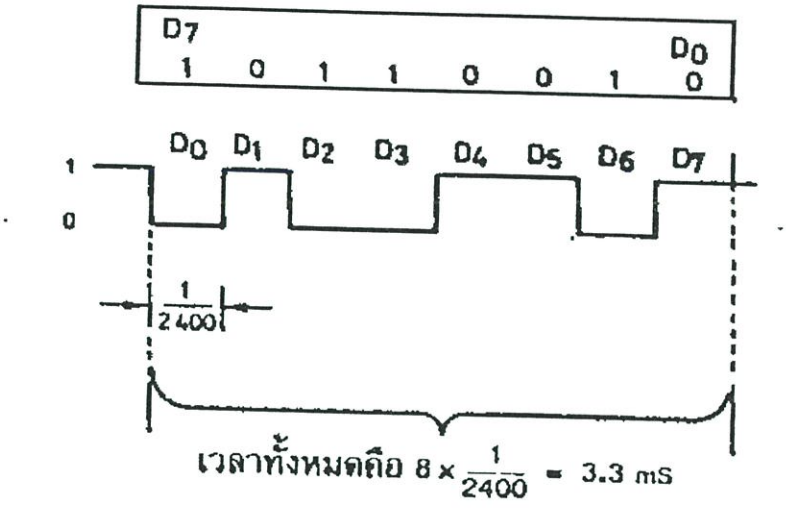
รูปที่ 2.12 การสื่อสารแบบอะซิงโครนัส (Asynchronous) [7]

การสื่อสารแบบซิงโครนัส (Synchronous) ข้อแตกต่างระหว่างวงจรการส่งข้อมูลอนุกรมแบบซึงโครนัสและแบบอะซิงโครนัสก็คือ ความต่อเนื่อง ของข้อมูลที่ส่ง ในแบบซิงโครนัส ข้อมูลที่ส่งออกมาเป็นแบบต่อเนื่อง ไม่มีบิตสตาร์ตหรือบิตสต็อป หรือแม้กระทั่งบิตพาริตี้รูปแบบที่ใช้ในการส่งข้อมูลแบบซิงโครนัสจึงแตกต่างไปจากการส่งข้อมูล แบบอะซิงโครนัส เช่น รูปแบบของบริษัทไอบีเอ็ม ใช้รูปแบบไบซิงก์ (Binary Synchronous Transmission)

การซิงโครไนซ์จะทำในระดับอักขระซึ่งหมายความว่าอักขระแต่ละตัวมีขอบเขตที่แน่นอนแต่ละอักขระ ไม่มีบิตสตาร์ต หรือบิตสต็อปเหมือนอะซิงโครนัส การซิงโครนัสจะกระทำที่จุดเริ่มต้นของการส่งข้อมูล สถานีส่งจะส่งสัญญาณที่เรียกว่า ตัวอักษรนำ (Leading Pad Character)ไปยังสถานีรับก่อนที่จะเริ่มส่งข้อมูล ตัวอักษรนำจะประกอบด้วย "0" และ "1" สลับกัน เพื่อให้สถานีรับจัดสัญญาณนาฬิกาให้ตรงกันก่อนส่ง ข้อมูลก็จะมีการส่งอักขระที่เรียกว่า Syn ตามหลังตัวอักษรนำออกมาสถานีส่งจำเป็นจะบอกความยาว ของข้อมูลมาในกลุ่มนี้ และต้องบอกเครื่องหมายที่เป็นตัวบอกจุดเริ่มต้นของข้อมูลด้วย

ในการส่งข้อมูลแบบอนุกรมมีสิ่งที่สำคัญที่จะต้องพิจารณาคือ ความเร็วของข้อมูลในการส่งซึ่งเราเรียกว่า อัตราบิต (Bit Rate) ตามที่กล่าวมา และกรณีที่ให้อัตราการเปลี่ยนแปลงของสัญญาณ 1 ครั้งต่อข้อมูล 1 บิต จะได้อัตราบิตเท่ากับอัตราโบด์ (Baud Rate)

อัตราโบด์ที่ใช้ในการส่งข้อมูลทั่วไปคือ 110,150,300,1200,2400,4800 และ 9600 สมมติว่า ถ้าต้องการส่งข้อมูลด้วยอัตราโบด์ 2400 โบด์ ข้อมูลจะได้รับการส่งออกไปดังรูปที่ 2.13



รูปที่ 2.13 การส่งข้อมูลด้วยอัตราโบด์ (Baud Rate) ที่ 2400โบด์ [7]

แสดงให้เห็นช่วงเวลาของการส่ง ซึ่งในช่วงเวลาของแต่ละบิตจะมีขนาดช่วงเวลาเท่ากับ 1/2400 เท่ากับ .000416 วินาที หรือ 416 ไมโครวินาที ดังนั้นถ้าต้องการส่งข้อมูลที่มีขนาด 8 บิต ก็จะใช้เวลาทั้งสิ้น 8X416 ไมโครวินาที หรือ เท่ากับ 3,2328 ไมโครวินาที ซึ่งเมื่อเทียบกับการส่งข้อมูลแบบขนาน จะใช้เวลาน้อยกว่า 1 ไมโครวินาที

2.8 รังสีอินฟราเรด (Infrared)

รังสีอินฟราเรด (Infrared) มีชื่อเรียกอีกชื่อว่า รังสีใต้แดง หรือรังสีความร้อน เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่นอยู่ระหว่างคลื่นวิทยุและแสงมีความถี่ในช่วง 1011 - 1014 เฮิร์ตซ์ มีความถี่ในช่วงเดียวกับไมโครเวฟ มีความยาวคลื่นอยู่ระหว่างแสงสีแดงกับคลื่นวิทยุสสารทุกชนิดที่มีอุณหภูมิอยู่ระหว่าง -200 องศาเซลเซียสถึง 4,000 องศาเซลเซียส จะปล่อยรังสีอินฟราเรดออกมา คุณสมบัติเฉพาะตัวของรังสีอินฟราเรด เช่น ไม่เบี่ยงเบนในสนามแม่เหล็กไฟฟ้า ที่แตกต่างกันก็คือคุณสมบัติที่ขึ้นอยู่กับความถี่ คือยิ่งความถี่สูงมากขึ้น พลังงานก็สูงขึ้นด้วย

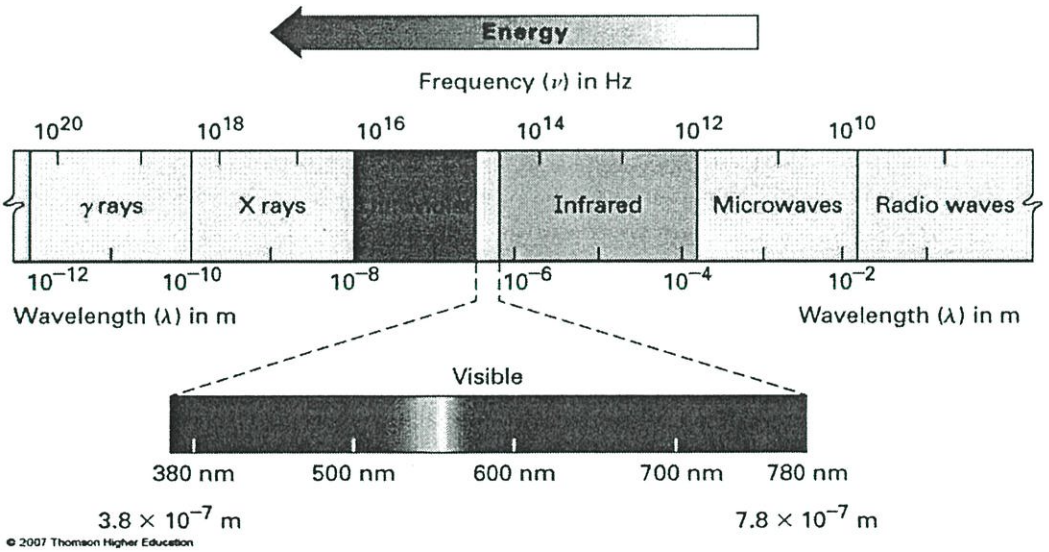
เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความถี่ อยู่ในระหว่างแสงที่ตามองเห็นลำแสงอินฟราเรดเดินทางเป็นเส้นตรง ไม่สามารถผ่านวัตถุทึบแสง และสามารถสะท้อนแสงในวัสดุผิวเรียบได้เหมือนกับแสงทั่วไปใช้มากในการสื่อสารระยะใกล้

2.8.1 คุณสมบัติเด่นของ Infrared

1) คลื่นสั้น การเคลื่อนที่ของคลื่นเป็นเส้นตรง
2) ราคาถูก
3) ปลอดภัยต่อการดักสัญญาณ
4) ไม่สามารถทะลุผ่านวัตถุ

ตารางที่ 2.1 ข้อดีและข้อเสียของอินฟราเรด

ข้อดี	ข้อเสีย
1. สามารถย้ายอุปกรณ์ได้ง่าย 2. ไม่ต้องติดตั้งสัญญาณใหม่	1. ระยะทางในการส่งข้อมูลสั้น 2. ต้องไม่มีสิ่งใดมากีดขวางเส้นสายตาของทั้งเครื่องรับและเครื่องส่ง



รูปที่ 2.14 ช่วงความยาวของคลื่นต่างๆ [8]

2.9 หน้าจอแสดงผล LCD

LCD ย่อมาจากคำว่า Liquid Crystal Display หลักการทำงานจะอาศัยของเหลวพิเศษที่มีคุณสมบัติการบิดแกนโพราไรซ์ของแสง ถ้าหากมีการจ่ายแรงดันไฟฟ้าเข้าไประหว่างสารเหลวนี้ โมเลกุลของมันจะบิดตัวและทำให้แสงไม่สามารถผ่านกระจกออกมาได้ แต่ถ้าไม่มีการจ่ายแรงดันไฟฟ้าแสงจะทะลุผ่านออกมาได้

การทำงานจะเกิดจากกระจกโพลาไรซ์ 2 แผ่น ที่มีแกนตั้งฉากกัน ดังนั้นถ้าไม่ทำอะไรเลยแสงจะไม่สามารถลอดผ่านออกมาได้ เหมือนเอาแว่นตาโพลาไรซ์สองอันมาบิดทำมุมตั้งฉากกัน แสงจะไม่ลอดผ่าน แต่ของเหลวชนิดนี้จะสามารถบิดแกนโพลาไรซ์ของแสงได้ จึงเกิดแสดงสว่างขึ้น

หากไม่มีการจ่ายแรงดันเข้าไป สารเหลวที่ว่านี้จะบิดแกนโพลาไรซ์ของแสงไป 90 องศา ทำให้แสงสามารถลอดออกมาได้จากกระจกโพลาไรซ์คู่นี้ได้ ในทางกลับกัน ถ้ามีแรงดันจ่ายไประหว่างสารเหลวนี้ จะไม่เกิดการปิดตัวของแสง ทำให้แสง "ไม่สามารถ" ลอดออกมาได้

2.9.1 ส่วนประกอบของจอ LCD

หน้าจอ LCD จะประกอบด้วย 3 ส่วนหลักๆ ได้แก่

2.9.1.1 ตัวแสดงผล (Display)

ภายในจะเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นจากภายนอก ดังนั้นจึงจะจ้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

2.9.1.2 ตัวควบคุม (Controller)

เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกเข้ามาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ , แสดงตัวอักษร , เลื่อนเคอร์เซอร์ (Cursor) เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ โดยตัวที่นิยมใช้คือ HD44780 ซึ่งจะควบคุม LCD แบบอักขระ, HD61380 จะควบคุม LCD แบบกราฟฟิก เป็นต้น

2.9.1.3 ตัวขับ (Driver)

เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้แสดงผลข้อมูลตามที่กำหนด โดยชิปที่ใช้ทำหน้าที่นี้ได้แก่ HD44100H , MSM5259 เป็นต้น

2.9.2 โครงสร้างภายในของตัวควบคุม (Controller)

2.9.2.1 Display Data RAM: DDRAM

เป็นแรม (RAM) เก็บข้อมูลแสดงผลโดยจะนำข้อมูลที่ได้มาจากหน่วยความจำรอมเก็บตัวอักษร (CGROM) หรือ หน่วยความจำแรมเก็บตัวอักษร (CGRAM) มาแสดงผลที่จอLCD

2.9.2.2 Charater Generator ROM: CGROM

รอมเก็บตัวอักษรเป็นหน่วยความจำรอมที่เก็บตัวอักษรหรือสัญลักษณ์ต่าง ๆไว้ถาวร ซึ่งจะเก็บในลักษณะ ASCII (ดูตารางได้จากคู่มือ LCD)

2.9.2.3 Charater Generator RAM: CGRAM

เป็นแรมเก็บตัวอักษรเป็นหน่วยความจำแรมที่เก็บตัวอักษรที่เราสร้างขึ้นเพิ่มเติมขึ้นมาใหม่ในกรณีที่ตัวอักษรหรือสัญลักษณ์ต่าง ๆ ไม่มีใน CGROM และต้องการสร้างขึ้นมาใช้เอง

2.9.3 ขาของ LCD MODULE

ขา 1	VSS	เป็นขา GND
ขา 2	VDD	เป็นขาไฟเลี้ยง +5volt DC
ขา 3	VO หรือ VEE	เป็นขาปรับค่าความสว่างที่หน้าจอ LCD
ขา 4	RS	เป็นขาควบคุมเลือกโหมดว่าเป็นคำสั่งหรือข้อมูล
	- RS =0	แสดงว่าข้อมูลที่ส่งเข้ามาที่ขาData (D0-D7) เป็นคำสั่ง
	- RS =1	แสดงว่าข้อมูลที่ส่งเข้ามาที่ขาData (D0-D7) เป็นข้อมูลที่จะแสดงผล
ขา 5	R/W	เป็นขาที่บอกกับ LCD ว่าต้องการอ่านหรือเขียน
	- R/W = 0	แสดงว่าเป็นการเขียนข้อมูลลงในจอแสดงผล LCD
	- R/W = 1	แสดงว่าเป็นการอ่านข้อมูลจากจอแสดงผล LCD

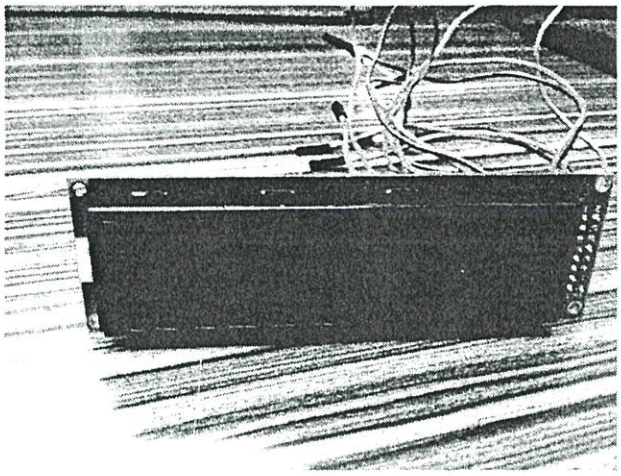
ซึ่งในการใช้งานโดยทั่วไปจะมีการใช้งานเฉพาะการเขียนข้อมูลลงในจอ แสดงผล LCD (R/W = 0) เราจึงไม่จำเป็นที่จะเอาขา R/W เชื่อมต่อกับขาของไมโครคอนโทรลเลอร์ก็ได้โดยต่อขา R/W ลง GNDทำให้ประหยัดขาพอร์ตของไมโครคอนโทรลเลอร์

ขา 6	ENA	เป็นขากำหนดการทำงานของจอแสดงผล LCD
ขา 7-14		เป็นขา Data (D0-D7) สำหรับอ่านหรือเขียนข้อมูลระหว่างจอแสดงผล LCD กับไมโครคอนโทรลเลอร์

2.9.4 ประเภทของ LCD

2.9.4.1 LCD แบบอักขระ (Character LCD Module)

สามารถแสดงตัวอักษรตัวเลข และเครื่องหมายต่างๆ ได้ โดยสร้างจากจุดเล็ก ๆ ที่เราเรียกว่า ดอตเมตริกซ์ (Dot Matrix) โดยทั่วไปจะมีอยู่ 2ขนาดคือ 5x7 จุด และขนาด 5x10 การใช้งาน เช่นโทรศัพท์มือถือ เครื่องคิดเลข เป็นต้น ซึ่งภายในโครงงานนี้จะใช้โมดูลจอแสดงผลประเภทนี้ แสดงดังรูปที่ 2.15



รูปที่ 2.15 จอแสดงผล LCD ที่ใช้ในการทำโครงงาน

2.9.4.2 LCD แบบกราฟิก(Graphic LCD Module)

มีโครงสร้างคล้ายกับLCD แบบอักขระ สามารถแสดงตัวอักษร ตัวเลข และเครื่องหมายต่างๆ ได้ แต่สิ่งที่แตกต่างคือสามารถสร้างรูปภาพได้มีความละเอียดดอตเมตริกซ์ (Dot - Matrix) มากกว่า ในปัจจุบันได้มีการพัฒนาเป็นแบบสีแล้ว การใช้งานเช่นโทรศัพท์มือถือ จอคอมพิวเตอร์ NOTE BOOK จอโทรทัศน์ เป็นต้น

2.9.4.3 LCD แบบเซกเม้นต์ (Segment LCD Module)

เป็นจอ LCD ที่มีขนาดเล็กกว่าแบบอื่น ๆ สร้างขึ้นมามีจุดประสงค์แสดงผลเฉพาะตัวเลขเท่านั้นเหมือน 7-Segment รูปแบบการใช้งานที่เห็นกันง่ายๆ ก็เครื่องคิดเลข ดิจิตอลมิเตอร์ เป็นต้น

2.10 Load-Cell

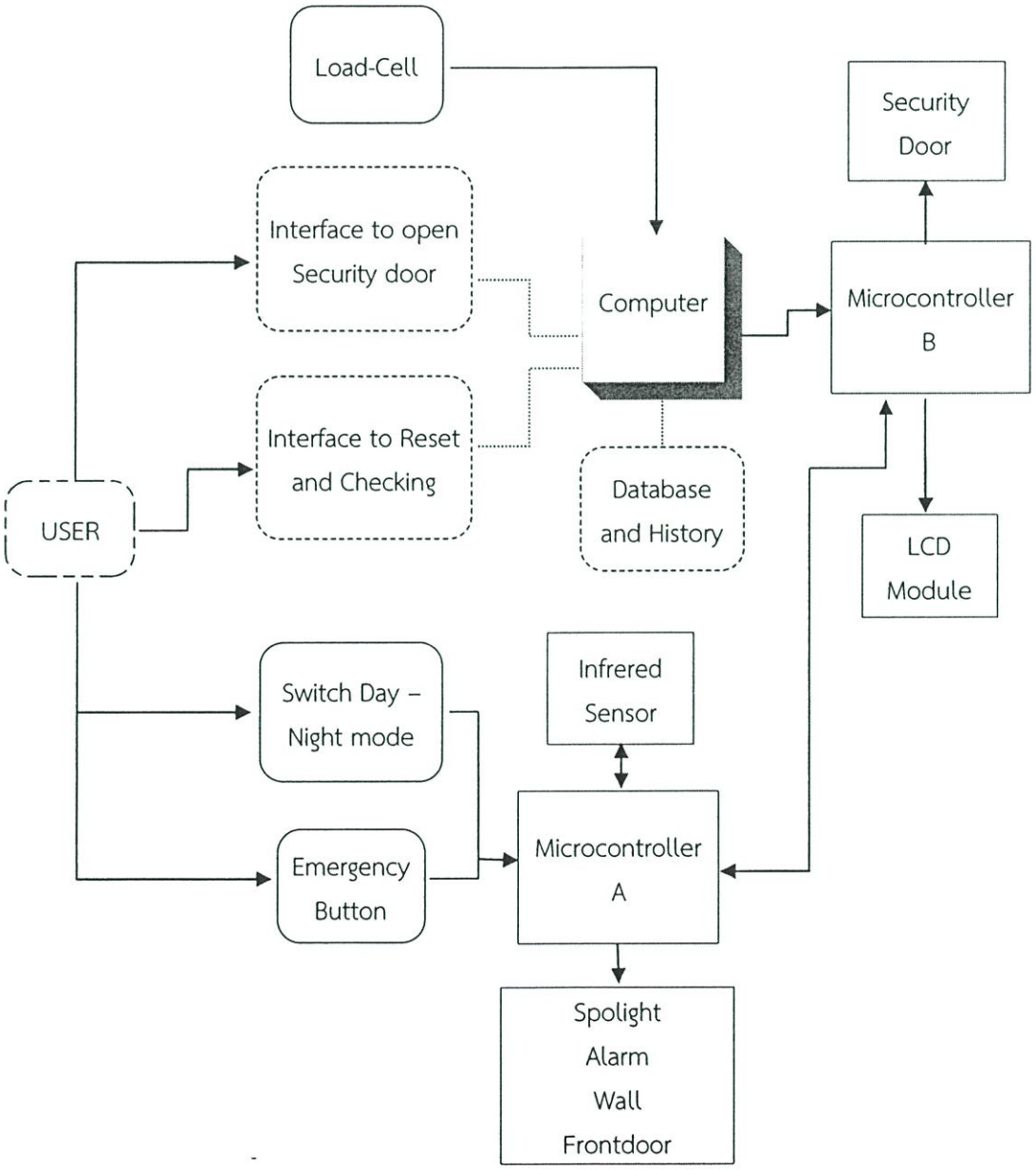
โหลดเซล (Load-cell) คืออุปกรณ์อิเล็กทรอนิกส์ที่ใช้แปลงค่าของแรงไปเป็นสัญญาณไฟฟ้า (Transducer) การแปลงค่านี้ไม่ใช่การแปลงค่าโดยตรงหากแต่เกิดขึ้นสองขั้นตอน จากการแปลงค่าทางกลศาสตร์ แรงจะถูกตรวจจับได้จากการเปลี่ยนรูปร่างของสเตนเกจ (Stain Gate) และสเตนเกจแปลงค่าการเปลี่ยนรูปร่าง (ความเครียด) นี้ไปเป็นสัญญาณไฟฟ้า โหลดเซลมักจะประกอบไปด้วยสเตนเกจสี่ตัวซึ่งจัดเรียงวงจรในรูปแบบของวงจรวิจสโตน บริดจ์ (Wheatstone Bridge Circuit) แต่โหลดเซลที่ประกอบด้วยสเตนเกจเพียงหนึ่งหรือสองตัวก็มีใช้เช่นกัน สัญญาณไฟที่จ่ายออกไปนี้มักจะมีขนาดเพียงไม่กี่มิลลิโวลต์ (Millivolt) และต้องการการขยายสัญญาณด้วยการใช้อุปกรณ์ขยายสัญญาณก่อนที่จะถูกนำไปใช้งานได้

บทที่ 3

การออกแบบและการจัดทำโครงงาน

3.1 การออกแบบและจัดทำ

ภายในบทนี้ จะกล่าวถึงการเขียนและออกแบบโปรแกรมที่ใช้ควบคุมการทำงาน รวมถึงโปรแกรมและวงจรที่ใช้ในการทดสอบโปรแกรมที่ควบคุมทำงาน และวงจรกับอุปกณ์ป้องกันและแจ้งเตือนภัย โดยจะสามารถแสดงบล็อกไดอะแกรม (Block Diagram) ของระบบได้ดังนี้



รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

จากบล็อกไดอะแกรมดังรูป ผู้ใช้สามารถเข้าใช้ระบบได้จาก 3 ทาง ได้แก่

1) โปรแกรมอินเตอร์เฟสที่ใช้เปิดปิดประตูนิรภัย

ในส่วนนี้ผู้ใช้จะต้องทำการเข้าระบบ (Login) โดยกรอกชื่อผู้ใช้ (Username) และรหัสผ่าน (Password) โดยระบบจะมีการตรวจสอบน้ำหนักของผู้ใช้ด้วย ซึ่งข้อมูลดังกล่าวจะต้องตรงกับฐานข้อมูลที่สร้างไว้ก่อนหน้านี้ หากถูกต้อง จึงจะสามารถผ่านประตูได้ แต่ผิดไปจากฐานข้อมูลเกินกว่า 4 ครั้ง ระบบจะมีการแจ้งเตือนเกิดขึ้น

2) โปรแกรมอินเตอร์เฟสที่ใช้เข้าสู่ระบบการตรวจสอบความพร้อมของอุปกรณ์และการปิดระบบระหว่างอุปกรณ์เกิดการทำงาน

ผู้ใช้จะสามารถเข้าสู่ระบบจากการ login และจะมีหน้าต่างอินเตอร์เฟสให้ทดสอบอุปกรณ์ป้องกันและแจ้งเตือนภัยต่างๆ รวมถึงการปิดอุปกรณ์ในกรณีที่อุปกรณ์ทำงานเช่น เสียงกริ่งดัง และกำแพงยกกั้น เป็นต้น

3) สวิทช์ปรับโหมดการใช้งาน (Day-Night Mode) และปุ่มกดในกรณีฉุกเฉิน

ระบบจะมีการทำงานได้ใน 2 โหมดได้แก่ ตอนกลางวันและกลางคืน โดยในช่วงเวลากลางคืนระบบจะมีการเปิดเซ็นเซอร์อินฟาเรด (Infrared Sensor) ส่วนในช่วงกลางวันจะสามารถกดปุ่มในกรณีที่มีเหตุฉุกเฉินเกิดขึ้น ซึ่งอุปกรณ์ป้องกันและแจ้งเตือนภัยจะมีการทำงานตามลำดับต่างๆกันไป

ระบบดังกล่าวจะมีการออกแบบโปรแกรมและอุปกรณ์ต่างโดยจะสามารถแบ่งออกเป็น 5 ส่วนหลักๆ ได้แก่

1) โปรแกรมอินเตอร์เฟส (Interface) สำหรับผู้ใช้ (User) เพื่อใช้ในการเข้ารหัส (Login) เปิดปิดห้องนิรภัย และการเช็คความพร้อมของโปรแกรมควบคุมอุปกรณ์รวมถึงการปิดระบบในระหว่างการทำงานโดยจะเก็บประวัติ (Histoy) การเช็คและตรวจสอบเทียบกับฐานข้อมูล (Database)

2) เซ็นเซอร์อินฟาเรด (Infrared sensor)

3) โปรแกรมและอุปกรณ์การรับค่าน้ำหนัก (Load-cell)

4) โปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยในไมโครคอนโทรลเลอร์ตัวที่ 1

5) โปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยในไมโครคอนโทรเลอร์ตัวที่ 2

3.1.1 โปรแกรมอินเตอร์เฟส (Interface) สำหรับผู้ใช้ (User)

เพื่อใช้ในการเข้ารหัส (Login) เปิดปิดห้องนิรภัย และการเช็คความพร้อมของโปรแกรมควบคุมอุปกรณ์รวมถึงการปิดระบบในระหว่างการทำงานโดยจะเก็บประวัติ (Histoy) การเช็คและตรวจสอบเทียบกับฐานข้อมูล (Database) จะเป็นการสร้างโปรแกรมอินเตอร์เฟส ซึ่งใช้สำหรับการลงชื่อเข้าใช้ของผู้ใช้ โดยการลงชื่อเข้าใช้จะประกอบไปด้วย ชื่อผู้ใช้ (Username) , รหัสผ่าน (Password) และน้ำหนัก (Weight) ซึ่งจะมีเชื่อมต่อกับฐานข้อมูล สามารถควบคุมและเปลี่ยนแปลงจากผู้ดูแลระบบ (Administer) เพื่อตรวจสอบความถูกต้อง และจะเป็นไปตามกลไกของระบบ

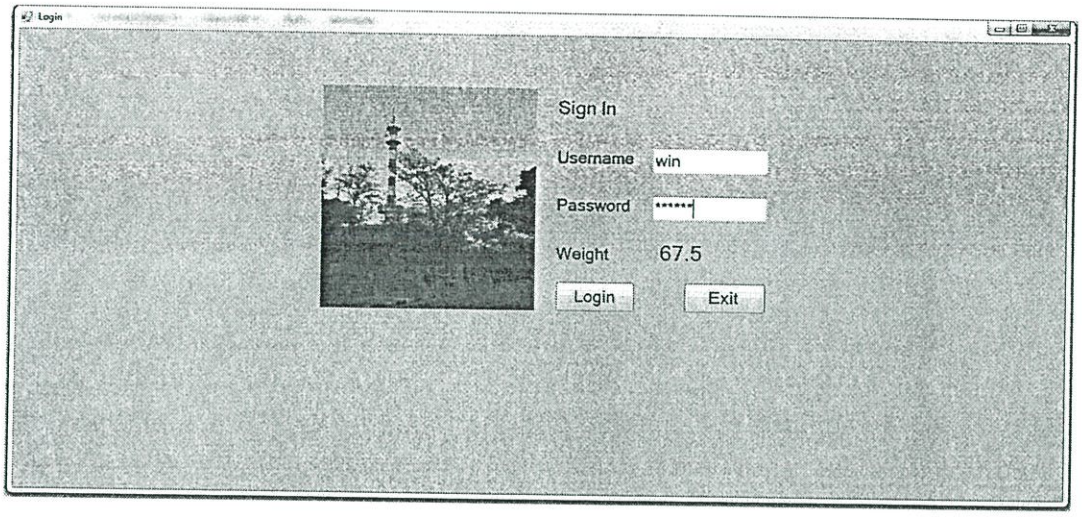
3.1.1.1 การเขียนโปรแกรมอินเตอร์เฟส (Interface) สำหรับผู้ใช้ (User) เพื่อใช้ในการเข้ารหัสเปิดประตู

ในส่วนนี้จะแบ่งเป็น 2 ส่วนด้วยกันแบ่งเป็น

1) ผู้ใช้บริการ (User)

2) ฐานข้อมูล (Database)

จะมีการติดต่อ (Interface) ระหว่าง 2 ส่วนโดยผู้ใช้บริการจะทำการพิมพ์ชื่อผู้ใช้ (Username) ,รหัสประจำผู้ใช้ (Password) และน้ำหนักของผู้ใช้ (Weight) ซึ่งระบบจะทำการอ่านข้อมูลในฐานข้อมูลตรวจสอบความถูกต้องของชื่อผู้ใช้ และรหัสผู้ใช้ และน้ำหนักของผู้ใช้ เมื่อถูกต้องจะทำการส่งข้อมูลเป็นรหัสแอสกี้ (ASCII) อักษรตัววาย (Y) และจะทำการดึงข้อมูลของผู้ใช้ออกมาแสดงในอีกหน้าต่างหนึ่ง ส่วนในกรณีที่ชื่อผู้ใช้ ,รหัสผู้ใช้และน้ำหนักของผู้ใช้ไม่ถูกต้องจะทำการส่งข้อมูลเป็นรหัสแอสกี้อักษรตัวเอ็น (N) โดยข้อมูลที่ส่งจะส่งออกเป็นสัญญาณข้อมูลรหัสแอสกี ข้อมูลจะถูกส่งผ่านพอร์ทอนุกรม RS-232ตามพอร์ทที่กำหนดไว้ในโค้ด ด้วยอัตราการส่งข้อมูล (Baud rate) เท่ากับ 9,600 ซึ่งในส่วนนี้จะใช้โปรแกรม Microsoft Visual Studio 2010 ภาษาซีชาร์ป (C#) ในการเขียนโปรแกรมเพื่อใช้เป็นระบบรักษาความปลอดภัย



รูปที่ 3.2 หน้าต่างแสดงรูปแบบในการระบุชื่อผู้ใช้ (Username) ,รหัสผู้ใช้ (Password)และน้ำหนักของผู้ใช้ (Weight)

เมื่อชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password) และน้ำหนักของผู้ใช้บริการ (Weight) ถูกต้องโปรแกรมจะทำการส่งข้อมูลเป็นตัวอักษรภาษาอังกฤษ วาย (Y) ด้วยสัญญาณข้อมูลรหัสแอสกี (ASCII) ดังนั้น รูปแบบในการเขียนโปรแกรมเพื่อที่จะทำการส่งข้อมูลออกพอร์ทอนุกรม (Serial Port) เป็นดังรูปที่ 3.3

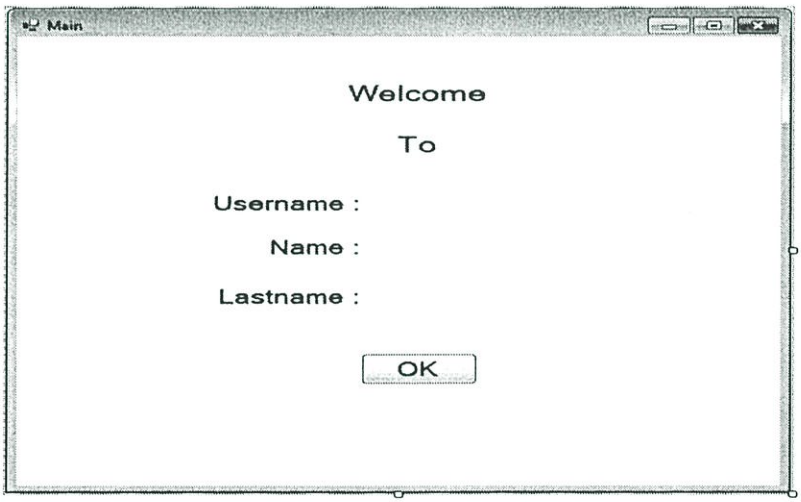
```
Next = "Y";
sp.Write(Next);
```

รูปที่ 3.3 การส่งสัญญาณข้อมูลเป็นรหัสแอสกี้ (ASCII)

เมื่อทำการกดปุ่ม “Login” แล้วระบบทำการตรวจสอบข้อมูลว่าตรงกับฐานข้อมูล (Database) ระบบจะทำการแสดงวันและเวลาเป็นกล่องข้อความ (Messegebox) ดังรูป 3.4

รูปที่ 3.4 กล่องข้อความ (Messagebox) วันและเวลา (DateTime)

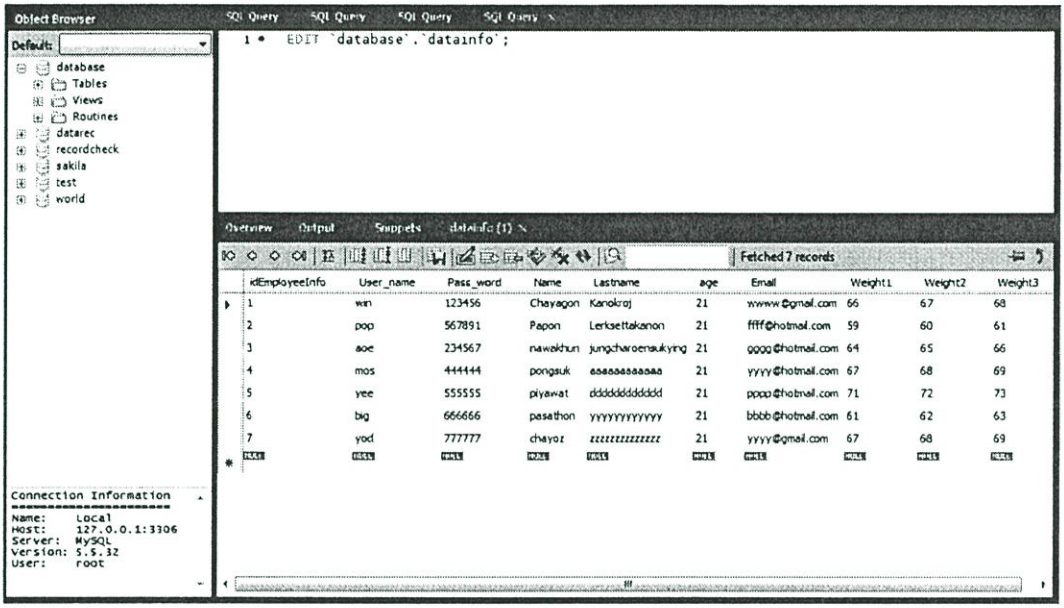
เมื่อทำการส่งข้อมูลเป็นรหัสแอสกี (ASCII) แล้วนั้นโปรแกรมจะทำการเปิดอีกหนึ่งหน้าต่างซึ่งเป็นหน้าต่างที่ไว้แสดงข้อมูลของผู้ใช้นั่นๆ จากฐานข้อมูล (Database)



รูปที่ 3.5 หน้าต่างต้อนรับผู้ใช้เมื่อลงชื่อเข้าใช้ถูกต้อง

ในส่วนของฐานข้อมูล (Database) จะใช้โปรแกรม MySQL Workbench 5.2 CE ซึ่งจะต้องทำการสร้างเซิร์ฟเวอร์ (Server) กำหนดการเชื่อมต่อ (Connection) ของฐานข้อมูล (Database) และทำการสร้างตารางข้อมูล (Table) เพื่อทำเป็นฐานข้อมูล (Database) ในการใช้งาน โดยการสร้างตาราง (Table) ในที่นี้จะกำหนดเป็นส่วนๆ ดังนี้ idEm loyeeInfo , User name , Pass word , Name , Lastname , Age , Email , Weight1 , Weight2 , Weight3 โดยแต่ละส่วนจะเป็นข้อมูลของผู้ใช้แต่ละคนซึ่งน้ำหนักของผู้ใช้ (Weight) จะมีข้อมูลในฐานข้อมูล 3ค่าโดยที่น้ำหนักของผู้ใช้1 (Weight1) และน้ำหนักของผู้ใช้3 (Weight3) จะเป็นค่าน้ำหนักของผู้ใช้ที่เป็นค่าที่สามารถผิดพลาดได้เป็นขอบเขตต่ำสุด (Min) และ

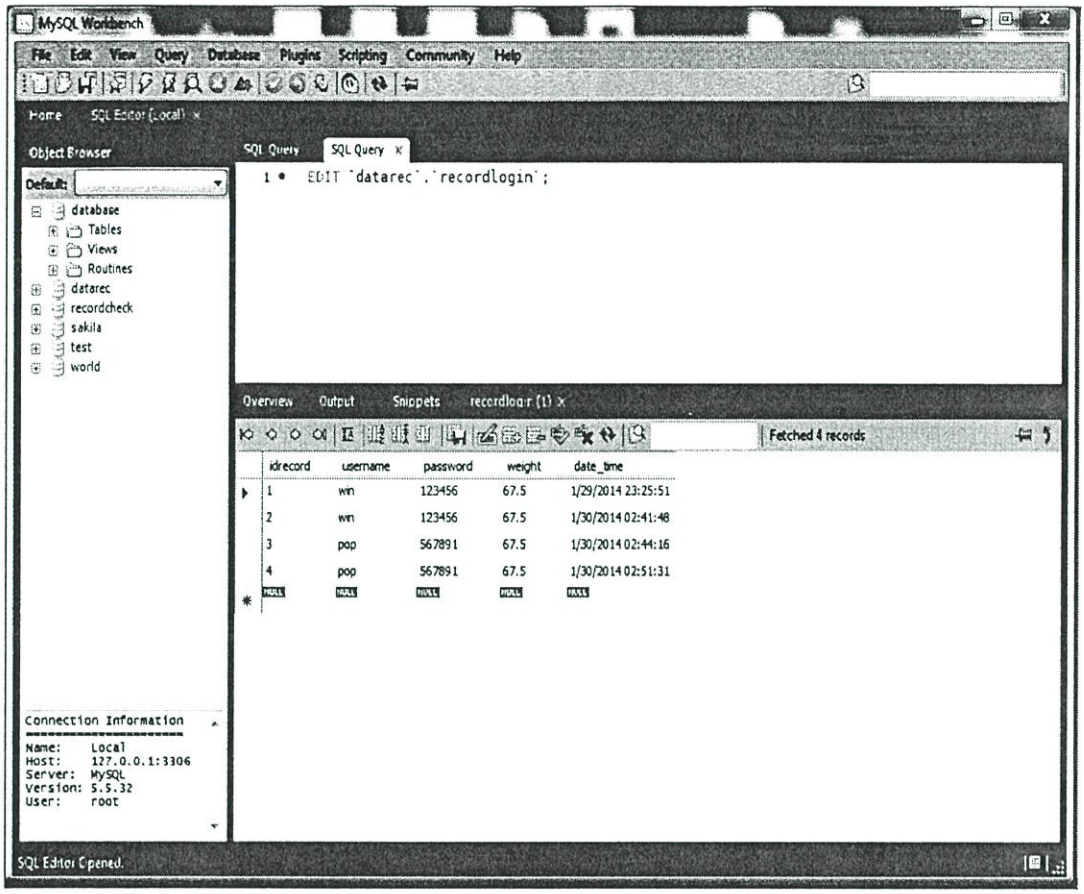
สูงสุด (Max) ซึ่งจะบวกลบจากน้ำหนักของผู้ใช้จริงหนึ่งกิโลกรัม (1Kg) โดยน้ำหนักของผู้ใช้จริงจะอยู่ในฐานข้อมูลในช่อง Weight2 ซึ่งฐานข้อมูล (Database) สามารถรองรับได้ประมาณ 1,000 รายชื่อผู้ใช้ (Username) โดยจะแสดงตัวอย่างฐานข้อมูลในรูปที่ 3.6



รูปที่ 3.6 ตารางฐานข้อมูล(Database)

3.1.1.2 สร้างตารางบันทึกการใช้ประตูชื่อผู้ใช้ (Username),รหัสผู้ใช้ (Password)และน้ำหนักของผู้ใช้ (Weight)

เป็นการสร้างตาราง (Table) ฐานข้อมูลเพื่อดึงข้อมูล ชื่อผู้ใช้ (Username) ,รหัสประจำตัวผู้ใช้ (Password) ,น้ำหนัก (Weight) และวันเวลา (DateTime) เมื่อมีการใช้งานระบบรักษาความปลอดภัยของประตูที่ต้องใส่รหัสผ่าน มาใส่ในตาราง (Table) recordlogin ของฐานข้อมูล datarec โดยฐานข้อมูลจะใช้โปรแกรม MySQL Workbench 5.2 CE เป็นโปรแกรมในการสร้างฐานข้อมูลดังรูปที่ 3.7



รูปที่ 3.7 ฐานข้อมูล (Database) ที่ใช้บันทึกข้อมูลการใช้ระบบประตูรหัส (Password)

เมื่อทำการเก็บข้อมูลจากฐานข้อมูล (Database) ของระบบประตูรหัส (Password) โดยโปรแกรม MySQL Workbench CE 5.2 ซึ่งโปรแกรมนี้สามารถดึงข้อมูล (Export) ออกมาให้เป็นไฟล์ Excel ดังรูปที่ 3.8

	A	B	C	D	E
1	idrecord	username	password	weight	date_time
2	1	win	123456	67.5	1/29/2014 23:25
3	2	win	123456	67.5	1/30/2014 02:41
4	3	pop	567891	67.5	1/30/2014 02:44
5	4	pop	567891	67.5	1/30/2014 02:51
6					

รูปที่ 3.8 ข้อมูลจากฐานข้อมูล (Database) ของประตูรหัส (Password) เป็นไฟล์ Excell

3.1.1.3 การเขียนโปรแกรมอินเตอร์เฟส สำหรับการตรวจสอบอุปกรณ์ที่มีในระบบ

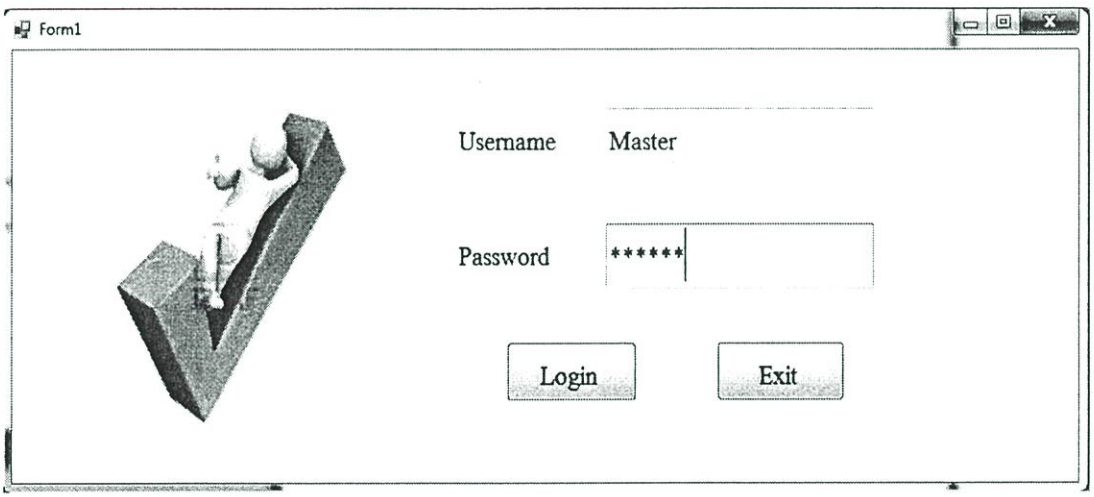
ในส่วนนี้จะแบ่งออกเป็น 2 ส่วนด้วยกัน

1). ชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password)

2). ส่วนตรวจสอบอุปกรณ์ในระบบ

1) ชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password)

ในส่วนนี้เป็นส่วนของความปลอดภัยของระบบ โดยในส่วนนี้จะเป็นมีการพิมพ์ ชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password) โดยในส่วนนี้จะใช้ ชื่อผู้ใช้(Username) เป็น Master และรหัสผู้ใช้เป็น 1 a 2w ซึ่งเหตุผลที่ใช้ชื่อผู้ใช้และรหัสผู้ใช้เพียง 1 ชุด เพราะระบบนี้จะใช้กับบุคคลที่มีหน้าที่ดูแลระบบรักษาความปลอดภัยเท่านั้น ซึ่งเมื่อชื่อผู้ใช้และรหัสผู้ใช้มีความถูกต้องระบบจะส่งรหัสแอสกี้ เป็นตัวอักษรซี (C) ไม่ยังระบบไมโครคอนโทรลเลอร์ (Microcontroller) เพื่อเปิดระบบในตัวไมโครคอนโทรลเลอร์

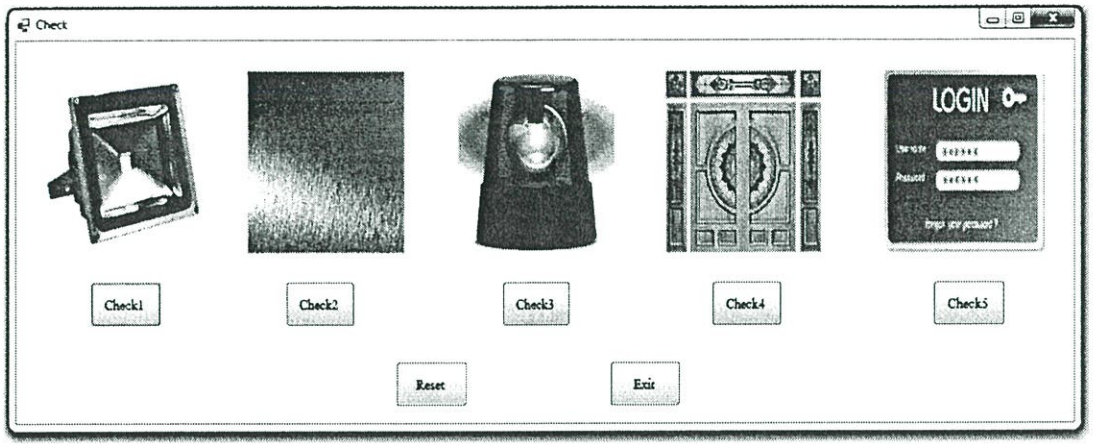


รูปที่ 3.9 หน้าต่างแสดงรูปแบบในการระบุชื่อผู้ใช้ (Username) ,รหัสผู้ใช้ (Password)

2) ส่วนตรวจสอบอุปกรณ์ของระบบ

เป็นส่วนที่มีไว้เพื่อทำการตรวจสอบการทำงานของอุปกรณ์โดยการกดปุ่ม Check ระบบจะทำการส่งรหัสแอสกี้ (ASCII) ออกไปให้ระบบไมโครคอนโทรลเลอร์ เมื่อทำการตรวจสอบการทำงานของสปอร์ตไลท์ (Sportlight) ระบบจะทำการส่งรหัสแอสกี้เป็นตัวอักษรเอล (L) ,กำแพงเลื่อนขึ้น (Wall) ระบบจะทำการส่งรหัสแอสกี้เป็นตัวอักษรดับเบิ้ลยู (W) ,เสียงเตือน (Alarm) ระบบจะทำการส่งรหัสแอสกี้เป็นตัวอักษรเอ (A) ,ประตูหน้าล็อก (Lock) ระบบจะทำการส่งรหัสแอสกี้เป็นตัวอักษรเอฟ (F) ,ประตูล็อกพาสเวิร์ด (Username&Password) ระบบจะทำการ

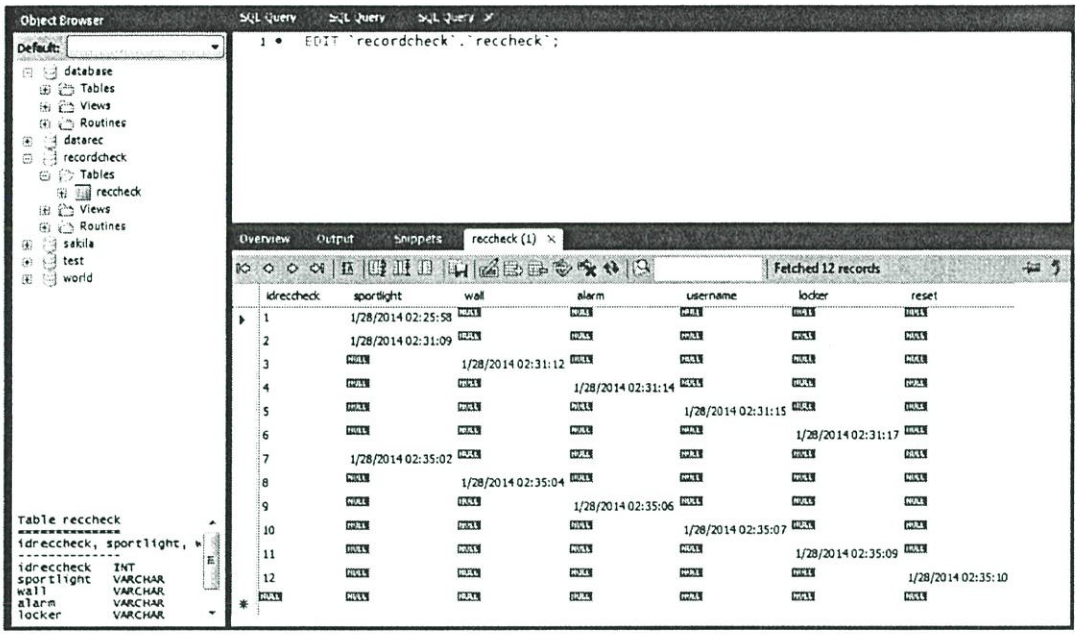
ส่งรหัสแอสกี้เป็นตัวอักษรบี (B) เมื่อจะทำการรีเซ็ตระบบ ระบบจะทำการส่งรหัสแอสกี้เป็นตัวอักษรอาร์ (R) และเมื่อกดปุ่มออก (Exit) ระบบจะส่งรหัสแอสกี้เป็นตัวอักษรเอ็ก (X)



รูปที่ 3.10 ส่วนอินเตอร์เฟส (Interface) ในส่วนของการตรวจสอบการทำงาน

3.1.1.4 การสร้างฐานข้อมูล (Database) เพื่อเก็บข้อมูลประวัติ (History) จากการใช้ระบบตรวจสอบอุปกรณ์ในระบบ

เป็นการสร้างตาราง (Table) ฐานข้อมูล เมื่อนำข้อมูลที่กด Check มาเปลี่ยนเป็นวันเวลา (DateTime) ที่ทำการตรวจสอบแล้วนำไปเก็บไว้ในตาราง reccheck ของฐานข้อมูล (Database) recordcheck โดยฐานข้อมูลจะใช้โปรแกรม MySQL Workbench 5.2 CE เป็นโปรแกรมในการสร้างฐานข้อมูลดังรูปที่ 3.11



รูปที่ 3.11 แสดงฐานข้อมูล (Database) ที่ใช้บันทึกข้อมูลการใช้ระบบตรวจสอบอุปกรณ์

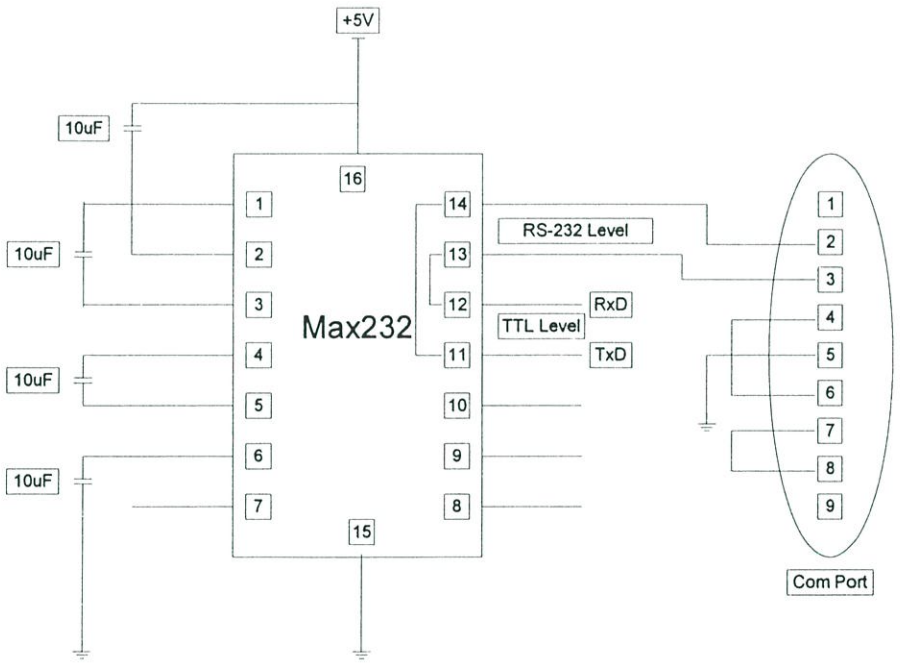
เมื่อทำการเก็บข้อมูลจากฐานข้อมูล (Database) ของระบบตรวจสอบอุปกรณ์ โดยโปรแกรม MySQL Workbench CE 5.2 ซึ่งโปรแกรมนี้สามารถดึงข้อมูล (Export) ออกมาให้เป็นไฟล์ Excel ดังรูปที่ 3.12

	idreccheck	sportlight	wall	alarm	username	locker	reset
2	1	1/28/2014 02:25					
3	2	1/28/2014 02:31					
4	3		1/28/2014 02:31				
5	4			1/28/2014 02:31			
6	5				1/28/2014 02:31		
7	6					1/28/2014 02:31	
8	7	1/28/2014 02:35					
9	8		1/28/2014 02:35				
10	9			1/28/2014 02:35			
11	10				1/28/2014 02:35		
12	11					1/28/2014 02:35	
13	12						1/28/2014 02:35
14	13	1/29/2014 01:03					
15	14		1/29/2014 01:05				
16	15			1/29/2014 01:06			
17	16				1/29/2014 01:07		
18	17				1/29/2014 01:08		
19	18				1/29/2014 01:08		
20	19				1/29/2014 01:08		
21	20				1/29/2014 01:08		
22	21				1/29/2014 01:08		
23	22				1/29/2014 01:08		
24	23				1/29/2014 01:08		

รูปที่ 3.12 ข้อมูลจากฐานข้อมูล (Database) ของระบบตรวจสอบอุปกรณ์ เป็นไฟล์ Excell

3.1.1.5 วงจรแปลงสัญญาณ RS-232 ที่ออกจากพอร์ตอนุกรม (Serial Port) เป็นสัญญาณ TTL

ในส่วนนี้เป็นการแปลงสัญญาณที่ส่งออกจากพอร์ทอนุกรม (Serial Port) หรือสัญญาณ RS-232 ให้เป็นสัญญาณ TTL (Transistor-Transistor Logic) โดยสัญญาณ RS-232 จะมีลอจิก (Logic) 1 ที่ 3 Volts ถึง -15 Volts และมีลอจิก (Logic) 0 ที่ +3 Volts ถึง +15 Volts ดังนั้นช่วงของสัญญาณ RS-232 จะอยู่ที่ 15 Volts ถึง +15 Volts ส่วนสัญญาณ TTL (Transistor-Transistor Logic) จะมีลอจิก (Logic) 1 ที่ 0 Volts ถึง +5 Volts และมีลอจิก (Logic) 0 ที่ 0 Volts เนื่องจากสัญญาณที่จะเข้าไมโครคอนโทรลเลอร์สำหรับลอจิก (Logic) 1 จะเป็น +5 Volts และลอจิก (Logic) 0 จะเป็น 0 Volts คือสัญญาณ TTL (Transistor-Transistor Logic) โดยสองสัญญาณจะสามารถแปลงกลับไป-มาได้ โดยใช้ไอซีเบอร์ MAX232 ดังรูปที่ 3.13 จากนั้นจึงจะเข้าสู่ตัวไมโครคอนโทรลเลอร์เพื่อประมวลผลต่อไปนั่นเอง



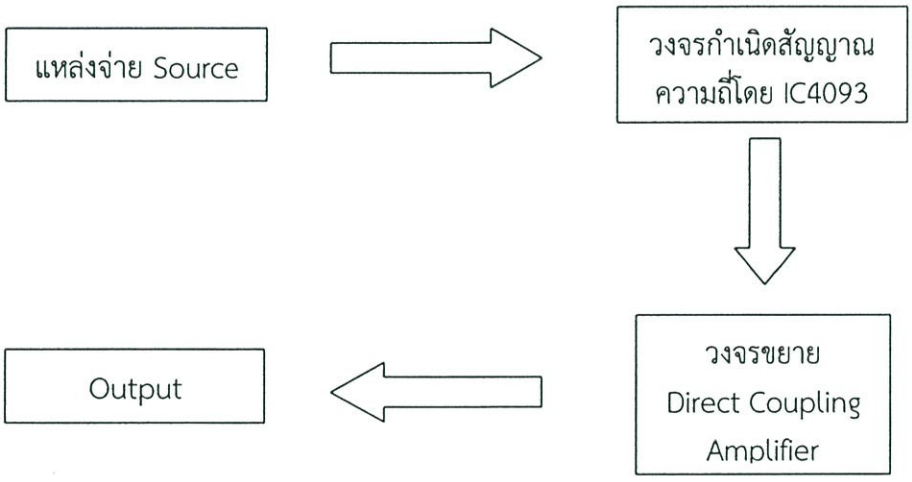
รูปที่ 3.13 วงจรแปลงสัญญาณ RS-232 เป็นสัญญาณ TTL

3.1.2 เซ็นเซอร์อินฟราเรด (Infrared sensor)

เซ็นเซอร์อินฟราเรดจะถูกเปิดใช้ในโหมดการทำงานเวลากลางคืนเพื่อตรวจจับสิ่งผิดปกติ โดยจะแบ่งออกเป็นทางฝั่งส่งและฝั่งรับ เมื่อมีวัตถุมากั้นระหว่างอุปกรณ์ทั้งสอง จะมีการส่งสัญญาณไปยังไมโครคอนโทรลเลอร์เพื่อทำการแจ้งเตือน

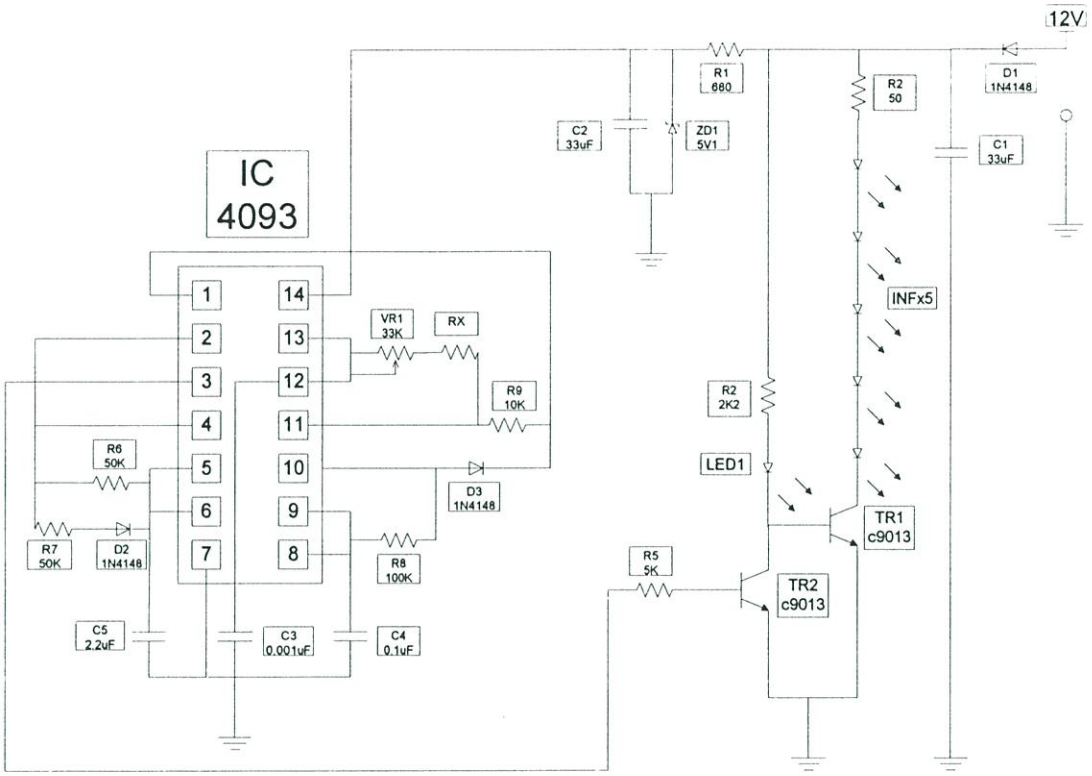
3.1.2.1 ภาคส่งของเซ็นเซอร์แสงอินฟราเรด (Infrared Transmitter)

ในส่วนนี้เป็นภาคการส่งสัญญาณอินฟราเรด (Infrared) ในภาคส่งจะเป็นฝั่งที่ทำหน้าที่ในการส่งสัญญาณแสงอินฟราเรดออกมายังฝั่งรับโดยการรับ-ส่งสัญญาณอินฟราเรดนั้น ฝั่งส่งจะส่งสัญญาณอินฟราเรดตลอดเวลาตามที่ฝั่งส่งยังมีการป้อนแหล่งจ่ายไฟอยู่ ซึ่งวงจรเซ็นเซอร์แสงอินฟราเรด (Infrared Sensor) จะใช้ตัวปล่อยแสงอินฟราเรดจำนวน 5 ตัวเพื่อเป็นการเพิ่มแรงในการส่งสัญญาณทำให้ระยะทางที่สามารถเซ็นเซอร์ (Sensor) ได้ไกลเพิ่มขึ้น โดยบล็อกไดอะแกรม (Block Diagram) การทำงานจะแสดงดังรูปที่ 3.14



รูปที่ 3.14 บล็อกไดอะแกรม (Block Diagram) การทำงานของวงจรส่งสัญญาณอินฟาเรด (Infrared)

3.1.2.2 วงจรเซ็นเซอร์อินฟราเรดภาคส่ง (Infrared Sensor Transmitter)

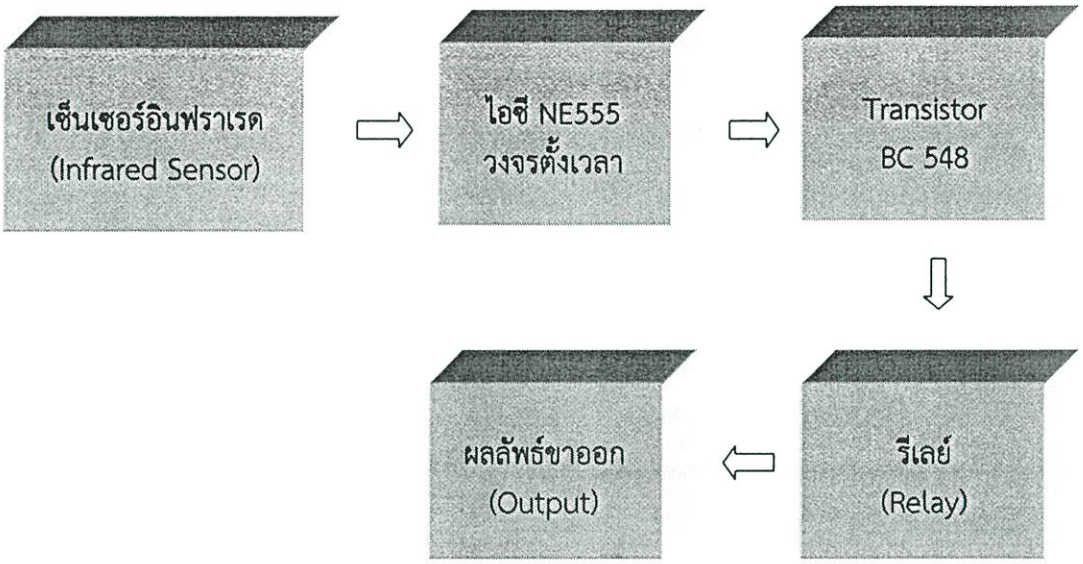


รูปที่ 3.15 วงจรภาคส่งของเซ็นเซอร์แสงอินฟราเรด (Infrared)

จากรูปที่ 3.15 ส่วนของแหล่งจ่ายจะใช้แรงดันที่ 12 Volts และเข้าสู่วงจรกำเนิดสัญญาณความถี่ที่ไอซีเบอร์ CD4093 โดยจะใช้ VR1 เป็นตัวปรับความถี่ของสัญญาณที่จะถูกส่งออกไปยังภาครับของวงจรเซ็นเซอร์แสงอินฟราเรด (Infrared) และเข้าสู่วงจรขยาย (Direct Coupling Amplifier Circuit) ที่เป็นการต่อโดยตรงแบบวงจรดาลิงตั้น (Darlington Circuit) โดยจะต่อทรานซิสเตอร์ (Transistor) รวมกันสองตัวโดยตรงทำให้อัตราขยายกระแสขยายจากทรานซิสเตอร์ตัวแรก จากนั้นขยายจากทรานซิสเตอร์ตัวที่สองอีกทอดหนึ่งด้วยเหตุนี้ทำให้มีอัตราขยาย (Gain) สูงมากซึ่งวงจรขยายแบบ Darlington จะทำการขยายทุกสัญญาณที่ผ่านเข้ามาโดยการขยายที่ไม่ต้องการจะเรียกว่าดริฟท์ (Drift) และส่งสัญญาณออกทางเอาต์พุตโดยหลอดไฟอินฟราเรด (Infrared) 5 ตัว

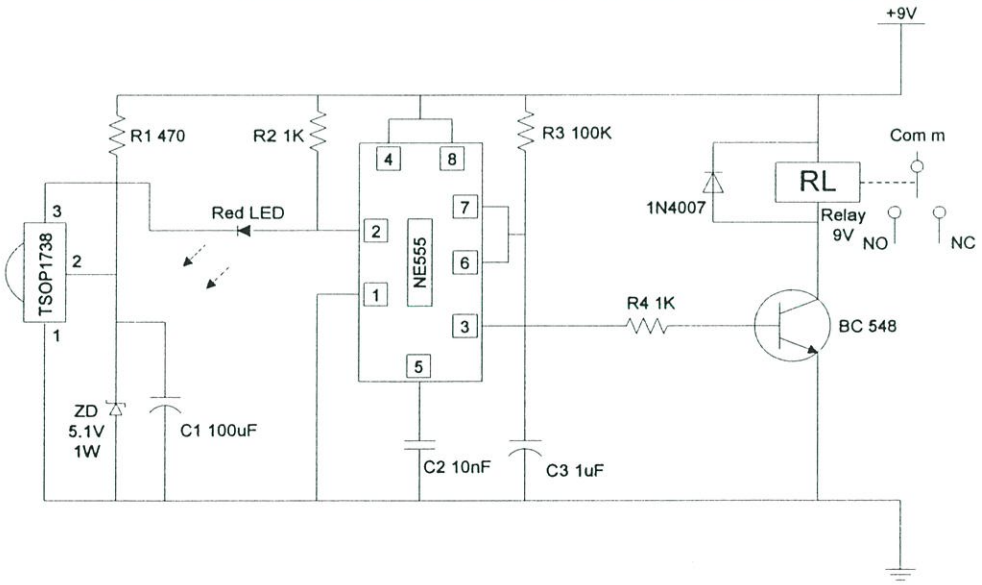
3.1.2.3 ภาครับของเซ็นเซอร์แสงอินฟราเรด (Infrared Transmitter)

ภาครับจะรับสัญญาณจากทางภาคส่งแล้วขยายกำลังโดยไอซีเบอร์ LM324N แล้วขยายกำลังอีกครั้งด้วยวงจรขยายจากนั้นรีเลย์ (Relay) จะเป็นตัวตัดสินใจในการทำงานของวงจรถ้าวงจรเกิดการทำงานก็จะได้ผลลัพธ์ออกมา ดังรูปที่ 3.16



รูปที่ 3.16 บล็อกไดอะแกรมภาครับเซ็นเซอร์อินฟราเรด

3.1.2.4 วงจรเซ็นเซอร์อินฟราเรดภาครับ (Infrared Sensor Receiver)



รูปที่ 3.17 วงจรเซ็นเซอร์อินฟราเรดภาครับ

ภาครับจะรับสัญญาณจากภาคส่งผ่านทางเซ็นเซอร์อินฟราเรด (Infrared Sensor) จากนั้นสัญญาณจะถูกส่งผ่านไปยัง IC NE555 Timer **เพื่อกำหนดให้** IC 555 ทำหน้าที่เป็นตัวตั้ง

เวลาให้ไปสั่ง Relay ทำงานโดยผ่าน Transistor BC548 หลังจากนี้สัญญาณจะถูกส่งเข้าไปที่รีเลย์ (Relay) ซึ่งเป็นตัวที่ทำหน้าที่ตัดหรือต่อวงจรคล้ายกับสวิตช์โดยใช้หลักการหน้าสัมผัส และการที่จะให้มันทำงานก็ต้องจ่ายไฟให้มันตามที่กำหนด เพราะเมื่อจ่ายไฟให้กับตัวรีเลย์ มันจะทำให้หน้าสัมผัสติดกัน กลายเป็นวงจรปิดโดยไฟเลี้ยงที่เราต้องใช้คือ 9 Volts เมื่อเราจ่ายไฟถึงขนาดที่กำหนดแล้วจะได้ผลลัพธ์คือ สัญญาณดิจิตอล (Digital Signal) ที่มีขนาด 5 Volts และ 0 Volts

การคำนวณวงจรตั้งเวลา โดย IC 555

จากสูตร $Time = 1.1 \times R_3 \times C_3$

$R_3 = 100K\Omega$

$C_3 = 1\mu F$

$Time = 1.1 \times 100 \times 10^{3} \times 1 \times 10^{-6}$

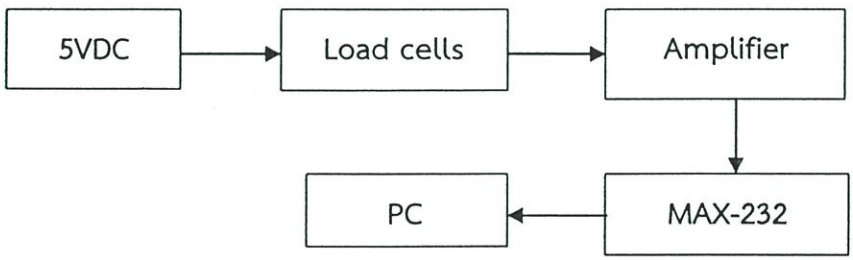
$= 0.11$ sec

3.1.3 โปรแกรมและอุปกรณ์การรับค่าน้ำหนัก (Load-cell)

โหลดเซลล์ (Load Cell) จะรับค่าน้ำหนักที่ถูกกดลงบนเครื่องชั่งน้ำหนักแล้วนำค่าที่อ่านได้ไปแสดงผลที่คอมพิวเตอร์

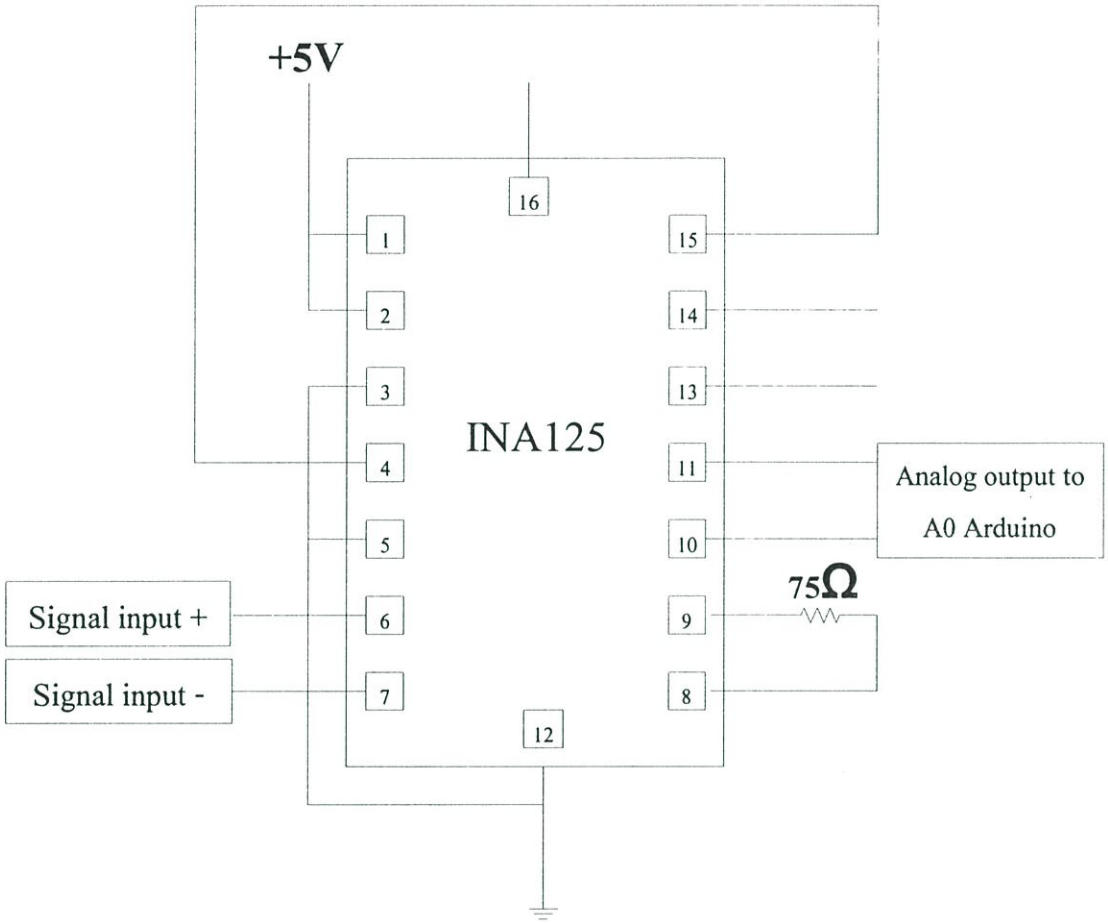
3.1.3.1 โหลดเซลล์ (Load-Cell)

โหลดเซลล์ (Load Cell) จะทำงานโดยรับแรงกดจากเครื่องชั่งน้ำหนักโดยต้องจ่ายไฟเลี้ยงขนาด 5 Volts ให้โหลดเซลล์ซึ่งโหลดเซลล์จะรับไฟเลี้ยงจากบอร์ดอาดูโน่ (Arduino) ซึ่งในบอร์ดอาดูโน่ก็จะทำการแปลงสัญญาณอนาล็อก (Analog Signal) ที่รับมาจากโหลดเซลล์เป็นสัญญาณดิจิตอล (Digital Signal) เมื่อได้ค่าของแรงแล้วจะต้องแปลงแรงกดเป็นสัญญาณไฟฟ้า แต่สัญญาณไฟฟ้าที่ได้จะมีขนาดน้อยมากมีหน่วยเป็นมิลลิโวลต์ (Millivolts) จึงต้องขยายแรงดันให้มีขนาด 0-5 Volts โดยใช้ไอซีเบอร์ INA125 เมื่อขยายแรงดันแล้วจึงใช้ MAX-232 แปลงสัญญาณ TTL ให้เป็นสัญญาณ RS-232 แล้วส่งไปแสดงผลที่คอมพิวเตอร์ ดังรูปที่ 3.18



รูปที่ 3.18 บล็อกไดอะแกรมโหลดเซลล์

3.1.3.2 วงจรขยายแรงดัน



รูปที่ 3.19 วงจรขยายแรงดัน

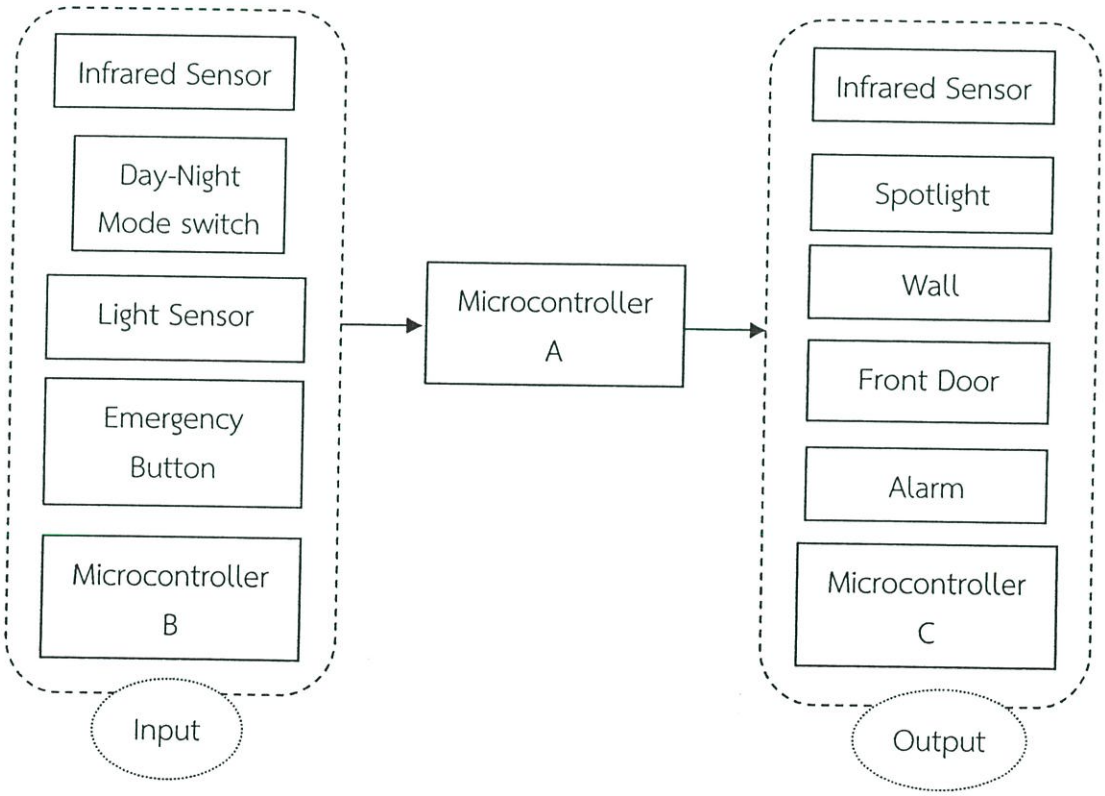
โหลดเซลล์ (Load-cell) จะรับน้ำหนักจากแรงกดที่เครื่องชั่งน้ำหนักแล้วแปลงแรงที่อ่านได้เป็นสัญญาณไฟฟ้าขนาด 0.2-4.2 mV แล้วขยายแรงดัน (Voltage Gain) ด้วยไอซีเบอร์ INA125 ที่ขาหก ให้แรงดันมีขนาด 1-5 V จากนั้นเอาท์พุต (Output) ที่ขาสิบของไอซีเอาท์พุตจะส่งออกจากพอร์ต Tx ของบอร์ดอาดูโน่ซึ่งสัญญาณที่ออกมาจะเป็นสัญญาณดิจิตอล (Digitai Signal) แล้วส่งเข้า MAX-232 เพื่อแปลงสัญญาณ TTL เป็นสัญญาณ RS-232 แล้วแสดงค่าน้ำหนักที่อ่านได้บนหน้าจอคอมพิวเตอร์

3.1.3.3 อาดูโน่ (Arduino)

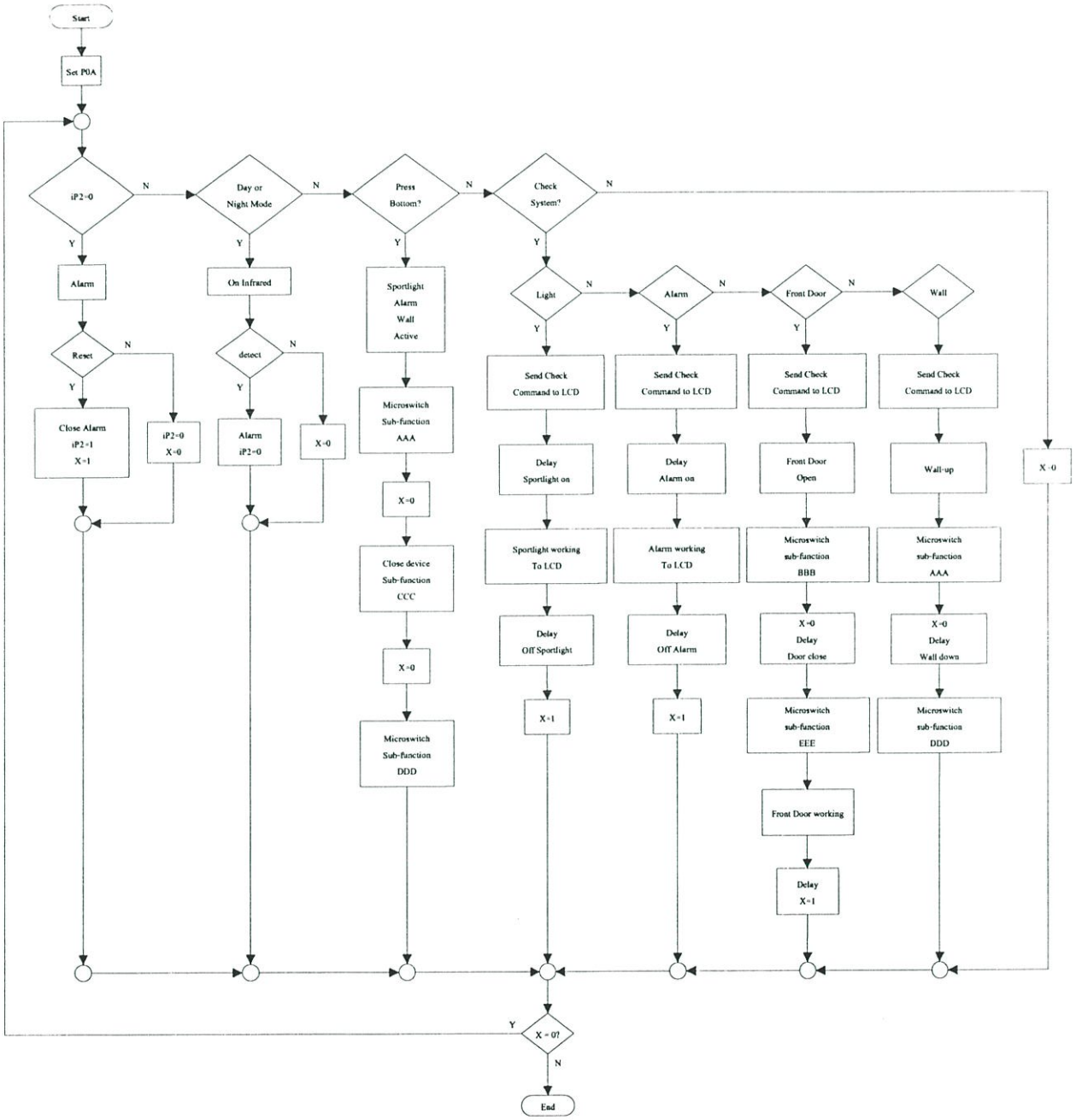
นำอาดูโน่มาใช้เพื่อรับสัญญาณอนาล็อก (Analog Signal) ที่รับมาจากวงจรขยายแรงดันเพื่ออ่านค่าแรงดันที่รับได้แล้วนำมาเปรียบเทียบกับน้ำหนักที่วัดได้จริงแล้วนำทั้งสองค่ามาทำการเทียบค่ากัน (Mapping)

3.1.4 โปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยในไมโครคอนโทรลเลอร์ตัวที่ 1

เป็นการใช้ไมโครคอนโทรเลอร์ (Microcontroller) เบอร์ AT89C52 เป็นตัวควบคุมการทำงานของอุปกรณ์ป้องกัน และแจ้งเตือนภัย โดยการกลไกและลำดับการทำงานจะเป็นไปตามความต้องการของผู้เขียน ซึ่งจะอธิบายบล็อกไดอะแกรม (Block Diagram) อินพุต (Input) และเอาท์พุต (Output) ที่ผู้เขียนกำหนดดังรูปที่ 3.20 โดยการทำงานจะเป็นไปตามแผนผังการทำงาน (Flow Chart) โดยใช้ภาษาซี (C) ซึ่งจะอธิบายการทำงานในส่วนนี้ได้ดังรูปที่ 3.21



รูปที่ 3.20 บล็อกไดอะแกรม Input-Outputในไมโครคอนโทรลเลอร์ตัวที่ 1



รูปที่ 3.21 แผนผังลำดับการทำงานของโปรแกรมป้องกันและแจ้งเตือนภัยในส่วนที่ 1

3.4.1 โปรแกรมภาษาซีที่ใช้ในการทำงานของไมโครคอนโทรลเลอร์ตัวที่ 1

ในส่วนของตัวโปรแกรมจะทำการเขียนโค้ดภาษาซี (C Code) จากแผนผังลำดับการทำงาน (Flow Chart) จากรูปที่ 3.21 ผ่านโปรแกรม Keil ซึ่งโค้ดภาษาซี (C Code) จะถูกแบ่งออกเป็น 3 ส่วน ได้แก่

3.1.4.1 ส่วนกำหนดตัวแปร (Variable) และฟังก์ชั่น (Function) ย่อย

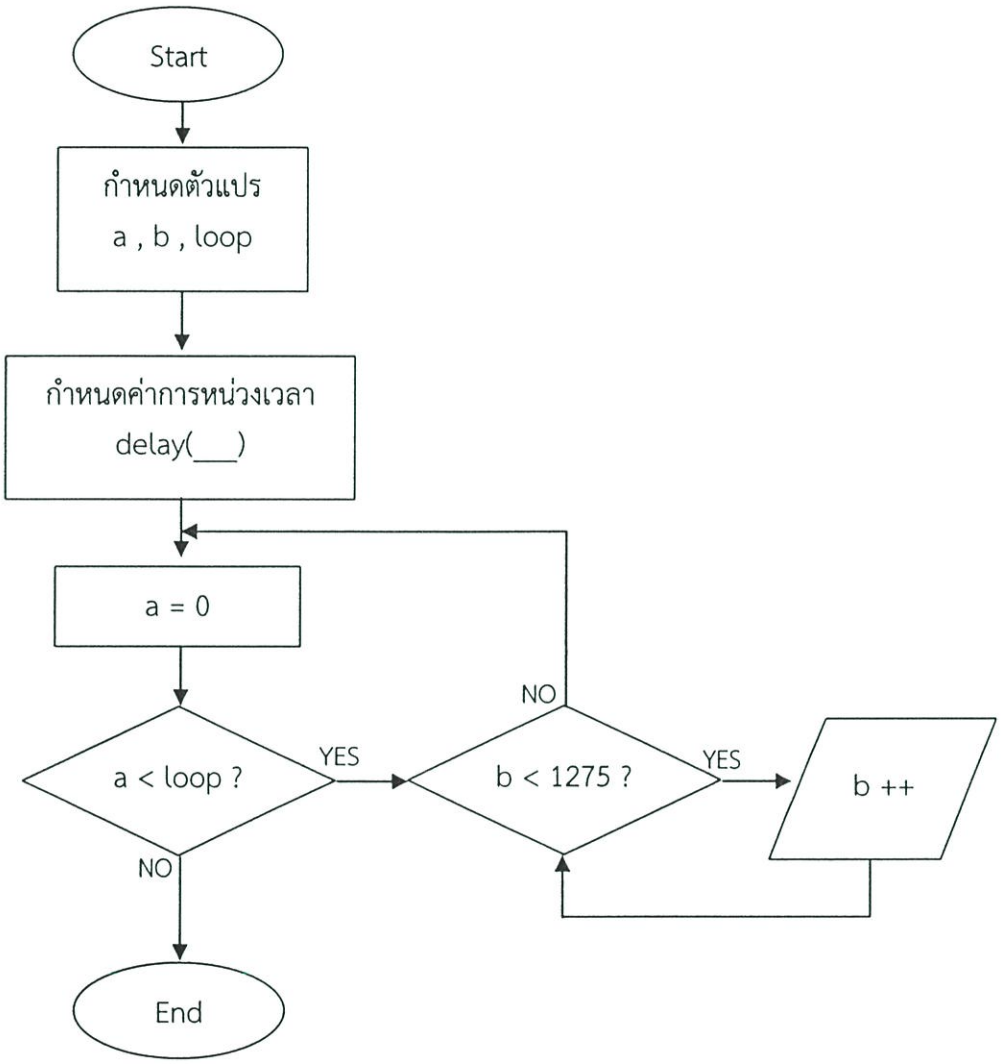
ภายในโปรแกรมจะมีการกำหนดตัวแปรของอินพุต (Input) และเอาท์พุต (Output) โดยจะกำหนดเป็นตัวแปรภายนอก (Global Variable) ซึ่งเป็นการกำหนดว่า ต้องการใช้พอร์ต (Port) ใดเป็นพอร์ตอินพุต (Input port) และพอร์ตเอาท์พุต (Output) จากนั้นจึงจะสามารถนำตัวแปร (Variable) ดังกล่าวไปใช้ในฟังก์ชั่นหลัก (Main Function) โดยการแทนค่าพอร์ตต่างๆ จะแสดงในรูป 3.22 และเนื่องจากโปรแกรมนี้มีการเรียกใช้ฟังก์ชั่นย่อย จึงจะต้องกำหนดฟังก์ชั่นนั้นๆ เช่นกัน เพื่อที่จะสามารถนำฟังก์ชั่นดังกล่าวไปใช้ในฟังก์ชั่นหลัก โดยในที่นี้ฟังก์ชั่นย่อยที่ใช้คือ ฟังก์ชั่นดีเลย์ (Delay Function) , ฟังก์ชั่นกำหนดการส่งอัตราบิต (Baud Rate Fucntion), ฟังก์ชั่นการรับค่าจากไมโครสวิทช์ (Microswitch), ฟังก์ชั่นรอคำสั่งปิดอุปกรณ์ ซึ่งจะแสดงในรูปที่ 3.22

```
001 #include <reg52.h>
002 #include <string.h>
003 #include <stdio.h>
004
005 sbit nightmode = P1^0 ;              /* day or night mode */
006 sbit lightcheck = P1^1 ;             /* infrared sensor */
007 sbit presscheck = P1^2 ;             /* emergency button */
008 sbit ip2 = P1^3 ;                    /* alarm if incorrect login */
009 sbit ChkS = P0^1 ;                   /* check system */
010 sbit ok1 = P0^4 ;                    /* say okay in check system to LCD */
011
012
013 sbit misw1fw = P1^4 ;                /* micro switch wall stop forward */
014 sbit misw2fw = P1^5 ;                /* micro switch front door stop forward */
015 sbit misw1bw = P1^6 ;                /* micro switch wall stop backward */
016 sbit misw2bw = P1^7 ;                /* micro switch front door stop backward        */
017                                      /*          P1 - Input
018                                               P2 - Output      P2.0-2.1 = wall
019                                                               P2.2-2.3 = front door
020                                                               P2.4-2.5 = enable motor
021                                                               P2.6 = spotlight
022                                                               P2.7 = alarm         */
023 sbit Reset = P0^0 ;                 /* RESET SYSTEM */
024 int x ;
025 char RxBuff ;                       /* Rx variable */
026 void delay(unsigned int loop) ;     /* Delay */
027 void uart init(void) ;              /* Baud rate */
028 void AAA();                        /* Stop wall-up */
029 void BBB();                        /* Stop front door-open */
030 void CCC();                        /* wait for RESET */
031 void DDD();                        /* Stop wall-down */
032 void EEE();                        /* Stop front door-close */
```

รูปที่ 3.22 การกำหนดตัวแปรและฟังก์ชั่นย่อยภายในตัวที่ 1

3.1.4.2 ฟังก์ชั่นย่อยดีเลย์ (Delay Function)

เป็นฟังก์ชั่น (Function) ที่ใช้กำหนดการหน่วงเวลาของโปรแกรม ซึ่งในที่นี้ถูกกำหนดเป็นฟังก์ชั่นย่อย โดยจะแสดงแผนผังลำดับการทำงาน (Flow Chart) ได้ ดังรูปที่ 3.23



รูปที่ 3.23 แผนผังลำดับการทำงานของฟังก์ชั่นดีเลย์ (Delay Function)

จากรูปที่ 3.23 จะนำแผนผังลำดับการทำงาน (Flow Chart) ดังกล่าวไปเขียนโค้ดภาษาซี (C Code) ได้ดังรูปที่ 3.24

```
034  /*DELAY FUNCTION*/
035  void delay(unsigned int loop)
036  {
037      unsigned int a , b ;
038      for(a=0;a<loop;a++)
039      {
040          for(b=0;b<1275;b++) ;
041      }
042  }
```

รูปที่ 3.24 ฟังก์ชั่นดีเลย์ (Delay Function)

การวนลูป (Loop) โดยใช้คำสั่ง for เมื่อการทำงานครบ 1 รอบ จะใช้เวลาจริงไปเท่ากับ 1 ไมโครวินาที โดยการวนลูปภายใต้คำสั่ง for ของตัวแปร b จะถูกกำหนดให้วน 1275 รอบ ซึ่งลูปดังกล่าวอยู่ภายใต้การวนลูป1 รอบ ภายใต้คำสั่ง for ของตัวแปร a ดังนั้นจะเท่ากับว่า เมื่อใส่ค่าฟังก์ชั่นดีเลย์ (Delay function) เป็น " delay(1) " ในฟังก์ชั่นหลัก จะเป็นการหน่วงเวลาไป 0.01275 วินาที หรือ 12.75 มิลลิวินาที (Millisecond) นั่นเอง และสามารถนำไปกำหนดได้ว่า จะต้องการหน่วงเวลาไปเท่าใด โดยการกำหนดจำนานรอบของลูปนั่นเอง

3.1.4.3 ฟังก์ชั่นย่อยอัตราบิตในพอร์ตอนุกรม (Baud Rate Function)

การสื่อสารทางพอร์ตอนุกรมจะต้องมีการตั้งค่าอัตราบิตที่จะใช้ในการสื่อสารให้ตรงกันระหว่างไมโครคอนโทรลเลอร์ที่ทำหน้าที่ส่งและรับ รวมถึงคอมพิวเตอร์ด้วย โดยในที่นี้จะกำหนดอัตราบิตเท่ากับ 9600 MBPS ซึ่งการตั้งค่าโค้ดภาษาซีของโปรแกรมจะอธิบายตามรูปที่ 3.25

```
044  /* BAUD RATE 9600Mbps (SERIAL PORT)*/
045  void uart_init(void)
046  {
047      TR1 = 0 ;           /*Timer 1 Disable*/
048      ET1 = 0 ;           /*Enable Time1 Interupt*/
049      TMOD = 0x20 ;       /*Set Timer 1 as Mode 2*/
050      TH1 = 0xFD ;        /*Set Buad rate 9600 bps*/
051      TR1 = 1 ;           /*Timer 1 Enable*/
052      SCON = 0x50 ;       /*Uart Enable and Set Uart as Model*/
053      TI = 1 ;            /*Set state for send*/
054  }
```

รูปที่ 3.25 ฟังก์ชั่นกำหนดอัตราการรับ-ส่งข้อมูล (Baud Rate Function)

โดยจะอธิบายการทำงานได้ดังนี้

บรรทัดที่ 47 : Set bit TR1 = 0 เพื่อทำให้แน่ใจว่าการทำงานของ Timer1 ยังไม่เริ่มต้นขึ้น

บรรทัดที่ 48 : กำหนดให้ใช้ Timer ตัวที่ 1 ในการทำงาน

บรรทัดที่ 49 : เป็นการกำหนดให้ตัว Timer1 ทำงานในโหมด 2 ซึ่งรีจิสเตอร์ TMOD (Timer/Counter Mode Control Register) เป็นรีจิสเตอร์ (Register) ขนาด 8 บิต และไม่สามารถเข้าถึงได้ในระดับบิตได้ แบ่งออกเป็น 2 ส่วนคือ ส่วน 4 บิตล่างใช้กำหนดโหมดการทำงานของไทเมอร์ (Timer) 0 และ 4 บิตบนใช้กำหนดการทำงานของไทเมอร์ 1 มีโครงสร้างดังแสดงในรูปที่ 3.26

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Timer 1				Timer 0			

รูปที่ 3.26 โครงสร้างของรีจิสเตอร์ TMOD (Timer/Counter Mode Control Register)

โดย บิตต่างๆของรีจิสเตอร์จะสามารถอธิบายได้ดังนี้

GATE : ใช้เลือกการควบคุมการทำงานของไทเมอร์ โดยถ้าเป็น 0 จะเป็นการควบคุมทางซอฟท์แวร์ นั้นคือไทเมอร์จะทำงานเมื่อบิต TR ในรีจิสเตอร์ TCON เป็น 1 แต่ในกรณีที่บิต GATE เป็น 1 จะเป็นการควบคุมทางฮาร์ดแวร์ (Hardware) นั้นคือไทเมอร์จะทำงานเมื่อบิต TR ในรีจิสเตอร์ TCON เป็น 1 และ ขาอินพุต INT ของไมโครคอนโทลเลอร์มีสถานะลอจิกเป็นเป็น 1 ด้วย

C/T (Timer or Counter Selector) ใช้เลือกการทำงานของ Timer/Counter โดยถ้าเป็น 0 จะทำงานเป็น Timer แต่ถ้าเป็น 1 จะทำงานเป็น Counter

M1, M0 (Mode Selector Bit) ใช้เลือกโหมดการทำงานของ Timer/Counter โดยถ้า เป็น 00 ทำงานในโหมด 0, 01 ทำงานในโหมด 1, 10 ทำงานในโหมด 2, 11 ทำงานในโหมด 3

การทำงานในโหมด 0 เป็นการทำงานเป็น Timer/Counter 13 บิต นั้นคือใช้งานรีจิสเตอร์ TL 5 บิต และ TH 8 บิต ดังนั้นเมื่อค่าในรีจิสเตอร์ TL และ TH เพิ่มขึ้นจนเป็น 1 หมดทั้ง 13 บิต บิต TF ภายในรีจิสเตอร์ TCON ก็จะถูกเซต เพื่อแสดงการเกิดโอเวอร์โฟล์ว (Overflow)

การทำงานในโหมด 1 เป็นการทำงานเป็น Timer/Counter 16 บิต นั้นคือใช้งานรีจิสเตอร์ TL และ TH ครบทั้ง 8 บิต ดังนั้นเมื่อค่าในรีจิสเตอร์ TL และ TH เพิ่มขึ้นจนเป็น 1 หมดทั้ง 16 บิต บิต TF ภายในรีจิสเตอร์ TCON ก็จะถูกเซต เพื่อแสดงการเกิด Overflow

การทำงานในโหมด 2 เป็นการทำงานเป็น Timer/Counter 8 บิต แบบตั้งค่าอัตโนมัติ (Auto Reload) โดยรีจิสเตอร์ TL ทำงานเป็นตัวนับ ส่วนรีจิสเตอร์ TH ทำงานเป็นตัวเก็บค่าเริ่มต้นของการนับ เมื่อเริ่มต้นการทำงานค่ารีจิสเตอร์ TL จะถูกเซตให้เหมือนกับค่าในรีจิสเตอร์ TH และเริ่มการนับแบบ 8 บิต เมื่อนับจนเกิด Overflow (ค่าในรีจิสเตอร์ TL เป็น "1" หมดทุกบิต) จะทำให้บิต TF ภายในรีจิสเตอร์ TCON จะถูกเซต ส่วนรีจิสเตอร์ TL ก็จะถูกเซตให้เหมือนกับค่าในรีจิสเตอร์ TH และเริ่มต้นการนับใหม่อีกครั้ง

การทำงานในโหมด 3 จะสามารถใช้ได้เฉพาะกับ Timer0 เท่านั้น โดยมีการทำงานของ TH0 และ TL0 แยกกันอิสระ TL0 สามารถเลือกการทำงานเป็นได้ Timer หรือ Counter ได้ตามปกติ ถูกควบคุมจากบิตภายในรีจิสเตอร์ TCON และขาสัญญาณ INT0 และเมื่อเกิด overflow เกิดขึ้นจะเกิดการเซตบิต TF0 ส่วนรีจิสเตอร์ TH0 นั้นถูกควบคุมการทำงานโดยบิต TR1 ในรีจิสเตอร์ TCON เท่านั้น และถ้าเกิดการ Overflow ขึ้น จะทำให้เกิดการเซตบิต TF1 ซึ่งในที่นี้การทำงานจะใช้ Timer1 โดยทำงานในโหมด 2

บรรทัดที่ 50 : เป็นการกำหนดค่า Baud Rate ซึ่งมาจากการเกิด Overflow ของ Timer1 เนื่องจากการใช้ Timer1 ให้โหมด 2 และตั้งค่าบิต SM0 และ SM1 ของรีจิสเตอร์ SCON เป็น 0 และ 1 ตามลำดับ โดยจะอธิบายการทำงานของรีจิสเตอร์ SCON ในบรรทัดที่ 52 เราจะสามารถคำนวณค่า TH1 ได้ดังนี้

อัตราการส่งข้อมูล = $2^{SMOD} \times freg.OSC / 32 \times 12 \times [256 - TH1]$

$$TH1 = 256 - (2^{SCON} \times freg.OSC) / (32 \times 12 \times \text{อัตราการส่งข้อมูล})$$

โดยหากต้องการให้ Baud rate = 9600 สามารถคำนวณหาค่า TH1 ได้ โดยการแทนค่าลงในสมการข้างต้น ซึ่งจะได้ว่า

$$TH1 = 256 - (2^{SCON} \times 11.059 \times 10^6) / (32 \times 12 \times 9600)$$

$$TH1 = 253 \text{ หรือ } FDH$$

บรรทัดที่ 51 : ใช้ควบคุมการ ปิด-เปิด การทำงานของ Timer1 สามารถเซต หรือเคลียร์ได้โดยซอฟท์แวร์ (Software) โดยถ้าต้องการให้ Timer1 ทำงานต้องเซตบิตนี้ให้เป็น "1"

บรรทัดที่ 52 และ 53 : เป็นรีจิสเตอร์ขนาด 8 บิต โดยมีโครงสร้างดังรูปที่ 3.27 และโหมดการทำงานของบิตที่ 6 และ 7 ดังตารางที่ 3.1 และการทำงานของแต่ละบิตในรีจิสเตอร์

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

รูปที่ 3.27 โครงสร้างของรีจิสเตอร์ SCON

SM 0	SM 1	โหมด	รายละเอียด	Baud Rate
0	0	0	Shift register 8 บิต	ความถี่สัญญาณนาฬิกา หาร 12
0	1	1	UART 8 บิต	ปรับค่าได้ (หาได้จากคำนวณบรรทัดที่ 23)
1	0	2	UART 9 บิต	ความถี่สัญญาณนาฬิกา หาร 32
1	1	3	UART 9 บิต	ปรับค่าได้

ตางรางที่ 3.1 โหมดการทำงานของรีจิสเตอร์ SCON ของบิตที่ 6 (SM1) และ 7 (SM0)

บิตที่ 5 - SM2 ใช้เพื่อการกำหนดการเลือกจำนาน CPU ที่ใช้ โดยบิต 0 จะเป็นการใช้ CPU ตัวเดียว ส่วนบิต 1 จะเป็นการใช้สำหรับต่อ CPU หลายตัว

บิตที่ 4 - REN จะกำหนดเป็นบิต 0 เมื่อไม่ใช้การสื่อสารผ่านพอร์ตอนุกรม (Serial Port) และบิต 1 เมื่อใช้การสื่อสารผ่านพอร์ตอนุกรม

บิตที่ 3 - TB8 เมื่อกำหนดบิตเป็น 0 จะเป็นการเลือกส่งบิตที่ 10 ในโหมด 2,3 และบิตหยุดในโหมด 1 และบิต 1 จะไม่ส่งบิตที่ 10 ในโหมด 2,3 และบิตหยุดในโหมด 1

บิตที่ 2 - RB8 เมื่อกำหนดบิตเป็น 0 จะเป็นการเลือกรับบิตที่ 10 ในโหมด 2,3 และบิตหยุดในโหมด 1 และบิต 1 จะไม่รับบิตที่ 10 ในโหมด 2,3 และบิตหยุดในโหมด 1

บิตที่ 1 - TI จะเป็น 1 เมื่อมีการส่งข้อมูลบิตสุดท้าย (บิตที่ 7) ออกไป

บิตที่ 0 - RI จะเป็น 1 เมื่อมีการรับข้อมูลบิตสุดท้าย (บิตที่ 7) เข้ามา SBUF

โดยการกำหนดค่ารีจิสเตอร์ SCON ในที่นี้จะเป็น 0x50 และจะสามารถอธิบายโหมดการทำงานที่ 1 ของรีจิสเตอร์ คือ เป็นการรับส่งข้อมูลขนาด 10 บิต โดยจะเริ่มจากบิตเริ่ม (Start Bit) ให้เป็นลอจิก "0" บิตข้อมูล (D0-D7) จำนวน 8 บิต และบิตหยุดให้เป็น "1" สำหรับบิตข้อมูลจะถูกเก็บไว้ในรีจิสเตอร์ SBUF และ บิตหยุด (Stop Bit) จะถูกเก็บไว้ใน RB8 (บิตที่ 2 ของ SCON) โดยจะสามารถคำนวณค่า Baud Rate ในโหมดนี้ จากการกำหนด Overflow ของ Timer1 ซึ่งแสดงในการคำนวณข้างต้นของบรรทัดที่ 50

3.1.4.4 ฟังก์ชั่นย่อยการรับค่าไมโครสวิทช์ (Microswitch Function) และฟังก์ชั่นปิดอุปกรณ์

การเปิดปิดของประตูและเลื่อนขึ้นลงของกำแพงจะมีติดตั้งไมโครสวิทช์ไว้เพื่อกำหนดให้รู้ว่า อุปกรณ์ควรจะหยุดเลื่อนเมื่อใด โดยในที่นี้เมื่ออุปกรณ์เลื่อนถึงและเกิดการแตะตัวไมโครสวิทช์ ซึ่งตัวไมโคสวิทช์จะต่อเข้ากับไมโครคอนโทรลเลอร์โดยมีสถานะปกติทางลอจิกเป็น 1 และเมื่อมีการกดจะเปลี่ยนสถานะ 0 อุปกรณ์ก็จะหยุดเลื่อนทันที โดยการเลื่อนจะใช้ดีซีมอเตอร์ (DC Motor) ซึ่งจะสามารถกำหนดการหมุนได้ดังตารางที่ 3.2

ตารางที่ 3.2 การกำหนดบิตให้กับวงจรขับมอเตอร์

State	+	-
Forward	1	0
Backward	0	1
Fast Stop	1	1
Stop	0	0

ในขณะที่มอเตอร์หมุนจะมีการกำหนดบิตผ่านพอร์ตไมโครคอนโทรลเลอร์ไปยังวงจรขับมอเตอร์ (Drive Motor) เป็นสถานะ Forward , Backward สถานะใดสถานะหนึ่ง ส่วนเมื่อต้องการจะให้หยุดหมุน จะกำหนดบิตให้มอเตอร์หยุดสถานะ Fast Stop ทันที ซึ่งจะไม่มีแรงเฉื่อยของหมุนตัวมอเตอร์ โดยฟังก์ชั่นดังกล่าวจะแสดงในรูปที่ 3.28 ถึง 3.31

```
222 void AAA()                                   /* Stop wall-up */
223 {
224             do
225                 {
226                     if(misw1fw == 0)        /* Microswitch is pushed */
227                         {
228                             P2 = 0xC3 ;     /* Fast Stop */
229                             x = 1 ;
230                         }
231                     else
232                         {
233                             x = 0 ;
234                         }
235                 }
236             while(x==0) ;
237             return ;
238 }
```

รูปที่ 3.28 ฟังก์ชั่นย่อย AAA ของไมโครสวิทช์ในไมโครคอนโทรลเลอร์ตัวที่ 1

```
241 void BBB()                                   /* Stop open front door*/
242 {
243             do
244                 {
245                     if(misw2fw == 0)        /* Microswitch is pushed */
246                         {
247                             P2 = 0xCC ;     /* Fast Stop */
248                             x = 1 ;
249                         }
250                     else
251                         {
252                             x = 0 ;
253                         }
254                 }
255             while(x==0);
256             return ;
257 }
```

รูปที่ 3.29 ฟังก์ชั่นย่อย BBB ของไมโครสวิทช์ในไมโครคอนโทรลเลอร์ตัวที่ 1

```
278  void DDD()                                  /* Stop wall down */
279  {
280              do
281              {
282                  if(misw1bw == 0)            /* Stop wall down */
283                      {
284                          P2 = 0x13 ;         /* Fast Stop */
285                          x = 1 ;
286                      }
287                  else
288                      {
289                          x = 0 ;
290                      }
291              }
292              while(x==0) ;
293              return ;
294  }
```

รูปที่ 3.30 ฟังก์ชั่นย่อย DDD ของไมโครสวิทช์ในไมโครคอนโทรลเลอร์ตัวที่ 1

```
297  void EEE()                                  /* Stop close front door */
298  {
299              do
300              {
301                  if(misw2bw == 0)
302                          {
303                              P2 = 0x2C ; /* Fast Stop */
304                              x = 1 ;
305                          }
306                      else
307                          {
308                              x = 0 ;
309                          }
310              }
311              while(x==0) ;
312              return ;
313  }
```

รูปที่ 3.31 ฟังก์ชั่นย่อย EEE ของไมโครสวิทช์ในไมโครคอนโทรลเลอร์ตัวที่ 1

3.1.4.5 ฟังก์ชั่นย่อยปิดอุปกรณ์ระหว่างการทำงาน

การทำงานของอุปกณ์เมื่อมีการรับค่าจากสถานะต่างๆ เช่น มีการกดปุ่มฉุกเฉิน (Push Emergency Button) , มีการเข้ารหัสผิด (Login Incorrect) หรือเซ็นเซอร์อินฟราเรด (Infrared Sensor) ระบบจะสั่งให้อุปกรณ์ทำงาน ซึ่งการจะหยุดการทำงานของอุปกรณ์เพื่อปรับเปลี่ยนให้เข้าสู่สถานะปกติ ผู้จะต้องมีการส่งคำสั่ง RESET จากโปรแกรมอินเตอร์เฟสของคอมพิวเตอร์เข้ามา จึงจะสามารถกลับสู่สถานะเดิมได้ แสดงดังรูปที่ 3.32 โดยในไมโครคอนโทรลเลอร์ตัวที่ 1 จะมีการรับค่าจากตัวที่ 2 ในกรณีที่มีการเข้ารหัสผิด และรับค่าเมื่อได้รับการกด RESET ส่วนการกดปุ่มฉุกเฉินและเซ็นเซอร์อินฟราเรดซึ่งจะอธิบายต่อไปในฟังก์ชั่นหลักในตัวที่ 1

```
259  void CCC()
260  {
261              do
262              {                                          /* wait for reset*/
263                  if(Reset == 0)
264                      {
265                          P2 = 0x12 ;                    /* backward wall down*/
266                          x = 1 ;
267                      }
268                  else
269                      {
270                          x = 0 ;
271                      }
272              }
273              while(x==0);
274              return ;
275  }
```

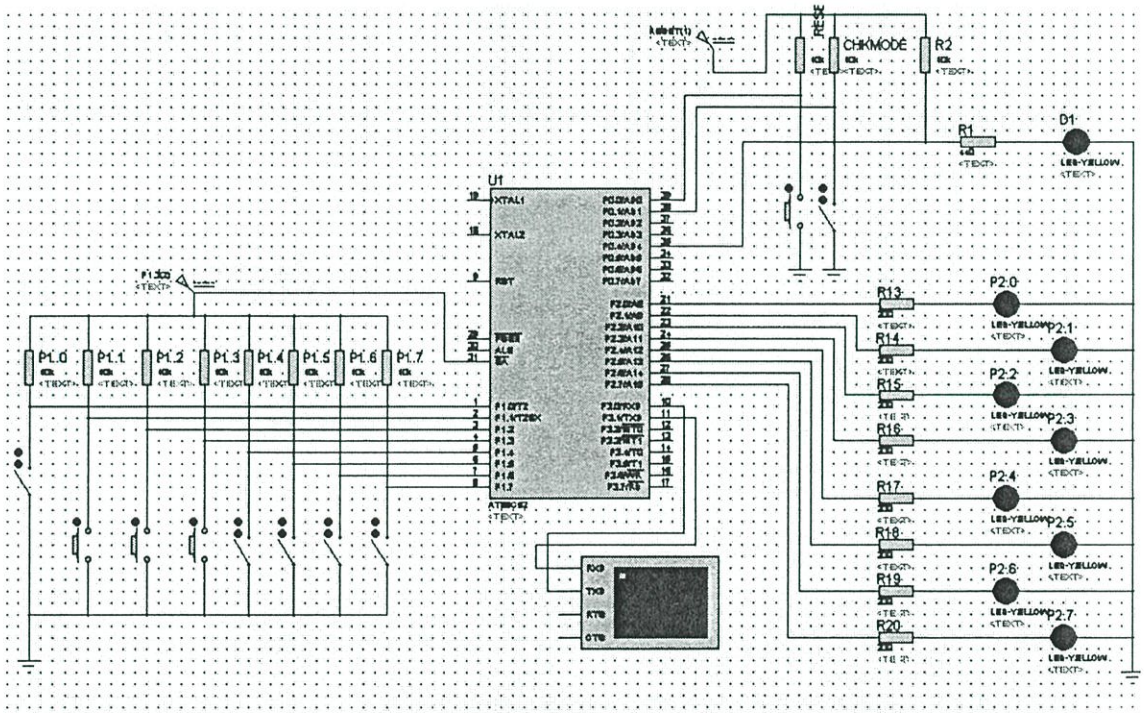
รูปที่ 3.32 ฟังก์ชั่นย่อย CCC ในในการปิดอุปกรณ์ในไมโครคอนโทรลเลอร์ตัวที่ 1

3.4.1.6 ฟังก์ชั่นหลัก (Main Function)

โปรแกรมในส่วนถัดมา จะเป็นโปรแกรมที่ควบคุมการแจ้งเตือนและป้องกันภัย โดยแผนผังลำดับการทำงานจะสามารถแสดงได้จากรูปที่ 3.21 ซึ่งจะมีการกำหนดให้พอร์ตหนึ่ง (Port1) และพอร์ตศูนย์ (Port0) ทำหน้าที่เป็นอินพุต (Input) และส่วนพอร์ตสอง (Port2) ทำหน้าที่เป็น Output ของวงจร ซึ่งจะแสดงออกมาในรูปการทำงานของอุปกณ์ต่างๆ หลังจากกำหนดสถานะเริ่มต้นของพอร์ต โปรแกรมจำนำค่าอินพุตข้างต้น จากส่วนต่างๆ เข้ามาสู่เงื่อนไขการตัดสินใจตามลำดับการทำงานในรูปที่ 3.21 และจะนำแผนผังดังกล่าวไปเขียนในรูปแบบภาษาซี ซึ่งโปรแกรมฟังก์ชั่นหลักจะแสดงได้ภาคผนวก และในที่นี้จะกำหนดความถี่ใช้งานของไมโครคอนโทรลเลอร์ (Microcontroller) เป็น 11.059 MHz

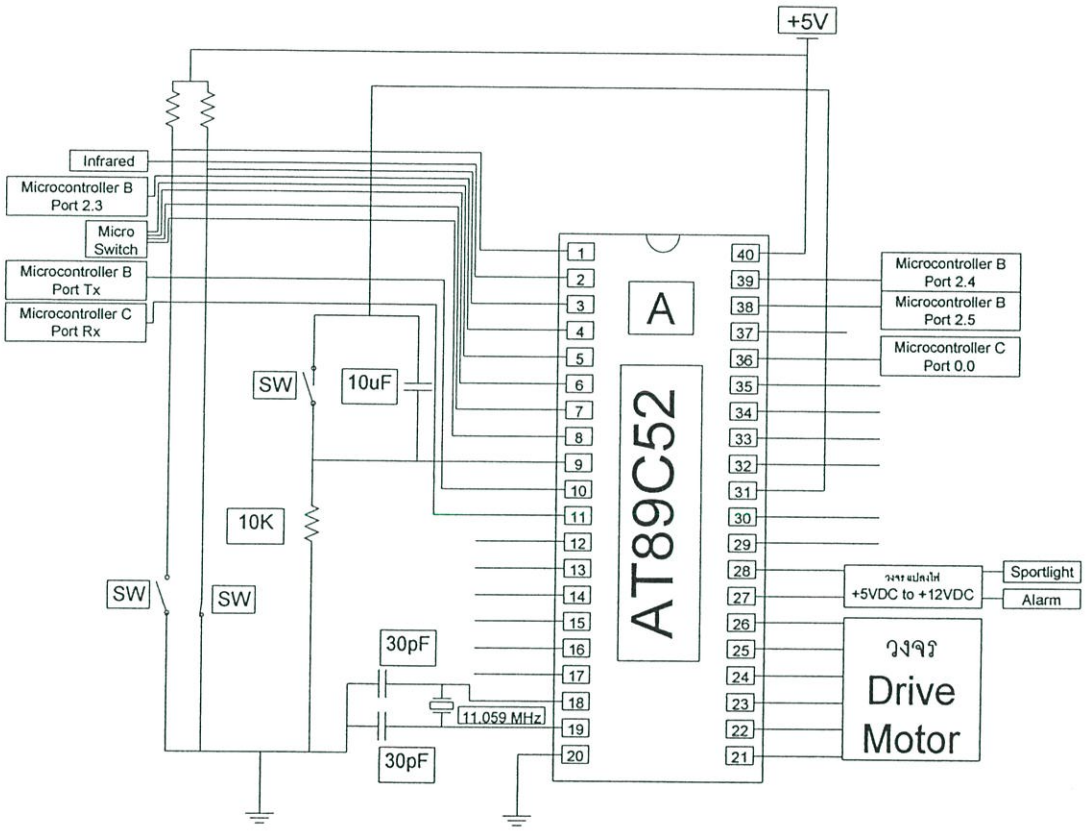
3.4.1.7 การทดสอบโค้ดภาษาซีผ่านโปรแกรมและวงจรที่ใช้ทดสอบ

ก่อนที่จะนำโค้ดภาษาซี (C Code) ไปทำการเบิร์น (Burn) ลงในไมโครคอนโทรลเลอร์ (Microcontroller) เบอร์ AT89C52 จะมีการนำตัวโค้ดภาษาซี (C Code) ไปทดสอบในโปรแกรมก่อน เพื่อเป็นการปรับแก้โปรแกรมภาษาซีให้ได้การทำงานตามที่ต้องการ โดยโปรแกรมที่ใช้ในการทดสอบคือ Proteus (ISIS Professional) และวงจรที่ใช้ทดสอบภายในโปรแกรมในรูปที่ 3.33 ซึ่งวงจรจะใช้สำหรับกำหนดและตรวจสอบค่าทางลอจิก (Logic) ให้ได้ตรงกับอินพุต-เอาท์พุต (Input-Output) ที่ต้องการ



รูปที่ 3.33 โปรแกรม Proteus ที่ใช้ทดสอบโค้ดภาษาซีของไมโครคอนโทรลเลอร์ตัวที่ 2

หลักจากการนำโค้ดภาษาซีดังกล่าวไปทดสอบการทำงาน เมื่อได้ผลการทำงานตามต้องการแล้ว ก็จะนำโค้ดภาษาซี (C Code) ไปเบิร์น (Burn) ลงตัวไมโครคอนโทรลเลอร์ (Microcontroller) AT89C52 เพื่อนำไปต่อกับวงจรไฟฟ้า และทดสอบใช้จริง โดยจะแสดงภาพวงจรในรูปที่ 3.34

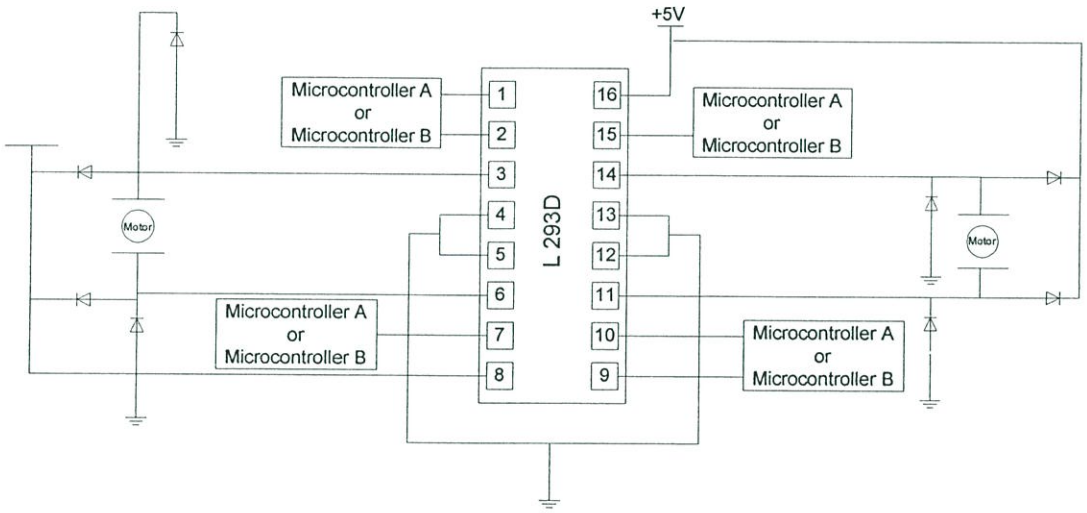


รูปที่ 3.34 วงจรอินพุต (Input) และเอาท์พุต (Output) ที่มีการควบคุมทำงานโดยไมโครคอนโทรลเลอร์ (Microcontroller) ตัวที่ 1

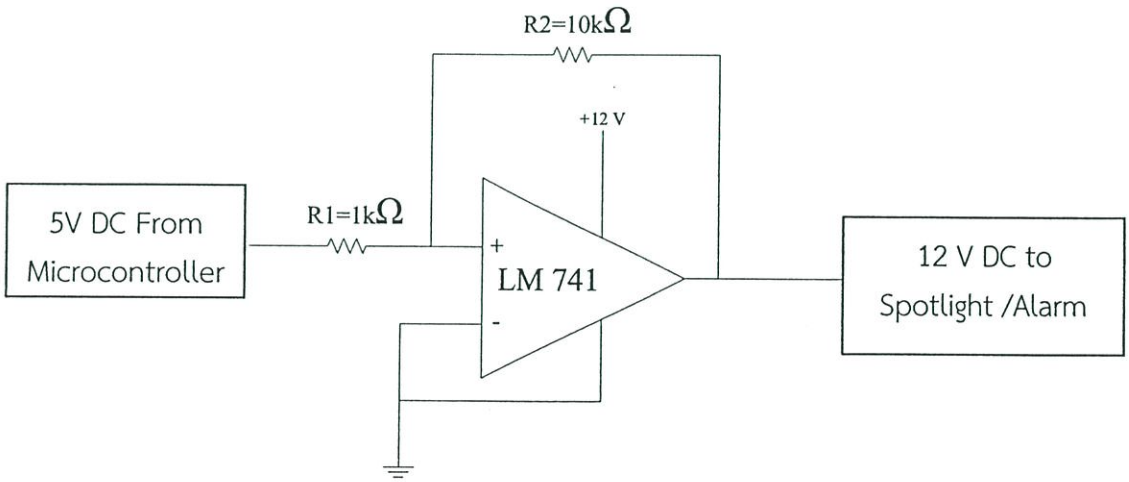
จากรูปที่ 3.34 วงจรฝั่งอินพุต (Input) จะใช้สวิทช์ (Switch) แบบสลับ 1 ตัว เป็นสวิทช์เลือกโหมดกลางวัน-กลางคืน โดยใช้พอร์ต 1.0 (Port1.0) , แบบปุ่มกด 1 ตัว เป็นสวิทช์สำหรับกดให้กรณีเกิดเหตุฉุกเฉินโดยใช้พอร์ต 1.1 (Port1.1) , รับค่าลอจิกจากไมโครคอนโทรลเลอร์ตัวที่ 2 โดยใช้พอร์ต 0.0, 0.1, 0.4 (Port0.0, 0.1, 0.4) , รับรหัสแอสกี้ (ASCII) จากพอร์ตอนุกรม (Serial Port) ของไมโครคอนโทรลเลอร์ตัวที่ 2 , ส่งรหัส ASCII ผ่านพอร์ตอนุกรมไปยังตัวที่ 3 และวงจรสำหรับรับแรงดันจากเซ็นเซอร์อินฟราเรด (Infrared Sensor) ใช้พอร์ต 1.2 (Port1.2) ซึ่งการตั้งค่าเริ่มต้นของพอร์ตอินพุต จะกำหนดค่าทางลอจิก (Logic) เป็น 1 ทุกพอร์ต และเมื่อสวิทช์ถูกสับหรือกด รวมทั้งรับค่าลอจิกจากเซ็นเซอร์อินฟราเรดและรับรหัส ASCII จากไมโครคอนโทรลเลอร์ตัว

อื่นๆ จะทำให้ค่าทางลอจิก (Logic) ของพอร์ตนั้นๆ เปลี่ยนไป ซึ่งจะไปเป็นตามเงื่อนไขของโค้ดภาษาซี (C Code) ตามข้างต้น โดยอินพุตจากเซ็นเซอร์อินฟราเรดจะใช้วงจรรีเลย์ เพื่อให้ได้การทำงานตามต้องการ

ในส่วนของวงจรด้านเอาท์พุตจะประกอบด้วยวงจรขับมอเตอร์ (Drive Motor) เพื่อใช้สำหรับเคลื่อนประตู (Front Door) และกำแพง (Wall) โดยจะแสดงดังรูปที่ 3.35 และวงจรแปลงแรงดัน 5 Volts ให้เป็น 12 Volts สำหรับเปิดสปอร์ตไลท์ (Spotlight) และเสียงกริ่ง (Alarm) จำลอง แสดงดังรูปที่ 3.36



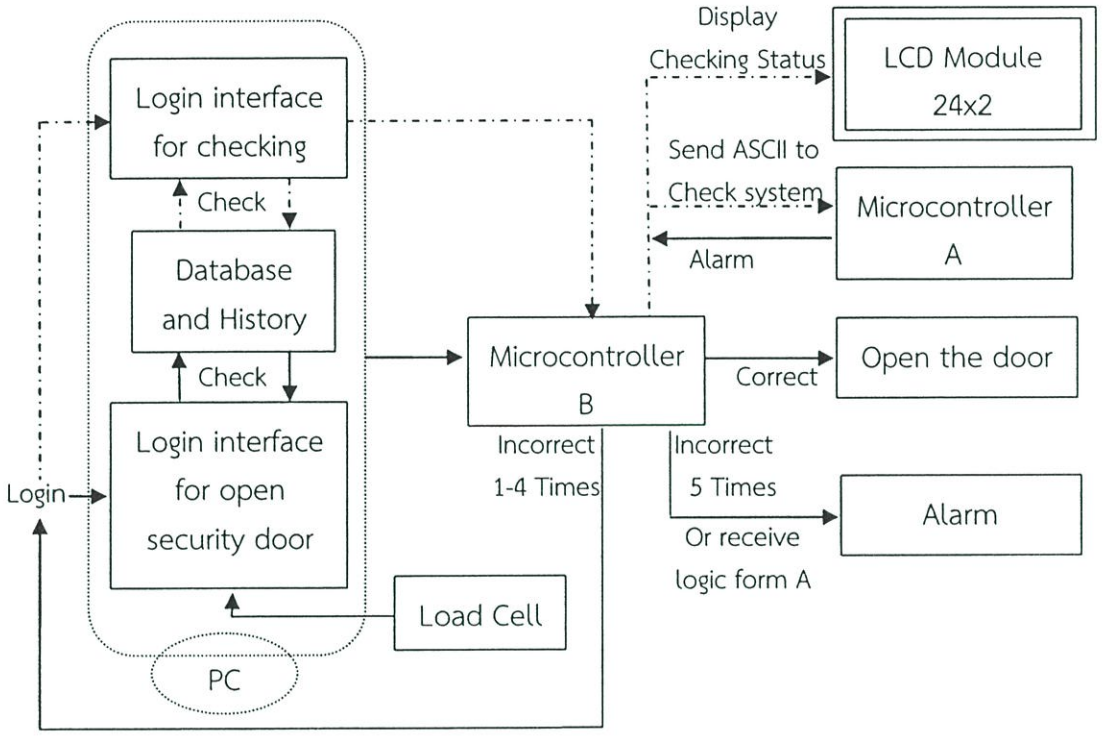
รูปที่ 3.35 วงจรขับมอเตอร์ดีซี (DC drive Motor circuit)



รูปที่ 3.36 วงจรแปลงแรงดันไฟดีซี 5 Volts to 12 Volts (Regulator 5v-DC to 12v-DC)

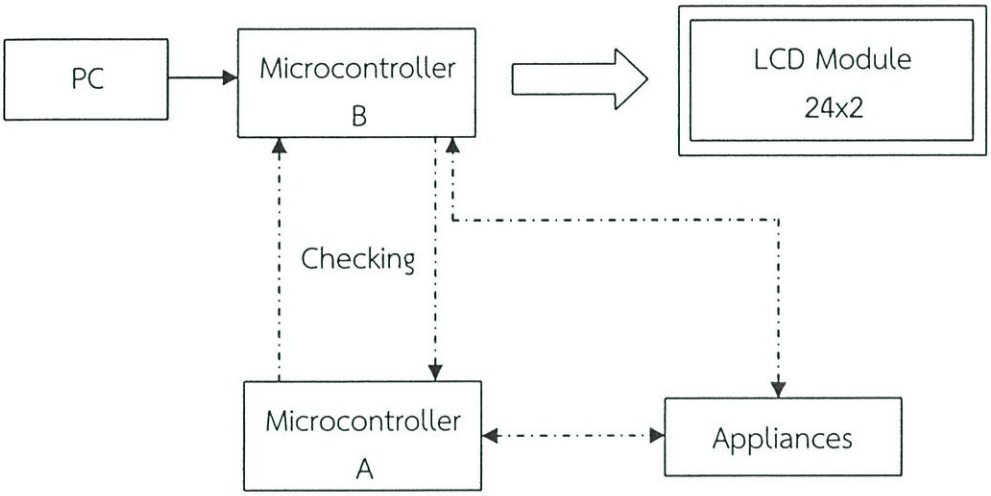
3.1.5 โปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยในไมโครคอนโทรเลอร์ตัวที่ 2

เป็นการควบคุมโดยใช้ไมโครคอนโทรลเลอร์ (Microcontroller) เช่นเดียวกับตัวที่ 1 โดยจะอธิบายบล็อกไดอะแกรม และแผนผังลำดับการทำงานได้ดังรูปที่ 3.37 และ 3.39 ตามลำดับ

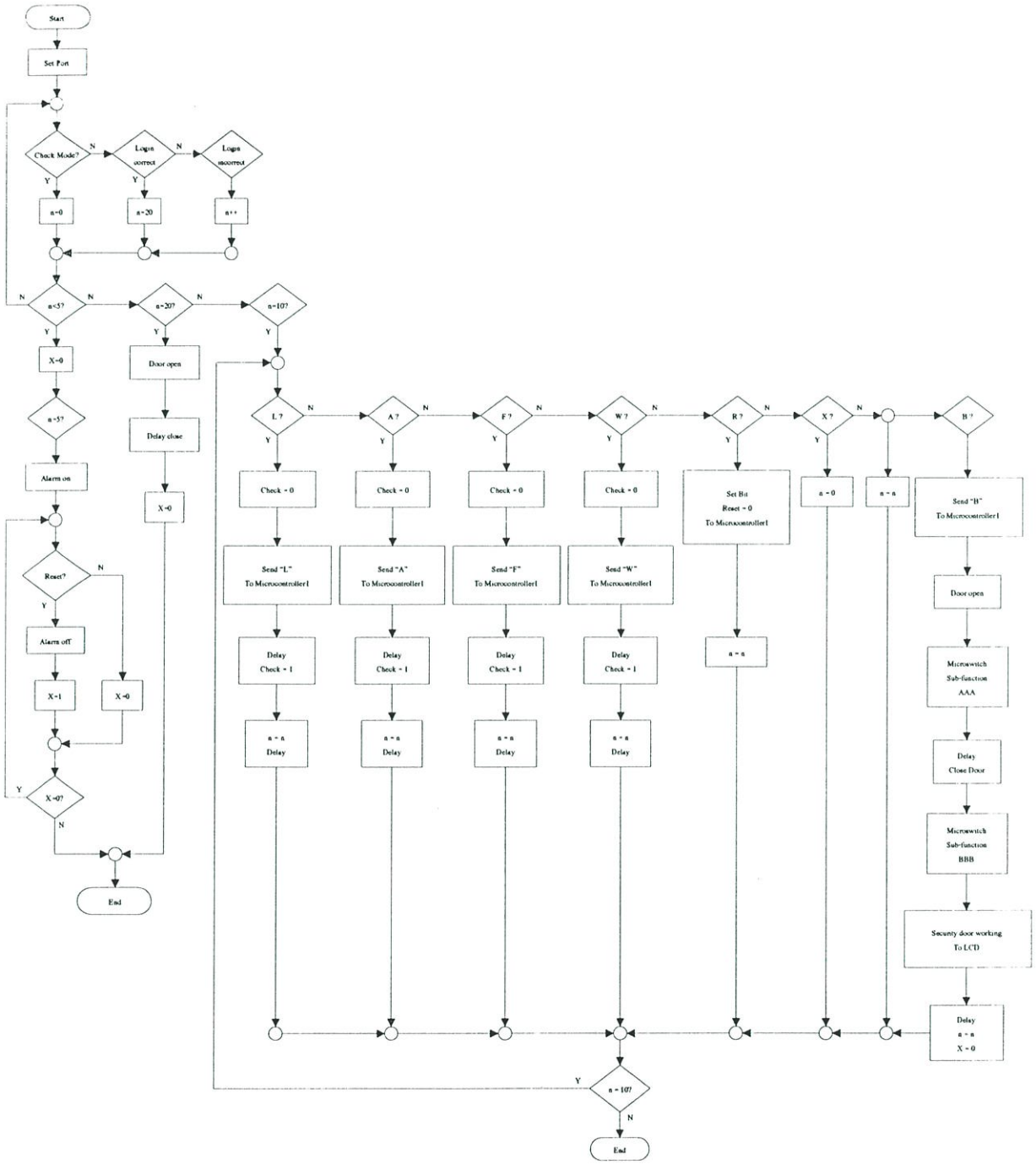


รูปที่ 3.37 บล็อกไดอะแกรมการทำงาน ในส่วนที่ 2

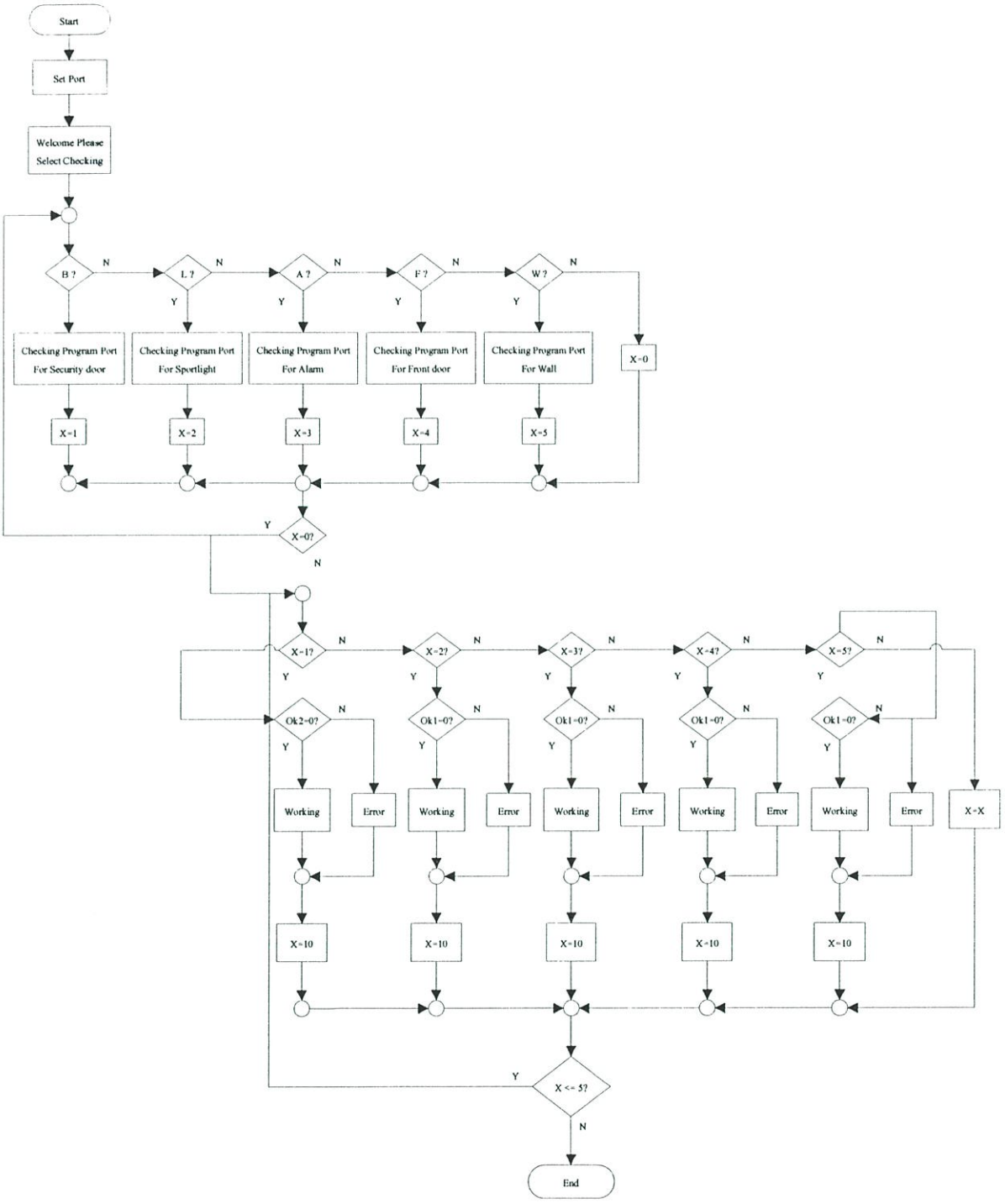
ส่วนการแสดงผลของการเช็คความพร้อมของอุปกรณ์จะถูกแสดงผ่านหน้าจอแสดงผล LCD โดยจะใช้ไมโครคอนโทรลเลอร์ตัวที่ 2 ในการประมวลผลโดยจะเป็นไปตามบล็อกไดอะแกรม ดังรูปที่ 3.38 และจะมีแผนผังการทำงาน (Flow Chart) ดังรูปที่ 3.40 โดยในที่นี้จะเลือกใช้โมดูล ขนาด 24 ตัวอักษร 2 บรรทัด (24x2)



รูปที่ 3.38 บล็อกไดอะแกรมการรับส่งค่าเพื่อแสดงผลทางจอ LCD



รูปที่ 3.39 แผนผังลำดับการทำงานของโปรแกรมป้องกันและแจ้งเตือนภัยในส่วนที่ 2



รูปที่ 3.40 แผนผังการทำงานของโปรแกรมแสดงผลจอ LCD

จากรูปที่ 3.37 และ 3.39 จะมีการรับค่าอินพุต (Input) จากเครื่องคอมพิวเตอร์ (Computer) ผ่าน RS-232 ซึ่งเป็นรหัสแอสกี้ (ASCII) เข้ามายังพอร์ตอนุกรม (Serial Port) ของไมโครคอนโทรลเลอร์ (Microcontroller) โดยโปรแกรมอินเตอร์เฟสจะแบ่งเป็น 2 ส่วนตามหัวข้อ 3.1 จากนั้นจะทำการตัดสินใจตามเงื่อนไขข้างต้น ซึ่งจะแบ่งเป็น 3 เงื่อนไขหลักคือ กรณีที่ผู้ใช้ต้องการตรวจสอบความพร้อมของอุปกรณ์ , ผู้ใช้มีการเข้ารหัส (Login) ถูกต้อง และผู้ใช้มีการเข้ารหัสผิดครบ 5 ครั้ง โดยมีการกำหนดรหัส ASCII ที่เข้ามายังพอร์ตอนุกรมเป็น "Y" ในกรณีผู้ใช้กรอกชื่อผู้ใช้ (Username) , รหัสผ่าน (Password) และมีน้ำหนักที่ถูกต้องตรงกับฐานข้อมูล (Database) ประตูนิรภัย (Security Door) จะสามารถเปิดได้ และ ในกรณีที่ไม่ถูกต้องจะเป็น "N" และประตูจะยังคงล็อค (Lock) อยู่ โดยหาก Login ไม่ถูกต้องติดต่อกัน 5 ครั้ง ระบบจะแจ้งเตือนโดยเสียงเตือนภัย (Alarm) ส่วนโหมดการเช็คความพร้อมอุปกรณ์จะใช้ตัวอักษร "C" ซึ่งเมื่อเข้าสู่เงื่อนไขนี้จะมีการเช็คอุปกรณ์เกิดขึ้นซึ่งจะอธิบายตามรูปที่ 3.38 และ 3.40 โดยจะเป็นไปแผงผันการทำงานต่อไป รวมถึงจะเก็บประวัติ (History) การเช็คอุปกรณ์ไว้ในฐานข้อมูลด้วยเมื่อมีการเรียกใช้งานในแต่ละครั้ง

โดยการเขียนโปรแกรมแสดงผลในส่วนของโมดูลจอแสดงผล LCD (LCD Module) จะเริ่มจากควบคุมโดยใช้ขา (Pin) ต่างๆ ดังต่อไปนี้

ขา RS (Register Select Pin) : เป็นขาที่ใช้เลือกรีจิสเตอร์ 2 ตัวได้แก่ รีจิสเตอร์สำหรับคำสั่ง (Command Register) และรีจิสเตอร์สำหรับข้อมูล (Data Register) โดยจะกำหนด RS = 0 เมื่อต้องการเลือกรีจิสเตอร์คำสั่ง และ 1 เมื่อต้องการเลือกรีจิสเตอร์ข้อมูล

ขา R/W (Read/Write Pin) : เป็นขาสำหรับให้ผู้ใช้เลือกว่าต้องการอ่านข้อมูลหรือเขียนข้อมูล โดยกำหนด R/W = 1 เมื่อต้องการอ่านข้อมูล และ 0 เมื่อต้องการเขียนข้อมูล

ขา E (Enable Pin) : เป็นขาอนุญาตให้เกิดการทำงาน โดยจะต้องกำหนดเป็นลอจิก 1 ตลอดเมื่อใช้งานโมดูล

ขา D0 - D7 (D0-D7 Pin) : ขาสำหรับใส่ข้อมูลแบบบัสจำนวน 8 Bit และขา D7 จะมีหน้าที่พิเศษเพิ่มเติมคือจะใช้เป็นขาสำหรับตรวจสอบว่าโมดูลอยู่ในลักษณะพร้อมที่จะรับข้อมูลหรือไม่ (BusyFlag) ระหว่างมีโมดูลเกิดการทำงาน โดยเมื่อถ้าใช้รีจิสเตอร์อ่านข้อมูลตรวจสอบที่ขา D7 แล้วมีลอจิกเท่ากับ 1 จะหมายถึง โมดูลยังอยู่ในสถานะไม่ว่าง ไม้พร้อมจะรับข้อมูลใดๆ แต่หากเท่ากับ 0 จะแสดงว่าโมดูลอยู่ในสถานะที่พร้อมจะรับข้อมูลแล้ว

โดยการเขียนโปรแกรมจะแบ่งเป็น 2 ลักษณะคือ ช่วงอ่านข้อมูลและช่วงเขียนข้อมูลของโมดูล ซึ่งจะอธิบายได้ดังต่อไปนี้

1) ช่วงสถานะการอ่านของโมดูล LCD

เมื่ออยู่ในช่วงสถานะการอ่าน ผู้เขียนจะกำหนดให้มีการอ่านข้อมูลขา D7 ของโมดูลเพื่อตรวจสอบสถานะของโมดูลว่าพร้อมจะรับข้อมูลหรือไม่ เนื่องจากโดยทั่วไปแล้วโมดูลจะมีเวลาที่ใช้ในการทำงานของแต่การสั่งงาน หากตั้งค่าโปรแกรมโดยไม่สนใจช่วงเหล่าเวลานี้ อาจทำให้โมดูลรับข้อมูลได้ไม่ครบถ้วนตามที่ต้องการ ซึ่งเมื่ออยู่ในช่วงนี้จะต้องกำหนดขา RS = 0 และ

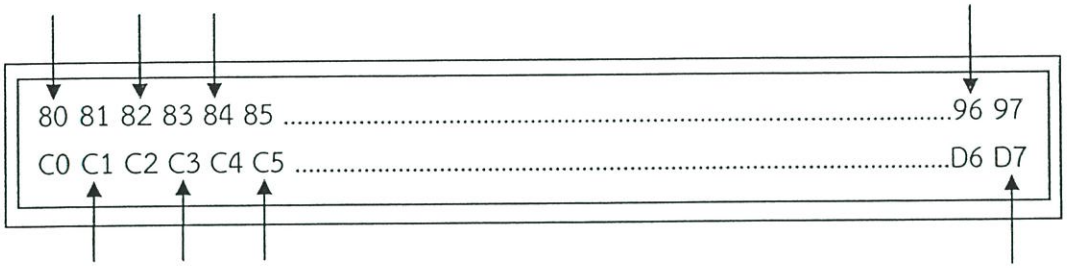
R/W = 1 ซึ่งการเลือกใช้วิธีการเช็คสถานะความพร้อมในการรับการทำงานของโมดูลด้วยวิธีนี้จะทำให้ประหยัดเวลาตั้งค่าดีเลย์ (Delay) ให้สอดคล้องกับเวลาที่โมดูลต้องการใช้ในแต่ละคำสั่งการทำงาน

2) ช่วงสถานะการเขียนของโมดูล LCD

ในช่วงการเขียนจะต้องเลือกรีจิสเตอร์ว่าต้องการเขียนในลักษณะใด ระหว่างคำสั่งกับข้อมูล โดยจะส่งทั้งคำสั่งและข้อมูลในลักษณะแบบบัสจำนวน 8 บิตผ่านขา D0-D7 ซึ่งการเขียนในลักษณะคำสั่งจะเลือกใช้รีจิสเตอร์คำสั่ง (Command Register) และการกำหนดลักษณะบัสจะเป็นไปตามตารางที่ 3.3 และรูปที่ 3.41 โดยจะกำหนดให้ขา RS = 0 และ R/W =0

The 8-bit data bus (Hex)	Command to LCD Instruction
1	Clear display screen
2	Return home
4	Decrement cursor (shift to left)
6	Decrement cursor (shift to right)
5	Shift display right
7	Shift display left
8	Display off , cursor off
A	Display off , cursor on
C	Display on , cursor off
E	Display on , cursor blinking
F	Display on , cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 1nd line
38	2 lines and 5x7 matrix

ตารางที่ 3.3 การกำหนดบัส 8 บิต สำหรับรีจิสเตอร์คำสั่ง (Command Register)



The 8-bit data bus (Hex) to force display position of 24x2's LCD module

รูปที่ 3.41 การกำหนดบัส 8 บิต สำหรับรีจิสเตอร์คำสั่ง (Command Register) ในการเลือกตำแหน่งแสดงผลของจอ LCD

ส่วนการเขียนในลักษณะข้อมูลจะป้อนข้อมูลตามตารางรหัสแอสกี้ (ASCII) ให้กับโมดูล LCD เช่น ป้อนบิต 00100001, โมดูลจะแสดงผลเป็นตัวอักษร "A" เป็นต้น โดยจะกำหนดให้ขา RS = 1 และ R/W = 0

3.1.5.1 ส่วนกำหนดตัวแปร (Variable) และฟังก์ชั่น (Function) ย่อย

จะมีการประกาศฟังก์ชั่นชั้นย่อยดีเลย์ (Delay Function) ฟังก์ชั่นกำหนดอัตราการรับ-ส่งข้อมูล (Baud Rate Function) , ฟังก์ชั่นของไมโครสวิทช์ , ฟังก์ชั่นรับส่งค่าเพื่อปิดอุปกรณ์ระหว่างการทำงาน เช่นเดียวกับโปรแกรมในไมโครคอนโทรลเลอร์ตัวที่ 1 รวมถึงฟังก์ชั่นในการควบคุมการแสดงผลผ่านหน้าจอ LCD แสดงได้ดังรูปที่ 3.42 ถึงรูปที่ 3.46

```
0001 #include <reg52.h>
0002 #include <stdio.h>
0003 #include <string.h>
0004
0005 char RxBuff ;                    /* SBUF */
0006 sbit misw1fw = P1^0 ;            /* stop forward  */
0007 sbit misw2bw = P1^1 ;            /* stop backward */
0008 sbit ChkL = P1^4 ;               /* say to microA to check Spotlight  */
0009 sbit ChkA = P1^5 ;               /* say to microA to check Alarm      */
0010 sbit ChkF = P1^6 ;               /* say to microA to check Front Door */
0011 sbit ChkW = P1^7 ;               /* say to microA to check Wall       */
0012 sbit alarm = P2^3 ;              /* alarm to microA */
0013 sbit reset = P2^4 ;              /* reset to microA */
0014 sbit ChkS = P2^5 ;               /* into check mode to microA */
0015 sbit okay = P2^6 ;
0016
0017 sbit RS = P1^4 ;                 /* Register Select pin*/
0018 sbit RW = P1^5 ;                 /* Read / Write pin */
0019 sbit ENA = P1^6 ;                /* Enable pin */
0020
0021 sbit check = P1^3 ;               /* Check mode */
0022
0023 sbit D0 = P0^0;            /* D0 pin */
0024 sbit D1 = P0^1;            /* D1 pin */
0025 sbit D2 = P0^2;            /* D2 pin */
```

รูปที่ 3.42 การกำหนดตัวแปรและฟังก์ชั่นย่อยภายในไมโครคอนโทรลเลอร์ตัวที่ 2 (1)

```
sbit D3 = P0^3;              /* D3 pin */
sbit D4 = P0^4;              /* D4 pin */
sbit D5 = P0^5;              /* D5 pin */
sbit D6 = P0^6;              /* D6 pin */
sbit busy = P0^7;               /* D7 pin - busy pin */
int x ;
int n ;

void uart_init(void) ;              /* Baud rate 9600 Mbps */
void delay(unsigned char loop) ;    /*delay function*/
void AAA() ;                        /* stop forward security door */
void BBB() ;                        /* stop backward security door */
void CCC() ;                        /* recieve "R"(reset) form PC
                                       and say to microA to close alarm */
void cmd(unsigned char CMD);     /* write - command */
void dsp(unsigned char alp);     /* write - data */
void lcdready() ;                /* Check busy flag */
void stand() ;                   /* wait for checking */
void complete();                 /* program's working */
void error() ;                   /* program error */
void welcome();                  /* message welcome */
void logincheck() ;              /* message logincheck */
void loginreset() ;           /* message loginreset */
```

รูปที่ 3.43 การกำหนดตัวแปรและฟังก์ชั่นย่อยภายในไมโครคอนโทรลเลอร์ตัวที่ 2 (2)

โดยฟังก์ชั่นของไมโครสวิทช์จะมีการสั่งให้มอเตอร์หยุดหมุนแบบทันที (Fast Stop) เช่นเดียวกับไมโครคอนโทรลเลอร์ตัวที่ 1 จะแสดงในรูปที่ 3.44 และ 3.45 ส่วนฟังก์ชั่นการปิดอุปกรณ์จะรับค่าจากคอมพิวเตอร์เมื่อผู้ใช้กดปุ่มคำว่า RESET จากนั้นจะส่งค่าลอจิกให้กับไมโครคอนโทรลเลอร์ตัวที่ 1 เพื่อปิดอุปกรณ์ ซึ่งในที่นี้คือฟังก์ชั่น CCC ของไมโครคอนโทรลเลอร์ทั้ง 2 ตัวนั่นเอง โดยจะแสดงในรูปที่ 3.46

```
void AAA()                                  /* stop forward security door */
{
        do
          {
            if(misw1fw == 0)            /* microswitch is pushed */
              {
                P2 = 0x7F ;             /* fast stop */
                x = 1 ;
              }
            else
              {
                x = 0 ;
              }
          }
        while(x==0) ;
        return ;
}
```

รูปที่ 3.44 ฟังก์ชั่นย่อยไมโครสวิทช์ AAA ในไมโครคอนโทรลเลอร์ตัวที่ 2

```
245 void BBB()                                   /* stop backward security door */
246 {
247             do
248                 {
249                     if(misw2bw == 0)          /* microswitch is pushed */
250                         {
251                             P2 = 0x7F ;       /* fast stop */
252                             x = 1 ;
253                         }
254                     else
255                         {
256                             x = 0 ;
257                         }
258                 }
259             while(x==0);
260             return ;
261 }
```

รูปที่ 3.45 ฟังก์ชั่นย่อยไมโครสวิทช์ BBB ในไมโครคอนโทรลเลอร์ตัวที่ 2

```
263 void CCC()
264 {
265                     do
266                     {
267                         if(RI==1)
268                         {
269                             RxBuff = SBUF ;
270                             RI = 0 ;
271                             if(RxBuff == 0x52)          /* RESET - R */
272                                 {
273                                     if(TI==1)           /* send "R" to micro1 */
274                                     {
275                                         printf("R") ;
276                                     }
277                                     P2 = 0x68 ;         /* Alarm off */
278                                     delay(10) ;
279                                     P2 = 0x78 ;         /* set bit 0 */
280                                     x = 1 ;
281                                 }
282                             else
283                                 {
284                                     x = 0 ;
285                                 }
286                         }
287                     }
288                     while(x==0);
289                         return ;
```

รูปที่ 3.46 ฟังก์ชั่นย่อย CCC ในการปิดอุปกรณ์ของไมโครคอนโทรลเลอร์ตัวที่ 2

3.1.5.2 ฟังก์ชั่นย่อยการอ่านข้อมูลตรวจสอบ Busy flag

การอ่านข้อมูลจะตรวจสอบจากขา D7 (D7 Pin) ซึ่งหากค่าทางลอจิกเป็น 1 โปรแกรมจะรอจนกว่าค่าลอจิกจะเปลี่ยนเป็น 0 จึงจะออกฟังก์ชั่นได้

```
135  void lcdready()
136  {
137      busy = 1 ;
138      RS = 0 ;
139      RW = 1 ;
140      while(busy==1)
141      {
142          ENA = 0 ;
143          delay(1);
144          ENA =1 ;
145      }
146
147  }
```

รูปที่ 3.47 ฟังก์ชั่นย่อยการอ่านข้อมูลตรวจสอบ Busy flag

3.1.5.3 ฟังก์ชั่นย่อยการเขียนโดยใช้รีจิสเตอร์คำสั่ง (Command Register)

การเขียนข้อมูลคำสั่งจะกำหนดตามตารางที่ 3.3 และ รูปที่ 3.41 โดยจะมีการเรียกใช้ฟังก์ชั่น Busyflag เพื่อตรวจสอบสถานะของโมดูล LCD (LCD Module) ก่อนจะทำการเขียน โดยโค้ดภาษาซีจะแสดงในรูปที่ 3.48

```
102  void cmd(unsigned char CMD)
103  {
104      lcdready() ;
105      P2 = CMD;
106      RS = 0 ;
107      RW = 0 ;
108      ENA = 1 ;
109      delay(1) ;
110      ENA = 0 ;
111      return ;
112  }
113
```

รูปที่ 3.48 ฟังก์ชั่นย่อยการเขียนโดยใช้รีจิสเตอร์คำสั่ง (Command Register)

3.1.5.4 ฟังก์ชั่นย่อยการเขียนโดยใช้รีจิสเตอร์ข้อมูล (Data Register)

```
114  void dsp(unsigned char alp)
115  {
116      lcdready() ;
117      P2 = alp ;
118      RS = 1;
119      RW = 0 ;
120      ENA = 1 ;
121      delay(1) ;
122      ENA = 0 ;
123      return ;
124  }
```

รูปที่ 3.49 ฟังก์ชั่นย่อยการเขียนโดยใช้รีจิสเตอร์ข้อมูล (Data Regeister)

3.1.5.5 ฟังก์ชั่นย่อยรอการเช็คความพร้อมของอุปกรณ์

เมื่อไมโครคอนโทรลเลอร์ที่ควบคุมโมดูล LCD (LCD Module) ได้รับสัญญาณจากคอมพิวเตอร์แล้ว จะมีการแสดงผลตามชนิดของอุปกรณ์ และมีการตั้งค่าดีเลย์ (Delay) หน่วงเวลาให้เหมาะสมกับการทำงานของอุปกรณ์ เพื่อรอรับค่าผลลัพธ์ที่ได้จากการเช็คอุปกรณ์ ตัวอย่างเช่น ผู้ใช้ (User) ทำการเช็คอุปกรณ์สปอร์ตไลท์ (Spotlight) ผ่านโปรแกรมอินเตอร์เฟส (Interface Program) ไมโครคอนโทรลเลอร์ตัวที่ 2 จะส่งสัญญาณตามอุปกรณ์ที่ได้กดเช็คไปยังไมโครคอนโทรลเลอร์ตัวที่ 1 และในระหว่างนั้นจะมีการแสดงผลในหน้าจอ LCD ว่า Checking Program-Port for Spotlight จากนั้นจะรอรับค่าผลลัพธ์ที่ได้จากไมโครคอนโทรลเลอร์ตัวที่ 1 โดยจะแสดงในฟังก์ชั่นย่อยอื่นๆ ต่อไป จะแสดงตัวอย่างขอฟังก์ชั่นนี้ดังรูปที่ 3.50

```
149 void stand()
150 {
151
152     x = 0 ;
153     do
154     {
155     if(RI==1)
156     {
157         RxBuff = SBUF ;
158         RI = 0 ;
159         if(RxBuff == 0x42)          /* B */
160         {                           /* Checking Program - port
161                                           for Security Door    */
162             cmd(0x38) ;         /* 2 lines 5x7 matrix */
163             cmd(0x0E) ;         /* display on , cursor blinking */
164             cmd(0x01) ;         /* clear display screen */
165             cmd(0x80) ;         /* force cursor to begening of 1st line */
166             dsp('C') ;
167             dsp('h') ;
168             dsp('e') ;
169             dsp('c') ;
170             dsp('k') ;
171             dsp('i') ;
172             dsp('n') ;
173             dsp('g') ;
174             dsp(' ') ;
175             dsp('P') ;
176             dsp('r') ;
177             dsp('o') ;
178             dsp('g') ;
179             dsp('r') ;
180             dsp('a') ;
181             dsp('m') ;
182             dsp(' ') ;
183             dsp('-') ;
184             dsp(' ') ;
185             dsp('P') ;
186             dsp('o') ;
187             dsp('r') ;
188             dsp('t') ;
189             cmd(0xC3) ;         /* shift display to position 4 line2 */
190             dsp('f') ;
191             dsp('o') ;
192             dsp('r') ;
193             dsp(' ') ;
194             dsp('S') ;
195             dsp('e') ;
196             dsp('c') ;
197             dsp('u') ;
198             dsp('r') ;
199             dsp('i') ;
200             dsp('t') ;
201             dsp('y') ;
202             dsp(' ') ;
```

รูปที่ 3.50 ตัวอย่างฟังก์ชั่นย่อยรอการเช็คความพร้อมของอุปกรณ์

3.1.5.6 ฟังก์ชั่นย่อยรับค่าผลลัพธ์ของการเช็ค

เมื่อไมโครคอนโทรลเลอร์ตัวที่ 1 (Microcontroller A) มีการส่งผลลัพธ์เป็นลอจิก (Logic) ทางพอร์ต 1.4 (P1.4) หรือพอร์ต 1.5 (P1.5) กลับมาแล้ว จะมีการแสดงผลหากลอจิกนั้นๆ เป็น 1 แสดงว่าการทำงานของอุปกรณ์มีความผิดพลาดเกิดขึ้น แสดงตัวอย่างดังรูปที่ 3.51 และรูปที่ 3.52 ส่วนถ้าลอจิกเป็น 0 แสดงว่าการทำงานของอุปกรณ์ยังคงใช้งานได้ แสดงตัวอย่างดังรูปที่ 3.53 และ 3.54

```
414 void complete()
415 {
416     do
417     {
418     if(x==1)
419     {
420         if(ok2 == 0)          /* microB send okay */
421         {
422             cmd(0x38) ;       /* 2 lines 5x7 matrix */
423             cmd(0x0E) ;       /* display on , cursor blinking */
424             cmd(0x01) ;       /* clear display screen */
425             cmd(0x84) ;       /* shift display to position 5 line 1 */
426             dsp('P') ;
427             dsp('r') ;
428             dsp('o') ;
429             dsp('g') ;
430             dsp('r') ;
431             dsp('a') ;
432             dsp('m') ;
433             dsp(' ') ;
434             dsp('S') ;
435             dsp('e') ;
436             dsp('c') ;
437             dsp('u') ;
438             dsp('r') ;
439             dsp('i') ;
440             dsp('t') ;
440              dsp('t') ;
441              dsp('y') ;
442              cmd(0xC3) ;
443              dsp('D') ;
444              dsp('o') ;
445              dsp('o') ;
446              dsp('r') ;
447              dsp(' ') ;
448              dsp('S') ;
449              dsp('t') ;
450              dsp('i') ;
451              dsp('l') ;
```

รูปที่ 3.51 ตัวอย่างฟังก์ชั่นย่อยรับค่าผลลัพธ์จากการเช็คเมื่ออุปกรณ์ยังใช้งานได้(1)

```
452             dsp('l') ;
453             dsp(' ') ;
454             dsp('W') ;
455             dsp('o') ;
456             dsp('r') ;
457             dsp('k') ;
458             dsp('i') ;
459             dsp('n') ;
460             dsp('g') ;
461
462             delay(500) ;
463             x = 10 ;        /* exit loop */
464             return ;
465         }
```

รูปที่ 3.52 ตัวอย่างฟังก์ชั่นย่อยรับค่าผลลัพธ์จากการเช็คเมื่ออุปกรณ์ยังใช้งานได้(2)

```
677 void error()
678 {
679             cmd(0x01) ;
680             cmd(0x82) ;
681             dsp('P') ;
682             dsp('r') ;
683             dsp('o') ;
684             dsp('g') ;
685             dsp('r') ;
686             dsp('a') ;
687             dsp('m') ;
688             dsp('-') ;
689             dsp('s') ;
690             dsp(' ') ;
691             dsp('s') ;
692             dsp('y') ;
693             dsp('s') ;
694             dsp('t') ;
695             dsp('e') ;
696             dsp('m') ;
697             dsp(' ') ;
698             dsp('h') ;
699             dsp('a') ;
700             dsp('v') ;
701             dsp('e') ;
```

รูปที่ 3.53 ฟังก์ชั่นย่อยรับค่าผลลัพธ์จากการเช็คเมื่ออุปกรณ์ใช้งานไม่ได้(1)

```
702             cmd(0xC2) ;
703             dsp('e') ;
704             dsp('r') ;
705             dsp('r') ;
706             dsp('o') ;
707             dsp('r') ;
708             dsp(',') ;
709             dsp('P') ;
710             dsp('l') ;
711             dsp('e') ;
712             dsp('a') ;
713             dsp('s') ;
714             dsp('e') ;
715             dsp(' ') ;
716             dsp('r') ;
717             dsp('e') ;
718             dsp('c') ;
719             dsp('h') ;
720             dsp('e') ;
721             dsp('c') ;
722             dsp('k') ;
723
724             x = 10 ;
725             delay(500) ;
726             return ;
```

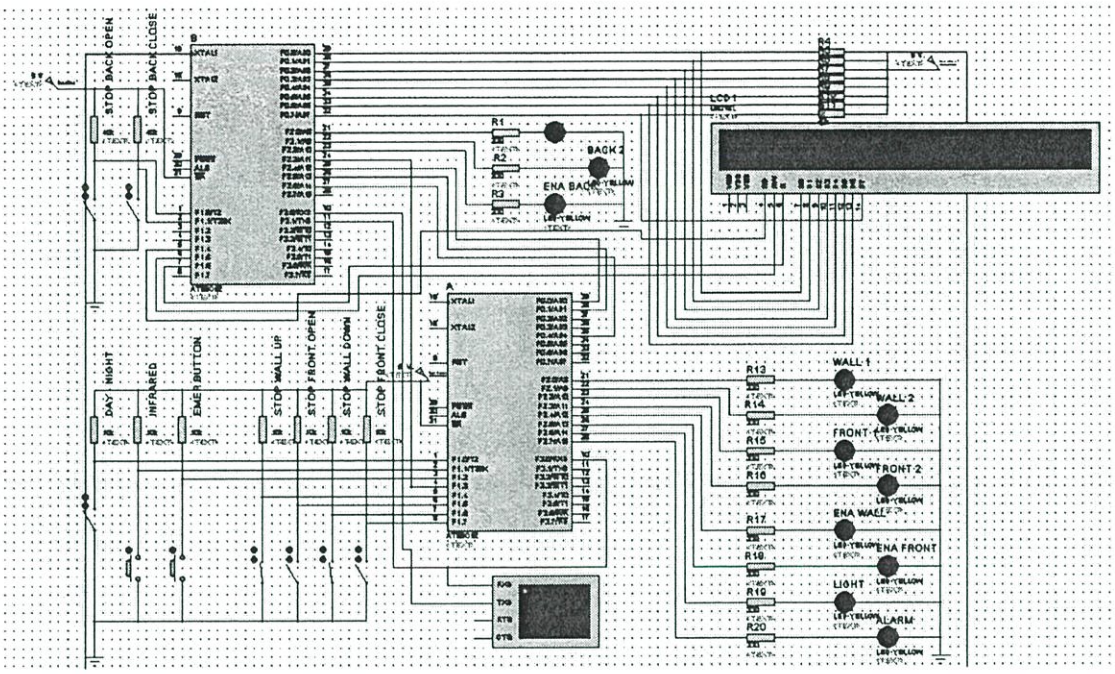
รูปที่ 3.54 ฟังก์ชั่นย่อยรับค่าผลลัพธ์จากการเช็คเมื่ออุปกรณ์ใช้งานไม่ได้ (2)

3.1.5.7 ฟังก์ชั่นหลัก (Main function)

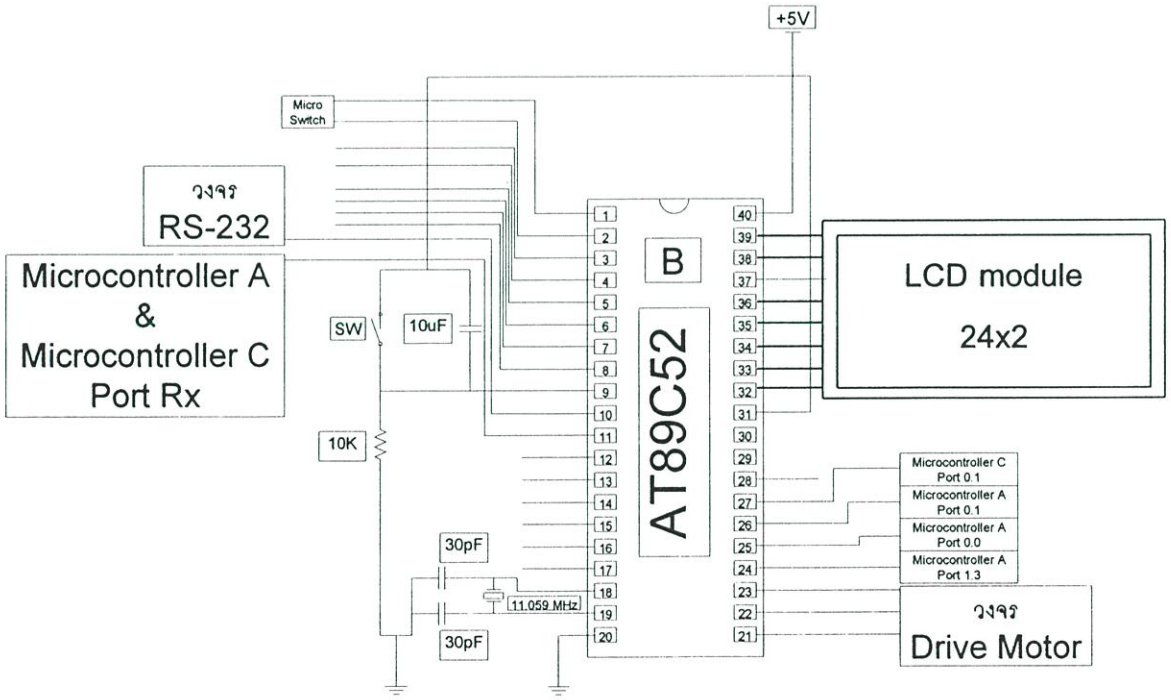
ฟังก์ชั่นหลักจะเริ่มการทำงานโดยแสดงข้อความต้อนรับขึ้น จากนั้นจะรอรับค่าจากคอมพิวเตอร์ว่าผู้ใช้ต้องการใช้งานส่วนใด ซึ่งจะแบ่งเป็นส่วนที่ล็อกอินจากเพื่อเปิดประตูห้องนิรภัย และส่วนล็อกอินเพื่อเข้าเช็คอุปกรณ์ตามที่กล่าวมานั่นเอง โดยจะแสดงโค้ดภาษาซีดังกล่าวในภาคผนวก

3.1.5.8 การทดสอบโค้ดภาษาซีผ่านโปรแกรมและวงจรที่ใช้ทดสอบ

การทดสอบจะใช้วงจรจากไมโครคอนโทรลเลอร์ตัวที่ 1 ยกมาต่อร่วมด้วยซึ่งจะแสดงวงจรที่ใช้ทดสอบโปรแกรมภาษาซีในโปรแกรม Proteus ได้ดังรูปที่ 3.55 และแสดงวงจรที่ใช้งานจริงดังรูปที่ 3.56



รูปที่ 3.55 โปรแกรม Proteus ที่ใช้ทดสอบโค้ดภาษาซีของไมโครคอนโทรลเลอร์ตัวที่ 2



รูปที่ 3.56 วงจรอินพุต (Input) และเอาท์พุต (Output) ที่มีการควบคุมทำงานโดย ไมโครคอนโทรลเลอร์ (Microcontroller) ตัวที่ 2

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 อุปกรณ์พื้นฐาน

1) อุปกรณ์ไฟฟ้าพื้นฐาน (Resistor , Capacitor , IC , Switch)
2) โมดูล LCD (LCD Module)
3) วงจรรับอินฟราเรด (Infrared Receiver)
4) วงจรส่งอินฟราเรด (Infrared Transmitter)
5) วงจรรับค่าทางน้ำหนัก (Load-cell)
6) Microcontroller เบอร์ AT89C52
7) Broad burn MCS-51 (ET-AFP V1.0)
8) Battery 12 volt
9) โมเดลจำลองการทำงานของระบบ (Simulation Model)

3.2.2 โปรแกรมที่ใช้ในการทดลอง

1) Keil
2) Proteus (ISIS 7 Professional)
3) Microsoft Visual Studio
4) Microsoft Office Visio
5) Microsoft Office Word
6) Math Type

3.3 การจัดเก็บผลการทดลอง

3.3.1 โปรแกรมอินเตอร์เฟส (Interface) สำหรับผู้ใช้ (User) เพื่อใช้ในการเข้ารหัสเปิดประตู และการเช็คอุปกรณ์

โปรแกรมอินเตอร์เฟส (Interface) หลังจากเขียนโค้ดภาษาซี (C Code) แล้วจะทำการทดสอบโปรแกรมผ่าน Microsoft Visual Studio ซึ่งจะแสดงถึงหน้าอินเตอร์เฟสของการระบุชื่อผู้ใช้ (Username) , รหัสผ่าน (Password) และน้ำหนัก (Weight) ของผู้ใช้ (User) รวมถึงการแก้ไข - เปลี่ยนแปลงส่วนของฐานข้อมูล (Database) , ประวัติการเข้าใช้ (History) และจะจัดเก็บผลของการส่งรหัสแอสกี้ผ่านหัวเชื่อมต่อ RS232 และผ่านวงจร IC MAX232

3.3.2 วงจรเซ็นเซอร์อินฟราเรด (Infrared Sensor)

วงจรอินฟราเรดจะมีกาทดสอบโดยตรวจสอบการใช้งานของวงจรหลังจากทำการสร้างวงจรตามแบบข้างต้น ซึ่งเมื่อมีสิ่งกีดขวางมากั้นระหว่างภาคส่งอินฟราเรด (Infrared Transmitter) และภาครับอินฟราเรด (Infrared Receiver) โดยจะมีการวัด

3.3.3 โปรแกรมควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยในไมโครคอนโทรลเลอรร์ ตัวที่ 1 , ตัวที่ 2

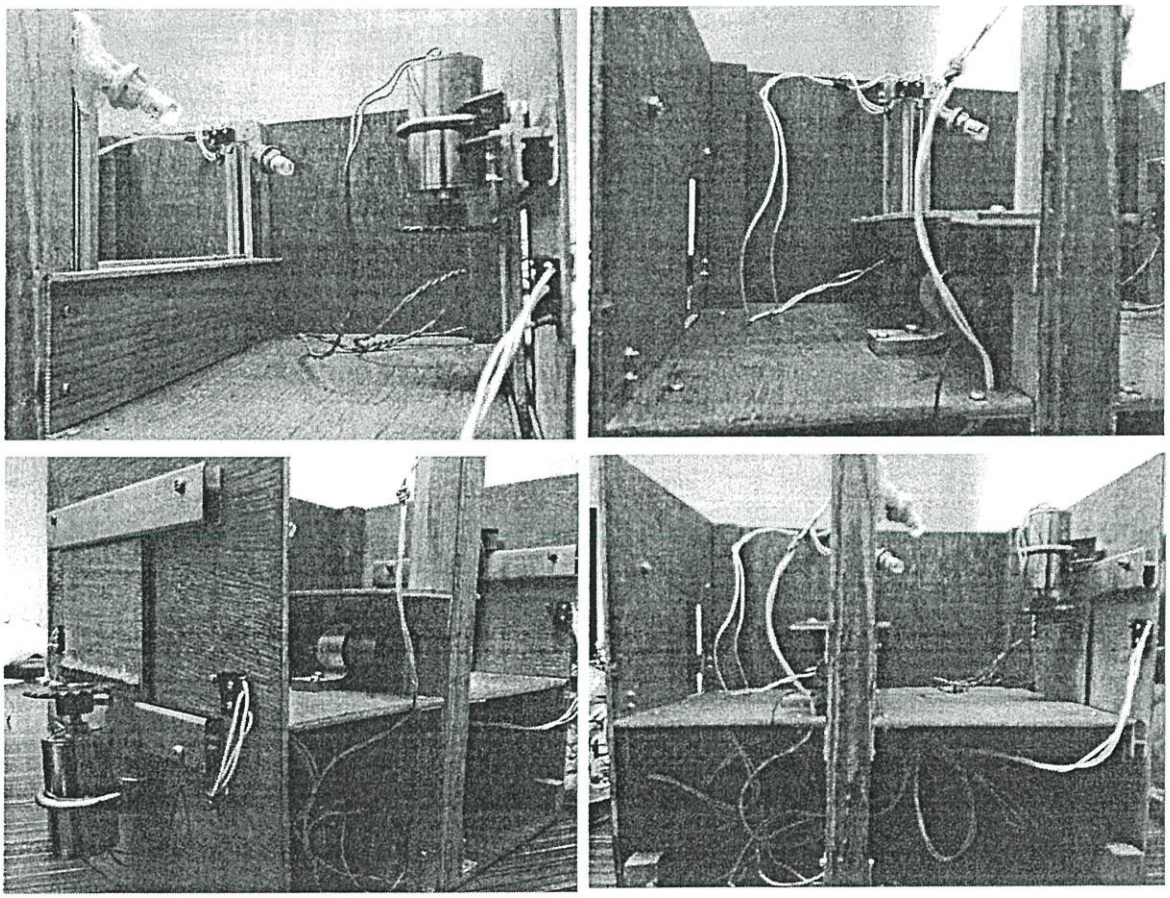
การทดสอบโปรแกรมควบคุมนี้ หลังจากเขียนโค้ดภาษาซี (C Code) แล้วจะทำการทดสอบโปรแกรมผ่านโปรแกรม Proteus เพื่อแก้ไขและเปลี่ยนแปลงโค้ดให้เป็นไปตรงกับการทำงานของอุปกรณ์ จากนั้นจะเบิร์น (Burn) โค้ดดังกล่าวลงในไมโครคอนโทรลเลอร์ และต่ออุปกรณ์ไฟฟ้าต่างๆ เพื่อทดสอบการใช้งานจริง โดยในที่นี้จะมีการออกแบบโมเดลจำลองห้องที่ใช้โดยจะติดตั้งอุปกรณ์จำลองต่างๆ ขึ้นเสมือนที่ออกแบบไว้เพื่อลองทดสอบการใช้งานจริง

บทที่ 4

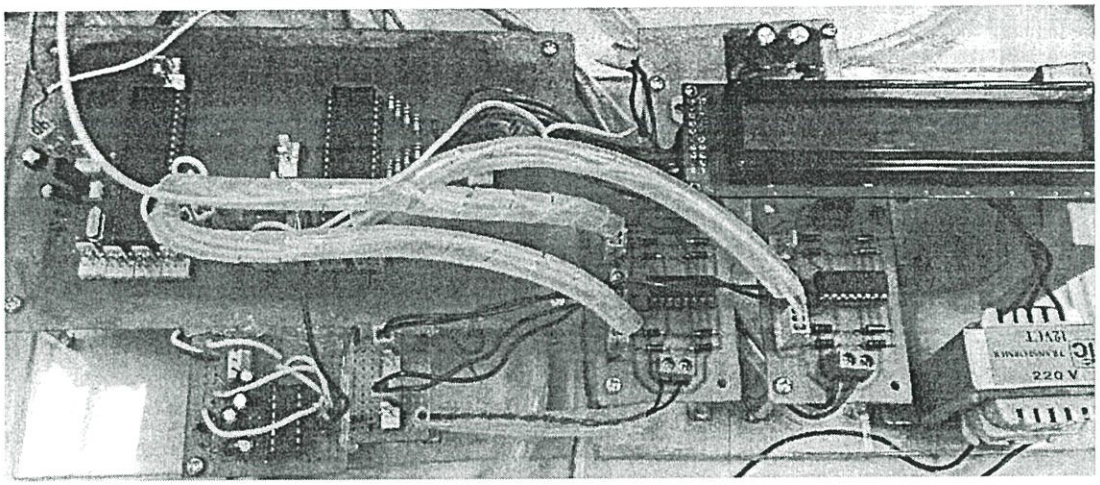
ผลการทดลอง

4.1 วงจรและแบบจำลองที่ใช้ในการทดลอง

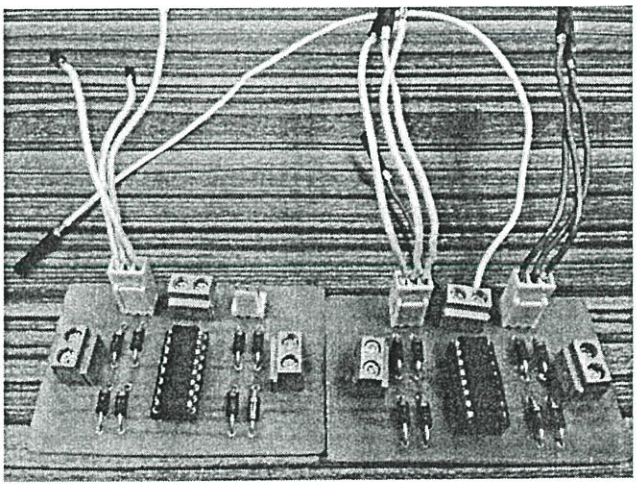
ในการทดลองจะใช้การทดสอบผ่านโปรแกรม Proteus ก่อน ทั้งในส่วนโปรแกรมป้องกันและแจ้งเตือนภัยในไมโครคอนโทรลเลอร์ตัวที่ 1, 2 และวงจรเซ็นเซอร์อินฟราเรด (Infrared Sensor) เพื่อทดสอบโค้ดภาษาซีจากนั้นจะสร้างวงจรขึ้นเพื่อทดสอบการใช้งานจริง โดยจะใช้โมเดลที่สร้างขึ้นในการทดสอบการทำงานของอุปกรณ์ต่างๆ



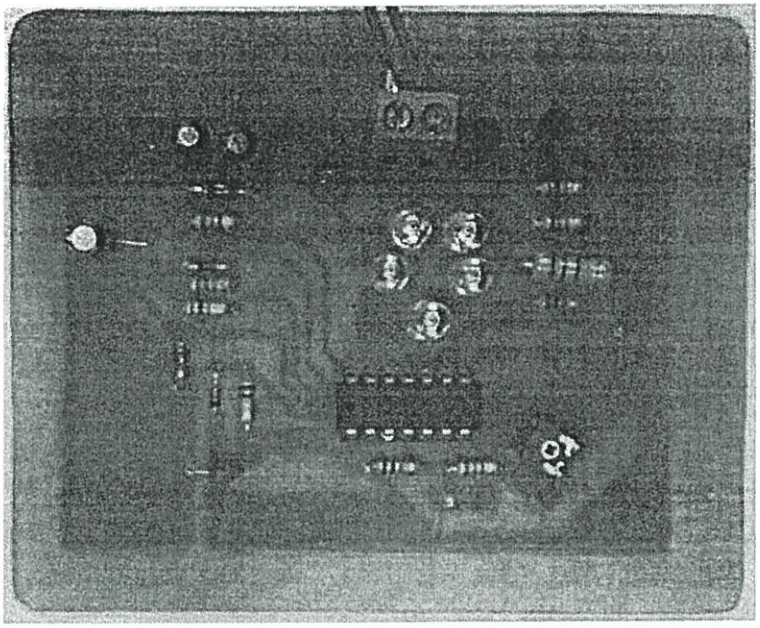
รูปที่ 4.1 แบบจำลองที่ใช้ในการทดสอบ



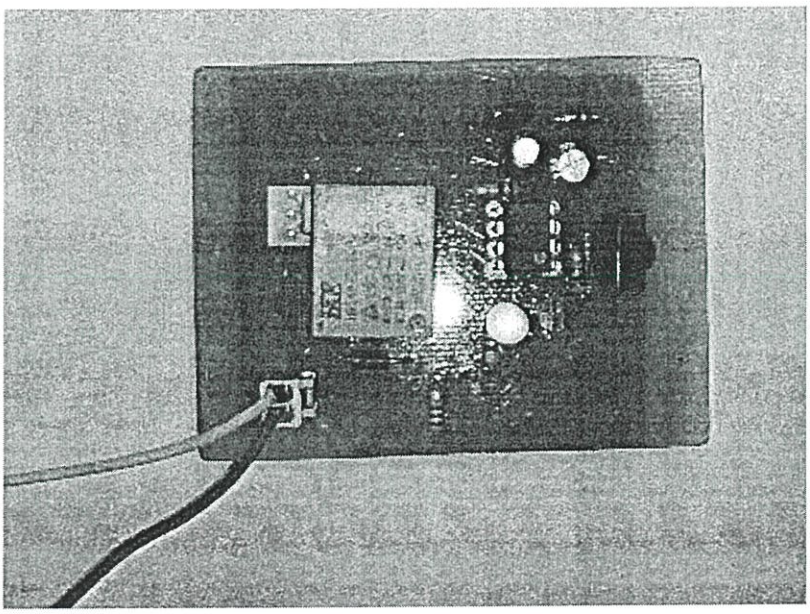
รูปที่ 4.2 วงจรไมโครคอนโทรลเลอร์



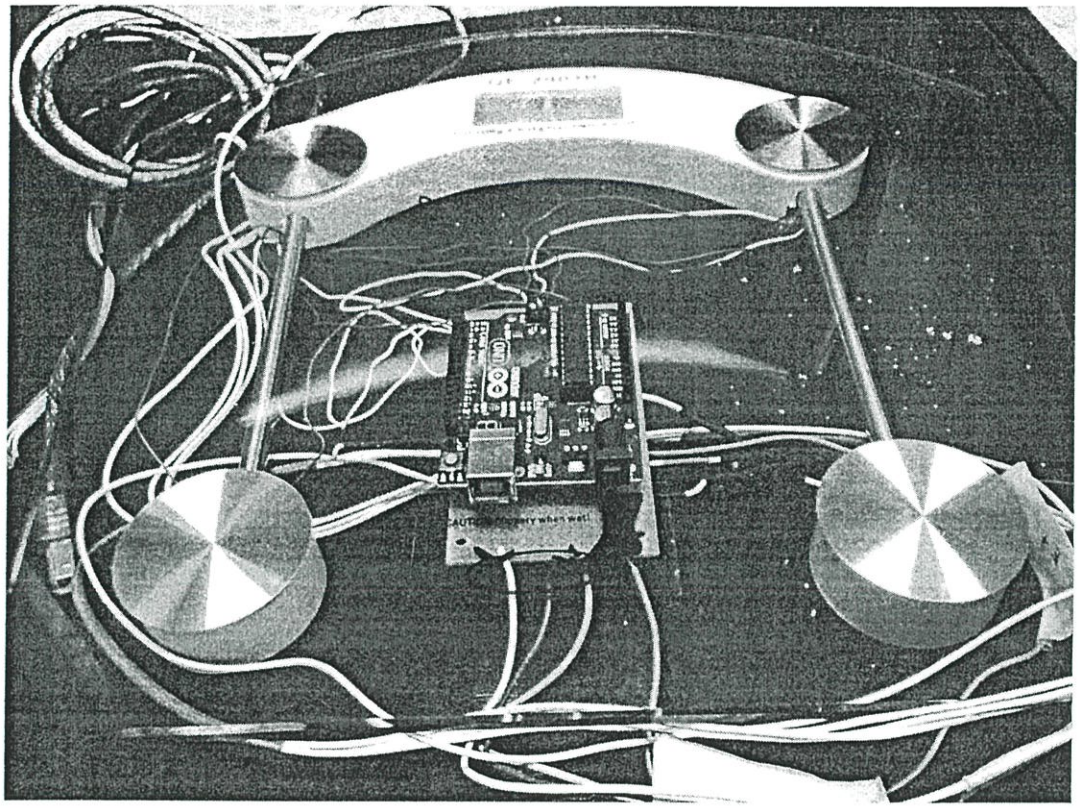
รูปที่ 4.3 วงจรขับมอเตอร์ (Drive motor circuit)



รูปที่ 4.4 วงจรเซ็นเซอร์อินฟราเรดภาคส่ง (Transmitter Infrared Sensor)



รูปที่ 4.5 วงจรเซ็นเซอร์อินฟราเรดภาครับ (Receiver Infrared Sensor)



รูปที่ 4.6 วงจรและอุปกรณ์รับค่าน้ำหนัก (Load-Cell)

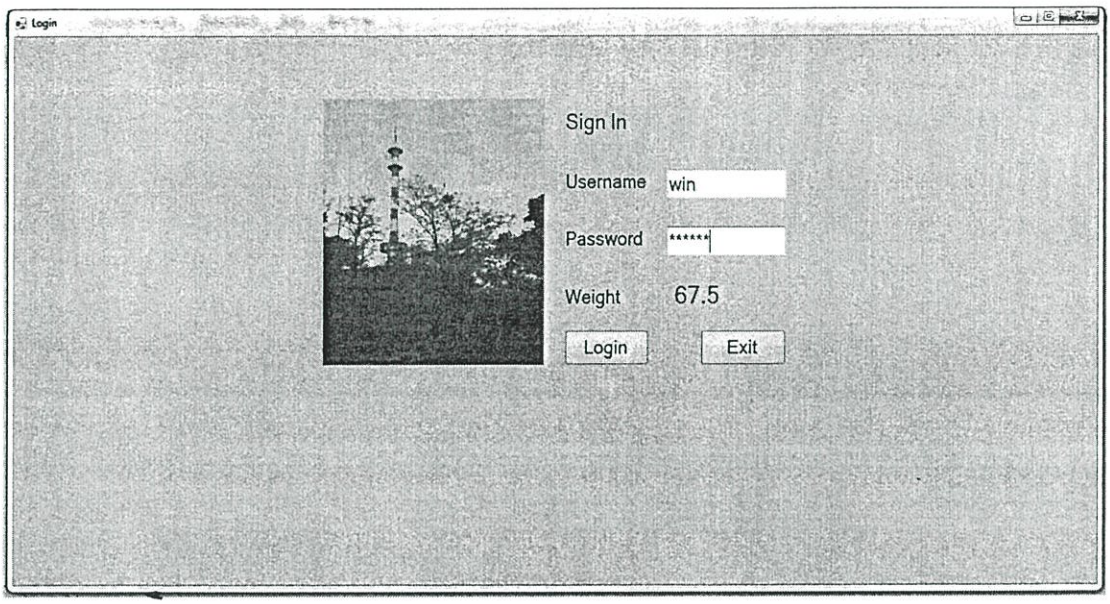
4.2 สัญญาณขาเข้าและออก (Input-Output) ของไมโครคอนโทรลเลอร์

ส่วนนี้จะแสดงถึงสัญญาณขาเข้าและออก (Input-Output) ของไมโครคอนโทรลเลอร์ โดยฝั่งอินพุตจะตั้งค่าลอจิกเป็น “1” ซึ่งจะมีแรงดันเป็น +5 โวลต์ และเมื่อมีการกด, สับสวิทช์ และมีวิ่งกีดขวางเซ็นเซอร์อินฟราเรด ลอจิกทางด้านนี้จะเลี่ยนเป็น “0” ซึ่งมีแรงดันเป็น 0 โวลต์ และฝั่งเอาต์พุตจะมีการตั้งค่าสถานะเริ่มต้นเป็นลอจิก “0” และจะเปลี่ยนแปลงเป็นลอจิก “1” เมื่อต้องการให้อุปกรณ์ป้องกันและแจ้งเตือนภัยทำงาน

4.3 การทดสอบโปรแกรมอินเตอร์เฟสที่ใช้ในการลงชื่อเข้าใช้ (Login)

การทดสอบโปรแกรมดังกล่าวจะแสดงใน Microsoft Visual Studio ขั้นตอนในการลงชื่อเข้าใช้ (Login) เพื่อเปิดประตูจะเริ่มจากรูปที่ 4.7 โดยจะเป็นหน้าโปรแกรมในการลงชื่อเข้าใช้ ซึ่งจะประกอบด้วยชื่อผู้ใช้ (Username) และรหัสผ่าน (Password) จากนั้นเมื่อผู้ใช้มีการเข้าลงชื่อเข้าระบบแล้ว ระบบจะมีการตรวจสอบความถูกต้องจากฐานข้อมูล (Database) ของผู้ดูแลระบบ

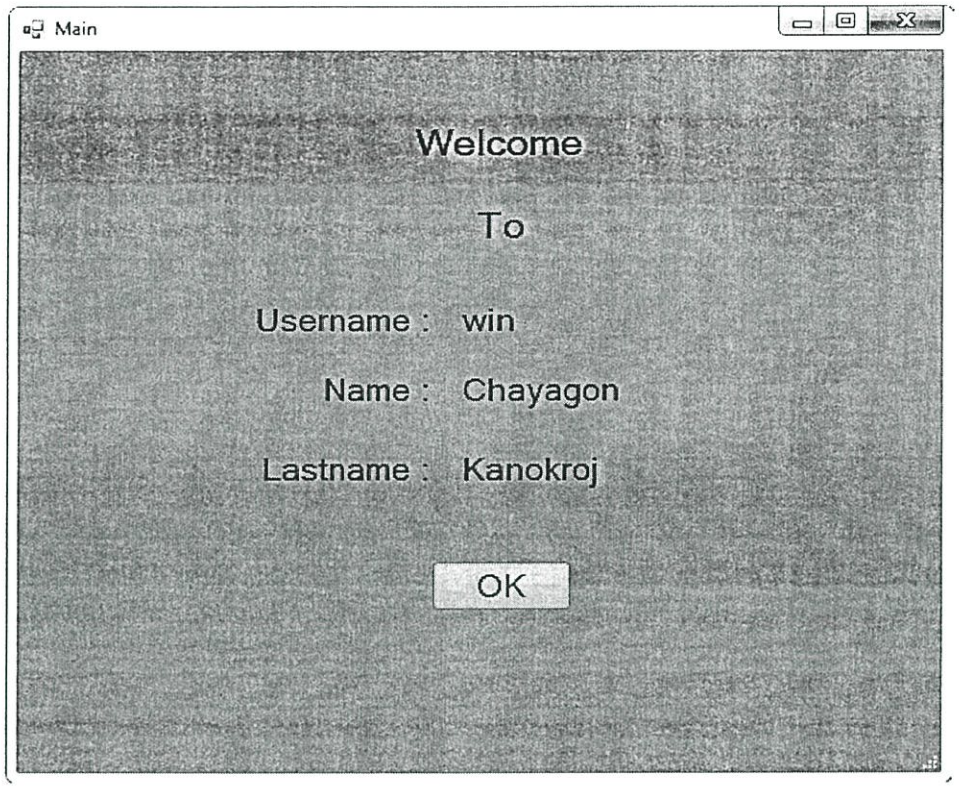
(Admin) ตามรูปที่ 4.8 โดยจะสามารถเปลี่ยนแปลงและแก้ไขได้ และเมื่อเข้าสู่ระบบแล้วจะมีหน้าต่างต้อนรับขึ้นตามชื่อผู้ใช้นั้นๆ ดังรูปที่ 4.9



รูปที่ 4.7 หน้าต่างการลงชื่อเข้าใช้ (Login) เพื่อเปิดประตู

idEmployeeInfo	User_name	Pass_word	Name	Lastname	age	Email	Weight1	Weight2	Weight3
1	win	123456	Chayagon	Kanokroj	21	wwww@gmail.com	66	67	68
2	pop	567891	Papon	Lerksettakanon	21	fffff@hotmail.com	59	60	61
3	aoe	234567	nawakhun	jungcharoensukying	21	gggg@hotmail.com	64	65	66
4	mos	444444	pongsuk	aaaaaaaaaaa	21	yyyy@hotmail.com	67	68	69
5	yee	555555	piyawat	ddddddddddd	21	pppp@hotmail.com	71	72	73
6	big	666666	pasathon	yyyyyyyyyyy	21	bbbb@hotmail.com	61	62	63
7	yod	777777	chayoz	zzzzzzzzzzzz	21	yyyy@gmail.com	67	68	69
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

รูปที่ 4.8 ฐานข้อมูลผู้ใช้ของระบบ



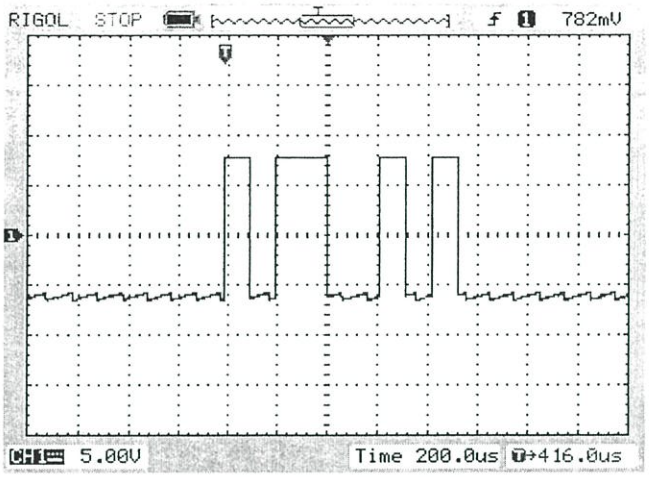
รูปที่ 4.9 หน้าต่างต้อนรับผู้ใช้เมื่อเข้าสู่ระบบ

4.4 การทดสอบวงจรแปลงสัญญาณ TTL

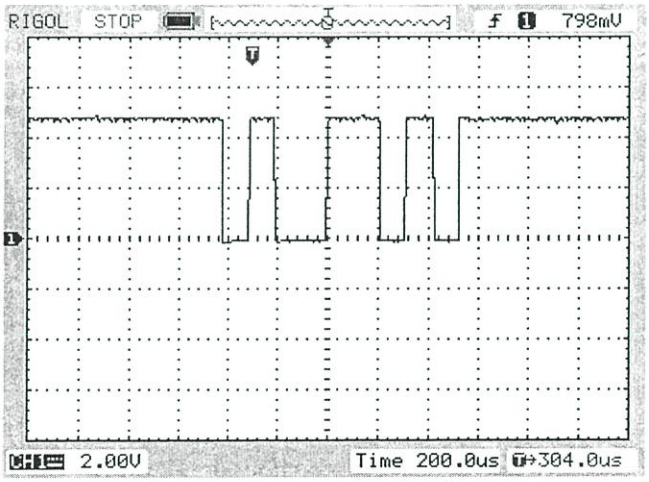
4.4.1 สัญญาณขาเข้าขาและออกของ IC MAX232 จากคอมพิวเตอร์สู่ไมโครคอนโทรลเลอร์

การส่งสัญญาณผ่านทางพอร์ตอนุกรมจากคอมพิวเตอร์ไปสู่ไมโครคอนโทรลเลอร์ จะต้องมีการเปลี่ยนแปลงสัญญาณ ซึ่งเดิมจะเป็นสัญญาณที่มีแรงดัน +8 Volts ถึง -8 Volts โดยจะเป็นสัญญาณขนาด 10 บิต ประกอบด้วย บิตเริ่มต้น (Start bit) 1 บิต , สัญญาณ 8 บิตเป็นรหัสแอสกี้ (ASCII) และบิตหยุด (Stop bit) 1 บิต ไปเป็นสัญญาณ TTL ซึ่งมีแรงดัน 0 Volts ถึง 5 Volts โดยลอจิก 1 จะมีแรงดันเท่ากับ 5 Volts และลอจิก 0 จะมีแรงดันเท่ากับ 0 Volts

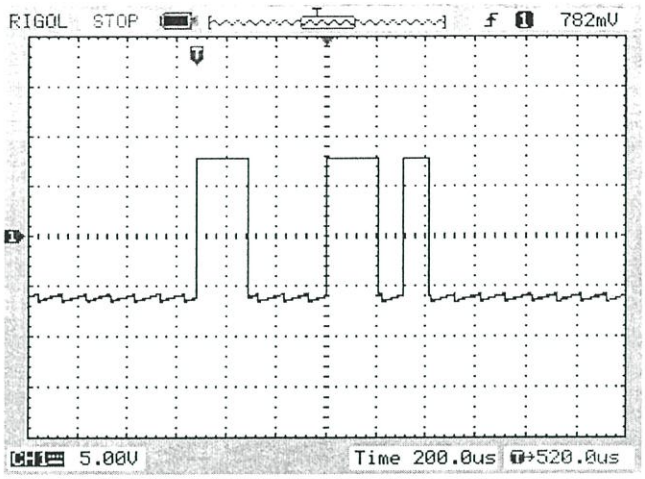
การวัดสัญญาณในส่วนนี้จะแบ่งออกเป็นสัญญาณจากคอมพิวเตอร์ซึ่งมีรหัสแอสกี้เป็น “Y” และ “N” โดยจะแสดงถึงการลงชื่อเข้าใช้ (Login) ที่ถูกต้องและผิดตามลำดับ โดยจะแสดงได้ดังรูปที่ 4.10 ถึง 4.13



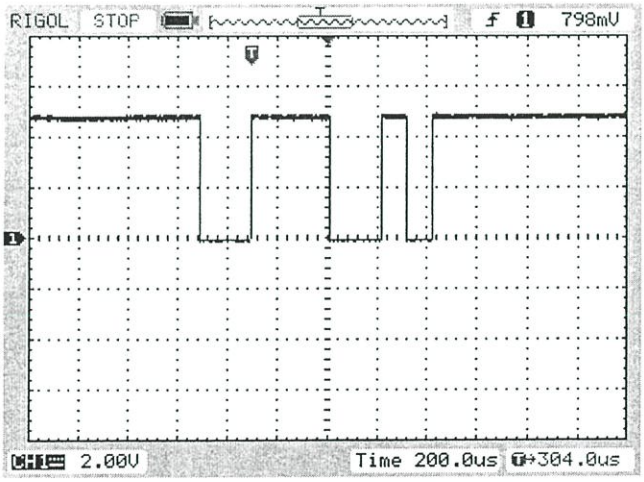
รูปที่ 4.10 สัญญาณขนาด 10 บิตจากคอมพิวเตอร์สู่ IC MAX232 ในการกรณีที่ผู้ใช้ (User) ลงชื่อเข้าใช้ (Login) ถูกต้อง



รูปที่ 4.11 สัญญาณขนาด 10 บิตซึ่งผ่านการแปลงจาก IC MAX232 สู่ไมโครคอนโทรเลอร์ ในการกรณีที่ผู้ใช้ (User) ลงชื่อเข้าใช้ (Login) ถูกต้อง



รูปที่ 4.12 สัญญาณขนาด 10 บิตจากคอมพิวเตอร์สู่ IC MAX232 ในการกรณีที่ผู้ใช้ (User) ลงชื่อเข้าใช้ (Login) ผิด



รูปที่ 4.13 สัญญาณขนาด 10 บิตซึ่งผ่านการแปลงจาก IC MAX232 สู่ไมโครคอนโทรเลอร์ ในการกรณีที่ผู้ใช้ (User) ลงชื่อเข้าใช้ (Login) ผิด

4.5 การทดสอบอินเตอร์เฟสของระบบตรวจสอบอุปกรณ์ (Check System)

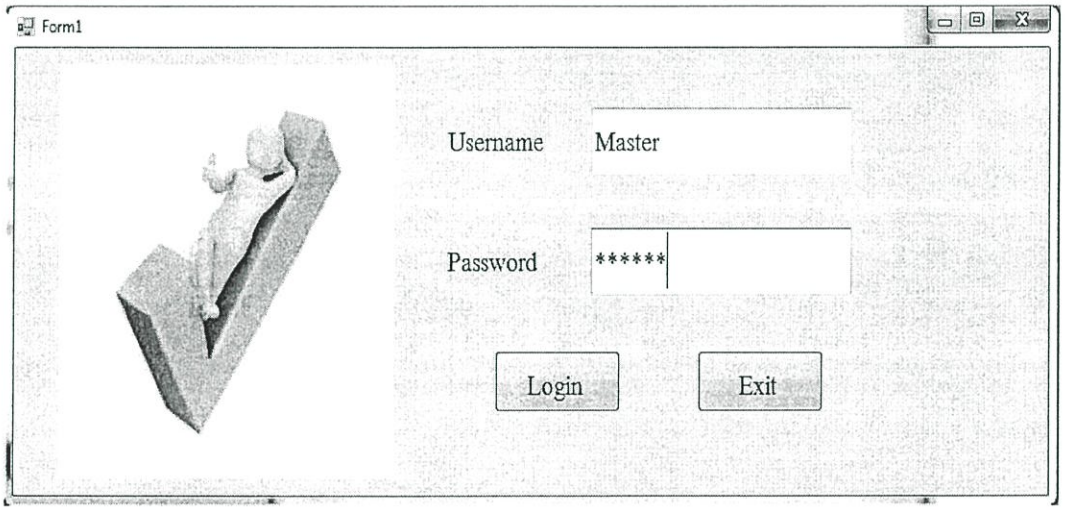
การทดสอบจะแบ่งออกเป็น 2 ส่วนหลักดังนี้

1) ลงชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password)

2) หน้าต่างของระบบตรวจสอบอุปกรณ์

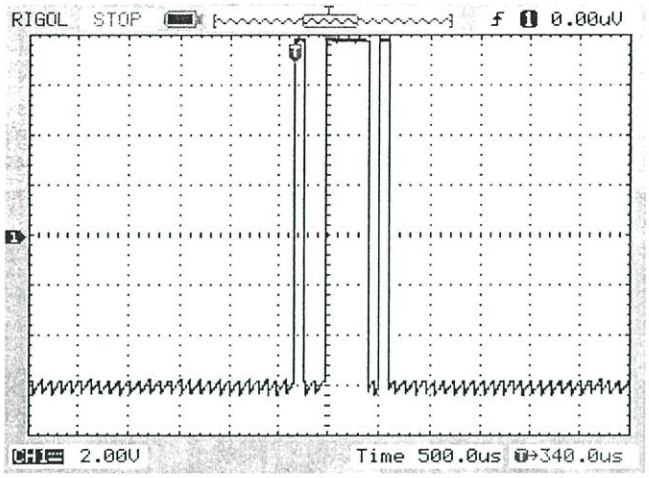
4.5.1 ลงชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password)

การทดสอบโปรแกรมดังกล่าวจะแสดงใน Microsoft Visual Studio ขั้นตอนในการทดสอบระบบตรวจสอบอุปกรณ์จะทำการลงชื่อผู้ใช้ (Username) และรหัสผ่าน (Password) จากนั้นเมื่อลงชื่อผู้ใช้และรหัสผ่านของผู้ใช้แล้ว ระบบจะทำการตรวจสอบชื่อผู้ใช้และรหัสผ่าน หากถูกต้องระบบจะทำการส่งรหัสแอสกี้ (ASCII) ตัวอักษรซี (C)

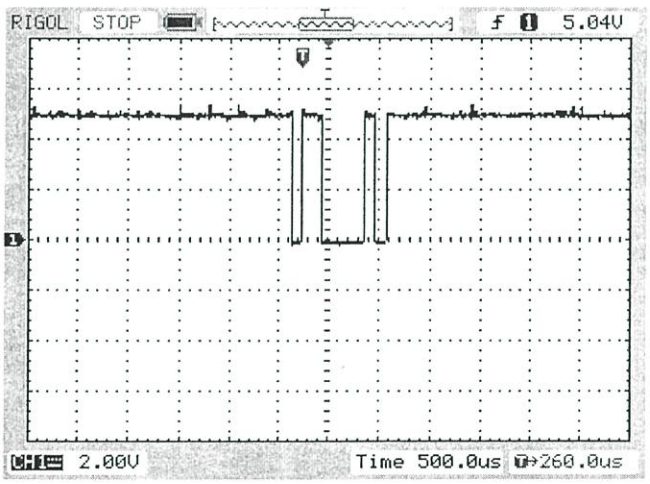


รูปที่ 4.14 หน้าต่างลงชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password)

เมื่อชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password) ถูกต้องระบบจะทำการส่งข้อมูลเป็นรหัสแอสกี้ (ASCII) ตัวอักษรซี (C)



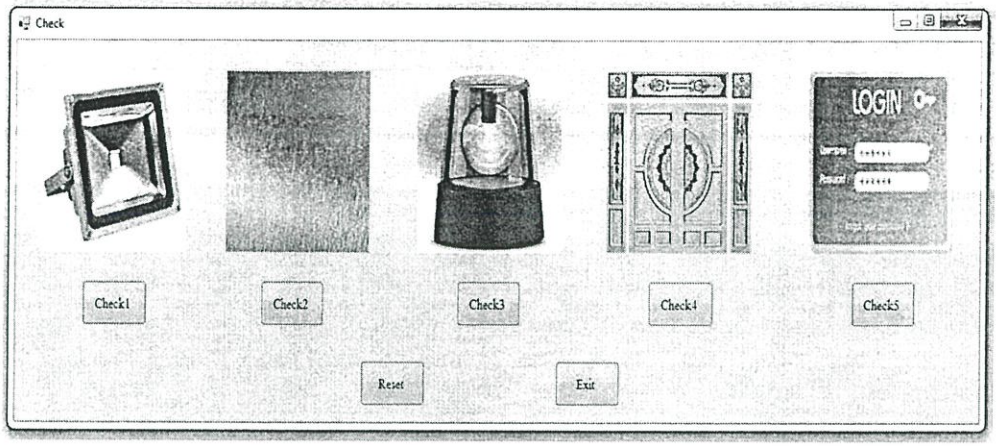
รูปที่ 4.15 รหัสแอสกี้ (ASCII) ตัวอักษรซี (C) สัญญาณ RS-232



รูปที่ 4.16 รหัสแอสกี้ (ASCII) ตัวอักษรซี (C) เมื่อผ่านวงจรแปลง Max232

4.5.2 หน้าต่างของระบบตรวจสอบอุปกรณ์

เมื่อชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password) ถูกต้องระบบจะทำการเข้าสู่หน้าต่างของระบบตรวจสอบ ดังรูปที่ 4.17



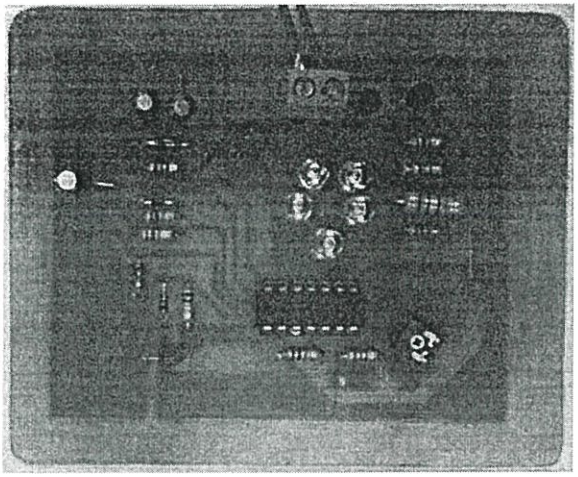
รูปที่ 4.17 หน้าต่างของระบบตรวจสอบ

เมื่อกดปุ่ม “Check1” ระบบจะทำการส่งรหัสแอสกี้ (ASCII) เป็นตัวอักษรแอล (L) ,กดปุ่ม “Check2” ระบบจะทำการส่งรหัสแอสกี้ (ASCII) เป็นตัวอักษรดับเบิ้ลยู (W) ,กดปุ่ม “Check3” ระบบจะทำการส่งรหัส ASCII เป็นตัวอักษรเอ (A) ,กดปุ่ม “Check4” ระบบจะทำการส่งรหัส ASCII เป็นตัวอักษรเอฟ (F) ,กดปุ่ม “Check5” ระบบจะทำการส่งรหัส ASCII เป็นตัวอักษรบี (B) ,กดปุ่ม “Reset” ระบบจะทำการส่งรหัส ASCII เป็นตัวอักษรอาร์ (R) และเมื่อกดปุ่ม “Exit” ระบบจะทำการแสดงกล่องข้อความ “Exit Application?” และทำการส่งรหัส ASCII เป็นตัวอักษรเอ็ก (X) โดยสัญญาณดังกล่าวจะถูกแปลงเป็นสัญญาณ TTL จากวงจร MAX-232 เช่นเดียวกัน จากนั้นจะเข้าสู่การทำงานของไมโครคอนโทลเลอร์ต่อไป

4.6 การทดสอบวงจรเซ็นเซอร์อินฟราเรด (Infrared Sensor)

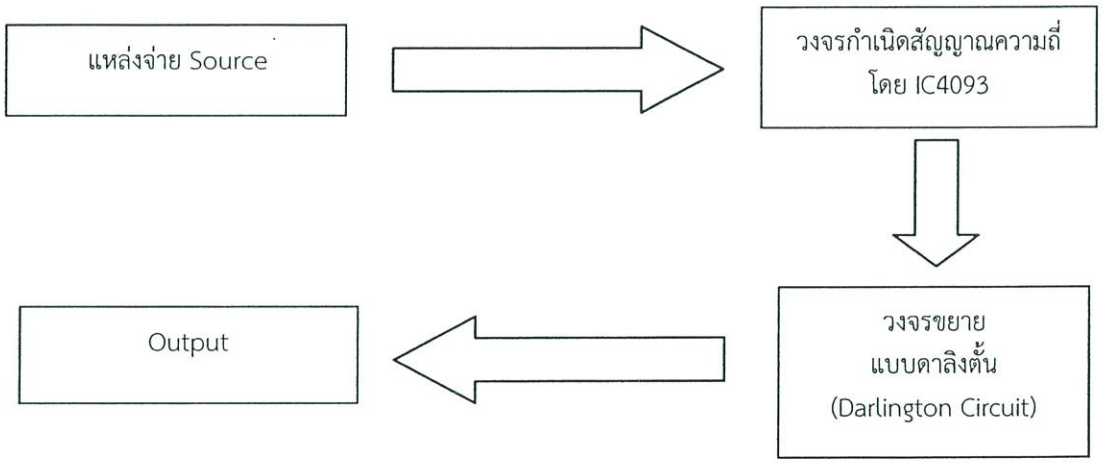
4.6.1 การทดสอบวงจรภาคส่งเซ็นเซอร์แสงอินฟราเรด (Sensor Infrared Transmitter Circuit)

ในส่วนนี้จะเป็นส่วนที่สร้างแหล่งกำเนิดแสงอินฟราเรด (Infrared) เพื่อส่งไปยังด้านรับ (Receiver) โดยแสงอินฟราเรดจะถูกส่งตลอดเวลาเมื่อป้อนไฟเลี้ยงแก่วงจร ส่วนทางด้านรับจะให้เอาท์พุตแตกต่างกันออกไปในขณะที่ได้รับแสงและไม่ได้รับแสงอินฟราเรด โดยบล็อกไดอะแกรมการทำงานของวงจรจะแสดงได้ดังรูปที่ 4.19



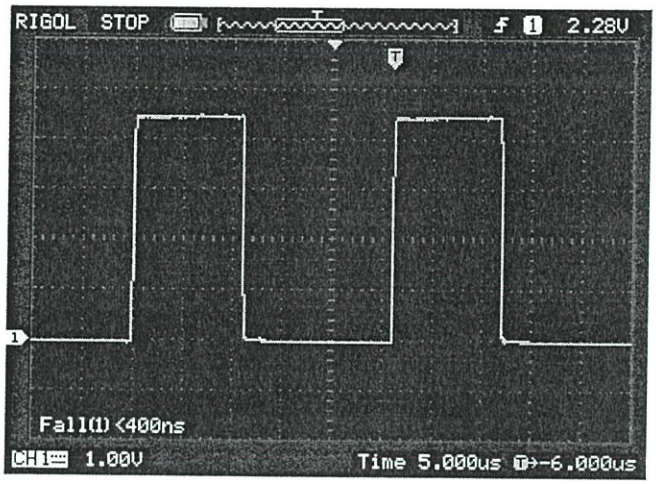
รูปที่ 4.18 วงจรภาคส่งเซ็นเซอร์อินฟราเรด (Sensor Infrared Transmitter circuit)

โดยจะทำการวัดผลของวงจรตามบล็อกไดอะแกรม



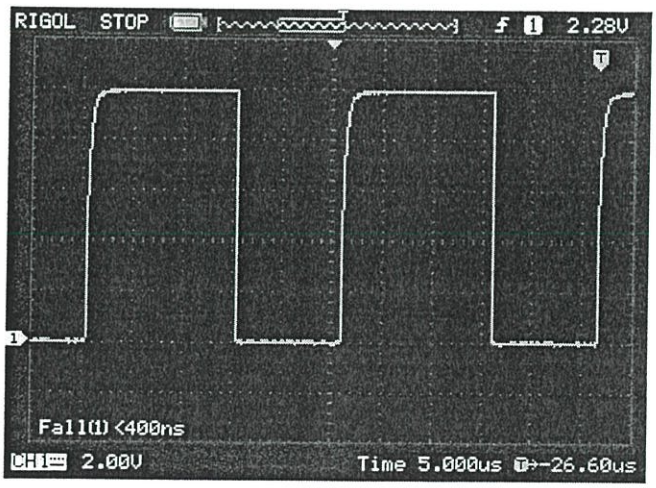
รูปที่ 4.19 บล็อกไดอะแกรมที่จะวัดผลการทดลอง

เมื่อจ่ายแหล่งจ่ายไฟ 12 โวลต์จะไปผ่านวงจรกำเนิดสัญญาณความถี่ในรูปแบบดิจิตอล โดยจะใช้ไอซีเบอร์ 4093 เป็นตัวผลิตสัญญาณความถี่แบบดิจิตอล จะได้ผลลัพธ์ตามรูปที่ 4.20 ซึ่งจะมีความถี่เท่ากับ 38 kHz โดยจะใช้ความถี่เท่ากันนี้ทั้งในฝั่งส่งและรับอินฟราเรด (Infrared)

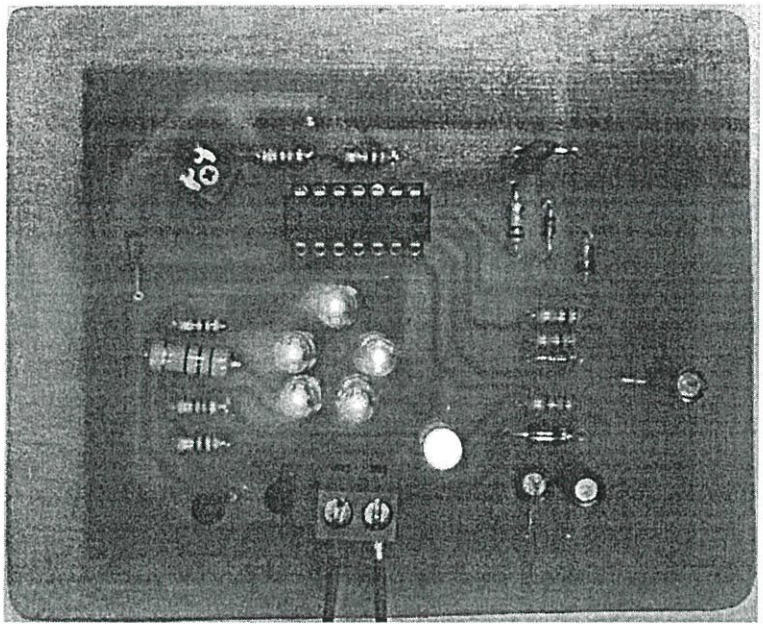


รูปที่ 4.20 สัญญาณที่ออกจากวงจรกำเนิดสัญญาณความถี่

จากนั้นจะไปผ่านวงจรขยายแบบดาลิงตั้น (Darlington Circuit) โดยจะผ่านทรานซิสเตอร์ (Transistor) ทั้งสองตัวและได้ผลลัพธ์ตามรูปที่ 4.21 โดยหากเปรียบเทียบกับรูปที่ 4.20 จะเห็นได้ว่าสัญญาณมีการขยายจาก 4.5 Volts เป็น 10 Volts



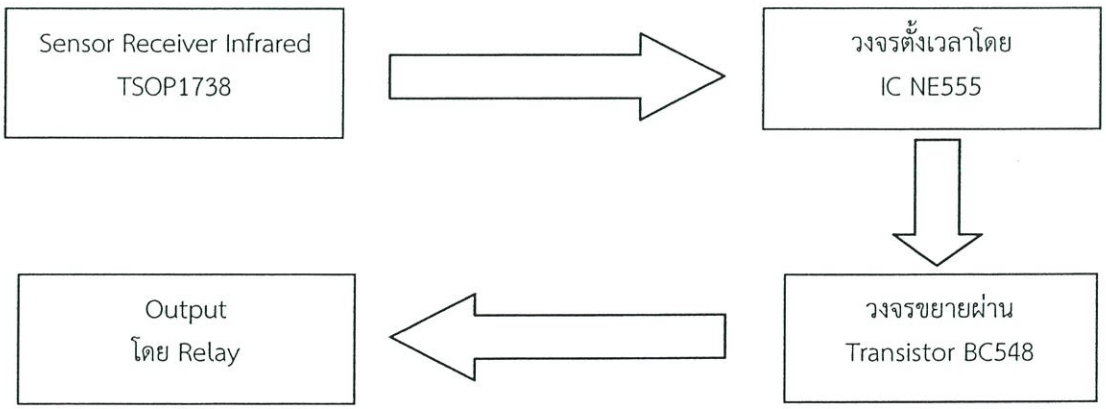
รูปที่ 4.21 สัญญาณที่ออกจากวงจรขยาย (Darlington Circuit)



รูปที่ 4.22 วงจรภาคส่งเซ็นเซอร์อินฟราเรด (Sensor Infrared Transmitter circuit) เมื่อทำงาน

4.6.2 การทดสอบวงจรภาครับเซ็นเซอร์แสงอินฟราเรด (Sensor Infrared Transmitter Circuit)

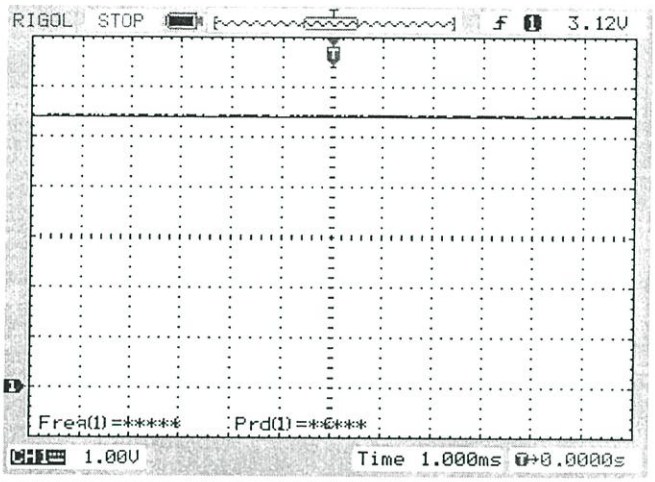
จะทำการวัดผลของวงจรตามบล็อกไดอะแกรม โดยการวัดจะแบ่งออกเป็น 2 ช่วง คือ ช่วงที่ไม่มีวัตถุทึบแสงมาคั่นระหว่างเซนเซอร์รับ-ส่งอินฟราเรด และช่วงที่มีวัตถุมาคั่นกลาง



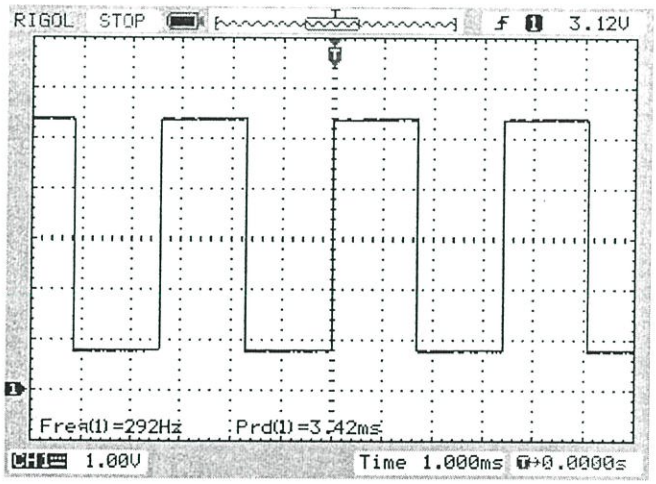
รูปที่ 4.23 บล็อกไดอะแกรมที่จะวัดผลการทดลองฝั่งรับอินฟราเรด

4.6.2.1 ผลเอาท์พุต (Output) จากตัวรับเซนเซอร์ (Infrared Receiver) TSOP1728

การวัดจะทำการวัดจากแรงดันขาออกออกจากอุปกรณ์ TSOP (pin 3)

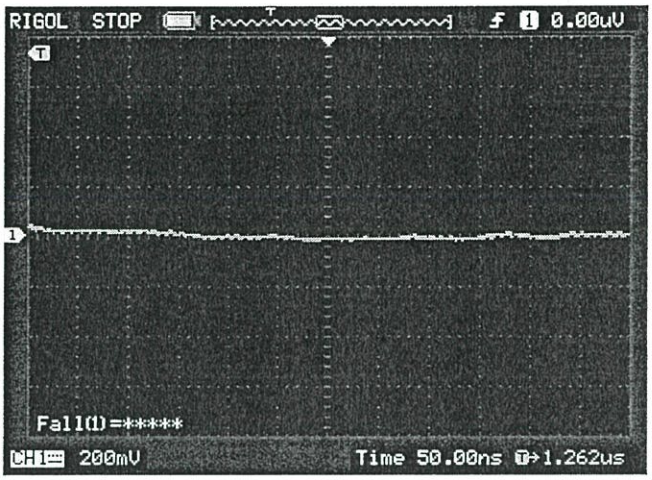


รูปที่ 4.24 สัญญาณขาออกจากอุปกรณ์ฝั่งรับเมื่อไม่มีวัตถุทึบแสงมาคั่นกลาง

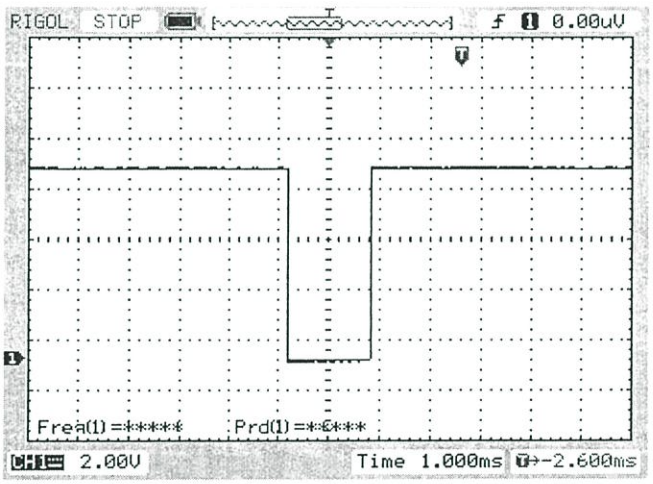


รูปที่ 4.25 สัญญาณขาออกจากอุปกรณ์ฝั่งรับเมื่อมีวัตถุทึบแสงมาคั่นกลาง

จากรูปที่ 4.24 และ 4.25 จะทราบว่าเมื่อมีวัตถุทึบแสงมาคั่นระหว่างตัวเซ็นเซอร์ สัญญาณจะเปลี่ยนรูปแบบจากแรงดันไฟตรงเป็นสัญญาณ TTL ส่งไปยัง IC เบอร์ 555 จากนั้นจะทำการวัดเอาท์พุตจากขาไอซีที่ต่อเข้ากับขารีเลย์ก่อนถึงตัวไมโครคอนโทรลเลอร์แสดงดังรูปที่ 4.26, 4.27



รูปที่ 4.26 สัญญาณขาออกจากไอซีเบอร์ 555 เมื่อไม่มีวัตถุทึบแสงมาคั่นกลาง



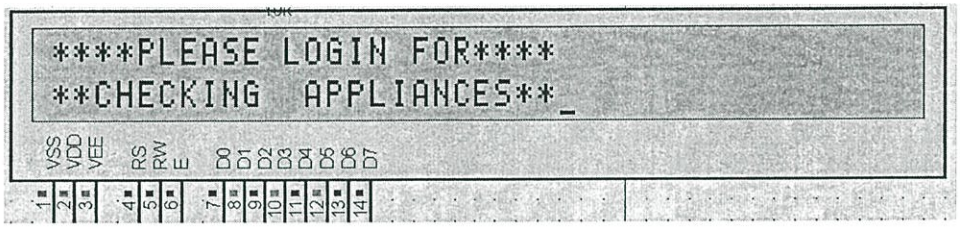
รูปที่ 4.27 สัญญาณขาออกจากไอซีเบอร์ 555 เมื่อมีวัตถุทึบแสงมาคั่นกลาง

จะเห็นจากรูป 4.27 เปรียบเทียบกับรูป 4.26 ได้ว่าเมื่อได้รับสัญญาณ TTL จากอุปกรณ์ TSOP แล้ว เอาท์พุตของไอซี 555 จะมีแรงดันเป็นไฟตรงประมาณ 7.5 volts และจะมีขาลงของแรงดันนานเท่ากับช่วงเวลาของแรงดันของสัญญาณ TTL ในรูปที่ 4.25 และจะสามารถกำหนดที่ตัวเก็บประจุของไอซีได้ว่าเมื่อมีวัตถุมาคั่นกลางระหว่างเซ็นเซอร์แล้วเป็นเวลานานเท่าใดจึงจะค่อยจ่ายแรงดันขาออกนี้ โดยการคำนวณจะอยู่ในหัวข้อ 3.1.2.4

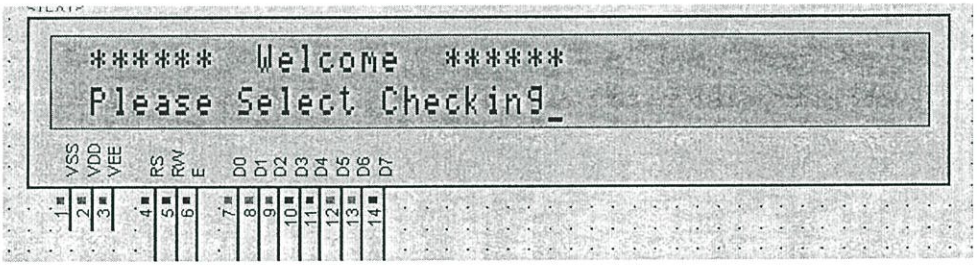
4.7 หน้าจอแสดงผล LCD

4.7.1 ข้อความต้อนรับเมื่อเปิดระบบ

เมื่อเปิดสวิทช์ของระบบ หน้าจอ LCD จะมีข้อความต้อนรับดังรูปที่ 4.28 เพื่อรอรับการล็อกอิน (Login) ในการเช็คอุปกรณ์ ซึ่งเมื่อมีการล็อกอินเข้าสู่ระบบ การแสดงผลจะเป็นไปตามรูปที่ 4.29



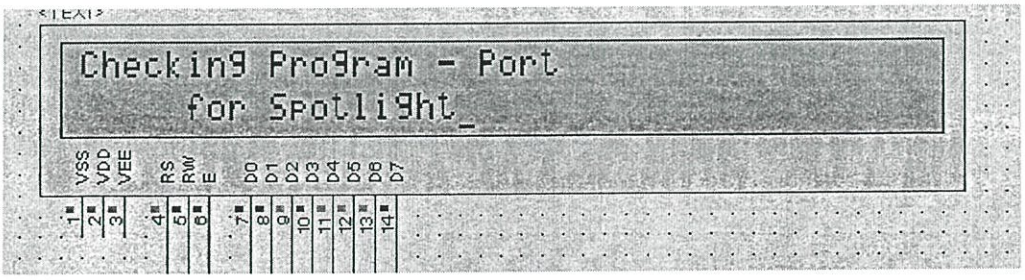
รูปที่ 4.28 ข้อความต้อนรับของจอแสดงผล LCD เมื่อเปิดระบบ



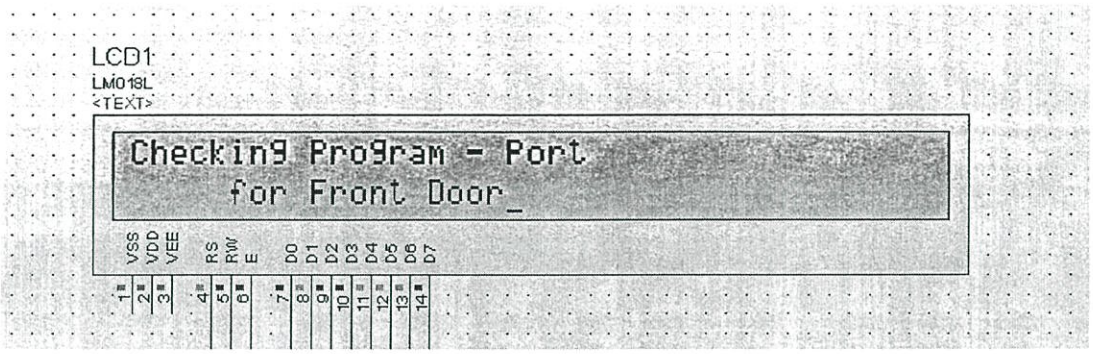
รูปที่ 4.29 ข้อความต้อนรับของจอแสดงผล LCD เมื่อมีการล็อกอินเพื่อตรวจสอบอุปกรณ์

4.7.2 ข้อความในขณะเช็คอุปกรณ์

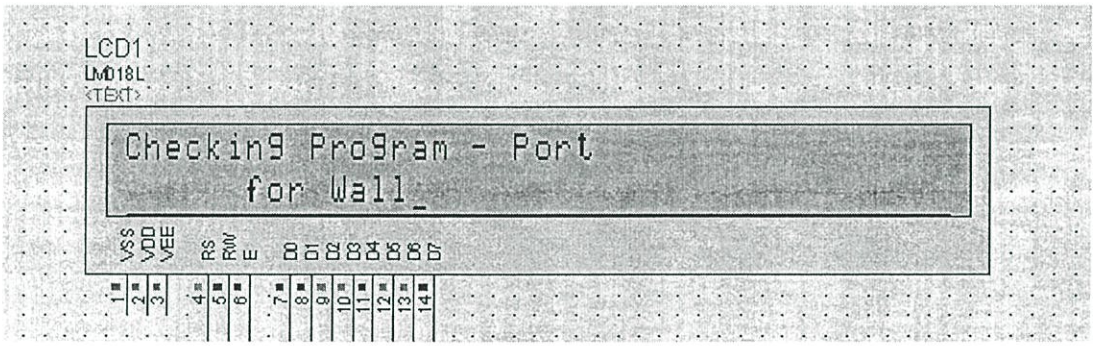
เมื่อไมโครคอนโทรลเลอร์ที่ควบคุมการแสดงผลได้รับสัญญาณจากไมโครคอนโทรลเลอร์ตัวอื่นๆ จะมีข้อความแสดงถึงอุปกรณ์ที่กำลังเช็คอยู่ โดยจะมีทั้งหมด 5 รูปแบบตามตัวอุปกรณ์ นั่นคือ สปร์ตไลท์ (Spotlight) , ประตูหน้า (Front Door) , กำแพง (Wall) , ประตูนิรภัย (Security Door) และเสียงกริ่งเตือน (Alarm) ดังรูปที่ 4.30 ถึง 4.34 ตามลำดับ



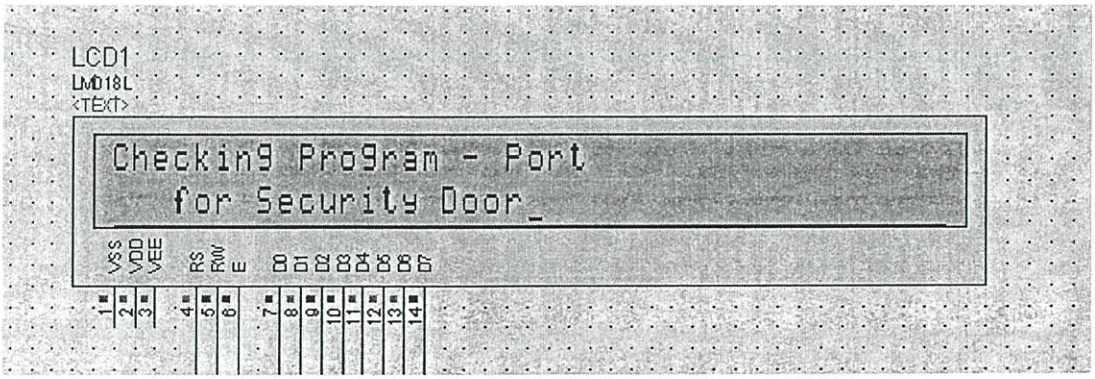
รูปที่ 4.30 ข้อความแสดงอุปกรณ์สปอร์ตไลท์ (Spotlight) ที่ระบบกำลังทำการตรวจสอบ



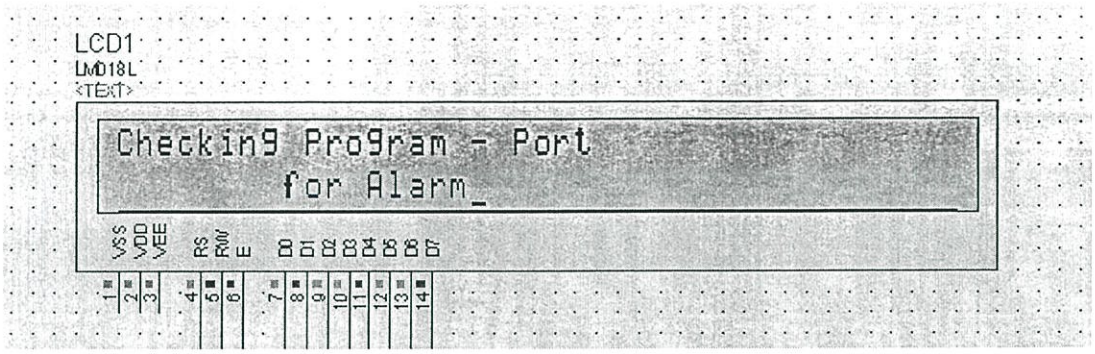
รูปที่ 4.31 ข้อความแสดงอุปกรณ์ประตูหน้า (Front Door) ที่ระบบกำลังทำการตรวจสอบ



รูปที่ 4.32 ข้อความแสดงอุปกรณ์กำแพง (Wall) ที่ระบบกำลังทำการตรวจสอบ



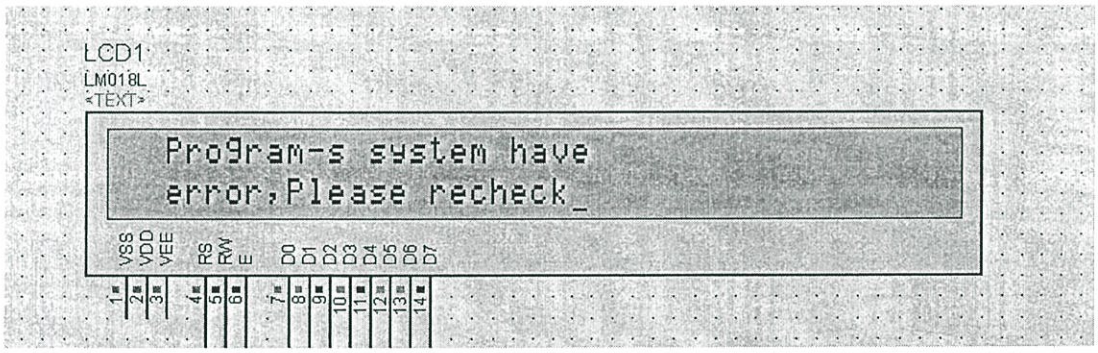
รูปที่ 4.33 ข้อความแสดงอุปกรณ์ประตูนิรภัย (Security Door) ที่ระบบกำลังทำการตรวจสอบ



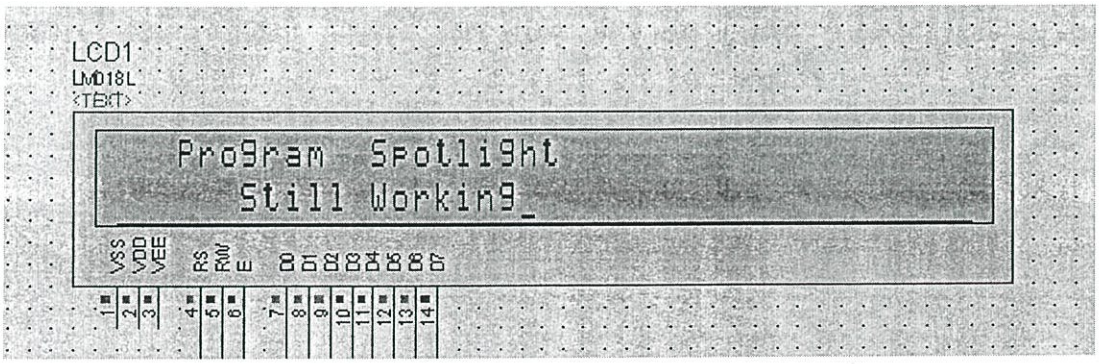
รูปที่ 4.34 ข้อความแสดงอุปกรณ์เสียงกริ่งเตือน (Alarm) ที่ระบบกำลังทำการตรวจสอบ

4.7.3 ข้อความแสดงผลลัพธ์ของการเช็คอุปกรณ์

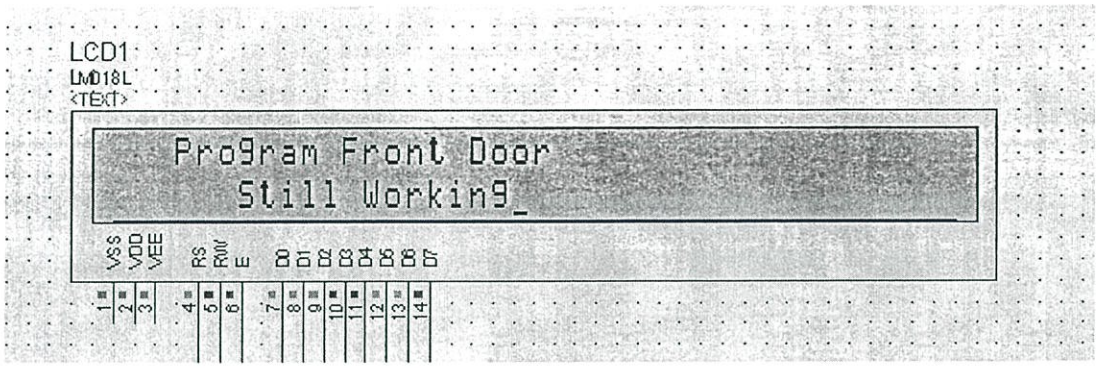
ข้อความแสดงผลลัพธ์จะแสดงว่าอุปกรณ์ยังใช้งานได้หรือไม่ หรืออุปกรณ์นั้นเกิดปัญหา โดยหากเกิดปัญหาจะแสดงดังรูปที่ 4.35 แต่หากอุปกรณ์ยังใช้งานได้ จะแสดงตามชนิดของอุปกรณ์ดังรูปที่ 4.36 ถึง 4.40



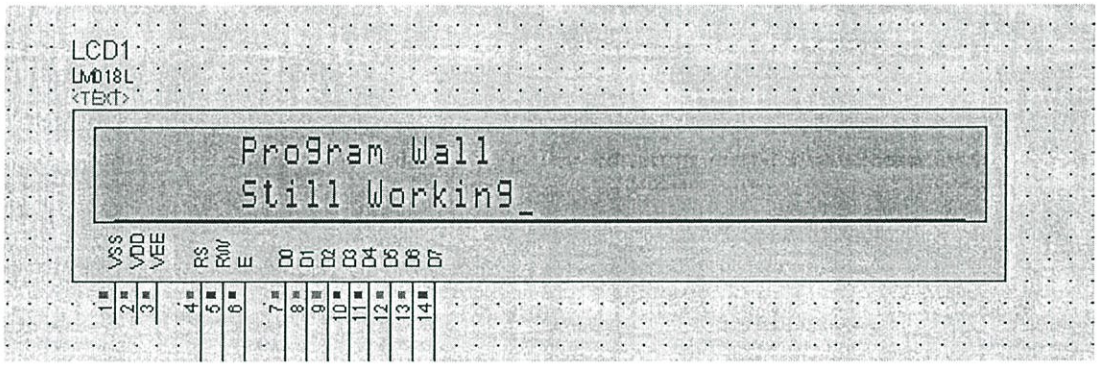
รูปที่ 4.35 ข้อความเมื่อผลของการเช็คอุปกรณ์แสดงว่าอุปกรณ์เกิดปัญหา



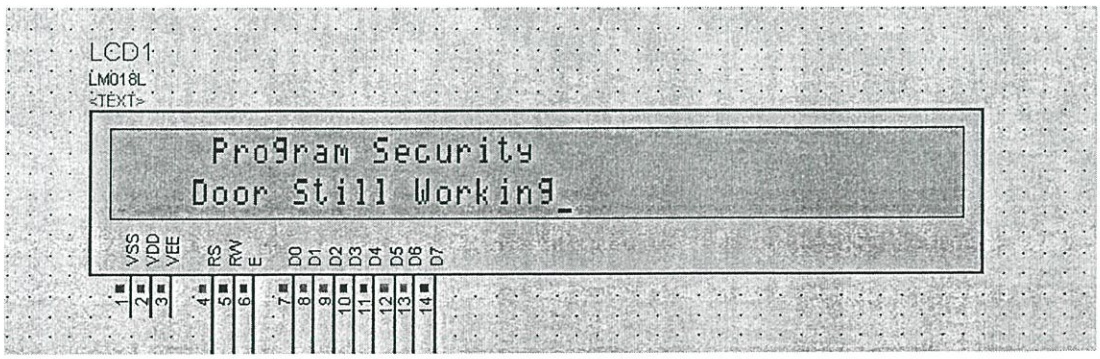
รูปที่ 4.36 ข้อความเมื่อผลของการเช็คอุปกรณ์สปอร์ตไลท์ (Spotlight) ยังใช้งานได้



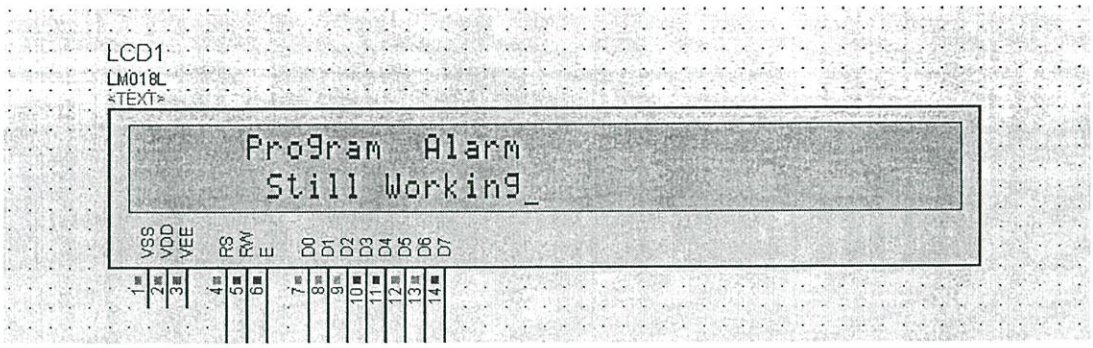
รูปที่ 4.37 ข้อความเมื่อผลของการเช็คอุปกรณ์ประตูหน้า (Front Door) ยังใช้งานได้



รูปที่ 4.38 ข้อความเมื่อผลของการเช็คอุปกรณ์กำแพง (Wall) ยังใช้งานได้



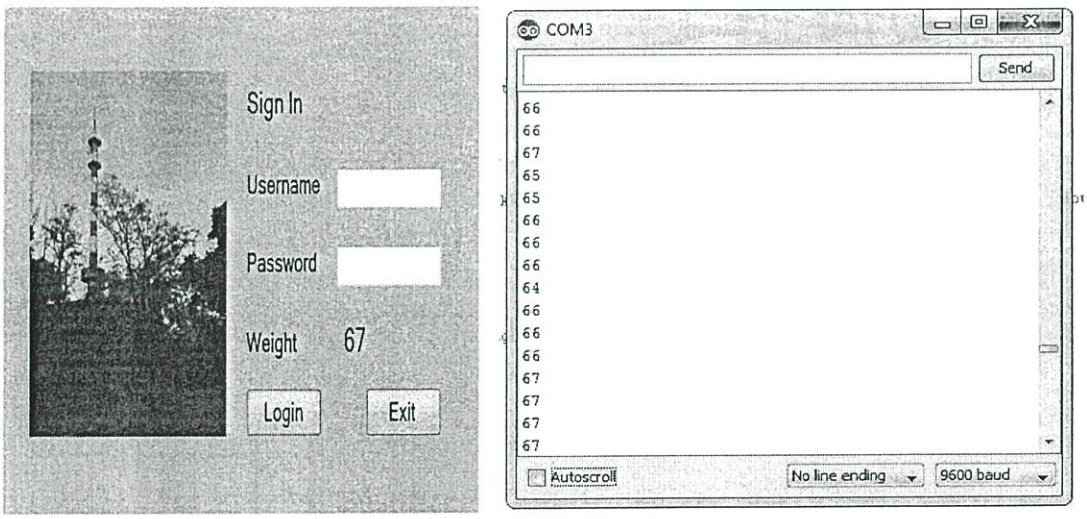
รูปที่ 4.39 ข้อความเมื่อผลของการเช็คอุปกรณ์ประตูนิรภัย (Security Door) ยังใช้งานได้



รูปที่ 4.40 ข้อความเมื่อผลของการเช็คอุปกรณ์เสียงกริ่งเตือน (Alarm) ยังใช้งานได้

4.8 Load-Cell

เมื่อชั่งน้ำหนักค่าน้ำหนักที่อ่านได้จากโปรแกรมอาดูโน่ (Arduino) จะส่งผ่านวงจร MAX-232 เพื่อแปลงสัญญาณ TTL เป็น RS-232 แล้วส่งเข้าพอร์ตอนุกรม (Serial Port) ไปแสดงผลที่โปรแกรมเข้ารหัสของคอมพิวเตอร์อีกเครื่องหนึ่ง เพื่อใช้เป็นอีกหนึ่งข้อมูลในการกำหนดความถูกต้องของชื่อผู้ใช้ (Username) , รหัสผ่าน (Password) และน้ำหนัก (Weight)



รูปที่ 4.41 เอาท์พุตจากบอร์ดอาร์ดูโน่ (Arduino) และการแสดงผลไปยังโปรแกรมอินเตอร์เฟสของการเข้ารหัส (Login)

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้เป็นการเสนอเกี่ยวกับการเขียนโปรแกรมไมโครคอนโทรลเลอร์ควบคุมการทำงานของอุปกรณ์ ได้แก่ สปอร์ตไลท์ (Spotlight) , เสียงกริ่งแจ้งเตือน (Alarm) , กำแพง (DC Motor-Wall) , ประตูหน้า (DC Motor-Front Door) , ประตูนิรภัย (DC Motor Security Door) , อุปกรณ์ตรวจสอบน้ำหนัก (Load-Cell) , เซ็นเซอร์อินฟราเรด (Infrared Sensor) เพื่อป้องกันและแจ้งเตือนภัยต่อการบุกรุก โดยผู้ใช้จะสามารถควบคุมจากสวิทช์ , โปรแกรมอินเตอร์เฟส (Interface Program) สำหรับเปิดประตูนิรภัยและหยุดการทำงานของอุปกรณ์ โดยการเข้าเช็คความพร้อมของอุปกรณ์โดยจะต้องมีการเข้ารหัส (Login) ด้วยชื่อผู้ใช้ (Username) , รหัสผ่าน (Password) และน้ำหนัก (Weight) โดยจะรวมถึงการสร้างและศึกษาวงจรเซ็นเซอร์อินฟราเรดทางด้านส่ง , ทางด้านรับ (Transmitter and Receiver Infrared Sensor) และ Load-Cell ซึ่งจะมีทดสอบการทำงานผ่านโปรแกรมจำลองต่างๆ ก่อน จากนั้นจะทำการทดสอบจริงโดยใช้แบบจำลอง (Model) ที่สร้างขึ้น

จากผลการทดลอง ระบบมีการทำงานเป็นไปตามที่กำหนดไว้ได้ดี แบบจำลองที่สร้างขึ้นมีการติดขัดในเรื่องของกลไกการทำงานเล็กน้อย และการรับค่าจากอุปกรณ์ช่างน้ำหนักยังมีความคลาดเคลื่อนบ้างเล็กน้อยเนื่องจากค่าที่รับได้ไม่คงที่

5.2 ข้อเสนอแนะ

แนวทางการพัฒนาคือ ควรจะต้องออกแบบวงจรที่ใช้ควบคุมอุปกรณ์ป้องกันและแจ้งเตือนภัยเพิ่มเติมทั้งทางด้านอุปกรณ์และกลไกลในการทำงานต่างๆ เพื่อให้มีการใช้งานได้จริงตามระบบที่ออกแบบไว้ รวมถึงควรจะพัฒนาระบบให้เชื่อมต่อกับอินเตอร์เน็ตได้ , ออกแบบโปรแกรมอินเตอร์เฟสเพิ่มเติมกับการใช้งานและเพิ่มโปรแกรมในการป้องกันและแจ้งเตือนภัยให้มากขึ้น เพื่อเป็นการเพิ่มความหลากหลายและความสะดวกในการใช้งาน รวมถึงควรแก้ไขการรับค่าน้ำหนักโดยทำให้มีความคลาดเคลื่อนที่น้อยลง ซึ่งทั้งนี้ก็จะเป็นการทำให้ระบบป้องกันมีประสิทธิภาพที่ดีขึ้นนั่นเอง

บรรณานุกรม

[1] http://www.nectec.or.th/schoolnet/library/webcontest2003/100team/dlnes137/am/Microcontroller.html

[2] http://www.mind-tek.net/port.php

[3] http://www.oocities.org/siliconvalley/station/3169/mcs51.htm

[4] http://www.semi-shop.com/knowledge/knowledge_detail.php?sk_id=32

[5] http://www.engineersgarage.com/microcontroller/8051projects/interface-serialport-RS232-AT89C51-circuit

[6] http://th.wikipedia.org/wiki/%E0%B9%81%E0%B8%AD%E0%B8%AA%E0%B8%81%E0%B8%B5

[7] http://www.cpe.ku.ac.th/~yuen/204323/pheriperal/s1.htm

[8] http://www.slri.or.th/th/index.php?option=com_content&view=article&id=42&Itemid=88

[9] http://www.softwaresiam.com/webboard/thread-7666-1-1.html

[10]http://th.wikipedia.org/wiki/%E0%B9%82%E0%B8%AB%E0%B8%A5%E0%B8%94%E0%B9%80%E0%B8%8B%E0%B8%A5

ผศ. ธีรวัฒน์ ประกอบผล.*การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี*. พิมพ์ครั้งที่6. กรุงเทพฯ : สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี ไทย-ญี่ปุ่น, 2545.

ทีมงานสมาร์ทเลิร์นนิ่ง. *เรียนรู้ไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษา C พร้อมโครงงาน.* กรุงเทพฯ : ห้างหุ้นส่วนสามัญสมาร์ทเลิร์นนิ่ง, 2552.

ดอนสัน ปงผาบ. *การเขียนโปรแกรมภาษาซีในงานควบคุม.* กรุงเทพฯ : สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี ไทย-ญี่ปุ่น, 2547

Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay. *The 8051 microcontroller and embedded systems : using Assembly and C.* ,2009

การใช้งานไทเมอร์ใน ไมโครคอนโทรลเลอร์ MCS-51
http://www.mind-tek.net/timer.php

การสื่อสารข้อมูลผ่านพอร์ตอนุกรม
http://www.mcuthailand.com/articles/mcs/Ex_Uart.html

โครงสร้างสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51
http://eng.sut.ac.th/tce/SeniorProjects/old/student2/ST51.htm

ไมโครคอนโทรเลอร์ Mcs51

ภาคผนวก

ก

```
#include <reg52.h>

sbit nightmode = P1^0 ;              /* day or night mode */
sbit lightcheck = P1^1 ;             /* infrared sensor */
sbit presscheck = P1^2 ;             /* emergency button */
sbit ip2 = P1^3 ;                    /* alarm if incorrect login */
sbit ChkS = P0^1 ;                   /* check system */
sbit ok1 = P0^4 ;                    /* say okay in check system to LCD */


sbit misw1fw = P1^4 ;                /* micro switch wall stop forward */
sbit misw2fw = P1^5 ;                /* micro switch front door stop forward */
sbit misw1bw = P1^6 ;                /* micro switch wall stop backward */
sbit misw2bw = P1^7 ;                /* micro switch front door stop backward        */
                                     /*          P1 - Input
                                              P2 - Output     P2.0-2.1 = wall
                                                              P2.2-2.3 = front door
                                                              P2.4-2.5 = enable motor
                                                              P2.6 = spotlight
                                                              P2.7 = alarm         */
sbit Reset = P0^0 ;                  /* RESET SYSTEM */
int x ;
char RxBuff ;                        /* Rx variable */
void delay(unsigned int loop) ;      /* Delay */
void uart_init(void) ;               /* Baud rate */
void AAA();                          /* Stop wall-up */
void BBB();                          /* Stop front door-open */
void CCC();                          /* wait for RESET */
void DDD();                          /* Stop wall-down */
void EEE();                          /* Stop front door-close */

/*DELAY FUNCTION*/
void delay(unsigned int loop)
{
    unsigned int a , b ;
    for(a=0;a<loop;a++)
    {
        for(b=0;b<1275;b++) ;
    }
}

/* BAUD RATE 9600Mbps (SERIAL PORT)*/
void uart_init(void)
{
    TR1 = 0 ;           /*Timer 1 Disable*/
    ET1 = 0 ;           /*Enable Time1 Interupt*/
    TMOD = 0x20 ;       /*Set Timer 1 as Mode 2*/
    TH1 = 0xFD ;        /*Set Buad rate 9600 bps*/
    TR1 = 1 ;           /*Timer 1 Enable*/
    SCON = 0x50 ;       /*Uart Enable and Set Uart as Model*/
    TI = 1 ;            /*Set state for send*/
}

/*MAIN FUNCTION*/
void main()
{
        uart_init() ;                   /* Baud rate function */
        P0 = 0xF3 ;                     /* set Port0 = F3 */
        P1 = 0xFF ;                     /* set Port1 = FF */
        P2 = 0x00 ;                     /* set Port2 = 0 */

    while(1)
    {
            do                                          /* do while x=0 */
            {
                    if(ip2 == 0)                        /* incorrect login */
                    {
                        P2 = 0x80 ;                     /* Alarm*/
```

```
                    if(Reset == 0)                  /* if Press Reset Button  */
                        {
                            P2 = 0x00 ;             /* close alarm */
                            x = 1 ;                 /* set x = 1 to exit loop */
                            ip2 = 1 ;               /* set ip2 = 1 to exit loop */
                        }
                    else
                        {
                            x = 0 ;                 /* set x = 0 to stay on loop */
                            ip2 = 0 ;               /* set ip2 = 0 to stay on loop
*/
                        }

                }

                else if(nightmode == 0)             /* sensor night on */
                {

                    if(lightcheck == 0)             /* if infrared sensor detect */
                        {
                            P2 = 0x80 ;             /* Alarm*/
                            ip2 = 0 ;               /* set ip2 = 0 to wait reset */
                        }
                    else
                        {
                            x = 0 ;                 /* set ip2 = 0 to stay on loop
*/
                        }
                }

                else if(presscheck == 0)            /* if press emergency button */
                {

                    P2 = 0xD1 ;                     /* Spotlight , Alarm , Wall - a
ctive*/
                    AAA();
                    delay(50);
                    x = 0 ;

                    CCC();
                    x = 0 ;
                    DDD();

                    delay(50) ;
                    x = 0 ;
                    P2 = 0x00 ;
                    delay(10) ;

                }
                else if(ChkS == 0)                      /* check system mode */
                    {

                    if(RI==1)                           /* Recieve ASCII form SBUF
*/
                    {
                    RxBuff = SBUF ;

                    if(RxBuff == 0x4C)                  /* check spotlight  */
                        {
                            delay(240) ;
                            P2 = 0x40 ;                 /* Spotlight on */
                            delay(100) ;

                            ok1 = 0 ;                   /* say to LCD that it's wor
king */
                            delay(650) ;                /* wait LCD recieve */
                            P2 = 0x00 ;                 /* close Spotlight */
```

```
                        ok1 = 1 ;
                        x = 1 ;                     /* set x = 1 to exit loop *
/
                    }
                    else if(RxBuff == 0x41)         /* A */
                    {

                        delay(240) ;
                        P2 = 0x80 ;                 /* Alarm on */
                        delay(100) ;

                        ok1 = 0 ;                   /* say to LCD that it's wor
king */
                        delay(650) ;                /* wait LCD recieve */
                        P2 = 0x00 ;                 /* close Alarm */
                        ok1 = 1 ;
                        x = 1 ;                     /* set x = 1 to exit loop *
/
                    }
                    else if(RxBuff == 0x46)         /* F */
                    {

                        delay(100) ;

                        P2 = 0x28 ;                 /* forward Front door */
                        BBB() ;
                        x = 0 ;
                        delay(200) ;
                        P2 = 0x24 ;                 /* backward front door */
                        EEE() ;
                        delay(10)   ;

                        ok1 = 0 ;
                        delay(450) ;                /* wait LCD recieve */
                        P2 = 0x00 ;                 /* close port */
                        ok1 = 1 ;
                        x = 1 ;                     /* set x = 1 to exit loop *
/
                    }

                    else if(RxBuff == 0x57)         /* W */
                    {
                        delay(240) ;
                        P2 = 0x11 ;                 /* forward Wall up */
                        AAA() ;
                        delay(200) ;
                        x = 0 ;
                        P2 = 0x12 ;                 /* backward Wall down */
                        DDD() ;
                        delay(10) ;

                        ok1 = 0 ;
                        delay(450) ;                /* wait LCD recieve */
                        P2 = 0x00 ;                 /* close port */
                        ok1 = 1 ;
                        x = 1 ;                     /* set x = 1 to exit loop *
/
                    }
                    RI = 0 ;
                }
                }
            else
            {
                x = 0 ;                             /* set x = 0 to stay on loo
p */
            }
        }                       /* } do */
```

```
            while(x==0);
    }                                   /* } while */

}                                       /* } main  */


/*SUB-FUNCTION*/
void AAA()                                      /* Stop wall-up */
{
            do
                {
                    if(misw1fw == 0)            /* Microswitch is pushed */
                        {
                            P2 = 0xD3 ;         /* Fast Stop */
                            x = 1 ;
                        }
                    else
                        {
                            x = 0 ;
                        }
                }
            while(x==0) ;
            return ;
}


void BBB()                                      /* Stop open front door*/
{
            do
                {
                    if(misw2fw == 0)            /* Microswitch is pushed */
                        {
                            P2 = 0xCC ;         /* Fast Stop */
                            x = 1 ;
                        }
                    else
                        {
                            x = 0 ;
                        }
                }
            while(x==0);
            return ;
}

void CCC()
{
            do
            {                                           /* wait for reset*/
                if(Reset == 0)
                    {
                        P2 = 0x12 ;                     /* backward wall down*/
                        x = 1 ;
                    }
                else
                    {
                        x = 0 ;
                    }
            }
            while(x==0);
            return ;
}


void DDD()                                  /* Stop wall down */
{
            do
            {
                if(misw1bw == 0)            /* Stop wall down */
```

```
                    {
                        P2 = 0x13 ;       /* Fast Stop */
                        x = 1 ;
                    }
                else
                    {
                        x = 0 ;
                    }
            }
            while(x==0) ;
            return ;
}

void EEE()                                /* Stop close front door */
{
            do
            {
                if(misw2bw == 0)
                        {
                            P2 = 0x2C ; /* Fast Stop */
                            x = 1 ;
                        }
                    else
                        {
                            x = 0 ;
                        }
            }
            while(x==0) ;
            return ;
}
```

```
#include <reg52.h>
#include <stdio.h>
#include <string.h>

char RxBuff ;                          /* SBUF */
sbit misw1fw = P1^0 ;                  /* stop forward  */
sbit misw2bw = P1^1 ;                  /* stop backward */
sbit ChkL = P1^4 ;                     /* say to microA to check Spotlight  */
sbit ChkA = P1^5 ;                     /* say to microA to check Alarm      */
sbit ChkF = P1^6 ;                     /* say to microA to check Front Door */
sbit ChkW = P1^7 ;                     /* say to microA to check Wall       */
sbit alarm = P2^3 ;                    /* alarm to microA */
sbit reset = P2^4 ;                    /* reset to microA */
sbit ChkS = P2^5 ;                     /* into check mode to microA */
sbit okay = P2^6 ;

sbit RS = P1^4 ;                       /* Register Select pin*/
sbit RW = P1^5 ;                       /* Read / Write pin */
sbit ENA = P1^6 ;                      /* Enable pin */

sbit check = P1^3 ;                    /* Check mode */

sbit D0 = P0^0;                 /* D0 pin */
sbit D1 = P0^1;                 /* D1 pin */
sbit D2 = P0^2;                 /* D2 pin */
sbit D3 = P0^3;                 /* D3 pin */
sbit D4 = P0^4;                 /* D4 pin */
sbit D5 = P0^5;                 /* D5 pin */
sbit D6 = P0^6;                 /* D6 pin */
sbit busy = P0^7;                   /* D7 pin - busy pin */
int x ;
int n ;

void uart_init(void) ;                 /* Baud rate 9600 Mbps */
void delay(unsigned char loop) ;       /*delay function*/
void AAA() ;                           /* stop forward security door */
void BBB() ;                           /* stop backward security door */
void CCC() ;                           /* recieve "R"(reset) form PC
                                          and say to microA to close alarm */
void cmd(unsigned char CMD);       /* write - command */
void dsp(unsigned char alp);       /* write - data */
void lcdready() ;                  /* Check busy flag */
void stand() ;                     /* wait for checking */
void complete();                   /* program's working */
void error() ;                     /* program error */
void welcome();                    /* message welcome */
void logincheck() ;                /* message logincheck */
void loginreset() ;            /* message loginreset */

/*DELAY FUNCTION*/
void delay(unsigned char loop)
{
    unsigned int a,b ;
    for(a=0;a<loop;a++)
    {
        for(b=0;b<1275;b++) ;
    }
}

/* ASSUME BAUD RATE'S SERIAL PORT*/
void uart_init(void)
{
    TR1 = 0 ;              /*Timer 1 Disable*/
    ET1 = 0 ;
    TMOD = 0x20 ;          /*Set Timer 1 as Mode 2*/
    TH1 = 0xFD ;           /*Set Buad rate 9600 bps*/
    TR1 = 1 ;              /*Timer 1 Enable*/
    SCON = 0x50 ;          /*Uart Enable and Set Uart as Model*/
```

```
    TI = 1 ;
}

/*MAIN FUNCTION*/
void main()
{
    uart_init() ;           /* Declare Baud rate function */
    P0 = 0x00 ;
    P1 = 0xF3 ;
    P2 = 0x78 ;
    n = 0 ;
    x = 0 ;

    while(1)
    {
        logincheck() ;
        do
        {
            if(RI==1)
            {
                RxBuff = SBUF ;

                    if(RxBuff == 0x43 )                 /* C */
                    {
                        n = 10 ;
                    }
                    else if(RxBuff == 0x59)             /* Y */
                    {
                        delay(50) ;
                        P2 = 0x7E ;                 /* Open the door */
                        AAA() ;                     /* stop FW */
                        delay(1000) ;               /* SET Open Delay */
                        delay(1000) ;
                        P2 = 0x7D ;                 /* Close the door */
                        x = 0 ;
                        BBB() ;                     /* stop BW */
                        delay(50);

                        P2 = 0x78 ;
                        x = 0 ;
                        n = 0 ;
                    }
                    else if(RxBuff == 0x4E)             /* N */
                    {
                        n++ ;
                    }
                    else
                    {
                        n = n ;
                    }
            RI = 0 ;
            }
        }
        while(n<5);

        x = 0 ;
        delay(10) ;
        if( n == 5 )
            {
                loginreset() ;
                P2 = 0x70 ;                     /* alarm */
                CCC();                          /* wait for reset */
                x = 0 ;
                n = 0 ;
            }
        else if( n == 10 )
        {
            welcome();
```

```
    do
    {

        if(RI==1)
        {
        RxBuff = SBUF ;

            if(RxBuff == 0x42)          /* B */
            {
                x = 1 ;
                stand();
                x = 0 ;
                P2 = 0x7E ;                 /* Open the door */
                AAA() ;                     /* stop FW */
                delay(1500) ;               /* SET Open Delay */
                P2 = 0x7D ;                 /* Close the door */
                x = 0 ;
                BBB() ;                     /* stop BW */
                delay(100) ;

                complete() ;
                P2 = 0x78 ;
                x = 0 ;
                n = n ;
                welcome() ;
            }
            else if(RxBuff == 0x4C)             /* L */
            {
                    ChkS = 0 ;
                    if(TI==1)
                    {
                        printf("L") ;
                        TI = 0 ;
                    }
                    x = 2 ;
                    stand() ;
                    ChkS = 1 ;
                    delay(500) ;
                    delay(120) ;
                    complete() ;
                    n = n ;
                    x = 0 ;
                    delay(50) ;
                    welcome() ;
            }
            else if(RxBuff == 0x41 )            /* A */
            {
                    ChkS = 0 ;
                    if(TI==1)
                    {
                        printf("A") ;
                        TI = 0 ;
                    }
                    x = 3 ;
                    stand() ;
                    ChkS = 1 ;
                    delay(500) ;
                    delay(120) ;
                    complete() ;
                    n = n ;
                    x = 0 ;
                    delay(50) ;
                    welcome() ;
            }
            else if(RxBuff == 0x46 )            /* F */
            {
                    ChkS = 0 ;
                    if(TI==1)
```

```
                {
                    printf("F") ;
                    TI = 0 ;
                }
                x = 4 ;
                stand() ;
                ChkS = 1 ;
                delay(500) ;
                delay(400) ;
                delay(300) ;
                delay(300) ;
                delay(300) ;
                complete() ;
                n = n ;
                x = 0 ;
                delay(50) ;
                welcome() ;
        }
        else if(RxBuff == 0x57 )                /* W */
        {
                ChkS = 0 ;
                if(TI==1)
                {
                    printf("W") ;
                    TI = 0 ;
                }
                x = 5 ;
                stand() ;
                ChkS = 1 ;
                delay(500) ;
                delay(400) ;
                delay(300) ;
                delay(300) ;
                delay(300) ;
                complete() ;
                n = n ;
                x = 0 ;
                delay(50) ;
                welcome() ;
        }

        else if(RxBuff == 0x52)                 /* R */
        {
                    reset = 0 ;
                    delay(10) ;
                    x = 6  ;
                    stand() ;
                    delay(350) ;
                    x = 0 ;
                    P2 = 0x78 ;
                    n = n ;
                    welcome() ;
        }
        else if(RxBuff == 0x58)                 /* X */
        {
                    n = 0 ;
                    delay(10) ;
        }
        RI = 0 ;
        }
        else
            {
                n = n ;
            }
        }

    while(n==10);
    }
```

```
        }

}

/*SUB-FUNCTION*/
void AAA()                                  /* stop forward security door */
{
            do
                {
                    if(misw1fw == 0)        /* microswitch is pushed */
                        {
                            P2 = 0x7F ;     /* fast stop */
                            x = 1 ;
                        }
                    else
                        {
                            x = 0 ;
                        }
                }
            while(x==0) ;
            return ;
}

void BBB()                                   /* stop backward security door */
{
            do
                {
                    if(misw2bw == 0)         /* microswitch is pushed */
                        {
                            P2 = 0x7F ;      /* fast stop */
                            x = 1 ;
                        }
                    else
                        {
                            x = 0 ;
                        }
                }
            while(x==0);
            return ;
}

void CCC()
{
                    do
                    {
                        if(RI==1)
                        {
                            RxBuff = SBUF ;
                            RI = 0 ;
                            if(RxBuff == 0x52)          /* RESET - R */
                                {
                                    P2 = 0x68 ;         /* off appliances */
                                    delay(10) ;
                                    P2 = 0x78 ;         /* set bit 0 */
                                    x = 1 ;
                                }
                            else
                                {
                                    x = 0 ;
                                }
                        }
                    }
                    while(x==0);
                        return ;
}

void cmd(unsigned char CMD)
```

```
{
    lcdready() ;
    P0 = CMD;
    RS = 0 ;
    RW = 0 ;
    ENA = 1 ;
    delay(1) ;
    ENA = 0 ;
    return ;
}

void dsp(unsigned char alp)
{
    lcdready() ;
    P0 = alp ;
    RS = 1;
    RW = 0 ;
    ENA = 1 ;
    delay(1) ;
    ENA = 0 ;
    return ;
}

void lcdready()
{
    busy = 1 ;
    RS = 0 ;
    RW = 1 ;
    while(busy==1)
    {
        ENA = 0 ;
        delay(1);
        ENA =1 ;
    }

}

void stand()
{
        if(x == 1)          /* B */
        {                           /* Checking Program - port
                                          for Security Door    */
            cmd(0x38) ;         /* 2 lines 5x7 matrix */
            cmd(0x0E) ;         /* display on , cursor blinking */
            cmd(0x01) ;         /* clear display screen */
            cmd(0x80) ;         /* force cursor to begening of 1st line */
            dsp('C') ;
            dsp('h') ;
            dsp('e') ;
            dsp('c') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;
            dsp(' ') ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('-') ;
            dsp(' ') ;
            dsp('P') ;
            dsp('o') ;
            dsp('r') ;
```

```
        dsp('t') ;
        cmd(0xC3) ;          /* shift display to position 4 line2 */
        dsp('f') ;
        dsp('o') ;
        dsp('r') ;
        dsp(' ') ;
        dsp('S') ;
        dsp('e') ;
        dsp('c') ;
        dsp('u') ;
        dsp('r') ;
        dsp('i') ;
        dsp('t') ;
        dsp('y') ;
        dsp(' ') ;
        dsp('D') ;
        dsp('o') ;
        dsp('o') ;
        dsp('r') ;

        RS = 0 ;
        RW = 0 ;
        ENA = 0 ;
        x = 0 ;              /* send to security door's loop */
        return ;
    }
    else if(x == 2)          /* L */
    {                        /* Checking Program - port
                                   for Spotlight             */
        cmd(0x38) ;          /* 2 lines 5x7 matrix */
        cmd(0x0E) ;          /* display on , cursor blinking */
        cmd(0x01) ;          /* clear display screen */
        cmd(0x80) ;          /* force cursor to begening of 1st line */
        dsp('C') ;
        dsp('h') ;
        dsp('e') ;
        dsp('c') ;
        dsp('k') ;
        dsp('i') ;
        dsp('n') ;
        dsp('g') ;
        dsp(' ') ;
        dsp('P') ;
        dsp('r') ;
        dsp('o') ;
        dsp('g') ;
        dsp('r') ;
        dsp('a') ;
        dsp('m') ;
        dsp(' ') ;
        dsp('-') ;
        dsp(' ') ;
        dsp('P') ;
        dsp('o') ;
        dsp('r') ;
        dsp('t') ;
        cmd(0xC5) ;          /* shift display to position 6 line2 */
        dsp('f') ;
        dsp('o') ;
        dsp('r') ;
        dsp(' ') ;
        dsp('S') ;
        dsp('p') ;
        dsp('o') ;
        dsp('t') ;
        dsp('l') ;
        dsp('i') ;
        dsp('g') ;
```

```
        dsp('h') ;
        dsp('t') ;

        delay(400);          /* set delay for waitng ok1 for micorA */

        return ;
    }
    else if(x == 3)         /* A */
    {                        /* Checking Program - port
                                    for Alarm            */
        cmd(0x38) ;          /* 2 lines 5x7 matrix */
        cmd(0x0E) ;          /* display on , cursor blinking */
        cmd(0x01) ;          /* clear display screen */
        cmd(0x80) ;          /* force cursor to begening of 1st line */
        dsp('C') ;
        dsp('h') ;
        dsp('e') ;
        dsp('c') ;
        dsp('k') ;
        dsp('i') ;
        dsp('n') ;
        dsp('g') ;
        dsp(' ') ;
        dsp('P') ;
        dsp('r') ;
        dsp('o') ;
        dsp('g') ;
        dsp('r') ;
        dsp('a') ;
        dsp('m') ;
        dsp(' ') ;
        dsp('-') ;
        dsp(' ') ;
        dsp('P') ;
        dsp('o') ;
        dsp('r') ;
        dsp('t') ;
        cmd(0xC8) ;          /* shift display to position 9 line2 */
        dsp('f') ;
        dsp('o') ;
        dsp('r') ;
        dsp(' ') ;
        dsp('A') ;
        dsp('l') ;
        dsp('a') ;
        dsp('r') ;
        dsp('m') ;

        delay(400);          /* set delay for waitng ok1 for micorA */

        return ;
    }
    else if(x == 4)         /* F */
    {                        /* Checking Program - port
                                    for Front Door       */
        cmd(0x38) ;          /* 2 lines 5x7 matrix */
        cmd(0x0E) ;          /* display on , cursor blinking */
        cmd(0x01) ;          /* clear display screen */
        cmd(0x80) ;          /* force cursor to begening of 1st line */
        dsp('C') ;
        dsp('h') ;
        dsp('e') ;
        dsp('c') ;
        dsp('k') ;
        dsp('i') ;
        dsp('n') ;
        dsp('g') ;
        dsp(' ') ;
```

```
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('-') ;
            dsp(' ') ;
            dsp('P') ;
            dsp('o') ;
            dsp('r') ;
            dsp('t') ;
            cmd(0xC5) ;          /* shift display to position 6 line2 */
            dsp('f') ;
            dsp('o') ;
            dsp('r') ;
            dsp(' ') ;
            dsp('F') ;
            dsp('r') ;
            dsp('o') ;
            dsp('n') ;
            dsp('t') ;
            dsp(' ') ;
            dsp('D') ;
            dsp('o') ;
            dsp('o') ;
            dsp('r') ;

            delay(700);          /* set delay for waitng ok1 for micorA */

            return ;
        }
        else if(x == 5)         /* W */
        {                        /* Checking Program - port
                                       for Wall          */
            cmd(0x38) ;          /* 2 lines 5x7 matrix */
            cmd(0x0E) ;          /* display on , cursor blinking */
            cmd(0x01) ;          /* clear display screen */
            cmd(0x80) ;          /* force cursor to begening of 1st line */
            dsp('C') ;
            dsp('h') ;
            dsp('e') ;
            dsp('c') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;
            dsp(' ') ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('-') ;
            dsp(' ') ;
            dsp('P') ;
            dsp('o') ;
            dsp('r') ;
            dsp('t') ;
            cmd(0xC6) ;          /* shift display to position 7 line2 */
            dsp('f') ;
            dsp('o') ;
            dsp('r') ;
```

```
            dsp(' ') ;
            dsp('W') ;
            dsp('a') ;
            dsp('l') ;
            dsp('l') ;


            delay(600);            /* set delay for waitng ok1 for micorA */
            return ;
        }
        else if(x == 6)          /* RESUME system */
        {
            cmd(0x38) ;            /* 2 lines 5x7 matrix */
            cmd(0x0E) ;            /* display on , cursor blinking */
            cmd(0x01) ;            /* clear display screen */
            cmd(0x87) ;            /* force cursor to begening of 1st line */
            dsp('S') ;
            dsp('y') ;
            dsp('s') ;
            dsp('t') ;
            dsp('e') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('h') ;
            dsp('a') ;
            dsp('s') ;
            cmd(0xC7) ;            /* shift display to position 7 line2 */
            dsp('b') ;
            dsp('e') ;
            dsp('e') ;
            dsp('n') ;
            dsp(' ') ;
            dsp('R') ;
            dsp('e') ;
            dsp('s') ;
            dsp('e') ;
            dsp('t') ;

            delay(500);            /* set delay for waitng ok1 for micorA */
            return ;
        }
        else
        {
            error();
            x = 0 ;
        }

}




            /*COMPLETE*/
void complete()
{
    if(x==1)
    {
        if(misw2bw == 0)
            {
            cmd(0x38) ;      /* 2 lines 5x7 matrix */
            cmd(0x0E) ;      /* display on , cursor blinking */
            cmd(0x01) ;      /* clear display screen */
            cmd(0x84) ;      /* shift display to position 5 line 1 */
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
```

```
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('S') ;
            dsp('e') ;
            dsp('c') ;
            dsp('u') ;
            dsp('r') ;
            dsp('i') ;
            dsp('t') ;
            dsp('y') ;
            cmd(0xC3) ;
            dsp('D') ;
            dsp('o') ;
            dsp('o') ;
            dsp('r') ;
            dsp(' ') ;
            dsp('S') ;
            dsp('t') ;
            dsp('i') ;
            dsp('l') ;
            dsp('l') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('o') ;
            dsp('r') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;

            delay(500) ;
            return ;
            }
        else if(misw2bw == 1)
            {
                error() ;
            }
    }

    else if(x==2)
    {
        if(okay == 0)           /* microB send okay */
        {
            cmd(0x38) ;         /* 2 lines 5x7 matrix */
            cmd(0x0E) ;         /* display on , cursor blinking */
            cmd(0x01) ;         /* clear display screen */
            cmd(0x83) ;         /* shift display to position 4 line 1 */
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp(' ') ;
            dsp('S') ;
            dsp('p') ;
            dsp('o') ;
            dsp('t') ;
            dsp('l') ;
            dsp('i') ;
            dsp('g') ;
            dsp('h') ;
            dsp('t') ;
```

```
            cmd(0xC6) ;
            dsp('S') ;
            dsp('t') ;
            dsp('i') ;
            dsp('l') ;
            dsp('l') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('o') ;
            dsp('r') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;

            delay(500) ;
            return ;
        }
        else if(okay == 1)
        {
            error() ;
        }
    }

    else if(x == 3)
    {
        if(okay == 0)
        {
            cmd(0x38) ;
            cmd(0x0E) ;
            cmd(0x01) ;
            cmd(0x85) ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp(' ') ;
            dsp('A') ;
            dsp('l') ;
            dsp('a') ;
            dsp('r') ;
            dsp('m') ;
            cmd(0xC6) ;
            dsp('S') ;
            dsp('t') ;
            dsp('i') ;
            dsp('l') ;
            dsp('l') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('o') ;
            dsp('r') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;

            delay(500) ;
            return ;
        }
        else if (okay == 1)
        {
            error() ;
        }
```

```
    }
    else if(x == 4)
    {
        if(okay == 0)
        {
            cmd(0x38) ;
            cmd(0x0E) ;
            cmd(0x01) ;
            cmd(0x83) ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('F') ;
            dsp('r') ;
            dsp('o') ;
            dsp('n') ;
            dsp('t') ;
            dsp(' ') ;
            dsp('D') ;
            dsp('o') ;
            dsp('o') ;
            dsp('r') ;
            cmd(0xC6) ;
            dsp('S') ;
            dsp('t') ;
            dsp('i') ;
            dsp('l') ;
            dsp('l') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('o') ;
            dsp('r') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;

            delay(500) ;
            return ;
        }
        else if(okay == 1)
        {
            error() ;
        }
    }
    else if(x == 5)
    {
        if(okay == 0)
        {
            cmd(0x38) ;
            cmd(0x0E) ;
            cmd(0x01) ;
            cmd(0x86) ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('a') ;
```

```
            dsp('l') ;
            dsp('l') ;
            cmd(0xC6) ;
            dsp('S') ;
            dsp('t') ;
            dsp('i') ;
            dsp('l') ;
            dsp('l') ;
            dsp(' ') ;
            dsp('W') ;
            dsp('o') ;
            dsp('r') ;
            dsp('k') ;
            dsp('i') ;
            dsp('n') ;
            dsp('g') ;

            delay(500) ;
            return ;
        }
        else if(okay == 1)
        {
            error() ;
        }
    }
        else
        {
            ENA = 0 ;
            RS = 0 ;
            RW = 0 ;
            x = x ;
            delay(1) ;
        }
}


/* ERROR */
void error()
{
            cmd(0x01) ;
            cmd(0x82) ;
            dsp('P') ;
            dsp('r') ;
            dsp('o') ;
            dsp('g') ;
            dsp('r') ;
            dsp('a') ;
            dsp('m') ;
            dsp('-') ;
            dsp('s') ;
            dsp(' ') ;
            dsp('s') ;
            dsp('y') ;
            dsp('s') ;
            dsp('t') ;
            dsp('e') ;
            dsp('m') ;
            dsp(' ') ;
            dsp('h') ;
            dsp('a') ;
            dsp('v') ;
            dsp('e') ;
            cmd(0xC2) ;
            dsp('e') ;
            dsp('r') ;
            dsp('r') ;
            dsp('o') ;
            dsp('r') ;
```

```
            dsp(',') ;
            dsp('P') ;
            dsp('l') ;
            dsp('e') ;
            dsp('a') ;
            dsp('s') ;
            dsp('e') ;
            dsp(' ') ;
            dsp('r') ;
            dsp('e') ;
            dsp('c') ;
            dsp('h') ;
            dsp('e') ;
            dsp('c') ;
            dsp('k') ;

            delay(500) ;
            return ;
}

void welcome()
{
        cmd(0x38) ;
        cmd(0x0E) ;
        cmd(0x01) ;
        cmd(0x80) ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('W') ;
        dsp('E') ;
        dsp('L') ;
        dsp('C') ;
        dsp('O') ;
        dsp('M') ;
        dsp('E') ;
        dsp(' ') ;
        dsp('P') ;
        dsp('L') ;
        dsp('E') ;
        dsp('A') ;
        dsp('S') ;
        dsp('E') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;

        cmd(0xC0) ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('S') ;
        dsp('E') ;
        dsp('L') ;
        dsp('E') ;
        dsp('C') ;
        dsp('T') ;
        dsp(' ') ;
        dsp('C') ;
        dsp('H') ;
        dsp('E') ;
        dsp('C') ;
        dsp('K') ;
```

```
        dsp('I') ;
        dsp('N') ;
        dsp('G') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;

        RS = 0 ;
        RW = 0 ;
        ENA = 0 ;
}

void logincheck()
{
        cmd(0x38) ;
        cmd(0x0E) ;
        cmd(0x01) ;
        cmd(0x80) ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('P') ;
        dsp('L') ;
        dsp('E') ;
        dsp('A') ;
        dsp('S') ;
        dsp('E') ;
        dsp(' ') ;
        dsp('L') ;
        dsp('O') ;
        dsp('G') ;
        dsp('I') ;
        dsp('N') ;
        dsp(' ') ;
        dsp('F') ;
        dsp('O') ;
        dsp('R') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;

        cmd(0xC0) ;
        dsp('*') ;
        dsp('*') ;
        dsp('C') ;
        dsp('H') ;
        dsp('E') ;
        dsp('C') ;
        dsp('K') ;
        dsp('I') ;
        dsp('N') ;
        dsp('G') ;
        dsp(' ') ;
        dsp(' ') ;
        dsp('A') ;
        dsp('P') ;
        dsp('P') ;
        dsp('L') ;
        dsp('I') ;
        dsp('A') ;
        dsp('N') ;
        dsp('C') ;
        dsp('E') ;
        dsp('S') ;
        dsp('*') ;
```

```
        dsp('*') ;

        RS = 0 ;
        RW = 0 ;
        ENA = 0 ;
}

void loginreset()
{
        cmd(0x38) ;
        cmd(0x0E) ;
        cmd(0x01) ;
        cmd(0x80) ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('P') ;
        dsp('L') ;
        dsp('E') ;
        dsp('A') ;
        dsp('S') ;
        dsp('E') ;
        dsp(' ') ;
        dsp('L') ;
        dsp('O') ;
        dsp('G') ;
        dsp('I') ;
        dsp('N') ;
        dsp(' ') ;
        dsp('F') ;
        dsp('O') ;
        dsp('R') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;

        cmd(0xC0);
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;
        cmd(0xC4);
        dsp('R') ;
        dsp('E') ;
        dsp('S') ;
        dsp('E') ;
        dsp('T') ;
        dsp(' ') ;
        dsp('A') ;
        dsp('P') ;
        dsp('P') ;
        dsp('L') ;
        dsp('I') ;
        dsp('A') ;
        dsp('N') ;
        dsp('C') ;
        dsp('E') ;
        dsp('S') ;
        dsp('*') ;
        dsp('*') ;
        dsp('*') ;

        RS = 0 ;
        RW = 0 ;
        ENA = 0 ;
}
```

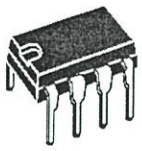
ภาคผนวก

ข

NE555
SA555 - SE555

GENERAL PURPOSE SINGLE BIPOLAR TIMERS

- LOW TURN OFF TIME
- MAXIMUM OPERATING FREQUENCY GREATER THAN 500kHz
- TIMING FROM MICROSECONDS TO HOURS
- OPERATES IN BOTH ASTABLE AND MONOSTABLE MODES
- HIGH OUTPUT CURRENT CAN SOURCE OR SINK 200mA
- ADJUSTABLE DUTY CYCLE
- TTL COMPATIBLE
- TEMPERATURE STABILITY OF 0.005% PER°C



N
DIP8
(Plastic Package)

D
SO8
(Plastic Micropackage)

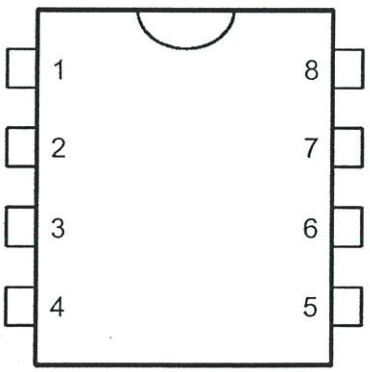
DESCRIPTION

The NE555 monolithic timing circuit is a highly stable controller capable of producing accurate time delays or oscillation. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200mA. The NE555 is available in plastic and ceramic minidip package and in a 8-lead micropackage and in metal can package version.

ORDER CODES

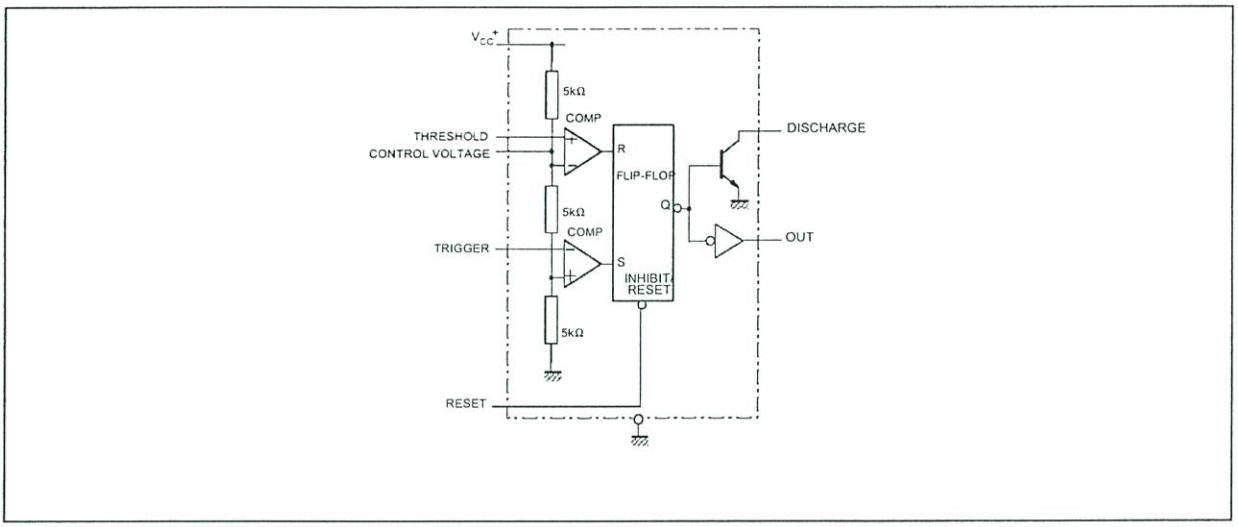
Part Number	Temperature Range	Package	
		N	D
NE555	0°C, 70°C	•	•
SA555	–40°C, 105°C	•	•
SE555	–55°C, 125°C	•	•

PIN CONNECTIONS (top view)

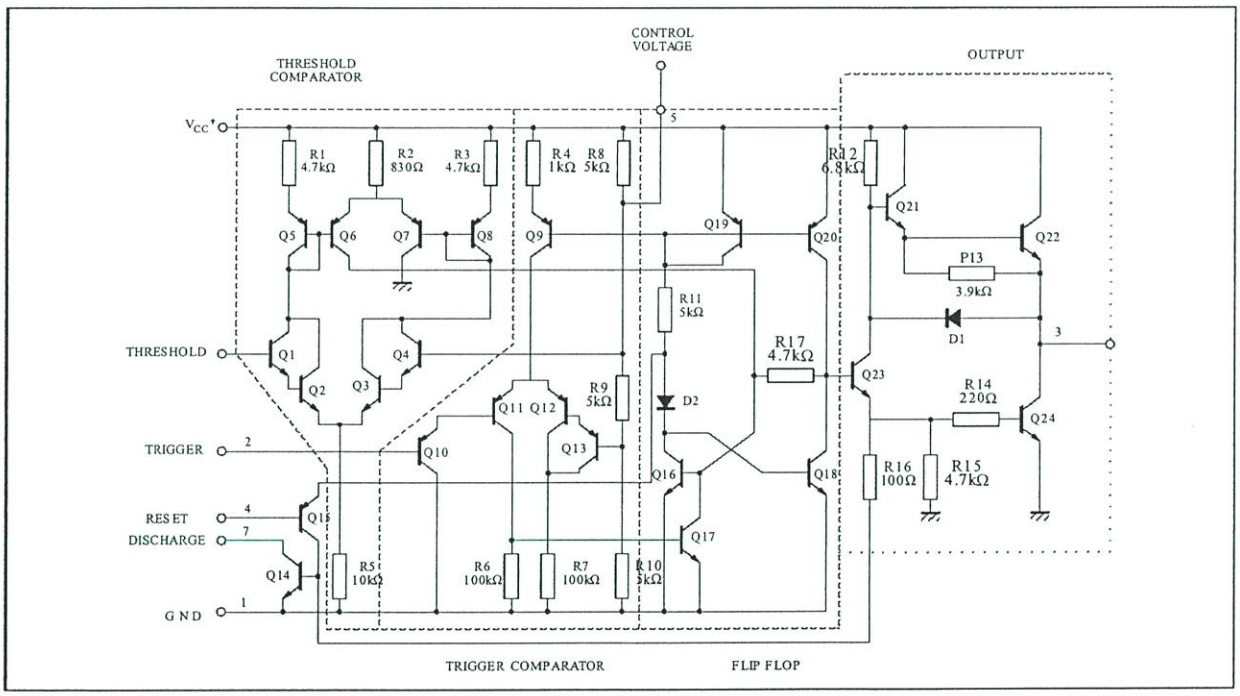


1 - GND
2 - Trigger
3 - Output
4 - Reset
5 - Control voltage
6 - Threshold
7 - Discharge
8 - V_{CC}

BLOCK DIAGRAM



SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter		Value	Unit
V_{cc}	Supply Voltage		18	V
T_{oper}	Operating Free Air Temperature Range	for NE555 for SA555 for SE555	0 to 70 –40 to 105 –55 to 125	°C
T_j	Junction Temperature		150	°C
T_{stg}	Storage Temperature Range		–65 to 150	°C

OPERATING CONDITIONS

Symbol	Parameter	SE555	NE555 - SA555	Unit
V_{CC}	Supply Voltage	4.5 to 18	4.5 to 18	V
V_{th}, V_{trig}, V_{cl}, V_{reset}	Maximum Input Voltage	V_{CC}	V_{CC}	V

ELECTRICAL CHARACTERISTICS

T_{amb} = +25°C, V_{CC} = +5V to +15V (unless otherwise specified)

Symbol	Parameter	SE555			NE555 - SA555			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
I_{CC}	Supply Current (R_L ∞) (- note 1) Low State V_{CC} = +5V V_{CC} = +15V High State V_{CC} = 5V		 3 10 2	 5 12		 3 10 2	 6 15	mA
	Timing Error (monostable) (R_A = 2k to 100kΩ, C = 0.1µF) Initial Accuracy - (note 2) Drift with Temperature Drift with Supply Voltage		 0.5 30 0.05	 2 100 0.2		 1 50 0.1	 3 0.5	 % ppm/°C %/V
	Timing Error (astable) (R_A, R_B = 1kΩ to 100kΩ, C = 0.1µF, V_{CC} = +15V) Initial Accuracy - (note 2) Drift with Temperature Drift with Supply Voltage		 1.5 90 0.15			 2.25 150 0.3		 % ppm/°C %/V
V_{CL}	Control Voltage level V_{CC} = +15V V_{CC} = +5V	 9.6 2.9	 10 3.33	 10.4 3.8	 9 2.6	 10 3.33	 11 4	V
V_{th}	Threshold Voltage V_{CC} = +15V V_{CC} = +5V	 9.4 2.7	 10 3.33	 10.6 4	 8.8 2.4	 10 3.33	 11.2 4.2	V
I_{th}	Threshold Current - (note 3)		0.1	0.25		0.1	0.25	µA
V_{trig}	Trigger Voltage V_{CC} = +15V V_{CC} = +5V	 4.8 1.45	 5 1.67	 5.2 1.9	 4.5 1.1	 5 1.67	 5.6 2.2	V
I_{trig}	Trigger Current (V_{trig} = 0V)		0.5	0.9		0.5	2.0	µA
V_{reset}	Reset Voltage - (note 4)	0.4	0.7	1	0.4	0.7	1	V
I_{reset}	Reset Current V_{reset} = +0.4V V_{reset} = 0V		 0.1 0.4	 0.4 1		 0.1 0.4	 0.4 1.5	mA
V_{OL}	Low Level Output Voltage V_{CC} = +15V, $I_{O(sink)}$ = 10mA $I_{O(sink)}$ = 50mA $I_{O(sink)}$ = 100mA $I_{O(sink)}$ = 200mA V_{CC} = +5V, $I_{O(sink)}$ = 8mA $I_{O(sink)}$ = 5mA		 0.1 0.4 2 2.5 0.1 0.05	 0.15 0.5 2.2 0.25 0.2		 0.1 0.4 2 2.5 0.3 0.25	 0.25 0.75 2.5 0.4 0.35	V
V_{OH}	High Level Output Voltage V_{CC} = +15V, $I_{O(source)}$ = 200mA $I_{O(source)}$ = 100mA V_{CC} = +5V, $I_{O(source)}$ = 100mA	 13 3	 12.5 13.3 3.3		 12.75 2.75	 12.5 13.3 3.3		V

Notes :
1. Supply current when output is high is typically 1mA less.
2. Tested at V_{CC} = +5V and V_{CC} = +15V.
3. This will determine the maximum value of R_A + R_B for +15V operation the max total is R = 20MΩ and for 5V operation the max total R = 3.5MΩ.

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	SE555			NE555 - SA555			Unit
		Min.	Typ.	Max.	Min.	Typ.	Max.	
$I_{dis(off)}$	Discharge Pin Leakage Current (output high) (V_{dis} = 10V)		20	100		20	100	nA
$V_{dis(sat)}$	Discharge pin Saturation Voltage (output low) - (note 5) V_{CC} = +15V, I_{dis} = 15mA V_{CC} = +5V, I_{dis} = 4.5mA		 180 80	 480 200		 180 80	 480 200	mV
t_r t_f	Output Rise Time Output Fall Time		100 100	200 200		100 100	300 300	ns
t_{off}	Turn off Time - (note 6) (V_{reset} = V_{CC})		0.5			0.5		µs

Notes : 5. No protection against excessive Pin 7 current is necessary, providing the package dissipation rating will not be exceeded.
6. Time mesaured from a positive going input pulse from 0 to 0.8x V_{CC} into the threshold to the drop from high to low of the output trigger is tied to treshold.

Figure 1 : Minimum Pulse Width Required for Trigering

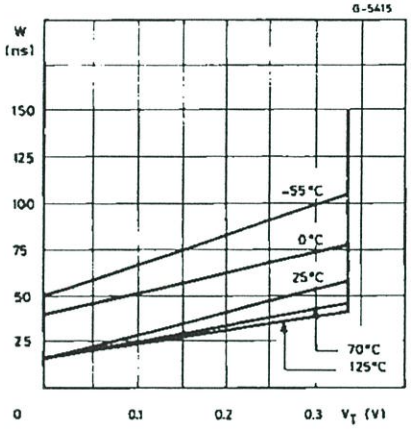


Figure 2 : Supply Current versus Supply Voltage

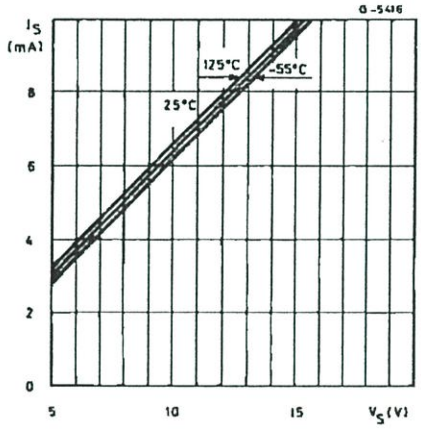


Figure 3 : Delay Time versus Temperature

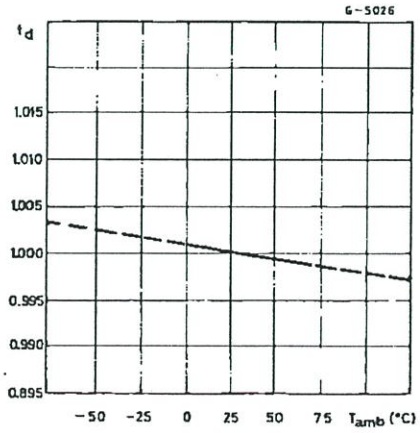


Figure 4 : Low Output Voltage versus Output Sink Current

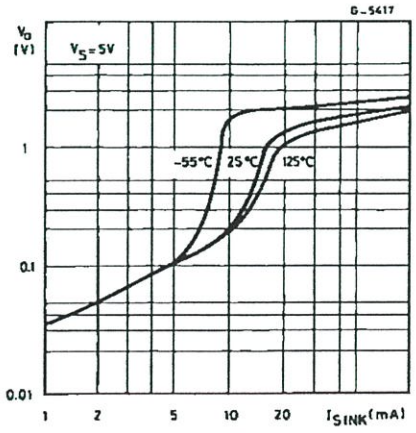


Figure 5 : Low Output Voltage versus Output Sink Current

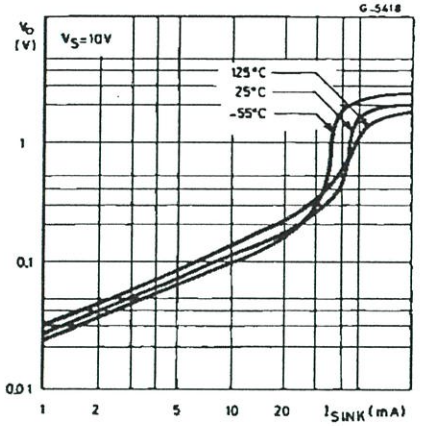


Figure 6 : Low Output Voltage versus Output Sink Current

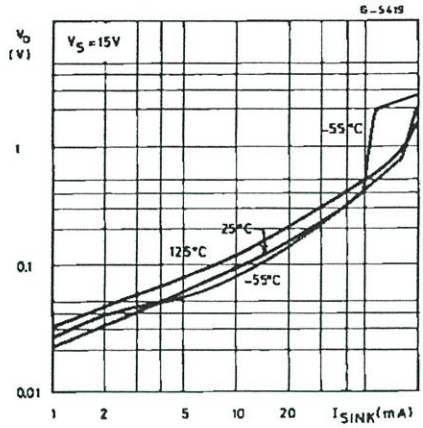


Figure 7 : High Output Voltage Drop versus Output

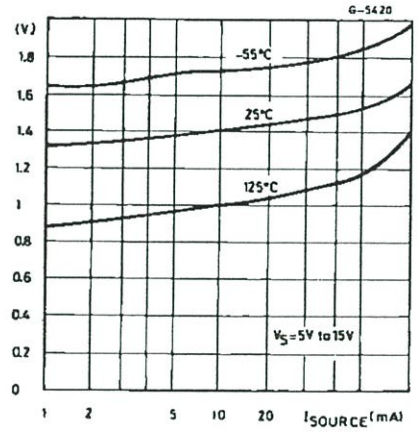


Figure 8 : Delay Time versus Supply Voltage

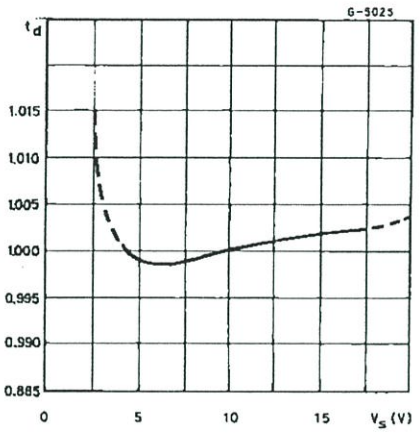
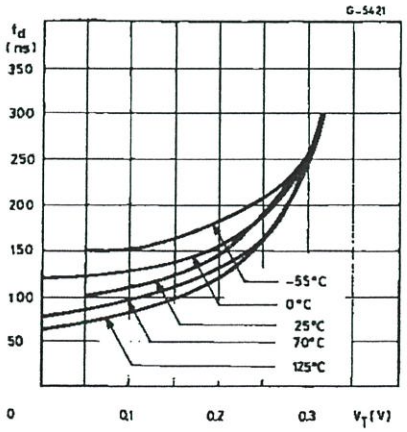


Figure 9 : Propagation Delay versus Voltage Level of Trigger Value

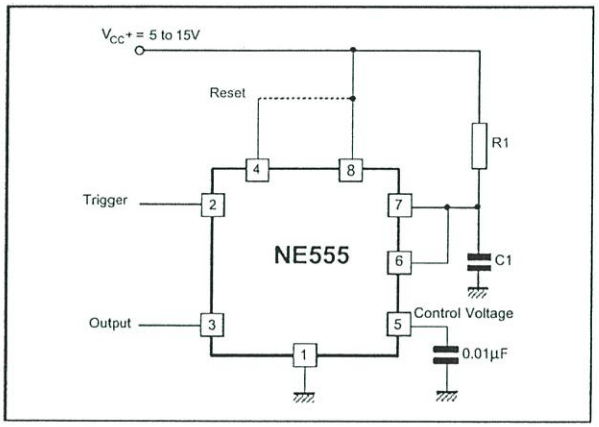


APPLICATION INFORMATION

MONOSTABLE OPERATION

In the monostable mode, the timer functions as a one-shot. Referring to figure 10 the external capacitor is initially held discharged by a transistor inside the timer.

Figure 10



The circuit triggers on a negative-going input signal when the level reaches 1/3 Vcc. Once triggered, the circuit remains in this state until the set time has elapsed, even if it is triggered again during this interval. The duration of the output HIGH state is given by $t = 1.1\ R_1C_1$ and is easily determined by figure 12.

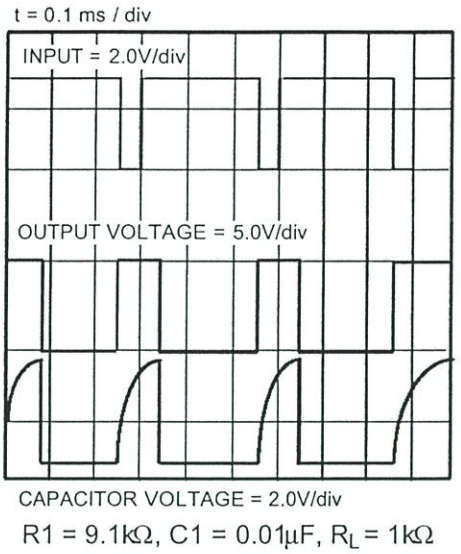
Notice that since the charge rate and the threshold level of the comparator are both directly proportional to supply voltage, the timing interval is independent of supply. Applying a negative pulse simultaneously to the reset terminal (pin 4) and the trigger terminal (pin 2) during the timing cycle discharges the external capacitor and causes the cycle to start over. The timing cycle now starts on the positive edge of the reset pulse. During the time the reset pulse in applied, the output is driven to its LOW state.

When a negative trigger pulse is applied to pin 2, the flip-flop is set, releasing the short circuit across the external capacitor and driving the output HIGH. The voltage across the capacitor increases exponentially with the time constant $\tau = R_1C_1$. When the voltage across the capacitor equals 2/3 V_{cc}, the comparator resets the flip-flop which then discharge the capacitor rapidly and drivers the output to its LOW state.

Figure 11 shows the actual waveforms generated in this mode of operation.

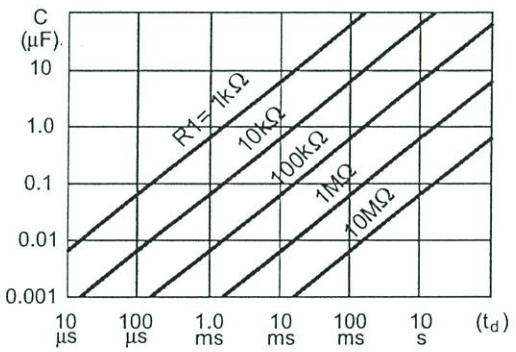
When Reset is not used, it should be tied high to avoid any possibly or false triggering.

Figure 11



$R1 = 9.1k\Omega$, $C1 = 0.01\mu F$, $R_L = 1k\Omega$

Figure 12



ASTABLE OPERATION

When the circuit is connected as shown in figure 13 (pin 2 and 6 connected) it triggers itself and free runs as a multivibrator. The external capacitor charges through R_1 and R_2 and discharges through R_2 only. Thus the duty cycle may be precisely set by the ratio of these two resistors.

In the astable mode of operation, C_1 charges and discharges between 1/3 V_{cc} and 2/3 Vcc. As in the triggered mode, the charge and discharge times and therefore frequency are independent of the supply voltage.

Figure 13

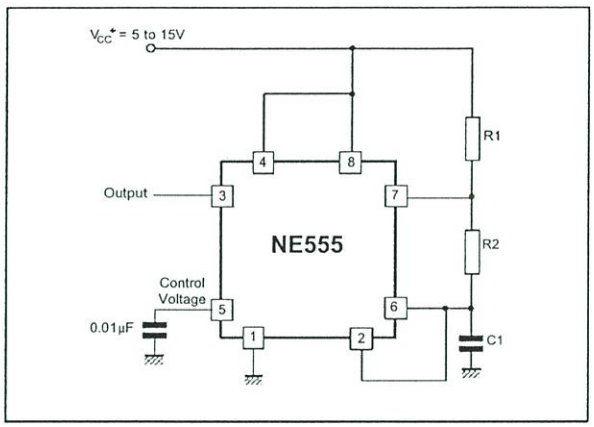


Figure 14 shows actual waveforms generated in this mode of operation.

The charge time (output HIGH) is given by :

$t_1 = 0.693 (R_1 + R_2) C_1$

and the discharge time (output LOW) by :

$t_2 = 0.693 (R_2) C_1$

Thus the total period T is given by :

$T = t_1 + t_2 = 0.693 (R_1 + 2R_2) C_1$

The frequency ofoscillation is them :

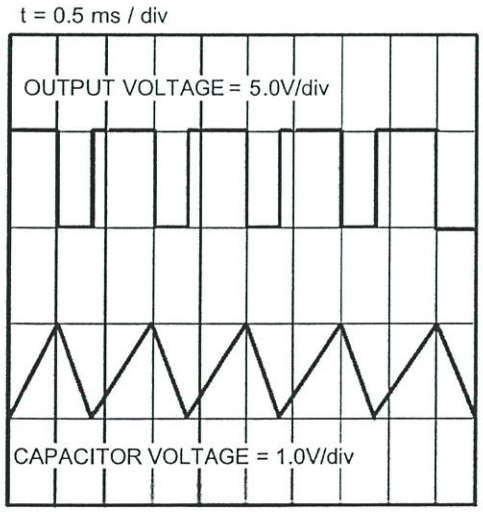
$$f = \frac{1}{T} = \frac{1.44}{(R_1 + 2R_2) C_1}$$

and may be easily found by figure 15.

The duty cycle is given by :

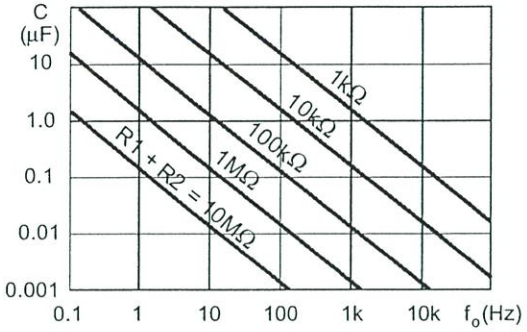
$$D = \frac{R_2}{R_1 + 2R_2}$$

Figure 14



$R1 = R2 = 4.8k\Omega$, $C1 = 0.1\mu F$, $R_L = 1k\Omega$

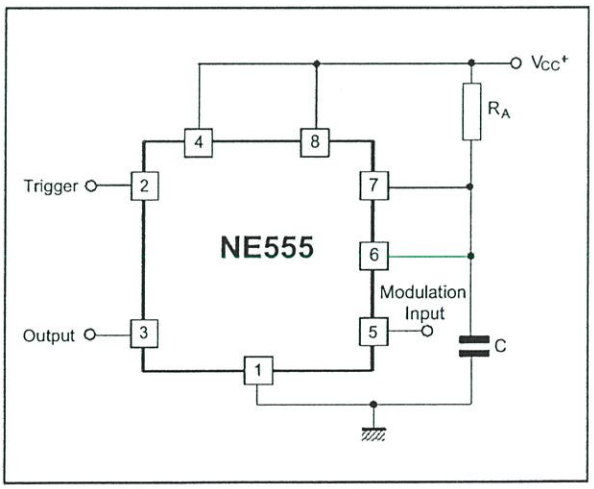
Figure 15 : Free Running Frequency versus R_1, R_2 and C_1



PULSE WIDTH MODULATOR

When the timer is connected in the monostable mode and triggered with a continuous pulse train, the output pulse width can be modulated by a signal applied to pin 5. Figure 16 shows the circuit.

Figure 16 : Pulse Width Modulator.



LINEAR RAMP

When the pullup resistor, R_A, in the monostable circuit is replaced by a constant current source, a linear ramp is generated. Figure 17 shows a circuit configuration that will perform this function.

Figure 17.

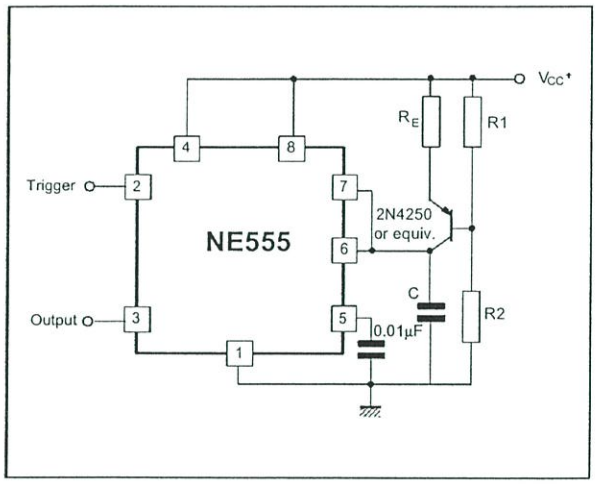
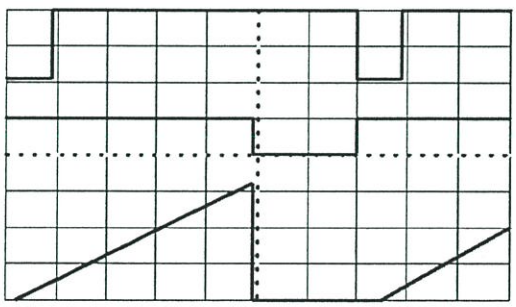


Figure 18 shows waveforms generator by the linear ramp.

The time interval is given by :

$$T = \frac{(2/3\ V_{CC}\ R_E\ (R_{1+}\ R_2)\ C}{R_1\ V_{CC} - V_{BE}\ (R_{1+}\ R_2)} \quad V_{BE} = 0.6V$$

Figure 18 : Linear Ramp.



V_{CC} = 5V
Time = 20µs/DIV
R_1 = 47kΩ
R_2 = 100kΩ
R_E = 2.7kΩ
C = 0.01µF

Top trace : input 3V/DIV
Middle trace : output 5V/DIV
Bottom trace : output 5V/DIV
Bottom trace : capacitor voltage 1V/DIV

50% DUTY CYCLE OSCILLATOR

For a 50% duty cycle the resistors R_A and R_E may be connected as in figure 19. The time preriod for the output high is the same as previous,

$t_1 = 0.693\ R_A\ C$.

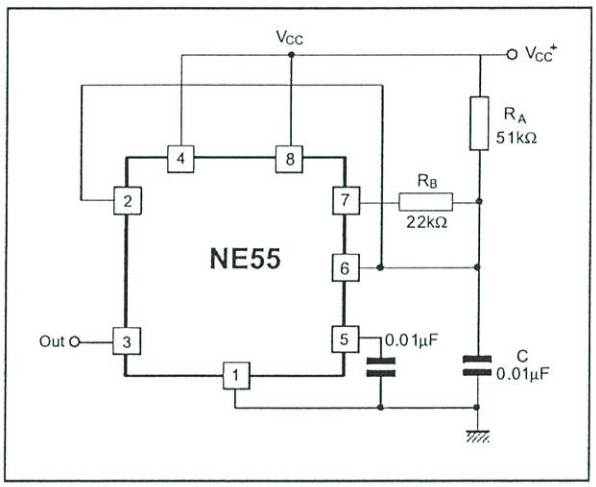
For the output low it is t_2 =

$$[(R_A R_B)/(R_A + R_B)]\ C \ln\left[\frac{R_B - 2R_A}{2R_B - R_A}\right]$$

Thus the frequency of oscillation is $f = \frac{1}{t_1 + t_2}$

Note that this circuit will not oscillate if R_B is greater

Figure 19 : 50% Duty Cycle Oscillator.

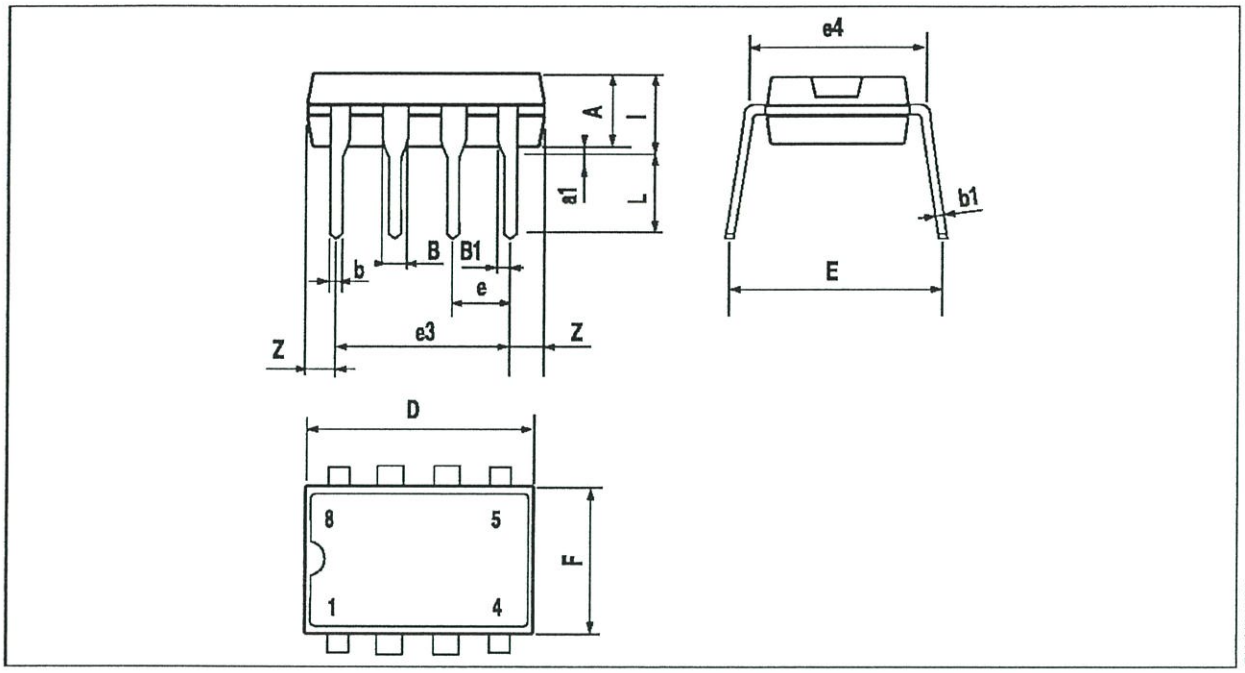


than 1/2 R_A because the junction of R_A and R_B cannot bring pin 2 down to 1/3 V_{CC} and trigger the lower comparator.

ADDITIONAL INFORMATION

Adequate power supply bypassing is necessary to protect associated circuitry. Minimum recommended is 0.1µF in parallel with 1µF electrolytic.

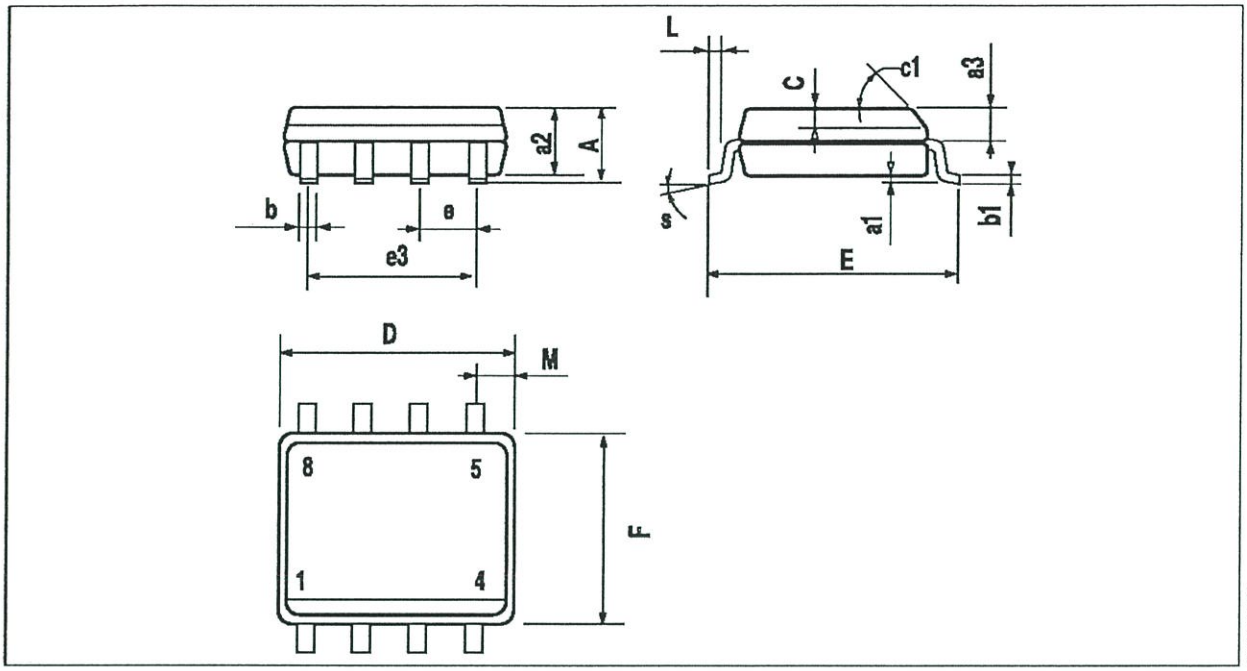
PACKAGE MECHANICAL DATA
8 PINS - PLASTIC DIP



Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A		3.32			0.131	
a1	0.51			0.020		
B	1.15		1.65	0.045		0.065
b	0.356		0.55	0.014		0.022
b1	0.204		0.304	0.008		0.012
D			10.92			0.430
E	7.95		9.75	0.313		0.384
e		2.54			0.100	
e3		7.62			0.300	
e4		7.62			0.300	
F			6.6			0260
i			5.08			0.200
L	3.18		3.81	0.125		0.150
Z			1.52			0.060

DIP8.TBL

PACKAGE MECHANICAL DATA
8 PINS - PLASTIC MICROPACKAGE (SO)



PM-SO8.EPS

Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.75			0.069
a1	0.1		0.25	0.004		0.010
a2			1.65			0.065
a3	0.65		0.85	0.026		0.033
b	0.35		0.48	0.014		0.019
b1	0.19		0.25	0.007		0.010
C	0.25		0.5	0.010		0.020
c1	45° (typ.)					
D	4.8		5.0	0.189		0.197
E	5.8		6.2	0.228		0.244
e		1.27			0.050	
e3		3.81			0.150	
F	3.8		4.0	0.150		0.157
L	0.4		1.27	0.016		0.050
M			0.6			0.024
S	8° (max.)					

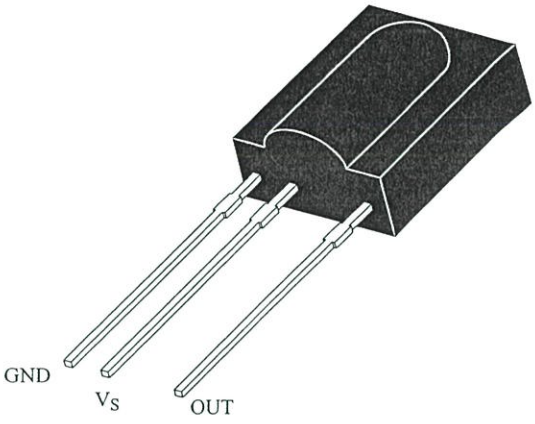
SO8.TBL

ORDER CODE :

Photo Modules for PCM Remote Control Systems

Description

The HS0038A2. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. HS0038A2. is the standard IR remote control receiver series, supporting all major transmission codes.

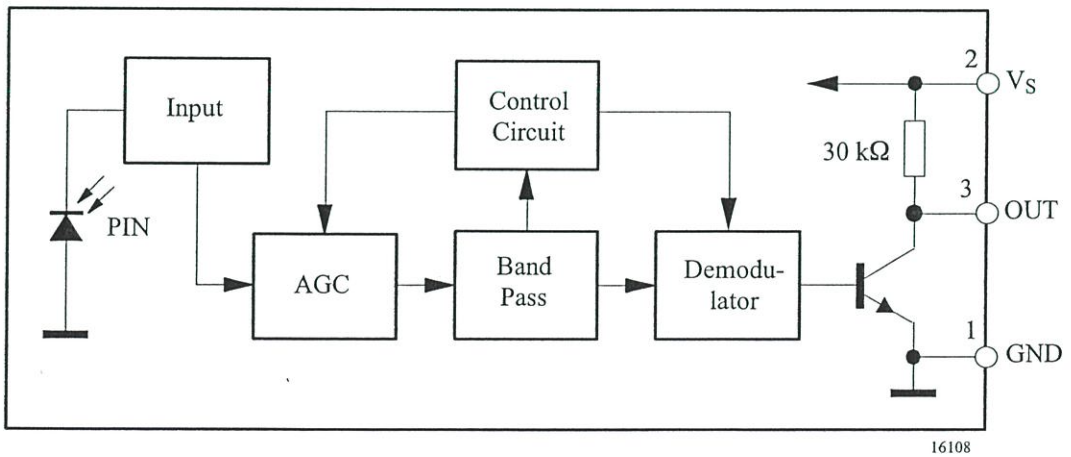


94 8691

Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (800 bit/s)
- Suitable burst length ≥10 cycles/burst

Block Diagram



16108

Absolute Maximum Ratings

T_{amb} = 25°C

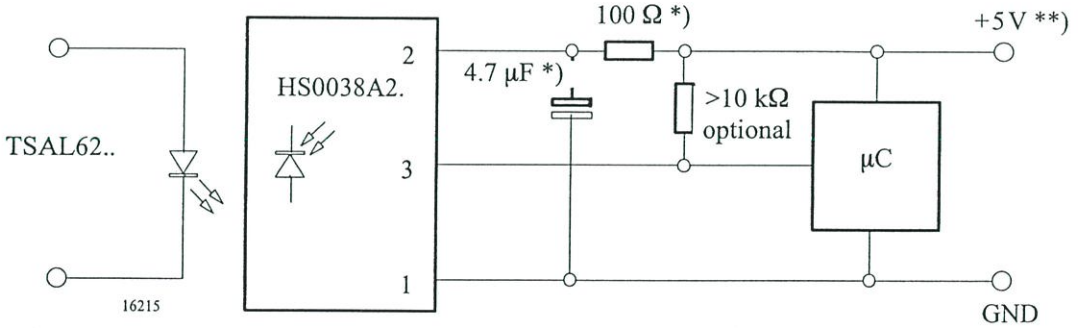
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 2)	V_S	–0.3...6.0	V
Supply Current	(Pin 2)	I_S	5	mA
Output Voltage	(Pin 3)	V_O	–0.3...6.0	V
Output Current	(Pin 3)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	–25...+85	°C
Operating Temperature Range		T_{amb}	–25...+85	°C
Power Consumption	(T_{amb} ≦ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≦ 10 s, 1 mm from case	T_{sd}	260	°C

Basic Characteristics

T_{amb} = 25°C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 2)	V_S = 5 V, E_v = 0	I_{SD}	0.8	1.1	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.4		mA
Transmission Distance	E_v = 0, test signal see fig.7, IR diode TSAL6200, I_F = 400 mA	d		35		m
Output Voltage Low (Pin 3)	I_{OSL} = 0.5 mA,E_e = 0.7 mW/m², f = f_o	V_{OSL}			250	mV
Irradiance (38 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig.7	$E_{e\ min}$		0.35	0.5	mW/m²
Irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$	$E_{e\ max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Application Circuit



*) recommended to suppress power supply disturbances
**) tolerated supply voltage range : 4.5 V<V_S<5.5 V

Suitable Data Format

The circuit of the HS0038A2. is designed in that way that unexpected output pulses due to noise or disturbance signals are avoided. A bandpassfilter, an integrator stage and an automatic gain control are used to suppress such disturbances.
The distinguishing mark between data signal and disturbance signal are carrier frequency, burst length and duty cycle.

The data signal should fullfill the following condition:

- Carrier frequency should be close to 38kHz.
- Burst length should be 10 cycles/burst or longer.
- After each burst which is between 10 cycles and 70 cycles a gap time of at least 14 cycles is neccessary.
- For each burst which is longer than 1.8ms a corresponding gap time is necessary at some time in the data stream. This gap time should be at least 4 times longer than the burst.
- Up to 800 short bursts per second can be received continuously.

Some examples for suitable data format are:
NEC Code (repetitive pulse), NEC Code (repetitive data), Toshiba Micom Format, Sharp Code

When a disturbance signal is applied to the HS0038A2. it can still receive the data signal. However the sensitivity is reduced to that level that no unexpected pulses will occure.

Some examples for such disturbance signals which are suppressed by the HS0038A2. are:

- DC light (e.g. from tungsten bulb or sunlight)
- Continuous signal at 38kHz or at any other frequency
- Signals from fluorescent lamps with electronic ballast with low modulation (see Figure A or B).

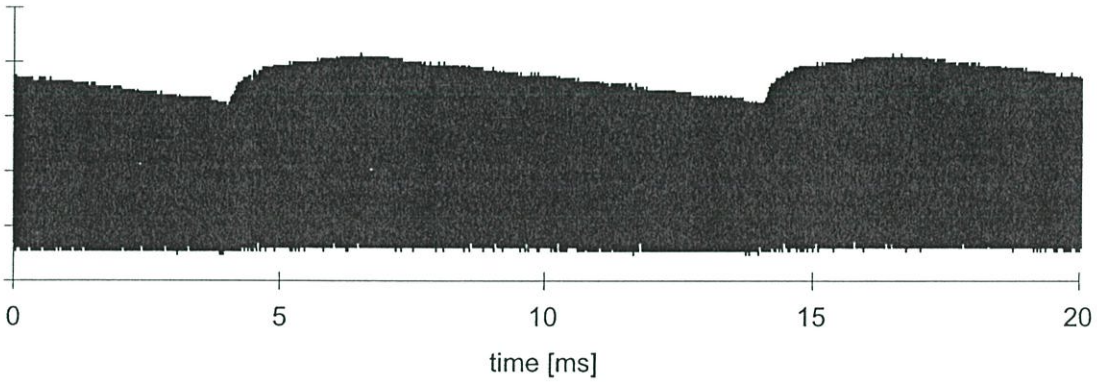


Figure A: IR Signal from Fluorescent Lamp with low Modulation

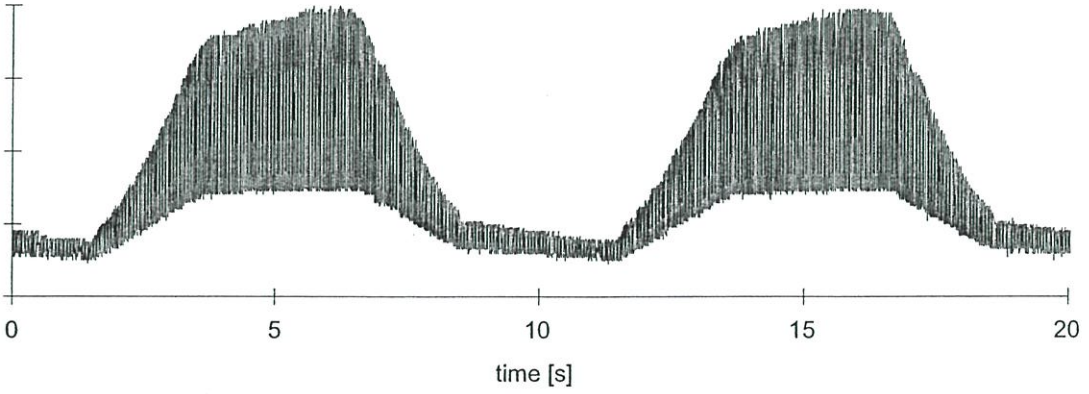


Figure B: IR Signal from Fluorescent Lamp with high Modulation

Typical Characteristics (T_{amb} = 25°C unless otherwise specified)

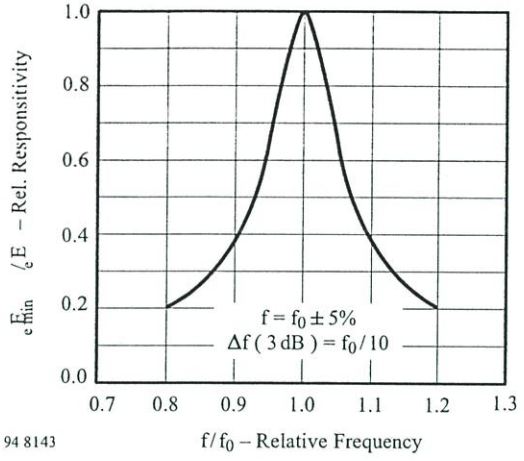


Figure 1. Frequency Dependence of Responsivity

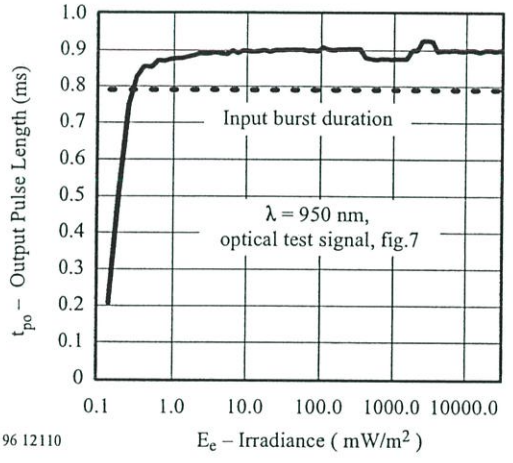


Figure 2. Sensitivity in Dark Ambient

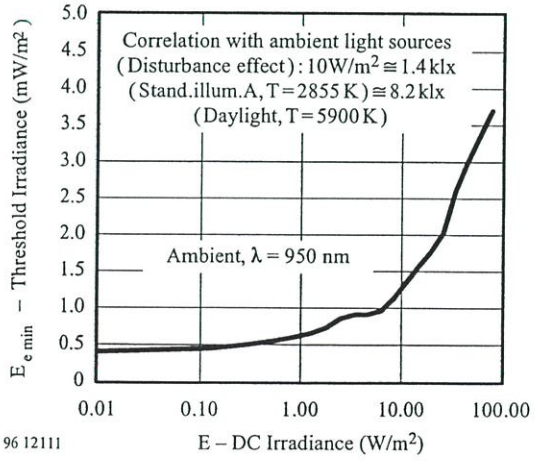


Figure 3. Sensitivity in Bright Ambient

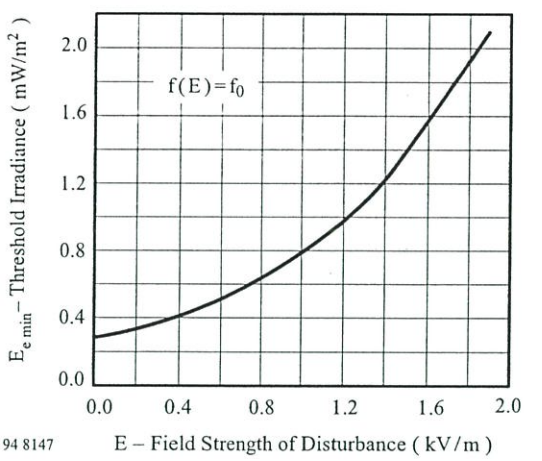


Figure 4. Sensitivity vs. Electric Field Disturbances

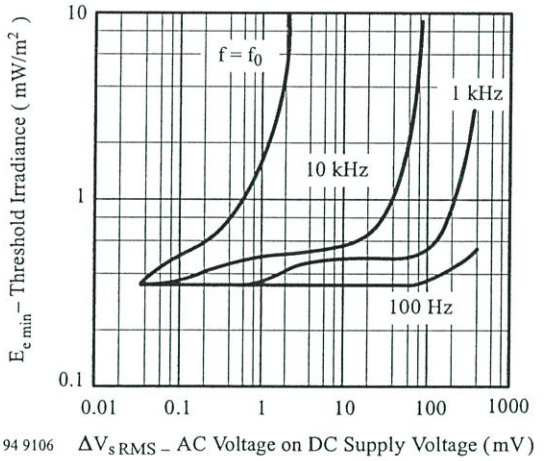


Figure 5. Sensitivity vs. Supply Voltage Disturbances

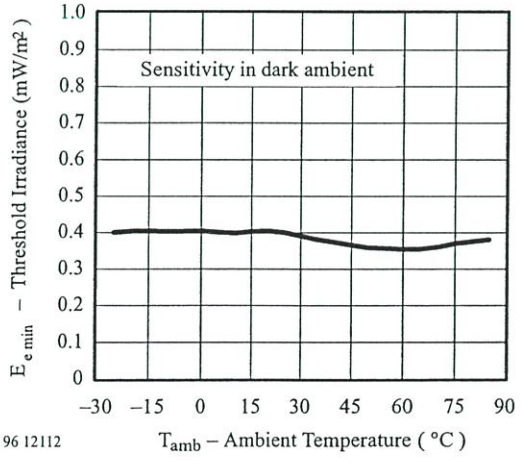
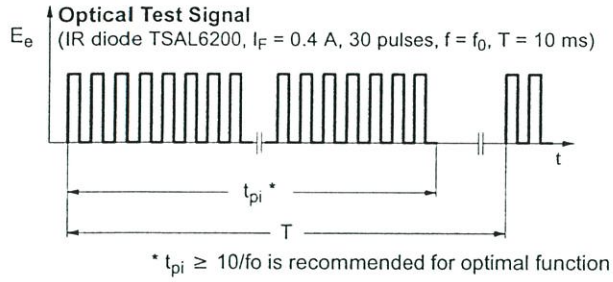


Figure 6. Sensitivity vs. Ambient Temperature



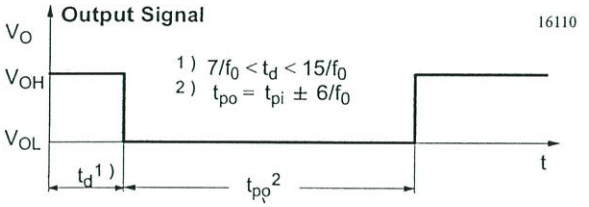


Figure 7. Output Function

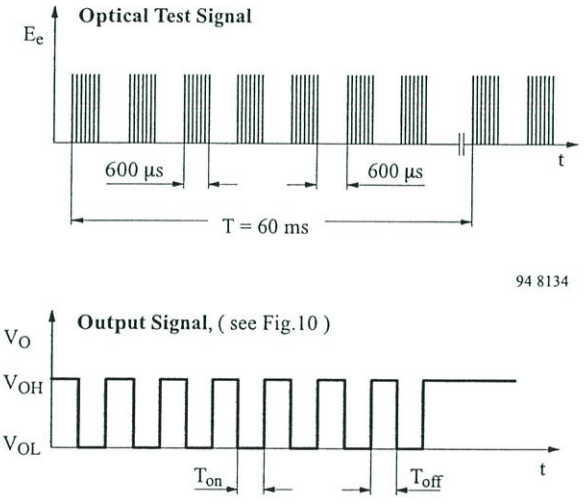


Figure 8. Output Function

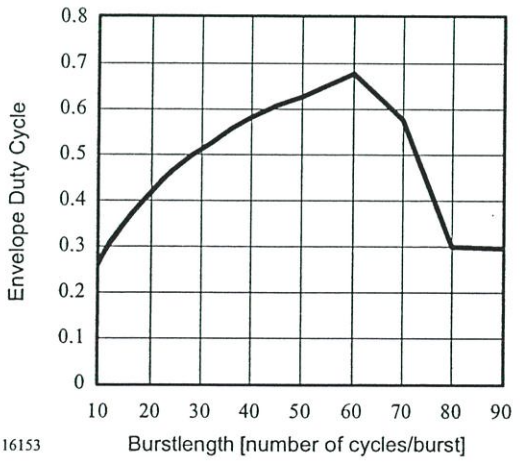


Figure 9. Max. Envelope Duty Cycle vs. Burstlength

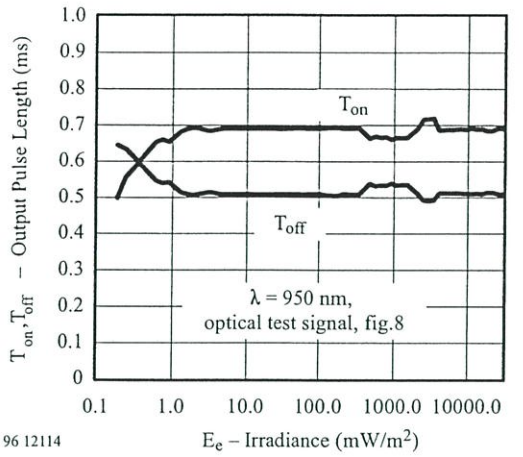


Figure 10. Output Pulse Diagram

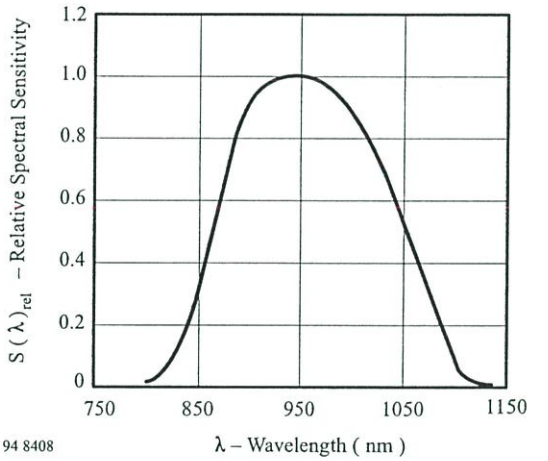


Figure 11. Relative Spectral Sensitivity vs. Wavelength

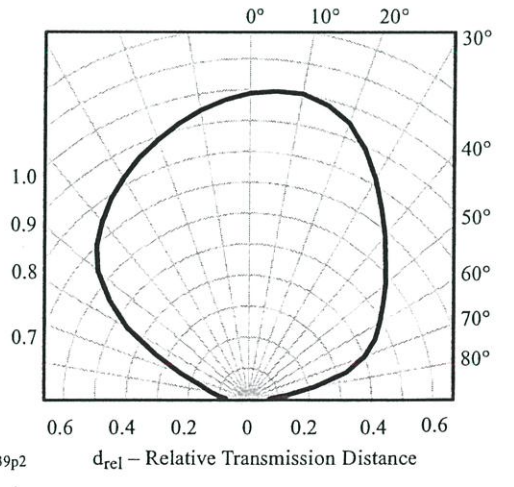


Figure 12. Vertical Directivity φ_y

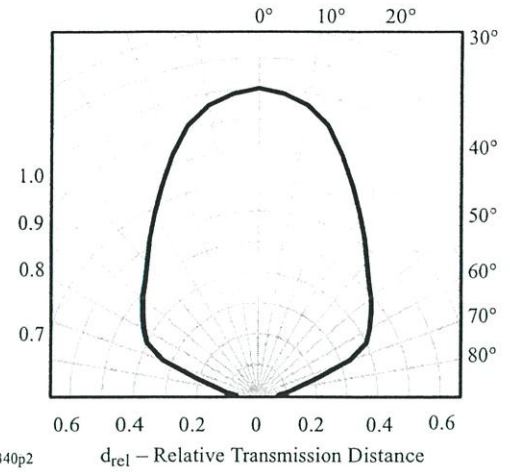
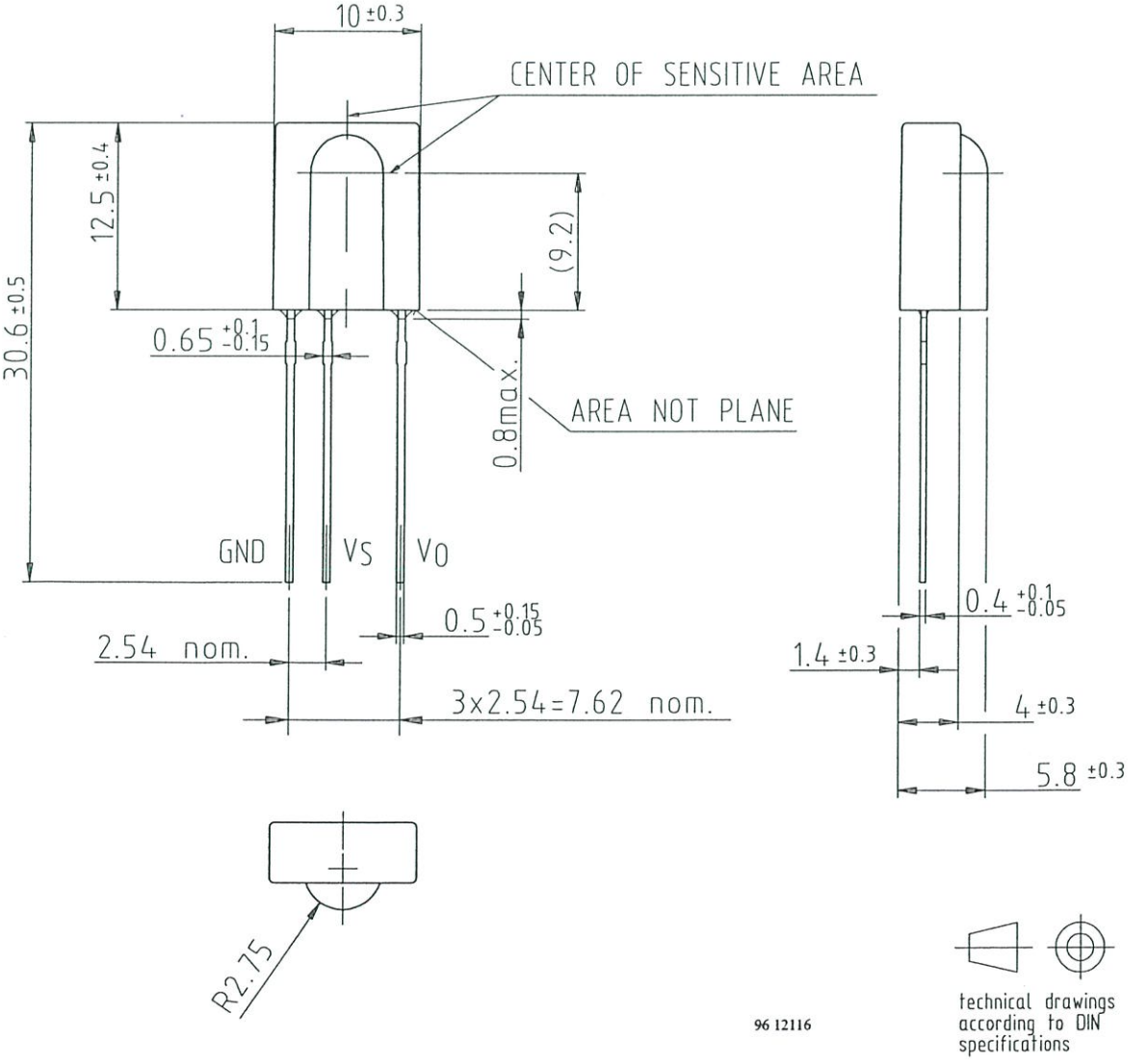


Figure 13. Horizontal Directivity φ_x

Dimensions in mm



Ozone Depleting Substances Policy Statement

It is the policy of **Vishay Semiconductor GmbH** to

1. Meet all present and future national and international statutory requirements.

2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

Vishay Semiconductor GmbH has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively

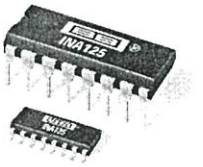
2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA

3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

Vishay Semiconductor GmbH can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

Vishay Semiconductor GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0)7131 67 2831, Fax number: 49 (0)7131 67 2423

BURR-BROWN®



INA125

INSTRUMENTATION AMPLIFIER
With Precision Voltage Reference

FEATURES

- LOW QUIESCENT CURRENT: 460μA
- PRECISION VOLTAGE REFERENCE: 1.24V, 2.5V, 5V or 10V
- SLEEP MODE
- LOW OFFSET VOLTAGE: 250μV max
- LOW OFFSET DRIFT: 2μV/°C max
- LOW INPUT BIAS CURRENT: 20nA max
- HIGH CMR: 100dB min
- LOW NOISE: 38nV/√Hz at f = 1kHz
- INPUT PROTECTION TO ±40V
- WIDE SUPPLY RANGE
  Single Supply: 2.7V to 36V
  Dual Supply: ±1.35V to ±18V
- 16-PIN DIP AND SO-16 SOIC PACKAGES

APPLICATIONS

- PRESSURE AND TEMPERATURE BRIDGE AMPLIFIERS
- INDUSTRIAL PROCESS CONTROL
- FACTORY AUTOMATION
- MULTI-CHANNEL DATA ACQUISITION
- BATTERY OPERATED SYSTEMS
- GENERAL PURPOSE INSTRUMENTATION

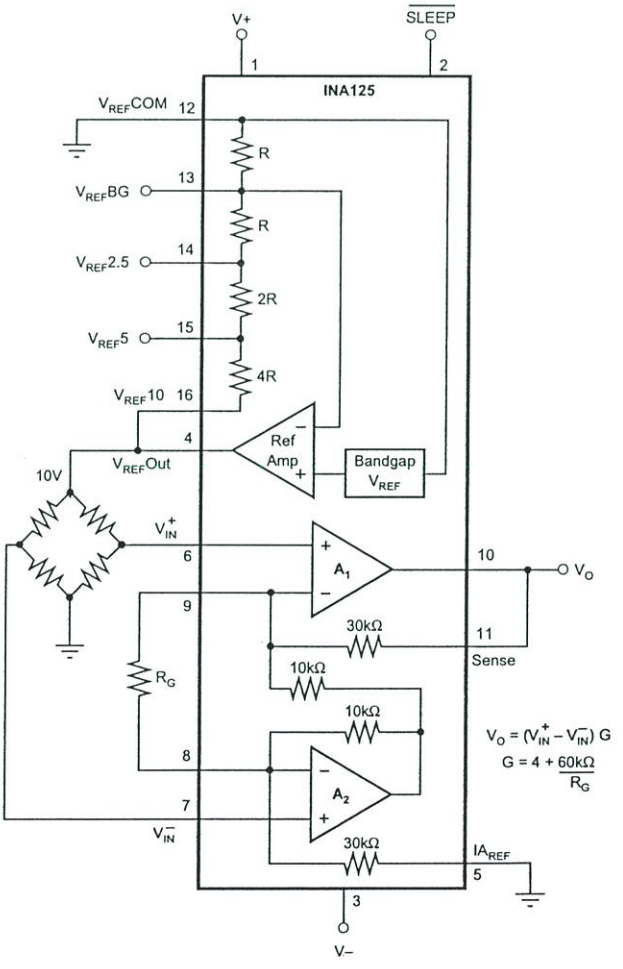
DESCRIPTION

The INA125 is a low power, high accuracy instrumentation amplifier with a precision voltage reference. It provides complete bridge excitation and precision differential-input amplification on a single integrated circuit.

A single external resistor sets any gain from 4 to 10,000. The INA125 is laser-trimmed for low offset voltage (250μV), low offset drift (2μV/°C), and high common-mode rejection (100dB at G = 100). It operates on single (+2.7V to +36V) or dual (±1.35V to ±18V) supplies.

The voltage reference is externally adjustable with pin-selectable voltages of 2.5V, 5V, or 10V, allowing use with a variety of transducers. The reference voltage is accurate to ±0.5% (max) with ±35ppm/°C drift (max). Sleep mode allows shutdown and duty cycle operation to save power.

The INA125 is available in 16-pin plastic DIP and SO-16 surface-mount packages and is specified for the –40°C to +85°C industrial temperature range.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: http://www.burr-brown.com/ • FAXLine: (800) 548-6133 (US/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1361B

Printed in U.S.A., February, 1998

SPECIFICATIONS: V_S = ±15V

At T_A = +25°C, V_S = ±15V, IA common = 0V, V_{REF} common = 0V, and R_L = 10kΩ, unless otherwise noted.

PARAMETER	CONDITIONS	INA125P, U			INA125PA, UA			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
INPUT								
Offset Voltage, RTI								
Initial			±50	±250		*	±500	µV
vs Temperature			±0.25	±2		*	±5	µV/°C
vs Power Supply	V_S = ±1.35V to ±18V, G = 4		±3	±20		*	±50	µV/V
Long-Term Stability			±0.2			*		µV/mo
Impedance, Differential			10^{11} || 2			*		Ω || pF
Common-Mode			10^{11} || 9			*		Ω || pF
Safe Input Voltage				±40			*	V
Input Voltage Range			See Text			*		
Common-Mode Rejection	V_{CM} = −10.7V to +10.2V							
	G = 4	78	84		72	*		dB
	G = 10	86	94		80	*		dB
	G = 100	100	114		90	*		dB
	G = 500	100	114		90	*		dB
BIAS CURRENT	V_{CM} = 0V		10	25		*	50	nA
vs Temperature			±60			*		pA/°C
Offset Current			±0.5	±2.5		*	±5	nA
vs Temperature			±0.5			*		pA/°C
NOISE, RTI	R_S = 0Ω							
Voltage Noise, f = 10Hz			40			*		nV/√Hz
f = 100Hz			38			*		nV/√Hz
f = 1kHz			38			*		nV/√Hz
f = 0.1Hz to 10Hz			0.8			*		µVp-p
Current Noise, f = 10Hz			170			*		fA/√Hz
f = 1kHz			56			*		fA/√Hz
f = 0.1Hz to 10Hz			5			*		pAp-p
GAIN								
Gain Equation			4 + 60kΩ/R_G			*		V/V
Range of Gain		4		10,000	*		*	V/V
Gain Error	V_O = −14V to +13.3V							
	G = 4		±0.01	±0.075		*	±0.1	%
	G = 10		±0.03	±0.3		*	±0.5	%
	G = 100		±0.05	±0.5		*	±1	%
	G = 500		±0.1			*		%
Gain vs Temperature								
	G = 4		±1	±15		*	*	ppm/°C
	G > 4[1]		±25	±100		*	*	ppm/°C
Nonlinearity	V_O = −14V to +13.3V							
	G = 4		±0.0004	±0.002		*	±0.004	% of FS
	G = 10		±0.0004	±0.002		*	±0.004	% of FS
	G = 100		±0.001	±0.01		*	*	% of FS
	G = 500		±0.002			*		% of FS
OUTPUT								
Voltage: Positive		(V+)−1.7	(V+)−0.9		*	*		V
Negative		(V−)+1	(V−)+0.4		*	*		V
Load Capacitance Stability			1000			*		pF
Short-Circuit Current			−9/+12			*		mA
VOLTAGE REFERENCE	V_{REF} = +2.5V, +5V, +10V							
Accuracy	I_L = 0		±0.15	±0.5		*	±1	%
vs Temperature	I_L = 0		±18	±35		*	±100	ppm/°C
vs Power Supply, V+	V+ = (V_{REF} + 1.25V) to +36V		±20	±50		*	±100	ppm/V
vs Load	I_L = 0 to 5mA		3	75		*	*	ppm/mA
Dropout Voltage, (V+) − V_{REF}[2]	Ref Load = 2kΩ	1.25	1		*	*		V
Bandgap Voltage Reference			1.24			*		V
Accuracy	I_L = 0		±0.5			*		%
vs Temperature	I_L = 0		±18			*		ppm/°C

19-4323; Rev 15; 1/06

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

AutoShutdown and UCSP are trademarks of Maxim Integrated Products, Inc.

Next-Generation Device Features

- **For Low-Voltage, Integrated ESD Applications MAX3222E/MAX3232E/MAX3237E/MAX3241E/ MAX3246E: +3.0V to +5.5V, Low-Power, Up to 1Mbps, True RS-232 Transceivers Using Four 0.1µF External Capacitors (MAX3246E Available in a UCSP™ Package)**
- **For Low-Cost Applications MAX221E: ±15kV ESD-Protected, +5V, 1µA, Single RS-232 Transceiver with AutoShutdown™**

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	0.047/0.33	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
V+ (Note 1) ..... (V_{CC} - 0.3V) to +14V
V- (Note 1) ..... +0.3V to +14V
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 2) ..... ±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) (Note 3) ..... +300°C

Note 1: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.
Note 2: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 3: Maximum reflow temperature for the MAX233A is +225°C.

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 Pullup/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/MAX242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/MAX242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V	V_{OUT} = ±15V		±0.01	±10	
		MAX220, V_{OUT} = ±12V			±25	
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	V_{OUT} = 0V	±7	±22		mA
		MAX220			±60	
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
		MAX220			±25	
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 4)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 4)		-0.5	-0.1	

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
RS-232 Input Hysteresis	All except MAX220/MAX243, V_{CC} = 5V, no hysteresis in SHDN		0.2	0.5	1.0	V
	MAX220			0.3		
	MAX243			1		
RS-232 Input Resistance	T_A = +25°C (MAX220)		3	5	7	KΩ
			3	5	7	
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
	I_{OUT} = 1.6mA (MAX220)				0.4	
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		
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
EN Input Threshold Low	MAX242			1.4	0.8	V
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	µA
		MAX222/MAX232A/MAX233A/ MAX242/MAX243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/MAX232A/MAX233A/ MAX242/MAX243		15		
Shutdown Supply Current	MAX222/ MAX242	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/MAX242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/MAX242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/MAX242		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	MAX222/MAX232A/MAX233/ MAX242/MAX243	6	12	30	V/µs
		MAX220	1.5	3	30.0	
Transmitter Propagation Delay TLL to RS-232 (Normal Operation), Figure 1	t_{PHLT}	MAX222/MAX232A/MAX233/ MAX242/MAX243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/MAX232A/MAX233/ MAX242/MAX243		1.5	3.5	
		MAX220		5	10	

Note 4: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

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

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

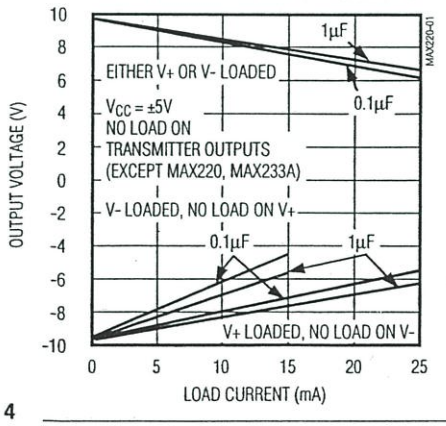
(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Receiver Propagation Delay RS-232 to TLL (Normal Operation), Figure 2	t_{PHLR}	MAX222/MAX232A/MAX233/ MAX242/MAX243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/MAX232A/MAX233/ MAX242/MAX243		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_{PHLS}	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/MAX242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ Goes Low), Figure 4	t_{DT}	MAX222/MAX242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (Normal Operation)	t_{PHLT} - t_{PLHT}	MAX222/MAX232A/MAX233/ MAX242/MAX243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (Normal Operation)	t_{PHLR} - t_{PLHR}	MAX222/MAX232A/MAX233/ MAX242/MAX243		100		ns
		MAX220		225		

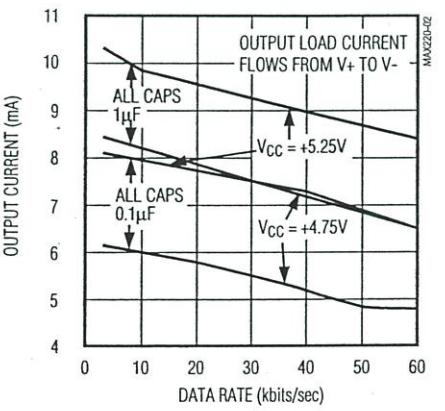
Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243

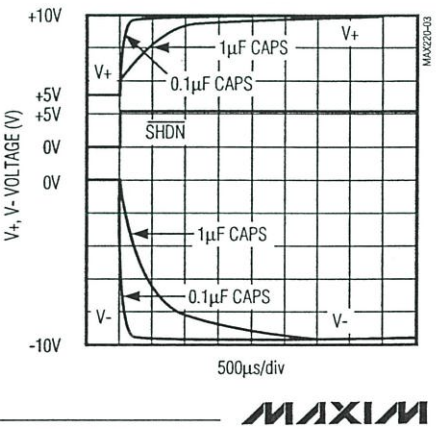
OUTPUT VOLTAGE vs. LOAD CURRENT

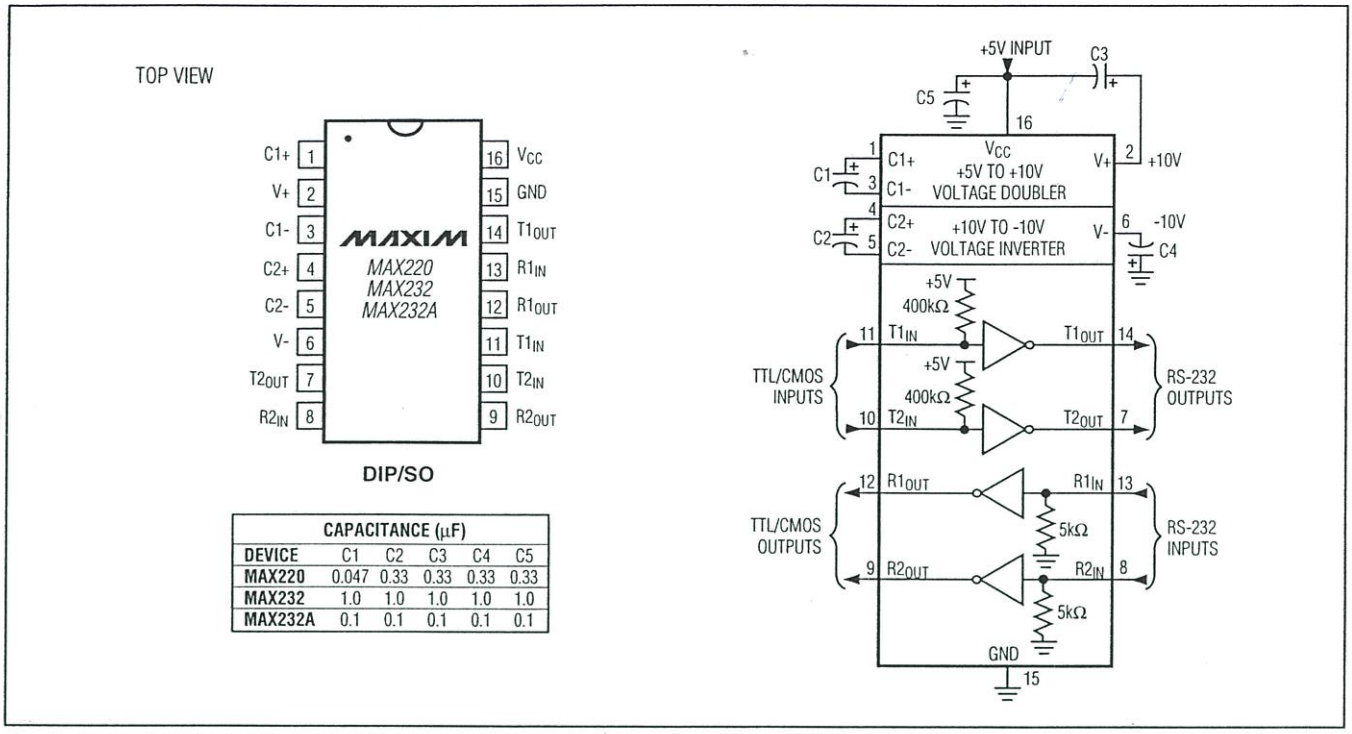


AVAILABLE OUTPUT CURRENT vs. DATA RATE



MAX222/MAX242 ON-TIME EXITING SHUTDOWN





CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
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

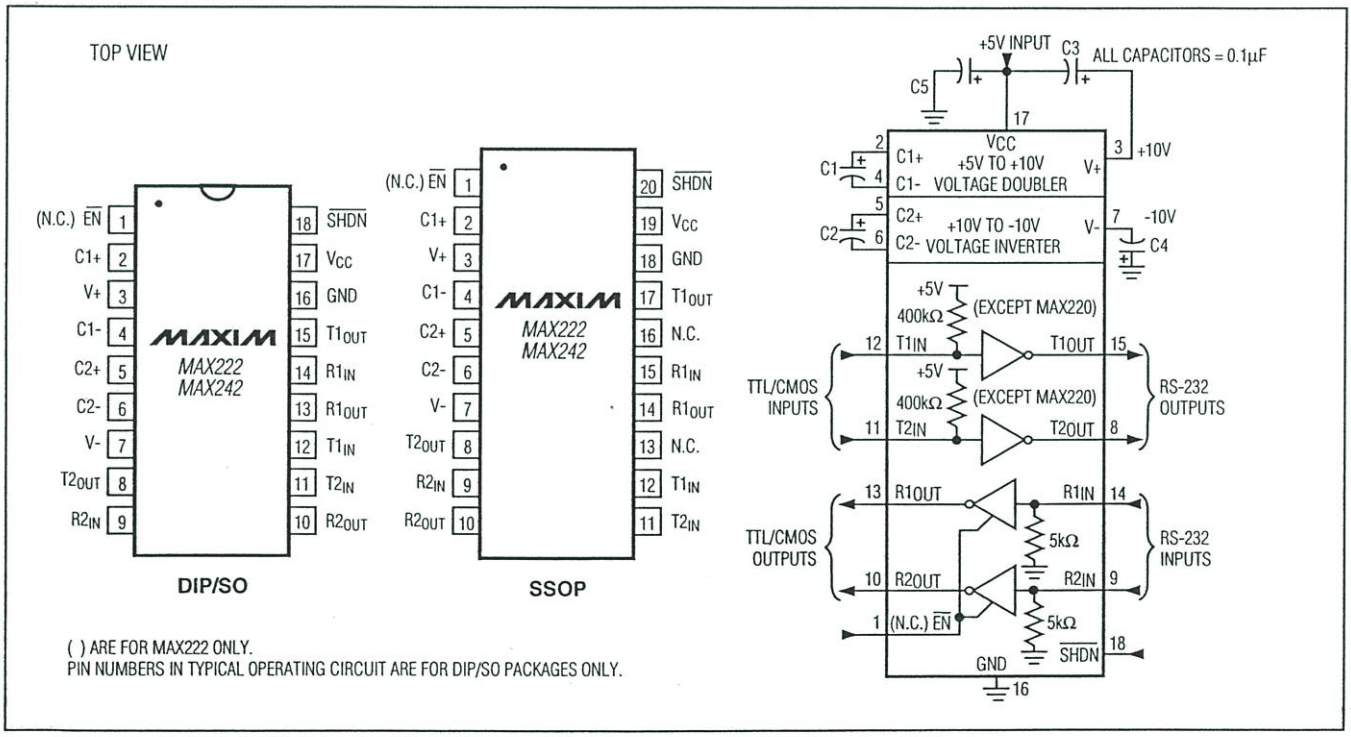


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit