ระบบควบคุมอุปกรณ์ไฟฟ้าและเตือนภัยผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ

WARNING AND ELECTRIC DEVICE CONTROL SYSTEMS VIA INTERNET AND MOBILE PHONE

โดย

นายอภิรักษ์ ทัดสอน
นายอรรถพล สุดสาคร
นางสาวอัญชลี วุฒิฐิโก
นายอานุภาพ สงวนสุข

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

ระบบควบคุมอุปกรณ์ไฟฟ้าและเตือนภัยผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ

WARNING AND ELECTRIC DEVICE CONTROL SYSTEMS VIA INTERNET AND MOBILE PHONE

โดย

นายอภิรักษ์ ทัดสอน 48011069

นายอรรถพล สุดสาคร 48011086

นางสาวอัญชลี วุฒิฐิโก 48011105

นายอานุภาพ สงวนสุข 48011117

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

ดร.สิรภพ ตู้ประกาย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมอุปกรณ์ไฟฟ้าและเตือนภัยผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ

WARNING AND ELECTRIC DEVICE CONTROL SYSTEMS

VIA INTERNET AND MOBILE PHONE

ผู้จัดทำ

1. นายอภิรักษ์ ทัดสอน 48011069
2. นายอรรถพล สุดสาคร 48011086
3. นางสาวอัญชลี วุฒิฐิโก 48011105
4. นายอานุภาพ สงวนสุข 48011117

_____ อาจารย์ที่ปรึกษา

(รศ.ดร.กอบชัย เดชหาญ)

_____ อาจารย์ที่ปรึกษา

(ดร.สิรภพ ตู้ประกาย)

ระบบควบคุมอุปกรณ์ไฟฟ้าและเตือนภัยผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ

WARNING AND ELECTRIC DEVICE CONTROL SYSTEMS VIA INTERNET AND MOBILE PHONE

โดย นายอภิรักษ์ ทัดสอน 48011069

นายอรรถพล สุดสาคร 48011086

นางสาวอัญชลี วุฒิฐิโก 48011105

นายอานุภาพ สงวนสุข 48011117

อาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ

ดร.สิรภพ ตู้ประกาย

บทคัดย่อ

โครงงานนี้เป็นการสร้างระบบควบคุมอุปกรณ์ไฟฟ้าและระบบเตือนภัยเมื่อเกิดเหตุการณ์ไม่ปกติ โดยการประยุกต์ใช้งานไมโครโปรเซสเซอร์ ที่มีชื่อว่า RABBIT ซึ่งเป็นไมโครโปรเซสเซอร์ที่สามารถเป็นไมโครคอนโทรลเลอร์และเป็นเซิร์ฟเวอร์ได้ด้วยตัวเอง โดยจะทราบสถานะการทำงานผ่านทางหน้าเว็บบราวเซอร์และจากโทรศัพท์มือถือ นอกจากนี้ผู้ใช้ยังจะสามารถควบคุมอุปกรณ์ต่างๆผ่านทางหน้าเว็บบราวเซอร์หรือโทรศัพท์มือถือได้อีกด้วย

ABSTRACT

This project presents the electric device control system and warning system when happening wrong event. By used microprocessor is RABBIT, which it is microprocessor and server. The status shows via the browser and mobile phone. Besides users can be control equipment via browser or mobile phone.

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันระบบเครือข่ายอินเตอร์เน็ตได้เข้ามามีบทบาทในการดำเนินชีวิตประจำวันของเรามากขึ้น ไม่ว่าจะเป็นการสืบค้นข้อมูล รับส่งอีเมล์ หรือแม้แต่การติดต่อกับหน่วยงานราชการ และบริษัทต่างๆ อินเตอร์เน็ตช่วยให้เราประหยัดเวลาและค่าใช้จ่ายได้เป็นอย่างดี รวมถึงโทรศัพท์มือถือซึ่งมีใช้กันแทบจะทุกคน ถ้าหากเครื่องใช้ไฟฟ้าในบ้านของเราสามารถควบคุมการเปิด-ปิดได้ โดยผ่านเครือข่ายอินเตอร์เน็ตหรือทางโทรศัพท์มือถือ ย่อมเพิ่มความสะดวกสบาย รวมไปถึงประหยัดค่าใช้จ่ายจากการเปิดเครื่องใช้ไฟฟ้าทิ้งไว้โดยไม่จำเป็น แม้กระทั่งเพิ่มความปลอดภัย หากเครื่องใช้ไฟฟ้านั้นลืมเปิดทิ้งไว้อาจก่อให้เกิดอัคคีภัยได้

ดังนั้นโครงงานนี้จึงได้ทำการศึกษา อุปกรณ์ที่ใช้ควบคุมอุปกรณ์อื่นๆ และสามารถเชื่อมต่อระบบเครือข่ายอินเตอร์เน็ตได้ โดยการควบคุมจะเป็นการควบคุมผ่านระบบเครือข่ายอินเตอร์เน็ตหรือโทรศัพท์มือถือได้

1.2 วัตถุประสงค์

1.เพื่อศึกษาและสร้างระบบควบคุมผ่านทางเครือข่ายอินเตอร์เน็ต

2.เพื่อศึกษาการทำงานของ Rabbit Core Module RCM3200

3.เพื่อศึกษาและออกแบบโปรแกรมสำหรับควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า

4.เพื่อศึกษาและออกแบบโปรแกรมสำหรับการเตือนภัยการถูกเปิด-ปิดประตู หน้าต่าง

5.เพื่อศึกษาและสร้างระบบควบคุมผ่านทางโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

ระบบสำหรับควบคุมอุปกรณ์ไฟฟ้าผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ โดยใช้ RCM3200 เป็นตัวเชื่อมต่อและควบคุมการทำงานของระบบผ่านวงจรควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า โดยจะสามารถเปิด-ปิดอุปกรณ์ไฟฟ้า ผ่านทางเว็บบราวเซอร์และโทรศัพท์มือถือ และยังสามารถเตือนภัยสำหรับการถูกเปิด-ปิดประตู หน้าต่างอีกด้วย รวมถึงการตั้งเวลาไว้ล่วงหน้าเพื่อเปิด-ปิดอุปกรณ์ต่างๆได้ด้วย

บทที่ 2
ทฤษฎีและหลักการ

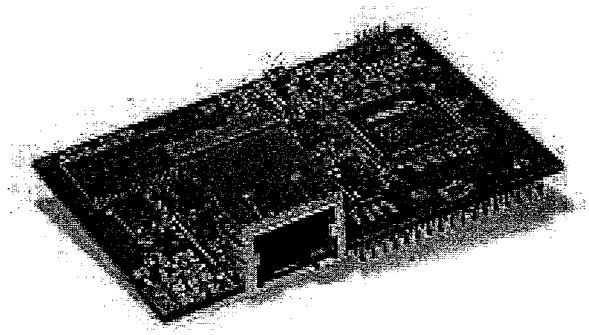
2.1 พื้นฐาน RCM3200 RabbitCore

ไมโครคอนโทรลเลอร์ RCM3200 RabbitCore ถูกออกแบบมาเพื่อเป็นหัวใจสำคัญของระบบควบคุมฝังตัว (embedded control system) ซึ่งมีคุณสมบัติพิเศษสามารถเชื่อมต่อระบบ LAN ระบบอินเตอร์เน็ต เพื่อนำไปสู่การสร้างระบบควบคุมการติดต่อสื่อสารผ่านเว็บบราวเซอร์ได้โดยง่าย โดยมีพอร์ตวงจรรวม Ethernet 10Base-T หัว RJ-45 ติดตั้งบนไมโครคอนโทรลเลอร์ RCM3200 RabbitCore เพื่อทำหน้าที่เชื่อมต่อกับการ์ด LAN ของ PC

ไมโครคอนโทรลเลอร์ RCM3200 RabbitCore มีไมโครโปรเซสเซอร์ Rabbit3000 ขนาด 8 บิต เป็นตัวขับเคลื่อน ดำเนินการที่สัญญาณนาฬิกา 44.2 MHz หน่วยความจำพื้นฐานเป็นแบบ SRAM หน่วยความจำโปรแกรมเป็นแบบ FLASH สัญญาณนาฬิกา 2 แหล่ง (ออสซิลเลเตอร์หลัก และที่ได้จากการบันทึกเวลา) มีวงจรสำหรับการรีเซต และแบตเตอรี่สำรองสำหรับการจัดการ Real-Time Clock (RTC) และหน่วยความจำ SRAM ภายในไมโครโปรเซสเซอร์ Rabbit3000

ไมโครคอนโทรลเลอร์ RCM3200 RabbitCore มีขาทำหน้าที่เป็นขาไฟเลี้ยง ขาอินพุต/เอาต์พุต ขาชี้ตำแหน่งแอดเดรส ขานำข้อมูล พอร์ตขนาน พอร์ตอนุกรม รวมทั้งหมด 68 ขาโดยแบ่งเป็นหัวคอนเน็คเตอร์ J1 และ J2 จำนวนหัวละ 34 ขา

ไมโครคอนโทรลเลอร์ RCM3200 RabbitCore ได้รับไฟเลี้ยง +3.3 โวลต์ จากการติดตั้งโมดูลบนบอร์ดทดลองของผู้ใช้และยังเชื่อมต่อกับอุปกรณ์จำพวกดิจิตอลผ่านทางบอร์ดทดลองของผู้ใช้เช่นกัน

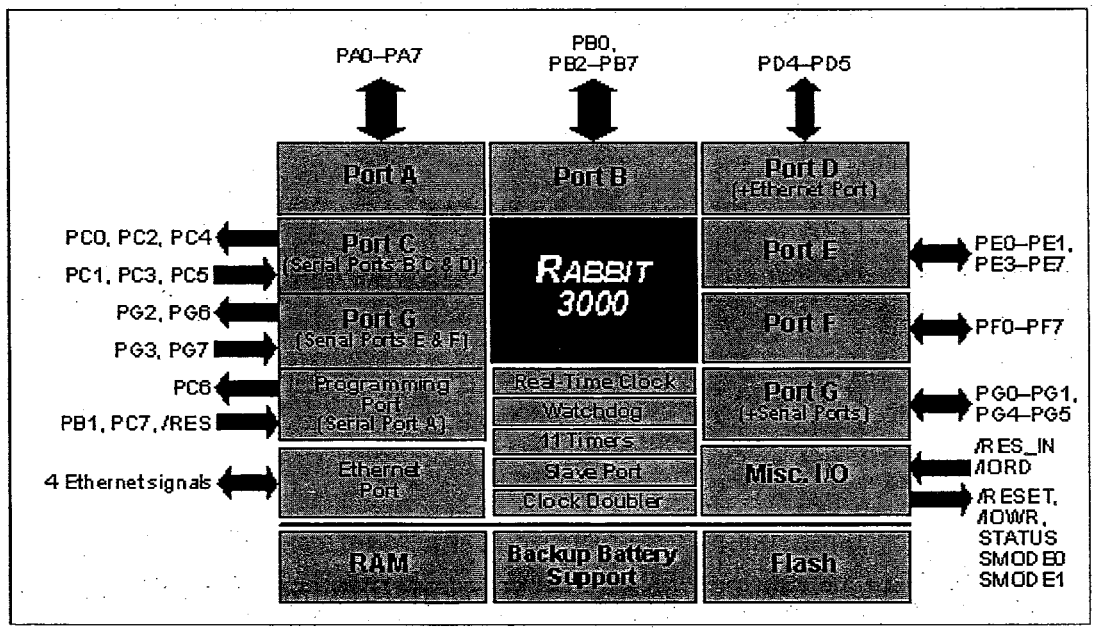


รูปที่ 2.1 บอร์ดวงจร RCM3200

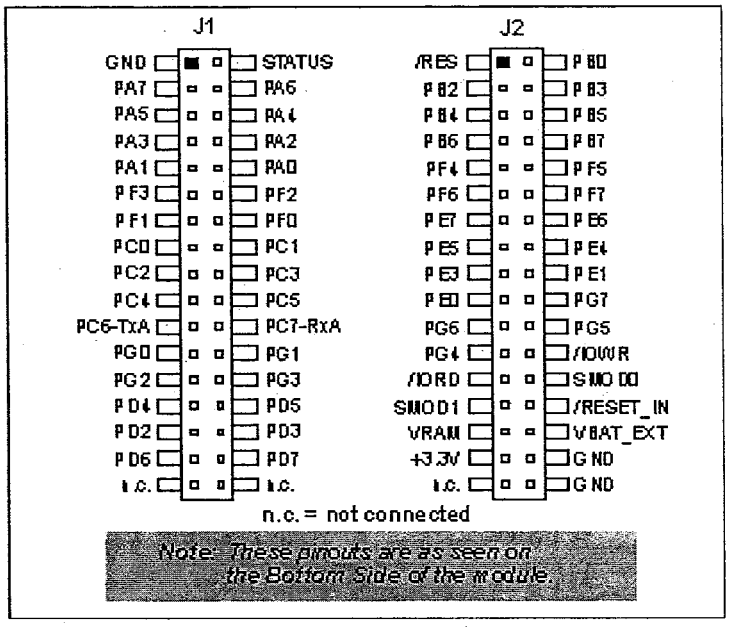
2.2 คุณสมบัติไมโครคอนโทรลเลอร์ RCM3200 RabbitCore

- ขนาดโมดูล 1.85 x 2.65 x 0.86 นิ้ว
- ไมโครโปรเซสเซอร์ : RABBIT3000 ที่สัญญาณนาฬิกา 44.2 MHz
- อินพุต/เอาต์พุต แบบขนาน 52 ขาโดยแบ่ง 44 ขาทำหน้าที่เป็นอินพุต/เอาต์พุต 4 ขาทำหน้าที่เป็นอินพุต และ 4 ขาทำหน้าที่เป็นเอาต์พุต

- เพิ่มเติม 2 ดิจิตอลอินพุต และ 2 ดิจิตอลเอาต์พุต
- สามารถใช้ขาอินพุต/เอาต์พุต ทำหน้าที่เสริมคือ ทำหน้าที่เป็นขานำข้อมูล 8 ขา และชี้แอดเดรส 6 ขา (ใช้งานร่วมกับพอร์ตขนาน อินพุต/เอาต์พุต) และ อินพุต/เอาต์พุต อ่าน/เขียน
- อินพุต รีเซตจากภายนอก
- ไทเมอร์ 10 ตัว มีขนาด 8 บิต และ 1 ตัวมีขนาด 10 บิต ด้วยการรวม 2 รีจีสเตอร์
- หน่วยความจำโปรแกรม execution ขนาด 512K แบบ SRAM และหน่วยความจำข้อมูล ขนาด 256K แบบ SRAM
- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 512K สามารถอ่าน/เขียนได้เป็น 100,000 ครั้ง
- วงจร Real-Time Clock (RTC)
- วงจร Watchdog Supervisor
- พอร์ต Ethernet 10Base-T หัว RJ-45
- 2 ช่องสัญญาณ Input Capture สามารถใช้เป็นสัญญาณอินพุตคาบเวลาจากขาพอร์ต
- 2 ช่องสัญญาณ Quadrature Decoder ตอบสนองอินพุตจากโมดูลเข้ารหัส (encoder) ภายนอก
- พอร์ตอนุกรม 6 พอร์ต โดยมีอัตราการส่งข้อมูลแบบ อะซิงโครนัสสูงถึง 5.5Mbps ซึ่ง 4 พอร์ตกำหนดให้เป็นพอร์ตอนุกรมสัญญาณนาฬิกา (SPI) และ 2 พอร์ตกำหนดเป็นพอร์ตอนุกรม SDLC/HDLC
- รองรับการส่ง IrDA อัตราการส่ง 1.15Mbps



รูปที่ 2.2 พอร์ตของโมดูล RCM3200



รูปที่ 2.3 ขาของโมดูล RCM3200

2.3 การจัดการขาของไมโครคอนโทรลเลอร์ RCM3200

ไมโครโปรเซสเซอร์ตระกูล RABBIT3000 รุ่น RCM3200 มีสถาปัตยกรรมพื้นฐาน ดังแสดงในรูปที่ 2.3

- ขา +3.3V ใช้สำหรับต่อไฟเลี้ยง +3.3V

- ขา GND เป็นกราวด์ สำหรับต่อกับกราวด์ของระบบ

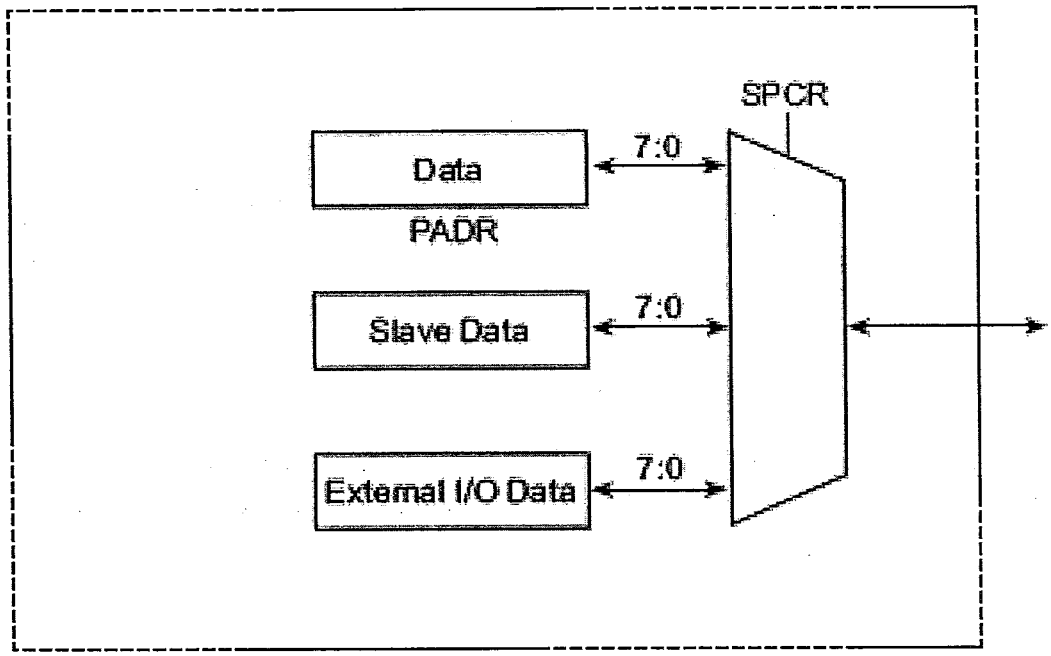
- ขา /RESET_IN ใช้ในการรีเซตการทำงานของไมโครโปรเซสเซอร์เพื่อให้เข้าสู่สถานะ การทำงานเริ่มต้น

- ขา VBAT_EXT ทำหน้าที่เป็นขาเชื่อมต่อแบตเตอรี่สำรองสำหรับจัดการ วงจร Real-Time Clock (RTC)

- ขาพอร์ต A (PA0-PA7) พอร์ต A มีขนาด 8 บิต แบบ 2 ทิศทาง สามารถกำหนดให้เป็นได้ทั้งอินพุตหรือเอาต์พุตสำหรับใช้งานทั่วไป นอกจากนี้พอร์ต A ยอมให้ใช้เป็นบัสข้อมูลสำหรับพอร์ต Slave (Slave Port Control Register (SPCR)) จะใช้ในการควบคุมพอร์ตขนาน A ซึ่งจะกำหนดเป็นเส้นทางข้อมูลแบบ Slave และ เส้นทาง (BUS) ข้อมูล สนับสนุนอินพุต/เอาต์พุต อินพุตแบบขนานหรือเอาต์พุตแบบขนาน ในการทำพอร์ตให้เป็นอินพุต ทำได้โดยการเก็บค่า 0x80 ไว้ในรีจีสเตอร์ SPCR และการทำพอร์ตให้เป็นเอาต์พุตโดยการเก็บค่า 0x84 ไว้ในรีจีสเตอร์ SPCR สุดท้ายคือพอร์ตขนาน A จะเซตเป็นพอร์ตอินพุตเมื่อถูก "รีเซต"

เมื่อพอร์ตทำหน้าที่อ่านข้อมูล ค่าที่อ่านได้จะตอบสนองกันกับแรงดันที่ขาโดยที่ "1" แทน High และ "0" แทน Low เช่นนี้เป็นการอ้างอิงการเก็บค่าในรีจีสเตอร์เอาต์พุตถ้าขาได้รับแรงดัน

พอร์ต A มีรีจีสเตอร์ที่ใช้ในการควบคุมการทำงานของพอร์ตอยู่ 2 ตัวด้วยกันและบล็อกโครงสร้างการทำงานของพอร์ต A แสดงดังรูปที่ 2.4



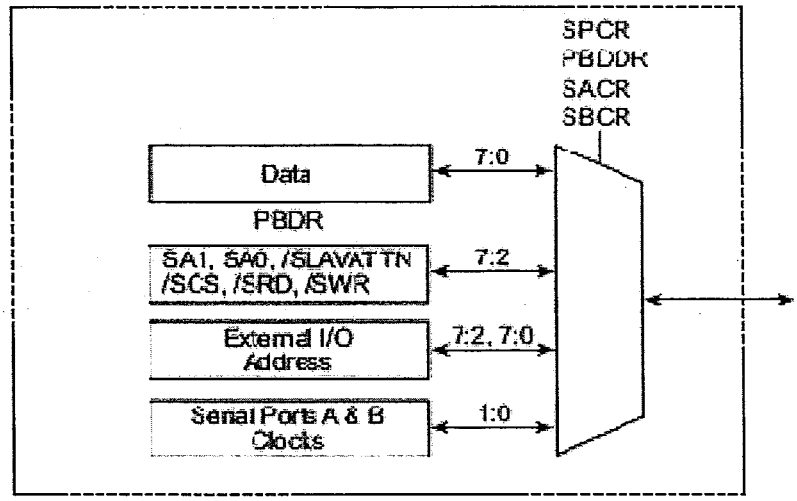
รูปที่ 2.4 บล็อกโครงสร้างการทำงานของ พอร์ต A

Port A Data Register (PADR) เป็นรีจีสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะของการทำงานเริ่มต้นของพอร์ตให้เป็น “0” หรือ “1”

Slave Port Control Register (SPCR) เป็นรีจีสเตอร์ที่ใช้ในการกำหนดการทำงานของพอร์ตว่าจะให้เป็นอินพุตหรือเอาต์พุต ในกรณีที่ต้องการให้พอร์ตมีการทำงานเป็นอินพุตจะต้องทำการเก็บค่า 0x80 ไว้ในรีจีสเตอร์ และถ้าต้องการให้มีการทำงานเป็นแบบเอาต์พุตจะต้องทำการเก็บค่า 0x84 ไว้ในรีจีสเตอร์

- ขาพอร์ต B (PB0-PB7) พอร์ตขนาน B มีทั้งหมด 8 ขา (บิต) ซึ่งสามารถแยกโปรแกรมให้ขาอินพุต/เอาต์พุต ภายหลังจาก “รีเซต” พอร์ต B จะกลับมาเป็น 6 อินพุต (PB0-PB5) และ 2 เอาต์พุต (PB6 และ PB7) ค่าเอาต์พุตที่ขา PB7 และ PB6 จะเป็น “0” เมื่อเส้นทาง (Bus) สนับสนุนอินพุต/เอาต์พุต (I/O) ถูกใช้งาน พอร์ตขนาน B บิตที่ 2-7 จัดเป็น 6 เส้นทาง แอดเดรสจาก 16 เส้นทางแอดเดรส เมื่อพอร์ต Slave ถูกใช้งานพอร์ตขนานขาที่ PB2-PB7 จะถูกกำหนดให้เป็นฟังก์ชันพอร์ต Slave อย่างไรก็ตามพอร์ตขนานขาที่ PB2-PB7 ยังใช้ในการอ่าน PB0-PB5 โดยใช้รีจิสเตอร์ข้อมูลพอร์ต B ได้ แม้ว่า PB2-PB7 จะถูกใช้เป็นพอร์ต Slave ก็ตาม นอกจากนี้ยังสามารถใช้เพื่ออ่านการขับสัญญาณ PB6 และ PB7 (สัญญาณนี้อยู่ในสายสัญญาณจากลอจิกพอร์ต Slave) ไม่ว่าพอร์ต Slave จะมีการใช้งานหรือไม่ PB0 จะทำหน้าที่สะท้อนอินพุตของขาถ้าพอร์ตอนุกรม B ไม่มีการใช้งานสัญญาณนาฬิกาภายในซึ่งเป็นสาเหตุให้ขานี้ถูกขับโดยสัญญาณนาฬิกาพอร์ตอนุกรม ถ้าพอร์ตอนุกรม A ไม่มีการใช้งานสัญญาณนาฬิกาภายใน PB1 จะทำหน้าที่สะท้อนอินพุตของขา

พอร์ต B มีรีจิสเตอร์ที่ใช้ในการควบคุมกรทำงานของพอร์ตอยู่ 2 ตัวด้วยกันและบล็อกโครงสร้างการทำงานของพอร์ต B แสดงรูปที่ 2.5



รูปที่ 2.5 บล็อกโครงสร้างการทำงานของ พอร์ต B

Port B Data Register (PBDR) เป็นรีจิสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะเริ่มต้นของพอร์ตให้เป็น “0” หรือ “1”

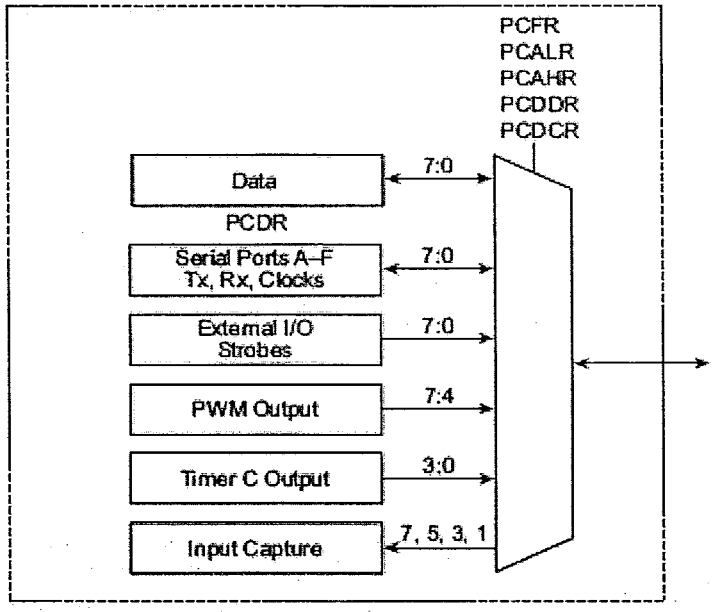
Port B Data Direction Register (PBDDR) เป็นรีจิสเตอร์ที่ใช้ในการกำหนดการทำงานของพอร์ตว่าจะให้เป็นอินพุตหรือเอาต์พุต ถ้าต้องการให้พอร์ตมีการทำงานเป็นอินพุตจะต้องเซตค่าให้มีค่าเท่ากับ “0” ถ้าต้องการให้พอร์ตมีการทำงานเป็นเอาต์พุตจะต้องเซตค่าให้มีค่าเท่ากับ “1”

- ขาพอร์ต C (PC0-PC7) พอร์ตขนาน C มี 4 ขาทำหน้าที่เป็นอินพุตและ 4 ขาทำหน้าที่เป็นเอาต์พุต พอร์ตเลขคู่คือ PC0, PC2, PC4 และ PC6 ทำหน้าที่เป็นเอาต์พุตและพอร์ตเลขคี่คือ PC1, PC3, PC5 และ PC7 ทำหน้าที่เป็นอินพุตเมื่อรีจิสเตอร์ข้อมูลถูกอ่านบิต 1, 3, 5 และ 7 จะคืนค่าของแรงดันที่ขา บิต 0, 2, 4 และ 6 จะคืนค่าสัญญาณที่กำลังขับบัฟเฟอร์เอาต์พุต ซึ่งสัญญาณที่กำลังขับบัฟเฟอร์เอาต์พุตและค่าของขาเอาต์พุตโดยปกติแล้วจะมีค่าเท่ากัน ถ้าไม่ใช้รีจิสเตอร์ข้อมูลพอร์ต C ที่ทำหน้าที่ในการขับขาเหล่านี้ ก็จะเป็นหนึ่งในสายสัญญาณถ่ายโอนพอร์ตอนุกรมตัวใดตัวหนึ่งที่ทำหน้าที่ในการขับขาบิตที่ถูกเซตในรีจิสเตอร์ PCFR จะเป็นตัวระบุว่ารีจิสเตอร์ข้อมูลหรือสายสัญญาณถ่ายโอนพอร์ตอนุกรมที่กำลังขับขา

พอร์ตขนาน C จะใช้ขาร่วมกับพอร์ตอนุกรม A-D อินพุตพอร์ตขนานสามารถกำหนดให้เป็นอินพุตพอร์ตอนุกรมขณะที่เอาต์พุตเป็นเอาต์พุตพอร์ตอนุกรม เมื่อถูกใช้ให้เป็นอินพุตอนุกรมสายสัญญาณข้อมูลสามารถถูกอ่านจากรีจิสเตอร์ข้อมูลพอร์ตขนาน C เอาต์พุตพอร์ตแบบขนานสามารถถูกเลือกให้เป็นเอาต์พุตพอร์ตอนุกรมโดยการเซตตำแหน่งของบิตในรีจิสเตอร์ PCFR เมื่อขาเอาต์พุตพอร์ตขนาน C ถูกเลือกให้เป็นเอาต์พุตพอร์ตอนุกรม ดังนั้นจึงไม่ต้องสนใจค่าที่ถูกเก็บในรีจิสเตอร์ข้อมูล

เมื่อทำการรีเซตบิตของรีจิสเตอร์ฟังก์ชันที่เป็นเลขคู่จะถูกกำหนดเป็น “0” ซึ่งส่งผลให้พอร์ต C ทำหน้าที่เป็นพอร์ตอินพุต/เอาต์พุต บิตที่ 6 ของรีจิสเตอร์ข้อมูลพอร์ต C จะถูกกำหนดเป็น “0” ขณะที่บิตคู่ขาอื่นถูกเซตให้เป็น “1”

พอร์ต C มีรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของพอร์ตอยู่ 2 ตัวด้วยกันและบล็อกโครงสร้างการทำงานของพอร์ต C แสดงรูปที่ 2.6



รูปที่ 2.6 บล็อกโครงสร้างการทำงานของ พอร์ต C

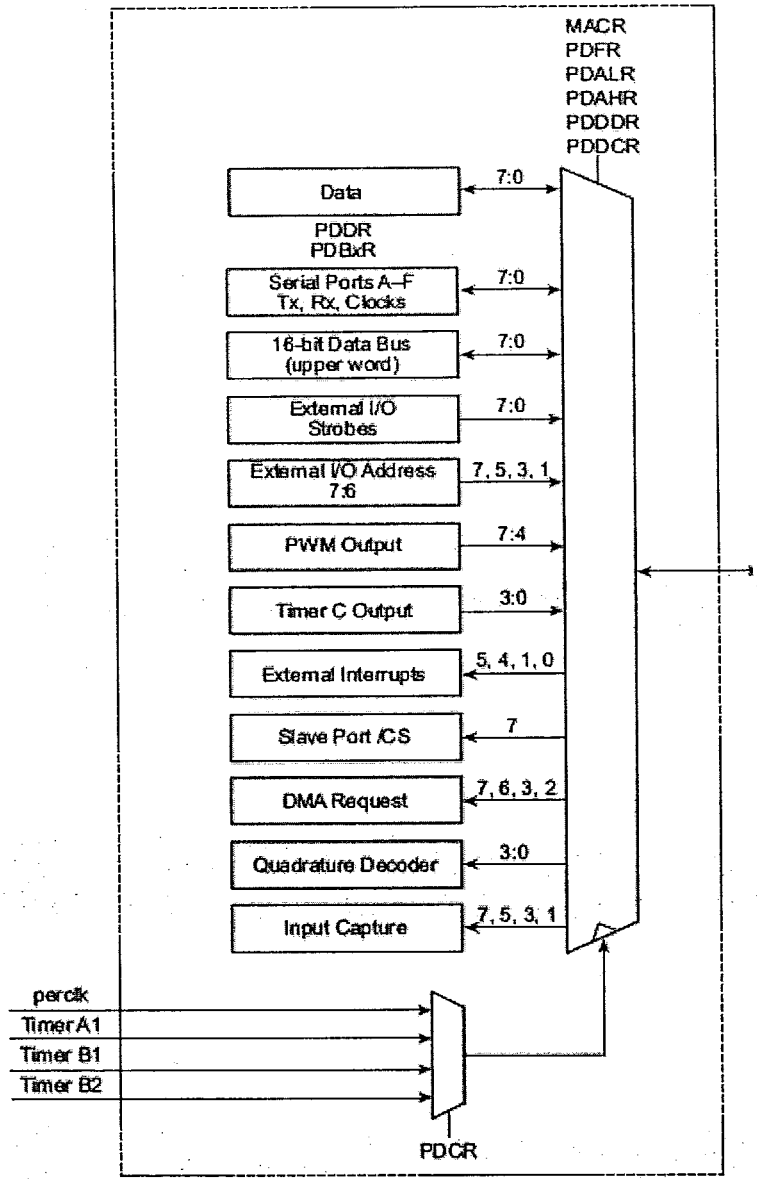
Port C Data Register (PCDR) เป็นรีจิสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะ การทำงานเริ่มต้นของพอร์ตให้เป็น “0” หรือ “1”

Port C Function Register (PCFR) เป็นรีจิสเตอร์ที่ใช้ในการควบคุมสถานะ การทำงานของพอร์ต

- ขาพอร์ต D (PD0-PD7) พอร์ตขนาน D ดังแสดงในรูปที่ 2.7 มีทั้งหมด 8 ขา (บิต) ซึ่งสามารถโปรแกรมเป็นอินพุตหรือเอาต์พุต เมื่อถูกโปรแกรมให้เป็นเอาต์พุต ขาสามารถถูกเลือกทีละขาให้เป็นเอาต์พุตแบบขาเปิด (Open drain) หรือเอาต์พุตแบบมาตรฐาน ขาของพอร์ต D สามารถกำหนดตำแหน่งโดยบิตได้ถ้าต้องการ รีจิสเตอร์เอาต์พุตถูกต่อแบบเลือกการทำงานและควบคุมด้วย Timer ซึ่งทำให้มันสามารถที่จะสร้างสัญญาณพัลส์ให้เกิดคาบเวลาที่แน่นอน พอร์ต D บิต 4 และ 5 สามารถใช้เป็นบิตแบบสลับ (Alternate) สำหรับพอร์ตอนุกรม B และพอร์ต D บิต 6 และ 7 สามารถใช้เป็นบิตแบบสลับ (Alternate) สำหรับพอร์ตอนุกรม A การกำหนดบิตของพอร์ตอนุกรมแบบสลับ ทำให้พอร์ตอนุกรมที่เหมือนกันเชื่อมต่อกับสายสัญญาณการติดต่อสื่อสารที่ต่างชนิดกันซึ่งจะไม่ทำงาน ณ เวลาเดียวกัน

เมื่อทำการรีเซตรีจิสเตอร์ชี้ข้อมูล (Data Direction Register) จะถูกกำหนดเป็น “0” ซึ่งทำให้ขาทุกขากลายเป็นอินพุต นอกจากนี้บิต 0, 1, 4 และ 5 ในรีจิสเตอร์ควบคุม (Control Register) จะถูกกำหนดเป็น “0” เพื่อให้แน่ใจว่าข้อมูลจะถูกควบคุมด้วยสัญญาณนาฬิกาภายในรีจิสเตอร์เอาต์พุตเมื่อข้อมูลถูกโหลด ส่วนรีจิสเตอร์อื่นๆ ทุกตัวที่เกี่ยวข้องกับพอร์ต D จะไม่ถูกกำหนดสถานะ การทำงานเริ่มต้นเมื่อทำการรีเซต

พอร์ตขนาน D สามารถโปรแกรมเป็นอินพุต/เอาต์พุต แบบแยกขาได้ เมื่อโปรแกรมเป็นเอาต์พุต แล้วสามารถเลือกการทำงานให้เป็นเอาต์พุตแบบขาเปิดหรือเอาต์พุตแบบมาตรฐาน อย่างใดอย่างหนึ่งได้



รูปที่ 2.7 บล็อกโครงสร้างการทำงานของ พอร์ต D

Port D Data Register (PDDR) เป็นรีจิสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะ การทำงานเริ่มต้นของพอร์ตให้เป็น “0” หรือ “1”

Port D Data Direction Register (PDDDR) คือรีจิสเตอร์ชี้ข้อมูลพอร์ตขนาน D นั่นคือถ้าเซตให้เป็น “1” จะทำให้ขาเป็นเอาต์พุต และถ้าเซตเป็น “0” จะเป็นอินพุต

Port D Drive Control Register (PDDCR) คือรีจิสเตอร์ควบคุมขาพอร์ตขนาน D นั่นคือถ้าเซตให้เป็น “0” จะทำให้ขาเป็นเอาต์พุตปกติ (มาตรฐาน) ถ้าเซตให้เป็น “1” จะทำให้ขาเป็นเอาต์พุตแบบขาเปิด

Port D Function Register (PDFR) คือรีจิสเตอร์ฟังก์ชันพอร์ตขนาน D ทำหน้าที่ในการควบคุมพอร์ตให้ทำงานเป็นแบบเอาต์พุตพอร์ตขนานโดยการเซตบิต 4 และบิตที่ 6 เป็น "1"

Port D Bit Register (PDBxR) รีจิสเตอร์ทั้ง 8 ตัวจะใช้ในการเซตเอาต์พุตแบบแยกขา (บิต)

Port D Control Register (PDCR) คือรีจิสเตอร์ควบคุมพอร์ตขนาน D เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการเกิดสัญญาณนาฬิกาแบบขอบขาขึ้น (Upper Nibble) หรือขอบขาลง (Lower Nibble) ของรีจิสเตอร์เอาต์พุตตัวสุดท้ายของพอร์ต

- ขาพอร์ต E (PE0-PE7) พอร์ตขนาน E มีขาที่ทำหน้าที่อินพุต/เอาต์พุต ทั้งหมด 8 ขา (บิต) ซึ่งสามารถโปรแกรมแยกขาให้เป็นอินพุตหรือเอาต์พุตได้ ขา PE7 จะถูกใช้เป็นตัวเลือกชิปพอร์ตแบบ Slave เมื่อมีการใช้งานพอร์ต Slave เอาต์พุตแต่ละขาของพอร์ต E สามารถกำหนดเป็นอินพุต/เอาต์พุตแบบ Strobe นอกจากนี้ 4 ขาของพอร์ต E สามารถใช้เป็นอินพุตร้องขอ Interrupt รีจิสเตอร์เอาต์พุตถูกต่อแบบเลือกการทำและถูกควบคุมด้วย Timer ทำให้สามารถสร้างสัญญาณพัลส์แบบกำหนดคาบเวลาที่ขาของพอร์ต

เมื่อทำการรีเซตรีจิสเตอร์ชี้ข้อมูล (Data Direction Register) และรีจิสเตอร์ฟังก์ชัน (Function Register) จะถูกกำหนดเป็น "0" ซึ่งทำให้ขาทุกขากลายเป็นอินพุตและเป็นการยกเลิกการใช้งาน (Disable) ส่วนฟังก์ชันการสลับ (Alternate) นอกจากนี้บิต 0, 1, 4 และ 5 ในรีจิสเตอร์ควบคุม (Control Register) จะถูกกำหนดเป็น "0" เพื่อให้แน่ใจว่าข้อมูลจะถูกควบคุมด้วยสัญญาณนาฬิกาภายในรีจิสเตอร์เอาต์พุตเมื่อข้อมูลถูกโหลด ส่วนรีจิสเตอร์อื่นๆ ทุกตัวที่เกี่ยวข้องกับพอร์ต E จะไม่ถูกกำหนดสถานะ การทำงานเริ่มต้นเมื่อทำการรีเซต

Port E Data Register (PEDR) เป็นรีจิสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะของการทำงานเริ่มต้นของพอร์ตให้เป็น "0" หรือ "1"

Port E Data Direction Register (PEDDR) คือรีจิสเตอร์ชี้ข้อมูลพอร์ตขนาน E นั่นคือเซต "1" จะทำให้ขาเป็นเอาต์พุต และเซต "0" จะเป็นอินพุต

Port E Function Register (PEFR) คือรีจิสเตอร์ฟังก์ชันพอร์ตขนาน E เมื่อเซต "1" ทำให้ขาเป็นเอาต์พุตที่ทำหน้าที่เป็น อินพุต/เอาต์พุต แบบสโตรป (I/O Strobe) โดยธรรมชาติของอินพุต/เอาต์พุต (I/O Bang Control Register: IBxCR) เป็นการชี้ข้อมูลที่ต้องการเซตไปที่เอาต์พุต อินพุต/เอาต์พุตแบบสโตรปที่ทำงานเมื่อเซต "0" ทำให้ขาเป็นเอาต์พุตโหมดปกติ

Port E Bit Register (PEBxR) คือรีจิสเตอร์แบบแยกการควบคุมโดยการเซตบิตเอาต์พุตให้เป็น ON หรือ OFF ("1" หรือ "0")

Port E Control Register (PECR) คือรีจีสเตอร์ควบคุมพอร์ตขนาน E เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการเกิดสัญญาณนาฬิกาแบบขาขึ้น (Upper Nibble) หรือขาลง (Lower Nibble) ของรีจิสเตอร์เอาต์พุตตัวสุดท้ายของพอร์ต

- ขาพอร์ต F (PF0-PF7) พอร์ตขนาน F คือพอร์ตแบบไบต์ซึ่งแต่ละบิตถูกโปรแกรมมาเพื่อการชี้ข้อมูล (Data Direction) และขับ (Drive) บิตเหล่านี้เป็นอินพุต/เอาต์พุตแบบง่ายซึ่งถูกควบคุมด้วยรีจิสเตอร์

ข้อมูลพอร์ต F (Port F Data Register) ในฐานะที่เป็นเอาต์พุต บิตของพอร์ตจะถูกกระทำในส่วนของบัฟเฟอร์ พร้อมด้วยข้อมูลที่ถูกเขียนไปยังรีจิสเตอร์ ข้อมูลพอร์ต F หรือ PFDR ที่ถูกส่งผ่านไปยังขาเอาต์พุตบนขอบคาบเวลาที่ถูกเลือก เอาต์พุตของไทม์เมอร์ A1 ไทม์เมอร์ B1 หรือ ไทม์เมอร์ B2 สามารถใช้งานกับฟังก์ชันนี้ได้พร้อมด้วยแต่ละขอบขาของพอร์ตที่มีพื้นที่การเลือกแยกกันเพื่อควบคุมคาบเวลานี้

นอกจากนี้อินพุต/เอาต์พุตเหล่านี้ยังถูกใช้เพื่อเข้าถึงรอบนอกอื่นๆ บนชิปในฐานะที่เป็นเอาต์พุต เอาต์พุตพอร์ตขนาน F สามารถนำพาเอาต์พุตมอดูเลเตอร์แบบ PWM 4 ตัว ในฐานะที่เป็นอินพุต อินพุตพอร์ตขนาน F สามารถนำพาอินพุตไปสู่การเข้ารหัสแบบ Quadrature เมื่อพอร์ตอนุกรม C และพอร์ตอนุกรม D ถูกใช้งานในโหมดสัญญาณนาฬิกาแบบอนุกรม 2 ขาของพอร์ต F จะถูกใช้เพื่อนำพาสัญญาณนาฬิกาอนุกรม เมื่อสัญญาณนาฬิกาภายในถูกเลือกให้ทำงานในพอร์ตอนุกรมเหล่านี้ บิตของพอร์ตขนาน F จะถูกเซตให้เป็นเอาต์พุต

Port F Data Register (PFDR) เป็นรีจิสเตอร์ที่ทำหน้าที่ในการควบคุมสถานะ การทำงานเริ่มต้นของพอร์ตให้เป็น “0” หรือ “1”

Port F Control Register (PFCR) คือรีจิสเตอร์ควบคุมพอร์ตขนาน F นั่นคือถ้าเซตเป็น “1” จะเป็นการใช้ฟังก์ชันการสลับขาเป็นเอาต์พุต บิต 7-4 จะใช้เป็นเอาต์พุตแบบ PWM และบิต 1-0 จะเป็นการใช้การสื่อสารพอร์ตอนุกรม C แบบ synchronous และเอาต์พุตสัญญาณนาฬิกา D สำหรับใช้เมื่อพอร์ตอนุกรมถูกกำหนดเพื่อการใช้สัญญาณนาฬิกาภายใน

Port F Drive Control Register (PFDCR) คือรีจิสเตอร์ควบคุมการขับพอร์ตขนาน F ถ้าเซตเป็น “0” จะทำให้ขาทำหน้าที่เป็นเอาต์พุตแบบปกติ (มาตรฐาน) ถ้าเซตเป็น “1” จะทำให้ขาทำหน้าที่เป็นเอาต์พุตแบบขาเปิด (Open Drain) และใช้ในการเขียน (Write) อย่างเดียว

Port F Data Direction Register (PFDDR) คือรีจิสเตอร์ชี้ข้อมูลพอร์ตขนาน F นั่นคือถ้าเซตเป็น “1” จะทำให้ขาทำหน้าที่เป็นเอาต์พุต และถ้าเซตเป็น “0” ขาจะทำหน้าที่เป็นอินพุต

- ขาพอร์ต G (PG0-PG7) พอร์ตขนาน G คือพอร์ตแบบไบต์ซึ่งประกอบด้วยบิตแต่ละบิตที่สามารถโปรแกรมสำหรับการชี้ข้อมูลและขับบิตเหล่านี้เป็นอินพุต/เอาต์พุตแบบง่ายซึ่งถูกควบคุมด้วยรีจิสเตอร์ข้อมูลพอร์ต G (Port G Data Register) ในฐานะที่เป็นเอาต์พุต บิตของพอร์ตจะถูกกระทำในส่วนของบัฟเฟอร์พร้อมด้วยข้อมูลที่ถูกเขียนไปยังรีจิสเตอร์ข้อมูลพอร์ต G ซึ่งถูกส่งผ่านไปยังขาเอาต์พุตบนขอบคาบเวลาที่ถูกเลือกเอาต์พุตของไทม์เมอร์ A1 ไทม์เมอร์ B1 หรือไทม์เมอร์ B2 สามารถใช้งานกับฟังก์ชันนี้ได้พร้อมด้วยแต่ละขอบขาของพอร์ตที่มีพื้นที่การเลือกแยกกันเพื่อควบคุมคาบเวลานี้

นอกจากนี้อินพุต/เอาต์พุตเหล่านี้ยังถูกใช้เพื่อเข้าถึงรอบนอกอื่นๆ บนชิปในฐานะที่เป็นเอาต์พุต เอาต์พุตพอร์ตขนาน G สามารถนำพาข้อมูลและเอาต์พุตสัญญาณนาฬิกาจากพอร์ตอนุกรม E และ F ในฐานะที่เป็นอินพุต อินพุตพอร์ตขนาน G สามารถนำพาข้อมูลและเอาต์พุตสัญญาณนาฬิกาสำหรับพอร์ตอนุกรม 2 พอร์ตนี้

Port G Data Register (PGDR) คือรีจิสเตอร์ข้อมูลพอร์ต G ทำหน้าที่อ่านค่าที่ขาและเขียนค่าข้อมูลไปยังรีจิสเตอร์ด้านโหลด

Port G Control Register (PGCR) คือรีจิสเตอร์ควบคุมพอร์ตขนาน G รีจิสเตอร์นี้ใช้ในการควบคุมขอบขาขึ้นหรือขอบขาลงของสัญญาณนาฬิการีจิสเตอร์เอาต์พุตสุดท้ายของพอร์ต ในการทำรีเซต บิต 0, 1, 4 และ 5 จะรีเซตเป็น "0"

Port G Function Register (PGFR) คือรีจิสเตอร์ฟังก์ชันพอร์ตขนาน G นั่นคือถ้าเซตเป็น "1" จะทำหน้าที่เป็นการใช้ฟังก์ชันการสลับขาเป็นเอาต์พุต บิต 6-2 จะใช้เป็นพอร์ตเอาต์พุตแบบอนุกรมของ E และ F สื่อสารแบบ Asynchronous หรือ SDLC/HDLC สำหรับพอร์ตอนุกรม E และ F (Serial Port E และ F)

Port G Drive Control Register (PGDCR) คือรีจิสเตอร์ควบคุมการขับพอร์ตขนาน F ถ้าเซตเป็น "0" จะทำให้ขาทำหน้าที่เป็นเอาต์พุตแบบปกติ (มาตรฐาน) ถ้าเซตเป็น "1" จะทำให้ขาทำหน้าที่เป็นเอาต์พุตแบบขาเปิดและใช้ในการเขียนอย่างเดียว

Port G Data Direction Register (PGDDR) รีจิสเตอร์ชี้ข้อมูลพอร์ตขนาน F ถ้าเซตเป็น "1" จะทำให้ขาทำหน้าที่เป็นเอาต์พุต และรีจิสเตอร์นี้จะเป็น "0" เมื่อถูกรีเซต
ในการทำรีเซตรีจิสเตอร์ชี้ข้อมูล (Data Direction Register) ถูกรีเซตกลับสถานะเดิมคือ "0" ซึ่งทำให้ขาทั้งหมดเป็นอินพุต นอกจากนี้บิต 0, 1, 4 และ 5 ในรีจิสเตอร์ควบคุม (Control Register) จะถูกเซตเป็น "0" เพื่อให้แน่ใจว่าข้อมูลจะถูกสัญญาณนาฬิกาควบคุมภายในรีจิสเตอร์เอาท์พุตเมื่อข้อมูลถูกโหลด รีจิสเตอร์อื่นๆ ที่เกี่ยวข้องกับพอร์ต G จะไม่ถูกกำหนดสถานะ การทำงานเริ่มต้นจากการทำรีเซต

2.4 โปรแกรม Dynamic C

Dynamic C คือ ระบบการพัฒนารวมของการเขียนซอฟแวร์สร้างมาจากระบบคอมพิวเตอร์ของ IBM ซึ่งออกแบบให้เข้ากับคอมพิวเตอร์ทั่วไปให้ใช้งานได้

Dynamic C มีการใช้ในเครือข่ายตั้งแต่ปี ค.ศ. 1989 ถูกออกแบบสำหรับการเขียนโปรแกรมฝังระบบ และมีความสามารถตรวจสอบในตัวของมันเองอย่างรวดเร็ว สำหรับไมโครโปรเซสเซอร์ Rabbit 3000 ที่ใช้งานกันทั่วไปสามารถติดต่อกันโดย สายแพ 10 สายที่พอร์ต B โปรแกรมพื้นฐานของระบบมีข้อมูลประมาณ 1000 ไบท์ที่ใช้ในการจัดเตรียม Debugging และการติดต่อข้อมูลต่างๆ Dynamic C ต้องการ BIOS เพื่อใช้ในการตรวจสอบโปรแกรมเพื่อที่จะใช้งานได้สะดวก ถ้าผู้ใช้หยุดการ RUN โปรแกรมและใช้โปรแกรมใหม่ BIOS ก็จะเช็คการทำงานใหม่ตลอด Dynamic C ออกแบบให้เข้ากับภาษา Assembly หรือใช้ได้กับโปรแกรมภาษา C

2.4.1 คุณสมบัติเด่นต่างๆ ของโปรแกรมเครื่องมือ Dynamic C

-โปรแกรมเครื่องมือ Dynamic C มี Text Editor อยู่ภายใน โปรแกรมสามารถจะ Executed หรือ Debugged โปรแกรมในระหว่างการทำงานในระดับ Source code หรือ Machine code ได้

- โปรแกรมเครื่องมือ Dynamic C สามารถรองรับการเขียนโปรแกรมด้วยภาษา Assembly ได้ กล่าวคือ ไม่จำเป็นที่ผู้อ่านจะพัฒนาโปรแกรมโดยใช้ภาษา C หรือ ภาษา Assembly อย่างใดอย่างหนึ่ง แต่

ในโปรแกรมเครื่องมือ Dynamic C นี้ผู้อ่านสามารถจะพัฒนาโปรแกรมโดยการรวม 2 ภาษานี้ไว้ในโค๊ดโปรแกรมเดียวของผู้ใช้ได้

- การ Debug ภายใต้โปรแกรมเครื่องมือ Dynamic C ผู้ใช้สามารถใช้คำสั่ง Printf, Watch expressions, Breakpoints และ อื่นๆ ในการดูผล และ ตรวจสอบ ข้อมูลต่างๆได้

- โปรแกรมเครื่องมือ Dynamic C รองรับการใช้งานแบบ Real-world embedded system development นอกจากนั้นโปรแกรมเครื่องมือ Dynamic C ยังรองรับการทำงานแบบ Cooperative และ Preemptive multi-tasking อีกด้วย

- โปรแกรมเครื่องมือ Dynamic C ได้เตรียมฟังก์ชัน และ ไลบรารี ไว้มากมาย ซึ่งไลบรารีเหล่านี้สามารถรองรับการโปรแกรมในแบบ Real-time, Machine level I/O และรองรับฟังก์ชันมาตรฐานที่เกี่ยวกับการคำนวณทางคณิตศาสตร์ และ ข้อความ ไว้อย่างครบถ้วน

- การ Compile ของโปรแกรมเครื่องมือ Dynamic C นั้นจะ Compile ตรงไปที่หน่วยความจำทันที ฟังก์ชัน และ ไลบรารี ต่างๆ จะถูก Compile, Link และ Download ต่อเนื่องกันโดยอัตโนมัติ โปรแกรมเครื่องมือ Dynamic C สามารถจะโหลดโค๊ดโปรแกรม 30,000 bytes ใน 5 วินาที ที่ Baud rate เท่ากับ 115,200 bps

2.4.2 Using Dynamic C

ผู้ใช้โปรแกรม มีตัวเลือกในการที่จะพัฒนาซอฟท์แวร์ ภาษาเขียนใน Flash Memory ขนาด 256 Kbyte หรือ ใน Static Ram ขนาด 128 Kbyte ผลการทำงานในหน่วยความจำ คือการบันทึกข้อมูลสามารถบันทึกได้ถึง 100,000 ครั้งของการเขียน

ข้อเสียของการใช้ Flash Memory เมื่อมีการดีบักโปรแกรมเพื่อขัดจังหวะการทำงานจะทำให้ Interrupt เกิดข้อผิดพลาด การทำงานของโปรแกรมก็จะหยุดตามไปด้วย

Dynamic C เป็นภาษาที่ใช้ในการสนับสนุน TCP/IP โดย Dynamic C จะประกอบไปด้วย Libraries ต่างๆโดยจะมี Libraries หลักคือ DCRTCP.LIB อีกทั้ง Dynamic C ยังมี Libraries ส่วนของ DNS (Domain Name Server), IP, TCP and UDP (User Datagram Protocal) คือ DNS.LIB, IP.LIB, NET.LIB, TCP.LIB and UDP.LIB ส่วนในการติดต่อหรือในส่วนของชั้นเครือข่ายของโปรโตคอล TCP/IP จะมี Libraries ที่ชื่อ ARP.LIB และ ICMP.LIB

ในส่วนของ Libraries หลัก DCRTCP.LIB จะประกอบไปด้วย macros ที่ทำการตั้งค่าไว้ ส่วนโครงสร้างข้อมูลและฟังก์ชันที่ใช้กับ IP เวอร์ชั่น 4 ได้มีการสนับสนุน โดย DCRTCP.LIB ในการคอมไพล์TCP/IP จะต้องให้ส่วนของบอร์ดควบคุมรู้ค่า IP address, netmask and default gateway

2.4.3 การเซตค่า IP Address

ค่า IP Address มีความจำเป็นที่ต้องทำการเซตค่าในช่วงของการคอมไพล์ โดยจะกำหนดการตั้งค่าไว้ที่ MY_IP_ADDRESS, MY_NETMASK, MY_GATEWAY และ MY_NAMESERVER ตามลำดับ ในช่วงของการคอมไพล์ในส่วนของฟังก์ชัน tcp_config sethostid, sethostname โดยสามารถที่จะควบคุมได้โดย macros

2.4.4 IP Address Set Dynamically

Library BOOTP.LIB จะยอมให้บอร์ดในส่วนของ BOOTP หรือ DHCP ของ Client ให้เป็นส่วนที่จะนำไปใช้ โดยโปรโตคอลจะยึดหลักของขนาดของเครื่องเซิร์ฟเวอร์ที่ทำการติดตั้งบนเครือข่ายภายในเครื่องเซิร์ฟเวอร์ BOOTP และ DHCP จะติดตั้งในส่วนกลางของระบบเครือข่ายภายในและจะคอยจัดการในการวางแผนงานของระบบเครือข่าย

โปรโตคอลทั้ง 2 นี้ จะมีค่าพารามิเตอร์ที่ใช้ในการส่งไปยัง Client รวมถึง

- IP address ของ Client
- Net mask
- รายชื่อ Gateway
- Host และรายชื่อโดเมนพื้นฐาน
- รายชื่อของเครื่องเซิร์ฟเวอร์

ในการนำไปใช้งาน ต้องโปรแกรมดังนี้

#define USE_DHCP

#use DCRTCP.LIB

2.4.5 BOOTP/DHCP Control Macros

มาโครต่างๆ สามารถที่จะทำการควบคุมการ DHCP หากมีการเซตค่าก่อนบรรทัด #use "dcrtcp.lib" ในส่วนโปรแกรมประยุกต์ USE_DHPC ถ้ามาโครนี้ถูกกำหนดจุดมุ่งหมายในการใช้ BOOTP หรือ DHCP เพื่อแก้ไขตัวแปรที่ต้องการ ถ้า USE_DHCP ไม่ถูกกำหนดจะทำให้ MY_IP_ADDRESS, MY_NETMASK, MY_GATEWAY และ MY_NAMESERVER อาจจะถูกกำหนดโดยในส่วนของโปรแกรมประยุกต์

2.4.6 Size for TCP/IP I/O Buffers

ในการเริ่มทำงานของไดนามิค C เวอร์ชัน 8.01 บัฟเฟอร์ TCP และ UDP I/O C ได้มีการแบ่งแยกออกมาเป็น

TCP_BUF_SIZE ได้มีการกำหนดขนาดบัฟเฟอร์ของ TCP ไว้ที่ 4096 ไบท์

UDP_BUF_SIZE ได้มีการกำหนดขนาดบัฟเฟอร์ของ UDP ไว้ที่ 4096 ไบท์

ถ้า SOCK_BUF_SIZE มีการกำหนดจะทำให้ค่า TCP_BUF_SIZE และ UDP_BUF_SIZE จะตรงกับ SOCK_BUF_SIZE แต่ถ้า SOCK_BUF_SIZE ไม่มีการกำหนด จะทำให้ค่า TCP_BUF_SIZE และ UDP_BUF_SIZE จะเท่ากับ tcp_MaxBufSize*2

2.4.7 Number of Sockets

การเริ่มต้นการทำงาน Dynamic C เวอร์ชัน 8.01 จะมีการกำหนดมาโคร 2 ตัว ให้กับหมายเลขซ็อกเกตได้ดังนี้

MAX_TCP_SOCKET_BUFFERS จะมีการกำหนดหมายเลขสูงสุดให้กับซ็อกเกตของ TCP โดยจะมีการเรียก tcp_open() หรือ tcp_listen()

MAX_UDP_SOCKET_BUFFERS จะมีการกำหนดหมายเลขสูงสุดให้กับซ็อกเกตของ UDP udp_open()

2.4.8 Passive Open

จะมีอยู่ 2 เส้นทางในการเปิดซ็อกเกตของ TCP คือ passive และ active ซึ่งการเปิดซ็อกเกตนี้จะต้องทำการเรียก tcp_listen(); โดยจะทำการรอการติดต่อกับอุปกรณ์และชนิดของการเปิดจะใช้กับเซิร์ฟเวอร์ของอินเตอร์เน็ตหรืออาจจะให้ tcp_listen() เป็นตัวชี้ตำแหน่งของข้อมูลของ tcp_Socket ถ้าคุณต้องการที่จะติดต่อโดยคุณได้รับข้อตกลงของหมายเลขพอร์ตและ IP Address โดยทำการเซตค่าให้เป็นศูนย์หรือหนึ่งทั้งคู่

2.4.9 Active Open

เมื่อมีการเรียกหน้าเว็บบราวเซอร์ก็จะทำการเชื่อมต่อกับเซิร์ฟเวอร์เพื่อทำการเปิด โดยจะทำการเรียกใช้ tcp_open() และยังใช้พารามิเตอร์คล้าย ๆ กับการใช้ tcp_listen() โดยหลักพารามิเตอร์นี้จะเกี่ยวกับ IP Address และหมายเลขพอร์ตที่คุณสามารถทำการติดต่อ โดยทำการเซตค่าพารามิเตอร์ 1 พอร์ต ให้เป็นศูนย์ ส่วน DCRTCP.LIB เป็นการแสดงถึงการเลือกพอร์ตระหว่าง 1024 และ 65535 หาก tcp_open() กลับค่ามาเป็นศูนย์อีกครั้งจะไม่สามารถทำการติดต่อได้

2.4.10 Delay a Connection

การยอมรับการร้องขอการติดต่อเมื่อกระบวนการวิธีการร้องขอไม่เหมาะสมก็จะทำการเรียกฟังก์ชัน tcp_reservport() เมื่อมีการตอบรับการติดต่อพารามิเตอร์ในส่วนหัวของ TCP จะทำการเซตค่าเป็นศูนย์ในช่วงเวลานี้จะทำการรอค่าจากพารามิเตอร์ tcp_clearreserve (port number) เพื่อทำการดูแลในการติดต่อ นอกจากนี้ยังมีมาโคร USE_RESERVEDPORTS จะทำการกำหนดเพื่อให้เกิดการทำงานขึ้นของสองฟังก์ชันคือเมื่อมีการใช้ tcp_reserveport, 2MSL (Maximum Segment Lifetime)

2.4.11 TCP/IP Stack Initialization

ใน TCP/IP จะมีฟังก์ชัน Main() เป็นส่วนเริ่มแรกและจะมีการเรียก sock_init() เพื่อใช้กับโครงสร้างข้อมูลภายในและชิป Ethernet หรือชิป RealTek ส่วน DCRTCP.LIB จะคอยจัดการกับแพ็คเกตที่เข้ามา

2.4.12 Packet Processing

เมื่อไรก็ตามเมื่อแพ็คเกตมีการนำเข้ามาจะมีการเรียก tcp_tick(), tcp_open(), udp_open(), sock_read, sock_write, sock_close และ sock_abort ซึ่งเป็นวิธีที่ดีที่สุดที่ tcp_tick() จะรู้ระยะเวลาในการประมวลผลโปรแกรมที่ส่งเข้ามาในรูปของแพ็คเกจ

2.4.13 Function Reference

ในส่วนนี้จะกล่าวถึงฟังก์ชันที่ใช้ใน DCRTCP.LIB ในเริ่มต้นนั้น DCRTCP.LIB ของ Dynamic C 7.05 จะมีฟังก์ชันที่มีการใช้อยู่ภายนอก DNS.LIB, IP.LIB. NET.LIB, TCP.LIB และ UDP.LIB และยังมีฟังก์ชันอีกส่วนหนึ่งที่สามารถเลือกใช้ได้ เช่น ARP.LIB, ICMP.LIB, BSDNAME.LIB และ XMEM.LIB

2.4.14 Macros

- MAX_SOCKETS มาโครนี้จะเป็นตัวกำหนดหมายเลขซ็อกเกตเพื่อทำการจัดสรรพื้นที่
- MY_GATEWAY มาโครนี้จะทำการส่งค่านี้เพื่อไปควบคุม Gateway ในช่วงเวลา Runtime
- MY_NETMASK มาโครนี้เป็นพื้นฐานของ Netmask ที่ใช้ในการควบคุม
- MY_NAMESERVER มาโครนี้เป็นการแสดงชื่อของเซิร์ฟเวอร์ในช่วง Run Program
- MY_DOMAIN มาโครนี้จะทำการควบคุมตำแหน่งเริ่มต้นของโดเมนในช่วงเวลา Runtime
- MY_IP_ADDRESS มาโครนี้จะทำการระบุตำแหน่ง IP Address เพื่อใช้ในการควบคุมในช่วงเวลา Runtime

2.5 HTTP Server

HTTP (Hypertext Transfer Protocol) เซิร์ฟเวอร์จะทำการสร้าง HTML (Hypertext Markup Language) เป็นส่วนที่ทำให้ Client ได้มีการเรียกใช้งานของเอกสารได้ง่ายขึ้น

Server Spec Type Field

SSPEC_FILE	ข้อมูลของไฟล์ที่ใช้
SSPEC_VARIABLE	ข้อมูลชนิด variable ที่ใช้ใน HTTP
SSPEC_FUNCTION	ข้อมูลของฟังก์ชันที่เรียกใช้งาน

โครงสร้างข้อมูลเครื่องเซิร์ฟเวอร์ HTTP

โครงสร้างข้อมูลใน HTTP.LIB ที่น่าสนใจมีอยู่ 4 ส่วนเพื่อผู้พัฒนาเครื่องเซิร์ฟเวอร์ HTTP นำไปพัฒนาต่อคือ

Http Spec

โครงสร้างข้อมูล Http Spec จะบรรจุแฟ้มของทั้งหมด ตัวแปร และฟังก์ชันการทำงานของเครื่องเว็บเซิร์ฟเวอร์ที่ใช้ในการเข้าถึงข้อมูลโครงสร้าง Server Spec มาจาก ZSERVER.LIB

-Type

ส่วนพื้นที่นี้จะเป็นตัวบอกเครื่องเซิร์ฟเวอร์ว่ามีไฟล์ข้อมูล ตัวแปร หรือฟังก์ชันเข้ามา (HTTPSPEC_FILE, HTTPSPEC_VARIABLE หรือ HTTPSPEC_FUNCTION ตามลำดับ)

-Name

พื้นที่นี้จะทำการระบุชื่อเฉพาะที่ใช้ในการอ้างอิงส่วนที่มีการนำข้อมูลเข้ามา

-Data

เป็นส่วนของตำแหน่งทางกายภาพของข้อมูล

-Addr

ส่วนนี้จะทำหน้าที่เป็นตัวชี้ข้อมูลไฟล์ของข้อมูลทั้งหมดต้องตำแหน่งทางกายภาพ ซึ่งตัวแปรและฟังก์ชันจะต้องใช้ตัวชี้ตำแหน่งของข้อมูล

-Varitype

ส่วนนี้จะเป็นตัวบอกชนิดของตัวแปร เช่น INT8, INT16, PTR16, INT32, FLOAT

-Format

ส่วนนี้จะเป็นการอ้างอิงถึงรูปแบบ printf เพื่อประกาศตัวแปร

-Realm

เป็นส่วนของชื่อและรหัสผ่านที่ใช้ในการเข้าถึงข้อมูล

Http Type

โครงสร้าง Http Type จะเกี่ยวข้องกับการขยายไฟล์ข้อมูลร่วมกับ MIME (Multipurpose Internet Mail Extension) และฟังก์ชันที่เกี่ยวข้อง MIME ผู้ใช้ยังสามารถควบคุม HTTP_MAXNAME ได้

Http Realm

โครงสร้าง Http Realm จะมีการใช้งานที่ควบคู่กันคือ ID ของผู้ใช้และรหัสผ่านสำหรับพื้นที่ที่การเรียก Realm โดย Realm จะมีการป้องกันข้อมูลในเครื่องเซิร์ฟเวอร์อีกทั้งยังมีการแบ่งพื้นที่เพื่อทำการเซตส่วนของพื้นที่ที่ใช้ในการป้องกันข้อมูล

ในโครงสร้าง Http Spec จะมีตัวชี้โดยตัวชี้นี้จะมีโครงสร้างชนิด Http Realm เพื่อเป็นรหัสผ่านเพิ่มชื่อ การเพิ่มรหัสผ่าน ถ้าหากไม่ต้องการรหัสผ่านหรือการป้องกันก็ไม่ต้องใช้

Http State

การใช้สำหรับฟังก์ชัน CGI

2.6 Configuration Macros

Macros ใน HTTP.LIB มีดังต่อไปนี้

HTTP_MAXNAME

เป็นขนาดความยาวที่สุดของชื่อในโครงสร้างของ Http Spec ซึ่งจะมีตัวอักษรได้ 20 ตัวอักษร โดยความยาวสูงสุดของชื่อใดๆนั้นจะมีได้ 19 ตัวอักษร เพราะว่าหนึ่งตัวอักษรได้มีการถูกใช้สำหรับเป็นส่วนท้าย

HTTP_MAXRAMSPEC

เป็นหมายเลขสูงสุดของส่วน Http Spec ซึ่งสามารถทำการเพิ่มการ runtime มาโครนี้ควบคุมโดย SSPEC_MAXSPEC

HTTP_MAXSERVERS

เป็นจำนวนสูงสุดของ HTTP servers บนพอร์ตหมายเลข 80 แต่ไม่เกิน 4 server

HTTP_PORT

มาโครนี้ยอมให้ผู้ใช้สามารถควบคุมพอร์ตโดยการกำหนด จะต้องกำหนดก่อนบรรทัด #use http.lib

Function

-cgi_redirectto

รูปแบบ

Void cgi_redirectto(HttpState*state,char*url);

คำอธบาย

เป็นส่วนที่ทำการเรียกฟังก์ชัน CGI

พารามิเตอร์

State = จะทำเป็นตัวเครื่องเซิร์ฟเวอร์หรือเป็นตัวรับฟังก์ชัน CGI

url = จะเป็นตัวแสดงคุณสมบัติของ url

Library

HTTP.LIB

-http_handler

รูปแบบ

int http_handler();

คำอธิบาย

เป็นฟังก์ชันพื้นฐานที่ใช้ขยายความของฟังก์ชันต่างๆ

Library

HTTP.LIB

-http_init

รูปแบบ

int http_init(void);

คำอธิบาย

เริ่มต้นการทำงานของ HTTP

Library

HTTP.LIB

2.7 โปรโตคอล TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นโปรโตคอลที่ถูกพัฒนาขึ้นเพื่อให้สามารถสื่อสารกันผ่านเครือข่ายที่ต่างระบบกันได้ จึงเป็นโปรโตคอลที่มีการใช้งานอย่างแพร่หลาย และเนื่องจากขั้นตอนการสื่อสารระหว่างคอมพิวเตอร์เป็นสิ่งที่ค่อนข้างซับซ้อน ดังนั้นโปรโตคอลจึงแบ่งเป็นชั้นย่อย (Layer) เพื่อเป็นการแยกการทำงานของ Application ของผู้ใช้ออกจาก Hardware ที่ใช้

รับส่งข้อมูลผ่านเครือข่าย โปรโตคอลชุดนี้จะมีการจัดรูปแบบที่แตกต่างจากแบบอ้างอิง OSI (Open System Interconnect) เล็กน้อย โดยมีข้อแตกต่างดังนี้

ตารางที่ 2.1 ตารางเปรียบเทียบ TCP/IP และ OSI Reference Model

<table>
<tr><th colspan="2">OSI Reference Model</th><th colspan="3">TCP/IP</th></tr>
<tr><td>7</td><td>Application</td><td rowspan="2">Application</td><td colspan="2" rowspan="2">FTP, Telnet, HTTP, SMTP,
SNMP, DNS, etc.</td></tr>
<tr><td>6</td><td>Presentation</td></tr>
<tr><td rowspan="2">5</td><td rowspan="2">Session</td><td rowspan="2">Host-to-Host</td><td rowspan="2">TCP</td><td rowspan="2">UDP</td></tr>
<tr><td>4</td><td>Transport</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">Network</td><td rowspan="2">Internet</td><td>ICMP, IGMP</td><td rowspan="2">ARP,
RARP</td></tr>
<tr><td>IP</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Data Link</td><td rowspan="2">Network Access</td><td colspan="2" rowspan="2">Not Specified</td></tr>
<tr><td>1</td><td>Physical</td></tr>
</table>

Internet Protocol (IP) ทำหน้าที่จัดการเกี่ยวกับการรับส่ง Packet หรือ Datagram ซึ่งเป็นหน่วยของข้อมูลที่รับมาจากโปรโตคอลที่อยู่ Layer ที่สูงกว่าเช่น TCP และ UDP

IP Address คือ เลขที่บอกที่อยู่เฉพาะของโหนดที่อยู่ในเครือข่าย รวมถึงคอมพิวเตอร์และเราเตอร์ที่อยู่บนระบบเครือข่าย IP Address ถูกจัดให้อยู่บน Layer ที่ 3 คือ Network Later โดย IP Address ของแต่ละเครื่องที่อยู่ในเครือข่ายเดียวกันต้องไม่ซ้ำกัน อย่างไรก็ตาม IP Address อาจมีได้มากกว่าหนึ่งหมายเลขสำหรับหนึ่งเครื่องก็ได้

2.8 พื้นฐาน ET-GSM SIM 300CZ

ET-GSM SIM 300CZ เป็นโมดูลสื่อสารระบบ GSM/GPRS รองรับระบบการสื่อสาร GSM ความถี่ 900/1800/1900 MHz โดยสั่งงานผ่านพอร์ตอนุกรม RS 232 ด้วยชุดคำสั่ง AT Command สามารถประยุกต์ใช้งานได้หลายรูปแบบ ไม่ว่าจะเป็นการรับส่งสัญญาณแบบ Voice, SMS และ Data

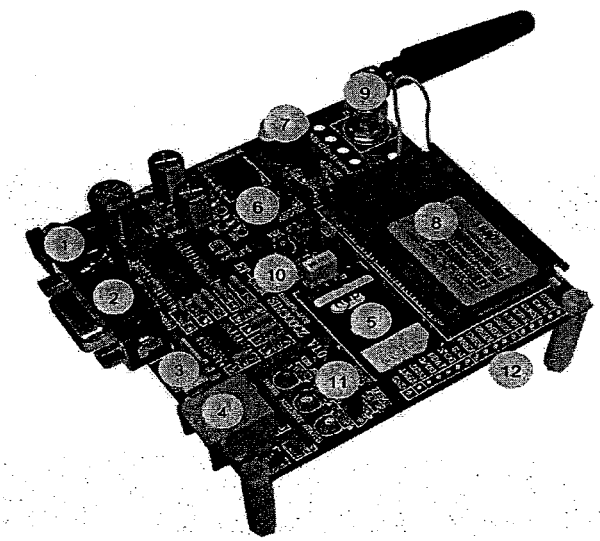
2.9 คุณสมบัติของบอร์ด ET-GSM SIM300CZ V1.0

- มีสวิตช์แบบ Push-Button สำหรับใช้สั่ง เปิด-ปิด การทำงานของโมดูลภายในบอร์ด
- มี Socket SIM รองรับ SIM Card พร้อมวงจร ESD ป้องกัน SIM เสียหาย
- มีวงจร Regulate แยกอิสระ จำนวน 2 ชุด สามารถใช้กับแหล่งจ่ายภายใน Adapter ขนาดตั้งแต่ +5V ขึ้นไป สามารถจ่ายกระแสให้กับโมดูล SIM300CZ และอุปกรณ์เชื่อมต่อต่างๆได้อย่างเพียงพอ

- มีวงจร Regulate ขนาด 4.2V/3A สำหรับจ่ายให้กับโมดูล SIM300CZ ได้อย่างเพียงพอ สามารถใช้กับ SIM ของระบบ GSM900MHz แบบ 2-Watt ได้อย่างไม่เกิดปัญหา
- มีวงจร Regulate ขนาด 3.3V/1A สำหรับจ่ายให้กับวงจรเชื่อมต่อภายนอกดังกระแสเกินพิกัดและสะดวกต่อการออกแบบวงจรเชื่อมต่อเพิ่มเติม โดยไม่ต้องกังวลว่ากระแสจะไม่พอจ่ายให้กับอุปกรณ์

- มีวงจร Line Driver สำหรับแปลงระดับสัญญาณลอจิกจากโมดูล SIM300CZ ให้เป็น RS-232 ระดับมาตรฐานครบทุกเส้นสัญญาณ ทั้งพอร์ตที่ใช้ในการสื่อสารสำหรับสั่งงานโมดูล และพอร์ตสำหรับใช้ในการพัฒนาโปรแกรม (Debug) สามารถเชื่อมต่อกับพอร์ต RS-232 มาตรฐานได้ทันที
- มี LED แสดงสถานะพร้อมในบอร์ด สำหรับแสดงสถานะของแหล่งจ่ายไฟ สถานะพร้อมทำงานของโมดูล สถานะในการเชื่อมต่อกับ Network และ Power-On/Power-Off ของโมดูล
- มีขั้วสำหรับเชื่อมต่อกับ Handset (ชุดปากพูด และหูฟัง ของโทรศัพท์บ้าน) โดยใช้ขั้วต่อแบบ RJ11 มาตรฐาน พร้อมวงจร Voice Filter สามารถนำชุด Handset ของโทรศัพท์บ้าน ต่อเข้ากับบอร์ดทางขั้วต่อแบบ RJ11 สำหรับใช้พูดคุย โทรออก และรับสายได้โดยสะดวก
- มี Buzzer พร้อมวงจรขับเพื่อสร้างสัญญาณเสียง ในกรณีมีการโทรเรียกเข้ามายังโมดูล
- มีจุดยึดเสาอากาศ สำหรับใช้เป็นจุดพักสำหรับเชื่อมต่อกับเสาอากาศแบบต่างๆได้โดยสะดวก
- มีขั้วต่อสำหรับติดตั้งโมดูล SIM300CZ พร้อมเสารองและสกรูยึดโมดูลกับตัวบอร์ด
- มีจุดต่อสัญญาณอื่นๆที่เหลือจากโมดูล เช่น Keyboard, Display, GPIO, Battery Charger ฯลฯ สำหรับให้ผู้ใช้ต่อขยายไปยังวงจรที่ออกแบบเพิ่มเติมได้โดยง่ายและสะดวก

2.10 โครงสร้างของบอร์ด ET-GSM SIM300CZ V1.0



รูปที่ 2.8 โครงสร้างของบอร์ด ET-GSM SIM300CZ

- หมายเลข1 เป็น JACK DC-IN แบบมีขั้ว โดยมีด้านนอกเป็นขั้วบวก และด้านในเป็น GND ใช้สำหรับรับแหล่งจ่ายไฟจากภายนอกโดยออกแบบให้ใช้กับ แหล่งจ่ายไฟขนาด 5V ขึ้นไปที่จ่ายกระแสได้ 1A ถึง 3A
- หมายเลข 2 เป็นขั้วต่อ RS-232 (DCE) แบบ DB9 ตัวเมีย สำหรับใช้เชื่อมต่อกับสัญญาณ RS-232 (DTE) แบบ DB9 ตัวผู้ จากคอมพิวเตอร์ PC หรืออุปกรณ์ภายนอกอื่นๆโดยใช้สาย 9 Pin แบบต่อตรง
- หมายเลข 3 เป็น ขั้วต่อ DEBUG ใช้สำหรับพัฒนา และ DEBUG โปรแกรม สำหรับต่อกับ RS-232 ในกรณีที่ต้องการพัฒนาโปรแกรมเพิ่มเติมให้กับโมดูล SIM300CZ เอง
- หมายเลข 4 เป็น ขั้วต่อ RJ11 สำหรับใช้เชื่อมต่อกับชุด Handset ในกรณีที่ต้องการใช้งานโมดูล SIM300CZ เพื่อโทรออกและรับสาย โดยสามารถเชื่อมต่อกับ Handset มาตรฐานได้ทั่วไป
- หมายเลข 5 เป็น Socket สำหรับติดตั้ง SIM Card ให้กับโมดูล
- หมายเลข 6 เป็น Switch Push-Button สำหรับใช้ Power-On และ Power-OFF ตัวโมดูล
- หมายเลข 7 เป็น Buzzer สำหรับสร้างเสียงเรียกเข้าในกรณีที่มีการโทรเข้ามายังโมดูล SIM300CZ
- หมายเลข 8 เป็นจุดรองรับโมดูล SIM300CZ พร้อมเสาและสกรูสำหรับยึดโมดูลกับบอร์ด
- หมายเลข 9 เป็นจุดยึด Connector เสาอากาศ GSM/GPRS ย่านความถี่ 900/1800/1900 MHz
- หมายเลข 10 เป็น LED แสดงแหล่งจ่าย VBAT โดยจะติดสว่างเมื่อมีการจ่ายไฟให้บอร์ดแล้ว
- หมายเลข 11 เป็น LED แสดงสถานะของบอร์ด ซึ่งมีด้วยกัน 3 ดวงคือ
    - POWER สีแดง จะติดสว่าง เมื่อโมดูลอยู่ในสถานะ Power-ON
    - NETLIGHT สีเหลือง จะกระพริบ เมื่อโมดูลอยู่ในสถานะ Power-ON
    - STATUS สีเขียว จะติดสว่าง เมื่อโมดูลอยู่ในสถานะ Power-ON
- หมายเลข 12 เป็น จุดต่อสัญญาณเพิ่มเติมในกรณีที่ต้องการประยุกต์ใช้งานโมดูลเพิ่มเติม

2.11 คุณสมบัติของโมดูล SIM300CZ

- รองรับความถี่ GSM/GPRS 900/1800/1900MHz
- รองรับ GPRS Multi-Slot Class 10 และ GPRS Mobile Station Class B
- รองรับมาตรฐานคำสั่ง AT Command (GSM07.07/07.05 และคำสั่งเพิ่มเติมจาก SIMCOM)
- รองรับ SIM Applications Toolkit
- ทำงานที่ย่านแรงดัน 3.4V ถึง 4.5V
- รองรับการเชื่อมต่อภายนอก
    - ใช้ได้กับ SIM 3V และ 1.8V
    - มีวงจร Analog Audio (MIC & Speaker) จำนวน 2 ชุด
    - รองรับ 5x5 Keypad Interface & SPI LCD Interface
    - มีระบบ RTC พร้อมวงจร Back up

- มีขั้วต่อเสาอากาศภายนอกออกแบบ Connector และจุดเชื่อมต่อแบบ PAD
- มีระบบ Battery Charge ในตัว

2.12 อุปกรณ์แสดงการทำงานของโดมูล SIM300CZ

สำหรับบอร์ด ET-GSM SIM300CZ V1.0 นั้น ได้ออกแบบอุปกรณ์แสดงผลการทำงานของบอร์ดไว้ในบอร์ดเพื่อใช้แสดงสถานะของการทำงานต่างๆให้ผู้ใช้ทราบด้วย คือ

- Buzzer ใช้แสดงในการทำงานของโมดูลเมื่อมีสายเรียกเข้า โดยการทำงานของ Buzzer นี้จะถูกควบคุมสัญญาณ BUZZER (PIN23) ของโมดูล SIM300CZ และสามารถปรับความดังของเสียงได้จากคำสั่ง "AT+CRSL" ได้อีกด้วย
- LED VBAT ใช้ทำหน้าที่แสดงสถานะของแหล่งจ่ายไฟจากภายนอกที่ต่อมาให้กับบอร์ด โดย LED นี้จะติดสว่างก็ต่อเมื่อมีการจ่ายไฟให้กับบอร์ดเป็นที่เรียบร้อยแล้ว
- LED POWER ใช้แสดงสถานะความพร้อมของโมดูล SIM300CZ ว่าอยู่ในสถานะPower ON หรือ Power OFF โดย LED ตัวนี้จะถูกควบคุมการทำงานด้วยสัญญาณ VDD_EXT(PIN15) ของโมดูลเมื่อทำงานจะมีสถานะทางลอจิกเป็นลอจิก "1" โดยถ้า LED Power ติดสว่าง แสดงว่าโมดูล SIM300CZ อยู่ในสถานะ Power ON และพร้อมทำงาน แต่ถ้า LED นี้ดับ แสดงว่าโมดูลอยู่ในสถานะ Power OFF อยู่
- LED NETLIGHT ใช้แสดงสถานะโมดูลในขณะทำการเชื่อมต่อกับเครือข่ายอยู่ โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณ NETLIGHT (PIN16) ของโมดูล SIM300CZ เมื่อทำงานจะมีสถานะทางลอจิกเป็นลอจิก "1" โดยเมื่อโมดูลอยู่ในสถานะพร้อมทำงาน LED นี้จะติดกระพริบด้วยความเร็วต่างๆ ซึ่งมีความหมายดั้งนี้
  - OFF แสดงว่าโมดูลอยู่ในสถานะของ POWER OFF (ไม่ทำงาน)
  - 64ms ON/800ms OFF แสดงว่า โมดูล SIM300CZ ทำงานปรกติและไม่ได้อยู่ระหว่างทำการค้นหาเครือข่ายอยู่
  - 64ms ON/3000ms OFF แสดงว่าโมดูล SIM300CZ กำลังทำการค้นหาเครือข่ายเพื่อทำกาเชื่อมต่อสัญญาณ
  - 64ms ON/3000ms OFF แสดงว่าโมดูล SIM300CZ อยู่ระหว่างการเชื่อมต่อกับเครือข่ายหรืออุปกรณ์อื่นๆด้วย GPRS อยู่
- LED STATUS ใช้แสดงสถานะของโมดูล SIM300CZ ว่าพร้อมทำงานหรือไม่ โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณ STATUS (pin19) ของโมดูล SIM300 CZ เมื่อทำงานจะมีสถานะทางลอจิกเป็นลอจิก "1" ซึ่งเมื่อ LED นี้ติดสว่างแสดงว่าโมดูลพร้อมรับคำสั่งต่างๆได้ แต่ถ้า LED ดับแสดงว่าโมดูลยังไม่พร้อมทำงาน

2.13 การติดต่อสื่อสารกับโมดูล SIM300CZ

การติดต่อสื่อสารกับโมดูล SIM300CZ ของบอร์ด ET-SIM300CZ นั้นจะเชื่อมต่อผ่านพอร์ต

สื่อสารอนุกรม RS-232 โดยใช้ขั้วต่อแบบ DB9 ตัวเมีย จัดเรียงสัญญาณตามมาตรฐาน RS-232 DCE มาตรฐาน โดยใช้สาย DB9 แบบต่อตรงได้ทันที โดยสัญญาณทั้งหมดที่ DB9 นี้ได้ผ่านวงจร LINE Driver เพื่อแปลงสัญญาณระดับลอจิกจากโมดูลให้เป็นสัญญาณระดับมาตรฐาน RS-232 เป็นที่เรียบร้อยแล้ว ซึ่งถ้าต้องการนำไปเชื่อมกับ RS-232 ของคอมพิวเตอร์ก็สามารถทำการเชื่อมต่อกันโดยตรงได้ทันที โดยไม่ต้องทำการสลับสายสัญญาณใดๆ ทั้งสิ้น โดยสัญญาณเชื่อมต่อทางด้านโมดูล SIM300CZ นั้น จะมีทั้งหมด 8 เส้น หรือ จะแยกต่อเพียง 3 เส้น (RXD, TXD และ GND) ก็ได้เช่นเดียวกัน โดยสามารถกำหนดได้จาก SETUP ค่า Configuration และคำสั่งใช้งาน โดยสัญญาณการเชื่อมต่อ RS-232 ด้านโมดูล SIM300CZ จะมีดังนี้

- pin 1 เป็นขา DCD (Data Carrier Detect) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS-232 แล้ว ซึ่งตามปรกติจะต่อเข้ากับ DCD input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์
- pin 2 เป็นขา TXD (Transmit Data) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS-232 แล้ว ซึ่งตามปรกติจะต่อเข้ากับ RXD (Receive Data) จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 3 เป็นขา RXD (Receive Data) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ สามารถรับสัญญาณระดับ RS-232 ได้โดยตรง ซึ่งตามปรกติจะต่อเข้ากับ TXD (Transmit Data) จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 4 เป็นขา DTR (Data Terminal Ready) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ DTR จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 5 เป็นสัญญาณ GND ของโมดูล SIM300CZ ต้องต่อเข้ากับ GND ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 6 ตามปรกติแล้วเป็นสัญญาณ DSR (Data Set Ready) แต่ในกรณี SIM300CZ จะไม่ได้ต่อใช้งาน แต่อย่างไรก็ตามในบอร์ดได้ทำการป้อนสัญญาณย้อนกลับหรือ Loop Back สัญญาณ DTR (Data Terminal Ready) ซึ่งเป็น Output ส่งมาจาก Host หรือ คอมพิวเตอร์ PC กลับไปแทนโดยจะถูกต่อไปเข้ากับสัญญาณ DSR Input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 7 เป็นขาสัญญาณ RTS (Request To Send) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ RTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin 8 เป็นขาสัญญาณ CTS (Clear To Send) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ CTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

- Pin 9 เป็นขาสัญญาณ RI (Ring Indicator) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ RI ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC

ตารางที่ 2.2 แผนผังการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับ คอมพิวเตอร์ PC

DB9 Female(SIM300CZ)		Signal Direction	DB9 Male(Computer PC)	
Pin	Signal		Signal	Pin
1	DCD	→	DCD	1
2	TXD	→	RXD	2
3	RXD	←	TXD	3
4	DTR	←	DTR	4
5	GND	—	GND	5
6	(DSR)	→	DSR	6
7	RTS	←	RTS	7
8	CTS	→	CTS	8
9	RI	→	RI	9

ตารางที่ 2.3 แผนผังการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับ ไมโครคอนโทรลเลอร์

DB9 Female(SIM300CZ)		Signal Direction	ไมโครคอนโทรลเลอร์
Pin	Signal		Signal
1	TXD	→	RXD
2	RXD	→	TXD
3	GND	—	GND

<u>คำแนะนำ</u> ในกรณีที่ใช้การเชื่อมต่อสัญญาณแบบ 3 เส้น (RXD, TXD, GND) ต้องกำหนดเงื่อนไขของ Flow Control ให้กับโมดูล SIM300CZ เป็น XON/XOFF โดยใช้คำสั่ง "AT+IFC=1, 1"

2.14 คุณสมบัติการทำงานของสัญญาณที่ควรรู้

- RI (Ring Indicator) เป็น Output จากโมดูล SIM300CZ ตามปรกติจะเป็น High แต่เมื่อมีสัญญาณเรียกเข้าจะ Active เป็น Low ตามเงื่อนไขต่อไปนี้

- เมื่อมีสัญญาณเรียกเข้า Voice Calling สัญญาณ RI จะ Active เป็น LOW ค้างอยู่จนกว่าจะมีการตอบรับ (ATA) หรือได้รับคำสั่งยกเลิกการเชื่อมต่อ (ATH) หรือผู้เรียกสายทำการวางสายก่อนจะมีการตอบรับ

- เมื่อมีสัญญาณเรียกเข้า Data Calling สัญญาณ RI จะ Active เป็น LOW ประมาณ 120mS และกลับเป็น HIGH โดยอัตโนมัติ

- DTR (Data Terminal Ready) เป็น Input ของโมดูล SIM300CZ เมื่อต้องการให้โมดูลทำงานต้องให้ขาสัญญาณนี้ได้รับลอจิก LOW ถ้าขา DTR ได้รับลอจิก HIGH โมดูลจะหยุดทำงานและเข้าสู่ Sleep Mode โดยอัตโนมัติ (ถ้ามีการสั่ง Enable Sleep Mode ด้วยคำสั่ง "AT+CSCLK=1" ไว้) ดังนั้นถ้าต้องการให้โมดูลทำงานตลอดเวลาต้องควบคุมให้ขาสัญญาณ DTR ด้านโมดูลได้รับลอจิก LOW โดยการควบคุมจากสัญญาณ DTR ด้านคอมพิวเตอร์ PC หรืออุปกรณ์ที่ควบคุมโมดูลอยู่ให้ทำการ Active สัญญาณ DTR ไว้ตลอดการเชื่อมต่อ สำหรับกรณีที่นำโมดูล SIM30CZไปเชื่อมต่อกับ RS-232 ระบบที่ไม่มีสัญญาณ DTR อยู่เช่น RS-232 ของไมโครคอนโทรลเลอร์บางรุ่นก็อาจเลือกกำหนด Jumper (DTR) ที่อยู่ใกล้กับขั้วต่อ DB9 บนบอร์ดไว้ทางด้าน GND เพื่อให้โมดูลทำงานตลอดเวลา หรือสั่งปิดการทำงานของ sleep Mode โดยใช้คำสั่ง "AT+CSCLK=0" แล้วบันทึกค่า configuration นี้ไว้ก็ได้เช่นเดียวกัน

- ADC (Analog to Digital)เป็น input แบบ ADC (ขา 12 ของโมดูล SIM300CZ) สามารถรับสัญญาณ Analog จากภายนอกได้ระหว่าง 0V ถึง 2.4 V โดยสามารถสั่งอ่านค่าแรงดันที่ขานี้ได้จากคำสั่ง "AT+CADC?" โดยจะได้ผลลัพธ์เป็นค่าระหว่าง 0 ถึง 2400

- GPIO 0 GPIO 1 เป็นขาสัญญาณ i/o สามารถเชื่อมต่อกับสัญญาณลอจิกระดับ 3.3 V

2.15 ชุดคำสั่ง AT Command (AT Command Set)

ชุดคำสั่งนี้ใช้ในการควบคุมการทำงานของโทรศัพท์เคลื่อนที่และโมเด็ม (MODEM) โดยการส่งข้อมูลด้วยรูปแบบการสื่อสารแบบอนุกรม (Serial Communication) โดยที่เกิดจากการคิดค้นของบริษัท Hayes Microcomputer Product Inc.เพื่อใช้งานโมเด็มสำหรับคอมพิวเตอร์ส่วนบุคคล และได้รับความนิยมอย่างมากจนถือเป็นมาตรฐานอันหนึ่ง มาตรฐานคำสั่งนี้มีชื่อเรียกอีกชื่อหนึ่งว่า Hayes Command Set เป็นคำสั่งที่ช่วยให้ผู้ใช้สามารถกำหนดการทำงานต่างๆของโมเด็มได้โดยใช้ซอฟแวร์สั่งงานจากคอมพิวเตอร์ไปยังโมเด็มโดยตรงซึ่งในระบบของโทรศัพท์เคลื่อนที่หลายๆ ยี่ห้อจะมีส่วนของโมเด็มประกอบอยู่ภายในด้วย ดังนั้นจึงสามารถที่จะใช้คำสั่ง AT Command ในการควบคุมการทำงานของโทรศัพท์เคลื่อนที่ได้โดยไม่ต้องใช้การป้อนคำสั่งผ่านทางปุ่มกดของโทรศัพท์เคลื่อนที่เพื่อเข้าถึงฟังก์ชันการทำงานต่างๆ เช่น การสั่งให้มีการโทรออก การส่ง SMS การรับสายเรียกเข้า การปรับความดังของเสียง

ตารางที่ 2.4 ตัวอย่างชุดคำสั่ง AT Command

คำสั่ง	ความหมาย
A/	ทวนคำสั่งล่าสุด
AT	ส่วนที่อยู่ข้างหน้าสำหรับทุกๆคำสั่ง
ATA	เป็นคำสั่งให้มีการตอบรับสัญญาณ ที่มีการเรียกเข้าเมื่อการกระทำคำสั่งนี้ จะเกิดการติดต่อระหว่างปลายทางทั้งสองด้านจะเริ่มขึ้น
ATD<str>	เป็นคำสั่งให้โทรออกอัตโนมัติหรือที่เรียกว่า AUTO DAILING โดยที่<str> จะแทนด้วยอักษร P หรือ T ซึ่งเป็นการแสดงว่าจะใช้ลักษณะการหมุนแบบ PULSE หรือ TONE ไม่จำเป็นต้องกำหนดก็ได้ แต่ส่วนที่สำคัญคือต้องใส่เครื่องหมาย; ไว้ท้ายหมายเลขโทรศัพท์ที่จะทำการโทรออก ตัวอย่างเช่นต้องการโทรไปยังหมายเลข 027390120 จะใช้คำสั่ง
ATD><n>;	เป็นคำสั่งให้โทรออกหมายเลขโทรศัพท์จากสมุดโทรศัพท์ปัจจุบันที่ตำแหน่งเบอร์ n การเลือกสมุดโทรศัพท์ได้โดยใช้คำสั่ง AT+cpbs
ATH	เป็นคำสั่งวางสาย
AT^SCNI	เป็นคำสั่งสำหรับตรวจสอบข้อมูลของเบอร์ที่โทรออก ^SCNI:1[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:2[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:3[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:4[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:5[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:6[,<cs>[,<number>,<type>]]<CR><LF> ^SCNI:7[,<cs>[,<number>,<type>]]<CR><LF> OK/ERROR+CME ERROR <cs>สถานะ การทำงานของหมายเลขที่เกี่ยวข้อง 0 กำลังคอยการเรียกสาย 1 ทำการเรียกสาย 2 การรอสาย <number> หมายเลขโทรศัพท์ <type> ประเภทของหมายเลข
AT+CMGS	คำสั่งที่ใช้สำหรับส่งข้อความ
AT+CMNI	คำสั่งที่ทำให้รู้ว่าได้รับข้อความเข้ามาที่โทรศัพท์ เอาไว้เก็บลำดับข้อความที่เข้ามาว่าเป็นข้อความที่เท่าไร

AT+CMGR	คำสั่งที่ใช้สำหรับอ่านข้อความ
AT+CMGD	คำสั่งที่ใช้สำหรับลบข้อความ

ตารางที่ 2.5 ผลตอบสนองต่างๆ จากโทรศัพท์เมื่อได้รับคำสั่ง

ผลตอบสนอง	ความหมาย
OK	ทำตามคำสั่งสำเร็จ
RING	ตรวจพบสัญญาณกระดิ่ง หรือมีสายเรียกเข้า
NO CARRIER	ไม่สามารถเชื่อมต่อได้สำเร็จหรือยกเลิกการติดต่อ
ERROR	ไม่สามารถปฏิบัติตามคำสั่งได้หรือคำสั่งมีความยาวเกินไป

2.16 SMS (Short Message Services)

ความหมายของ SMS (Short Message Service)

SMS หรือ การส่งข้อความสั้น โดยลักษณะของการส่งข้อความสั้นจะมีลักษณะคล้ายกับการส่งข้อความไปยังเพจเจอร์คือผู้ใช้สามารถส่งข้อความไปยังผู้รับ โดยที่ผู้รับสามารถกดอ่านได้จากเครื่องโทรศัพท์มือถือได้ทันที ข้อดีของ SMS ที่ทำให้ต่างกับเพจเจอร์ก็คือ ผู้ใช้หรือผู้ที่ต้องการส่งข้อความสามารถพิมพ์ข้อความได้เองจากโทรศัพท์มือถือ และสามารถส่งไปยังโทรศัพท์มือถือของผู้รับได้ทันที

SMS เป็นบริการมาตรฐานในการรับส่งข้อความระหว่างโทรศัพท์เคลื่อนที่ และอุปกรณ์อื่นๆ สามารถส่งได้ในรูปแบบของตัวเลข ตัวอักษร และสัญลักษณ์ต่างๆ SMS ได้ถูกสร้างขึ้นมาครั้งแรกให้ทำงานร่วมกับโทรศัพท์เคลื่อนที่แบบดิจิตอลระบบ GSM โดยข้อความแรกได้ถูกส่งในเดือนธันวาคม 1992 จากเครื่องคอมพิวเตอร์ส่วนบุคคลไปสู่เครื่องโทรศัพท์บนโครงข่ายระบบ GSM ของ Vodafone ในประเทศอังกฤษ ปัจจุบันบริการ SMS สนับสนุนโครงข่าย GSM, CDMA และ TDMA สำหรับการส่ง SMS ภาษาไทยจะส่งได้ 70 ตัวอักษร ภาษาอังกฤษส่งได้ 160 ตัวอักษร

เนื่องจากการรับส่ง SMS เป็นเทคนิคการสื่อสารที่ไม่จำเป็นต้องใช้การสร้างวงจรสนทนา (Call Set-up) จึงทำให้สามารถรับหรือส่งข้อความได้ในขณะที่กำลังสนทนาอยู่ หรือในขณะที่เปิดเครื่องทิ้งไว้เฉยๆ บริการ SMS เป็นบริการที่ได้รับความนิยมมากในปัจจุบัน บริการ SMS มีรายละเอียดดังนี้

ตารางที่ 2.6 รายละเอียดของ SMS

Feature	SMS
Store and Forward (non real time)	Yes
Confirmation of message delivery	Yes
Communications Type	Person to Person

Media supported	Text plus binary
Protocols	SMS specific e.g. SMPP
Configuration	Simple telephone number
Platforms	SMS Center
Principle Application	Simple person to person
User behavior	Discrete

บริการ SMS ไม่ใช่บริการแบบ Real time เนื่องจากการส่งข้อความต้องส่งผ่าน Platform กลาง คือ Short Message Center หรือ SMS-C ซึ่งเป็นอุปกรณ์ที่ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ติดตั้งไว้เพื่อให้บริการรับ-ส่งข้อความผ่านทางเครือข่ายโทรศัพท์เคลื่อนที่ไปสู่เครื่องลูกข่ายอื่นๆได้

2.16.1 หลักการทำงานของ SMS

SMS เป็นเทคโนโลยี การรับ-ส่งข้อมูลแบบเก็บและส่งต่อ (Store and Forward) ในเครือข่าย GSM เรียกอุปกรณ์ที่ใช้ในการเก็บและส่งต่อข้อมูลว่า Short Message Service Center (SMS-C)

การใช้งาน SMS กระทำได้โดย เมื่อองค์กรต้องการการส่งข้อความสั้น (จำนวนมากสุด 160 อักษร) ก็จะทำการป้อนข้อความพร้อมทั้งระบุเลขหมายปลายทางที่ต้องการจะส่งไปด้วย แต่เครื่องลูกข่ายที่ต้องการจะส่ง SMS จะต้องระบุเลขหมายของ SMS-C ก่อนจะทำการตรวจสอบเลขหมายปลายทางกับ HLR ว่าเลขหมายปลายทางอยู่ที่ไหนในเครือข่าย เมื่อทราบแล้ว SMS-C ก็จะส่ง SMS ไปยังโทรศัพท์เคลื่อนที่ปลายทาง กรณี SMS ระบุหมายเลขปลายทางเป็นโทรศัพท์เคลื่อนที่นอกเครือข่าย เช่น ส่งจาก DTAC ไป AIS ชุมสายโทรศัพท์ต้นทางในเครือข่าย DTAC (MSC) จะตรวจสอบจากหมายเลขปลายทาง และเมื่อทราบว่าจุดหมายปลายทางเป็นหมายเลขของ AIS ชุมสายโทรศัพท์เคลื่อนที่ (MSC) ของ DTAC จะส่ง SMS ดังกล่าวไปลงที่ SMS-C ของ AIS โดยตรง

2.16.2 รูปแบบการให้บริการของผู้ให้บริการเครือข่าย (Operator) ในปัจจุบัน

รูปแบบการให้บริการของ Operator หรือผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ในปัจจุบันมี 2 ลักษณะ คือ

- Bulk SMS

การทำงานผ่าน Corporate SMS Platform เป็นการให้บริการ SMS ในลักษณะองค์กร (Corporate Short Message Service) ลักษณะการใช้งานของ Bulk คือการส่งจากผู้ส่ง (องค์กร) ไปยังผู้รับ (ลูกค้า) ซึ่งมีได้ 2 ลักษณะ คือ One-To-One ส่งข้อความรายบุคคล และ One-to-Many ส่งข้อความจากต้นทางเดียวถึงปลายทางในเวลาเดียวกัน การเรียกเก็บเงินจะเก็บเงินจากกับองค์กรที่ใช้บริการ

- CPA SMS

การทำงานผ่าน Content Provider Access Platform (CPA Platform) เป็นการบริการที่ Operator (ผู้ให้บริการเครือข่าย) สร้างModel Service เองหรือเปิดบริการให้ตัวแทนที่มีความพร้อมในเรื่องการสร้าง Model Service หรือ ที่เรียกว่า Content Provider นำเสนอโครงการให้ Operator พิจารณาเพื่อดำเนินธุรกิจ

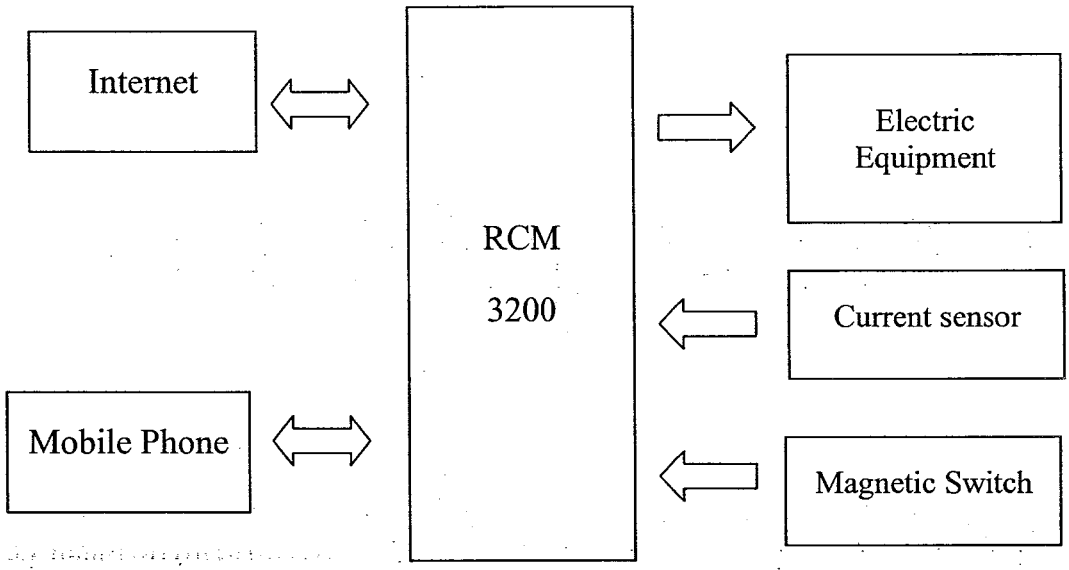
ร่วมกัน (Co-partner) โดย Operator จะเป็นผู้ดำเนินการเก็บค่าบริการให้ ซึ่งการแบ่งส่วนของรายได้ระหว่าง Operator กับ ตัวแทน

2.16.3 ลักษณะของการส่ง SMS

การส่ง SMS หรือ Short Message Service คือ การส่งข้อความสั้นๆ หรือข้อมูลสั้นจากเครื่องโทรศัพท์มือถือผู้ส่ง ไปยังเครื่องโทรศัพท์มือถือของผู้รับโดยส่งผ่านเครือข่ายศูนย์บริการ Short Message Service Center (SMSC) โดยการส่งแบบ SMS นี้เราจะสามารถเลือกได้ว่าจะส่งข้อความสั้น หรือเป็นรูปภาพโลโก้ หรือเสียงเพลงริงโทน ซึ่งจะมีวิธีการส่งที่แตกต่างกัน 2 แบบ คือ โหมดตัวอักษร หรือ Text-Mode และ โหมดพีดียู หรือ PDU (Protocol Data Unit)โดย Text–Mode คือโหมดที่เราสามารถส่งข้อความส่งข้อความสั้นๆประมาณ 160 ตัวอักษร ไปยังเครื่องโทรศัพท์มือถือของผู้รับ โดยลักษณะข้อความนั้นจะอยู่ในรูปแบบรหัส ASCII ส่วน PDU MODE คือโหมดที่สามารถส่งได้ทั้งข้อความสั้นๆ ส่งรูปภาพ และเพลงริงโทนได้ ซึ่ง PDU–Mode จะมารูปแบบการวางข้อมูลที่จะส่งแตกต่างกับ Text-Mode คือ PDU-Mode จะมีการเข้ารหัสที่จะแปลงข้อความเป็นรูปแบบของเลขฐานสิบหก และต้องมีการส่งหัวข้อของชุดข้อมูล (Heading) แต่ในText-Mode จะเป็นการส่งแบบรหัส ASCII และไม่จำเป็นต้องส่งหัวข้อของชุดข้อมูล

บทที่ 3
การคำนวณและการสร้าง

3.1 หลักการทำงานของโครงงาน

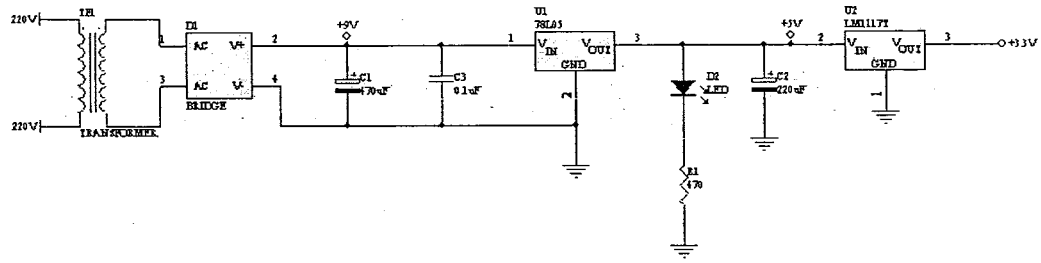


รูปที่ 3.1 บล็อกไดอะแกรมของโครงงานทั้งหมด

จากบล็อกไดอะแกรมแสดงการทำงานของโครงงานจะเห็นว่าการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้านั้นจะสามารถควบคุมผ่านทางเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือ จากนั้นไมโครโปรเซสเซอร์ RCM3200 จะรับคำสั่งและประมวลผลส่งต่อไปเพื่อควบคุมอุปกรณ์ไฟฟ้าและ Magnetic Switch จะทำหน้าที่ตรวจการเปิด-ปิดประตู หน้าต่างจากนั้นจึงส่งข้อมูลไปยังไมโครโปรเซสเซอร์ RCM3200 เพื่อประมวลผลและเตือนภัยให้ทราบพร้อมทั้งแสดงสถานะผ่านทางเว็บบราวเซอร์ได้

3.2 การออกแบบแหล่งจ่ายไฟ

เมื่อป้อนไฟแรงดัน 220 VAC เข้าที่หม้อแปลง แล้วเลือกใช้งานขดทางออกขนาดแรงดัน 6 VAC โดยผ่านไดโอดบริจด์ เรกติไฟเออร์ และต่อเข้ากับ IC เบอร์ LM7805 จะทำให้มีแรงดันเอาต์พุตที่คงที่ มีค่า 5 โวลท์ และต่อเข้ากับ IC เบอร์ LM1117 ซึ่งจะทำให้แรงดันเอาต์พุตมีค่า 3.3 V เพื่อป้อนเป็นไฟเลี้ยงให้กับวงจรโดยมี LED เป็นตัวแสดงสถานะ การทำงาน และคอนเดนเซอร์ C ทำหน้าที่กรองกระแสให้เรียบ



รูปที่ 3.2 วงจรแหล่งจ่ายไฟ

3.3 รีเลย์

รีเลย์เป็นอุปกรณ์ทำหน้าที่เป็นสวิทช์มีหลักการทำงานคล้ายกับขดลวดแม่เหล็กไฟฟ้าหรือโซลินอยด์รีเลย์ใช้ในการควบคุมวงจรไฟฟ้าได้อย่างหลากหลาย รีเลย์เป็นสวิทช์ควบคุมที่ทำงานด้วยไฟฟ้า แบ่งออกตามลักษณะการใช้งานได้เป็น 2 ประเภทคือ

1. รีเลย์กำลัง (Power relay) หรือมักเรียกกันว่าคอนแทกเตอร์ (Contactor or Magnetic contactor) ใช้ในการควบคุมไฟฟ้ากำลัง มีขนาดใหญ่กว่ารีเลย์ธรรมดา

2. รีเลย์ควบคุม (Control relay) มีขนาดเล็ก กำลังไฟฟ้าต่ำ ใช้ในวงจรควบคุมทั่วไปที่มีกำลังไฟฟ้าไม่มากนัก หรือเพื่อการควบคุมรีเลย์หรือคอนแทกเตอร์ขนาดใหญ่ รีเลย์ควบคุมบางทีเรียกกันง่าย ๆว่า "รีเลย์"

ชนิดของรีเลย์แบ่งตามลักษณะของคอยล์หรือแบ่งตามลักษณะการใช้งานได้แก่

1. รีเลย์กระแส (Current relay) คือรีเลย์ที่ทำงานโดยใช้กระแสมีทั้งชนิดกระแสขาด (Under-current) และกระแสเกิน (Over-current)

2. รีเลย์แรงดัน (Voltage relay) คือรีเลย์ที่ทำงานโดยใช้แรงดันมีทั้งชนิดแรงดันขาด (Under-voltage) และแรงดันเกิน (Over-voltage)

3. รีเลย์ช่วย (Auxiliary relay) คือรีเลย์ที่เวลาใช้งานจะต้องประกอบเข้ากับรีเลย์ชนิดอื่นจึงจะทำงานได้

4. รีเลย์กำลัง (Power relay) คือรีเลย์ที่รวมเอาคุณสมบัติของรีเลย์กระแสและรีเลย์แรงดันเข้าด้วยกัน

5. รีเลย์เวลา (Time relay) คือรีเลย์ที่ทำงานโดยมีเวลาเข้ามาเกี่ยวข้องด้วย มีอยู่ด้วยกัน 4 แบบ

-รีเลย์กระแสเกินชนิดเวลาผกผันกับกระแส (Inverse time over current relay) คือรีเลย์ที่มีเวลาทำงานเป็นส่วนกลับกับกระแส

-รีเลย์กระแสเกินชนิดทำงานทันที (Instantaneous over current relay) คือรีเลย์ที่ทำงานทันทีทันใดเมื่อมีกระแสไหลผ่านเกินกว่าที่กำหนดที่ตั้งไว้

-รีเลย์แบบดิฟฟินิตไทม์เล็ก คือรีเลย์ที่มีเวลาการทำงานไม่ขึ้นอยู่กับความมากน้อยของกระแสหรือค่าไฟฟ้าอื่นๆ ที่ทำให้เกิดงานขึ้น

-รีเลย์แบบอินเวิร์สดิฟฟินิตมินิมั่มไทม์เล็ก คือรีเลย์ที่ทำงานโดยรวมเอาคุณสมบัติของเวลาผกผันกับกระแส (Inverse time) และแบบดิฟฟินิตไทม์เล็กเข้าด้วยกัน

6. รีเลย์กระแสต่าง (Differential relay) คือรีเลย์ที่ทำงานโดยอาศัยผลต่างของกระแส

7. รีเลย์มีทิศ (Directional relay) คือรีเลย์ที่ทำงานเมื่อมีกระแสไหลผิดทิศทาง มีแบบรีเลย์กำลังมีทิศ (Directional power relay) และรีเลย์กระแสมีทิศ (Directional current relay)

8. รีเลย์ระยะทาง (Distance relay) คือรีเลย์ระยะทางมีแบบต่างๆ ดังนี้

-รีแอกแตนซ์รีเลย์

-อิมพีแดนซ์รีเลย์

-โมห์รีเลย์

-โอห์มรีเลย์

-โพลาไรซ์โมห์รีเลย์

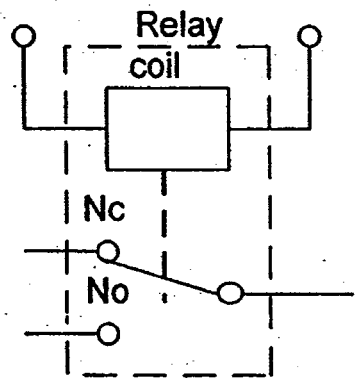
-ออฟเซทโมห์รีเลย์

9. รีเลย์อุณหภูมิ (Temperature relay) คือรีเลย์ที่ทำงานตามอุณหภูมิที่ตั้งไว้

10. รีเลย์ความถี่ (Frequency relay) คือรีเลย์ที่ทำงานเมื่อความถี่ของระบบต่ำกว่าหรือมากกว่าที่ตั้งไว้

11. บูคโฮลซ์รีเลย์ คือรีเลย์ที่ทำงานด้วยก๊าซใช้กับหม้อแปลงที่แช่อยู่ในน้ำมันเมื่อเกิดฟอลต์ ขึ้นภายในหม้อแปลง จะทำให้น้ำมันแตกตัวและเกิดก๊าซขึ้นภายในไปดันหน้าสัมผัสให้รีเลย์ทำงาน

ซึ่งในโครงงานนี้จะใช้รีเลย์ควบคุม เพื่อทำการควบคุมการเปิด-ปิดสวิทช์อุปกรณ์ไฟฟ้า ซึ่งสั่งงานมาจากหน้าเว็บเพจ



รูปที่ 3.3 สัญลักษณ์ของรีเลย์ที่ใช้งาน

จากรูปที่ 3.3 เป็นสัญลักษณ์ของรีเลย์ จากรูปจะประกอบไปด้วยคอยล์ สถานะของรีเลย์ปกติปิดและสถานะของรีเลย์ปกติเปิด

การทำงานของรีเลย์ โดยรีเลย์จะอาศัยหลักการเหนี่ยวนำของแม่เหล็กซึ่งเมื่อจ่ายแรงดันให้กับคอยล์ของรีเลย์ ก็จะทำให้เกิดการสวิทช์จากสถานะปกติปิดก็จะเปิดและเช่นเดียวกันสถานะปกติเปิดก็จะ

ปิด จากสถานะ การทำงานข้างต้นที่ได้กล่าวมานั้นเราก็สามารถทำการควบคุมรีเลย์ให้ทำการเปิดไฟฟ้าหรือปิดไฟฟ้าได้โดยการควบคุมแรงดันไฟฟ้าที่ผ่านคอยล์

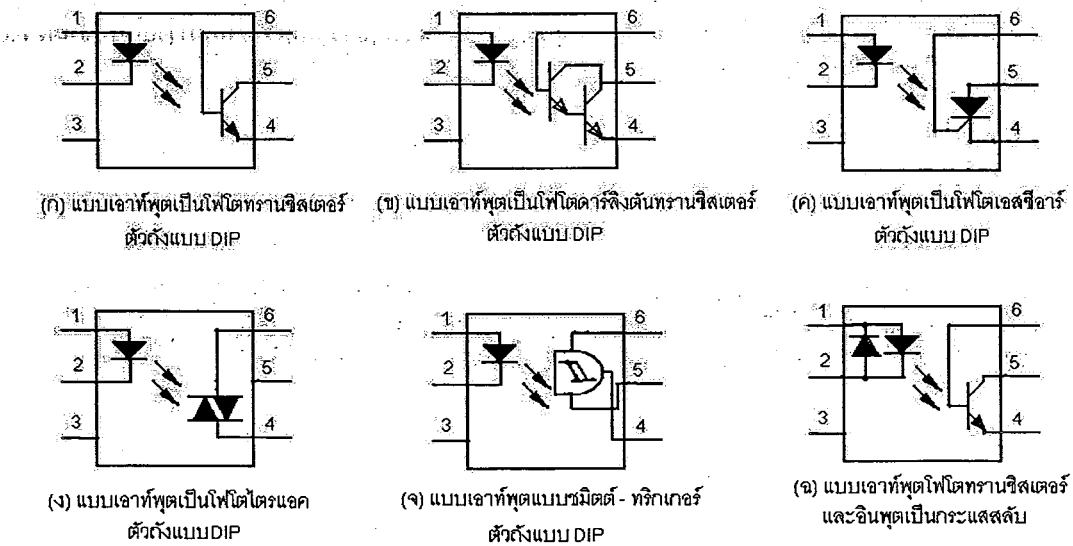
การใช้งานรีเลย์ โดยจะควบคุมการเปิดและปิดหลอดไฟฟ้าจาก RCM3200 โดยสัญญาณควบคุมจะส่งผ่านไปยังอินพุตของไอซี ULN2803 ซึ่งเป็นไอซีขับแบบอินเวอร์เตอร์ที่สามารถจ่ายกระแสได้สูงสุด 500 มิลลิแอมป์ และภายในตัวไอซีจะมีไดโอดป้องกันการยุบตัวของสนามแม่เหล็กที่คอยล์ของรีเลย์ เราสามารถนำเอาสัญญาณเอาต์พุตจากไอซี ULN2803 ไปทำการขับรีเลย์ได้เลย โดยที่อีกขั้วหนึ่งของรีเลย์จะต่อกับแรงดันไฟ 12 โวลต์ เมื่อรีเลย์ทำงานก็จะทำให้วงจรของหลอดไฟฟ้าครบวงจร หลอดไฟฟ้าก็จะติด และเช่นเดียวกัน ถ้าหากรีเลย์ยังไม่ทำงาน วงจรของหลอดไฟฟ้าก็ยังไม่ครบวงจร หลอดไฟฟ้าก็จะดับ

3.4 ตัวเชื่อมต่อผ่านแสง (Opto-coupler)

ตัวเชื่อมต่อผ่านแสง (Opto-coupler) หรือตัวแยกวงจรด้วยแสง (Opto-isolator) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการเชื่อมต่อสัญญาณ หรือข้อมูล ระหว่างกันของวงจรสองวงจร ที่จำเป็นต้องแยกระบบสายกราวด์ (Ground Loop) ของวงจรทั้งสองออกจากกันโดยเด็ดขาด เช่น แยกวงจรระหว่างภาคอินพุต (Input) กับภาคเอาต์พุต (Output) ออกจากกันทางไฟฟ้า แยกวงจรระหว่างภาคควบคุมที่ใช้สัญญาณระบบลอจิก กับภาคกำลังที่ต้องใช้ระบบไฟฟ้ากระแสตรง หรือไฟฟ้ากระแสสลับที่ใช้แรงดันไฟฟ้าและกระแสสูง ๆ เป็นต้น ทั้งนี้เพื่อป้องกันการรบกวนซึ่งกันและกัน

องค์ประกอบพื้นฐาน

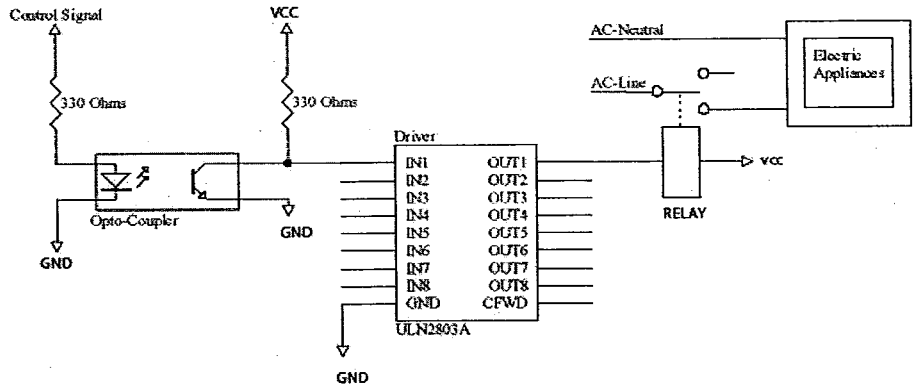
โครงสร้างพื้นฐานของ ตัวเชื่อมต่อผ่านแสง จะประกอบด้วย ตัวกำเนิดแสง (เช่น LED ชนิดต่างๆ) และตัวรับแสง (เช่น โฟโตทรานซิสเตอร์ โฟโตไดโอด หรืออุปกรณ์ไวแสงอื่น ๆ) อุปกรณ์ตัวกำเนิดแสงและตัวรับแสงนี้จะประกอบอยู่ในตัวถังแบบทึบแสงเพื่อป้องกันแสงภายนอกที่อาจมารบกวน



รูปที่ 3.4 ตัวอย่างตัวเชื่อมต่อผ่านแสงแบบต่าง ๆ

3.5 การออกแบบวงจรเปิด-ปิดอุปกรณ์ไฟฟ้า

วงจรนี้ทำหน้าที่ในการรับสัญญาณไฟฟ้าจาก Rabbit RCM3200 ซึ่ง Rabbit RCM3200 จะส่งสัญญาณควบคุม (Control Signal) ผ่านทาง PA0-PA7 ดังรูปที่ 3.5



รูปที่ 3.5 วงจรเปิด-ปิดอุปกรณ์ไฟฟ้า

จากรูปที่ 3.5 เป็นวงจรที่ออกแบบเพื่อรับสัญญาณไฟฟ้า (Control Signal) เพื่อทำให้ Opto-Coupler ทำงาน คือส่งสัญญาณที่เป็นแสงเพื่อไปกระตุ้นให้ Transistor ที่อยู่ภายในทำงาน คือ นำกระแสจากแหล่งจ่ายไฟ VCC ลงสู่ GND ซึ่งแสดงให้ทราบว่า เมื่อ Control Signal เป็น Logic "1" สัญญาณที่ส่งให้กับ Driver ULN2803A จะเป็น Logic "0" และเป็น Logic "1" เมื่อ Control Signal เป็น Logic "0" และเมื่อสัญญาณของ Driver เป็น Logic "1" จะส่งผลให้ Driver ในขา OUT1 เชื่อมต่อกับ GND ทำให้รีเลย์ทำงานและเชื่อมต่อ AC-Line ที่ถูกตัดออกในขณะที่รีเลย์ไม่ทำงาน เป็นผลให้อุปกรณ์ไฟฟ้าไม่ทำงาน

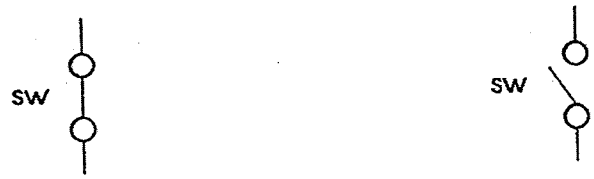
3.6 อุปกรณ์ตรวจจับการเปิด-ปิด

ในส่วนของอุปกรณ์ตรวจจับนี้จะใช้สวิทช์แม่เหล็ก (Magnetic Switch) จากรูปที่ 3.6 รูปร่างของสวิทช์แม่เหล็กที่นำมาใช้เป็นตัวตรวจจับเซ็นเซอร์ รูปที่ 3.6 (ก) คือรูปร่างของสวิทช์แม่เหล็กแบบปกติปิด รูปที่ 3.6 (ข) คือรูปร่างของสวิทช์แม่เหล็กแบบปกติเปิด

(ก) (ข)

รูปที่ 3.6 รูปร่างของสวิทช์แม่เหล็ก

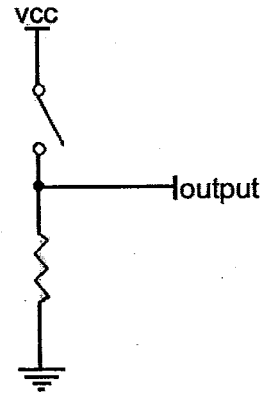
จากรูปที่ 3.7 เป็นโครงสร้างของสวิทช์แม่เหล็กที่นำมาใช้งาน ส่วนรูปที่ 3.7 (ก) เป็นโครงสร้างของสวิทช์แบบปกติปิด หน้าสัมผัสของสวิทช์ต่อกันอยู่ (NC) และรูปที่ 3.7 (ข) เป็นโครงสร้างของสวิทช์แบบปกติเปิด หน้าสัมผัสของสวิทช์จะแยกกัน (NO)



(ก) (ข)

รูปที่ 3.7 โครงสร้างของสวิทช์แม่เหล็ก

การออกแบบวงจรเซ็นเซอร์ จะใช้สวิทช์แม่เหล็กเป็นตัวตรวจจับสิ่งผิดปกติที่เกิดขึ้น โดยสวิทช์จะยึดติดกับขอบประตูและหน้าต่าง

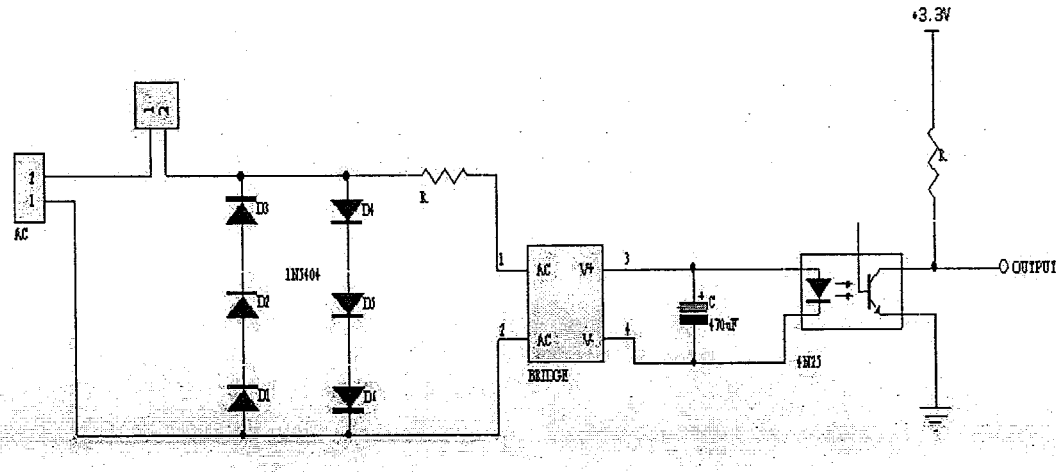


รูปที่ 3.8 วงจรเซ็นเซอร์โดยใช้สวิทช์แม่เหล็ก

3.7 วงจรตรวจสอบสถานะอุปกรณ์ไฟฟ้า

วงจรตรวจสอบสถานะ ดังรูปที่ 3.9 จะทำหน้าที่ตรวจสอบการทำงานของอุปกรณ์ไฟฟ้าแต่ละตัว โดยใช้สัญญาณที่ได้จากวงจรตรวจสอบสถานะเป็นสัญญาณอินพุตให้กับไมโครคอนโทรลเลอร์ โดยถ้าสัญญาณเป็นลอจิก “0” แสดงว่าอุปกรณ์ทำงาน แต่ถ้าสัญญาณเป็นลอจิก “1” แสดงว่าอุปกรณ์ไม่ทำงาน โดยวงจรตรวจสอบสถานะในโครงงานจะกำหนดให้อุปกรณ์หรือโหลดใช้กระแสไม่เกิน 1 แอมป์ โดยใช้ออปโตทรานซิสเตอร์ (Optotransistor) เบอร์ 4N25 ซึ่งเป็นอุปกรณ์สารกึ่งตัวนำของอิเลคทรอนิกส์ตัวหนึ่ง โดยการใช้เป็นสวิทช์เพื่อแยกทางไฟฟ้า ระหว่างอินพุตกับเอาต์พุตซึ่งเชื่อมโยงด้วยแสง (Optocoupling) โดยใช้ไดโอดเบอร์ 1N5404 ซึ่งสามารถรับแรงดันได้ 400 โวลต์ และทนกระแสได้ 3 แอมป์ โดยจะต่อ

ไดโอดบริจด์ เรกติไฟเออร์ เพื่อให้กระแสเป็นกระแสตรง และผ่านตัวเก็บประจุเพื่อกรองกระแสให้เรียบ โดยเอาต์พุตจากออปโตคัปเปอร์จะส่งเข้าไปยังพอร์ตของ Rabbit RCM3200



รูปที่ 3.9 วงจรตรวจสอบสถานะอุปกรณ์ไฟฟ้า

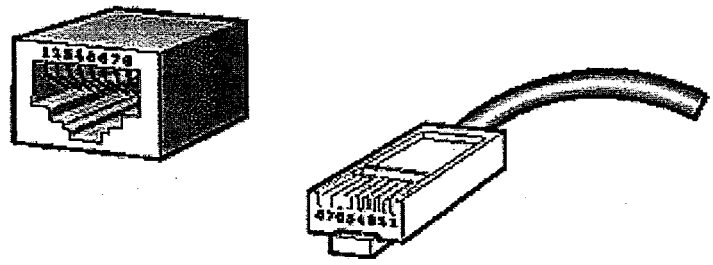
จากโหลด จะมีกระแสไฟฟ้าไหลผ่านมายังไดโอด และมีแรงดันตกคร่อมที่แอโนดและแคโทด ประมาณ 2.1 โวลต์ ซึ่งแรงดันไฟฟ้าที่ได้จะเป็นเสมือนแหล่งจ่ายแรงดันให้เป็นอินพุตของออปโตทรานซิสเตอร์ ส่วน R จะทำหน้าที่กำจัดกระแสที่จะไหลเข้าไปยังออปโตทรานซิสเตอร์ ส่วนทางด้านเอาต์พุตจะต่อ R เพื่อให้กระแสไหลเพียงเล็กน้อยเมื่อออปโตทรานซิสเตอร์ทำงาน โดยนำแรงดันตกคร่อมระหว่างขาคอลเล็กเตอร์กับกราวนด์ป้อนเข้าพอร์ตที่ใช้เช็คสถานะของ RCM3200

3.8 สาย UTP (Unshielded Twisted Pair Cabling)

สาย UTP (Unshielded Twisted Pair) เป็นสายสัญญาณที่ผลิตตามมาตรฐาน EIA/TIA 568 ซึ่งการใช้สายนี้ความยาวต้องไม่เกิน 100 เมตร

ลักษณะของสายที่ใช้กับ RJ-45 เป็นแบบ 8 เส้น โดยตีเกลียวกัน 4 คู่ โดยไม่มีการชิลด์เรียกว่า UTP เป็นมาตรฐานที่ใช้กันในปัจจุบันคือสายแบบ CAT-5

มาตรฐาน EIA/TIA 568 เป็นมาตรฐานที่กำหนดขึ้นโดยความร่วมมือของ 3 องค์กร ได้แก่ สำนักงานมาตรฐานแห่งสหรัฐอเมริกา (American National Standards Institute : ANSI), สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) และ สมาคมอุตสาหกรรมโทรคมนาคม (Telecommunications Industry Association : TIA) โดยใช้ชื่อมาตรฐานว่า "EIA/TIA 568"



รูปที่ 3.10 ลักษณะของสาย UTP

ตารางที่ 3.1 หน้าที่ของสาย UTP

การกำหนดหน้าที่ของสายย่อยภายในสายแบบ UTP	
หมายเลขสาย	หน้าที่ของสายสัญญาณ
1	เอาต์พุต Transmit Data +
2	เอาต์พุต Transmit Data -
3	อินพุต Receive Data +
6	อินพุต Receive Data -
4, 5, 7, 8	Reserved for other use

การเข้าหัวสายกับขั้วต่อ มี 2 แบบ คือ

1.การต่อเชื่อมระหว่างคอมพิวเตอร์ กับ RCM 3200 เป็นการเชื่อมต่อแบบ Peer to Peer 2 เครื่องไม่ต้อง HUB การต่อสายต้องมีการสลับสายนำสัญญาณทั้งสองด้านดังตารางที่ 3.2

ตารางที่ 3.2 การต่อเชื่อมระหว่างคอมพิวเตอร์ กับ RCM 3200

Computer	สายที่	RCM 3200
ขาวส้ม	1	ขาวเขียว
ส้ม	2	เขียว
ขาวเขียว	3	ขาวส้ม
น้ำเงิน	4	น้ำเงิน
ขาวน้ำเงิน	5	ขาวน้ำเงิน
เขียว	6	ส้ม
ขาวน้ำตาล	7	ขาวน้ำตาล
น้ำตาล	8	น้ำตาล

2.การต่อเชื่อมระหว่าง HUB กับ RCM 3200 จะมีการเข้าสายกับขั้ว RJ-45 จะเหมือนกันทั้งสองข้าง

ตารางที่ 3.3 การต่อเชื่อมระหว่าง HUB กับ RCM 3200

HUB	สายที่	RCM 3200
ขาวส้ม	1	ขาวส้ม
ส้ม	2	ส้ม
ขาวเขียว	3	ขาวเขียว
น้ำเงิน	4	น้ำเงิน
ขาวน้ำเงิน	5	ขาวน้ำเงิน
เขียว	6	เขียว
ขาวน้ำตาล	7	ขาวน้ำตาล
น้ำตาล	8	น้ำตาล

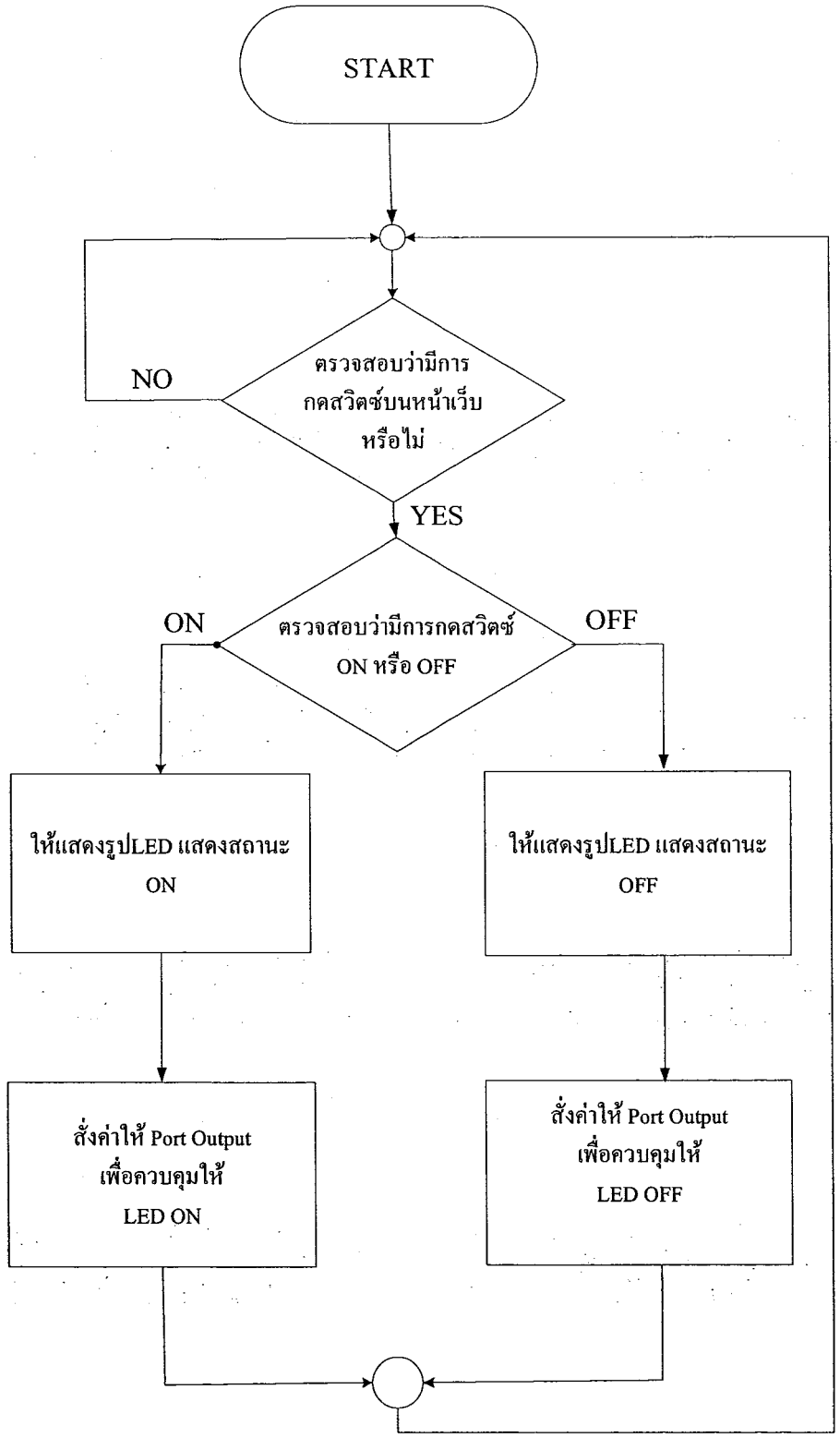
ปัญหาของสาย UTP ส่วนใหญ่เกิดขึ้นจากสาเหตุเหล่านี้ พอจะแยกสาเหตุออกได้ดังนี้

1. การเข้าหัวสายไม่ดี ทำให้เกิดปัญหา Crosstalk ทำให้เกิดปัญหา เชื่อมต่อได้บ้างไม่ได้บ้าง ต้องคอยขยับสาย รวมทั้งเกิดปัญหา Runt Frame อย่างมากมายในเครือข่าย (Runt Frame เป็น เฟรมข้อมูลที่มีขนาดเล็กผิดปกติ มีสาเหตุมาจากการเข้าหัว RJ-45 ไม่เรียบร้อย รวมทั้งปัญหาของ LAN Card
2. ปัญหาสายขาดใน ทำให้เชื่อมต่อกันไม่ได้
3. ปัญหาการสลับสี หรือสลับคู่สายสัญญาณไม่เป็นไปตามมาตรฐาน มักทำให้เกิดอาการ หลายประการ เช่น ไม่สามารถใช้ได้กับ Hub บางรุ่น ที่มีความเข้มงวดในเรื่องของ สัญญาณ ไม่สามารถใช้งานได้ รวมทั้ง เครื่อง hang หรือเครือข่ายล่ม ทันทีที่คอมพิวเตอร์ที่มีปัญหาสายสัญญาณนี้ มีการทำงานเกิดขึ้นบนเครือข่าย
4. ปัญหาการนำสาย Crosswire ไปใช้เชื่อมต่อระหว่าง คอมพิวเตอร์กับ Hub ทำให้ไม่สามารถเชื่อมต่อได้ เช่นเดียวกับการเอา สายธรรมดา (Straight-Through) ไปเชื่อมต่อกันระหว่าง Hub กับ Hub ด้วยกัน ซึ่งทำไม่ได้อยู่แล้ว
5. การใช้สาย Console มาเชื่อมต่อกันระหว่างคอมพิวเตอร์กับ Hub หรือ Hub กับ Hub ด้วยกัน ก็ไม่สามารถทำได้เช่นกัน

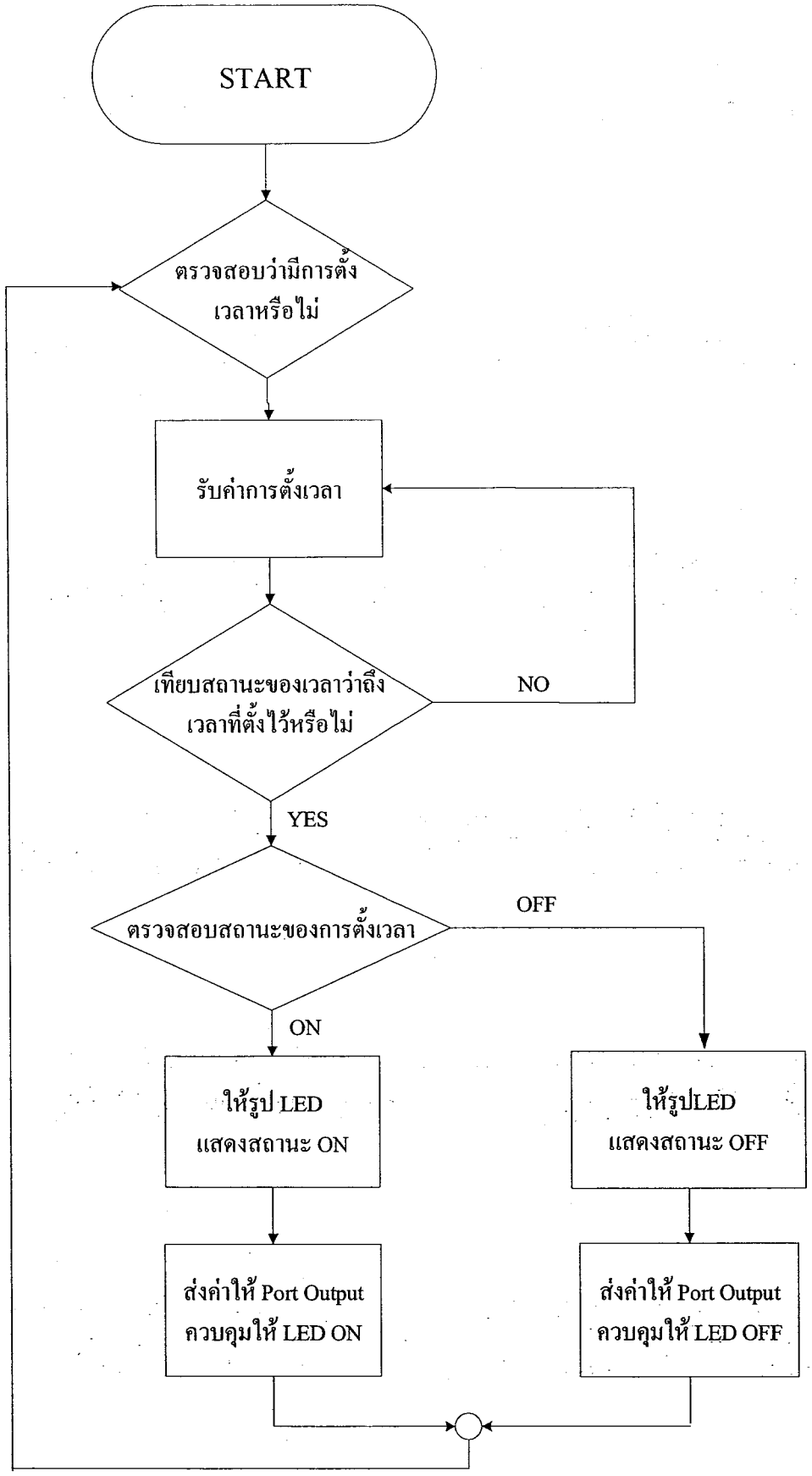
เครื่องมือที่ใช้ตรวจสอบสาย UTP

นอกจาก Multimeter ที่ใช้วัดความต่อเนื่องของสายแล้ว ยังมีเครื่องมือขนาดพกพา ที่นอกจากจะช่วยวัดความต่อเนื่องของสายแล้ว ยังใช้วัดการเข้าหัว RJ-45 ว่า ถูกต้องหรือไม่ รวมทั้งสามารถวัดค่าความเสื่อมถอย และ อัตราของ Crosstalk ได้อีกด้วย

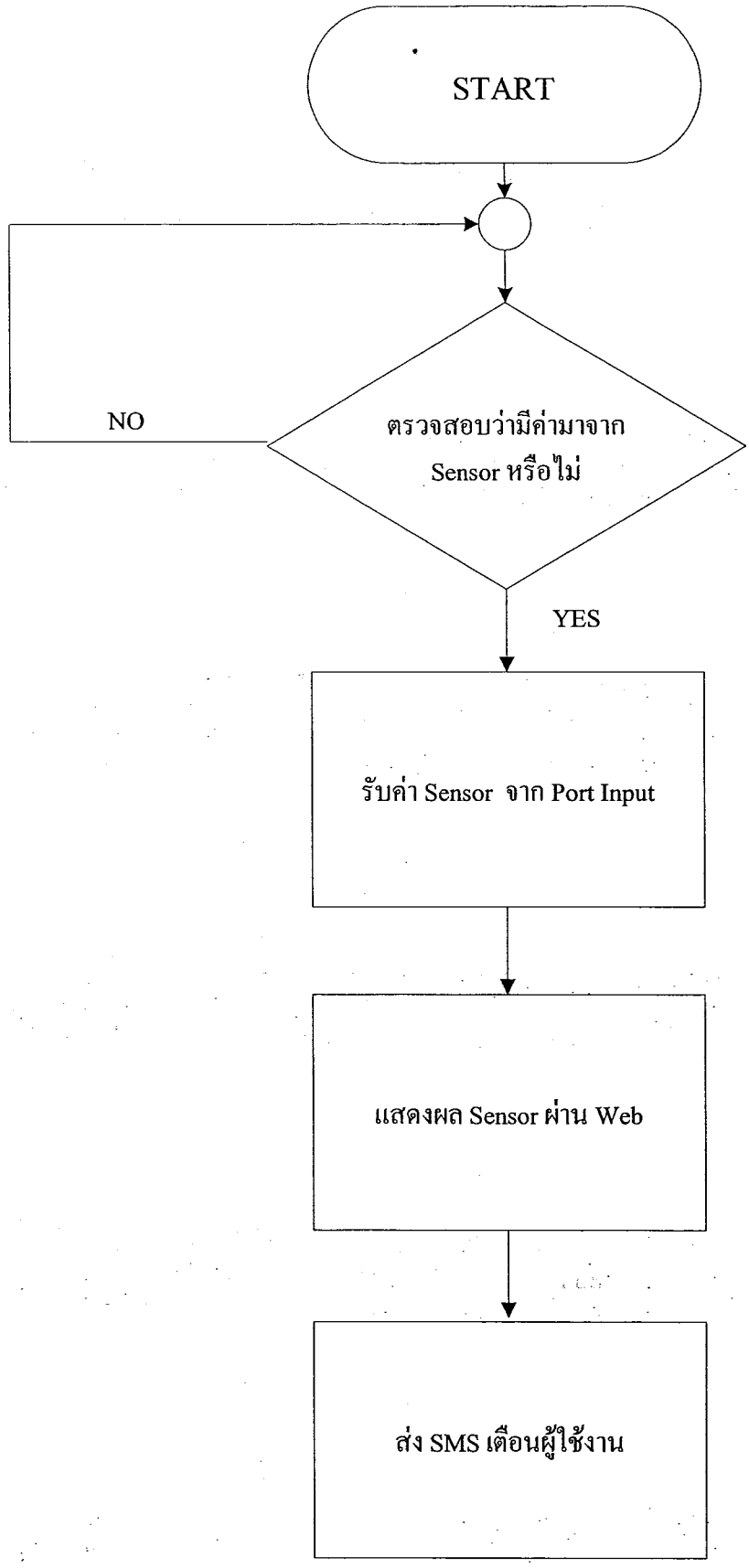
3.9 การออกแบบโปรแกรม



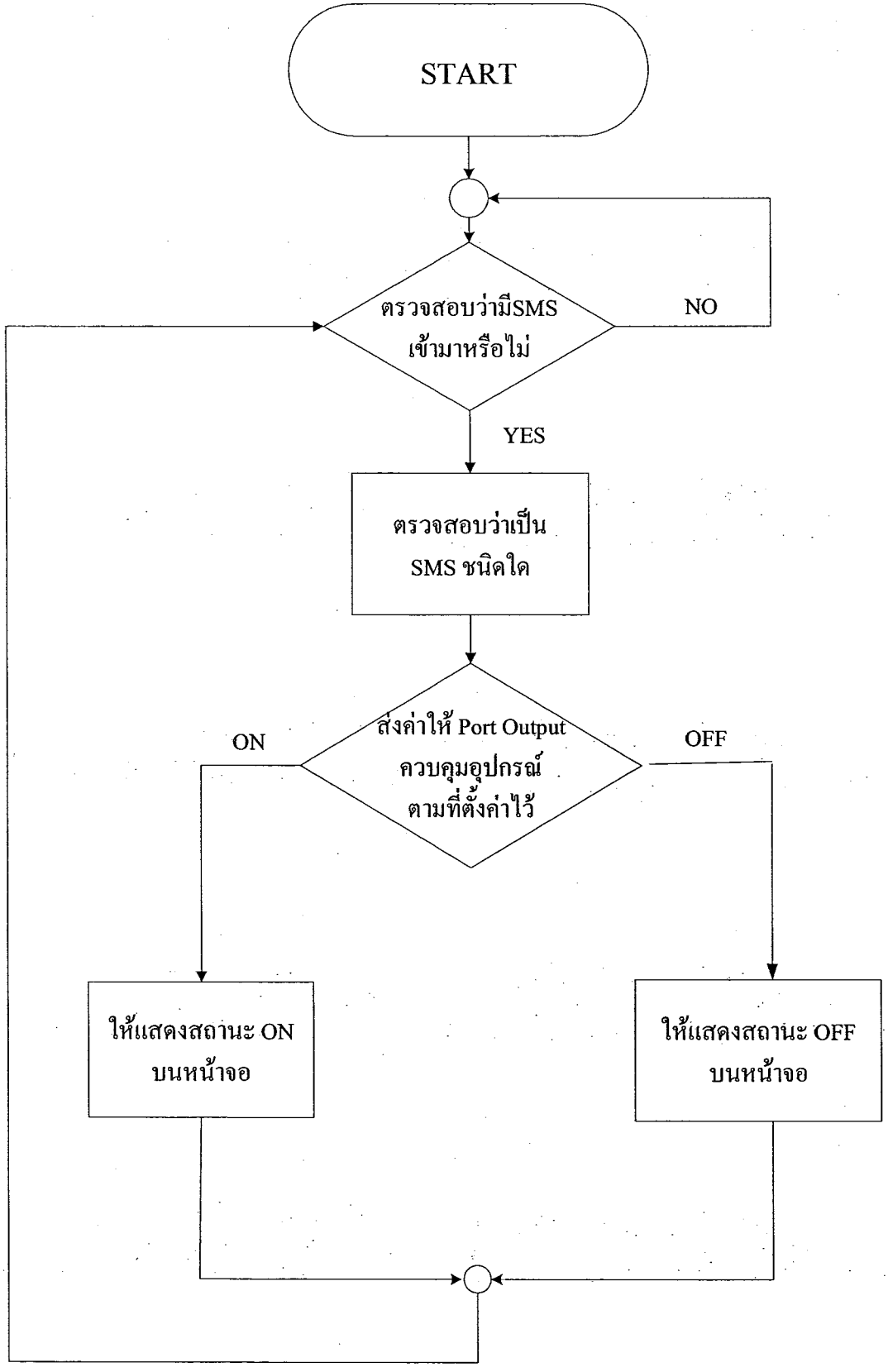
รูปที่ 3.11 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมเมื่อกดสวิตช์จากหน้าเว็บ
เพื่อควบคุมหลอดไฟแต่ละหลอด



รูปที่ 3.12 โฟลว์ชาร์ตแสดงการทำงานของอุปกรณ์ตามคำสั่งการควบคุมอัตโนมัติ



รูปที่ 3.13 โฟลว์ชาร์ตแสดงการทำงานของ Sensor



รูปที่ 3.14 โฟลว์ชาร์ตแสดงการทำงานเมื่อมี SMS เข้ามาควบคุม

บทที่ 4

ผลการทดลอง และวิเคราะห์ผลการทดลอง

4.1 อุปกรณ์ที่ใช้ในการทดลอง

1. Rabbit Core Module RCM3200

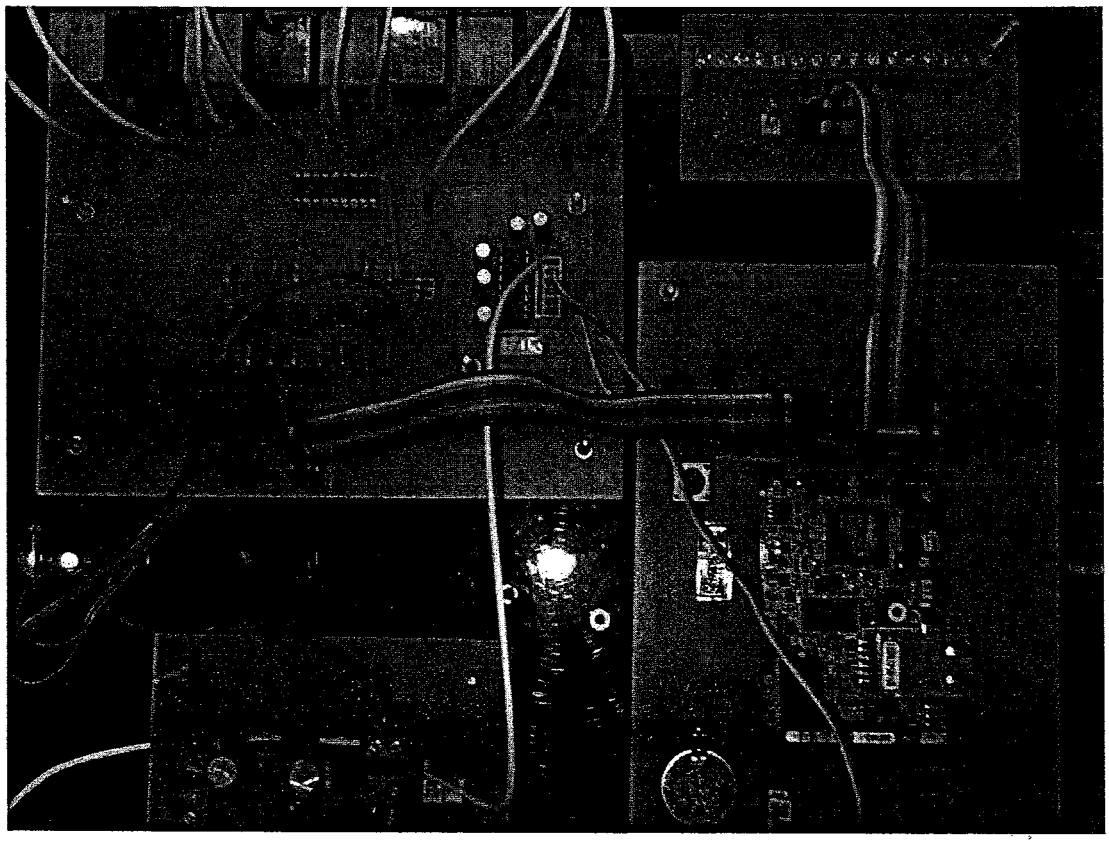
2.วงจรแหล่งจ่ายไฟ

3.วงจรสวิตช์

4.วงจร Magnetic Switch

5.วงจรตรวจสอบสถานะอุปกรณ์ไฟฟ้า

5.บอร์ด ET-GSM SIM300CZ V1.0

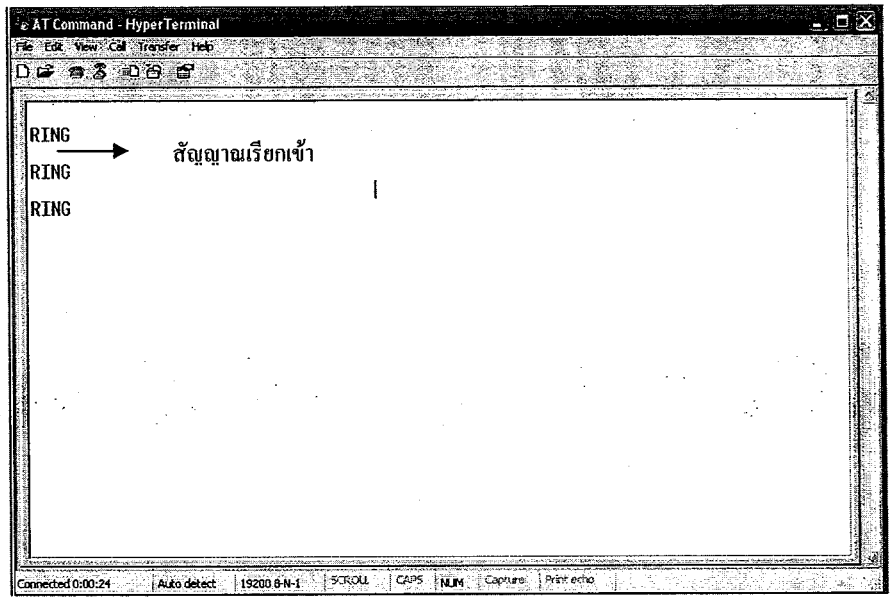


รูปที่ 4.1 รูปวงจรรวม

4.2 ส่วนตรวจจับสัญญาณต่างๆที่เข้ามายังโมดูลโทรศัพท์มือถือ

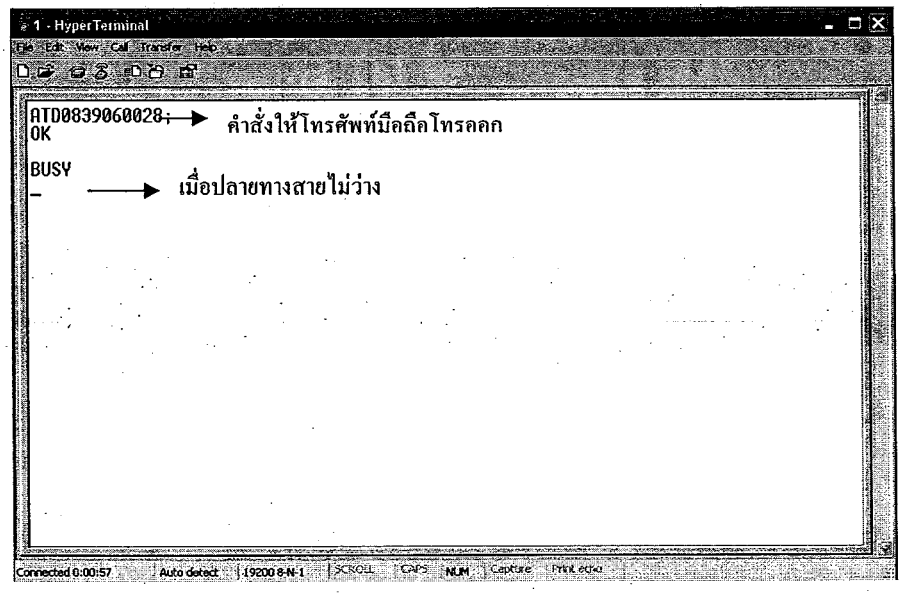
ในส่วนนี้จะตรวจจับสัญญาณจากขา Rx ของโมดูลโทรศัพท์มือถือ ET-GSM SIM300CZ โดยโมดูลโทรศัพท์มือถือจะส่งข้อมูลแบบอนุกรมออกมาด้วยอัตราเร็ว 19,200 บิต/วินาที ดังนั้นจึงต้องทำการตั้งค่าอัตราการรับส่งข้อมูลของ RABBIT RCM3200 ให้เป็น 19,200 บิต/วินาที ด้วยเช่นกัน

โดยในส่วนของการตรวจจับสัญญาณเรียกเข้านั้น เมื่อมีสายเรียกเข้าโมดูลโทรศัพท์จะส่งคำว่า “RING” ออกมาทางขา Rx ของโมดูลโทรศัพท์ โดยแสดงดังรูปที่ 4.2 เป็นการใช้ Hyper Terminal ในการทดสอบการรับสัญญาณจากโมดูลโทรศัพท์เมื่อมีสายเรียกเข้า



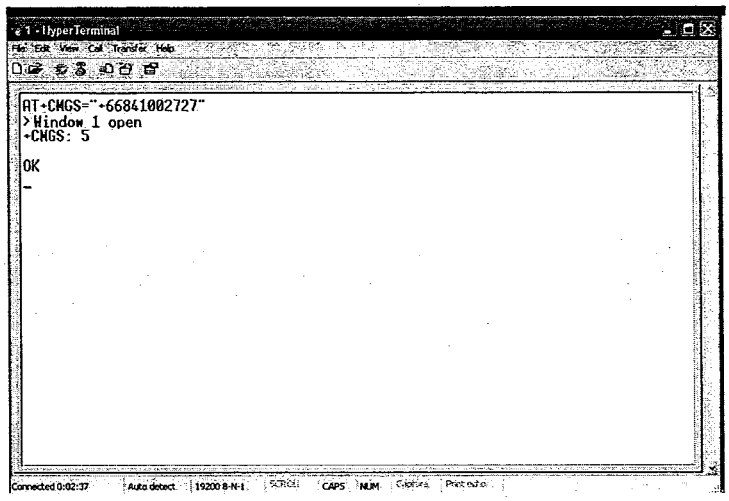
รูปที่ 4.2 แสดงการใช้ Hyper Terminal ในการรับสัญญาณเรียกเข้าจากโทรศัพท์

ในการตรวจสอบสัญญาณเมื่อปลายทางสายไม่ว่าง หรือ Busy Tone จะทำในลักษณะเดียวกับการตรวจจับสัญญาณเรียกเข้า โดยสัญญาณที่ไมโครคอนโทรลเลอร์จะได้จากโมดูลโทรศัพท์เป็นคำว่า “BUSY” ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 แสดงการใช้ Hyper Terminal ในการรับสัญญาณ BUSY เมื่อปลายทางสายไม่ว่าง

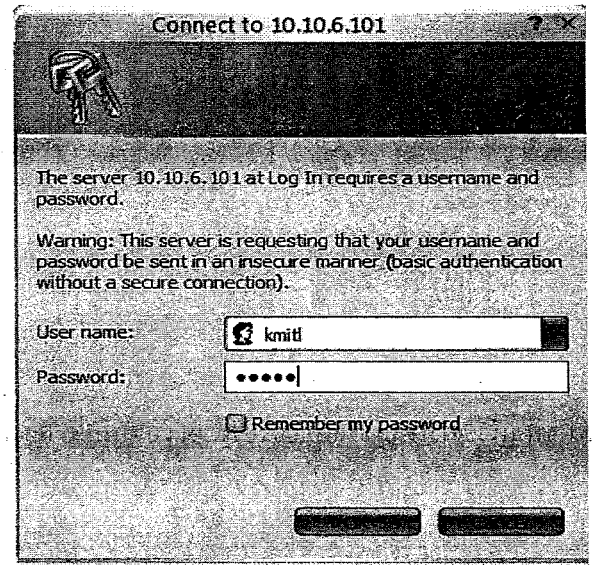
ในการส่งข้อความสั้น (SMS) จะสามารถตรวจสอบการส่งโดยใช้ Hyper Terminal ซึ่งการส่งนั้นจะแสดงดังรูปที่ 4.4



รูปที่ 4.4 แสดงการใช้ Hyper Terminal ในส่งข้อความจากโมดูลโทรศัพท์ไปยังเลขหมายที่ต้องการ

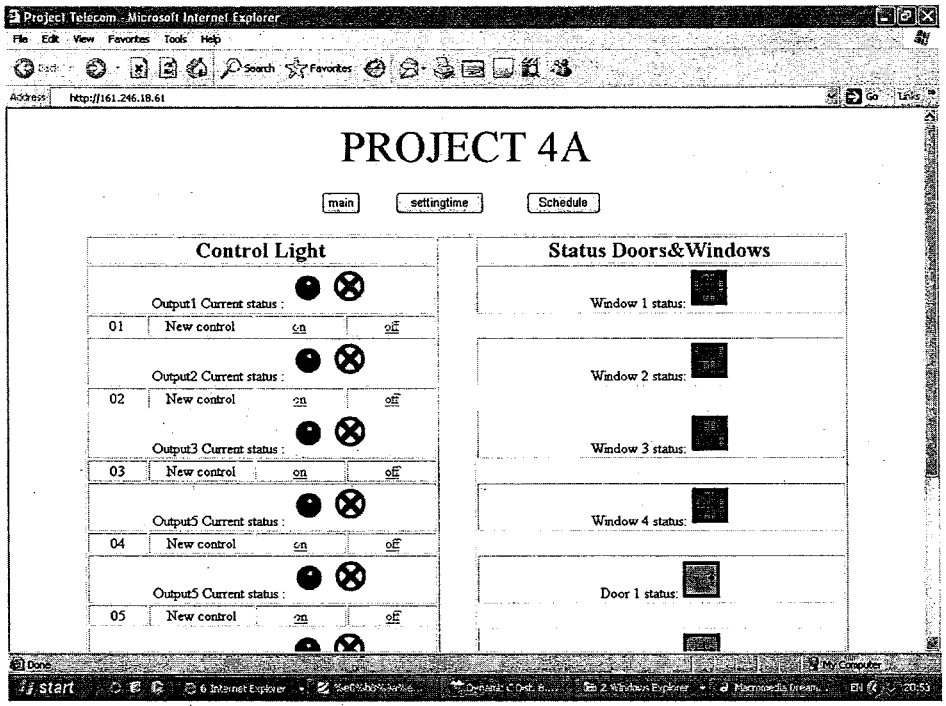
4.3 การทดลองแสดงผลผ่านทางหน้าเว็บเพจ

เมื่อทำการป้อนค่า IP ที่เป็นของ RCM 3200 เพื่อเข้าสู่หน้าจอเว็บเพจ โดยจะต้องใส่ USER NAME และ PASSWORD ให้ถูกต้องจึงจะเข้าไปสู่หน้าจอ ที่ใช้ในการสั่งงาน เพื่อป้องกันผู้ที่ไม่ได้รับอนุญาตเข้าไปใช้งาน ซึ่ง USER NAME ที่ตั้งไว้คือ “kmitl” และ PASSWORD คือ “telec” เมื่อใส่ถูกต้องโปรแกรมจึงจะยอมให้ผ่านเข้าไปหน้าเว็บเพจ แต่ถ้าใส่ไม่ถูกก็ไม่สามารถใช้งานได้ โปรแกรมจะนำค่าที่ใส่ส่งไปยัง RCM 3200 ซึ่งเป็น server และนำค่าไปใช้งานกับค่าที่ตั้งไว้ว่าตรงกันหรือไม่



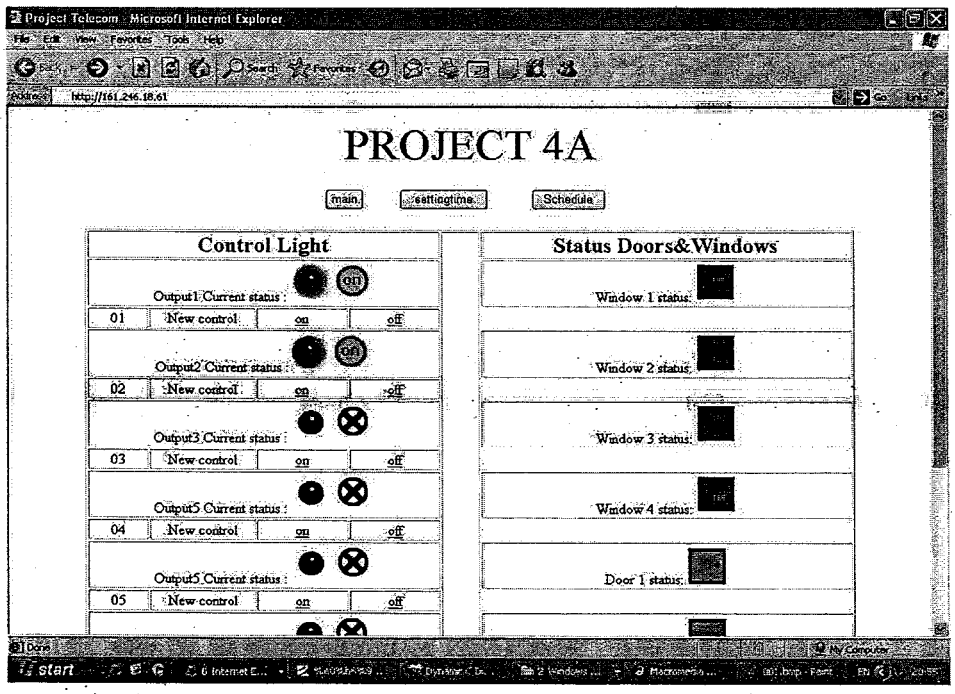
รูปที่ 4.5 แสดงหน้า Web Browser เพื่อการเข้าสู่ระบบ

เมื่อป้อน User name และ Password ถูกต้องแล้ว จะปรากฏหน้าจอดังรูปที่ 4.6



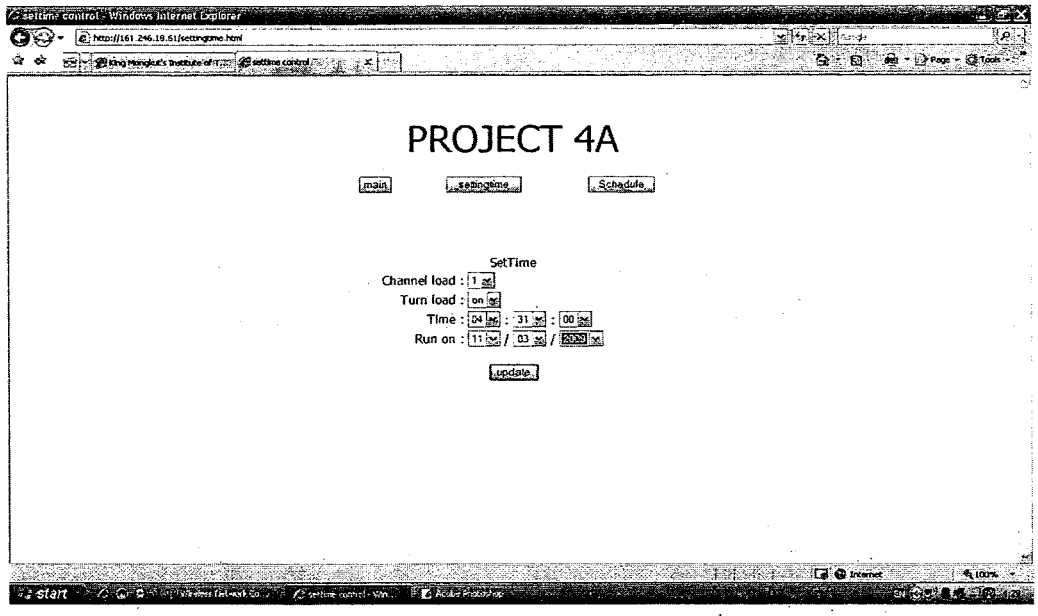
รูปที่ 4.6 หน้าจอที่ปรากฏเมื่อเข้าสู่ระบบแล้ว

ส่วนที่สำคัญในการควบคุม คือส่วนที่ใช้สำหรับสั่งเปิด-ปิดอุปกรณ์ไฟฟ้า และแสดงค่าสถานะจากการถูกเปิด-ปิดประตูหรือหน้าต่าง ซึ่งจะสามารถทำงานบนเว็บบราวเซอร์ได้ ดังรูปที่ 4.7



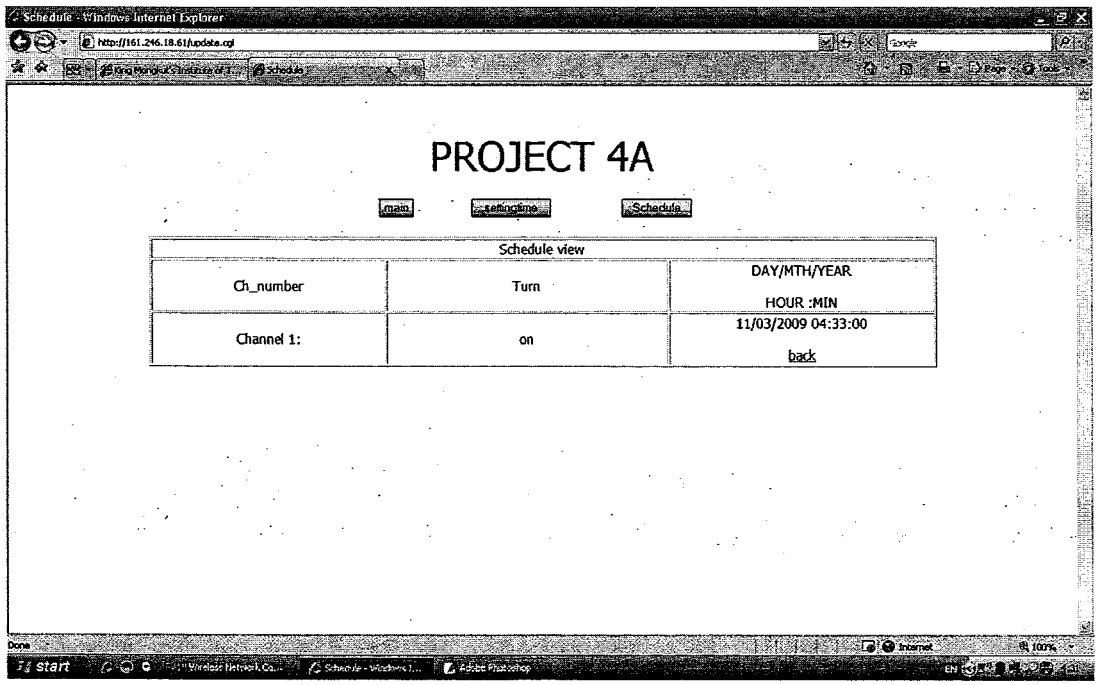
รูปที่ 4.7 ภาพโปรแกรมการควบคุมด้วยตนเองและหน้าจอแสดงผล

ทดสอบตั้งเวลาเปิด-ปิดอัตโนมัติ โดยเลือกที่เมนู Setting time ซึ่งจะปรากฏหน้าจอดังรูปที่ 4.8



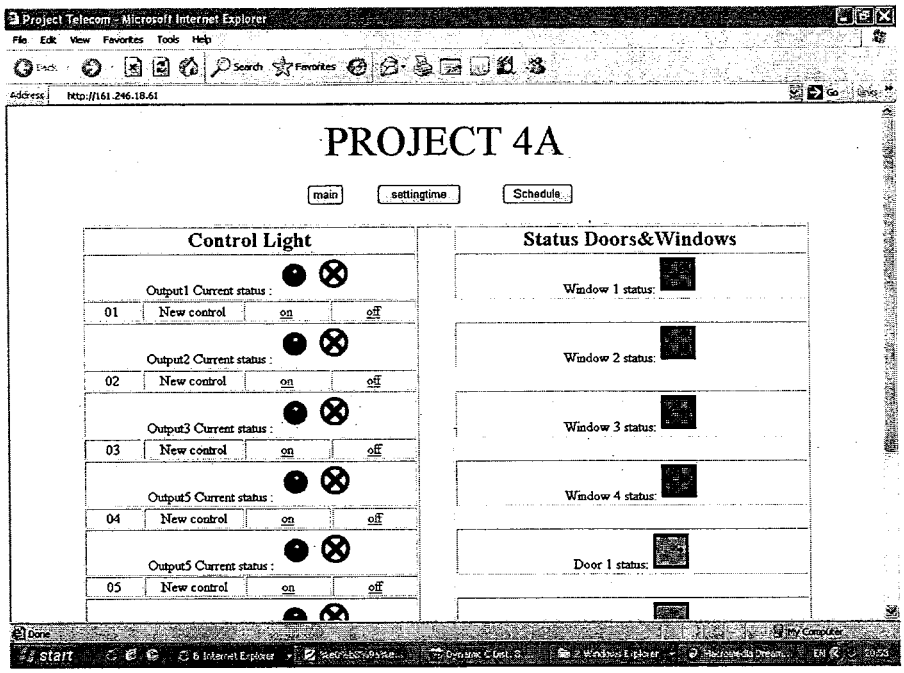
รูปที่ 4.8 หน้าจอสำหรับผู้ใช้ตั้งวัน-เวลาสำหรับเปิด-ปิดการทำงาน

และจากการทดลองดังรูปที่ 4.8 จะเห็นว่าเราตั้งเวลาเพื่อเปิดหลอดไฟดวงที่ 1 โดยจะเปิดไฟในวันที่ 11/03/2009 ในเวลา 04:31:00 โดยจะสามารถดูตารางการทำงานที่ได้ดังรูปที่ 4.9



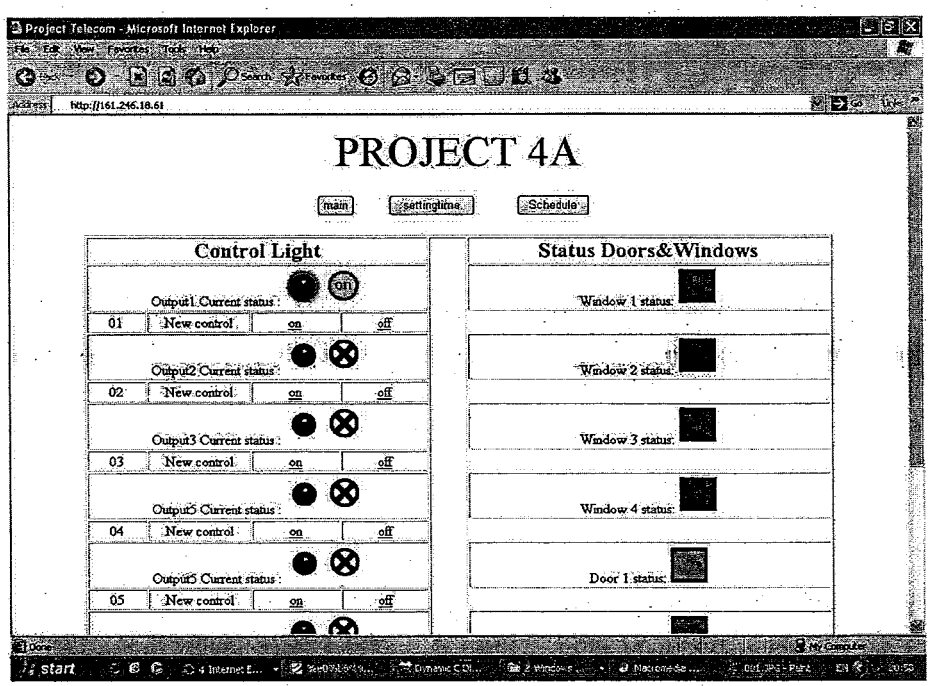
รูปที่ 4.9 แสดง Schedule ที่กำหนดไว้

เมื่อยังไม่ถึงเวลาที่กำหนด หน้าเว็บเพจก็จะปกติ ไฟจะยังไม่ติด



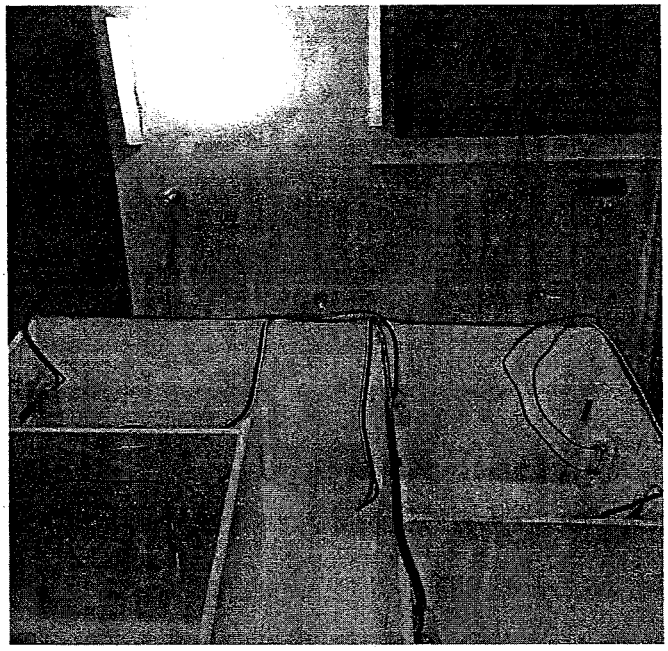
รูปที่ 4.10 หน้าจอแสดงสถานะเมื่อยังไม่ถึงเวลาที่ตั้งไว้

เมื่อถึงเวลาที่ตั้งไว้หน้าจอแสดงสถานะ อุปกรณ์จะแสดงสถานะของหลอดไฟดวงที่ 1 ว่าเปิด ดังรูปที่ 4.11



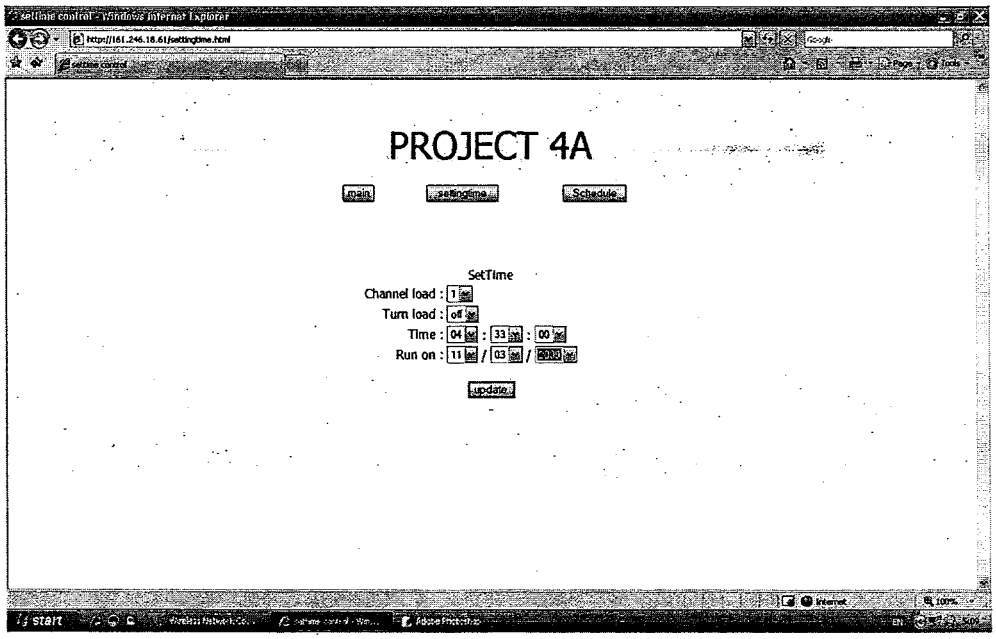
รูปที่4.11 หน้าจอแสดงสถานะเมื่อหลอดไฟดวงที่1 เปิด

เมื่อถึงวันที่ 11/03/2009 เวลา04:31:00 ตาม Schedule ระบบจะทำงาน ทำให้หลอดไฟดวงที่ 1 ติด ดังรูปที่ 4.12



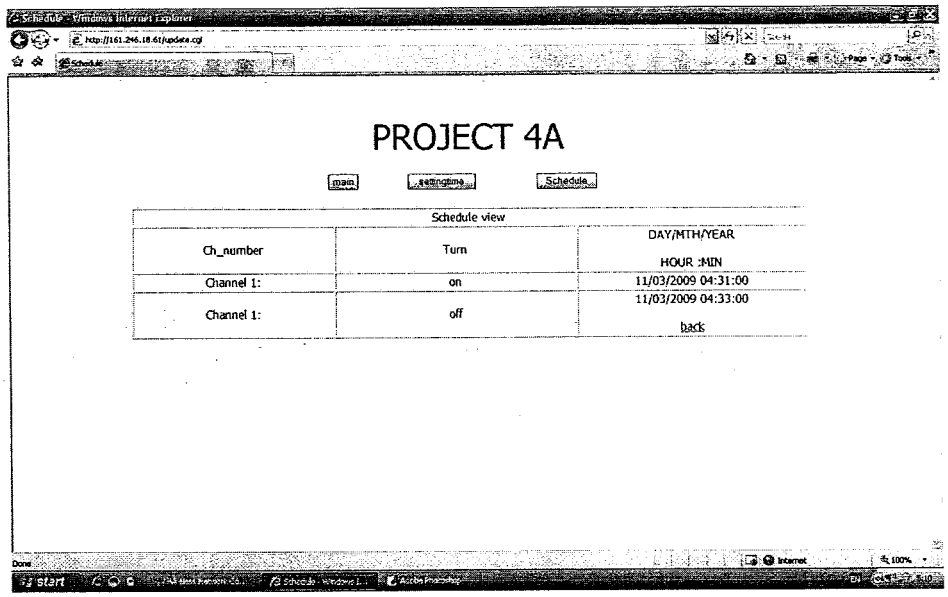
รูปที่ 4.12 รูปหลอดไฟดวงที่ 1ติดเมื่อถึงเวลาที่ตั้งไว้

ทดสอบการตั้งเวลาปิดอุปกรณ์ โดยตั้งเวลาเพื่อปิดหลอดไฟดวงที่ 1 ในวันที่ 11/03/09 เวลา 04:33:00 ดังรูปที่ 4.13



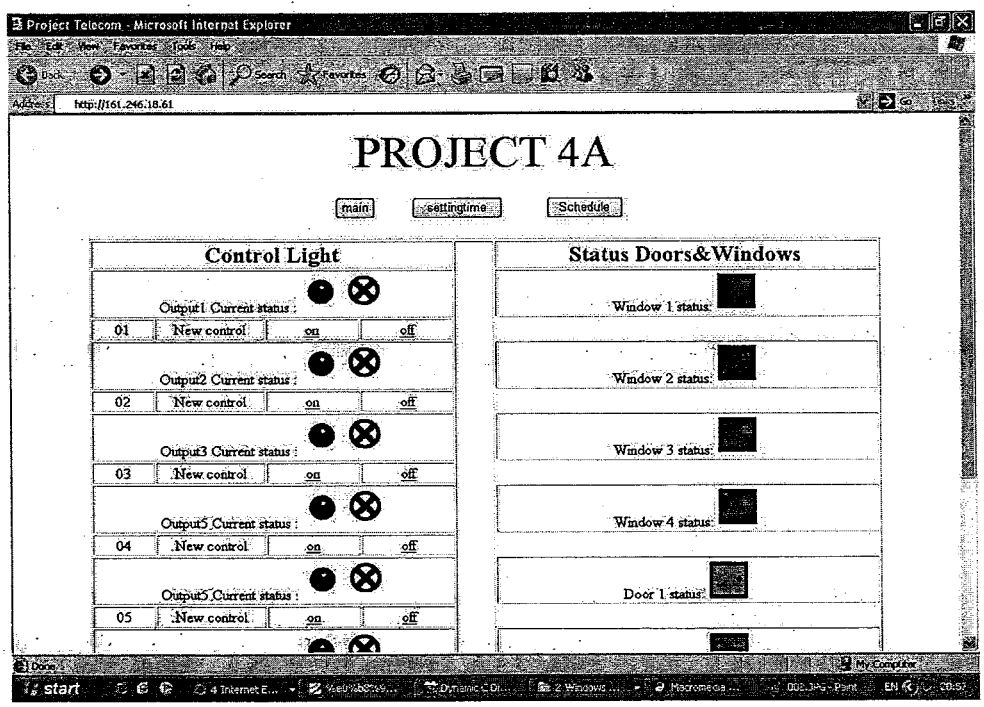
รูปที่ 4.13 หน้าจอแสดงการตั้งเวลาปิดอุปกรณ์

จากการทดลองดังรูปที่ 4.12 จะเห็นว่าเราตั้งเวลาเพื่อปิดหลอดไฟดวงที่ 1 โดยจะปิดไฟในวันที่ 11/03/2009 ในเวลา 04:33:00 โดยจะสามารถดูตารางการทำงานที่ได้ดังรูปที่ 4.14



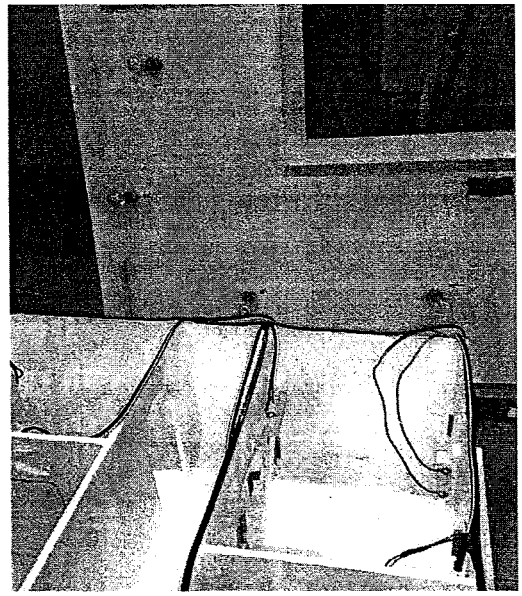
รูปที่ 4.14 แสดง Schedule ที่กำหนดไว้

เมื่อถึงเวลาที่ตั้งไว้หน้าจอแสดงสถานะอุปกรณ์ จะแสดงสถานะของหลอดไฟดวงที่ 1 ว่าปิด ดังรูปที่ 4.15



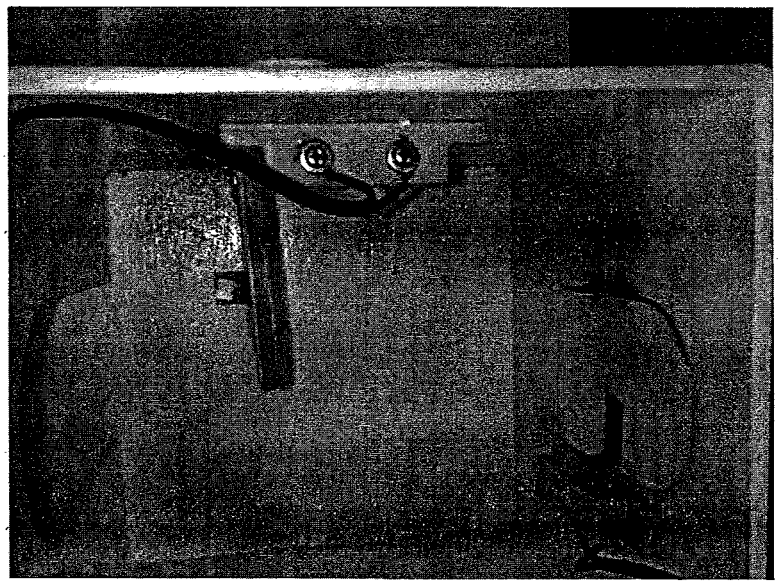
รูปที่ 4.15 หน้าจอแสดงสถานะเมื่อหลอดไฟดวงที่1 ปิด

เมื่อถึงวันที่ 11/03/2009 เวลา04:33:00 ตาม Schedule ระบบจะทำงาน ทำให้หลอดไฟดวงที่ 1 ปิด ดังรูปที่ 4.16



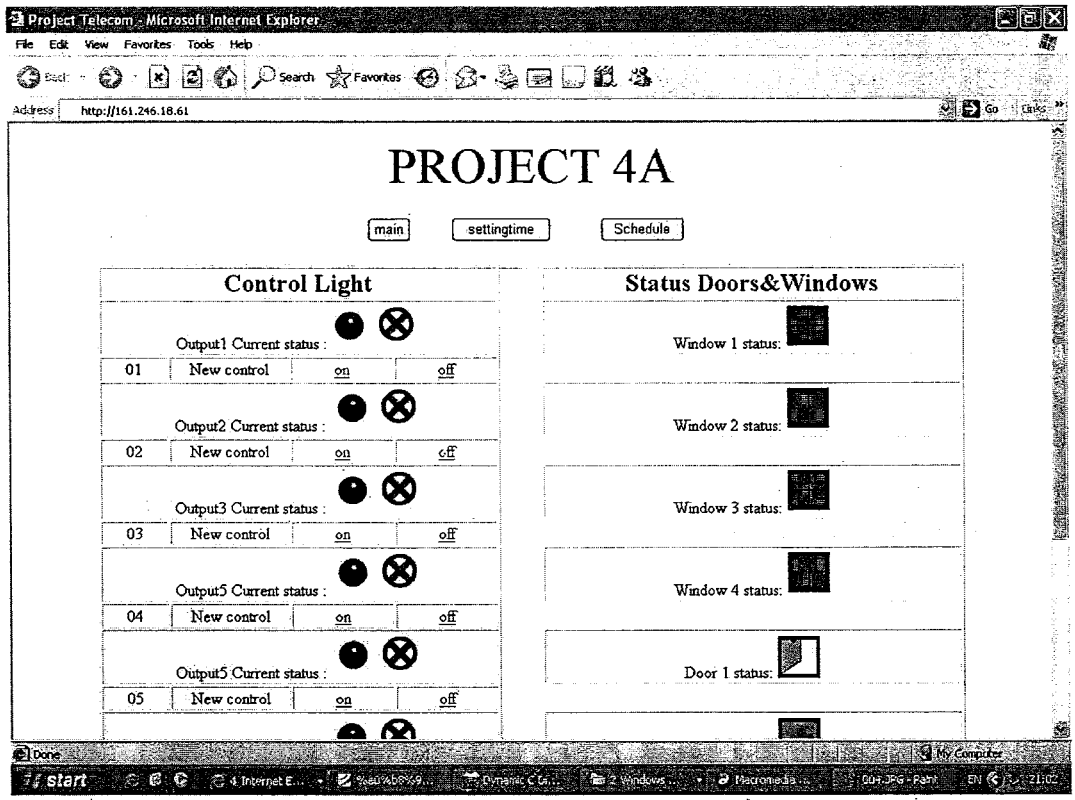
รูปที่ 4.16 รูปหลอดไฟดวงที่ 1ปิดเมื่อถึงเวลาที่กำหนดไว้

ทดสอบการส่งข้อความเตือน เมื่อมีการเปิดประตูที่ 1ดังรูปที่ 4.17

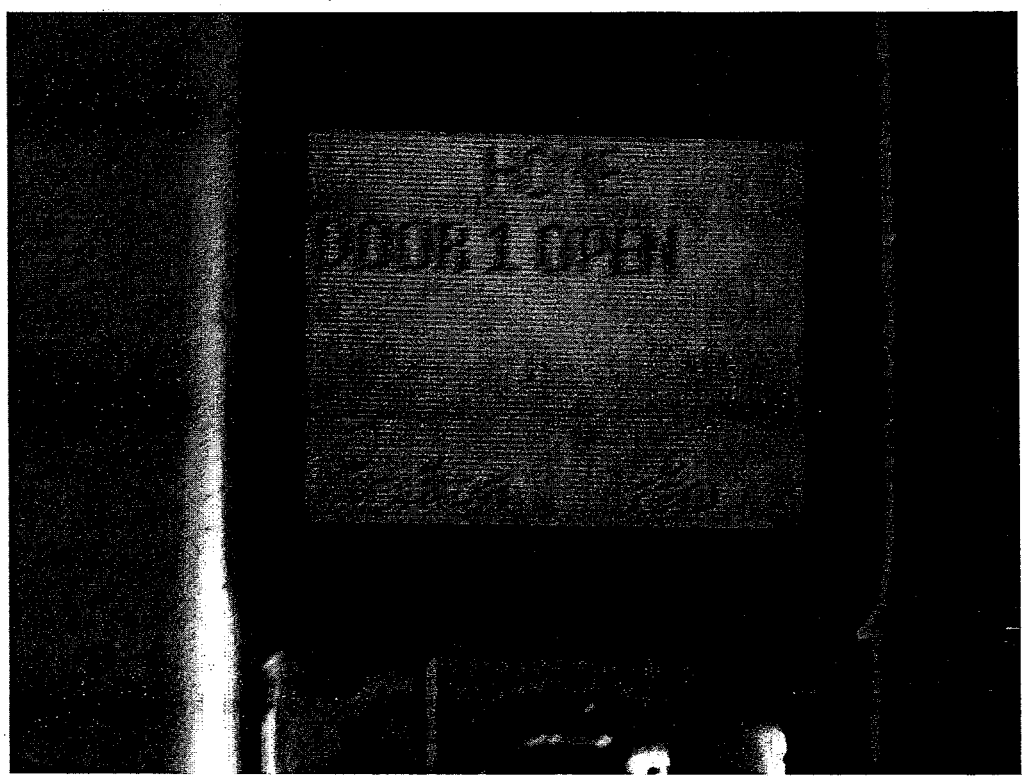


รูปที่ 4.17 รูปแสดงการเปิดประตูที่ 1

เมื่อประตูถูกเปิด หน้าจอแสดงสถานะจะแสดงว่าประตูที่ 1 ถูกเปิดดังรูปที่ 4.18 และจะมีข้อความส่งมาเตือนดังรูปที่ 4.19

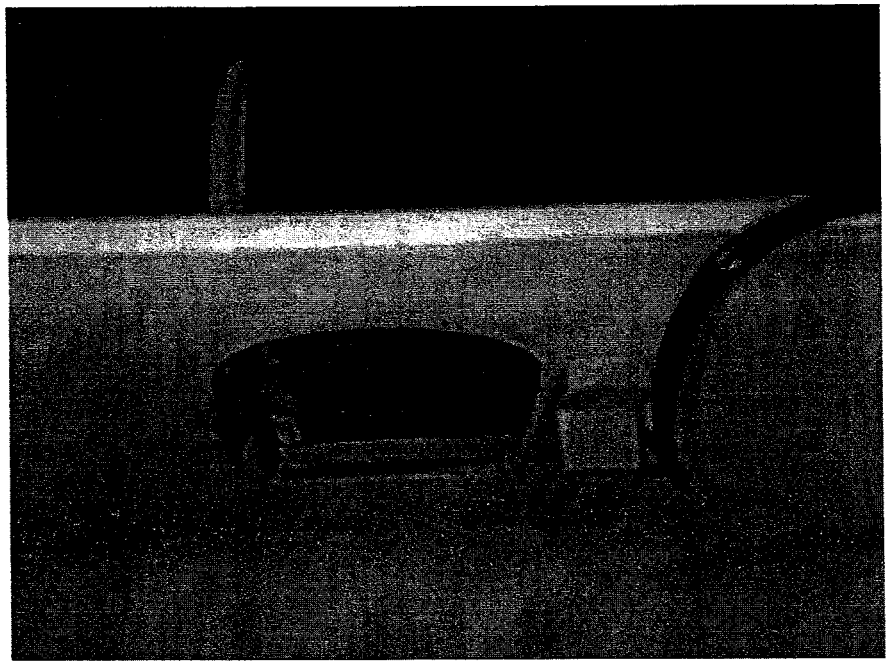


รูปที่ 4.18 หน้าจอแสดงสถานะเมื่อประตูที่ 1 เปิด



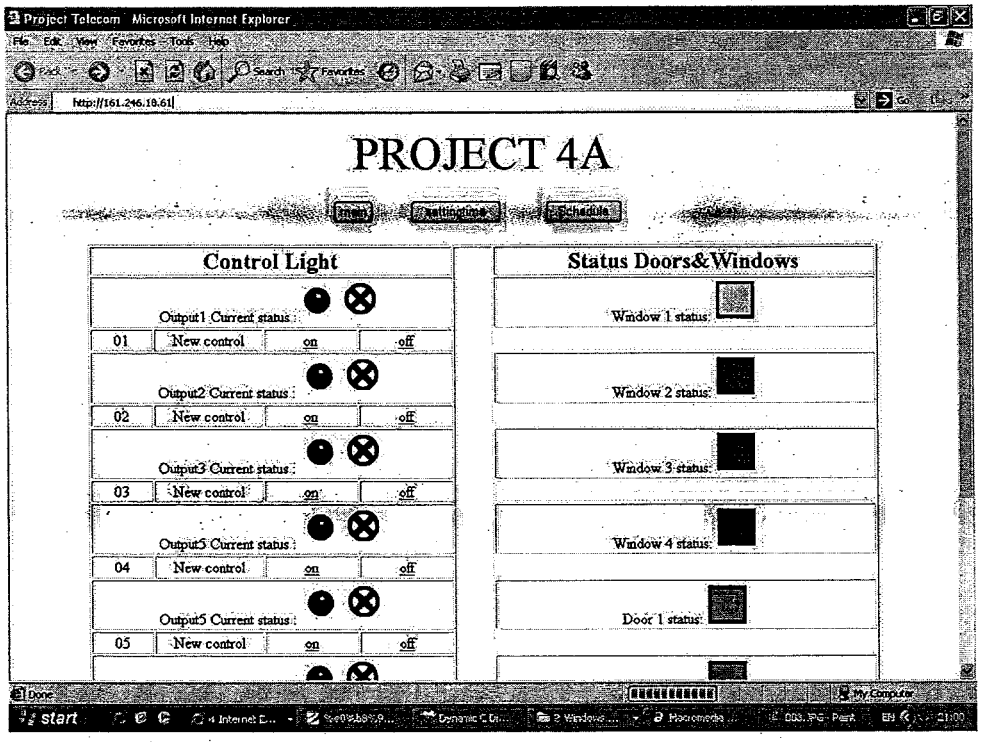
รูปที่ 4.19 ข้อความเตือน เมื่อประตูที่ 1 ถูกเปิดออก

ทดสอบการส่งข้อความเตือนเมื่อหน้าต่างที่ 1 ถูกเปิด ดังรูปที่ 4.20



รูปที่ 4.20 แสดงการเปิดหน้าต่างที่ 1

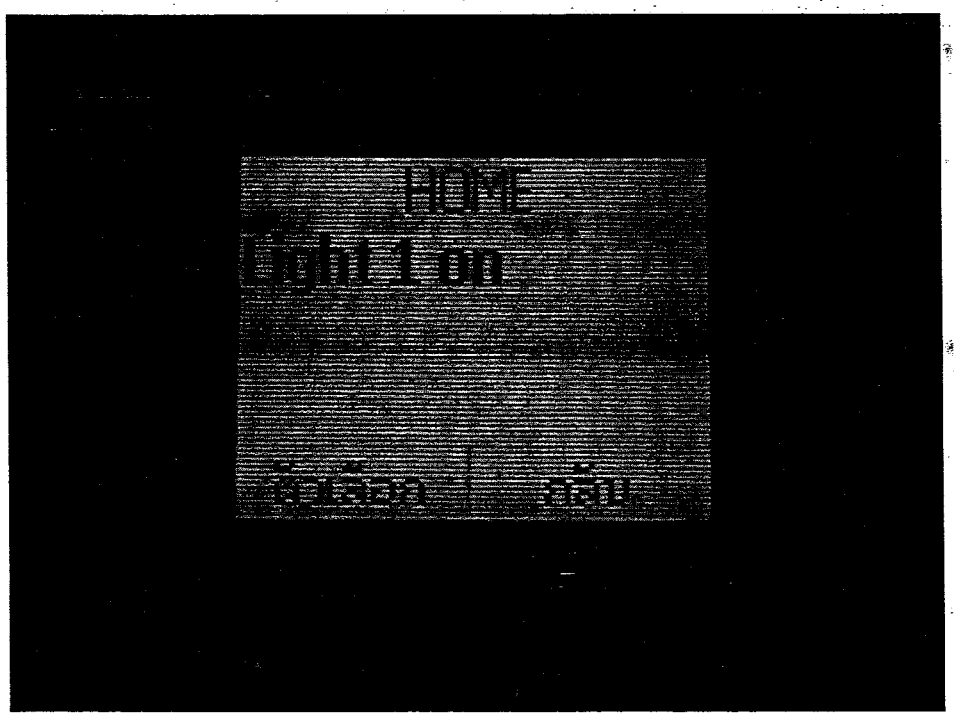
เมื่อหน้าต่างบานที่ 1 ถูกเปิดหน้าจอแสดงสถานะจะแสดงดังรูปที่ 4.21 และจะมีข้อความส่งมาเตือนดังรูปที่ 4.22



รูปที่ 4.21 หน้าจอแสดงสถานะเมื่อหน้าต่างที่ 1 ถูกเปิด

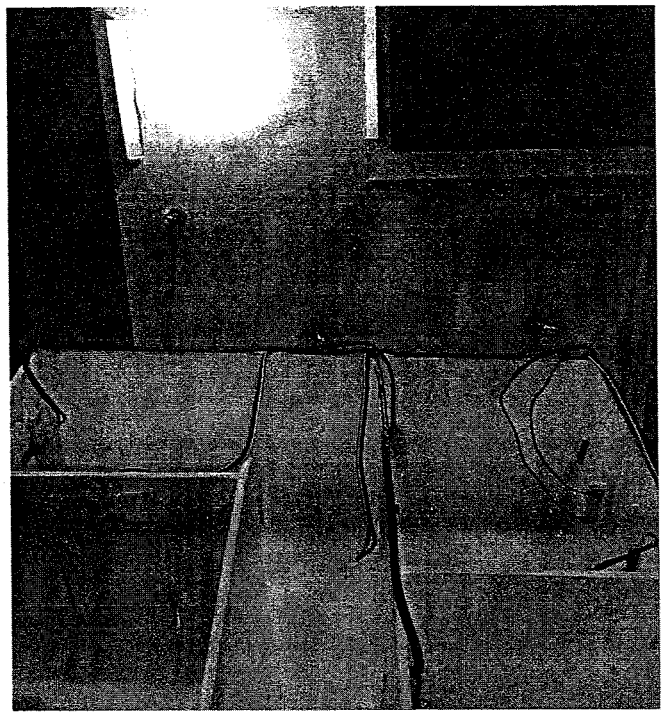
รูปที่ 4.22 ข้อความเตือน เมื่อ หน้าต่างบานที่ 1 ถูกเปิด

ทดลองควบคุมอุปกรณ์โดยการส่งข้อความเข้า โดยพิมพ์ข้อความซึ่งเป็นแบบแผนตายตัว โดยทดลองส่งข้อความเพื่อเปิดและปิดไฟดวงที่หนึ่งดังรูปที่ 4.23



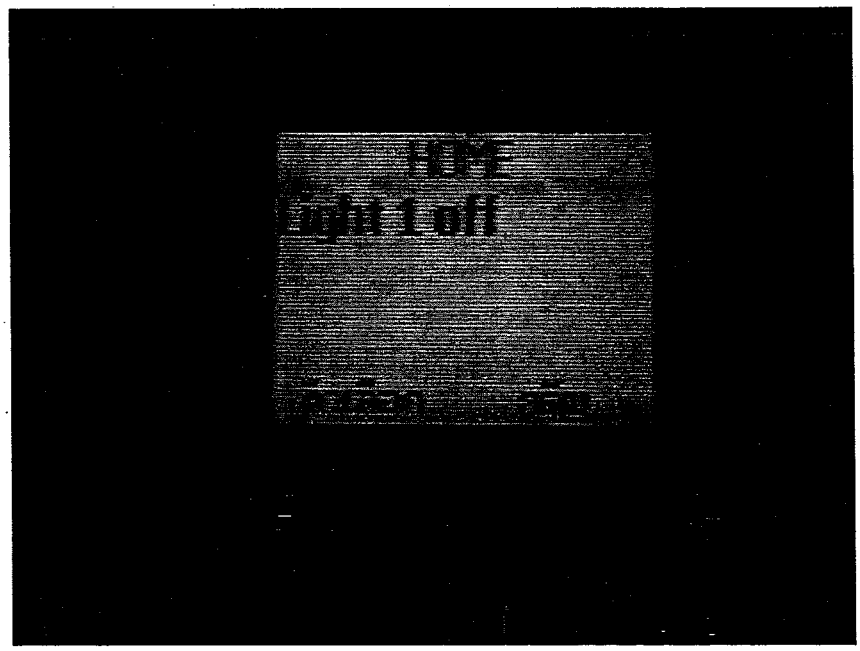
รูปที่ 4.23 แสดงแบบข้อความที่ส่งเพื่อควบคุมหลอดไฟดวงที่ 1

ผลจากการส่งข้อความ ไปควบคุมให้มีการเปิดหลอดไฟ จะทำให้หลอดไฟดวงที่ 1 ติด



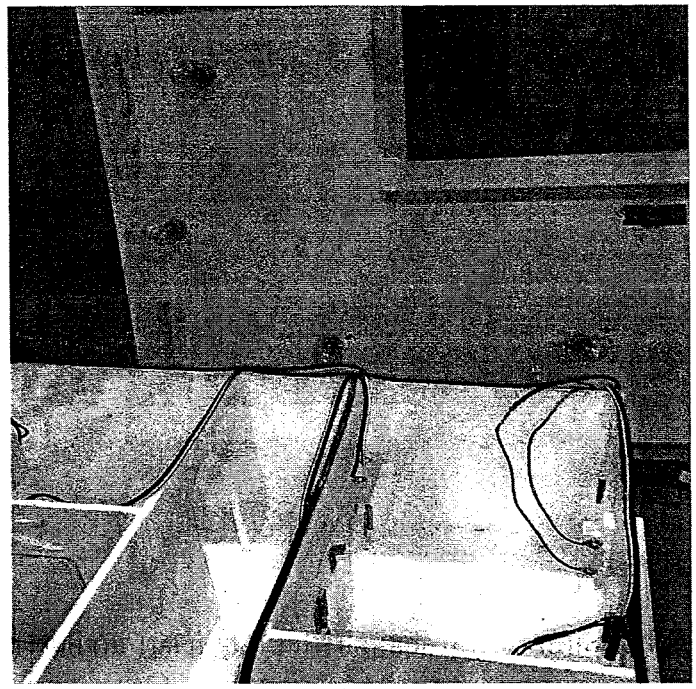
รูปที่ 4.24 แสดงรูปหลอดไฟดวงที่ 1 เมื่อมีข้อความสั่งให้เปิด

ทดลองส่งข้อความเพื่อปิดหลอดไฟดวงที่ 1



รูปที่ 4.25 แสดงแบบข้อความที่ส่งเพื่อควบคุมหลอดไฟดวงที่ 1

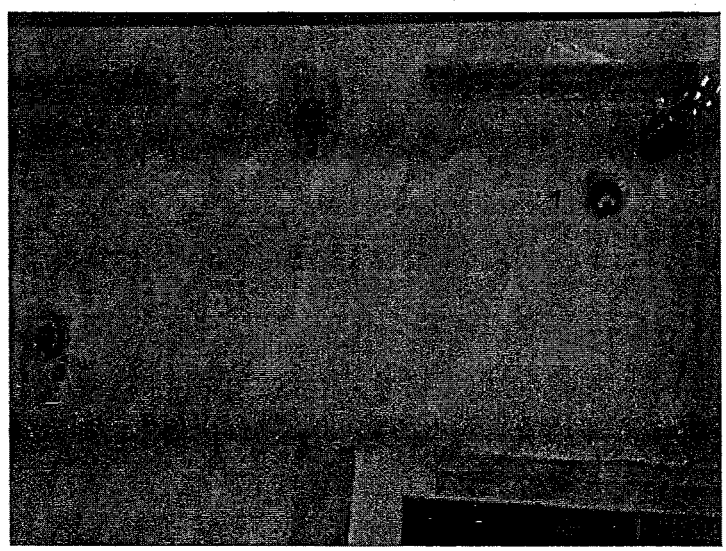
ผลจากการส่งข้อความ ไปควบคุมให้มีการปิดหลอดไฟ จะทำให้หลอดไฟดวงที่ 1 ดับ



รูปที่ 4.26 แสดงรูปหลอดไฟดวงที่ 1 เมื่อมีข้อความสั่งให้ปิด

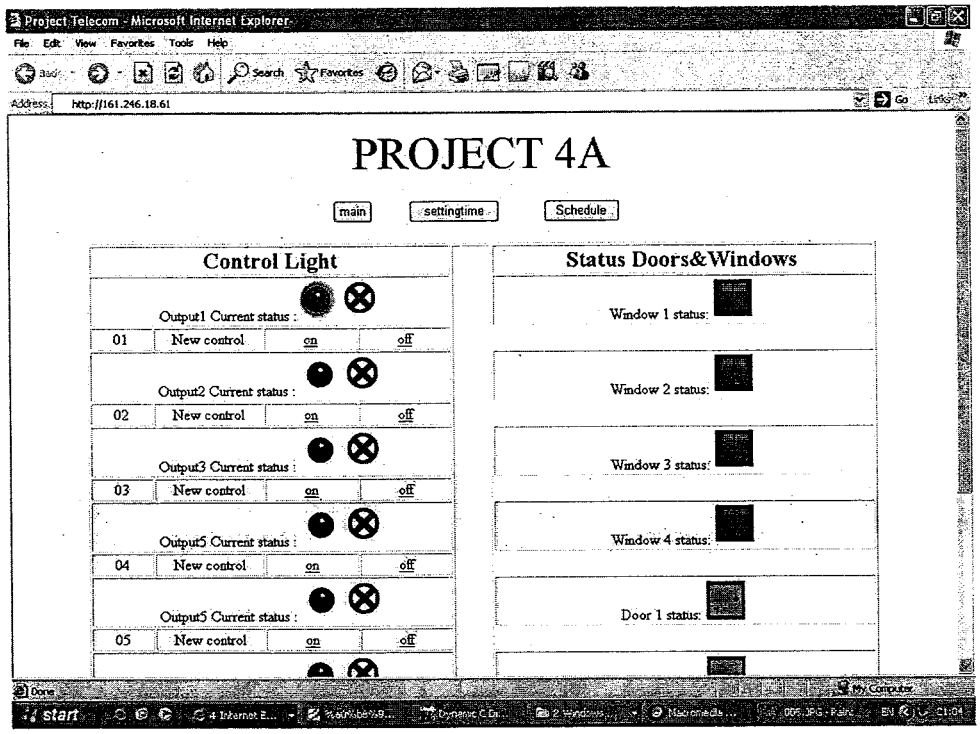
การสั่งเปิดหลอดไฟทั้งจากทางหน้าเว็บเพจและจากทางการส่ง SMS ไปควบคุม หากปลายทางที่หลอดไฟเกิดข้อผิดพลาด เช่น อุปกรณ์เกิดความเสียหาย หรือไม่ได้มีการต่ออุปกรณ์นั้นๆ จริง วงจรตรวจสอบสถานะจะส่งสถานะมายังหน้าเว็บเพจเพื่อให้ผู้ใช้ทราบเพื่อแก้ไขต่อไป

ทดลองโดยการนำหลอดไฟหลอดที่ 1 ออกจากขั้วหลอด



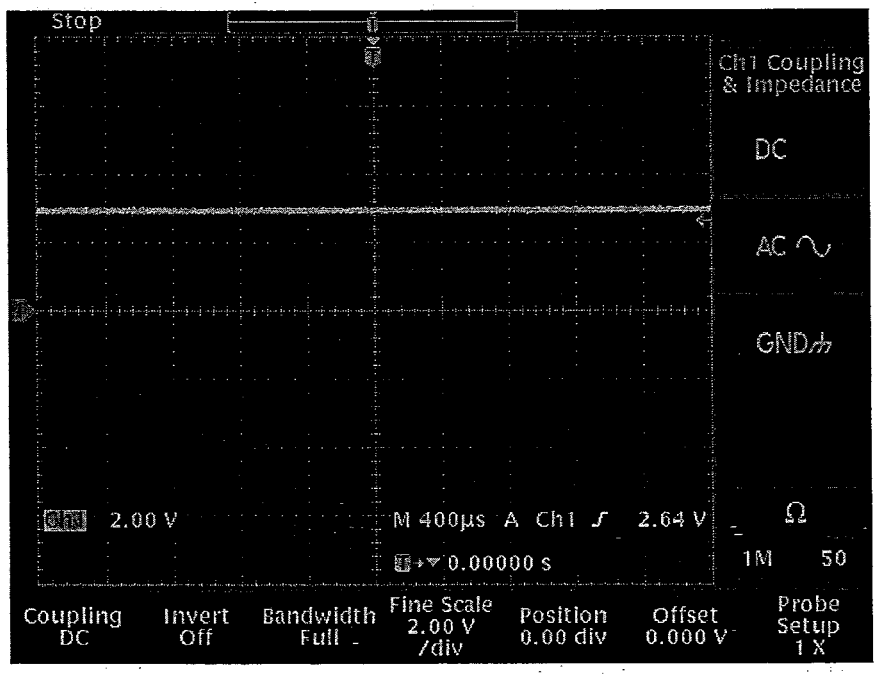
รูปที่ 4.27 เมื่อนำหลอดไฟหลอดที่ 1 ออกจากขั้วหลอด

เมื่อสั่งเปิดไฟหลอดที่ 1 จะเห็นว่าทางหน้าเว็บเพจนั้นจะแสดงสถานะดังรูปที่ 4.28

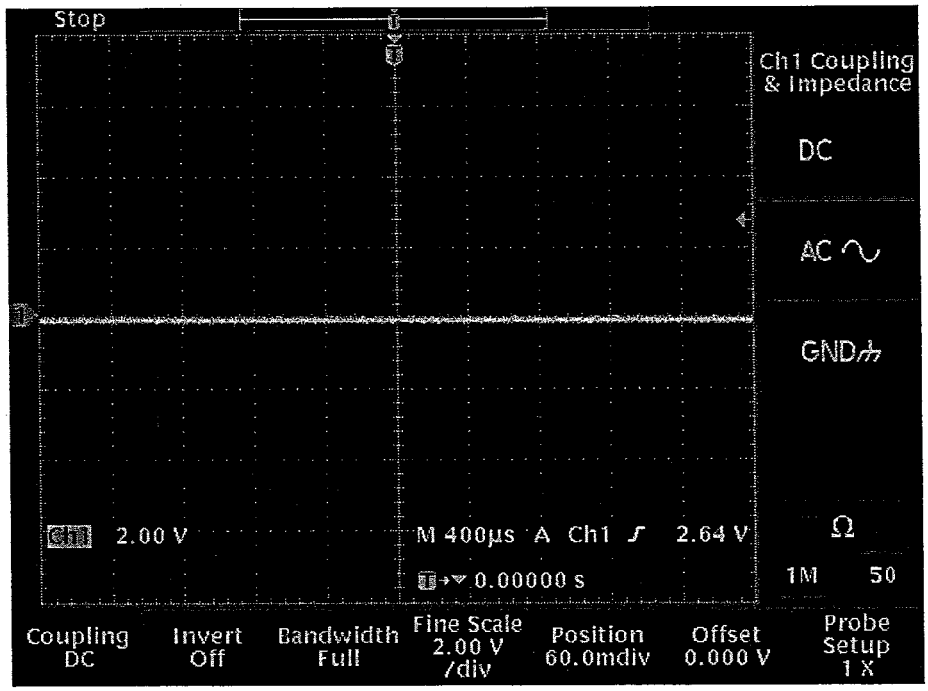


รูปที่ 4.28 แสดงสถานะเมื่อสั่งเปิดไฟแต่เกิดข้อผิดพลาดที่ปลายทาง

4.4 การทดลองแสดงผลทางฮาร์ดแวร์

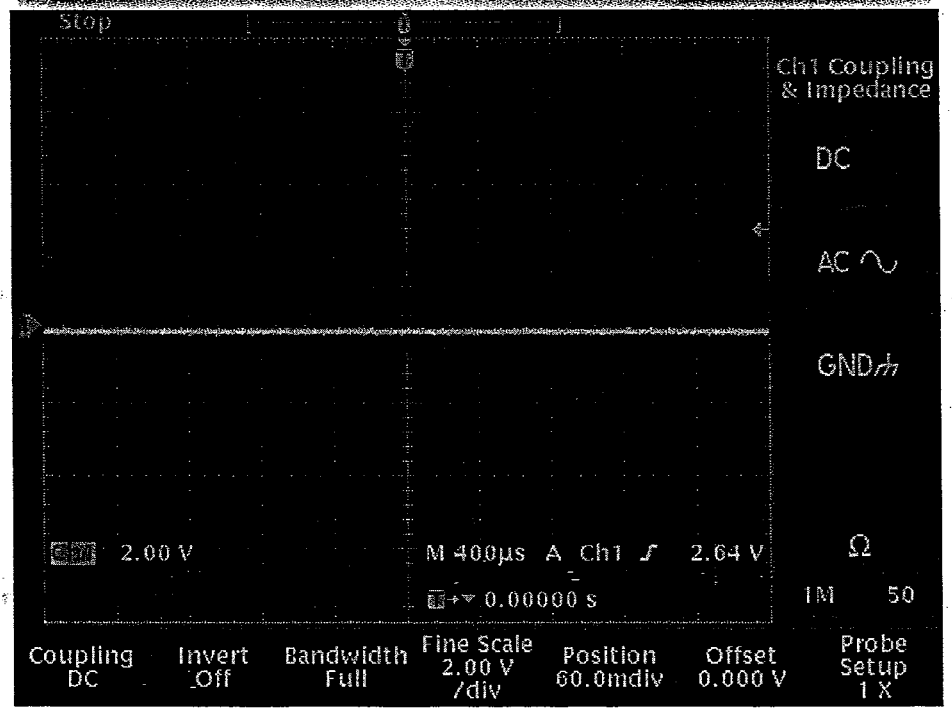


รูปที่ 4.29 แสดงสภาวะแรงดันเมื่อกดปุ่มเปิดไฟ

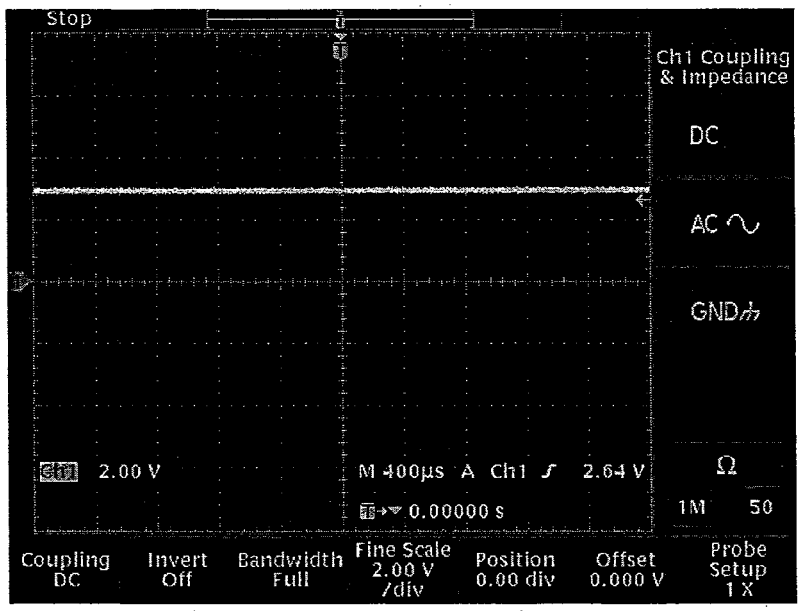


รูปที่ 4.30 แสดงสภาวะแรงดันเมื่อกดปุ่มปิดไฟ

จากรูปที่ 4.29 และ 4.30 เป็นการวัดสัญญาณที่ พอร์ต B ของ RCM 3200 ซึ่งเป็นพอร์ตที่จะส่ง ค่าไปควบคุมการเปิด-ปิด หลอดไฟของวงจรสวิตช์



รูปที่ 4.31 แสดงสภาวะแรงดันเมื่อเปิดประตู หน้าต่าง



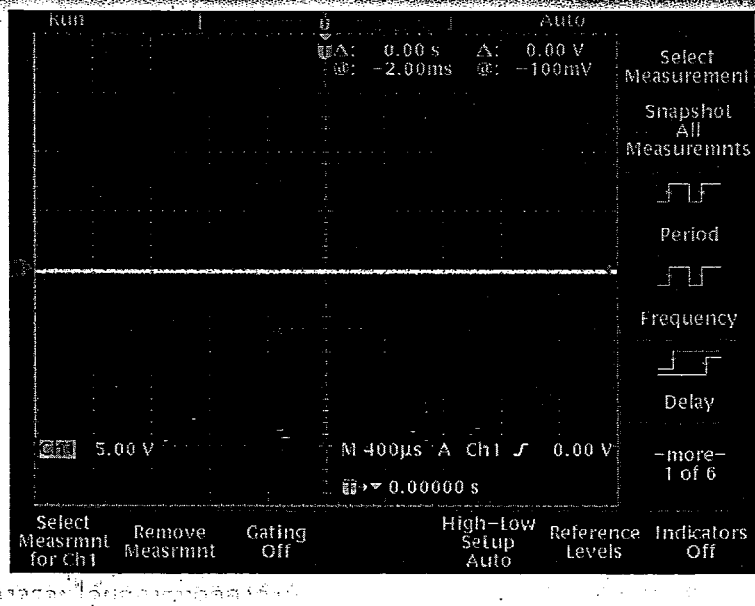
รูปที่ 4.32 แสดงสภาวะแรงดันเมื่อปิดประตู หน้าต่าง

จากรูปที่ 4.31 และ 4.32 เป็นการวัดสัญญาณที่เข้ามาที่ พอร์ต A ของ RCM 3200 ซึ่งเป็นพอร์ตที่ทำการรับค่า การเปิด-ปิดของประตูและหน้าต่าง แล้วจึงทำการประมวลผลโดยจะแสดงผลผ่านทางหน้าเว็บเพจ

ผลการทดลองของวงจรตรวจสอบสถานะของอุปกรณ์ไฟฟ้า

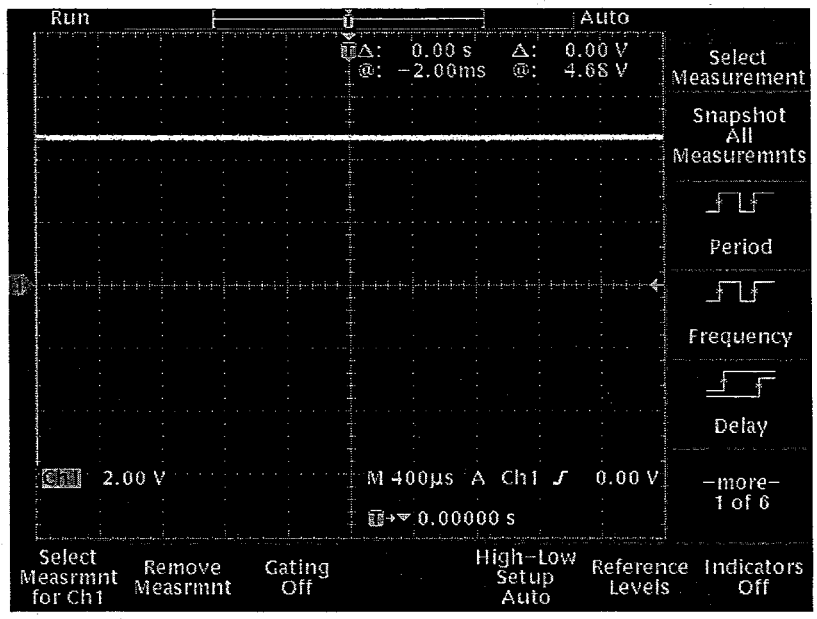
จากการทดลอง เมื่อต่อวงจรตรวจสอบสถานะของอุปกรณ์ไฟฟ้า แล้วต่อวงจรกับโหลด จากนั้นจึงจ่ายไฟให้กับวงจรจะได้ผลการทดลองดังนี้

เมื่ออุปกรณ์ไฟฟ้าทำงานสัญญาณเอาต์พุตจะเป็น “0” ซึ่งได้สัญญาณ ดังรูปที่ 4.33



รูปที่ 4.33 สัญญาณเอาต์พุตของวงจรตรวจสอบสถานะเมื่ออุปกรณ์ไฟฟ้าทำงาน

เมื่ออุปกรณ์ไฟฟ้าไม่ทำงานสัญญาณเอาต์พุตจะเป็น “0” ซึ่งได้สัญญาณ ดังรูปที่ 4.34



รูปที่ 4.34 สัญญาณเอาต์พุตของวงจรตรวจสอบสถานะเมื่ออุปกรณ์ไฟฟ้าไม่ทำงาน

บทที่ 5
สรุปผลและวิจารณ์การทดลอง

จากการศึกษาการทำงานของไมโครโปรเซสเซอร์ Rabbit RCM3200 ที่มีความสามารถพร้อมจะใช้งานในระบบเครือข่ายอินเตอร์เน็ต ผนวกกับความรู้ที่ได้ศึกษาเพิ่มเติม ทำให้สามารถออกแบบระบบที่มีความสมบูรณ์ในการควบคุมการเปิด-ปิดอุปกรณ์ โดยการควบคุมผ่านทางเว็บบราวเซอร์หรือทางโทรศัพท์มือถือ และสามารถป้องกันภัยจากการถูกบุกรุกทางประตูหรือหน้าต่างได้ รวมไปถึงระบบป้องกันจากผู้ที่ไม่ได้รับอนุญาตให้เข้ามาสั่งงานควบคุมและยังสามารถตรวจสอบสถานะการทำงานของอุปกรณ์ผ่านทางหน้าเว็บเพจได้ด้วย รวมถึงสามารถตั้งเวลาไว้ล่วงหน้าเพื่อควบคุมอุปกรณ์ต่างๆได้ด้วย

5.1 สรุปผลของการดำเนินโครงงาน

โครงงานนี้ได้นำ RCM3200 มาใช้ในการสื่อสารผ่านระบบอินเตอร์เน็ต ซึ่งใช้โปรโตคอล TCP/IP มาใช้ในการทำงาน โดยการเชื่อมต่อเป็นระบบควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า โดยระบบควบคุมสามารถควบคุมการเปิด-ปิดของอุปกรณ์ไฟฟ้าได้จริงผ่านทางหน้าเว็บเพจหรือโทรศัพท์มือถือและสามารถป้องกันภัยจากการถูกบุกรุกทางประตูหรือหน้าต่างได้ โดยการตรวจสอบสถานะนั้นยังสามารถทำได้โดยตรวจสอบผ่านทางหน้าเว็บเพจ

5.2 แนวทางการพัฒนาโครงงาน

1. สามารถสั่งงานควบคุมอุปกรณ์ไฟฟ้าชนิดอื่น ๆได้
2. สามารถนำไปประยุกต์เพื่อการรักษาความปลอดภัยได้มากขึ้น
3. สามารถนำไปประยุกต์เพื่อพัฒนาในเชิงพานิชย์ได้

ภาคผนวก

โปรแกรมหลัก

```
#class auto
#define TCPCONFIG 1
#define HTTP_MAXSERVERS 1
#define MAX_TCP_SOCKET_BUFFERS 1
#define REDIRECTHOST            _PRIMARY_STATIC_IP
#define REDIRECTTO              "http://" REDIRECTHOST ""

#define DINBUFSIZE 15
#define DOUTBUFSIZE 15
#define TIMEOUT 20UL   // will time out 20 milliseconds after receiving any
#define MAXSIZE  128
#define MAXSIZE2 160

#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"

#ximport "C:/bird/bee/main2.html"     index_html
#ximport "C:/bird/bee/form.html"    form_html
#ximport "C:/bird/bee/settingtime.html" settingtime_html
#ximport "C:/bird/bee/ledon.gif"     ledon_gif
#ximport "C:/bird/bee/ledoff.gif"    ledoff_gif
#ximport "C:/bird/bee/windowclose.gif"     windowclose_gif
#ximport "C:/bird/bee/windowopen.gif"    windowopen_gif
#ximport "C:/bird/bee/doorclose.gif"     doorclose_gif
#ximport "C:/bird/bee/dooropen.gif"    dooropen_gif

int j,k,g,p,n;
const HttpType http_types[] =
{
  { ".shtml", "text/html", shtml_handler}, // ssi
  { ".html", "text/html", NULL},         // html
  { ".cgi", "", NULL},                 // cgi
```

```
  { ".gif", "image/gif", NULL}
};
const char led_on_gif[] = { "ledon.gif" };
const char led_off_gif[] = { "ledoff.gif" };
/*
const HttpRealm admin =
{
        "kmitl","kmitl","Log In"
}; */

char led1[15];
char led2[15];
char led3[15];
char led4[15];
char led5[15];
char led6[15];
char led7[15];
char led8[15];
char mssg[MAXSIZE];
char msg2[MAXSIZE2];
char
mmm[15],msg[MAXSIZE],a[15],sn[15],wd1[50],wd2[50],wd3[50],wd4[50],wd5[50],wd6[50],wd7[50],
wd8[50];
int abc,xyz;
int tmp1,tmp2,tmp3,tmp4,tmp5,tmp6,tmp7,tmp8;
//Group of string for control gsm module
static const char s[] = "AT+CMGS=\"+66858297830\"\x0d";
static const char s1[] = "Warning Window1open\x1a\x0d";
static const char s2[] = "Warning Window2open\x1a\x0d";
static const char s3[] = "Warning Window3open\x1a\x0d";
static const char s4[] = "Warning Window4open\x1a\x0d";
static const char s5[] = "Warning Door1open\x1a\x0d";
static const char s6[] = "Warning Door2open\x1a\x0d";
static const char s7[] = "Warning Door3open\x1a\x0d";
```

```
static const char s8[] = "Warning Door4open\x1a\x0d";

#define MAX_FORMSIZE            64
typedef struct {
        char *name;
        char value[MAX_FORMSIZE];
} FORMType;
FORMType FORMSpec[9];

typedef struct{ //structure  for keep time & date
        char *name;
  char ch[MAX_FORMSIZE];
  char turn[MAX_FORMSIZE];
  char day[MAX_FORMSIZE];
  char month[MAX_FORMSIZE];
  char year[MAX_FORMSIZE];
  char hour[MAX_FORMSIZE];
  char min[MAX_FORMSIZE];
  char sec[MAX_FORMSIZE];
}statustype;
statustype st_con[9];

typedef struct{//structure for keep object from schedule table
  char *name;
  char ch[MAX_FORMSIZE];
  char status[MAX_FORMSIZE];
  char date[MAX_FORMSIZE];
  char time[MAX_FORMSIZE];
}keeptype;
keeptype keep_con[9];

void sendsms()  /*This function use for sending the sms to the mobile phone via serial
port  from rabbit to gsm module. */
{
```

```
//Send atcommand AT+CMGS
        serDputs(s);
        serDrdFlush();
// wait for answer form gsm module
        while((n=serDread(mssg,MAXSIZE -1,TIMEOUT))==0);
        mssg[n]=0;
//send message
        serDputs(sn);
        serDrdFlush();
        while((n=serDread(mssg,MAXSIZE -1,TIMEOUT))==0);
        mssg[n]=0;
        serDrdFlush();
        while((n=serDread(mssg,MAXSIZE -1,TIMEOUT))==0);
        mssg[n]=0;
}
void update_input() //Show Sensor status
{
        if(BitRdPortI(PADR,0)==1)
        {
                tmp1=0 ;
                strcpy(wd1,"windowclose.gif");
        }
        if(BitRdPortI(PADR,0)==0)
        {
                tmp1++;
                strcpy(wd1,"windowopen.gif");
                strcpy(sn,s1);
                if(tmp1==1)
                {
                        sendsms();
                        tmp1 = 20;
                }
        }
        if(BitRdPortI(PADR,1)==1)
```

```
{
        tmp2=0;
        strcpy(wd2,"windowclose.gif");
}
if(BitRdPortI(PADR,1)==0)
{
        tmp2++;
        strcpy(wd2,"windowopen.gif");
        strcpy(sn,s2);
        if(tmp2==1)
        {
                sendsms();
                tmp2 = 20;
        }
}
if(BitRdPortI(PADR,2)==1)
{
        tmp3=0;
        strcpy(wd3,"windowclose.gif");
}
if(BitRdPortI(PADR,2)==0)
{
        tmp3++;
        strcpy(wd3,"windowopen.gif");
        strcpy(sn,s3);
        if(tmp3==1)
        {
                sendsms();
                tmp3 = 20;
        }
}
if(BitRdPortI(PADR,3)==1)
{
        tmp4=0 ;
```

```
        strcpy(wd4,"windowclose.gif");
}
if(BitRdPortI(PADR,3)==0)
{
        tmp4++;
        strcpy(wd4,"windowopen.gif");
        strcpy(sn,s4);
        if(tmp4==1)
        {
                sendsms();
                tmp4 = 20;
        }
}
if(BitRdPortI(PADR,4)==1)
{
        tmp5=0;
        strcpy(wd5,"doorclose.gif");
}
if(BitRdPortI(PADR,4)==0)
{
        tmp5++;
        strcpy(wd5,"dooropen.gif");
        strcpy(sn,s5);
        if(tmp5==1)
        {
                sendsms();
                tmp5 = 20;
        }
}
if(BitRdPortI(PADR,5)==1)
{
        tmp6=0 ;
        strcpy(wd6,"doorclose.gif");
}
```

```
if(BitRdPortI(PADR,5)==0)
{
        tmp6++;
        strcpy(wd6,"dooropen.gif");
        strcpy(sn,s6);
        if(tmp6==1)
        {
                sendsms();
                tmp6 = 20;
        }
}
if(BitRdPortI(PADR,6)==1)
{
        tmp7=0;
        strcpy(wd7,"doorclose.gif");
}
if(BitRdPortI(PADR,6)==0)
{
        tmp7++;
        strcpy(wd7,"dooropen.gif");
        strcpy(sn,s7);
        if(tmp7==1)
        {
                sendsms();
                tmp7 = 20;
        }
}
if(BitRdPortI(PADR,7)==1)
{
        tmp8=0;
        strcpy(wd8,"doorclose.gif");
}
if(BitRdPortI(PADR,7)==0)
{
```

```
                tmp8++;
                strcpy(wd8,"dooropen.gif");
                strcpy(sn,s8);
                if(tmp8==1)
                {
                        sendsms();
                        tmp8 = 20;
                }
        }
}
void check()//check Msg Number
{
  switch (mssg[p])
  {
        case 48 :
                strcpy(a,"0");
                break;
        case 49 :
                strcpy(a,"1");
                break;
        case 50 :
                strcpy(a,"2");
                break;
        case 51 :
                strcpy(a,"3");
                break;
        case 52 :
                strcpy(a,"4");
                break;
        case 53 :
                strcpy(a,"5");
                break;
        case 54 :
                strcpy(a,"6");
```

```
                break;
        case 55 :
                strcpy(a,"7");
                break;
        case 56 :
                strcpy(a,"8");
                break;
        case 57 :
                strcpy(a,"9");
                break;
        default :
                strcpy(a,"");
                break;
  }
}
//function on or off for  show status
int on1(HttpState* state)
{
  if (strcmp(led1,"ledoff.gif")==0)
    strcpy(led1,"ledon.gif");

   cgi_redirectto(state,REDIRECTTO);
  return 0;
}

int off1(HttpState* state)
{
  if (strcmp(led1,"ledon.gif")==0)
    strcpy(led1,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on2(HttpState* state)
{
```

```
  if (strcmp(led2,"ledoff.gif")==0)
    strcpy(led2,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}

int off2(HttpState* state)
{
  if (strcmp(led2,"ledon.gif")==0)
    strcpy(led2,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on3(HttpState* state)
{
  if (strcmp(led3,"ledoff.gif")==0)
    strcpy(led3,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int off3(HttpState* state)
{
  if (strcmp(led3,"ledon.gif")==0)
    strcpy(led3,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on4(HttpState* state)
{
  if (strcmp(led4,"ledoff.gif")==0)
    strcpy(led4,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}
```

```
int off4(HttpState* state)
{
  if (strcmp(led4,"ledon.gif")==0)
    strcpy(led4,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on5(HttpState* state)
{
  if (strcmp(led5,"ledoff.gif")==0)
    strcpy(led5,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int off5(HttpState* state)
{
  if (strcmp(led5,"ledon.gif")==0)
    strcpy(led5,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on6(HttpState* state)
{
  if (strcmp(led6,"ledoff.gif")==0)
    strcpy(led6,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}

int off6(HttpState* state)
{
  if (strcmp(led6,"ledon.gif")==0)
    strcpy(led6,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
```

```
  return 0;
}
int on7(HttpState* state)
{
  if (strcmp(led7,"ledoff.gif")==0)
    strcpy(led7,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}

int off7(HttpState* state)
{
  if (strcmp(led7,"ledon.gif")==0)
    strcpy(led7,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
int on8(HttpState* state)
{
  if (strcmp(led8,"ledoff.gif")==0)
    strcpy(led8,"ledon.gif");
   cgi_redirectto(state,REDIRECTTO);
  return 0;
}

int off8(HttpState* state)
{
  if (strcmp(led8,"ledon.gif")==0)
    strcpy(led8,"ledoff.gif");
  cgi_redirectto(state,REDIRECTTO);
  return 0;
}
/*
 * parse the url-encoded POST data into the FORMSpec struct
```

```
* (ie: parse 'foo=bar&baz=qux' into the struct
*/
int parse_post(HttpState* state)
{
        auto int retval;
        auto int i;

        // state->s is the socket structure, and state->p is pointer
        // into the HTTP state buffer (initially pointing to the beginning
        // of the buffer).  Note that state->p was set up in the submit
        // CGI function.  Also note that we read up to the content_length,
        // or HTTP_MAXBUFFER, whichever is smaller.  Larger POSTs will be
        // truncated.
        retval = sock_aread(&state->s, state->p,
                        (state->content_length < HTTP_MAXBUFFER-1)?
                         (int)state->content_length:HTTP_MAXBUFFER-1);
        if (retval < 0) {
                // Error--just bail out
                return 1;
        }

        // Using the subsubstate to keep track of how much data we have received
        state->subsubstate += retval;

        if (state->subsubstate >= state->content_length) {
                // NULL-terminate the content buffer
                state->buffer[(int)state->content_length] = '\0';

                // Scan the received POST information into the FORMSpec structure
                for(i=0; i<(sizeof(FORMSpec)/sizeof(FORMType)); i++) {
                        http_scanpost(FORMSpec[i].name, state->buffer, FORMSpec[i].value,
                                MAX_FORMSIZE);
                }
```

```
                // Finished processing--returning 1 indicates that we are done
                return 1;
        }
        // Processing not finished--return 0 so that we can be called again
        return 0;
}


void check_time()//check the time for schedule
{
  struct tm rtc;
  char time[60];
  char date[60];
  unsigned long t0;
  unsigned int tmp;
  char year[5],month[3],day[3],hour[3],min[3],sec[3];
  t0 = read_rtc();
  mktm(&rtc,t0);
  tmp = 1900+rtc.tm_year;
  sprintf(year,"%04d",tmp);
  sprintf(month,"%02d",rtc.tm_mon);
  sprintf(day,"%02d",rtc.tm_mday);
  sprintf(hour,"%02d",rtc.tm_hour);
  sprintf(min,"%02d",rtc.tm_min);
  sprintf(sec,"%02d",rtc.tm_sec);


  strcpy(time,hour);
  strcat(time,":");
  strcat(time,min);
  strcat(time,":");
  strcat(time,sec);
  strcpy(date,day);
  strcat(date,"/");
```

```
 strcat(date,month);
 strcat(date,"/");
 strcat(date,year);
 printf("...%s...%s....",hour,min);
for(k=1;k<j;k++)
 {
 if(strcmp(keep_con[k].ch,"1")==0 && strcmp(keep_con[k].status,"on")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
   strcpy(led1,"ledon.gif");
 }
 else if(strcmp(keep_con[k].ch,"1")==0 && strcmp(keep_con[k].status,"off")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
   strcpy(led1,"ledoff.gif");
 }
 else if(strcmp(keep_con[k].ch,"2")==0 && strcmp(keep_con[k].status,"on")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
   strcpy(led2,"ledon.gif");
 }
 else if(strcmp(keep_con[k].ch,"2")==0 && strcmp(keep_con[k].status,"off")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
   strcpy(led2,"ledoff.gif");
 }
 else if(strcmp(keep_con[k].ch,"3")==0 && strcmp(keep_con[k].status,"on")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
   strcpy(led3,"ledon.gif");
 }
 else if(strcmp(keep_con[k].ch,"3")==0 && strcmp(keep_con[k].status,"off")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
 {
```

```
      strcpy(led3,"ledoff.gif");
    }
    else if(strcmp(keep_con[k].ch,"4")==0 && strcmp(keep_con[k].status,"on")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led4,"ledon.gif");
    }
    else if(strcmp(keep_con[k].ch,"4")==0 && strcmp(keep_con[k].status,"off")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led4,"ledoff.gif");
    }
    else if(strcmp(keep_con[k].ch,"5")==0 && strcmp(keep_con[k].status,"on")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led5,"ledon.gif");
    }
    else if(strcmp(keep_con[k].ch,"5")==0 && strcmp(keep_con[k].status,"off")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led5,"ledoff.gif");
    }
    else if(strcmp(keep_con[k].ch,"6")==0 && strcmp(keep_con[k].status,"on")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led6,"ledon.gif");
    }
    else if(strcmp(keep_con[k].ch,"6")==0 && strcmp(keep_con[k].status,"off")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
    {
      strcpy(led6,"ledoff.gif");
    }
    else if(strcmp(keep_con[k].ch,"7")==0 && strcmp(keep_con[k].status,"on")==0 &&
  strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
```

```
  {
    strcpy(led7,"ledon.gif");
  }
   else if(strcmp(keep_con[k].ch,"7")==0 && strcmp(keep_con[k].status,"off")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
  {
    strcpy(led7,"ledoff.gif");
  }
   else if(strcmp(keep_con[k].ch,"8")==0 && strcmp(keep_con[k].status,"on")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
  {
    strcpy(led8,"ledon.gif");
  }
   else if(strcmp(keep_con[k].ch,"8")==0 && strcmp(keep_con[k].status,"off")==0 &&
strcmp(keep_con[k].date,date)==0 && strcmp(keep_con[k].time,time)==0)
  {
    strcpy(led8,"ledoff.gif");
  }
  }
}

void get_ch_status()
{
        strcpy(st_con[1].ch,FORMSpec[1].value);
      strcpy(st_con[1].turn,FORMSpec[2].value);
      strcpy(st_con[1].hour,FORMSpec[3].value);
      strcpy(st_con[1].min,FORMSpec[4].value);
      strcpy(st_con[1].day,FORMSpec[5].value);
      strcpy(st_con[1].month,FORMSpec[6].value);
      strcpy(st_con[1].year,FORMSpec[7].value);
      strcpy(st_con[1].sec,FORMSpec[8].value);
      keep_time();

}
```

```
int schedule(HttpState* state)  //page for the web
{
        auto int i;
  char var[15];
        if(state->length) {
                /* buffer to write out */
                if(state->offset < state->length) {
                        state->offset += sock_fastwrite(&state->s,
                                        state->buffer + (int)state->offset,
                                        (int)state->length - (int)state->offset);
                } else {
                        state->offset = 0;
                        state->length = 0;
                }
        } else {
                switch(state->substate) {
                case 0:

                        strcpy(state->buffer, "HTTP/1.0 200 OK\r\n\r\n");
                        state->length = strlen(state->buffer);
                        state->offset = 0;
                        state->substate++;
                        break;

                 case 1:
                        strcpy(state->buffer,
"<html><head><title>Schedule</title></head><body>\r\n");
                        state->length = strlen(state->buffer);
                        state->substate++;
                        break;

      case 2:
                        /* init the FORMSpec data */
                        FORMSpec[0].value[0] = '\0';
```

```
                    FORMSpec[1].value[0] = '\0';
        FORMSpec[2].value[0] = '\0';
        FORMSpec[3].value[0] = '\0';
        FORMSpec[4].value[0] = '\0';
        FORMSpec[5].value[0] = '\0';
        FORMSpec[6].value[0] = '\0';
        FORMSpec[7].value[0] = '\0';
        FORMSpec[8].value[0] = '\0';
        state->p = state->buffer;
                    state->substate++;
                    break;

            case 3:
                    /* parse the POST information */
                    if(parse_post(state)) {
        state->p = state->buffer;
        get_ch_status();
        strcpy(state->buffer,"<p align=\"center\"> </p><p align=\"center\"><font
size=\"+4\">PROJECT 4A</font></p><div align=\"center\"><table width=\"36%\"
border=\"0\"><tr><td width=\"20%\" height=\"43\"><div align=\"center\">");
        strcat(state->buffer,"<form method=\"post\" action=\"http://10.10.6.101/\"><input type=\"submit\"
name=\"Submit\" value=\"main\"></form>");
        strcat(state->buffer,"</div></td><td width=\"34%\"><div align=\"center\"><form
method=\"post\" action=\"settingtime.html\"><input type=\"submit\" name=\"Submit\"
value=\"settingtime\"></form></div></td>");
            strcat(state->buffer,"<td width=\"34%\"><div align=\"center\"><form method=\"post\"
action=\"sched.cgi\"><input type=\"submit\" name=\"Submit\"
value=\"Schedule\"></form></div></td></tr></table></div>");
        strcat(state->buffer,"<table width=\"75%\"border=\"1\"align=\"center\">");
        strcat(state->buffer,"<tr><td colspan=\"4\"><div align=\"center\">Schedule
view</div></td></tr>");
        strcat(state->buffer,"<tr><td width=\"22%\"><div align=\"center\">Ch_number</div></td>");
```

```
strcat(state->buffer,"<td width=\"26%\"><div align=\"center\">Turn</div></td><td
width=\"25%\"><p align=\"center\">DAY/MTH/YEAR</p><p align=\"center\">HOUR
:MIN</p></td></tr>");
state->length = strlen(state->buffer);
                        state->substate++;
                        }else {
                        }
                        break;

                case 4:
    strcpy(state->buffer," ");
    for(k=1;k<g;k++)
{
strcat(state->buffer,"<tr><td><div align=\"center\">Channel ");
strcat(state->buffer,keep_con[k].ch);
strcat(state->buffer,":</div></td><td><div align=\"center\">");
strcat(state->buffer,keep_con[k].status);
strcat(state->buffer,"</div></td><td><div align=\"center\">");
strcat(state->buffer,keep_con[k].date);
strcat(state->buffer," ");
strcat(state->buffer,keep_con[k].time);
}
strcat(state->buffer,"<p><a href=\"/settingtime.html\">back</a></body></html>\r\n");
state->length = strlen(state->buffer);
                        state->substate++;
                        break;

            default:

                        state->substate = 0;

                        return 1;
                }
        }
```

```
        return 0;
}

int keep_time()
{
        strcpy(keep_con[g].ch,st_con[1].ch);
    strcpy(keep_con[g].status,st_con[1].turn);
    strcpy(keep_con[g].time,st_con[1].hour);
    strcat(keep_con[g].time,":");
    strcat(keep_con[g].time,st_con[1].min);
    strcat(keep_con[g].time,":");
    strcat(keep_con[g].time,st_con[1].sec);
    strcpy(keep_con[g].date,st_con[1].day);
    strcat(keep_con[g].date,"/");
    strcat(keep_con[g].date,st_con[1].month);
    strcat(keep_con[g].date,"/");
    strcat(keep_con[g].date,st_con[1].year);
  g++;
  j++;
/*  for(k=1;k<j;k++)
  {
        printf("%s  %s  %d",keep_con[k].time,keep_con[k].date,j);
  }       */
}

void del_1(HttpState* state)
{
 strcpy(st_con[1].ch,"1");
 strcpy(st_con[1].turn,"-");
 strcpy(st_con[1].hour,"-");
 strcpy(st_con[1].min,"-");
 strcpy(st_con[1].sec,"-");
 strcpy(st_con[1].day,"-");
```

```
 strcpy(st_con[1].month,"-");
 strcpy(st_con[1].year,"-");
 schedule(state);
}
void del2(HttpState* state)
{
 strcpy(st_con[2].ch,"2");
 strcpy(st_con[2].turn,"-");
 strcpy(st_con[2].hour,"-");
 strcpy(st_con[2].min,"-");
 strcpy(st_con[2].sec,"-");
 strcpy(st_con[2].day,"-");
 strcpy(st_con[2].month,"-");
 strcpy(st_con[2].year,"-");
 schedule(state);
}
void del3(HttpState* state)
{
 strcpy(st_con[3].ch,"3");
 strcpy(st_con[3].turn,"-");
 strcpy(st_con[3].hour,"-");
 strcpy(st_con[3].min,"-");
 strcpy(st_con[3].sec,"-");
 strcpy(st_con[3].day,"-");
 strcpy(st_con[3].month,"-");
 strcpy(st_con[3].year,"-");
 schedule(state);
}
void del4(HttpState* state)
{
 strcpy(st_con[4].ch,"4");
 strcpy(st_con[4].turn,"-");
 strcpy(st_con[4].hour,"-");
 strcpy(st_con[4].min,"-");
```

```
 strcpy(st_con[4].sec,"-");
 strcpy(st_con[4].day,"-");
 strcpy(st_con[4].month,"-");
 strcpy(st_con[4].year,"-");
 schedule(state);
}
void del5(HttpState* state)
{
 strcpy(st_con[5].ch,"5");
 strcpy(st_con[5].turn,"-");
 strcpy(st_con[5].hour,"-");
 strcpy(st_con[5].min,"-");
 strcpy(st_con[5].sec,"-");
 strcpy(st_con[5].day,"-");
 strcpy(st_con[5].month,"-");
 strcpy(st_con[5].year,"-");
 schedule(state);
}
void del6(HttpState* state)
{
 strcpy(st_con[6].ch,"6");
 strcpy(st_con[6].turn,"-");
 strcpy(st_con[6].hour,"-");
 strcpy(st_con[6].min,"-");
 strcpy(st_con[6].sec,"-");
 strcpy(st_con[6].day,"-");
 strcpy(st_con[6].month,"-");
 strcpy(st_con[6].year,"-");
 schedule(state);
}
void del7(HttpState* state)
{
        g= g-1;
  j= j-1;
```

```
 schedule(state);
}
void del8(HttpState* state)
{
 strcpy(st_con[8].ch,"8");
 strcpy(st_con[8].turn,"-");
 strcpy(st_con[8].hour,"-");
 strcpy(st_con[8].min,"-");
 strcpy(st_con[8].sec,"-");
 strcpy(st_con[8].day,"-");
 strcpy(st_con[8].month,"-");
 strcpy(st_con[8].year,"-");
 schedule(state);
}
int sched(HttpState* state) //page for the web
{
  auto int i;
  char var[15];
        if(state->length) {
                /* buffer to write out */
                if(state->offset < state->length) {
                        state->offset += sock_fastwrite(&state->s,
                                        state->buffer + (int)state->offset,
                                        (int)state->length - (int)state->offset);
                } else {
                        state->offset = 0;
                        state->length = 0;
                }
        } else {
  switch(state->substate)
  {
                case 0:
     strcpy(state->buffer, "HTTP/1.0 200 OK\r\n\r\n");
                        state->length = strlen(state->buffer);
```

```
                        state->offset = 0;
                        state->substate++;
                        break;
    case 1:
                        strcpy(state->buffer,
"<html><head><title>Schedule</title></head><body>\r\n");
                        state->length = strlen(state->buffer);
                        state->substate++;
                        break;
    case 2:
        strcpy(state->buffer,"<p align=\"center\"> </p><p align=\"center\"><font
size=\"+4\">PROJECT 4A</font></p><div align=\"center\"><table width=\"36%\"
border=\"0\"><tr><td width=\"20%\" height=\"43\"><div align=\"center\">");
      strcat(state->buffer,"<form method=\"post\" action=\"http://10.10.6.101/\"><input type=\"submit\"
name=\"Submit\" value=\"main\"></form>");
      strcat(state->buffer,"</div></td><td width=\"34%\"><div align=\"center\"><form
method=\"post\" action=\"settingtime.html\"><input type=\"submit\" name=\"Submit\"
value=\"settingtime\"></form></div></td>");
        strcat(state->buffer,"<td width=\"34%\"><div align=\"center\"><form method=\"post\"
action=\"sched.cgi\"><input type=\"submit\" name=\"Submit\"
value=\"Schedule\"></form></div></td></tr></table></div>");
      strcat(state->buffer,"<table width=\"75%\"border=\"1\"align=\"center\">");
      strcat(state->buffer,"<tr><td colspan=\"4\"><div align=\"center\">Schedule
view</div></td></tr>");
      strcat(state->buffer,"<tr><td width=\"22%\"><div align=\"center\">Ch_number</div></td>");
      strcat(state->buffer,"<td width=\"26%\"><div align=\"center\">Turn</div></td><td
width=\"25%\"><p align=\"center\">DAY/MTH/YEAR</p><p align=\"center\">HOUR
:MIN</p></td><td width=\"27%\"><div align=\"center\">Del Tasks</div></td></tr>");
      state->length = strlen(state->buffer);
                        state->substate++;
        break;
    case 3:
        strcpy(state->buffer," ");
        for(k=1;k<g;k++)
```

```
    {
    strcat(state->buffer,"<tr><td><div align=\"center\">Channel ");
    strcat(state->buffer,keep_con[k].ch);
    strcat(state->buffer,":</div></td><td><div align=\"center\">");
    strcat(state->buffer,keep_con[k].status);
    strcat(state->buffer,"</div></td><td><div align=\"center\">");
    strcat(state->buffer,keep_con[k].date);
    strcat(state->buffer," ");
    strcat(state->buffer,keep_con[k].time);
    }
    strcat(state->buffer,"<p><a href=\"/settingtime.html\">back</a></body></html>\r\n");
    state->length = strlen(state->buffer);
                        state->substate++;
                        break;

                default:

                        state->substate = 0;

                        return 1;
                }
    }
    return 0;

}
const HttpSpec http_flashspec[] = //fixed option for http
{
  { HTTPSPEC_FILE,     "/",              index_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,     "/index.html",    index_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,     "/form.html",     form_html,     NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,     "/settingtime.html", settingtime_html, NULL, 0, NULL, NULL},
  //{ HTTPSPEC_FILE,   "/main.html", main_html, NULL, 0, NULL, NULL},
  //{ HTTPSPEC_FUNCTION,     "/submit.cgi",          0,  submit, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/ledon.gif",    ledon_gif,     NULL, 0, NULL, NULL},
```

```
{ HTTPSPEC_FILE,  "/ledoff.gif",    ledoff_gif,    NULL, 0, NULL, NULL},
{ HTTPSPEC_VARIABLE, "led1",  0,    led1, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led2",  0,    led2, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led3",  0,    led3, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led4",  0,    led4, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led5",  0,    led5, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led6",  0,    led6, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led7",  0,    led7, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "led8",  0,    led8, PTR16,   "%s", NULL},
{ HTTPSPEC_FILE,  "/windowclose.gif",    windowclose_gif,    NULL, 0, NULL, NULL},
{ HTTPSPEC_FILE,  "/windowopen.gif",    windowopen_gif,    NULL, 0, NULL, NULL},
{ HTTPSPEC_FILE,  "/doorclose.gif",    doorclose_gif,    NULL, 0, NULL, NULL},
{ HTTPSPEC_FILE,  "/dooropen.gif",    dooropen_gif,    NULL, 0, NULL, NULL},
{ HTTPSPEC_VARIABLE, "aaa1",  0,    wd1, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa2",  0,    wd2, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa3",  0,    wd3, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa4",  0,    wd4, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa5",  0,    wd5, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa6",  0,    wd6, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa7",  0,    wd7, PTR16,   "%s", NULL},
{ HTTPSPEC_VARIABLE, "aaa8",  0,    wd8, PTR16,   "%s", NULL},

{ HTTPSPEC_FUNCTION, "/on1.cgi", 0, on1, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on2.cgi", 0, on2, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on3.cgi", 0, on3, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on4.cgi", 0, on4, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on5.cgi", 0, on5, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on6.cgi", 0, on6, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on7.cgi", 0, on7, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/on8.cgi", 0, on8, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/off1.cgi", 0, off1, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/off2.cgi", 0, off2, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/off3.cgi", 0, off3, 0, NULL, NULL},
{ HTTPSPEC_FUNCTION, "/off4.cgi", 0, off4, 0, NULL, NULL},
```

```
    { HTTPSPEC_FUNCTION, "/off5.cgi", 0, off5, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/off6.cgi", 0, off6, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/off7.cgi", 0, off7, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/off8.cgi", 0, off8, 0, NULL, NULL},

    { HTTPSPEC_FUNCTION, "/update.cgi", 0, schedule, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch1.cgi", 0, del_1, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch2", 0, del2, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch3", 0, del3, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch4", 0, del4, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch5", 0, del5, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch6", 0, del6, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch7", 0, sched, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/del_ch8", 0, del8, 0, NULL, NULL},
    { HTTPSPEC_FUNCTION, "/sched.cgi", 0, sched, 0, NULL, NULL}
};

void update_outputs()
{
WrPortI(PBDDR,&PBDDRShadow, 0xff);
        if( strcmp(led1,led_on_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,1,0);
   if ( strcmp(led1,led_off_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,0,0);
  if( strcmp(led2,led_on_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,1,2);
        if ( strcmp(led2,led_off_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,0,2);
  if( strcmp(led3,led_on_gif)==0) {
  BitWrPortI(PBDR ,&PBDRShadow,1,3);   }
   if ( strcmp(led3,led_off_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,0,3);
  if( strcmp(led4,led_on_gif)==0)
  BitWrPortI(PBDR ,&PBDRShadow,1,4);
```

```
 if ( strcmp(led4,led_off_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,0,4);
 if( strcmp(led5,led_on_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,1,5);
 if ( strcmp(led5,led_off_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,0,5);
 if( strcmp(led6,led_on_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,1,6);
 if ( strcmp(led6,led_off_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,0,6);
 if( strcmp(led7,led_on_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,1,7);
 if ( strcmp(led7,led_off_gif)==0)
BitWrPortI(PBDR ,&PBDRShadow,0,7);
 if ( strcmp(led8,led_on_gif)==0)
BitWrPortI(PDDR ,&PADRShadow,1,7);
 if ( strcmp(led8,led_off_gif)==0)
BitWrPortI(PDDR ,&PADRShadow,0,7);

}
void main()
{
        WrPortI(SPCR, & SPCRShadow, 0x80);
 WrPortI(PBDDR,&PBDDRShadow, 0xff);
 WrPortI(PADR,&PADRShadow,0x00);
 WrPortI(PBDR,&PBDRShadow,0x00);
 //WrPortI(PDDR,&PDDRShadow,0x00);
 //WrPortI(PDDDR,&PDDDRShadow,0xff);
        /* init FORM searchable names - must init ALL FORMSpec structs! */
 #if _BOARD_TYPE_ == 0x1200 || _BOARD_TYPE_ == 0x1201
 brdInit();
        #endif
 serDopen(19200);
 FORMSpec[0].name = "pd";
```

```
FORMSpec[1].name = "ch_load";
FORMSpec[2].name = "Turn_load";
FORMSpec[3].name = "hour_load";
FORMSpec[4].name = "min_load";
FORMSpec[5].name = "day_load";
FORMSpec[6].name = "month_load";
FORMSpec[7].name = "year_load";
FORMSpec[8].name = "sec_load";
strcpy(led1,"ledoff.gif");
strcpy(led2,"ledoff.gif");
strcpy(led3,"ledoff.gif");
strcpy(led4,"ledoff.gif");
strcpy(led5,"ledoff.gif");
strcpy(led6,"ledoff.gif");
strcpy(led7,"ledoff.gif");
strcpy(led8,"ledoff.gif");
        sock_init();
        http_init();
        tcp_reserveport(80);
g = 1;
j = 2;


        while (1) {

serDrdFlush();
//FORMSpec[0].name = "user_name";
while ((n = serDread(mssg, MAXSIZE-1, TIMEOUT)) == 0)
{
                http_handler();
  update_outputs();
  update_input();
  check_time();
}
```

```
        strcpy(msg,"AT+CMGR=");
        mssg[n] = 0;

p = 14;
check();
strcat(msg,a);

p = 15;
check();
strcat(msg,a);
strcat(msg,"\x0d");
serDputs(msg);
serDrdFlush();    // Remove any waiting chars.
while ((n = serDread(mssg, MAXSIZE-1, TIMEOUT)) == 0) ;
mssg[n] = 0;
serDrdFlush();    // Remove any waiting chars.
while ((n = serDread(mssg, MAXSIZE-1, TIMEOUT)) == 0) ;
mssg[n] = 0;
sprintf(msg2, "%s", mssg);
xyz = strlen(msg2);
p = 68;
check();
strcpy(mmm,a);
printf("\n\r%s %d\n\r",mmm,xyz);
switch (mssg[68])
{
case 49 :
if(xyz==80){
strcpy(led1,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led1,"ledoff.gif");
printf("offffff");
```

```
}
break;
case 50 :
if(xyz==80){
strcpy(led2,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led2,"ledoff.gif");
printf("offffff");
}
break;
case 51 :
if(xyz==80){
strcpy(led3,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led3,"ledoff.gif");
printf("offffff");
}
break;
case 52 :
if(xyz==80){
strcpy(led4,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led4,"ledoff.gif");
printf("offffff");
}
break;
case 53 :
if(xyz==80){
```

```
strcpy(led5,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led5,"ledoff.gif");
printf("offffff");
}
break;
case 54 :
if(xyz==80){
strcpy(led6,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led6,"ledoff.gif");
printf("offffff");
}
break;
case 55 :
if(xyz==80){
strcpy(led7,"ledon.gif");
printf("onnnnn");
}
else if(xyz==81){
strcpy(led7,"ledoff.gif");
printf("offffff");
}
break;
default:
break;
}
serDrdFlush();
}
}
```

โปรแกรมตั้งเวลา

```
#class auto


int main()
{
        char *p, *endptr, s[80], *e;
        unsigned int month, day, year, hour, minute, second, done;
        struct tm       rtc;                                    // time struct
        unsigned long   t0;                                     // seconds
        int i, j;

        for (i = 0; i < 80; ++i) s[i] = 0;

        ////////////////////////////////////////////////
        // print current date/time

        printf("The current RTC date/time is:\n\n");
        tm_rd(&rtc);
        printf("%02d/%02d/%04d %02d:%02d:%02d\n\n",
                        rtc.tm_mon, rtc.tm_mday, 1900+rtc.tm_year,
                        rtc.tm_hour, rtc.tm_min, rtc.tm_sec);

        done = 0;
        while (!done) {
                printf("Enter the correct date/time as:  mm/dd/yy hh:mm:ss\n");
                printf("or just press Enter to leave the date/time unchanged:\n\n");
                gets(s);
                p = s;
                if (!*p) {
                        printf("The date and time are unchanged.\n");
                        return 0;
                }
                month = (unsigned int)strtod(p, &endptr);
```

```
if (endptr != p && month > 0 && month < 13) {
        p = endptr + 1;
        day = (unsigned int)strtod(p, &endptr);
        if (endptr != p && day > 0 && day < 32) {
                p = endptr + 1;
                year = (unsigned int)strtod(p, &endptr);
                if (endptr != p && (year < 48 || year > 79)) {
                        if (year < 48) year += 100;
                        p = endptr + 1;
                        hour = (unsigned int)strtod(p, &endptr);
                        if (endptr != p && hour < 25) {
                                p = endptr + 1;
                                minute = (unsigned int)strtod(p, &endptr);
                                if (endptr != p && minute < 61) {
                                        p = endptr + 1;
                                        second = (unsigned int)strtod(p,
&endptr);

                                        if (second < 61) {
                                                done = 1;
                                        }
                                        else {
                                                e = "SECOND";
                                        }
                                }
                                else {
                                        e = "MINUTE";
                                }
                        }
                        else {
                                e = "HOUR";
                        }
                }
                else {
                        e = "YEAR";
```

```
                }
            }
            else {
                e = "DAY";
            }
        }
        else {
            e = "MONTH";
        }

        if (!done) {
            printf("\n\n%s Error\n\n\n", e);
            continue;
        }
    }

    ////////////////////////////////////////////////
    // change the date/time via tm_wr

    rtc.tm_sec  = second;       // 0-59
    rtc.tm_min  = minute;       // 0-59
    rtc.tm_hour = hour;         // 0-23
    rtc.tm_mday = day;          // 1-31
    rtc.tm_mon  = month;        // 1-12
    rtc.tm_year = year;         // 80-147, add 1900 to get year
                                //
    (i.e. 99 -> 1999)

    tm_wr(&rtc);                // set clock
    printf("RTC changed!\n\n");

    ////////////////////////////////////////////////
```

```
    // print new date/time

    // Note that read_rtc() will report the correct time now,
    // but tm_rd() will read incorrectly until a program is restarted!

    printf("The RTC date/time now is:\n\n");
    t0 = read_rtc();
    mktm(&rtc, t0);
    printf("%02d/%02d/%04d %02d:%02d:%02d\n\n",
                    rtc.tm_mon, rtc.tm_mday, 1900+rtc.tm_year,
                    rtc.tm_hour, rtc.tm_min, rtc.tm_sec);

    printf("The SEC_TIMER variable used by tm_rd() will be reset\n");
    printf("properly when you restart this (or any other) program.\n");
    return 0;
}
```

ตารางรหัส ASCII

ตารางรหัส ASCII ที่แสดงในภาคผนวกนี้จะแบ่งตามประเภทของอักขระ เพื่อความสะดวกในการค้นหาและง่ายต่อการอ่าน

อักขระควบคุม (Control Characters)

Char	Oct	Dec	Hex	Control-key	Control Action
NUL	0	0	0	^@	Null character
SOH	1	1	1	^A	Start of heading, = console interrupt
STX	2	2	2	^B	Start of text, maintenance mode on HP console
ETX	3	3	3	^C	End of text
EOT	4	4	4	^D	End of transmission, not the same as ETB
ENQ	5	5	5	^E	Enquiry, goes with ACK; old HP flow control
ACK	6	6	6	^F	Acknowledge, clears ENQ logon hand
BEL	7	7	7	^G	Bell, rings the bell...
BS	10	8	8	^H	Backspace, works on HP terminals/computers
HT	11	9	9	^I	Horizontal tab, move to next tab stop
LF	12	10	a	^J	Line feed
VT	13	11	b	^K	Vertical tab
FF	14	12	c	^L	From Feed, page eject
CR	15	13	d	^M	Carriage Return
SO	16	14	e	^N	Shift Out, alternate character set
SI	17	15	F	^O	Shift In, resume default character set
DLE	20	16	10	^P	Data link escape
DC1	21	17	11	^Q	XON, with XOFF to pause listings; ":okay to send".
DC2	22	18	12	^R	Device control 2, block-mode flow control
DC3	23	19	13	^S	XOFF with XOFF to pause listing
DC4	24	20	14	^T	Device control 4
NAK	25	21	15	^U	Negative acknowledge
SYN	26	22	16	^V	Synchronous idle
ETB	27	23	17	^W	End transmission block, not the same as EOT
CAN	30	24	18	^X	Cancel line , MPE echoes
EM	31	25	19	^Y	END of medium, Control-Y interrupt

Char	Oct	Dec	Hex	Control-key	Control Action
SUB	32	26	1a	^Z	Substitute
ESC	33	27	1b	^(	Escape, Next character is not echoed
FS	34	28	1c	^\	File separator
GS	35	29	1d	^)	Group separator
RS	36	30	1e	^^	Record separator, block-mode terminator
US	37	31	1f	^_	Unit separator

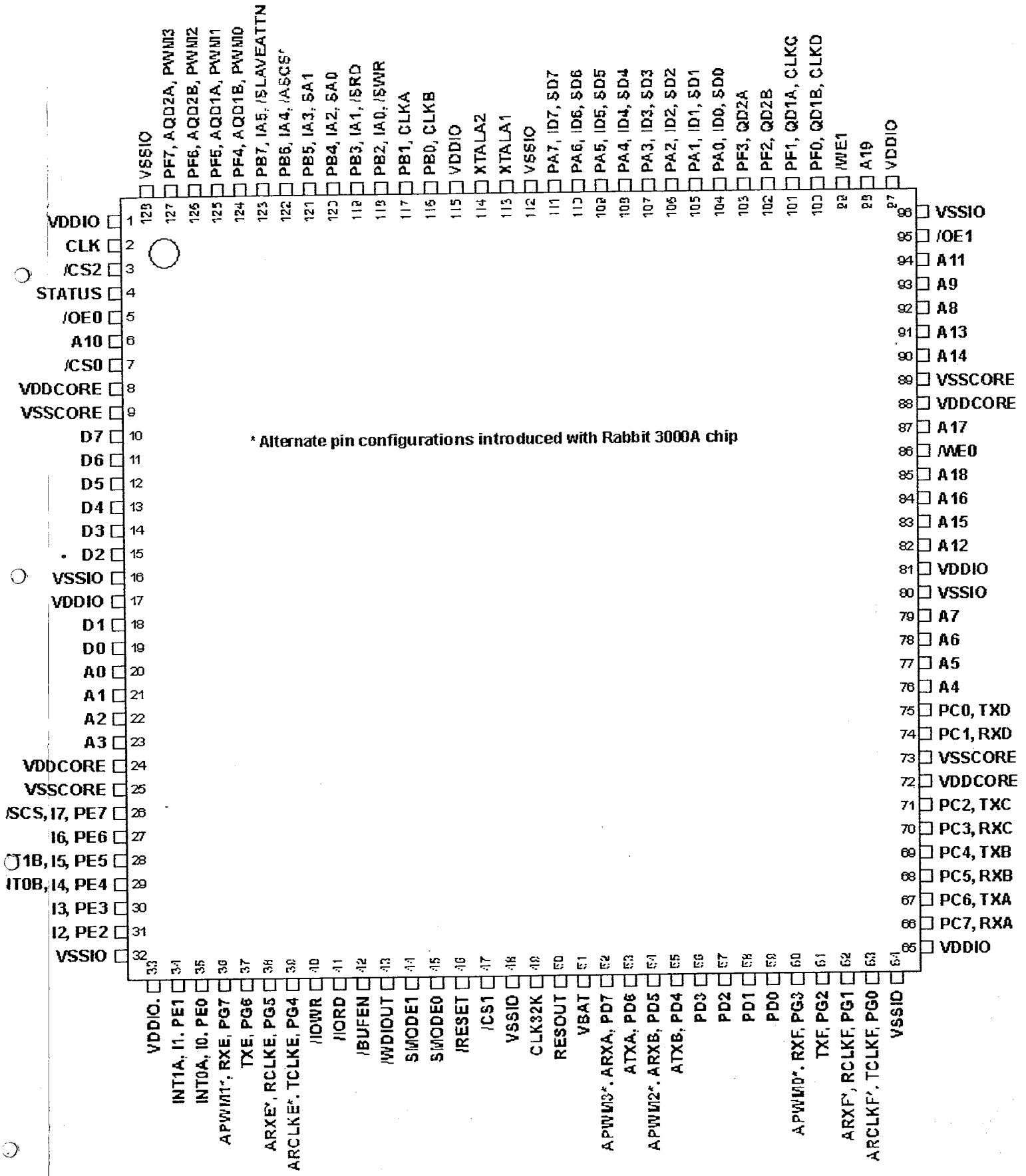
อักขระที่สามารถพิมพ์หรือแสดงผลได้ (Printing Characters)

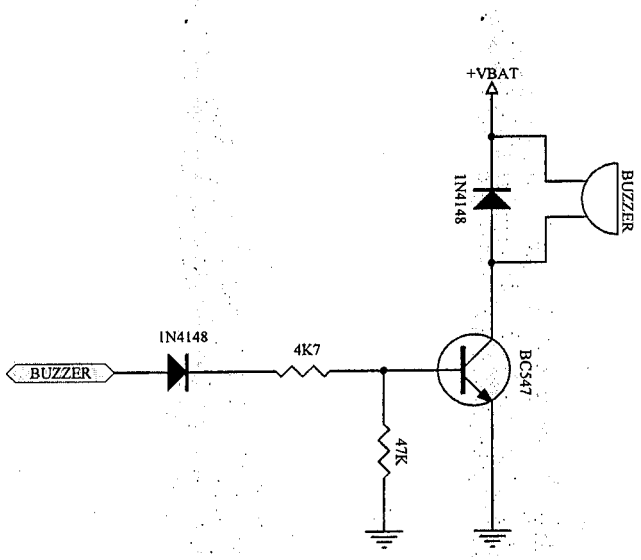
Char	Octal	Dec	Hex	Description
sp	40	32	20	Space
!	41	33	21	Exclamation mark
“	42	34	22	Quotation mark(" in HTML)
#	43	35	23	Cross hatch (number sign)
$	44	36	24	Dollar sign
%	45	37	25	Percent sign
&	46	38	26	Ampersand
-	47	39	27	Closing single quote (apostrophe)
(	50	40	28	Opening parentheses
)	51	41	29	Closing parentheses
*	52	42	2a	Asterisk (star, multiply)
+	53	43	2b	Plus
,	54	44	2c	Comma
.	55	45	2d	Hyphen, dash, minus
-	56	46	2e	Period
/	57	47	2f	Slant
0	60	48	30	Zero
1	61	49	31	One
2	62	50	32	Two
3	63	51	33	Three
4	64	52	34	Four

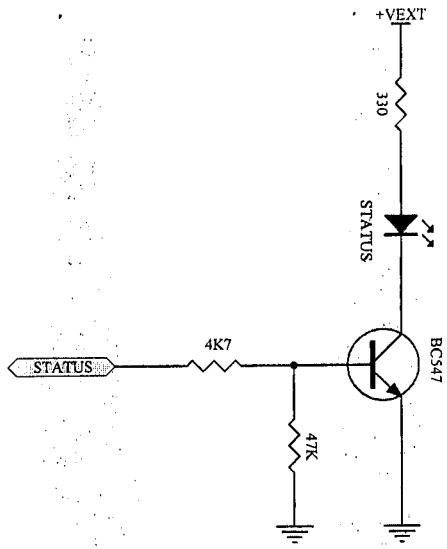
Char	Octal	Dec	Hex	Description
5	65	53	35	Five
6	66	54	36	Six
7	67	55	37	Seven
8	70	56	38	Eight
9	71	57	39	Nine
:	72	58	3a	Colon
;	73	59	3b	Semicolon
<	74	60	3c	Less then sign (< in HTML)
=	75	61	3d	Equals sign
>	76	62	3e	Greater than sign (> in HTML)
?	77	63	3f	Question mark
@	100	64	40	At-sign
A	101	65	41	Uppercase A
B	102	66	42	Uppercase B
C	103	67	43	Uppercase C
D	104	68	44	Uppercase D
E	105	69	45	Uppercase E
F	106	70	46	Uppercase F
G	107	71	47	Uppercase G
H	110	72	48	Uppercase H
I	111	73	49	Uppercase I
J	112	74	4a	Uppercase J
K	113	75	4b	Uppercase K
L	114	76	4c	Uppercase L
M	115	77	4d	Uppercase M
N	116	78	4e	Uppercase N
O	117	79	4f	Uppercase O
P	120	80	50	Uppercase P
Q	121	81	51	Uppercase Q
R	122	82	52	Uppercase R
S	123	83	53	Uppercase S
T	124	84	54	Uppercase T

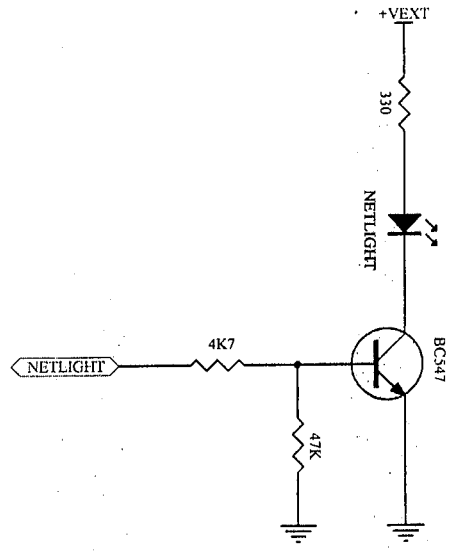
Char	Octal	Dec	Hex	Description
U	125	85	55	Uppercase U
V	126	86	56	Uppercase V
W	127	87	57	Uppercase W
X	130	88	58	Uppercase X
Y	131	89	59	Uppercase Y
Z	132	90	5a	Uppercase Z
[	133	91	5b	Opening square bracket
\	134	92	5c	Reverse slant (Backslash)
]	135	93	5d	Closing square bracket
^	136	94	5e	Caret (Circumflex)
_	137	95	5f	Underscore
‘	140	96	60	Opening single quote
a	141	97	61	Lowercase a
b	142	98	62	Lowercase b
c	143	99	63	Lowercase c
d	144	100	64	Lowercase d
e	145	101	65	Lowercase e
f	146	102	66	Lowercase f
g	147	103	67	Lowercase g
h	150	104	68	Lowercase h
i	151	105	69	Lowercase i
j	152	106	6a	Lowercase j
k	153	107	6b	Lowercase k
l	154	108	6c	Lowercase l
m	155	109	6d	Lowercase m
n	156	110	6e	Lowercase n
o	157	111	6f	Lowercase o
p	160	112	70	Lowercase p
q	161	113	71	Lowercase q
r	162	114	72	Lowercase r
s	163	115	73	Lowercase s
t	164	116	74	Lowercase t

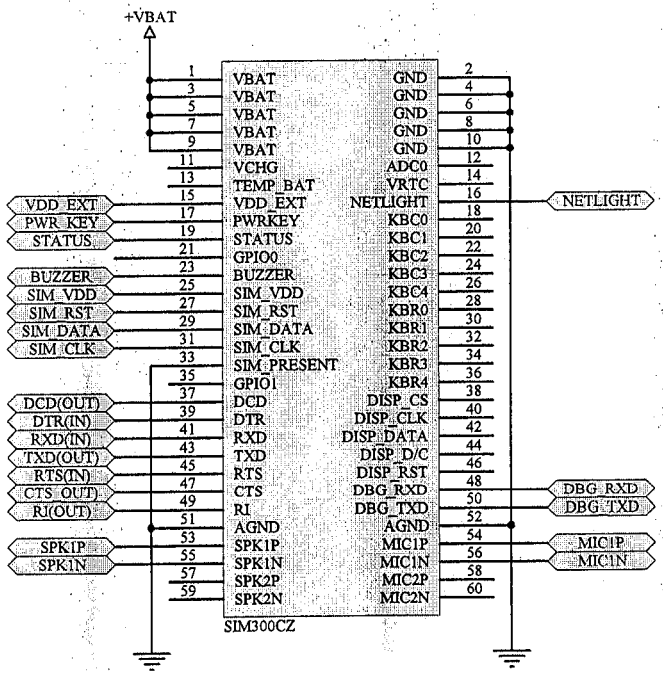
Char	Octal	Dec	Hex	Description
u	165	117	75	Lowercase u
v	166	118	76	Lowercase v
w	167	119	77	Lowercase w
x	170	120	78	Lowercase x
y	171	121	79	Lowercase y
z	172	122	7a	Lowercase z
{	173	123	7b	Opening curly brace
|	174	124	7c	Vertical line
}	175	125	7d	Closing curly brace
~	176	126	7e	Tilde (approximate)
DEL	177	127	7f	Delete (rubout),cross-hatch box

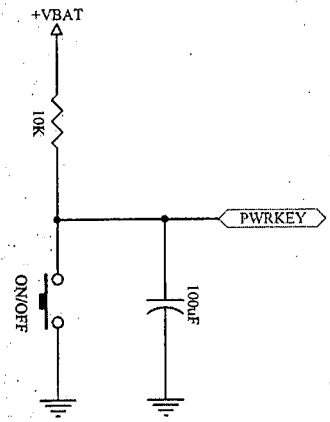

VDDIO 1
CLK 2
/CS2 3
STATUS 4
/OE0 5
A10 6
/CS0 7
VDDCORE 8
VSSCORE 9
D7 10
D6 11
D5 12
D4 13
D3 14
D2 15
VSSIO 16
VDDIO 17
D1 18
D0 19
A0 20
A1 21
A2 22
A3 23
VDDCORE 24
VSSCORE 25
/SCS, I7, PE7 26
I6, PE6 27
I1B, I5, PE5 28
INT0B, I4, PE4 29
I3, PE3 30
I2, PE2 31
VSSIO 32
VDDIO. 33
INT1A, I1. PE1 34
INT0A, I0. PE0 35
APWM1*, RXE, PG7 36
TXE, PG6 37
ARXE*, RCLKE, PG5 38
ARCLKE*. TCLKE, PG4 39
/IOWR 40
/IORD 41
/BUFEN 42
/WDIOUT 43
SMODE1 44
SMODE0 45
/RESET 46
/CS1 47
VSSID 48
CLK32K 49
RESOUT 50
VBAT 51
APWM3*. ARXA, PD7 52
ATXA, PD6 53
APWM2*. ARXB, PD5 54
ATXB, PD4 55
PD3 56
PD2 57
PD1 58
PD0 59
APWM0*. RXF, PG3 60
TXF, PG2 61
ARXF*, RCLKF, PG1 62
ARCLKF*, TCLKF, PG0 63
VSSID 64
VDDIO 65
PC7, RXA 66
PC6, TXA 67
PC5, RXB 68
PC4, TXB 69
PC3, RXC 70
PC2, TXC 71
VDDCORE 72
VSSCORE 73
PC1, RXD 74
PC0, TXD 75
A4 76
A5 77
A6 78
A7 79
VSSIO 80
VDDIO 81
A12 82
A15 83
A16 84
A18 85
/WE0 86
A17 87
VDDCORE 88
VSSCORE 89
A14 90
A13 91
A8 92
A9 93
A11 94
/OE1 95
VSSIO 96
VDDIO 97
A19 98
/WE1 99
PF0, QD1B, CLKD 100
PF1, QD1A, CLKC 101
PF2, QD2B 102
PF3, QD2A 103
PA0, ID0, SD0 104
PA1, ID1, SD1 105
PA2, ID2, SD2 106
PA3, ID3, SD3 107
PA4, ID4, SD4 108
PA5, ID5, SD5 109
PA6, ID6, SD6 110
PA7, ID7, SD7 111
VSSIO 112
XTALA1 113
XTALA2 114
VDDIO 115
PB0, CLKB 116
PB1, CLKA 117
PB2, IA0, /SWR 118
PB3, IA1, /SRD 119
PB4, IA2, SA0 120
PB5, IA3, SA1 121
PB6, IA4, /ASCS* 122
PB7, IA5, /SLAVEATTN 123
PF4, AQD1B, PWM0 124
PF5, AQD1A, PWM1 125
PF6, AQD2B, PWM2 126
PF7, AQD2A, PWM3 127
VSSIO 128
* Alternate pin configurations introduced with Rabbit 3000A chip

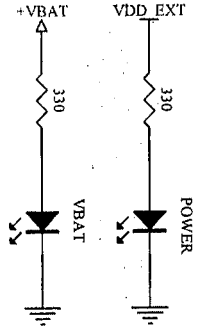




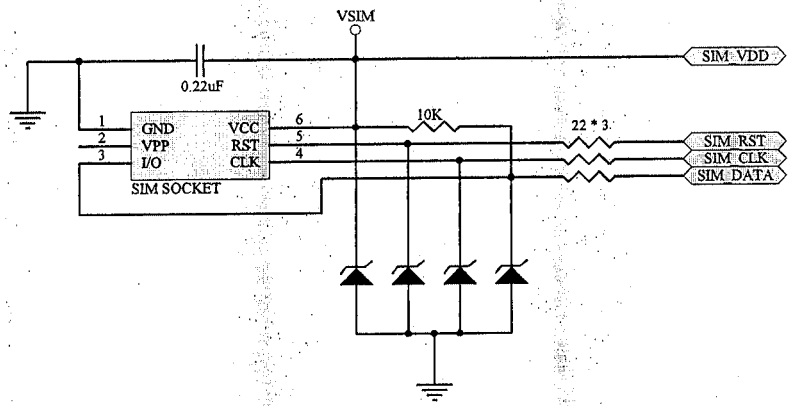


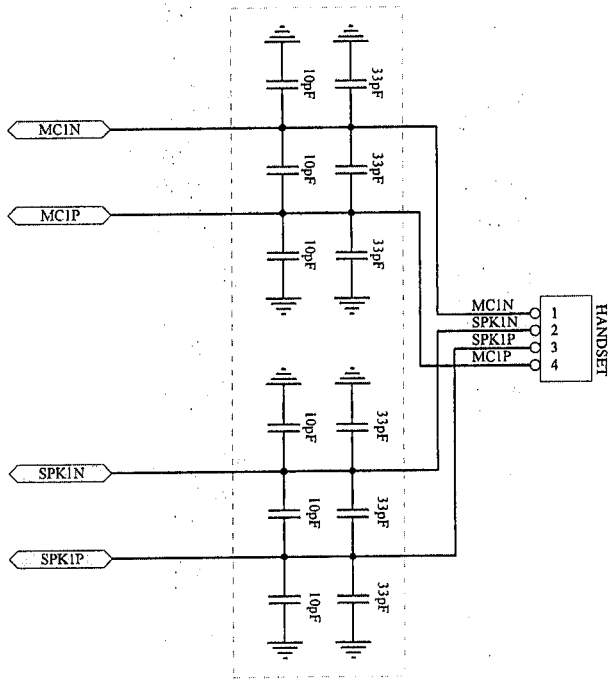




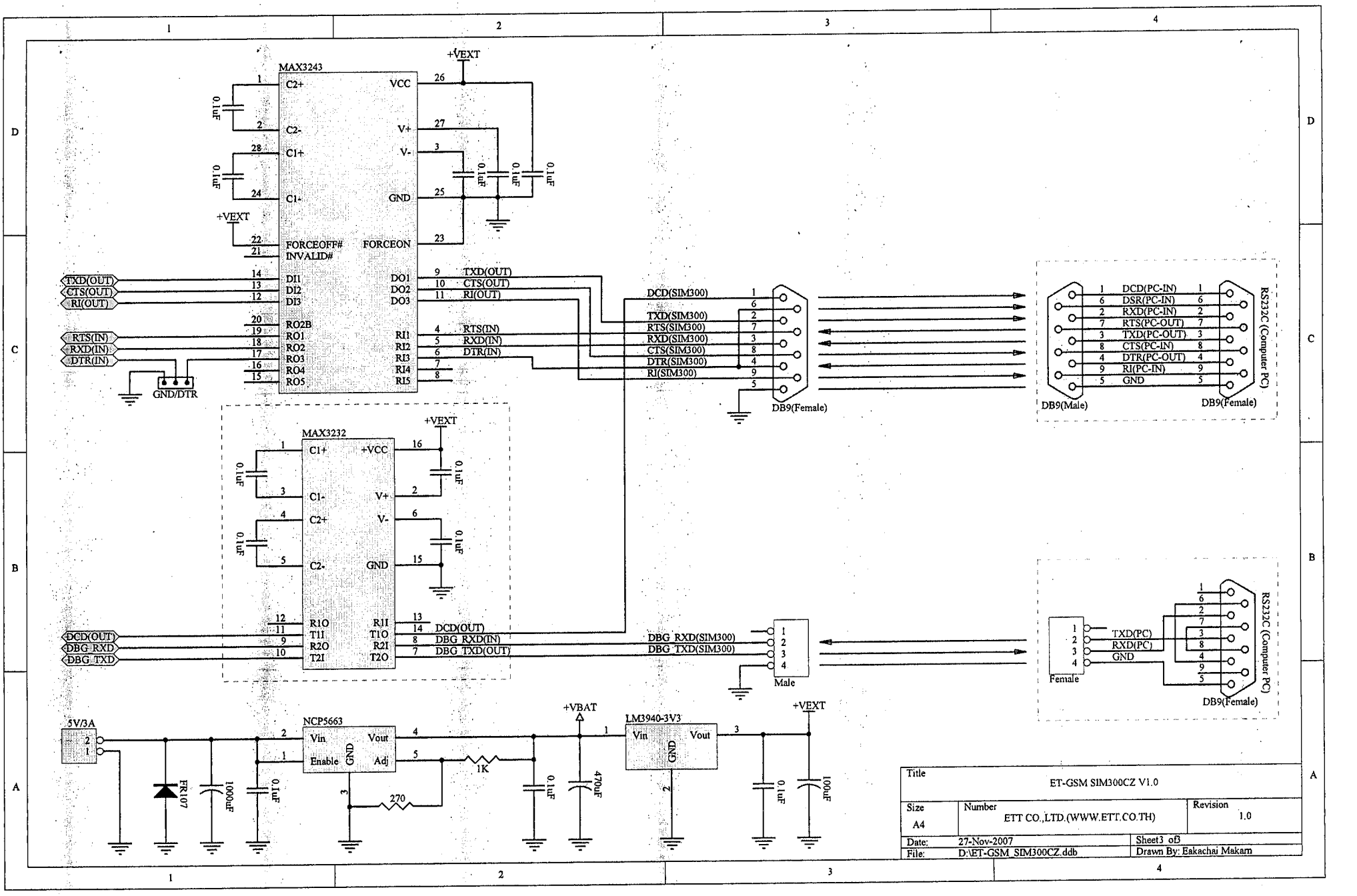


Title	ET-GSM SIM300CZ V1.0		
Size A4	Number ETT CO.,LTD.(WWW.ETT.CO.TH)		Revision 1.0
Date:	27-Nov-2007	Sheet1 of3	
File:	D:\ET-GSM_SIM300CZ.ddb	Drawn By: Eakachai Makarn	





Title	ET-GSM SIM300CZ V1.0	
Size: A4	Number: ETT CO.,LTD.(WWW.ETT.CO.TH)	Revision: 1.0
Date: 27-Nov-2007		Sheet2 of3
File: D:\ET-GSM_SIM300CZ.ddb		Drawn By: Eakachai Makarn


MAX3243
+VEXT
C2+
C2-
C1+
C1-
VCC
V+
V-
GND
FORCEOFF#
INVALID#
FORCEON
0.1uF
DI1
DI2
DI3
DO1
DO2
DO3
TXD(OUT)
CTS(OUT)
RI(OUT)
RO2B
RO1
RO2
RO3
RO4
RO5
RI1
RI2
RI3
RI4
RI5
RTS(IN)
RXD(IN)
DTR(IN)
GND/DTR
DCD(SIM300)
TXD(SIM300)
RTS(SIM300)
RXD(SIM300)
CTS(SIM300)
DTR(SIM300)
RI(SIM300)
DB9(Female)
DCD(PC-IN)
DSR(PC-IN)
RXD(PC-IN)
RTS(PC-OUT)
TXD(PC-OUT)
CTS(PC-IN)
DTR(PC-OUT)
RI(PC-IN)
GND
DB9(Male)
RS232C (Computer PC)
MAX3232
+VCC
R1O
T1I
R2O
T2I
R1I
T1O
R2I
T2O
DCD(OUT)
DBG_RXD
DBG_TXD
DBG_RXD(IN)
DBG_TXD(OUT)
DBG_RXD(SIM300)
DBG_TXD(SIM300)
Male
Female
TXD(PC)
RXD(PC)
5V/3A
NCP5663
Vin
Vout
Enable
Adj
1K
270
+VBAT
LM3940-3V3
FR107
1000uF
470uF
100uF
Title
ET-GSM SIM300CZ V1.0
Size
A4
Number
ETT CO.,LTD.(WWW.ETT.CO.TH)
Revision
1.0
Date:
27-Nov-2007
Sheet3 of3
File:
D:\ET-GSM_SIM300CZ.ddb
Drawn By: Eakachai Makarn

SILICON RECTIFIER

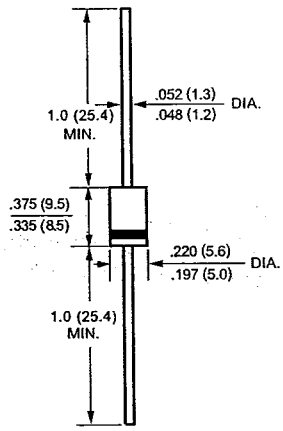
VOLTAGE RANGE - 50 to 1000 Volts CURRENT - 3.0 Amperes

MECHANICAL DATA

* Case: Molded plastic
* Epoxy: UL 94V-0 rate flame retardant
* Lead: MIL-STD-202E, Method 208 guaranteed
* Polarity: Color band denotes cathode end
* Mounting position: Any
* Weight: 1.18 grams

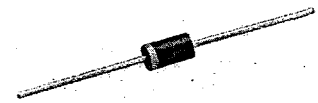
FEATURES

* Low cost
* Low leakage
* Low forward voltage drop
* High current capability



Dimensions in inches and (millimeters)

DO-27



MAXIMUM RATINGS AND ELECTRICAL CHARACTERISTICS

Ratings at 25℃ ambient temperature unless otherwise specified.Single phase, half wave, 60 Hz, resistive or inductive load.
For capacitive load, derate current by 20%.

PARAMETER	SYMBOL	1N5400	1N5401	1N5402	1N5404	1N5406	1N5407	1N5408	UNITS
Maximum Recurrent Peak Reverse Voltage	V_{RRM}	50	100	200	400	600	800	1000	Volts
Maximum RMS Voltage	V_{RMS}	35	70	140	280	420	560	700	Volts
Maximum DC Blocking Voltage	V_{DC}	50	100	200	400	600	800	1000	Volts
Maximum Average Forward Rectified Current .375*(9.5mm) lead length at TL = 105℃	I_O	3.0							Amps
Peak Forward Surge Current 8.3 ms single half sine-wave Superimposed on rated load (JEDEC Method)	I_{FSM}	200							Amps
Maximum Instantaneous Forward Voltage at 3.0A DC	V_F	1.1							Volts
Maximum DC Reverse Current at Rated DC Blocking Voltage @T_A = 25℃	I_R	5.0							uAmps
Maximum DC Reverse Current at Rated DC Blocking Voltage @T_A = 100℃	I_R	500							uAmps
Maximum Full Load Reverse Current Average, Full Cycle .375*(9.5mm) lead length at TL = 75℃	I_R	30							uAmps
Typical Junction Capacitance (Note)	C_J	40							pF
Typical Thermal Resistance	RΘJA	30							℃/W
Operating and Storage Temperature Range	T_J,T_{STG}	-65 to +175							℃

NOTES : Measured at 1 MHz and applied reverse voltage of 4.0 volts

Wing Shing Computer Components Co., (H.K.)Ltd. Tel:(852)2341 9276 Fax:(852)2797 8153
Homepage: http://www.wingshing.com E-mail: wsccltd@hkstar.com

RATING AND CHARACTERISTIC CURVES

FIG. 1 - TYPICAL FORWARD CURRENT DERATING CURVE

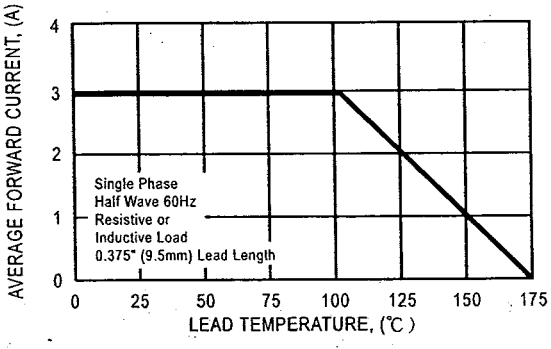


FIG. 2 - MAXIMUM NON-REPETITIVE FORWARD SURGE CURRENT

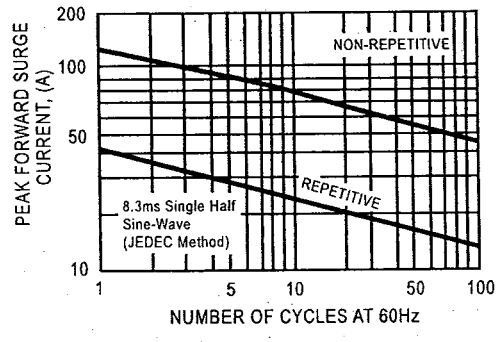


FIG. 3 - TYPICAL INSTANTANEOUS FORWARD VOLTAGE, (V)

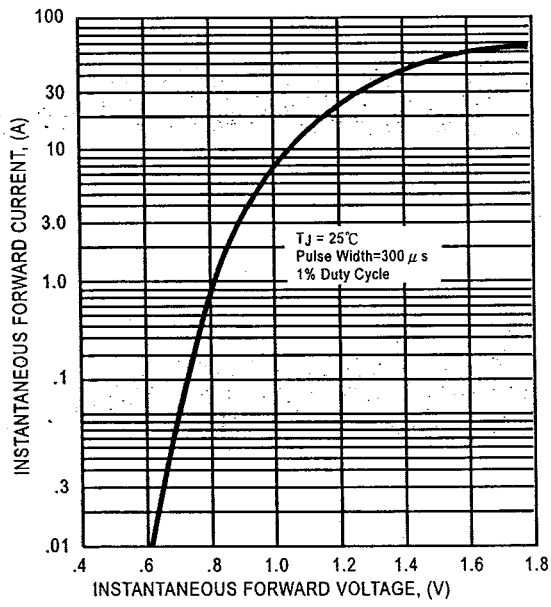


FIG. 4 - TYPICAL JUNCTION CAPACITANCE

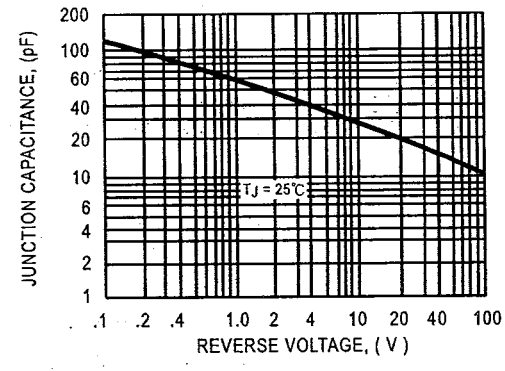
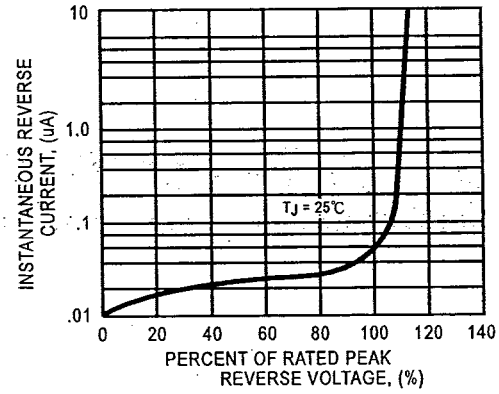


FIG. 5 - TYPICAL REVERSE CHARACTERISTICS



Wing Shing Computer Components Co., (H.K.)Ltd.
Homepage: http://www.wingshing.com
Tel:(852)2341 9276 Fax:(852)2797 8153
E-mail: wsccltd@hkstar.com

FAIRCHILD
SEMICONDUCTOR™

GlobalOptoisolator™

VDE | UL | CSA | SETI | SEMKO | DEMKO | NEMKO | BABT

4N25
4N26
4N27
4N28

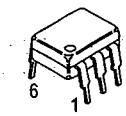
6-Pin DIP Optoisolators Transistor Output

The 4N25, 4N26, 4N27 and 4N28 devices consist of a gallium arsenide infrared emitting diode optically coupled to a monolithic silicon phototransistor detector.

- Most Economical Optoisolator Choice for Medium Speed, Switching Applications
- Meets or Exceeds All JEDEC Registered Specifications
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

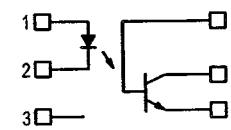
Applications

- General Purpose Switching Circuits
- Interfacing and coupling systems of different potentials and impedances
- I/O Interfacing
- Solid State Relays



STANDARD THRU HOLE

SCHEMATIC



PIN 1. LED ANODE
2. LED CATHODE
3. N.C.
4. EMITTER
5. COLLECTOR
6. BASE

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

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
LED Power Dissipation @ T_A = 25°C with Negligible Power in Output Detector Derate above 25°C	P_D	120 1.41	mW mW/°C
OUTPUT TRANSISTOR			
Collector–Emitter Voltage	V_{CEO}	30	Volts
Emitter–Collector Voltage	V_{ECO}	7	Volts
Collector–Base Voltage	V_{CBO}	70	Volts
Collector Current — Continuous	I_C	150	mA
Detector Power Dissipation @ T_A = 25°C with Negligible Power in Input LED Derate above 25°C	P_D	150 1.76	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage(1) (Peak ac Voltage, 60 Hz, 1 sec Duration)	V_{ISO}	7500	Vac(pk)
Total Device Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.94	mW mW/°C
Ambient Operating Temperature Range	T_A	–55 to +100	°C
Storage Temperature Range	T_{stg}	–55 to +150	°C
Soldering Temperature (10 sec, 1/16" from case)	T_L	260	°C

1. Isolation surge voltage is an internal device dielectric breakdown rating. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)(1)

Characteristic		Symbol	Min	Typ(1)	Max	Unit
INPUT LED						
Forward Voltage (I_F = 10 mA)	T_A = 25°C T_A = –55°C T_A = 100°C	V_F	— — —	1.15 1.3 1.05	1.5 — —	Volts
Reverse Leakage Current (V_R = 3 V)		I_R	—	—	100	µA
Capacitance (V = 0 V, f = 1 MHz)		C_J	—	18	—	pF
OUTPUT TRANSISTOR						
Collector–Emitter Dark Current (V_{CE} = 10 V, T_A = 25°C	4N25,26,27 4N28	I_{CEO}	— —	1 1	50 100	nA
(V_{CE} = 10 V, T_A = 100°C)	All Devices	I_{CEO}	—	1	—	µA
Collector–Base Dark Current (V_{CB} = 10 V)		I_{CBO}	—	0.2	—	nA
Collector–Emitter Breakdown Voltage (I_C = 1 mA)		$V_{(BR)CEO}$	30	45	—	Volts
Collector–Base Breakdown Voltage (I_C = 100 µA)		$V_{(BR)CBO}$	70	100	—	Volts
Emitter–Collector Breakdown Voltage (I_E = 100 µA)		$V_{(BR)ECO}$	7	7.8	—	Volts
DC Current Gain (I_C = 2 mA, V_{CE} = 5 V)		h_{FE}	—	500	—	—
Collector–Emitter Capacitance (f = 1 MHz, V_{CE} = 0)		C_{CE}	—	7	—	pF
Collector–Base Capacitance (f = 1 MHz, V_{CB} = 0)		C_{CB}	—	19	—	pF
Emitter–Base Capacitance (f = 1 MHz, V_{EB} = 0)		C_{EB}	—	9	—	pF
COUPLED						
Output Collector Current (I_F = 10 mA, V_{CE} = 10 V)	4N25,26 4N27,28	I_C (CTR)(2)	2 (20) 1 (10)	7 (70) 5 (50)	— —	mA (%)
Collector–Emitter Saturation Voltage (I_C = 2 mA, I_F = 50 mA)		$V_{CE(sat)}$	—	0.15	0.5	Volts
Turn–On Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)(3)		t_{on}	—	2.8	—	µs
Turn–Off Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)(3)		t_{off}	—	4.5	—	µs
Rise Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)(3)		t_r	—	1.2	—	µs
Fall Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)(3)		t_f	—	1.3	—	µs
Isolation Voltage (f = 60 Hz, t = 1 sec)(4)		V_{ISO}	7500	—	—	Vac(pk)
Isolation Resistance (V = 500 V)(4)		R_{ISO}	10^{11}	—	—	Ω
Isolation Capacitance (V = 0 V, f = 1 MHz)(4)		C_{ISO}	—	0.2	—	pF

1. Always design to the specified minimum/maximum electrical limits (where applicable).
2. Current Transfer Ratio (CTR) = I_C/I_F x 100%.
3. For test circuit setup and waveforms, refer to Figure 11.
4. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

TYPICAL CHARACTERISTICS

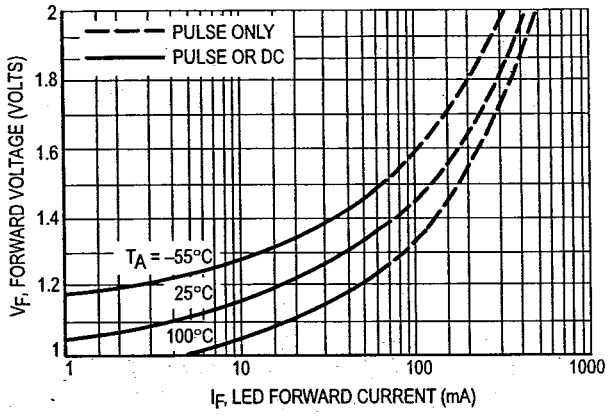


Figure 1. LED Forward Voltage versus Forward Current

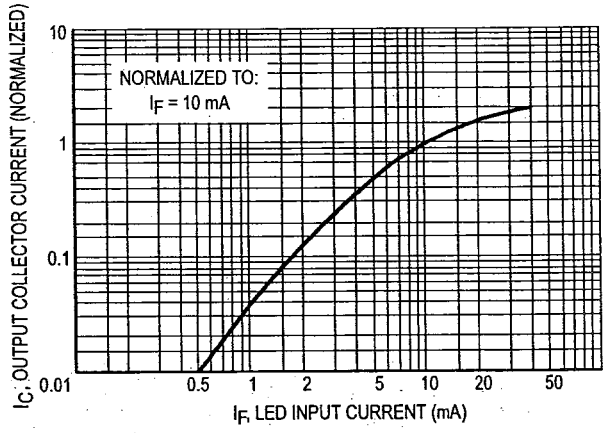


Figure 2. Output Current versus Input Current

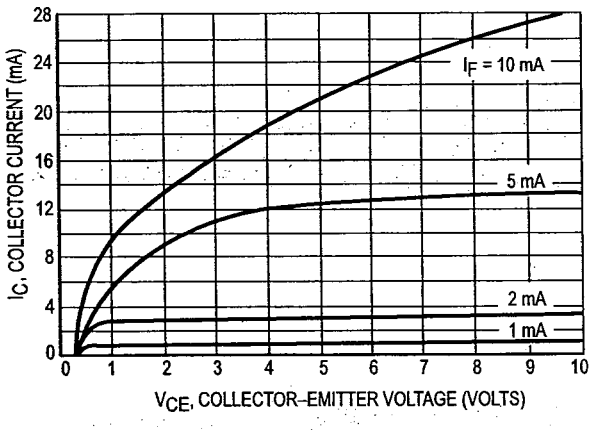


Figure 3. Collector Current versus Collector–Emitter Voltage

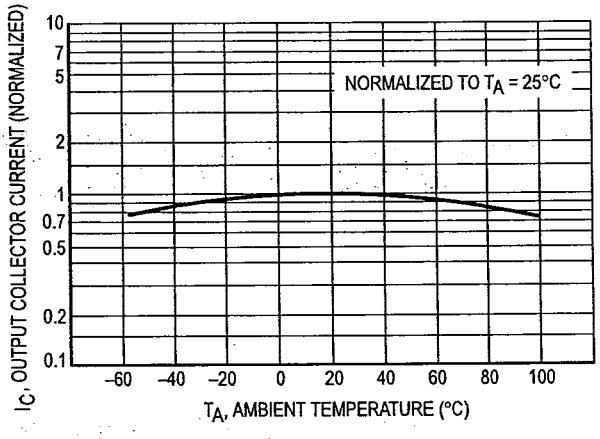


Figure 4. Output Current versus Ambient Temperature

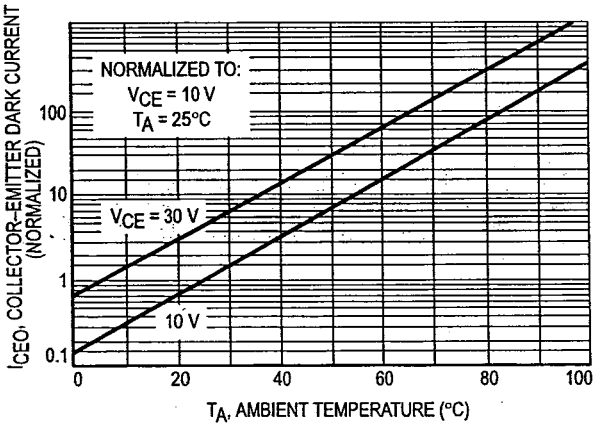


Figure 5. Dark Current versus Ambient Temperature

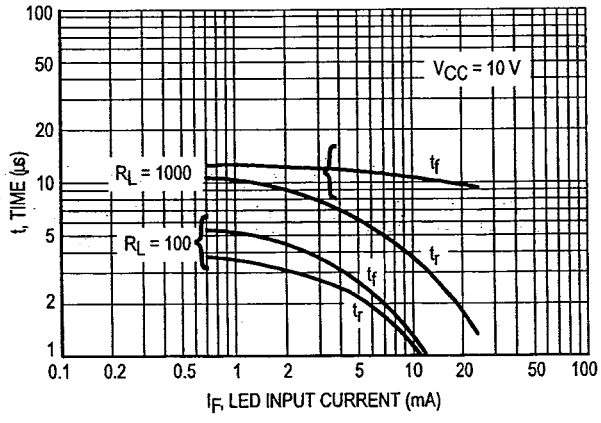


Figure 6. Rise and Fall Times (Typical Values)

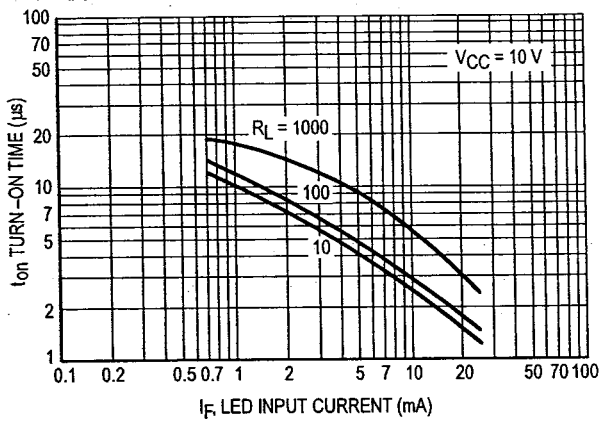


Figure 7. Turn–On Switching Times (Typical Values)

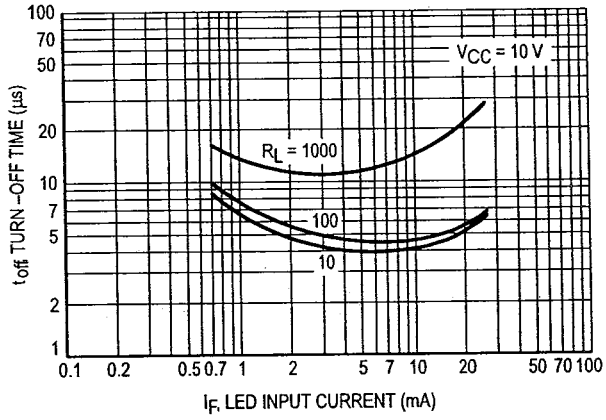


Figure 8. Turn–Off Switching Times (Typical Values)

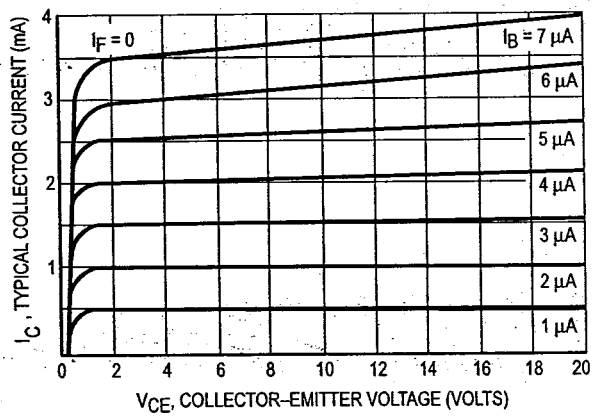


Figure 9. DC Current Gain (Detector Only)

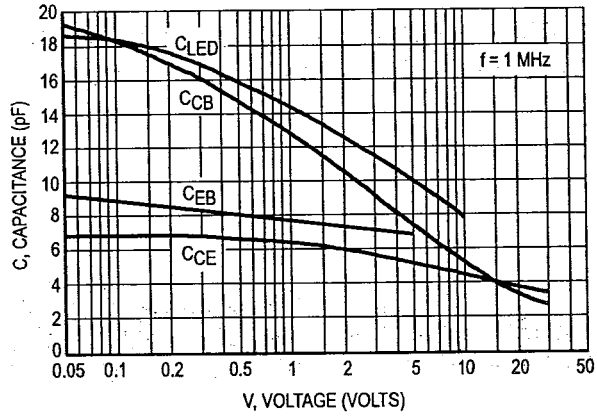


Figure 10. Capacitances versus Voltage

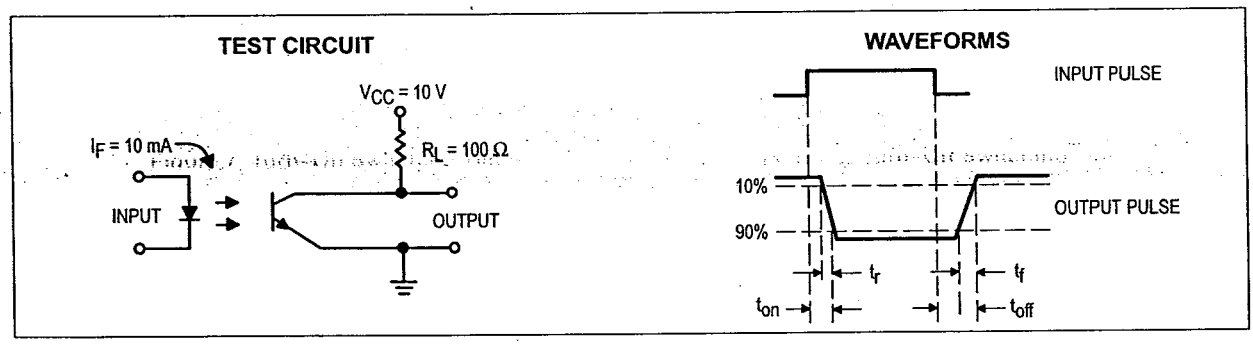
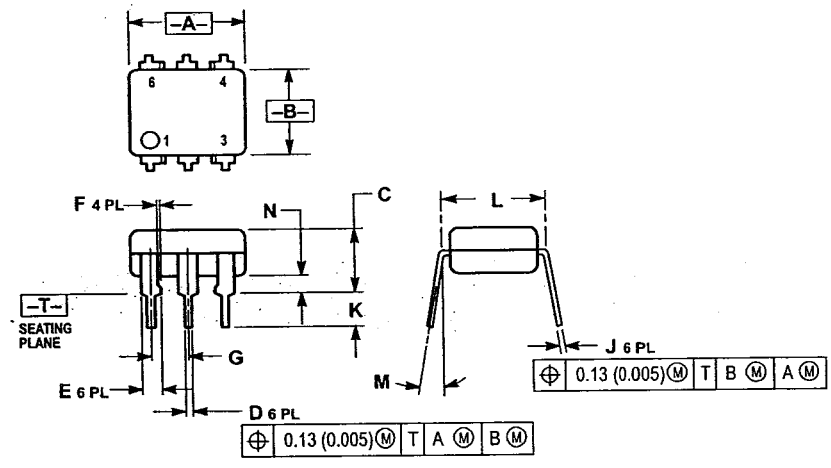


Figure 11. Switching Time Test Circuit and Waveforms

PACKAGE DIMENSIONS

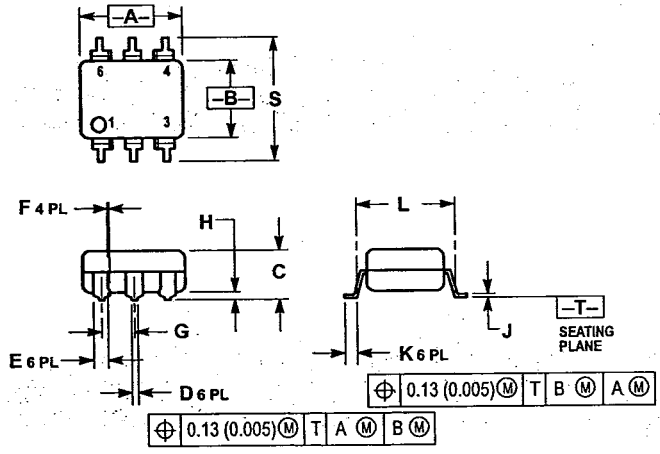


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
J	0.008	0.012	0.21	0.30
K	0.100	0.150	2.54	3.81
L	0.300 BSC		7.62 BSC	
M	0°	15°	0°	15°
N	0.015	0.100	0.38	2.54

STYLE 1:
PIN 1. ANODE
2. CATHODE
3. NC
4. EMITTER
5. COLLECTOR
6. BASE

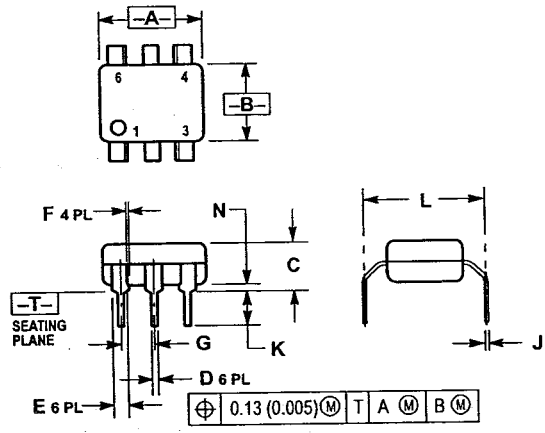
THRU HOLE



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
H	0.020	0.025	0.51	0.63
J	0.008	0.012	0.20	0.30
K	0.006	0.035	0.16	0.88
L	0.320 BSC		8.13 BSC	
S	0.332	0.390	8.43	9.90

SURFACE MOUNT



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

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
J	0.008	0.012	0.21	0.30
K	0.100	0.150	2.54	3.81
L	0.400	0.425	10.16	10.80
N	0.015	0.040	0.38	1.02

0.4" LEAD SPACING

FAIRCHILD
SEMICONDUCTOR™

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.

2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

เอกสารอ้างอิง

1. Z-World Inc, RabbitCore RCM2200 : User's Manual, 2001.
2. Z-World Inc, RabbitCore RCM2200 : Getting Started Manual, 2001.
3. Z-World Inc, Dynamic C : TCP/IP User's Manual, 2001.
4. Z-World Inc, An Introduction to TCP/IP For Embedded System Designer, 2001.
5. Z-World Inc, TCP/IP Development Kit : Getting Started Manual, 2000.
6. Z-World Inc, Rabbit 2000 Microprocessor : User's Manual, 1999.
7. Z-World Inc, Dynamic C Premier : User's Manual, 1999.
8. Z-World Inc, www.zworld.com.
9. Rabbit Semiconductor Inc, www.rabbitsemiconductor.com.
10. ThaiIO Website, www.thaiio.com.
11. ETT CO., LTD. , ET-GSM SIM300CZ: User's Manual, 2007
12. คู่มือการเขียนโปรแกรมภาษา C ฉบับเริ่มต้น. นนทบุรี:อินโฟเพรส, 2545.