ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

Automatic Telephone Recorder

ชื่อนักศึกษา

1. นายกอเฉ็ม เตาวะโต	รหัสประจำตัว	45035250
2. นายเดชชาติ สุขจิต	รหัสประจำตัว	45035261
3. นายปรมี แสงเพ็ชร	รหัสประจำตัว	45035267
4. นายปรีชา คงแก้ว	รหัสประจำตัว	45035269

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา อาจารย์สมชาย หมื่นสายญาติ

อาจารย์ที่ปรึกษาร่วม 1. อาจารย์ปิยะ ศุภวราสุวัฒน์

2. อาจารย์พงษ์เกียรติ เชษฐพิทักษ์สกุล

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์สุระชัย พิมพ์สาลี	
2. อาจารย์โกศล ตราชู	
3. อาจารย์สมชาย หมื่นสายญาติ	
4. อาจารย์ปิยะ ศุภวราสุวัฒน์	
5. ผศ.พีระวุฒิ สุวรรณจันทร์	

วัน/เดือน/ปีที่สอบ วันพุธที่ 31 มีนาคม พ.ศ. 2547 เวลา 15.00 น.

สถานที่สอบ ห้อง ค.311 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.สุรสิทธิ์ ราตรี)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่.....1.....เดือน.....ตุลาคม.....พ.ศ.....๒๕๔๗.....

<BT4620322>

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ปริญญานิพนธ์

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

AUTOMATIC TELEPHONE RECORDER

นายกอเฉ็ม เตาะโต
นายเดชชาติ สุขจิต
นายปรมี แสงเพ็ชร
นายปรีชา คงแก้ว

เลขหมู่.....
เลขทะเบียน 51034
วัน,เดือน,ปี 29 มิ.ย. 2547

.b.....
.i.....

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

ปริญญานิพนธ์

เรื่อง เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

Automatic Telephone Recorder

วัตถุประสงค์

1. เพื่อศึกษาการทำงานของระบบโทรศัพท์ สัญญาณต่างๆ ของโทรศัพท์ และไมโครคอนโทรลเลอร์
2. เพื่อออกแบบโปรแกรมการทำงานและวงจรควบคุมการทำงานต่างๆ ได้
3. เพื่อสร้างเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติได้
4. เพื่อทดลองและทดสอบการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติได้
5. เพื่อนำเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติไปใช้งานจริงได้

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เรื่องการทำงานของระบบโทรศัพท์ สัญญาณต่างๆ ของโทรศัพท์ และไมโครคอนโทรลเลอร์
2. ได้โปรแกรมการทำงานและวงจรควบคุมการทำงานต่างๆ
3. ได้เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ
4. ได้ผลการทดลองและทดสอบการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ
5. นำเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติไปใช้งาน

ชื่อหัวข้อ	เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	
ชื่อนักศึกษา	นายกอเฉ็ม	เตาวะโต
	นายเดชชาติ	สุขจิต
	นายปรมี	แสงเพ็ชร
	นายปรีชา	คงแก้ว
อาจารย์ที่ปรึกษา	ดร.สมชาย	หมื่นสายญาติ
อาจารย์ที่ปรึกษาร่วม	อาจารย์ปิยะ	ศุภวราสุวัฒน์
	อาจารย์พงษ์เกียรติ	เชษฐพิทักษ์สกุล
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต	
สาขาวิชา	วิศวกรรมโทรคมนาคม	
ปีการศึกษา	2546	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบ และสร้างเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ โดยการนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้งานกับระบบโทรศัพท์ และวงจรตรวจจับสัญญาณโทรศัพท์ ในส่วนของการโทรเข้าจะสามารถรับสายเรียกเข้าได้ปกติ แต่ในส่วนของการโทรออกจะมีการตรวจสอบรหัสผ่านก่อนการโทรออก เมื่อมีการรับสายจากปลายทาง เครื่องจะทำการบันทึกข้อมูลการโทรออก ข้อมูลที่ถูกบันทึกสามารถเชื่อมต่อออกไปยังคอมพิวเตอร์ และพิมพ์ข้อมูลออกมาได้ ในส่วนหมายเลขพิเศษที่ไม่คิดค่าบริการสามารถโทรออกได้โดยไม่ต้องผ่านระบบรหัสผ่าน การทำงานต่างๆ จะแสดงผลด้วยจอแอลซีดี 20 ตัวอักษร 2 บรรทัด กำหนดรหัสผ่านได้จำนวน 4 หลัก สำหรับผู้ใช้ 20 รหัสผ่าน มีระบบรักษาความปลอดภัย เมื่อมีการกดรหัสผ่านผิด 3 ครั้ง ให้เครื่องทำงานเข้าสู่ระบบการป้องกันการโทรออกอัตโนมัติ ไม่สามารถใช้งานได้จนกว่าระบบกลับคืนสู่การทำงานปกติ เมื่อครบเวลา 20 นาทีที่กำหนดไว้ หรือเมื่อมีการแก้ไขโดยผู้ดูแลระบบ

Thesis Title	Automatic Telephone Recorder	
Students	Mr.Kochem	Taowato
	Mr.Detchart	Sukjit
	Mr.Porramee	Sangpetch
	Mr.Preecha	Kongkaew
Advisor	Dr.Somchai	Maunsaiyat
Co-Advisors	Mr.Piya	Supavarasuwat
	Mr.Pongkiat	Chedpitaksakul
Education Level	Bachelor of Science in Industrial Education	
Program in	Telecommunication Engineering	
Academic Year	2003	

ABSTRACT

This thesis presented a design and development of the automatic telephone recorder. The applications of the microcontroller, the telephone system and, the telephone signal check circuit were used in the recorder. For the incoming call, the recorder worked as usual, and for the outgoing call, the four-digit password was checked first, and when the destination phone was picked up, the data of the outgoing call were recorded. Those data could be printed out though the computer. The setting emergency call numbers (free service call) could be used directly with no password required. A LCD with two lines of 20 characters was used as the display of the recorder. The maximum users for the recorder were 20. Three times of the wrong password could be pressed, after that the outgoing call protection (safety system) would operate. In that status the machine cannot be used for setting 20 minutes or before resetting the machine.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี เนื่องมาจากได้รับคำปรึกษาและสนับสนุนจากคณาจารย์หลายท่าน อีกทั้งอาจารย์ เพื่อนๆ และผู้จัดทำปริญญานิพนธ์ฉบับนี้ได้ช่วยให้คำแนะนำพร้อมกับแก้ไขปัญหาและอุปสรรคต่างๆ ในช่วงเวลาของการสร้างปริญญานิพนธ์ฉบับนี้ จนสามารถทำงานได้อย่างสมบูรณ์และทำงานได้จริง

ขอขอบพระคุณอาจารย์ทุกท่านในคณะครุศาสตร์อุตสาหกรรม ผู้ซึ่งประสิทธิ์ประสาทวิชาความรู้และให้คำปรึกษา นับตั้งแต่ที่ได้เข้ามาศึกษาในสถาบันแห่งนี้ ขอขอบพระคุณอาจารย์และคณะกรรมการทุกท่าน ที่ได้สละเวลาอันมีค่าในการตรวจสอบและให้คำแนะนำในการจัดทำเนื้อหาในปริญญานิพนธ์ ขอขอบพระคุณอาจารย์สมชาย หมื่นสายญาติ ที่ได้กรุณาให้คำปรึกษาในการเขียนบทคัดย่อภาษาอังกฤษ อาจารย์ปิยะ ศุภวราสุวัฒน์และอาจารย์อมรชัย ชัยชนะ ที่คอยให้คำปรึกษาทุกเรื่อง ขอขอบพระคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม ที่ให้สำหรับการค้นคว้าหาข้อมูลต่างๆ เพื่อใช้ประกอบการทำปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อและคุณแม่ที่ให้การอุปการะและให้กำลังใจตลอดจนขอขอบพระคุณพี่น้องและเพื่อนๆ ที่ให้คำปรึกษาและช่วยเหลือ ในการจัดทำปริญญานิพนธ์ฉบับนี้ให้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญรูป	IX
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 ขีดความสามารถของโครงงาน	1
1.3 เนื้อหาโดยสังเขป	1
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 กล่าวนำ	3
2.2 วงจรของเครื่องรับโทรศัพท์	3
2.2.1 วงจรกระดิ่ง	3
2.2.2 วงจรปากพูด-หูฟัง	3
2.2.3 วงจรเข้ารหัสตัวเลขชนิดปุ่มกด	3
2.3 สัญญาณที่รับส่งระหว่างผู้เช่าและชุมสาย	4
2.3.1 สัญญาณที่ส่งจากผู้เช่าไปยังชุมสาย	4
2.3.2 สัญญาณที่ส่งจากชุมสายมายังผู้เช่า	4
2.4 โทรศัพท์แบบความถี่คู่	4
2.5 ไอซีถอดรหัสความถี่โทรศัพท์	5
2.5.1 คุณสมบัติของไอซี MT8870	5
2.5.2 การนำไอซี MT8870 ไปใช้งานด้านอื่นๆ	6
2.5.3 โครงสร้างภายในของไอซี MT8870	6
2.6 ไอซีฐานเวลา	7

สารบัญ (ต่อ)

เรื่อง	หน้า
2.7 ไมโครคอนโทรลเลอร์	8
2.8 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม	10
2.9 รูปแบบของการสื่อสารข้อมูลอนุกรม	11
2.10 การใช้งานโปรแกรมวิชวลเบสิค	12
บทที่ 3 การออกแบบ การสร้าง และการทำงาน	16
3.1 แผนผังการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	16
3.2 การออกแบบวงจร	17
3.2.1 วงจรแหล่งจ่ายไฟ	17
3.2.2 วงจรทวีแรงดัน	17
3.2.3 วงจรถอดรหัสสัญญาณความถี่คู่	18
3.2.4 วงจรตรวจจับสัญญาณกระดิ่ง	19
3.2.5 วงจรตรวจจับการยกหูโทรศัพท์	20
3.2.6 วงจรตัดต่อคู่สายโทรศัพท์	21
3.2.7 วงจรตั้งเวลาระงับการโทรออกชั่วคราว	21
3.2.8 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	22
3.2.9 วงจรไมโครคอนโทรลเลอร์	23
3.3 ส่วนของซอฟต์แวร์	24
3.4 การทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	25
3.4.1 กรณีการโทรออก	25
3.4.2 กรณีการโทรเข้า	25
3.4.3 กรณีของผู้ดูแลระบบ	25
3.4.4 กรณีการเชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์	25
บทที่ 4 การทดลองและผลการทดลอง	26
4.1 กล่าวนำ	26
4.2 การทดลองส่วนของวงจรถอดรหัสสัญญาณความถี่คู่	26
4.2.1 ลำดับขั้นการทดลอง	26
4.2.2 ผลการทดลอง	26

สารบัญ (ต่อ)

เรื่อง	หน้า
4.3 การทดลองส่วนของวงจรตรวจจับสัญญาณกระดิ่ง	28
4.3.1 ลำดับขั้นการทดลอง	28
4.3.2 ผลการทดลอง	28
4.4 การทดลองส่วนของวงจรตรวจจับการยกหูโทรศัพท์	28
4.4.1 ลำดับขั้นการทดลอง	28
4.4.2 ผลการทดลอง	29
4.5 การทดลองส่วนของวงจรตัดต่อคู่สายโทรศัพท์	29
4.5.1 ลำดับขั้นการทดลอง	29
4.5.2 ผลการทดลอง	29
4.6 การทดลองส่วนของวงจรตั้งเวลาระงับการโทรออกชั่วคราว	30
4.6.1 ลำดับขั้นการทดลอง	30
4.6.2 ผลการทดลอง	31
4.7 การทดลองส่วนของวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	31
4.7.1 ลำดับขั้นการทดลอง	31
4.7.2 ผลการทดลอง	32
4.8 การทดลองส่วนของวงจรไมโครคอนโทรลเลอร์	32
4.8.1 ลำดับขั้นการทดลอง	32
4.8.2 ผลการทดลอง	33
4.9 การทดลองการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	33
4.9.1 กรณีการโทรออก	33
4.9.2 กรณีการโทรเข้า	34
4.9.3 กรณีการโทรศัพท์ออกหมายเลขพิเศษ	34
4.9.4 กรณีของผู้ดูแลระบบ	35
4.9.5 กรณีการเชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์	37

สารบัญ (ต่อ)

เรื่อง	หน้า
บทที่ 5 บทสรุป	38
5.1 สรุป	38
5.2 ปัญหาและแนวทางแก้ไข	38
5.3 แนวทางการพัฒนา	39
บรรณานุกรม	40
ภาคผนวก ก เครื่องต้นแบบ	41
ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์	45
ภาคผนวก ค รายการอุปกรณ์	57
ภาคผนวก ง แผนผังการทำงานและรหัสต้นฉบับของโปรแกรม	65
ภาคผนวก จ คู่มือการใช้งาน	117
ภาคผนวก ฉ รายละเอียดและคุณสมบัติของอุปกรณ์	126
ประวัติผู้แต่ง	151

สารบัญตาราง

ตารางที่	หน้า
4.1 ผลการทดลองวงจรถอดรหัสสัญญาณความถี่คู่	27
ค.1 รายการอุปกรณ์ของวงจรตรวจจับสัญญาณกระดิ่ง	58
ค.2 รายการอุปกรณ์ของวงจรถอดรหัสสัญญาณความถี่คู่	58
ค.3 รายการอุปกรณ์ของวงจรตรวจจับการยกหูโทรศัพท์	59
ค.4 รายการอุปกรณ์ของวงจรทวีแรงดัน	60
ค.5 รายการอุปกรณ์ของวงจรตั้งเวลาระงับการโทรออกชั่วคราว	60
ค.6 รายการอุปกรณ์ของวงจรตัดต่อคู่สายโทรศัพท์	61
ค.7 รายการอุปกรณ์ของวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	62
ค.8 รายการอุปกรณ์ของวงจรไมโครคอนโทรลเลอร์	63
ค.9 รายการอุปกรณ์ของวงจรแหล่งจ่ายไฟ	64

สารบัญรูป

รูปที่	หน้า
2.1 รายละเอียดขาไอซี MT8870	5
2.2 ลักษณะตัวถังและการจัดขาของไอซี DS1307	7
2.3 ผังการทำงานภายในของไอซี DS1307	7
2.4 โครงสร้างภายในของไมโครคอนโทรลเลอร์	9
2.5 การเชื่อมต่อใช้งานกับคอนเนคเตอร์ DB-9 และการจัดขาของไอซี MAX232	11
3.1 แผนผังการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	16
3.2 วงจรแหล่งจ่ายไฟ	17
3.3 วงจรทวีแรงดัน	17
3.4 วงจรถอดรหัสสัญญาณความถี่คู่	18
3.5 วงจรตรวจจับสัญญาณกระดิ่ง	19
3.6 วงจรตรวจจับการยกหูโทรศัพท์	20
3.7 วงจรตัดต่อคู่สายโทรศัพท์	21
3.8 วงจรตั้งเวลาระงับการโทรออกชั่วคราว	22
3.9 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	23
3.10 วงจรไมโครคอนโทรลเลอร์	24
4.1 วงจรถอดรหัสสัญญาณความถี่คู่	27
4.2 วงจรตรวจจับสัญญาณกระดิ่ง	28
4.3 วงจรตรวจจับการยกหูโทรศัพท์	29
4.4 วงจรตัดต่อคู่สายโทรศัพท์	30
4.5 วงจรตั้งเวลาระงับการโทรออกชั่วคราว	30
4.6 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	31
4.7 วงจรไมโครคอนโทรลเลอร์	32
4.8 รูปแบบโปรแกรมในขณะใช้งาน	37
ก.1 วงจรแหล่งจ่ายไฟ	42
ก.2 วงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว	42
ก.3 วงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์	43
ก.4 วงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	43

สารบัญรูป (ต่อ)

รูปที่	หน้า
ก.5 วงจรไมโครคอนโทรลเลอร์	44
ก.6 เครื่องต้นแบบของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ	44
ข.1 รูปวงจรรวมของภาคโทรศัพท์	46
ข.2 วงจรแหล่งจ่ายไฟ	47
ข.3 ลายวงจรแหล่งจ่ายไฟ	47
ข.4 การวางอุปกรณ์วงจรแหล่งจ่ายไฟ	47
ข.5 วงจรทวีแรงดัน	48
ข.6 วงจรตั้งเวลาระงับการโทรออกชั่วคราว	48
ข.7 วงจรตัดต่อคู่สายโทรศัพท์	48
ข.8 ลายวงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว	49
ข.9 การวางอุปกรณ์วงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว	49
ข.10 วงจรถอดรหัสสัญญาณความถี่คู่	50
ข.11 วงจรตรวจจับการยกหูโทรศัพท์	50
ข.12 ลายวงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์	51
ข.13 การวางอุปกรณ์วงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์	51
ข.14 วงจรตรวจจับสัญญาณกระดิ่ง	52
ข.15 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	52
ข.16 ลายวงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	53
ข.17 วางอุปกรณ์วงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ	53
ข.18 วงจรไมโครคอนโทรลเลอร์	54
ข.19 ลายวงจรไมโครคอนโทรลเลอร์	55
ข.20 การวางอุปกรณ์วงจรไมโครคอนโทรลเลอร์	56
ง.1 แผนผังการทำงานของภาคโทรออก	66
ง.2 แผนผังการทำงานในส่วนของผู้ดูแลระบบ	67
ง.3 แผนผังการทำงานภาคโทรเข้า	68
จ.1 ส่วนประกอบและปุ่มควบคุมของเครื่องบันทึกข้อมูลการใช้งานโทรศัพท์อัตโนมัติ	119
จ.2 รูปแบบโปรแกรมในขณะใช้งาน	124

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ปัจจุบันการสื่อสารได้เข้ามามีบทบาทต่อการใช้ชีวิตของมนุษย์มากขึ้น และโทรศัพท์จัดว่าเป็นอุปกรณ์ที่ใช้ในการสื่อสารที่สำคัญประเภทหนึ่งซึ่งมีใช้อยู่โดยทั่วไป แต่มักจะมีปัญหาเกี่ยวกับค่าใช้จ่ายซึ่งไม่สามารถตรวจสอบได้ว่าผู้ใช้โทรศัพท์เป็นใคร โทรออกเบอร์อะไร และเวลาเท่าไร โดยเฉพาะในหอพักหรือบ้านเช่า จากปัญหาที่เกิดขึ้นนี้ จึงได้นำมาเป็นแนวคิดในการสร้างเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติเพื่อลดปัญหาดังกล่าว โดยการใช้งานโทรศัพท์เพียง 1 เลขหมาย แต่มีผู้ใช้จำนวนหลายคน สามารถที่จะตรวจสอบและระบุได้ว่าผู้ใช้โทรศัพท์เป็นใคร โทรออกเบอร์อะไร จำนวนนาทีในการโทร และทำการบันทึก วัน เดือน ปี เวลาในการโทรออกอีกด้วย โดยการนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้งานกับระบบโทรศัพท์ และวงจรตรวจจับสัญญาณโทรศัพท์ พร้อมทั้งสามารถเชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์และสามารถพิมพ์ข้อมูลออกมาได้

1.2 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. แสดงผลด้วยจอแอลซีดี 20 ตัวอักษร 2 บรรทัด
2. กำหนดรหัสผ่านสำหรับผู้ใช้ได้ 20 คน ซึ่งกำหนด 1 คน ต่อรหัส 4 หลัก
3. ใส่รหัสผ่านผิดได้ไม่เกิน 3 ครั้ง
4. มีหน่วยความจำสำหรับบันทึกรหัสผู้ใช้ หมายเลขที่โทรออก เวลา วัน เดือน ปี และจำนวนนาทีในการใช้โทรศัพท์
5. สามารถใช้หมายเลขพิเศษได้โดยไม่ต้องผ่านระบบรหัสผ่านของเครื่องโทรศัพท์
6. เชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์และพิมพ์ข้อมูลออกมาได้
7. มีสวิตช์เลือกไปยังโหมดการใช้งานปกติของโทรศัพท์เมื่อไฟฟ้าดับ

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อสะดวกต่อการศึกษาและทำความเข้าใจ ในแต่ละบทจะประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 บทนำ กล่าวถึงความเป็นมาและ ความสำคัญของปริญญานิพนธ์ ขีดความสามารถของโครงงาน และเนื้อหาในบทต่างๆ โดยสังเขป

บทที่ 2 ทฤษฎีและหลักการ ประกอบด้วยทฤษฎีต่างๆ เกี่ยวกับโครงสร้างของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ องค์ประกอบของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ และคุณสมบัติของตัวอุปกรณ์

บทที่ 3 การออกแบบ การสร้าง และการทำงาน กล่าวถึงเนื้อหาที่เกี่ยวกับผังวงจรต่างๆ ที่ใช้ในโครงงาน ตลอดจนขั้นตอนในการออกแบบวงจร วิธีการสร้าง พร้อมทั้งการทำงานของวงจร และการทำงานของโปรแกรมควบคุมต่างๆ โดยละเอียด

บทที่ 4 การทดลองและผลการทดลอง ประกอบด้วย การทดลอง ผลการทดลองและปัญหาที่เกิดขึ้นพร้อมทั้งการแก้ไขปัญหาของวงจรต่างๆ

บทที่ 5 บทสรุป เป็นการสรุปผลการจัดทำโครงงาน ปัญหาที่เกิดขึ้นและแนวทางการแก้ไขปัญหา รวมทั้งแนวทางในการนำโครงงานไปพัฒนาต่อไป

ภาคผนวก ก แสดงภาพเครื่องต้นแบบ

ภาคผนวก ข ประกอบด้วยผังรายละเอียดวงจรและแผ่นวงจรพิมพ์

ภาคผนวก ค แสดงรายการอุปกรณ์ที่ใช้งานในแต่ละวงจร

ภาคผนวก ง แสดงแผนผังการทำงานและโปรแกรมควบคุมการทำงาน

ภาคผนวก จ คู่มือการใช้งาน

ภาคผนวก ฉ แสดงรายละเอียดและคุณสมบัติของอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

หลักการและทฤษฎีที่เกี่ยวข้อง คือ วงจรของเครื่องรับโทรศัพท์ สัญญาณที่รับส่งระหว่างผู้เช่าและชุมสาย โทรศัพท์แบบความถี่คู่ ไอซีถอดรหัสความถี่โทรศัพท์ (MT8870) ไอซีฐานเวลา (DS 1307) จอแสดงผลแบบผลึกเหลว (Liquid Crystal Display : LCD) ไมโครคอนโทรลเลอร์ MCS-51 หน่วยความจำภายนอก มาตรฐานการเชื่อมต่อพอร์ตอนุกรมแบบ RS-232 จังหวะและเวลาของการสื่อสารข้อมูลอนุกรม รูปแบบของการสื่อสารข้อมูลอนุกรม และการใช้งานโปรแกรมวิชวลเบสิค

2.2 วงจรของเครื่องรับโทรศัพท์

สามารถแยกออกได้เป็น 3 ส่วนด้วยกัน วงจรกระดิ่ง วงจรปากพูด-หูฟัง และวงจรเข้ารหัสตัวเลขชนิดปุ่มกด

2.2.1 วงจรกระดิ่ง

ทำหน้าที่เรียกเป็นเสียงกระดิ่งทุกครั้งที่มีการเรียกเข้ามาเพื่อให้เจ้าของเครื่องทราบว่ามีการโทรเข้ามา วงจรกระดิ่งอาจใช้กระดิ่งหรือไอซีโทนริงเกอร์ก็ได้ หากใช้กระดิ่งเมื่อมีการเรียกเข้ามาแรงดันไฟฟ้าสลับประมาณ 150 โวลต์ ถูกส่งเข้ามายังโซลินอยด์ ทำให้โซลินอยด์มีอำนาจแม่เหล็กเกิดขึ้นเหมือนกระดิ่งโดยทั่วไป หากเป็นระบบไอซีจะมีการนำพัลส์ดังกล่าวเข้าสู่ระบบเร็กติไฟเออร์เพื่อจ่ายแรงดันให้กับหน่วยผลิตความถี่เสียง และส่งความถี่เสียงออกลำโพง

2.2.2 วงจรปากพูด-หูฟัง

ทำหน้าที่เสมือนวงจรเครื่องรับและเครื่องส่ง เพียงแต่เครื่องรับและเครื่องส่งของระบบจริงๆ นั้น การเรียกจากฝ่ายหนึ่งไปยังอีกฝ่ายหนึ่ง สัญญาณเรียกจะถูกส่งผ่านระบบชุมสายและจากชุมสายจะมีการแยกด้วยระบบมัลติเพล็กซ์เพื่อแยกคู่สายไปยังหมายเลขที่ผู้เรียกต้องการติดต่อสายออกไปด้วยแรงดันไฟ 100 โวลต์ เมื่อเครื่องรับยกหูจะมีกระแสไหลในวงจรด้วยระบบรักษากระแสคงที่ไว้ที่ 23 มิลลิแอมป์ ส่งผลให้แรงดันไฟในคู่สายโทรศัพท์ตกลงมาเหลือประมาณ 6-8 โวลต์ หากไม่ยกหูโทรศัพท์ ระดับแรงดันจากชุมสายโทรศัพท์มายังคู่สายโทรศัพท์วัดแรงดันได้ 48 โวลต์

2.2.3 วงจรเข้ารหัสตัวเลขชนิดปุ่มกด

เมื่อต้องการเรียกคู่สนทนา ท่านต้องยกหูโทรศัพท์แล้วกดหมายเลขที่ต้องการจากแป้นกด วงจรเข้ารหัสจะเปลี่ยนหมายเลขให้เป็นความถี่คู่แล้วส่งความถี่ 2 โทน (สูง-ต่ำ) ไปยังระบบชุมสาย แล้วชุมสายจะส่งข้อมูลไปตามเครือข่ายต่อไป

2.3 สัญญาณที่รับส่งระหว่างผู้เช่าและชุมสาย

2.3.1 สัญญาณที่ส่งจากผู้เช่าไปยังชุมสาย

1) ON HOOK หมายถึง สภาพผู้เช่าวางหู ลักษณะของวงจรเป็นเสมือนวงจรเปิดที่มีความต้านทานสูง

2) OFF HOOK หมายถึง สภาพผู้เช่ายกหู สายโทรศัพท์จะมีสภาพเสมือนวงจรปิดมีความต้านทานต่ำ

3) DIALLING หมายถึง ผู้เช่าทำการหมุนหมายเลข

2.3.2 สัญญาณที่ส่งจากชุมสายมายังผู้เช่า

1) Dialling Tone เป็นสัญญาณที่แจ้งให้ผู้เรียกทราบว่า ขณะนี้อุปกรณ์ที่ชุมสายนั้นพร้อมที่จะรับรหัสการหมุนหมายเลขจากผู้เรียก ให้ผู้เรียกทำการส่งหมายเลขได้ สัญญาณให้หมุนนี้เป็นสัญญาณต่อเนื่องความถี่ 425 เฮิรตซ์ มอดูเลตด้วย 50 เฮิรตซ์ ผู้เช่าจะได้ยินเสียงเมื่อยกหูโทรศัพท์

2) Busy Tone เป็นสัญญาณที่แจ้งให้ทราบว่าอุปกรณ์ปลายทางของชุมสายหรือโทรศัพท์ปลายทางไม่ว่าง ความถี่ของสัญญาณไซน์ 425 เฮิรตซ์

3) Ringing Tone เป็นสัญญาณที่แจ้งให้ผู้ถูกเรียกทราบหลังจากหมุนหมายเลขครบเพื่อบอกให้ผู้เรียกทราบว่ามีผู้ต้องการติดต่อ ด้วยความถี่ของสัญญาณไซน์ 425 เฮิรตซ์

4) Ringback Tone เป็นสัญญาณที่ชุมสายส่งไปยังผู้เรียกเพื่อบอกให้ผู้เรียกทราบว่าได้ต่อเส้นทางกับผู้ถูกเรียกได้แล้ว

2.4 โทรศัพท์แบบความถี่คู่

การหมุนหมายเลขโทรศัพท์ที่ใช้กันอยู่ในปัจจุบันนี้มีอยู่ 2 แบบ คือ แบบพัลส์และแบบโทนหรือแบบกดปุ่ม (Dual Tone Multi Frequency : DTMF) แบบดีทีเอ็มเอฟเป็นแบบที่กำหนดสัญญาณเสียง 2 โทนเสียง โดยความถี่จะแยกออกเป็น 2 กลุ่มใหญ่ คือ กลุ่มความถี่สูงกับกลุ่มความถี่ต่ำ กลุ่มความถี่ต่ำได้แก่ 627 เฮิรตซ์ 770 เฮิรตซ์ 852 เฮิรตซ์ และ 941 เฮิรตซ์ ส่วนกลุ่มความถี่สูงได้แก่ความถี่ 1,209 เฮิรตซ์ 1,336 เฮิรตซ์ 1,477 เฮิรตซ์ และ 1,633 เฮิรตซ์ โดยใช้ระบบคีย์แบบ เอ็ก-

วาย (X-Y Matrix) แบ่งโซนออกเป็นกลุ่มความถี่ทางแนวตั้งกับกลุ่มความถี่ทางแนวนอน โดยที่กลุ่มความถี่ทางแนวนอนเป็นกลุ่มของความถี่ต่ำและความถี่แนวตั้งเป็นกลุ่มของความถี่สูง

ข้อดีของการใช้โทรศัพท์แบบโทน

1. ลดระยะเวลาในการส่งหมายเลขโทรศัพท์ไปยังชุมสายโทรศัพท์
2. สามารถใช้ไอซีสำเร็จรูปได้ทำให้มีขนาดเล็กลง
3. ลดจำนวนอุปกรณ์ในการกำหนดหมายเลข ทำให้ชุมสายมีขนาดเล็กลง
4. สามารถนำไปเชื่อมต่อกับอุปกรณ์ภายในชุมสายโทรศัพท์ได้อย่างมีประสิทธิภาพ
5. สามารถนำไปใช้กับระบบอัตโนมัติต่างๆ ได้กว้างขวางขึ้น เช่น ระบบต่อหมายเลขภายในอัตโนมัติชุดตอบรับและโอนสายโทรศัพท์อัตโนมัติ เป็นต้น

2.5 ไอซีถอดรหัสความถี่โทรศัพท์

ใช้สำหรับถอดรหัสความถี่ของโทรศัพท์ชนิดกดปุ่ม ให้เป็นตัวเลขฐานสอง ขนาด 4 บิต ความหมายของคำว่า ถอดรหัสความถี่ของโทรศัพท์ หมายถึง การแปลงสัญญาณความถี่ซึ่งเกิดจากการกดปุ่มตัวเลขโทรศัพท์ชนิดกดปุ่ม ให้เป็นระบบตัวเลขดิจิตอลซึ่งใช้ไอซี MT8870 ในการแปลงความถี่โทรศัพท์ให้เป็นเลขฐานสองขนาด 4 บิต

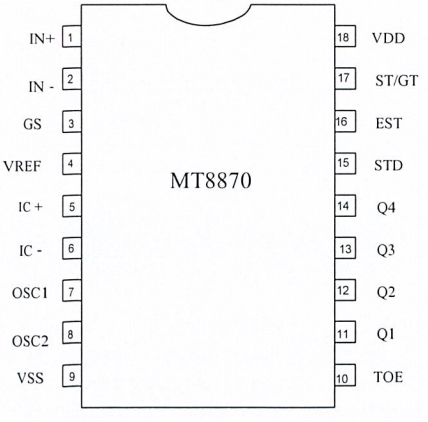
2.5.1 คุณสมบัติของไอซี MT8870

1) เป็นตัวรับถอดรหัสความถี่คู่ (DTMF Receiver)

2) กินไฟน้อย ใช้ไฟเลี้ยงระดับเดียวกับทีทีแอล (TTL)

3) สามารถปรับช่วงคาบเวลาของความถี่ (Guard Time) ที่เข้ามาได้

4) สามารถปรับอัตราการขยายในตัวไอซีได้



รูปที่ 2.1 รายละเอียดขาของไอซี MT8870

2.5.2 การนำไอซี MT8870 ไปใช้งานด้านอื่นๆ

1) นำไปใช้งานด้านการควบคุมระยะไกล

2) ใช้กับเครื่องป้องกันโทรศัพท์ทางไกล

3) ใช้กับงานเกี่ยวกับบัตรเครดิต

4) ใช้งานร่วมกับคอมพิวเตอร์

5) ใช้ในงานตู้สาขาอัตโนมัติ

6) ใช้ในงานด้านโทรศัพท์ทั่วไป

7) เครื่องกันขโมย

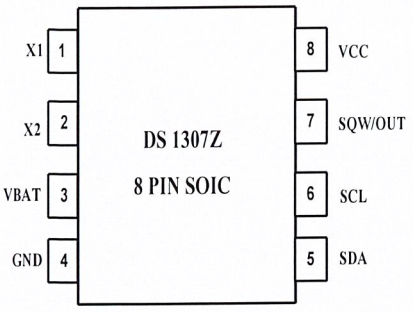
8) การควบคุมอุปกรณ์ทางโทรศัพท์

2.5.3 โครงสร้างภายในของไอซี MT8870

โครงสร้างภายในของไอซี MT8870 ประกอบไปด้วยวงจรกรองความถี่และวงจรถอดรหัสฟังก์ชั่นทางดิจิตอล เป็นไอซีที่สร้างโดยใช้เทคโนโลยี ISO^2-CMOS ในส่วนของวงจรกรองความถี่ใช้เทคนิคของสวิตช์คาปาซิเตอร์ฟิลเตอร์ สำหรับกรองความถี่สูงและต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอลเพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ ออกแบบเป็นเลขฐานสองขนาด 4 บิต และตรวจสอบช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุตเป็นออปแอมป์ซึ่งสามารถปรับอัตราการขยายได้โดยต่อกับอุปกรณ์ภายนอก เอาต์พุตเป็นวงจรแลทช์ 3 สถานะ

2.6 ไอซีฐานเวลา

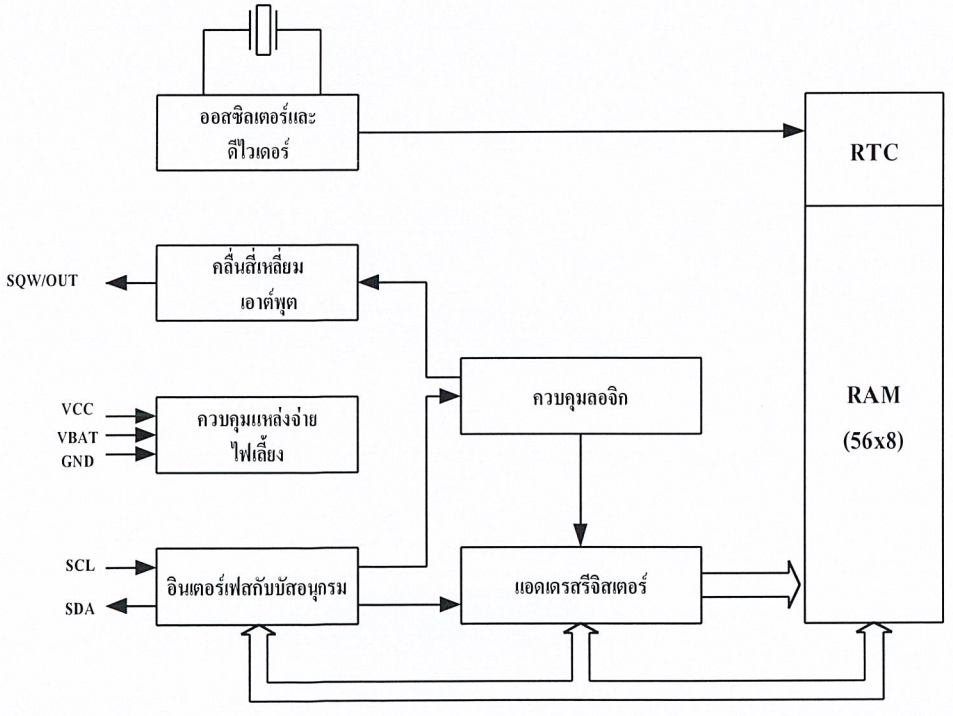
ไอซี DS1307 เป็นไอซีฐานเวลาแบบเรียลไทม์ที่มีอัตรากินกำลังงานต่ำมากพร้อมกับปฏิทินเวลาแบบบีซีดี (BCD) เต็มรูปแบบ ภายในยังประกอบด้วยหน่วยความจำสแตติกแรม แบบนอนโวลาไทล์ (Nonvolatile) ขนาด 56 ไบต์ อีกด้วย การส่งผ่านข้อมูลและแอดเดรสจะส่งผ่านในรูปของบัสอนุกรม 2 สายแบบ Bi-Directional การแสดงผลเป็นทั้งแบบฐานเวลา และปฏิทินที่มีการแสดงผลตั้งแต่เวลาเป็นวินาที นาที ชั่วโมง วัน วันที่ เดือนและปี โดยมีการปรับเปลี่ยนวันที่เป็นวันสุดท้ายของแต่ละเดือนอย่างอัตโนมัติในแต่ละเดือน ซึ่งมีการแสดงที่สูงสุดไม่เกิน 31 วัน และมีการแก้ไขความถูกต้องเมื่อครบปีอย่างอัตโนมัติเช่นกัน ส่วนการทำงานในการแสดงเวลานั้นจะกำหนดได้ทั้งแบบ 24 ชั่วโมงหรือ 12 ชั่วโมงในรูปแบบการแสดงผลเป็น PM และ AM นอกจากนั้นแล้วไอซี DS1307 ยังมีวงจรตรวจสอบความผิดปกติของแรงดันร่วมอยู่ภายใน หากมีความผิดพลาดของแหล่งจ่ายไฟหลักก็จะทำการเลือกสวิตช์ไปยังแหล่งจ่ายที่เป็นแบตเตอรี่สำรองแทนทันที เพื่อการสำรองข้อมูลไม่ให้เกิดความเสียหายหรือเกิดความผิดพลาดขึ้นภายในตัวไอซี DS1307



รูปที่ 2.2 ลักษณะตัวถังและการจัดขาของไอซี DS1307

2.6.1 การทำงานเบื้องต้น

ไอซี DS1307 จะทำงานเป็นอุปกรณ์เสริมเข้าไปในบัสอนุกรม การเข้าถึงข้อมูลภายในหน่วยความจำของไอซีสามารถกระทำได้โดยการเซตให้อยู่ในสภาวะเริ่มต้น (Start) และตามด้วยรหัสประจำตัวของอุปกรณ์จากรีจิสเตอร์แอดเดรส การเข้าถึงลำดับของรีจิสเตอร์ย่อยก็สามารถกระทำและเข้าถึงได้อย่างเป็นลำดับจนกระทั่งเลือกการทำงานอยู่ในตำแหน่งหยุด (Stop)



รูปที่ 2.3 ผังการทำงานภายในของไอซี DS1307

ในรูปที่ 2.3 แสดงแผนผังการทำงาน ส่วนการทำงานต่างๆ ภายในตัวไอซีพิจารณาดูส่วนตัวตรวจจับแรงดันไฟเลี้ยงผิดพลาดเมื่อแรงดัน Vcc ต่ำลงเมื่อต่ำกว่า 1.25 x V_{bat} ไอซีจะหยุดการทำงานลงแต่การเข้าถึงต่างๆ ภายในยังคงดำเนินการต่อไป และจะเกิดการรีเซตการนับแอดเดรสของอุปกรณ์ไว้เป็นอันดับสุดท้าย สัญญาณต่างๆ ที่ป้อนเข้าสู่อินพุตของไอซีจะไม่มีการบันทึกใดๆ ไว้ทั้งสิ้น หรืออินพุตของไอซีจะไม่รับสัญญาณใดๆ ทั้งสิ้นในขณะนั้น เพื่อเป็นการป้องกันความผิดพลาดของข้อมูลจากการที่เริ่มเขียนเพื่อบันทึกไว้ภายในไอซีและนอกเหนือจากการยอมรับได้ของระบบในขณะที่แรงดัน Vcc ตกลงต่ำกว่าค่าแรงดัน V_{bat} ไอซีจะทำการสวิตช์เลือกแหล่งจากไปสู่โหมดแบตเตอรี่สำรองที่จะกินกำลังต่ำกว่ามากทันที ในทำนองเดียวกันไอซีก็จะทำการสวิตช์เลือกกลับจากแบตเตอรี่สำรองไปสู่แหล่งจ่ายหลัก Vcc อีกครั้งก็ต่อเมื่อแรงดันไฟเลี้ยง Vcc มีค่าสูงกว่า V_{bat} +0.2 โวลท์ จะเริ่มรับรู้สัญญาณข้อมูลต่างๆ ทางอินพุตได้เมื่อค่าแรงดัน Vcc มีค่าสูงกว่า 1.25 x V_{bat} ซึ่งการสวิตช์เลือกแหล่งแรงดันไฟเลี้ยงนี้จะเป็นไปโดยอัตโนมัติ จึงมั่นใจได้ว่าข้อมูลต่างๆ ที่ตัวไอซีจะไม่มีการผิดพลาดอย่างแน่นอน

2.7 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ผลิตโดยบริษัทอินเทล มีการนำมาใช้ประโยชน์กันอย่างมากมายในปี ค.ศ. 1980 ต่อมาบริษัทฟิลิปปีและซีเมนส์ได้รับลิขสิทธิ์ในการผลิตจำหน่ายและได้มีการเพิ่มประสิทธิภาพมากขึ้น ทำให้ในปัจจุบันมีไมโครคอนโทรลเลอร์ที่มีมาตรฐานมาจาก MCS-51 ของบริษัทอินเทลเป็นจำนวนมาก

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วยไมโครคอนโทรลเลอร์หลายรุ่น ซึ่งจะมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน จะต่างกันเพียงขนาดของหน่วยความความจำภายในและหน่วยความจำภายนอกเท่านั้น ในการใช้งานสามารถเลือกใช้ได้ตามความต้องการและความเหมาะสม ไมโครคอนโทรลเลอร์ MCS-51 จัดเป็นไมโครคอนโทรลเลอร์แบบชิพเดี่ยวที่ไม่ต้องต่อร่วมกับอุปกรณ์ภายนอกก็สามารถทำงานได้

2.7.1 คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์

1) หน่วยประมวลผลกลาง (Central Processing Unit : CPU)

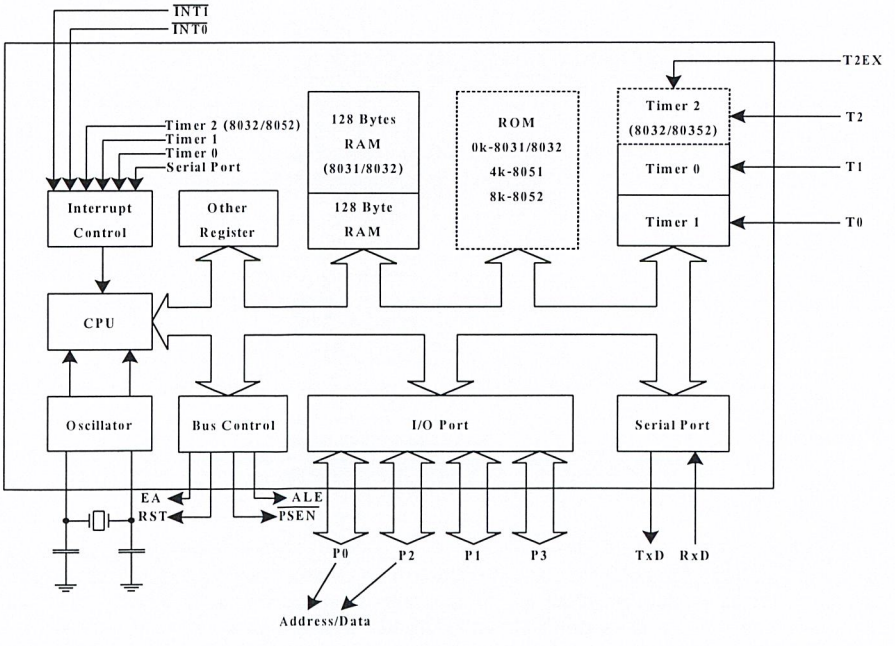
ส่วนนี้จะทำหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่นๆ เรียกว่า วงจรควบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุม ได้แก่ สัญญาณสำหรับการติดต่อกับหน่วยความจำ อุปกรณ์รับข้อมูลหรือส่งข้อมูลออก ซึ่งส่วนควบคุมการขัดจังหวะและส่วนควบคุมบัสก็เป็นส่วนหนึ่งของวงจรควบคุมด้วย การสร้างสัญญาณจากวงจรควบคุมจากหน่วยประมวลผลกลางนี้ทำการ

สร้างสัญญาณนาฬิกาที่สร้างจากวงจรออสซิลเลเตอร์ เพื่อให้ทุกๆ ส่วนทำงานประสานกันอย่างถูกต้อง ในหน่วยประมวลผลกลางยังประกอบด้วยส่วนประมวลผล (Arithmetic Logic Unit) ที่ทำหน้าที่ประมวลผลข้อมูล เช่น การลบ บวก คูณหรือหารข้อมูล แล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำที่ต้องการ

2) หน่วยความจำ (Memory)

หน่วยความจำมีไว้สำหรับจดจำข้อมูล ในการนำข้อมูลเข้าและออกจากหน่วยความจำจำเป็นต้องรู้ตำแหน่ง (Address) ของหน่วยความจำ ในการนำข้อมูลเข้าไปเก็บในหน่วยความจำเรียกว่าการเขียนข้อมูลและการนำข้อมูลออกจากหน่วยความจำ เรียกว่า การอ่านข้อมูล ในไมโครคอนโทรลเลอร์ MCS-51 ข้อมูลในแต่ละตำแหน่งจะมีขนาด 8 บิต ดังนั้น แต่ละตำแหน่งของหน่วยความจำจะสามารถเก็บข้อมูลมีค่าได้ระหว่าง 00000000_2 ถึง 11111111_2 หรือ 00H ถึง 0FFH ในการติดต่อกับหน่วยความจำจะต้องมีสัญญาณ 3 กลุ่ม คือ

ตำแหน่งที่ต้องการติดต่อกับหน่วยความจำ ซึ่ง MCS-51 สามารถติดต่อกับหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลที่มีขนาดสูงสุดชนิดละ 65,536 ตำแหน่ง (64 กิโลไบต์) ดังนั้นการอ้างตำแหน่งของหน่วยความจำเป็นจะต้องใช้เส้นแสดงตำแหน่งในเลขฐานสองทั้งหมด 16 เส้น (2^{16} เท่ากับ 65,536) ข้อมูลที่อ่านหรือเขียนกับหน่วยความจำในตำแหน่งที่เราต้องการโครงสร้างของไมโครคอนโทรลเลอร์ จะประกอบด้วย 3 ส่วนหลักๆ ดังรูปที่ 2.4



รูปที่ 2.4 โครงสร้างภายในของไมโครคอนโทรลเลอร์

สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำเพื่อบอกกับหน่วยความจำว่าต้องการอ่านหรือเขียนข้อมูล โดยวงจรถอดรหัสคำสั่งทำการสร้างสัญญาณควบคุมจากคำสั่งที่อ่านเข้ามาจากหน่วยความจำโปรแกรม

3) อุปกรณ์อินพุต/เอาต์พุต (Input/Output Device)

อุปกรณ์อินพุต/เอาต์พุตเป็นส่วนที่ใช้ส่งข้อมูลเข้าหรือนำข้อมูลออกจาก MCS-51 ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกได้ อุปกรณ์อินพุต/เอาต์พุตได้แก่ 4 I/O Port, Time/Counter 0, Time/Counter 1 และ Serial Port

1) 4 อินพุต/เอาต์พุต (4 I/O Port) หรือพอร์ตแบบขนาน เป็นที่สำหรับใช้รับส่งข้อมูลซึ่งเป็นสัญญาณดิจิตอลเข้าหรือออกจากตัว MCS-51 มีทั้งหมด 4 พอร์ต โดยแต่ละพอร์ตจะรับส่งข้อมูลได้ 8 บิต มีพอร์ตจะใช้งานมากกว่า 1 อย่างก็ได้

2) ไทม์เมอร์/เคาน์เตอร์ 0 (Time/Counter) ซึ่งเป็นวงจรนับที่สามารถทำการนับจำนวนไซเคิลของสัญญาณที่ต่อจากภายนอก MCS-51 ก็ได้ สามารถตั้งค่าเริ่มต้นของการนับและอ่านค่าการนับได้โดยหน่วยประมวลผลกลาง

3) พอร์ตอนุกรม (Serial Port) หน่วยประมวลผลกลางจะอ่านและเขียนข้อมูลกับพอร์ตอนุกรมเป็นแบบ 8 บิต แต่ข้อมูลจะถูกส่งออกจาก MCS-51 เรียงไปที่ละบิตออกจากขา Tx และในการรับข้อมูลจะรับเข้ามาที่ละบิตทางขา Rx แล้วจึงจัดเรียงใหม่เป็น 8 บิต เพื่อให้หน่วยประมวลผลกลางอ่านไปใช้งานต่อไป

ใน MCS-51 มีพอร์ตให้ใช้งานได้หลายแบบ ทำให้สะดวกแก่การนำไปใช้งานต่างๆ ได้มากมาย การจะนำพอร์ตไปใช้งานได้จะต้องเขียนโปรแกรมขึ้นมาควบคุม

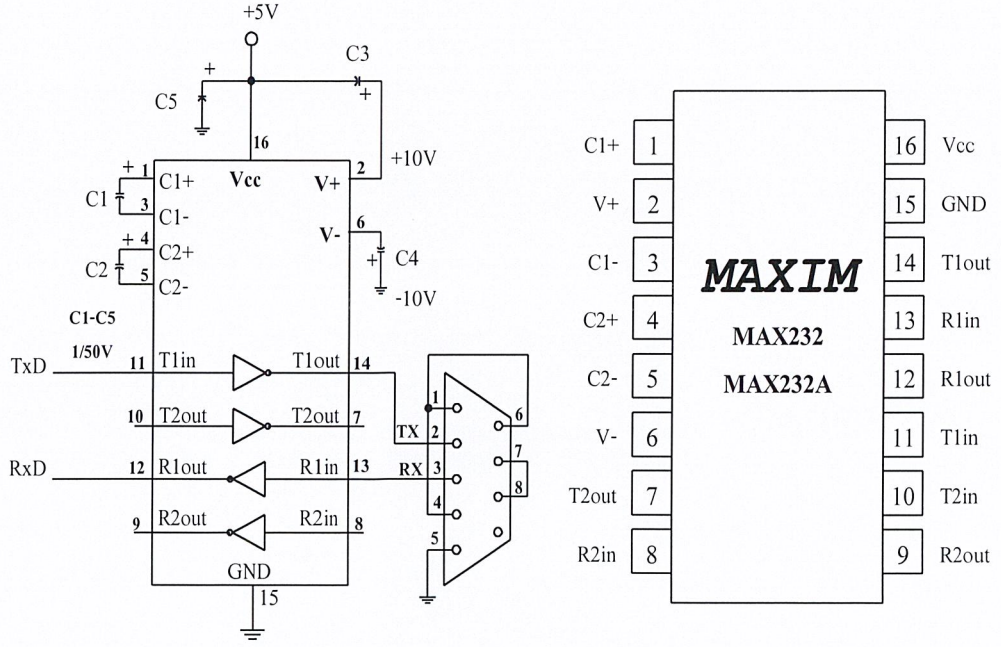
2.8 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม

ในการสื่อสารข้อมูลแบบอนุกรมเพื่อรับหรือส่งข้อมูลจะเป็นลักษณะของกลุ่มข้อมูล ดังนั้นอัตราความเร็วจะต้องมีค่าเท่ากันระหว่างการรับและการส่งโดยทั่วไปเราจะระบุความเร็วของจำนวนบิตในการรับและส่งข้อมูล เป็นจำนวนของบิตที่จะส่งใน 1 วินาที โดยเรียกความเร็วในการส่งข้อมูลว่าอัตราบอร์ด (Baud Rate) ซึ่งมีหน่วยเป็นบิตต่อวินาที เช่น 300, 1200, 2400, 4800 และ 9600 บิตต่อวินาที ถ้าหากมีการส่งข้อมูลด้วยความเร็ว 9600 บิตต่อวินาที จะใช้เวลาในการส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104.1 ไมโครวินาที และเวลาในการรับส่งข้อมูลทั้ง 8 บิตจะมีค่าเท่ากับ 8x104.1 หรือ 832.8 ไมโครวินาที

2.8.1 มาตรฐานการเชื่อมต่อพอร์ตอนุกรมแบบ RS-232

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ± 3 ถึง ± 12 โวลท์ ในขณะที่ระดับสัญญาณของไมโครคอนโทรเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

ไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้คือไอซี MAX232 ที่ทำการแปลงข้อมูลส่งจากไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และจะทำการแปลงข้อมูลรับจากคอมพิวเตอร์ จากระดับของ RS-232 เป็นระดับทีทีแอล เพื่อที่จะให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์



รูปที่ 2.6 การเชื่อมต่อใช้งานกับคอนเนคเตอร์ DB-9 และการจัดขาของไอซี MAX232

2.9 รูปแบบของการสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัส เป็นวิธีการรับและส่งข้อมูลโดยที่ไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่ม

บิตข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริง เพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้น ซึ่งประกอบด้วย 4 ส่วน คือ

1) บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นระดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสาร ขณะที่ยังไม่มีการส่งข้อมูล

2) บิตข้อมูล (Data bit) จะเริ่มจากบิตที่มีนัยสำคัญต่ำสุดก่อนหรือบิต LSB ก่อน โดยข้อมูลที่จะส่งอาจจะมีขนาด 5, 6, 7 หรือ 8 บิต

3) บิตแสดงสภาวะเลขคู่หรือเลขคี่ (Parity bit) มีขนาด 1 บิต โดยบิตนี้จะไปต่อท้ายกับบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็น “1” โดยเลือกการส่งข้อมูลเป็นแบบพาริตี้คี่ ตัวอย่าง ถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลข 1 เป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็น “1” เพื่อจะได้จำนวนเลข 1 เป็นคู่นั้นเอง ทำนองเดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็น “1” รวมทั้งบิตพาริตี้ 1 บิต ถ้ามีค่า “1” เป็นจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง (สามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้)

4) บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสภาวะลอจิกเป็น “1” ซึ่งอาจมีจำนวนมากกว่าหนึ่งบิตก็ได้ เช่น 1 บิต 1.5 บิต หรือ 2 บิต

2.10 การใช้งานโปรแกรมวิชวลเบสิค

ความเป็นมาและการพัฒนาโปรแกรมวิชวลเบสิค (Visual Basic 6.0) นั้นเป็นภาษาที่พัฒนาขึ้นโดยบริษัท ไมโครซอฟต์ คอร์ปอเรชั่น ซึ่งเป็นบริษัทยักษ์ใหญ่ที่สร้างระบบปฏิบัติการเพื่อให้เป็นภาษาสำหรับใช้โปรแกรมในระบบปฏิบัติการวินโดว์ส (Windows) และระบบปฏิบัติการวินโดว์ เอ็น ที (Windows NT) ซึ่งเป็นระบบปฏิบัติการของบริษัทดังกล่าวเช่นกันหลักการสำคัญของโปรแกรมวิชวลเบสิค 6.0 คือ การที่โปรแกรมเมอร์ออกแบบหน้าจอสำหรับติดต่อกับผู้ใช้ (User Interface : UI) โดยใช้การวาดภาพ (Drawing) และวาดองค์ประกอบต่างๆ (Components) ลงบนหน้าจอได้ตามความต้องการของโปรแกรมเมอร์ ซึ่งจะทำให้โปรแกรมเมอร์เห็นและแก้ไขหน้าจอได้ทันทีด้วยการเห็นสิ่งที่โปรแกรมเมอร์ ต้องการให้ระบบงานประยุกต์เป็นไปตามนั้นหรือเรียกว่า What You See Is What You Get (WYSIWYG) เมื่อวางองค์ประกอบต่างๆ ของหน้าจอเสร็จจะเป็นขั้นตอนการเขียน

จุดเริ่มต้นของโปรแกรมภาษาที่ใช้เขียนโปรแกรมวิชวลเบสิค เป็นโปรแกรมที่มีโครงสร้างของภาษาเบสิค (Beginning’s All-Purpose Symbolic Instruction Code : BASIC) เกิดขึ้นเมื่อปี ค.ศ.

1967 โดย John Kemeny และ THOMAS KURTZ เป็นผู้ที่ได้ทำการคิดค้นและถูกพัฒนาต่อมาเรื่อยๆ จนเป็น BASICA GWBASIC QBASIC ทางบริษัทไมโครซอฟต์เห็นว่าภาษาเบสิค น่าจะเป็นอนาคตจึงจัดตั้งทีมทำการพัฒนาโปรแกรมวิชวลเบสิค ที่เห็นกันแทบทุกวันนี้

โปรแกรมวิชวลเบสิค เป็นที่นิยมกันในหมู่นักพัฒนาอย่างแพร่หลาย ตั้งแต่รุ่นที่ 3.0 ซึ่งใช้กับระบบปฏิบัติการวินโดวส์ รุ่น 3.11 จนกระทั่งปัจจุบันได้มีการพัฒนามาจนถึงรุ่น 5.0/6.0 ใช้กับระบบปฏิบัติวินโดวส์ 95/98 ที่จะสร้างความสำพันธ์ระหว่างองค์ประกอบต่างๆ ของหน้าจอ เช่น เมนูฟอร์มรายงานเข้าด้วยกัน เป็นต้น โดยการพัฒนาแบบแยกส่วนนี้ทำให้การพัฒนาโปรแกรมด้วยเครื่องมือและภาษานี้เป็นไปอย่างรวดเร็ว หลักการพัฒนาโปรแกรมวิชวลเบสิค อย่างหนึ่ง ทำให้โปรแกรมวิชวลเบสิค เป็นภาษาที่มีความคล่องตัวและได้รับความนิยมอย่างสูง คือ การแบ่งการทำงานของโปรแกรมออกเป็นส่วนย่อยๆ แล้วจึงนำมาต่อกัน เสมือนกับการต่อร่วมที่เป็นโปรแกรมเดียวกันดังตัวอย่าง เช่น ฟอร์มที่ได้ทำการออกแบบและโปรแกรมให้ใช้กับระบบหนึ่งได้ทั้งหมด รวมจนถึงหน้าจอและรหัสของโปรแกรมต่างๆ ที่เขียนไว้ อีกทั้งยังสามารถนำมาทำการปรับเปลี่ยนเพื่อให้เหมาะสมกับงานนั้นๆ มากขึ้น เป็นการช่วยประหยัดเวลาในการพัฒนาโปรแกรมได้ เป็นต้น

ในปัจจุบันผู้ใช้งานส่วนใหญ่จะทำงานอยู่บนระบบปฏิบัติการวินโดวส์ โดยมีรูปภาพที่เป็นตัวสื่อสารระหว่างผู้ใช้กับคอมพิวเตอร์และโปรแกรมต่างๆ ดังนั้นการทำงานส่วนใหญ่จึงไม่มีความจำเป็นที่จะต้องพิมพ์คำสั่งเพียงแต่ทำการใช้เมาส์คลิกเลือกคำสั่งหรือสัญลักษณ์รูป (Icon) ที่ต้องการ การใช้แนวความคิดเดียวกันในการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented) คือ การมองทุกๆ ขององค์ประกอบในโปรแกรมเป็นแบบวัตถุ (Object) หนึ่งชิ้น ซึ่งวัตถุแต่ละชิ้นจะมีคุณสมบัติที่มีความแตกต่างกัน เช่น ปุ่มสามารถเลือกได้โดยใช้เมาส์ กรอบข้อความ (Textbox) สามารถรับค่าจากแป้นพิมพ์ได้โดยตรง โดยมีการทำงานที่มีความแตกต่างกันออกไปตามแต่ละเหตุการณ์ในขณะนั้น ดังนั้นภาษาโปรแกรมรุ่นใหม่ จึงจำเป็นต้องมีการพัฒนาไปตามแนวคิดของการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented) และเขียนโปรแกรมแบบตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้น (Event Driven) คือ การใช้รูปแบบของเหตุการณ์และสภาพแวดล้อมในขณะนั้น เป็นตัวกำหนดความต้องการที่ให้วัตถุประสงค์แต่ละตัวมีปฏิบัติการอย่างไร และแน่นอนโปรแกรมวิชวลเบสิค ก็เป็นภาษารุ่นใหม่ที่พัฒนาภายใต้แนวคิดแบบการกระตุ้นจากเหตุการณ์ที่เกิดขึ้นเช่นกัน

2.10.1 การใช้งานโปรแกรมวิชวลเบสิค ติดต่อพอร์ตอนุกรม

ในการใช้งานโปรแกรมวิชวลเบสิค เพื่อที่จะติดต่อกับพอร์ตอนุกรมนั้นเป็นเรื่องที่สามารถกระทำได้ง่ายมาก เพราะว่าทางผู้ผลิตโปรแกรมทางด้านบริษัทไมโครซอฟต์ได้จัดเตรียมเครื่องมือที่สามารถนำมาใช้เชื่อมต่อกับพอร์ตอนุกรมเอาไว้แล้ว ซึ่งเริ่มตั้งแต่โปรแกรมวิชวลเบสิค 2.0 เป็นต้น

มา สำหรับโปรแกรมวิชวลเบสิค 5.0 และ 6.0 จะมีชื่อเรียกว่า MsComm32.ocx ซึ่งจัดทำขึ้นเพื่อใช้งานกับระบบปฏิบัติการ 32 บิตในระบบปฏิบัติการวินโดวส์ 95/98

รูปแบบของเครื่องมือในการติดต่อสื่อสารภายในโปรแกรมวิชวลเบสิค 6.0 ด้วยคุณสมบัติของเครื่องมือที่มีชื่อเรียกว่า MsComm32.ocx ที่ถูกจัดเตรียมไว้มีวิธีการสื่อสาร 2 ลักษณะดังนี้

ก) ลักษณะการสื่อสารด้วยการกระตุ้นจากเหตุการณ์ที่เกิดขึ้น ซึ่งการสื่อสารลักษณะนี้จะมีประสิทธิภาพมาก กล่าวคือ ตัวควบคุมพอร์ตอนุกรมจะตอบสนองต่อเหตุการณ์แบบทันทีทันใด คือจะเริ่มทำงานเมื่อพอร์ตอนุกรมถูกกระตุ้นทำให้เกิดการเปลี่ยนแปลงที่ขา 1 ของหัวต่อแบบ D-Type ชนิด DB-9 ใช้สำหรับต่อร่วมกับคอมพิวเตอร์ในกรณีของการเขียนโปรแกรมสื่อสารที่จะต้องทำการกำหนดให้การเปลี่ยนแปลงหรือมีการตรวจสอบสัญญาณข้อมูลชนิดพาห์ที่ขา 1 (Data Carrier Detect : DCD) หรือขา 7 ซึ่งเป็นการกำหนดมีการส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางที่ต่อร่วมกันอยู่ส่งสัญญาณข้อมูลกลับมายังคอมพิวเตอร์ (Request to Sent : RTS) โดยที่โปรแกรมวิชวลเบสิค 6.0 จะมีตัวบ่งชี้ของการเกิดการเปลี่ยนแปลงที่พอร์ตอนุกรมได้หลากหลายกรณี

ข) ลักษณะการสื่อสารด้วยการคอยตรวจสอบเหตุการณ์ (Polling) จะทำหน้าที่ตรวจสอบสถานะของเหตุการณ์และค่าความผิดพลาดที่จะเกิดขึ้น (Comm Event) ของการติดต่อพอร์ตอนุกรม

ค) สรุปการใช้งานพอร์ตอนุกรม

1) กำหนดหมายเลขพอร์ตลงในส่วนของคุณสมบัติ เช่น Mscomm1.Commport=1: เป็นการใช้พอร์ต 1 ของคอมพิวเตอร์ เป็นต้น หมายเลขซึ่งจะเป็นตัวเลือกพอร์ตที่จะใช้งาน

2) กำหนดค่าลงในส่วนของคุณสมบัติซึ่งจะเป็นค่าเพื่อที่ใช้ในการติดต่อของพอร์ตอนุกรม ประกอบด้วยความเร็วในการส่งข้อมูลบิตพาริตี้ จำนวนบิตหยุดและรูปแบบที่ใช้สำหรับการกำหนดค่าดังตัวอย่างต่อไปนี้ Mscomm.Setting = 9600, N, 8, 1; จะมีความหมาย คือความเร็วในการส่งข้อมูลเป็นแบบ 8 บิต 1 บิต หยุด

3) กำหนดค่าให้กับ InputLen ซึ่งเป็นการอ่านค่าการอ่านข้อมูลของพอร์ตอนุกรมให้กำหนดค่าในส่วน PROPERTY ดังนี้ MsComm.InputLen = 0 เป็นการอ่านค่าทั้งหมดที่เข้ามาทางอินพุต

4) กำหนดค่าสำหรับการใช้งานพอร์ต ในส่วนของคุณสมบัติ ดังนี้

MsComm1.PortOpen = True; เป็นการเปิดใช้งานพอร์ตอนุกรม

MsComm1.PortOpen = False; เป็นการปิดใช้งานพอร์ตอนุกรม

5) ในการเขียนโปรแกรมรับค่าพอร์ตอนุกรมใช้คุณสมบัติทางด้านอินพุตโดยเขียนดังนี้ Data = MsComm1.Input; เป็นการรับค่าจากพอร์ตอนุกรมมาเก็บไว้ในตัวแปรชื่อ Data

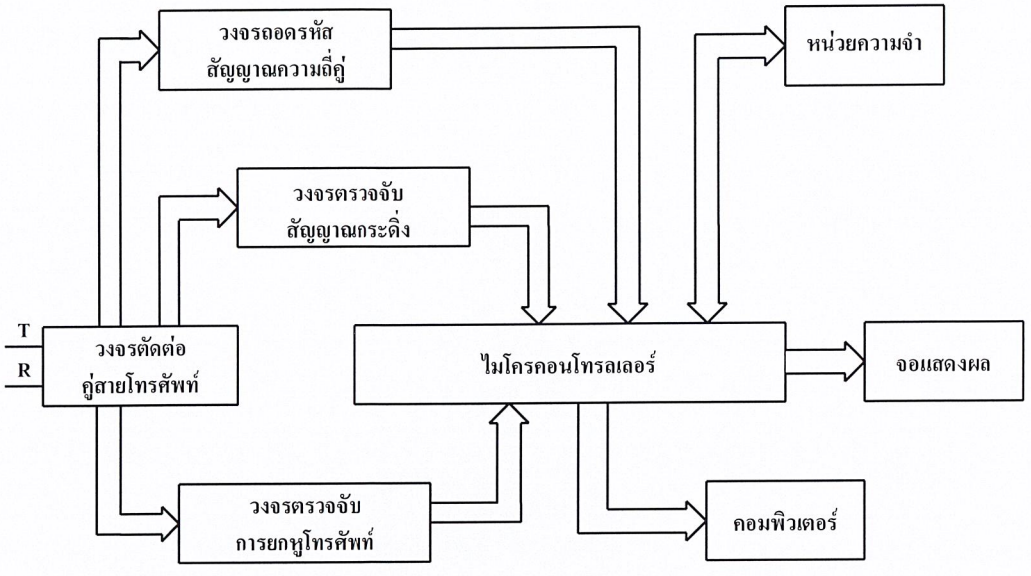
6)ในการเขียนโปรแกรมรับส่งค่าไปยังพอร์ตอนุกรมใช้คุณสมบัติทางด้านเอาต์พุต โดยเขียนดังนี้ MsComm1.Output = Chr$ (Data); เป็นการรับส่งค่าข้อมูลที่อยู่ภายในตัวแปรชื่อ Data ไปยังพอร์ตอนุกรม

การใช้งานพอร์ตอนุกรมด้วยโปรแกรมวิชวลเบสิค 6.0 (Visual Basic 6.0) ค่อนข้างง่ายและสะดวก ข้อมูลในการใช้งานมีคุณสมบัติ (Property) อยู่หลายตัว แต่ที่ใช้งานบ่อยๆ และจำเป็นจริงๆ มีไม่กี่ตัว แต่ทั้งนี้ทั้งนั้นจะต้องมีการศึกษารายละเอียดให้เข้าใจก่อนที่จะนำมาใช้งานทางด้านวิศวกรรมเพื่อจะได้เกิดประโยชน์สูงสุด

บทที่ 3

การออกแบบ การสร้าง และการทำงาน

3.1 แผนผังการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ



รูปที่ 3.1 แผนผังการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

จากรูปที่ 3.1 เป็นแผนผังการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติโดยประกอบด้วยส่วนต่างๆ ดังนี้

1) วงจรแหล่งจ่ายไฟ

2) วงจรทวีแรงดัน

3) วงจรถอดรหัสสัญญาณความถี่คู่

4) วงจรตรวจจับสัญญาณกระดิ่ง

5) วงจรตรวจจับการยกหูโทรศัพท์

6) วงจรตัดต่อคู่สายโทรศัพท์

7) วงจรตั้งเวลาระงับการโทรออกชั่วคราว

8) วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

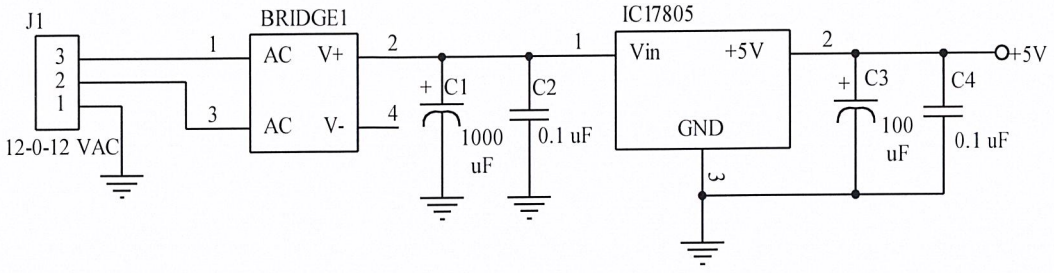
9) วงจรไมโครคอนโทรลเลอร์

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

3.2 การออกแบบวงจร

3.2.1 วงจรแหล่งจ่ายไฟ

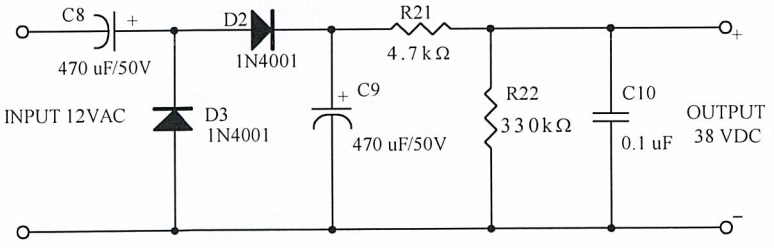
รูปที่ 3.2 เป็นวงจรแหล่งจ่ายไฟ 5 โวลต์ ซึ่งทำการจ่ายไฟให้กับวงจรตรวจจับสัญญาณโทรศัพท์และจ่ายไฟให้กับวงจรไมโครคอนโทรลเลอร์ โดยมีไดโอดบริดจ์เป็นวงจรเรียงกระแส ทำหน้าที่แยกไฟบวกและลบออกจากกันโดยมีตัวเก็บประจุ ทำหน้าที่กรองกระแสและแรงดันให้เรียบ และใช้ไอซีเร็กกูเรเตอร์ เบอร์ 7805 เป็นตัวรักษาระดับของแรงดันให้คงที่ในระดับแรงดัน 5 โวลต์



รูปที่ 3.2 วงจรแหล่งจ่ายไฟ

3.2.2 วงจรทวีแรงดัน

เป็นวงจรที่ทำหน้าที่สร้างแรงดันคู่สายเลี้ยงให้แก่โทรศัพท์ เพื่อใช้ในการกดหมายเลขของโทรศัพท์ขณะที่วงจรถูกปลดสายออกจากคู่สายขององค์การโทรศัพท์ โดยวงจรทวีแรงดันนี้จะป้อนอินพุตเป็นแรงดันไฟสลับ 12 โวลต์ แรงดันไฟตรงที่เอาท์พุตก็จะได้ประมาณ 38 โวลต์



รูปที่ 3.3 วงจรทวีแรงดัน

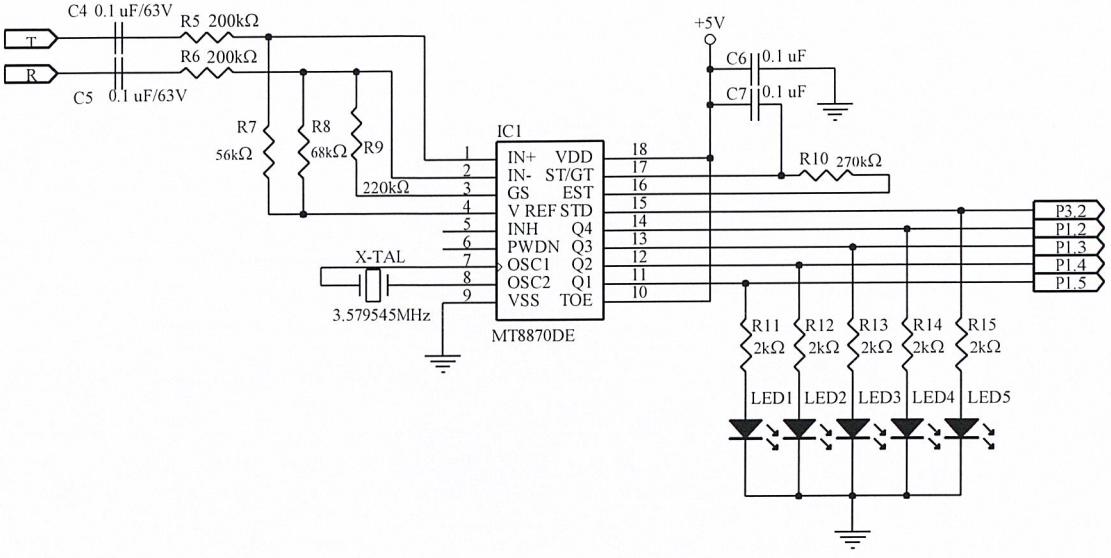
จากรูปที่ 3.3 เป็นวงจรทวีแรงดัน การทำงานประกอบด้วยวงจรแคลมป์ (C8 และ D3) ซึ่งจะให้เอาท์พุตเป็นแรงดันไฟสลับ ที่มีค่าสูงสุดเท่ากับค่าแรงดันจากยอดถึงยอดของสัญญาณอินพุต

51034

และวงจรดีเท็กค์ค่าแรงดันยอด (D2 และ C9) ซึ่งจะให้เอาท์พุตเป็นแรงดันไฟตรงที่มีค่าเท่ากับแรงดันสูงสุดของแรงดันอินพุตที่ D2

3.2.3 วงจรถอดรหัสสัญญาณความถี่คู่

วงจรถอดรหัสสัญญาณความถี่คู่ ทำหน้าที่ถอดรหัสสัญญาณความถี่คู่ที่มาจากการกดปุ่มเพื่อการโทรออกหรือเลือกการทำงานให้เป็นรหัสไบนารี่ขนาด 4 บิต ก่อนส่งให้ไมโครคอนโทรลเลอร์เพื่อทำการประมวลผลการทำงานในขั้นต่อไป



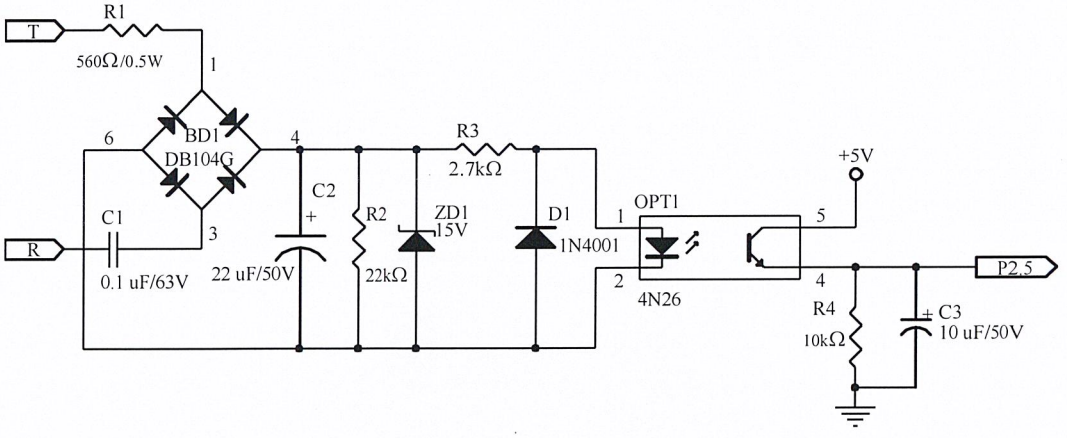
รูปที่ 3.4 วงจรถอดรหัสสัญญาณความถี่คู่

จากรูปที่ 3.4 เมื่อมีสัญญาณความถี่คู่เข้ามา C4 และ C5 จะเป็นตัวส่งผ่านสัญญาณแล้วส่งผ่าน R5 และ R6 มาเข้าที่ขาอินพุตของไอซี MT8870 (IC1) ซึ่งเป็นอุปกรณ์ที่สำคัญของวงจรส่วน R5 กับ R9 เป็นตัวกำหนดอัตราขยายของออปแอมป์ที่อยู่ภายในไอซี ส่วนตัวความต้านทาน R7 กับ R8 เป็นตัวกำหนดแรงดันอ้างอิงและขา OSC1 และ OSC2 จะต่อกับแร่คริสตอลขนาด 3.579545 เมกะเฮิรตซ์ เพื่อกำเนิดความถี่เป็นสัญญาณนาฬิกาให้กับไอซีและวงจรจะมีการตรวจสอบช่วงความถี่ที่เข้ามาว่ามีระยะเวลาตามที่กำหนดหรือไม่ โดยสามารถสังเกตได้จากระยะเวลาการกดปุ่มโทรศัพท์ ซึ่งจะต้องมีช่วงเวลานานพอสมควร คือให้ระยะเวลานานเท่ากันหรือมากกว่าช่วงเวลาที่ตั้งไว้ จึงจะยอมรับและถือว่าสัญญาณนั้นถูกต้อง โดยที่ขา EST จะเป็น “1” นานใกล้เคียงกับระยะเวลาที่มีความถี่คู่ซึ่งทำให้แรงดันที่ขา ST/GT สูงขึ้นจนถึงค่าเทรสโฮลด์ วงจรถอดรหัสจึงจะถอดรหัสออกมาเป็นตัวเลขขนาด 4 บิตทางขา Q1, Q2, Q3 และ Q4 แต่ถ้าเปรียบเทียบกันแล้ว

ช่วงเวลาน้อยกว่าที่ตั้งไว้ ก็จะไม่มีการถอดรหัสเป็นตัวเลขขนาด 4 บิต ซึ่งช่วงเวลานี้เราสามารถตั้งได้โดยการกำหนดค่าของ R10 และ C7 ส่วนขา TOE นั้นจะทำงานคล้าย ๆ กับเป็นขา Enable คือ ในสภาวะปกติไม่มีสัญญาณเอาต์พุตจะทำให้ขาเอาต์พุตนี้มีค่าอิมพีแดนซ์สูงมาก ทำให้สัญญาณต่างๆ ไม่สามารถผ่านขา Q ได้ แต่เมื่อมีสัญญาณเอาต์พุตมาจะมีอิมพีแดนซ์ต่ำลง แล้วจึงส่งสัญญาณเอาต์พุตออกไปได้ สำหรับขา STD ในวงจรนี้ก็เป็นขาเอาต์พุตอีกขาหนึ่ง

3.2.4 วงจรตรวจจับสัญญาณกระดิ่ง

เป็นวงจรที่ใช้ในการตรวจจับสัญญาณกระดิ่ง ซึ่งจะสามารถทำได้โดยการตรวจจับการเกิดแรงดันไฟสลับที่เกิดขึ้นชั่วขณะหนึ่ง ซึ่งก็คือสัญญาณกระดิ่งนั่นเอง



รูปที่ 3.5 วงจรตรวจจับสัญญาณกระดิ่ง

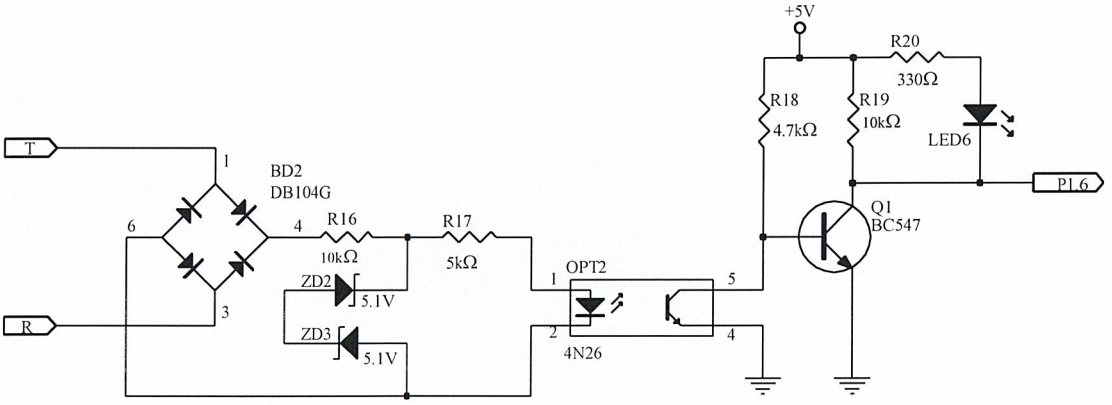
จากรูปที่ 3.5 วงจรตรวจจับสัญญาณกระดิ่งจะใช้ออปโต้ทรานซิสเตอร์เป็นหลัก สัญญาณจากคู่สายโทรศัพท์จะต่อเข้าวงจรโดยผ่านตัวความต้านทาน R1 ทำหน้าที่ลดขนาดของแรงดันไฟฟ้าของสัญญาณกระดิ่งที่เข้ามาและตัวเก็บประจุ C1 ทำหน้าที่กั้นแรงดันไฟตรงจากคู่สายโทรศัพท์ไม่ให้ผ่านเข้าสู่วงจรได้ ไดโอดบริดจ์จะทำการเปลี่ยนระดับแรงดันของสัญญาณกระดิ่งให้เป็นแรงดันไฟตรงโดยมีตัวเก็บประจุ C2 และตัวซีเนอร์ไดโอด ZD1 ทำหน้าที่รักษาระดับแรงดันให้คงที่ ตัวความต้านทาน R2 กับ R3 ร่วมกันทำหน้าที่เป็นวงจรดิไวเดอร์หรือลดขนาดแรงดันไฟตรงที่ผ่านมาทางไดโอดบริดจ์ให้ต่ำลง ซึ่งตัวความต้านทาน R2 ยังทำหน้าที่เป็นตัวดิสชาร์จให้กับตัวเก็บประจุ C2 ช่วงที่ไม่มีสัญญาณกระดิ่งเข้ามาอีกด้วย ไดโอด D1 ทำหน้าที่ป้องกันแรงดันย้อนกลับ อันเป็นเหตุให้แอลอีดี ภายในออปโตโอโซเลเตอร์เสียหายได้

เมื่อสัญญาณกระดิ่งเข้ามา วงจรบริดจ์จะทำการเปลี่ยนระดับแรงดันของสัญญาณกระดิ่งให้เป็นระดับแรงดันไฟตรง ป้อนเข้าขาแอโนดของแอลอีดีภายในออปโตโอโซเลเตอร์ที่ขา 1 เมื่อ

แอลอีดีภายในทำงานก็จะส่งผลให้ทรานซิสเตอร์ภายในทำงานด้วยเช่นกัน ทำให้ขา 4 ของออปโตโอโซเลเตอร์หรือขา E ของทรานซิสเตอร์ภายในถูกต่อกับแหล่งจ่ายไฟ 5 โวลต์ แรงดันไฟเอาต์พุตที่ขา 4 จาก 0 โวลต์ จะมีค่าเกือบเท่าแหล่งจ่าย วิธีการทำงานทั้งหมดจะเป็นตามจังหวะของสัญญาณกระดิ่งที่เข้ามา

3.2.5 วงจรตรวจจับการยกหูโทรศัพท์

เป็นวงจรที่ใช้ในการตรวจจับการยกหูและวางหูโทรศัพท์ โดยใช้หลักการตรวจสอบแรงดัน ซึ่งอาศัยความแตกต่างของแรงดันในขณะทำการยกหูและวางหูโทรศัพท์ ซึ่งแรงดันขณะที่ยกหู (Off Hook) มีค่าแรงดันอยู่ประมาณ 8 ถึง 12 โวลต์และแรงดันขณะที่วางหูโทรศัพท์ (On Hook) จะมีอยู่ประมาณ 48 ถึง 52 โวลต์ทั้งนั้นขนาดของแรงดัน จะขึ้นอยู่กับอิมพีแดนซ์ของเครื่องโทรศัพท์แต่ละเครื่องและระยะทางของคู่สายโทรศัพท์



รูปที่ 3.6 วงจรตรวจจับการยกหูโทรศัพท์

จากรูปวงจรในรูปที่ 3.6 แรงดันไฟตรงประมาณ 48 ถึง 52 โวลต์ จากคู่สายโทรศัพท์จะถูกจัดขั้วแรงดันใหม่ โดยบริดจ์ไดโอด BD2 แล้วผ่านตัวความต้านทาน R16 และตัวซีเนอร์ไดโอด ZD2, ZD3 เพื่อลดขนาดของแรงดันให้ลดลงและจำกัดแรงดันไว้ประมาณ 10 โวลต์ จากนั้นก็ผ่านตัวความต้านทาน R17 เข้าขาแอโนดของ LED ที่อยู่ภายในออปโตโอโซเลเตอร์ OPT2 ทำงานส่งผลให้ทรานซิสเตอร์ Q1 ไม่ทำงานเนื่องจากขา B ของ Q1 ถูกต่อลงกราวด์ผ่านขา C และ E ของทรานซิสเตอร์ Q1 วัดได้ประมาณ 0 โวลต์ ส่งผลให้แรงดันเอาต์พุตที่ขา C สูงเกือบเท่าแหล่งจ่ายหรือประมาณ 5 โวลต์ LED6 ซึ่งไม่ทำงาน

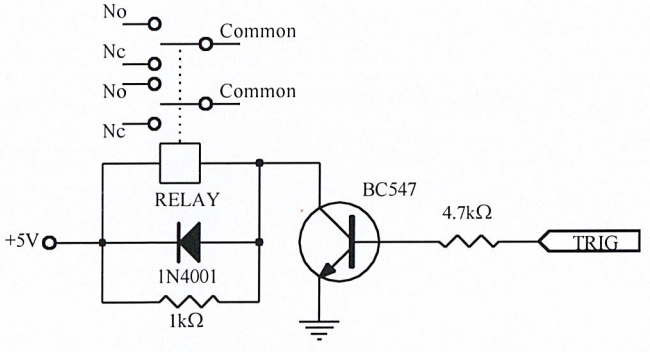
เมื่อมีการยกหูเกิดขึ้น จะทำให้แรงดันที่คู่สายตกลงเหลือประมาณ 8 ถึง 12 โวลต์ ซึ่งแรงดันจากคู่สายโทรศัพท์จะถูกจัดขั้วแรงดันใหม่โดยบริดจ์ไดโอด BD2 แล้วผ่านตัวความต้านทาน R16

และตัวซีเนอร์ไดโอด ZD2 และ ZD3 เพื่อลดแรงดันให้ต่ำลงเช่นเดียวกัน แต่แรงดันที่ลดขนาดลงนี้ไม่เพียงพอที่จะทำให้ออปโตโอโซเลเตอร์ทำงานได้ ส่งผลให้ทรานซิสเตอร์ Q1 ทำงานเนื่องจากที่ขา B ได้รับแรงดันผ่านมาทางตัวความต้านทาน R18 ทำให้เกิดกระแสไหลผ่านขา C และ E ลงกราวด์ ดังนั้นแรงดันเอาต์พุตขา C ของทรานซิสเตอร์ Q1 จึงตกลงเหลือประมาณ 0 โวลต์ สังเกต LED6 จะติดสว่าง

3.2.6 วงจรตัดต่อคู่สายโทรศัพท์

จากการออกแบบวงจรเพื่อให้สอดคล้องกับการเขียนโปรแกรม โดยอาศัยหลักการตัดต่อหน้าสัมผัสของรีเลย์ โดยใช้รีเลย์แบบ 2 หน้าสัมผัส

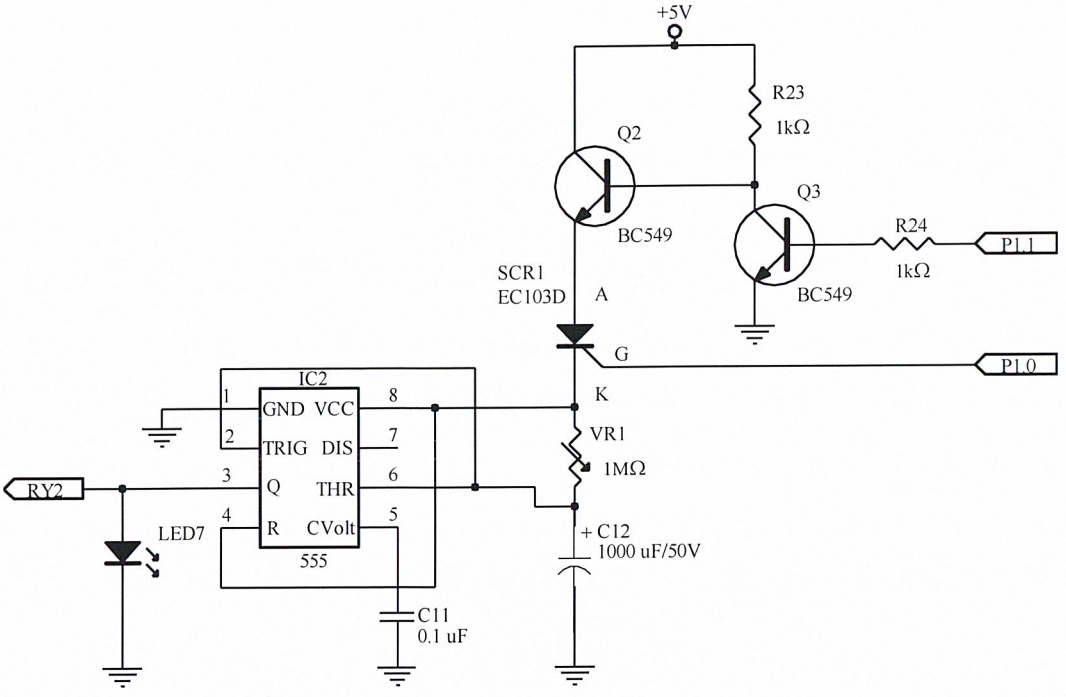
หลักการทำงานของชุดตัดรีเลย์ คือ สภาวะปกติ จะต่ออยู่กับคู่สายโทรศัพท์ แต่เมื่อมีการยกหูขึ้นไมโครคอนโทรลเลอร์ จะส่งแรงดันไฟ 5 โวลต์ ไปกระตุ้นขาเบสของทรานซิสเตอร์ให้ทำงาน ทำให้หน้าสัมผัสของรีเลย์ตัดออกจากคู่สาย ต่อกับคู่สายก็ได้ โดยควบคุมจากไมโครคอนโทรลเลอร์



รูปที่ 3.7 วงจรตัดต่อคู่สายโทรศัพท์

3.2.7 วงจรตั้งเวลาระงับการโทรออกชั่วคราว

วงจรจะทำงานในกรณีที่มีกดรหัสผ่านผิด ไมโครคอนโทรลเลอร์จะส่งสัญญาณไปกระตุ้นขาเกตของเอสซีอาร์ SCR1 ทำให้วงจรตั้งเวลาทำงานโดยใช้ไอซี 555 ต่อแบบโมโนสเตเบิล ตัวความต้านทานปรับค่าได้ VR1 และตัวเก็บประจุ C12 จะเป็นตัวกำหนดเวลา โดยตั้งเวลาไว้ที่ประมาณ 20 นาที เพื่อให้วงจรตัดต่อคู่สายโทรศัพท์ตัดคู่สายออกจากโทรศัพท์ตามเวลาที่ตั้ง แต่ถ้าหากผู้ดูแลระบบทำการเคลียร์เวลาที่ตั้งเอาไว้ก่อน 20 นาที ไมโครคอนโทรลเลอร์ก็จะส่งสัญญาณไปกระตุ้นที่ขาเบสของทรานซิสเตอร์ Q3 ให้ทำงาน ทำให้เป็นการตัดแหล่งจ่ายวงจรตั้งเวลา วงจรจึงหยุดทำงาน

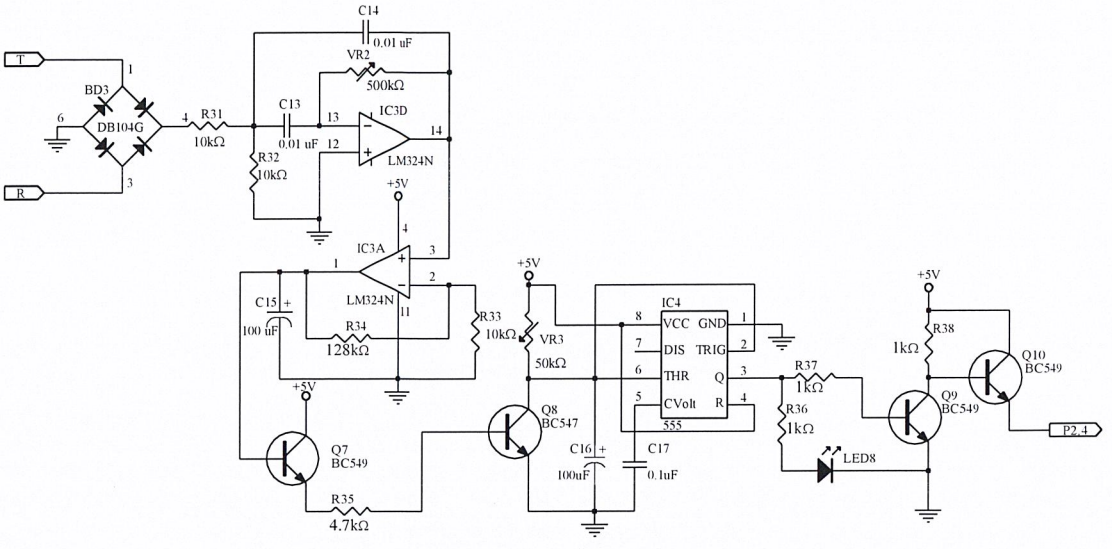


รูปที่ 3.8 วงจรตั้งเวลาระงับการโทรออกชั่วคราว

3.2.8 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

เป็นวงจรส่วนที่ใช้ในการควบคุมการตรวจจับสัญญาณกระดิ่งตอบกลับในคู่สาย โดยใช้ไอซีออปแอมป์เบอร์ LM324N ต่อวงจรกรองความถี่แบบแบนด์พาสฟิลเตอร์ กำหนดความถี่เรโซแนนซ์ที่ 400 เฮิรตซ์โดยใช้หลักการที่ว่าสัญญาณกระดิ่งตอบกลับมีความถี่ 400 เฮิรตซ์ดังนั้น วงจรแบนด์พาสจึงทำหน้าที่กรองความถี่ 400 เฮิรตซ์ของสัญญาณกระดิ่งตอบกลับออกมา แต่เนื่องจากสัญญาณเสียงพูดของมนุษย์ซึ่งก็มีความถี่ 400 เฮิรตซ์ออกมาเมื่อมีการพูดเหมือนกัน วงจรแบนด์พาสที่กรองความถี่ 400 เฮิรตซ์นี้ก็สามารถที่จะถอดความถี่ 400 เฮิรตซ์ของเสียงพูดออกมาได้เช่นกัน แต่ความแตกต่างของสัญญาณระหว่างสัญญาณกระดิ่งตอบกลับกับเสียงพูด อยู่ที่ความต่อเนื่องของความถี่ 400 เฮิรตซ์ จากจุดนี้จึงทำให้เราสามารถแยกแยะสัญญาณเสียงพูดกับสัญญาณความถี่ 400 เฮิรตซ์ของสัญญาณกระดิ่งตอบกลับได้จากวงจรเอาต์พุตของการกรองความถี่จาก ไอซี LM324N จะต่อเข้าไปยังขาเบสของทรานซิสเตอร์ Q7 โดยมี C15 เป็นตัวตรวจสอบความต่อเนื่องของสัญญาณจากการกรองความถี่ โดยเอาต์พุตจากขา E ของทรานซิสเตอร์ Q7 จะเป็นตัวกำหนดจังหวะการเรียกของสัญญาณกระดิ่งตอบกลับ โดยมีไอซี 555 ตั้งเวลาช่วงเวลาเกิดสัญญาณกระดิ่งตอบกลับ ซึ่งจังหวะการส่งสัญญาณกระดิ่งตอบกลับจากองค์การโทรศัพท์จะเป็นลักษณะมีสัญญาณความถี่ 400 เฮิรตซ์ 2 วินาทีและหยุด 4 วินาที สลับกันไป โดยหลักการตรวจสอบว่ามีการรับสาย

ปลายทางหรือยัง จากวงจรตรวจจับสัญญาณกระดิ่งตอบกลับนี้ คือ ต้องมีสัญญาณกระดิ่งตอบกลับครั้งแรกก่อนและต้องมีครั้งต่อมาภายในเวลา 4 วินาที จะถือว่าปลายทางได้รับสายแล้วเอาต์พุตของไอซี 555 ก็จะส่งออกไปยังโปรแกรมเพื่อให้เริ่มบันทึก

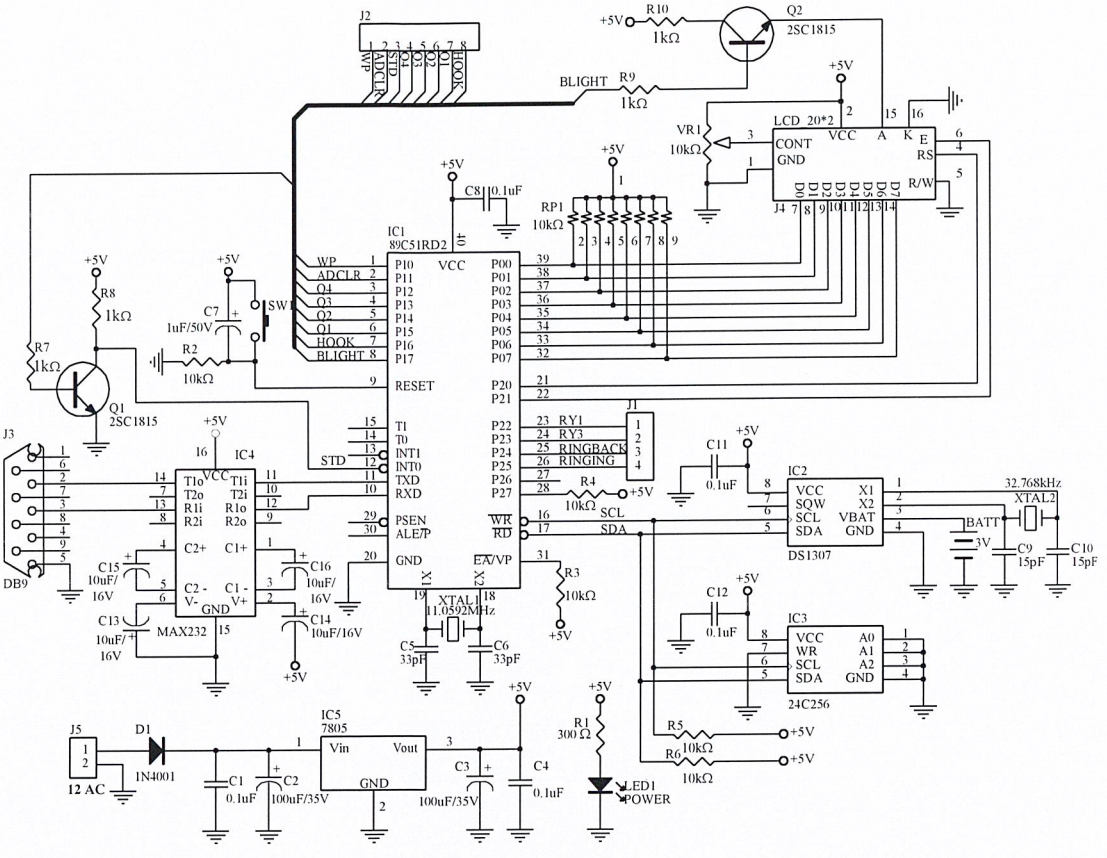


รูปที่ 3.9 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

3.2.9 วงจรไมโครคอนโทรลเลอร์

วงจรไมโครคอนโทรลเลอร์ เป็นวงจรส่วนที่ใช้ในการควบคุมการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติทั้งภาครับ ภาคโทรออก รวมถึงการแสดงผลข้อมูลการใช้งานเครื่องโทรศัพท์ผ่านทางหน้าจอคอมพิวเตอร์

จากรูปที่ 3.10 เป็นวงจรไมโครคอนโทรลเลอร์ ประกอบด้วย IC1 เบอร์ 89C51RD2 เป็นหน่วยประมวลผลกลางใช้ในการควบคุมการทำงานของเครื่องทั้งหมด IC2 เบอร์ DS1307 เป็นไอซีฐานเวลา ใช้ในการแสดงผล วัน เวลาปัจจุบัน IC3 เบอร์ 24C256 เป็นหน่วยความจำแบบ EEPROM ขนาดความจุ 32 กิโลไบต์ ใช้ในการบันทึกข้อมูล ส่วน IC4 เบอร์ MAX232 ใช้สำหรับส่งผ่านข้อมูลแบบอนุกรมไปยังคอมพิวเตอร์ ในส่วนของแอลซีดี เป็นแบบ 20 ตัวอักษร 2 บรรทัด ใช้ในการแสดงผลการทำงานของเครื่อง



รูปที่ 3.10 วงจรไมโครคอนโทรลเลอร์

3.3 ส่วนของซอฟต์แวร์

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ จะทำงานโดยมีซอฟต์แวร์ในการควบคุมการทำงาน การทำงานในส่วนการโทรออก เมื่อมีการยกหู เครื่องจะแสดงข้อความที่จอแอลซีดี เพื่อให้กดหมายเลขรหัสผ่านจำนวน 4 หมายเลข เมื่อเครื่องตรวจสอบรหัสผ่านถูกต้องแล้ว ก็สามารถโทรออกได้ตามปกติ เมื่อปลายทางรับสาย เครื่องจะทำการบันทึกรหัสผ่านผู้ใช้ หมายเลขโทรศัพท์ เวลา วัน เดือน ปี และจำนวนนาทีในการใช้โทรศัพท์ ถ้าต้องการกดโทรออกหมายเลขพิเศษ ให้กดเครื่องหมาย # แล้วตามด้วยหมายเลขพิเศษที่ต้องการโทรออก ส่วนการทำงานในภาครับ เมื่อมีผู้โทรเข้าจะสามารถรับสายเรียกเข้าได้ปกติ โดยไม่ต้องกดรหัสผ่าน

3.4 การทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัตินี้มีการทำงานเป็นส่วนๆ โดยขึ้นอยู่กับการใช้งาน ซึ่งสามารถแบ่งเป็นกรณีต่างๆ ดังนี้

3.4.1 กรณีการโทรออก

เมื่อมีการยกหูโทรศัพท์ โปรแกรมจะแสดงข้อความที่จอแสดงผล เพื่อให้ผู้ใช้กดหมายเลขรหัสผ่านส่วนตัว เมื่อกดครบ 4 หลัก โปรแกรมจะทำการตรวจสอบว่าถูกต้องหรือไม่ ถ้ากดรหัสผ่านผิดโปรแกรมจะให้กดรหัสผ่านใหม่ โดยถ้ากดรหัสผ่านผิดถึง 3 ครั้ง เครื่องจะทำงานเข้าสู่ระบบการป้องกันการโทรออกอัตโนมัติ แต่ถ้ากดรหัสผ่านถูกต้องผู้โทรก็จะสามารถโทรออกได้ตามปกติ เมื่อปลายทางรับสายเครื่องจะทำการบันทึกรหัสผ่านผู้ใช้ หมายเลขโทรศัพท์ เวลา วัน เดือน ปี และจำนวนนาทีในการโทรออกแต่ละครั้ง เมื่อมีการวางหูสิ้นสุดการสนทนา ก็จะกลับสู่การเริ่มต้นการทำงานใหม่ คือ รอการยกหู

3.4.2 กรณีการโทรเข้า

เมื่อมีการโทรเข้ามา จะมีสัญญาณกระดิ่งเข้ามาที่วงจรตรวจสอบสัญญาณกระดิ่ง เครื่องจะยอมให้รับสายได้ตามปกติ โดยไม่ต้องผ่านระบบรหัสผ่าน

3.4.3 กรณีของผู้ดูแลระบบ

ผู้ดูแลระบบจะมีรหัสผ่านส่วนตัวนอกเหนือจากรหัสผ่านผู้ใช้ ซึ่งใช้ในกรณีของการเคลียร์โหมดระงับการโทรออกชั่วคราว การเปลี่ยนแปลงรหัสผ่านผู้ใช้ใหม่ และการตั้งเวลา โดยจะมีการแสดงเมนูให้เลือก และแสดงผลแต่ละขั้นตอนการทำงานผ่านทางจอแอลซีดี

3.4.4 กรณีการเชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์

เมื่อเครื่องเก็บข้อมูลการโทรออกแล้ว การแสดงผลข้อมูลการโทรจะสามารถแสดงได้โดยผ่านทางหน้าจอคอมพิวเตอร์ แล้วสามารถพิมพ์รายการข้อมูลการโทรออกผ่านทางปริ้นเตอร์ เพื่อใช้ในการตรวจสอบคิดเงินค่าโทร

บทที่ 4

การทดลองและผลการทดลอง

4.1 กล่าวนำ

ในบทนี้กล่าวถึงการทดลองและผลการทดลอง การทำงานของวงจรหลักต่างๆ ในแต่ละส่วนได้แก่ วงจรถอดรหัสสัญญาณความถี่คู่ วงจรตรวจจับสัญญาณกระดิ่ง วงจรตรวจจับการยกหูโทรศัพท์ วงจรตัดต่อคู่สายโทรศัพท์ วงจรตั้งเวลาระงับการโทรออกชั่วคราว วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ วงจรไมโครคอนโทรลเลอร์และการทดลองการทำงานของเครื่องบันทึกข้อมูลการใช้งานโทรศัพท์อัตโนมัติ

4.2 การทดลองส่วนของวงจรถอดรหัสสัญญาณความถี่คู่

4.2.1 ลำดับขั้นการทดลอง

1) ประกอบวงจรถอดรหัสสัญญาณความถี่คู่ ตามรูปที่ 4.1

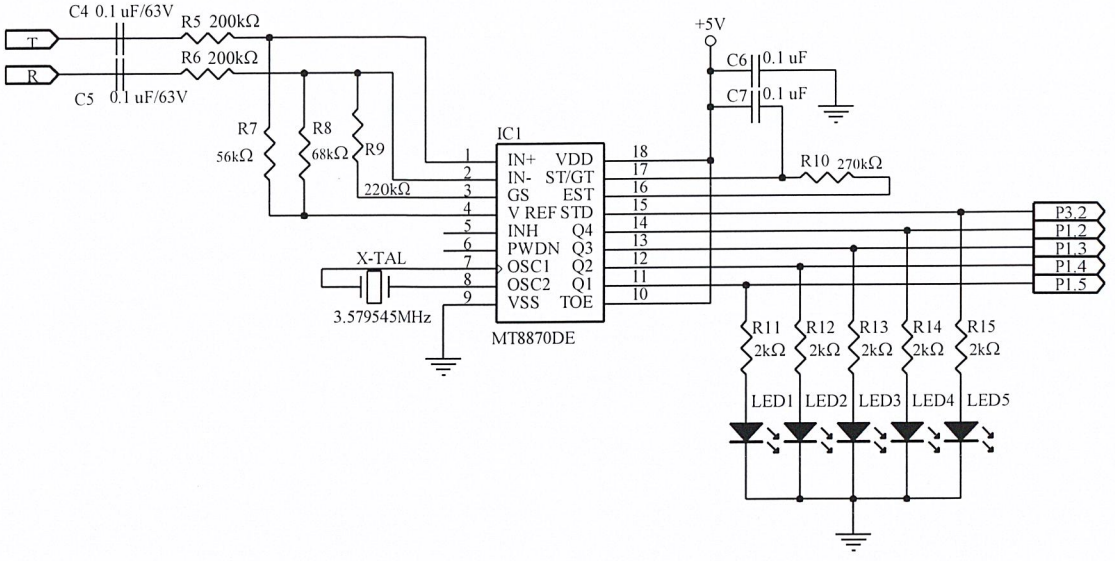
2) ตรวจสอบการต่อวงจรเพื่อความถูกต้อง แล้วทำการจ่ายไฟเลี้ยงให้วงจร

3) ต่อคู่สายโทรศัพท์เข้ากับวงจรถอดรหัสสัญญาณความถี่คู่ที่ใช้ทดลอง

4) กดหมายเลขโทรศัพท์เข้ามายังคู่สายที่ต่ออยู่กับวงจร ซึ่งจะเห็นได้ว่า แอลอีดีสว่างขึ้นเป็นเลขฐานสองขนาด 4 บิตตามค่าหมายเลขที่กดเข้ามา ส่วน STD จะสว่างทุกครั้งที่มีการกดปุ่มใดๆ ของโทรศัพท์ และ Q1-Q4 ก็จะมีสภาวะค้างอยู่จนกว่ากดหมายเลขต่อไป

4.2.2 ผลการทดลอง

จากการทดลองเมื่อต่อวงจรเสร็จแล้ว ให้ทดลองกดเลขหมายที่หน้าปัทม์โทรศัพท์ จะเห็นแอลอีดีที่ต่ออยู่ที่ขาเอาต์พุต Q ของไอซี MT8870 สว่างขึ้นเป็นระบบเลขฐานสองขนาด 4 บิต ซึ่งเมื่อแปลงเป็นเลขฐานสิบแล้วจะได้ค่าตรงตามค่าที่กดเลขหมายมา ทำให้ทราบว่าผู้เรียกกดเลขหมายใดๆ ดังแสดงในตารางที่ 4.1



รูปที่ 4.1 วงจรถอดรหัสสัญญาณความถี่คู่

ตารางที่ 4.1 ผลการทดลองวงจรถอดรหัสสัญญาณความถี่คู่

เลขหมายหน้าปัทม์โทรศัพท์	เอาต์พุต					เลขฐานสิบ
	Q4	Q3	Q2	Q1	STD	
1	0	0	0	1	1	1
2	0	0	1	0	1	2
3	0	0	1	1	1	3
4	0	1	0	0	1	4
5	0	1	0	1	1	5
6	0	1	1	0	1	6
7	0	1	1	1	1	7
8	1	0	0	0	1	8
9	1	0	0	1	1	9
0	1	0	1	0	1	10
*	1	0	1	1	1	11
#	1	1	0	0	1	12

4.3 การทดลองส่วนของวงจรตรวจจับสัญญาณกระดิ่ง

4.3.1 ลำดับขั้นการทดลอง

1) ประกอบวงจรตรวจจับสัญญาณกระดิ่ง ตามรูปที่ 4.2

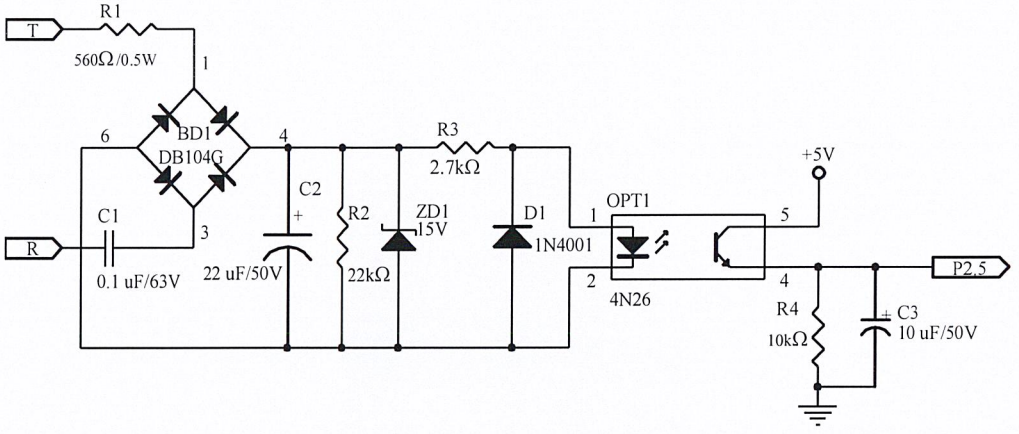
2) ตรวจสอบการต่อวงจร เพื่อความถูกต้อง

3) ทำการทดลองโดยจ่ายไฟตรง 5 โวลต์ แล้วทำการวัดแรงดันเอาต์พุต

4) ต่อคู่สายโทรศัพท์ เข้ากับวงจรตรวจจับสัญญาณกระดิ่ง

5) ทดลองโทรเข้ามายังหมายเลขที่ได้ทำการต่อตรวจสอบสัญญาณกระดิ่ง

6) วัดแรงดันเอาต์พุตขณะที่มีสัญญาณกระดิ่งเข้ามา



รูปที่ 4.2 วงจรตรวจจับสัญญาณกระดิ่ง

4.3.2 ผลการทดลอง

เมื่อทำการวัดค่าแรงดันเอาต์พุตในขณะที่ไม่มีสัญญาณกระดิ่งเอาต์พุตจะได้ 0 โวลต์ และเอาต์พุตขณะที่มีสัญญาณกระดิ่งได้แรงดันเอาต์พุตที่พอร์ต P2.5 เท่ากับ 4.7 โวลต์ เป็นจังหวะตามสัญญาณกระดิ่ง

4.4 การทดลองส่วนของวงจรตรวจจับการยกหูโทรศัพท์

4.4.1 ลำดับขั้นการทดลอง

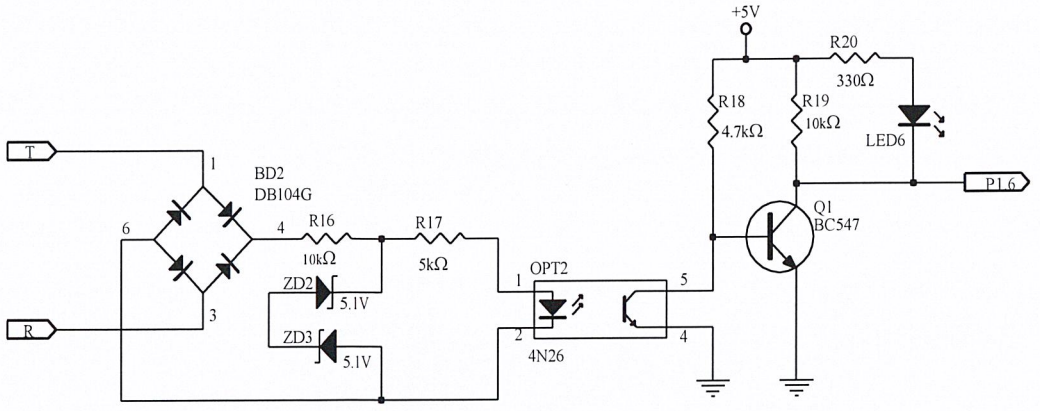
1) ประกอบวงจรตรวจจับการยกหูโทรศัพท์ ตามรูปที่ 4.3

2) ตรวจสอบการต่อวงจร เพื่อความถูกต้อง

3) จ่ายไฟเลี้ยง 5 โวลต์ ให้กับวงจร

4) ต่อคู่สายโทรศัพท์เข้ากับวงจร วัดค่าแรงเอาต์พุต

5) ทดลองยกหูโทรศัพท์ วัดค่าแรงดัน



รูปที่ 4.3 วงจรตรวจจับการยกหูโทรศัพท์

4.4.2 ผลการทดลอง

เมื่อทำการต่อคู่สายโทรศัพท์ เข้ากับวงจรตรวจจับการยกหูโทรศัพท์ วัดค่าแรงดันเอาต์พุตที่พอร์ต P1.6 จะได้ประมาณ 5 โวลต์ และเมื่อทดลองยกหูโทรศัพท์ แล้วทำการวัดค่าแรงดันเอาต์พุตจะได้แรงดันประมาณ 0 โวลต์

4.5 การทดลองส่วนของวงจรตัดต่อคู่สายโทรศัพท์

4.5.1 ลำดับขั้นการทดลอง

1) ประกอบวงจรตัดต่อคู่สายโทรศัพท์ ตามรูปที่ 4.4

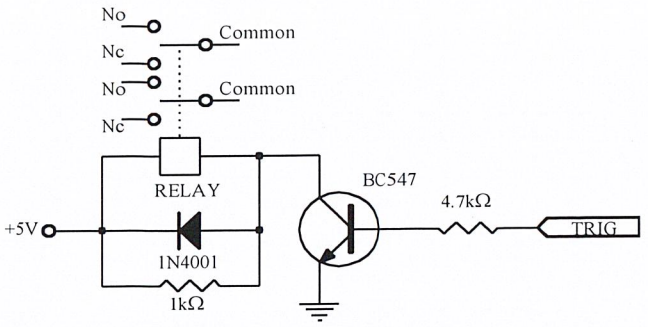
2) ตรวจสอบการต่อวงจร เพื่อความถูกต้อง

3) จ่ายไฟเข้าวงจรและทดลองกระตุ้นที่ขาเบสให้ทรานซิสเตอร์ทำงาน

4) ดูผลการเปลี่ยนแปลงของหน้าสัมผัสรีเลย์

4.5.2 ผลการทดลอง

เมื่อทำการกระตุ้นขาเบสให้ทรานซิสเตอร์ทำงานด้วยแรงดันไฟ 5 โวลต์ ซึ่งหน้าสัมผัสของรีเลย์ก็จะตัดต่อวงจรกับคู่สายโทรศัพท์ได้ตามต้องการ



รูปที่ 4.4 วงจรตัดต่อคู่สายโทรศัพท์

4.6 การทดลองส่วนของวงจรตั้งเวลาระงับการโทรออกชั่วคราว

4.6.1 ลำดับขั้นการทดลอง

1) ประกอบวงจรตั้งเวลาระงับการโทรออกชั่วคราว ตามรูปที่ 4.5

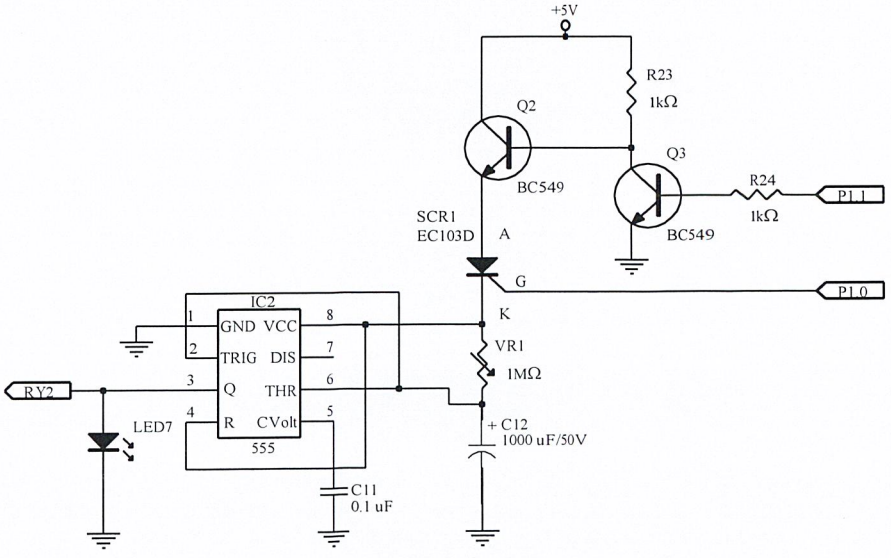
2) ตรวจสอบการต่อวงจรเพื่อความถูกต้อง

3) จ่ายไฟให้กับวงจรตั้งเวลา

4) จ่ายไฟกระตุ้นที่ขาเกตของ SCR สังเกตที่เอาต์พุต

5) ทดลองจับเวลา ซึ่งแสดงผลโดยแอลอีดี

6) ปรับค่าความต้านทาน VR1 ให้ได้เวลาตามต้องการ



รูปที่ 4.5 วงจรตั้งเวลาระงับการโทรออกชั่วคราว

4.6.2 ผลการทดลอง

เมื่อทำการทดลองวงจรตั้งเวลาระงับการโทรออกชั่วคราวตามหลักการขั้นตอนดังกล่าว วงจรตั้งเวลา ก็สามารถตั้งเวลาและทำงานตามที่กำหนดได้

4.7 การทดลองส่วนของวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

4.7.1 ลำดับขั้นการทดลอง

1) ประกอบวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ ตามรูปที่ 4.6

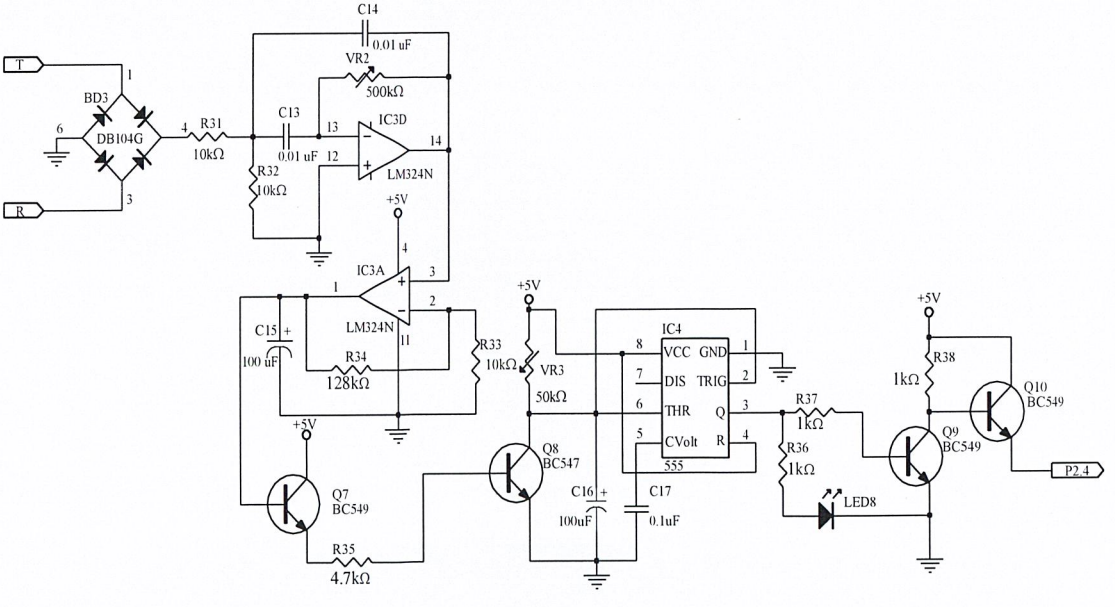
2) ตรวจสอบการต่อวงจรเพื่อความถูกต้อง

3) ต่ออินพุตเข้ากับคู่สายองค์การโทรศัพท์

4) ทำการหมุนหมายเลขไปยังเลขหมายที่สามารถที่จะติดต่อได้เพื่อที่รอรับสัญญาณกระดิ่งตอบกลับ

5) สังเกต LED8 ขณะมีสัญญาณกระดิ่งตอบกลับ วัดไฟที่ขา E ของทรานซิสเตอร์ Q10

6) เมื่อผู้ถูกเรียกปลายทางรับสาย สัญญาณกระดิ่งตอบกลับจะหมดไป สังเกต LED8 และทำการวัดไฟที่ขา E ของทรานซิสเตอร์ Q10



รูปที่ 4.6 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

4.7.2 ผลการทดลอง

เมื่อทดลองตามขั้นตอนที่กล่าวมาแล้วได้ผล คือ วงจรสามารถตรวจจับสัญญาณกระดิ่งตอบกลับได้ โดยในขณะที่มีสัญญาณกระดิ่งตอบกลับ วัดไฟ 0 โวลต์ แต่ถ้าหากมีการรับสายเกิดขึ้น สัญญาณกระดิ่งตอบกลับจะหายไป และสามารถวัดไฟที่พอร์ต P2.4 ได้ประมาณ 4.55 โวลต์

4.8 การทดลองส่วนของวงจรไมโครคอนโทรลเลอร์

4.8.1 ลำดับขั้นการทดลอง

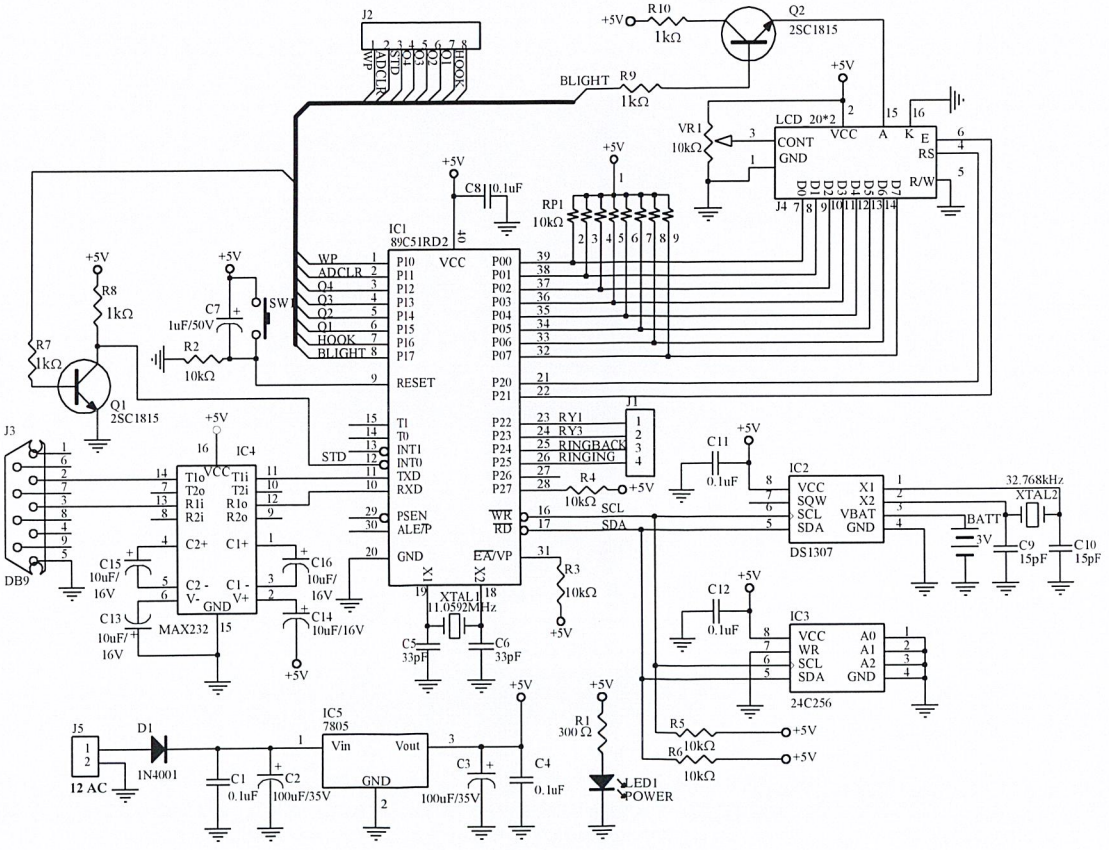
1) ประกอบวงจรไมโครคอนโทรลเลอร์ ตามรูปที่ 4.7

2) ตรวจสอบความถูกต้องของวงจรและจ่ายแรงดันไฟให้กับวงจร

3) ทดสอบโปรแกรมที่ใช้แสดงผลหน้าจอ LCD และไอซีฐานเวลา

4) ทดสอบโปรแกรมในการติดต่อพอร์ตอนุกรม RS-232

5) สังเกตผลที่ได้จากการทดสอบ



รูปที่ 4.7 วงจรไมโครคอนโทรลเลอร์

4.8.2 ผลการทดลอง

เมื่อทำการทดลองได้ผล คือ ที่หน้าจอจะปรากฏ วัน วันที่ เดือน ปี และเวลาปัจจุบันตามโปรแกรมที่เขียนไว้ และสามารถติดต่อสื่อสารผ่านทางพอร์ตอนุกรม RS-232 กับคอมพิวเตอร์ได้

4.9 การทดลองการทำงานของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัตินี้มีการทำงานเป็นส่วนๆ โดยขึ้นอยู่กับการใช้งาน ซึ่งผลการทดลองการทำงานของเครื่องสามารถแบ่งเป็นกรณีต่างๆ ตามจุดประสงค์การใช้งานดังนี้

4.9.1 กรณีการโทรออก

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL
25/01/04 21:35:23

2. เมื่อมีการยกหูโทรศัพท์ จอแสดงผล LCD จะแสดงดังนี้

PASSWORD :

3.เมื่อมีการใส่รหัสผ่านของผู้ใช้จำนวน 4 หลัก จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : ****

4. เมื่อใส่รหัสถูกต้อง จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : ****
TEL. No. :

5. หลังจากนั้นจึงกดเลขหมายโทรศัพท์ที่ต้องการโทรออก จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : ****
TEL. No. : 067151826

6. หลังจากกดเลขหมายโทรศัพท์เสร็จและมีคนรับสาย จอแสดงผล LCD จะแสดงดังนี้

LINE IN USE
25/01/04 21:36:01

7.เมื่อสิ้นสุดการสนทนาให้วางหูโทรศัพท์ จอแสดงผล LCD จะแสดงดังนี้

KMITL 25/01/04 21:40:21

8. ถ้าหากมีการกดรหัสผ่านผิด 3 ครั้ง เครื่องจะเข้าสู่โหมดการป้องกันการโทรออกชั่วคราว เป็นเวลา 20 นาที หรือจนกว่าจะได้รับการแก้ไขจากผู้ดูแลระบบ เครื่องก็จะเข้าสู่การทำงานปกติอีกครั้ง

SAFETY MODE

4.9.2 กรณีการโทรเข้า

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL 25/01/04 21:35:23

2.เมื่อมีสัญญาณกระดิ่งเรียกเข้า จอแสดงผล LCD จะแสดงดังนี้

LINE IN WAIT FOR RECEIVING

3.เมื่อมีการรับสาย จอแสดงผล LCD จะแสดงดังนี้

LINE IN RECEIVING

4.เมื่อสิ้นสุดการสนทนา จอแสดงผล LCD จะแสดงดังนี้

KMITL 25/01/04 21:36:02

4.9.3 กรณีการโทรศัพท์ออกหมายเลขพิเศษ

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL 25/01/04 21:35:23

2.เมื่อมีการยกหู ให้กดเครื่องหมาย #

PASSWORD : #

3.กดหมายเลขพิเศษตามที่ผู้ดูแลระบบเซ็ตไว้

PASSWORD : # TEL. No. : 191

4.เมื่อกดหมายเลขเสร็จแล้ว เครื่องทำการต่อเข้ากับคู่สายองค์การโทรศัพท์

LINE IN USE 25/01/04 21:36:05

5.เมื่อสิ้นสุดการสนทนา

KMITL 25/01/04 21:36:45

4.9.4 กรณีของผู้ดูแลระบบ

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL 25/01/04 21:35:23

2.เมื่อมีการยกหู

PASSWORD :

3.ผู้ดูแลระบบป้อนรหัสส่วนตัว 4 หลัก โดยรหัสจะกดเครื่องหมาย * ตามด้วยตัวเลขสองหลักและหลักสุดท้ายกดเครื่องหมาย # (รหัสผู้ดูแลระบบที่ตั้งไว้ คือ *11#)

PASSWORD : ****

4. เมื่อกดรหัสถูกต้องแล้ว จอแสดงผล LCD จะแสดงเมนูขึ้นมา เพื่อที่ผู้ดูแลระบบจะทำการเซ็ตระบบต่างๆ

1.SAF CLR 2.SET PAS 3.SET TIM 4.ADM PAS

5. เมื่อกด 1 จะทำการเคลียร์ระบบ SAFETY MODE เพื่อที่จะสามารถใช้งานได้ตามปกติ

6. เมื่อกด 2 เพื่อที่จะเปลี่ยนรหัสผ่านของผู้ใช้ จอแสดงผล LCD จะแสดงดังนี้

SET PAS No. SELECT 01-20 :

7. เมื่อต้องการเปลี่ยนรหัสผ่านผู้ใช้คนใด ก็ให้กดเลขหมายโดยกดเลขหมายสองหลัก เช่น ถ้าต้องการเปลี่ยนรหัสผ่านผู้ใช้คนที่ 4 ก็ให้กด หมายเลข 04

SET PAS No. SELECT 01-20 : 04

8. เมื่อกดหมายเลขผู้ใช้เสร็จแล้ว ให้กดรหัสผ่านใหม่ จอแสดงผล LCD จะแสดงดังนี้

NEW PASSWORD : ****

9. เมื่อกดเปลี่ยนรหัสผ่านเสร็จแล้ว จอแสดงผล LCD จะแสดงดังนี้

COMPLETED

10. เมื่อกด 3 จะแสดงถึงการตั้งวัน เดือน ปีและเวลา โดยจะเริ่มตั้งวันที่จอแสดงผล LCD จะแสดงดังนี้

SET DATE & TIME DATE :

11. เมื่อกดหมายเลข เพื่อเปลี่ยนวันที่เสร็จแล้ว จอแสดงผล LCD จะแสดงให้เปลี่ยนเดือนโดยอัตโนมัติ ซึ่งจะเรียงตั้งแต่ วันที่ เดือน ปี ชั่วโมง นาทีและวินาทีตามลำดับ จอแสดงผล LCD จะแสดงดังนี้

SET DATE & TIME MON :

SET DATE & TIME YEAR :

SET DATE & TIME HOUR :

SET DATE & TIME MIN :

SET DATE & TIME SEC :

12. เมื่อกด 4 จะเป็นการเข้าไปเซ็ตรหัสผ่านของผู้ดูแลระบบโดยจะทำการเปลี่ยนรหัสที่เป็นตัวเลขหลักที่สองและสามได้เท่านั้น เช่น จาก *11# ให้เป็น *99# ก็กดเฉพาะตัวเลข 99 เท่านั้น จอแสดงผล LCD จะแสดงดังนี้

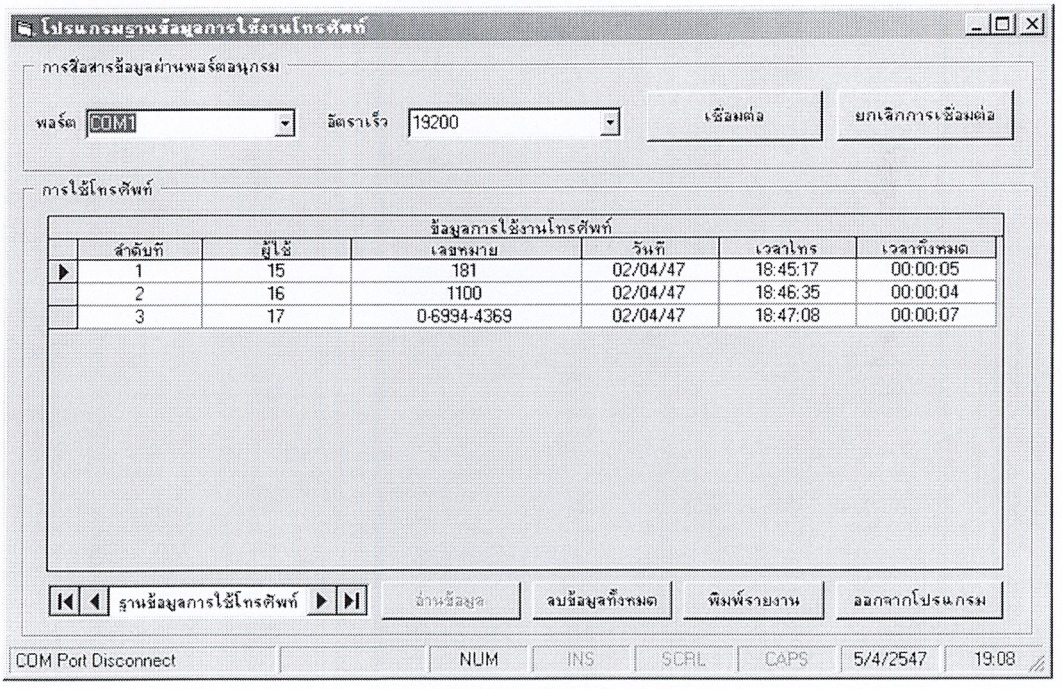
SET ADMIN PASS NEW PASSWORD :

4.9.5 กรณีการเชื่อมต่อข้อมูลออกไปยังคอมพิวเตอร์

1. เชื่อมต่อสาย DB-9 จากตัวเครื่องเข้ากับพอร์ตคอมพิวเตอร์
2. เปิดหน้าจอโปรแกรมฐานข้อมูลการใช้งานโทรศัพท์
3. ทำการเลือกพอร์ต COM อัตราความเร็ว 19200 บิตต่อวินาที
4. กดปุ่มเชื่อมต่อและอ่านข้อมูล จอแสดงผล LCD จะแสดงดังนี้

UPLOADING
PLEASE WAIT

5. ถ้าต้องการพิมพ์รายงานก็ทำการกดที่ปุ่มพิมพ์รายงาน โดยหน้าจอโปรแกรมในขณะใช้งานแสดงไว้ดังรูปที่ 4.8



รูปที่ 4.8 รูปแบบโปรแกรมในขณะใช้งาน

บทที่ 5
บทสรุป

5.1 สรุป

เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ แบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นฮาร์ดแวร์ และส่วนที่เป็นซอฟต์แวร์ ในส่วนของฮาร์ดแวร์จะประกอบไปด้วยส่วนหลัก คือ วงจรตรวจสอบสัญญาณโทรศัพท์ และวงจรไมโครคอนโทรลเลอร์ สำหรับในส่วนของซอฟต์แวร์ จะประกอบด้วยโปรแกรมที่ใช้ในการควบคุมการทำงานของเครื่องและโปรแกรมวิชวล เบสิค ที่ใช้ในการแสดงผลรายงานการบันทึกข้อมูลการใช้โทรศัพท์ผ่านทางคอมพิวเตอร์

ขีดความสามารถของโครงงานนี้ สามารถบันทึกข้อมูลการใช้งานโทรศัพท์ในกรณีที่มีการโทรออกได้จริง โดยมีการตรวจสอบรหัสผ่านจำนวน 4 หลักของผู้ใช้แต่ละคน เมื่อกดรหัสถูกเครื่องจะยอมให้มีการโทรออกได้ตามปกติ เมื่อมีการรับสายจากปลายทาง เครื่องก็จะทำการบันทึกข้อมูลการโทรออกซึ่งประกอบไปด้วย ชื่อผู้ใช้ หมายเลขที่โทรออก วันและเวลาที่โทรออก และจำนวนเวลาที่ใช้โทรออก ในกรณีเมื่อมีสายเรียกเข้าสามารถที่จะรับสายได้ทันทีโดยไม่ต้องกดรหัสผ่านการทำงานแต่ละขั้นตอนของเครื่อง มีการแสดงผลทางจอแอลซีดี 20 ตัวอักษร 2 บรรทัดและสามารถเชื่อมต่อข้อมูลการโทรออกที่บันทึกจากเครื่องไปยังเครื่องคอมพิวเตอร์ได้โดยแสดงผลผ่านทางโปรแกรมฐานข้อมูลการใช้งานโทรศัพท์และทำการพิมพ์ออกมาดูได้ในรูปแบบเอกสาร

5.2 ปัญหาและแนวทางแก้ไข

จากการดำเนินการสร้างและทดสอบโครงงานพบว่ามีปัญหาเกิดขึ้นหลายประการ ซึ่งสรุปได้ดังต่อไปนี้

1. ปัญหา วงจรถอดรหัสสัญญาณความถี่คู่ทำงานผิดพลาด เนื่องจากมีความถี่เข้ามารบกวนจากวงจรทวีแรงดัน ทำให้ไอซีถอดรหัสความถี่ MT8870 ไม่สามารถถอดรหัสความถี่คู่จากการกดปุ่มโทรศัพท์ได้

แนวทางแก้ไข ใช้ตัวเก็บประจุค่า 0.1 ไมโครฟารัด ต่อคร่อมที่เอาต์พุตของวงจรทวีแรงดันเพื่อทำการกรองเอาเฉพาะแรงดันไฟตรง ส่วนความถี่รบกวนจะผ่านตัวเก็บประจุลงกราวด์

2. ปัญหา เมื่อใช้แรงดันไฟตรงจากวงจรทวีแรงดัน จ่ายเลี้ยงให้โทรศัพท์ในกรณีที่โทรศัพท์ถูกตัดออกจากคู่สาย ปรากฏว่าสามารถกดปุ่มโทรศัพท์ได้ แต่มีเสียงฮัมรบกวนเกิดขึ้น

แนวทางแก้ไข ใช้ตัวต้านทานค่า 330 กิโลโอห์ม และตัวเก็บประจุค่า 0.1 ไมโครฟารัด ต่อคร่อมที่เอาต์พุตของวงจรทวีแรงดัน

3. ปัญหา วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ ไม่สามารถแยกแยะสัญญาณ Ringback Tone กับสัญญาณเสียงพูดได้ เนื่องจากมีความถี่ 400 เฮิรตซ์เหมือนกัน

แนวทางแก้ไข ต้องมีการตรวจสอบความต่อเนื่องของสัญญาณความถี่ 400 เฮิรตซ์ โดยใช้ C_{15} ค่า 100 ไมโครฟารัดเป็นตัวตรวจสอบความต่อเนื่องของสัญญาณและตัวต้านทาน R_{34} ที่มีความต้านทานที่เหมาะสมค่า 128 กิโลโอห์ม ที่สามารถรับสัญญาณ Ringback Tone ได้และลดสัญญาณเสียงพูดที่เข้ามารบกวนดีที่สุด

4. ปัญหา วงจรตรวจจับการยกหูต่ออยู่กับคู่สาย ในขณะที่มีสัญญาณกระดิ่งเข้ามา ทำให้สัญญาณกระดิ่งเข้าไปรบกวนวงจรตรวจจับการยกหูทำให้ไมโครคอนโทรลเลอร์ MCS-51 ตรวจจับว่าเป็นการยกหู ทำให้เกิดการทำงานผิดพลาด

แนวทางแก้ไข สร้างวงจรโมโนสเตเบิลต่อที่เอาต์พุตของวงจรตรวจจับสัญญาณกระดิ่ง และเมื่อลอจิกเปลี่ยนจาก 0 เป็น 1 แสดงว่ามีการยกหูรับสายหรือเลิกการเรียกสาย แล้วจึงตรวจจับการยกหู

5. ปัญหา ไมโครคอนโทรลเลอร์ MCS-51 ไม่สามารถส่งลอจิกเอาต์พุตไปควบคุมให้ทรานซิสเตอร์ทำงานได้

แนวทางแก้ไข ใช้ไอซีบัฟเฟอร์เบอร์ 74HC244 ต่อออกจากพอร์ตเอาต์พุตของไมโครคอนโทรลเลอร์ MCS-51 ก่อนที่จะต่อเข้ากับวงจรโทรศัพท์

5.3 แนวทางการพัฒนา

1. ออกแบบและพัฒนาวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ เพื่อใช้การตรวจสอบการรับสายปลายทางให้เที่ยงตรงสมบูรณ์ยิ่งขึ้น

2. ควรพัฒนาให้มีการตอบรับสายเรียกเข้าได้โดยอัตโนมัติ และสามารถบันทึกข้อความเสียงได้ในกรณีไม่มีผู้รับสาย

3. เพิ่มจำนวนรหัสผ่านผู้ใช้ให้มีจำนวนมากขึ้น

4. ควรพัฒนาให้มีการสั่งงานควบคุมอุปกรณ์ไฟฟ้าผ่านทางโทรศัพท์ได้

5. ควรพัฒนาให้มีการแสดงเลขหมายโทรเข้า

บรรณานุกรม

ชัยวัฒน์ ลิ้มพรจิตรวิไล, วรพจน์ กรแก้ววัฒนกุล. **เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์** กรุงเทพฯ : บริษัท อินโนเวตีฟ เอ็กเพอร์เมนต์ จำกัด. 2542

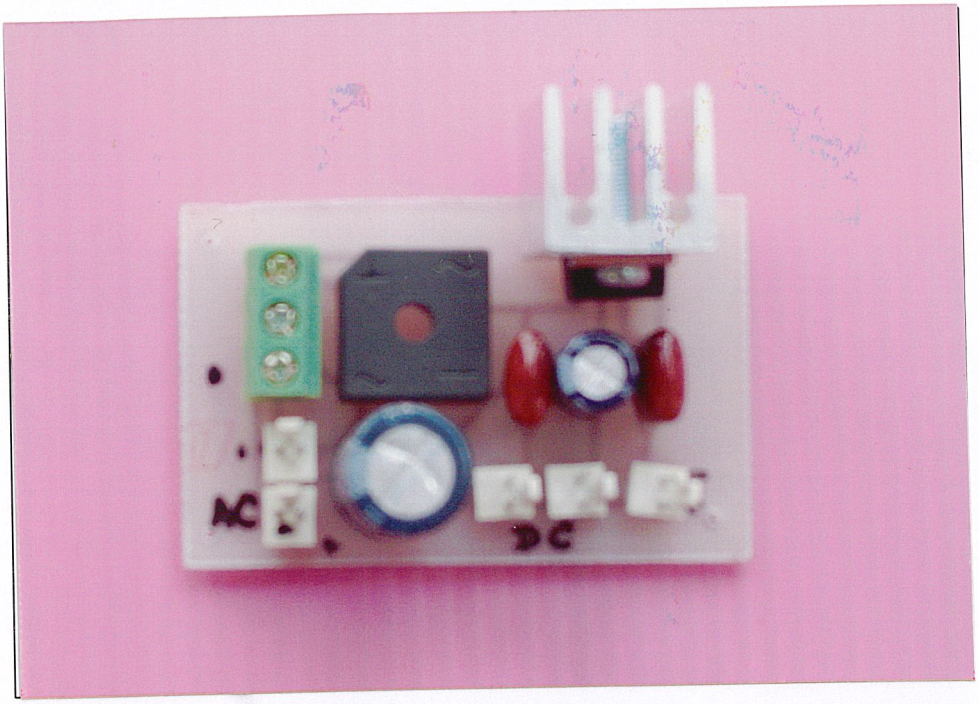
บัณฑิต จามรภูติ. **คู่มือการใช้งาน Protel 99**. เชียงใหม่ : บัณฑิตเพลส. 2544

สมยศ จุณณะปิยะ. **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์**. พิมพ์ครั้งที่ 3. กรุงเทพฯ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2543

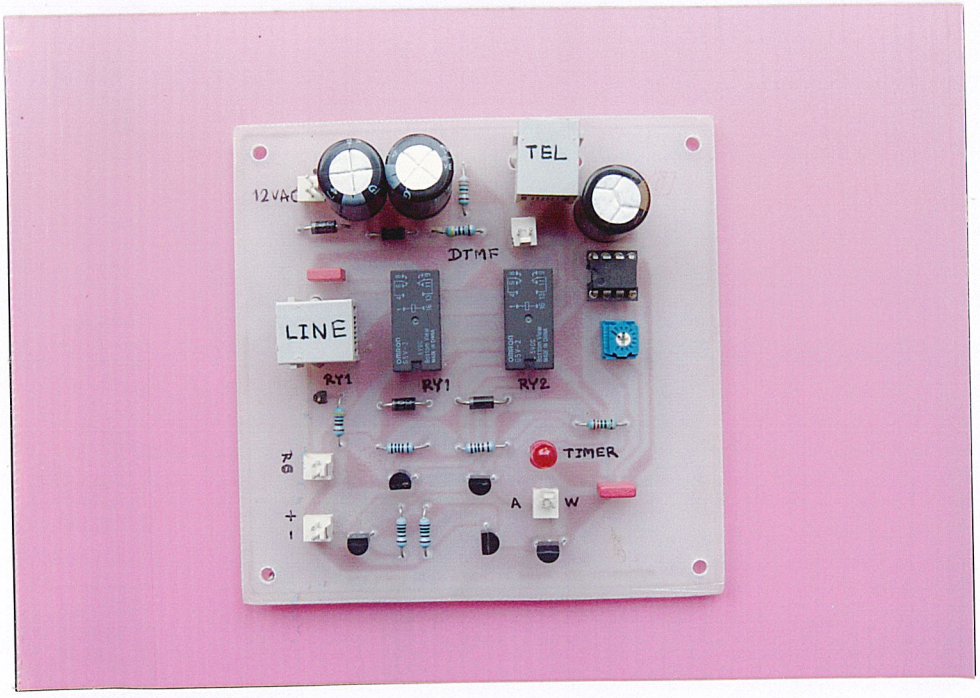
อุดม จีนประดับ. **ไมโครคอนโทรลเลอร์ MCS-51**. พิมพ์ครั้งที่ 1. กรุงเทพฯ : สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ. 2541

อภิชาติ ภู่พลับ.**เริ่มต้นเขียนโปรแกรมติดต่อ และควบคุมฮาร์ดแวร์ด้วย Visual Basic**.นนทบุรี: อินโฟเพลส,2546.

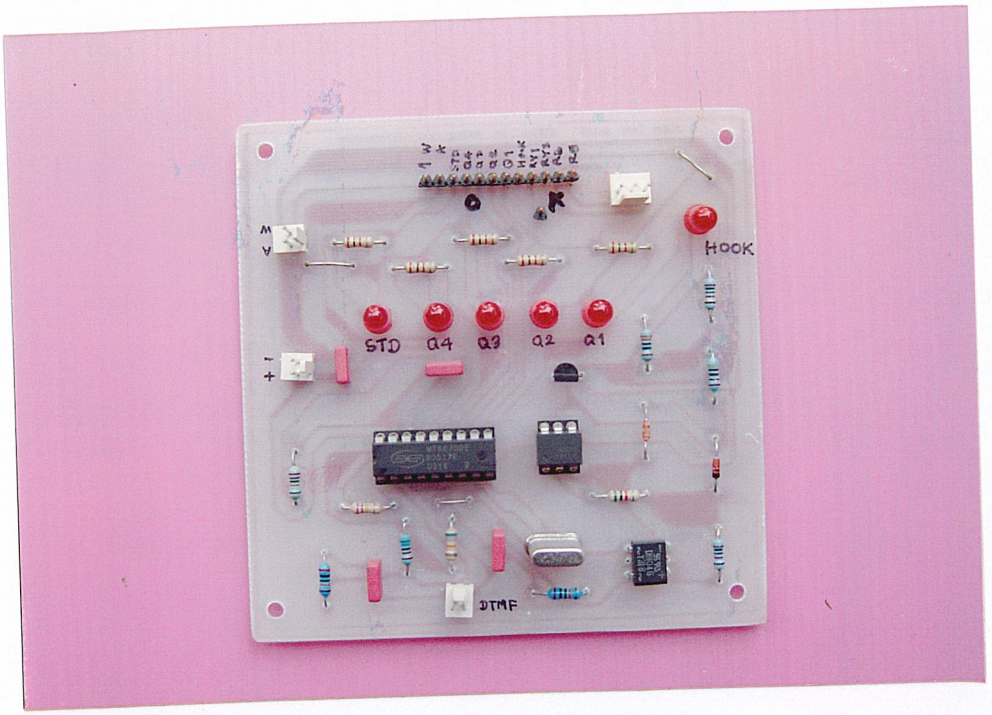
ภาคผนวก ก
เครื่องต้นแบบ



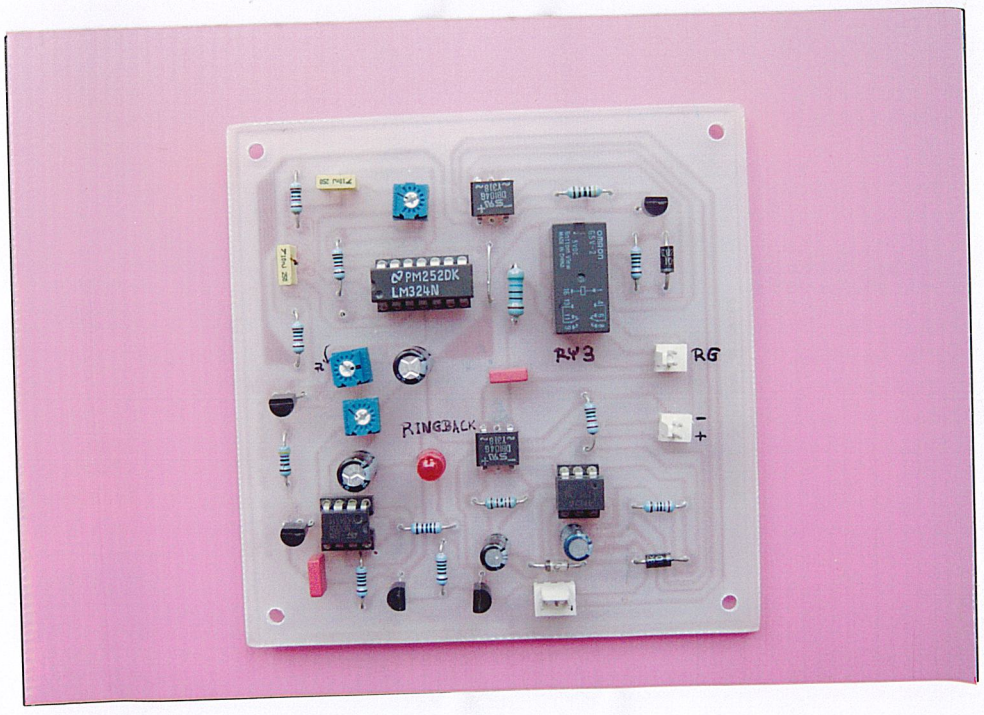
รูปที่ ก.1 วงจรแหล่งจ่ายไฟ



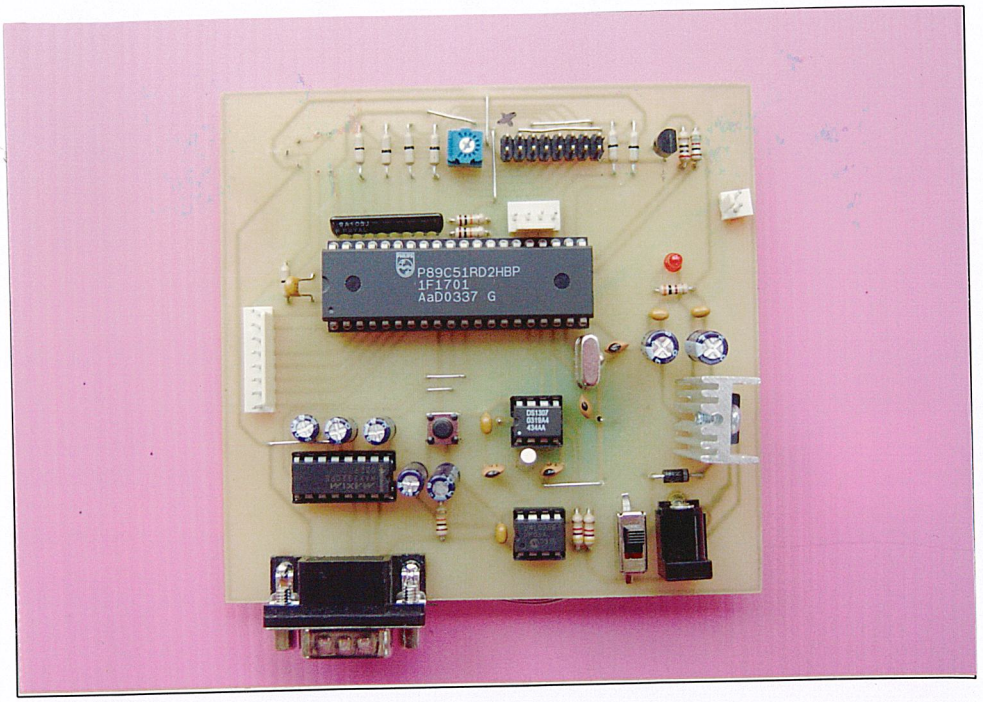
รูปที่ ก.2 วงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว



รูปที่ ก.3 วงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์



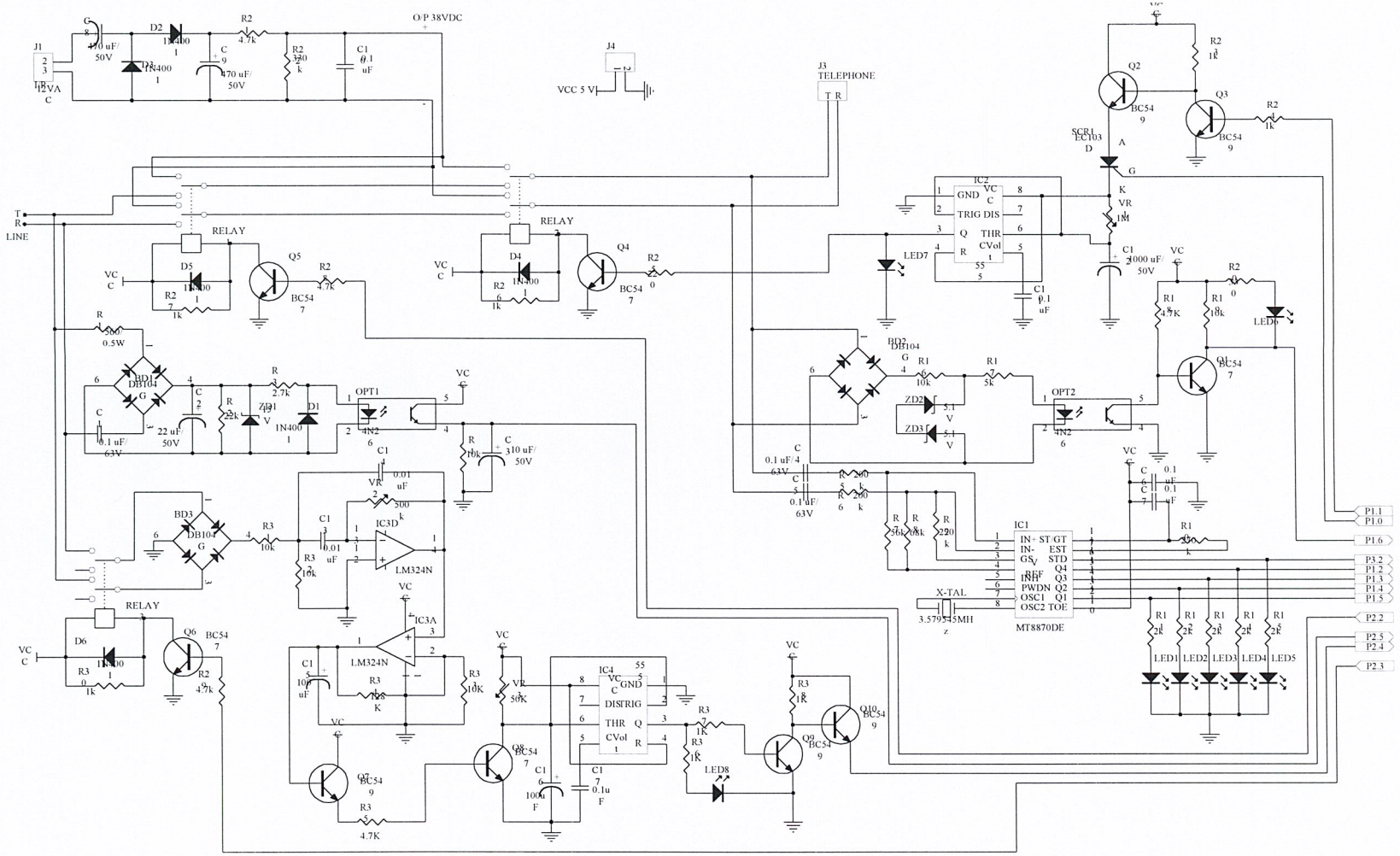
รูปที่ ก.4 วงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ



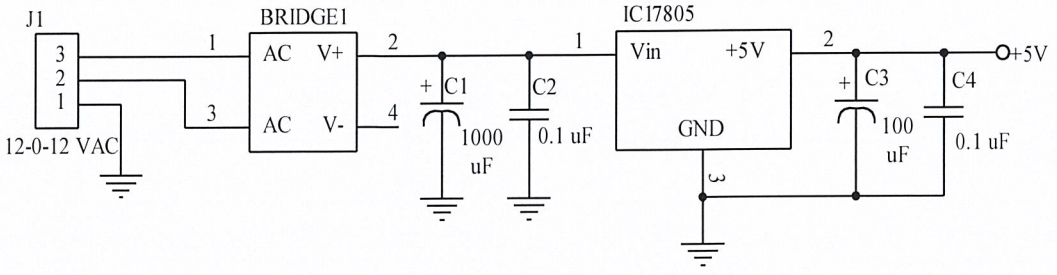
รูปที่ ก.5 วงจรไมโครคอนโทรลเลอร์

รูปที่ ก.6 เครื่องต้นแบบของเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

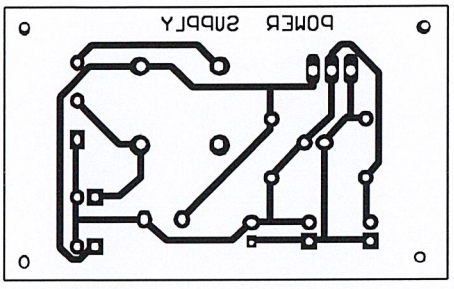
ภาคผนวก ข
วงจรและแผ่นวงจรพิมพ์



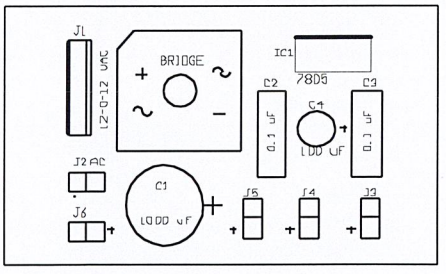
รูปที่ ข.1 วงจรรวมของภาคโทรศัพท์



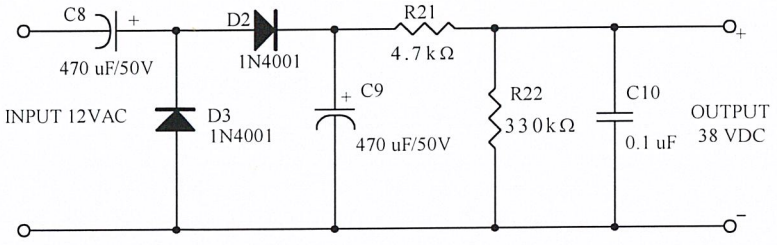
รูปที่ ข.2 วงจรแหล่งจ่ายไฟ



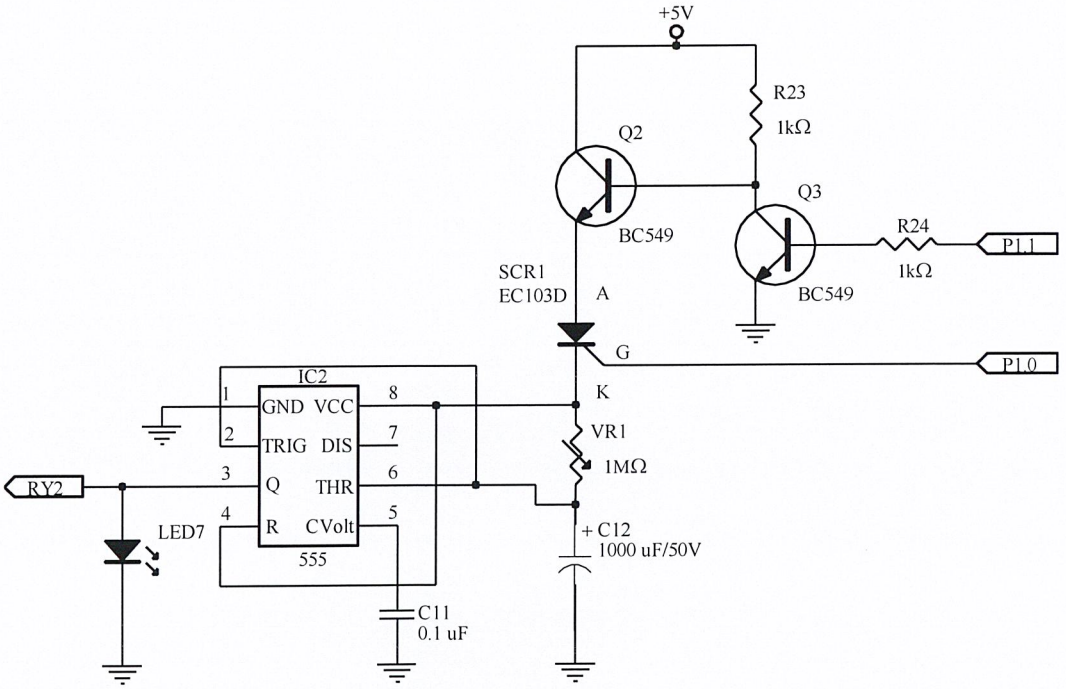
รูปที่ ข.3 ลายวงจรแหล่งจ่ายไฟ



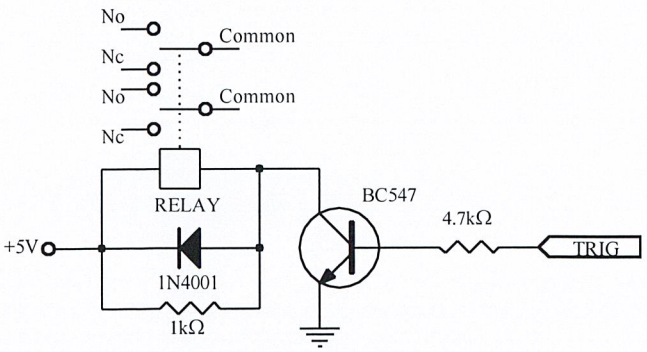
รูปที่ ข.4 การวางอุปกรณ์วงจรแหล่งจ่ายไฟ



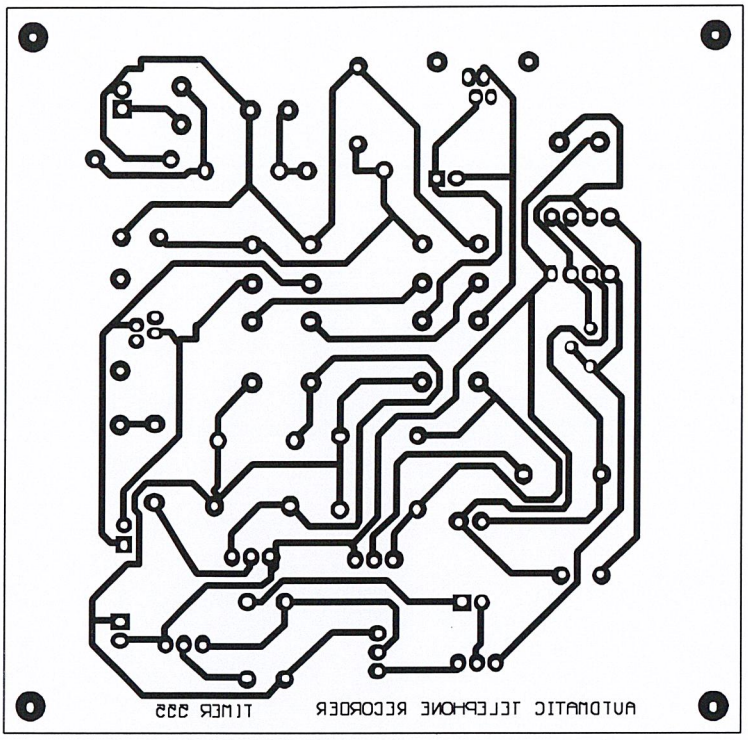
รูปที่ ข.5 วงจรทวีแรงดัน



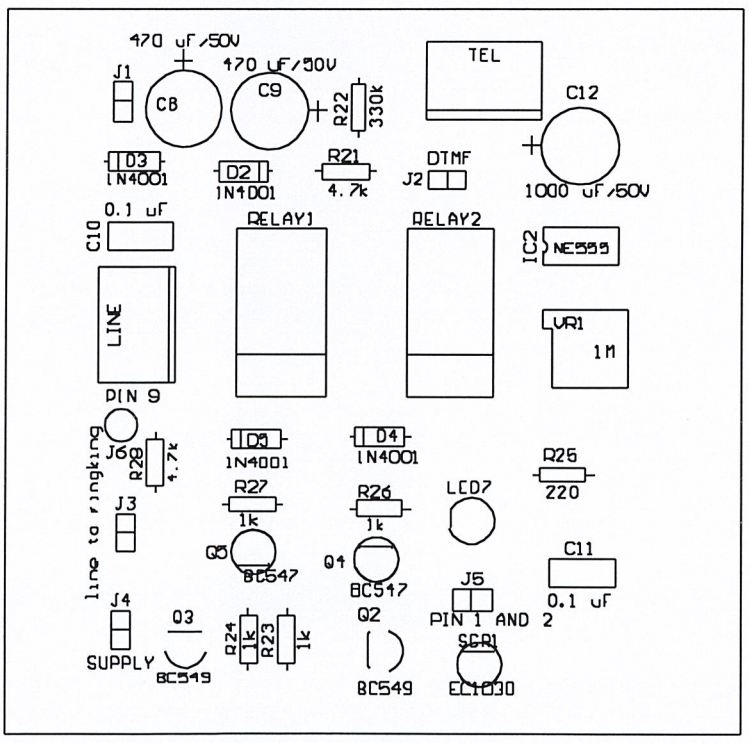
รูปที่ ข.6 วงจรตั้งเวลาระงับการโทรออกชั่วคราว



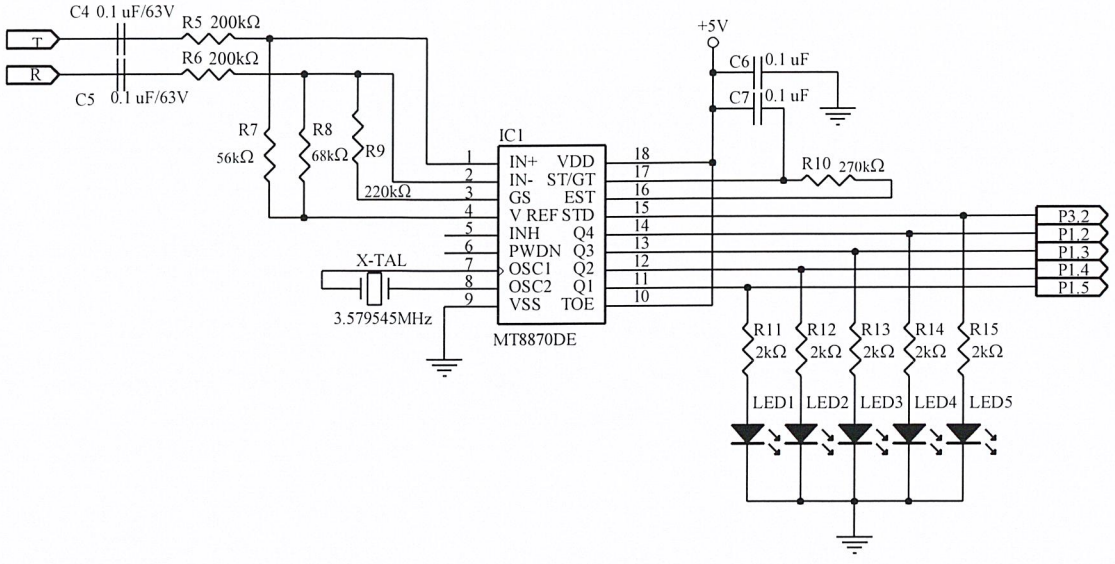
รูปที่ ข.7 วงจรตัดต่อคู่สายโทรศัพท์



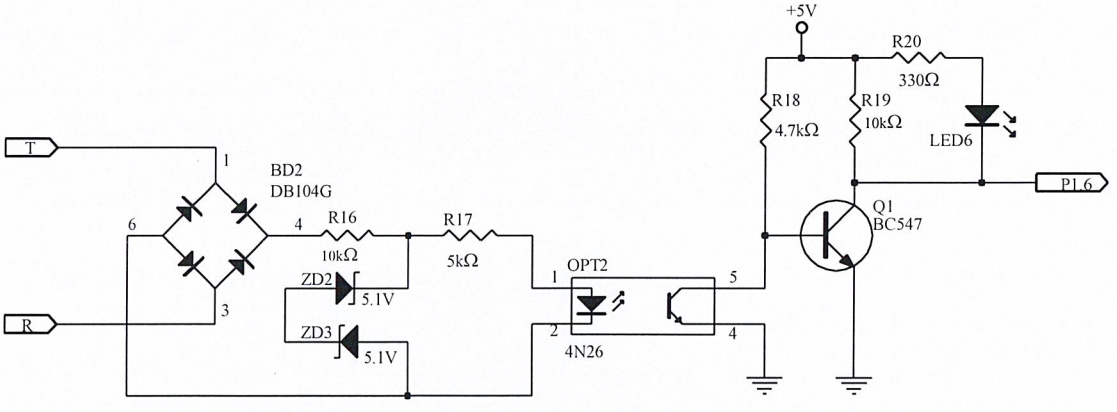
รูปที่ ข.8 ลายวงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว



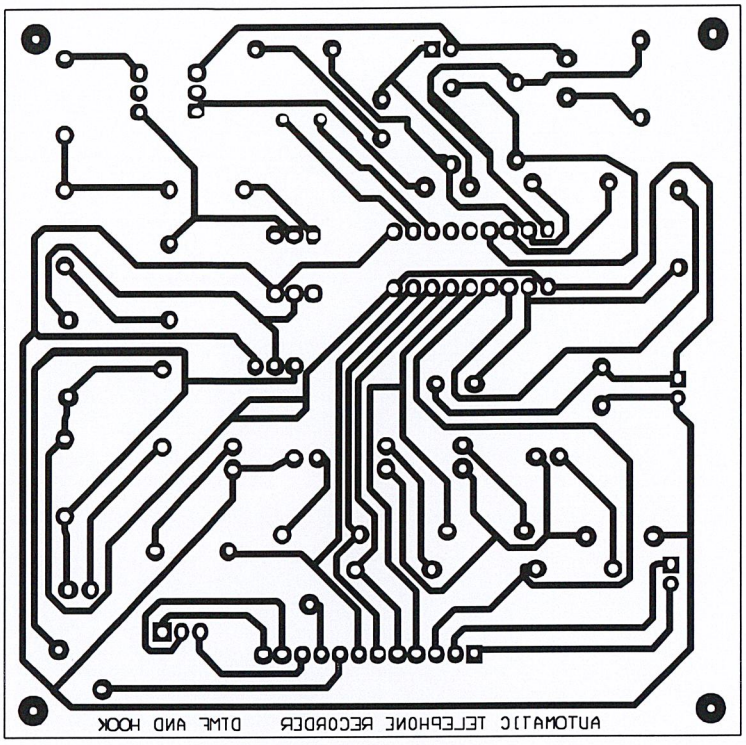
รูปที่ ข.9 การวางอุปกรณ์วงจรทวีแรงดันและวงจรตั้งเวลาระงับการโทรออกชั่วคราว



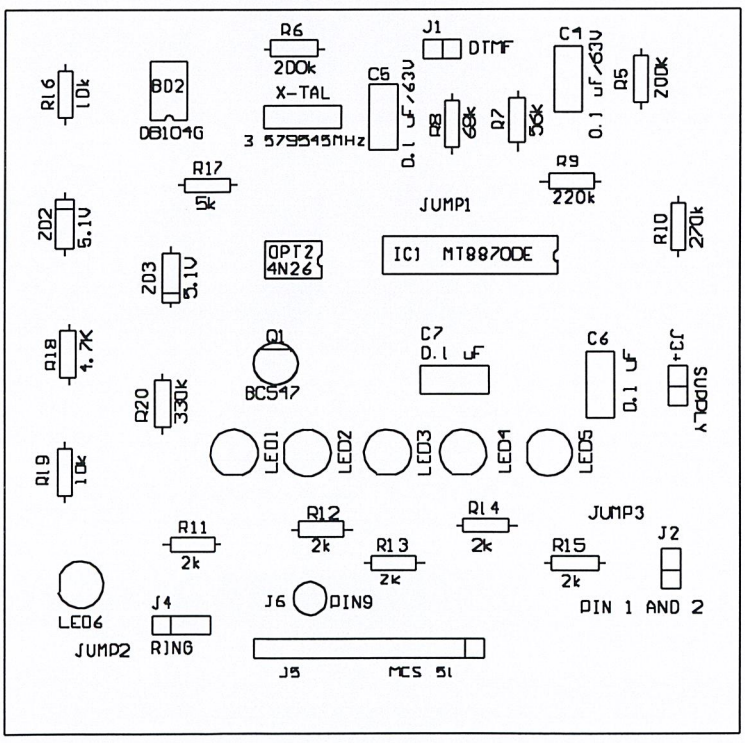
รูปที่ ข.10 วงจรถอดรหัสสัญญาณความถี่คู่



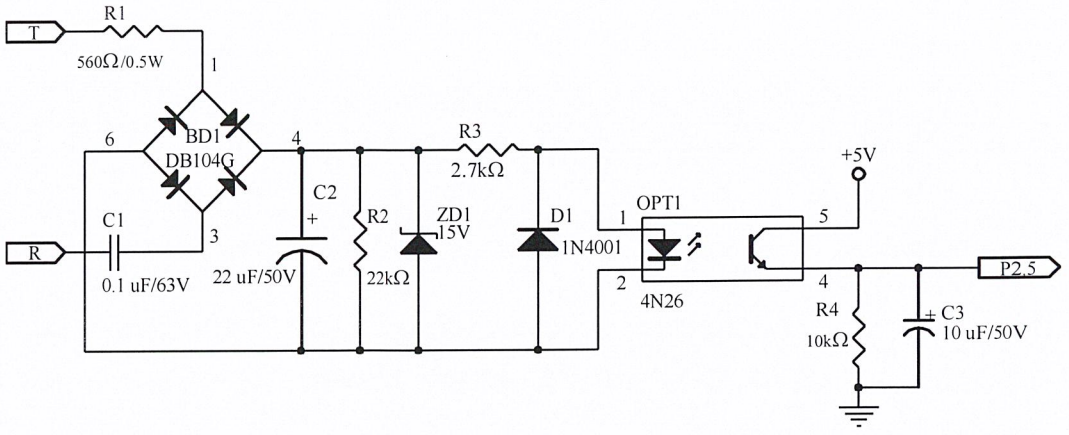
รูปที่ ข.11 วงจรตรวจจับการยกหูโทรศัพท์



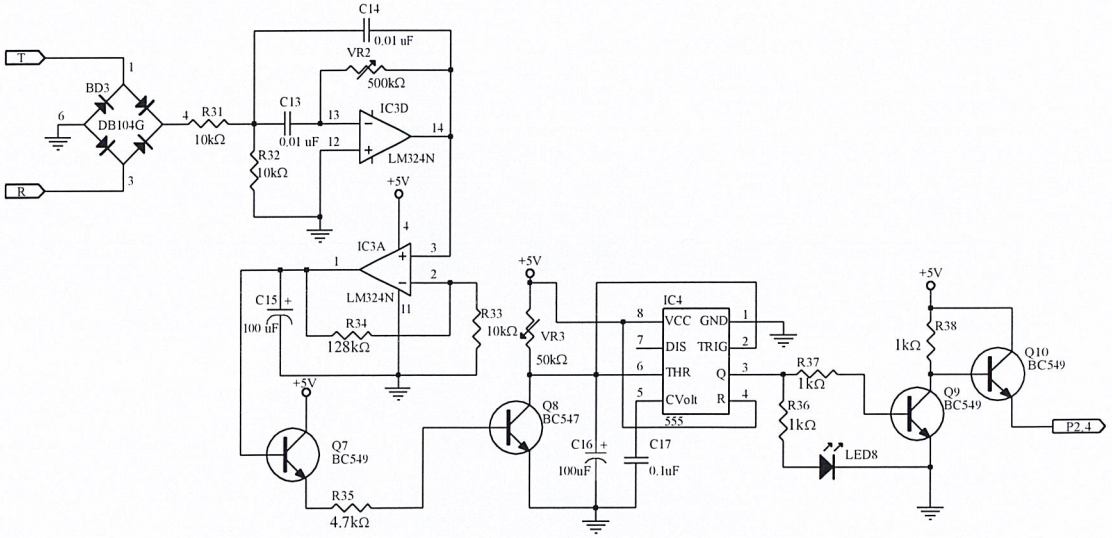
รูปที่ ข.12 ลายวงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์



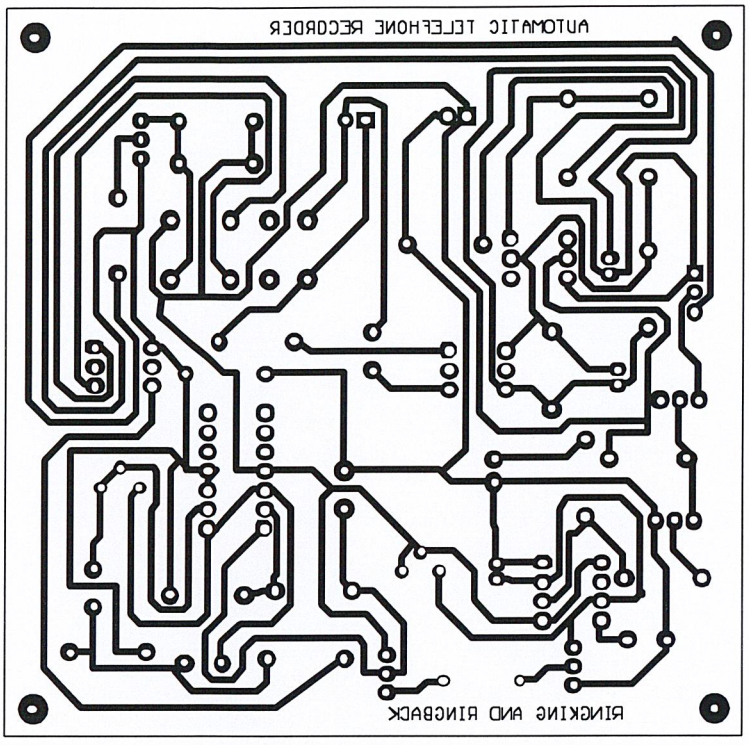
รูปที่ ข.13 การวางอุปกรณ์วงจรถอดรหัสสัญญาณความถี่คู่และวงจรตรวจจับการยกหูโทรศัพท์



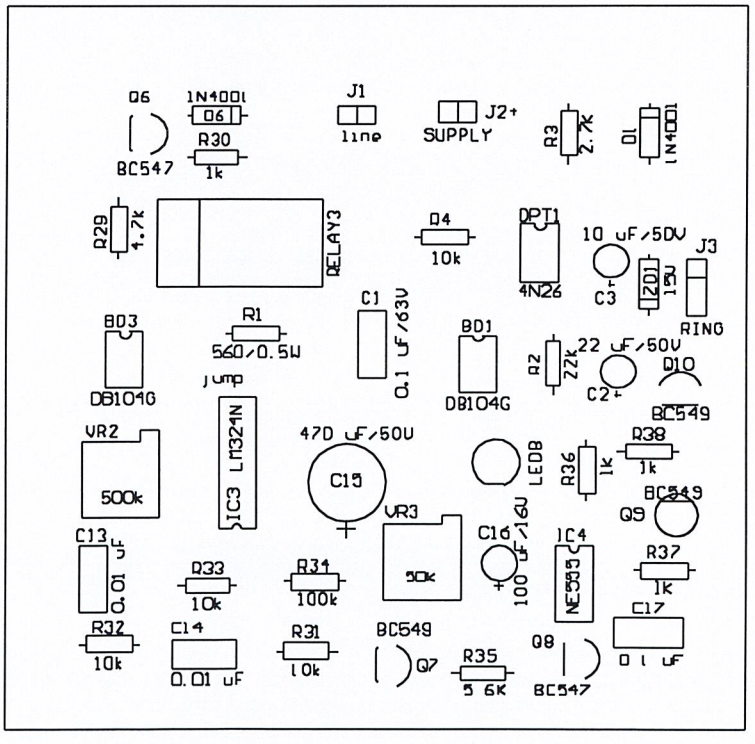
รูปที่ ข.14 วงจรตรวจจับสัญญาณกระดิ่ง



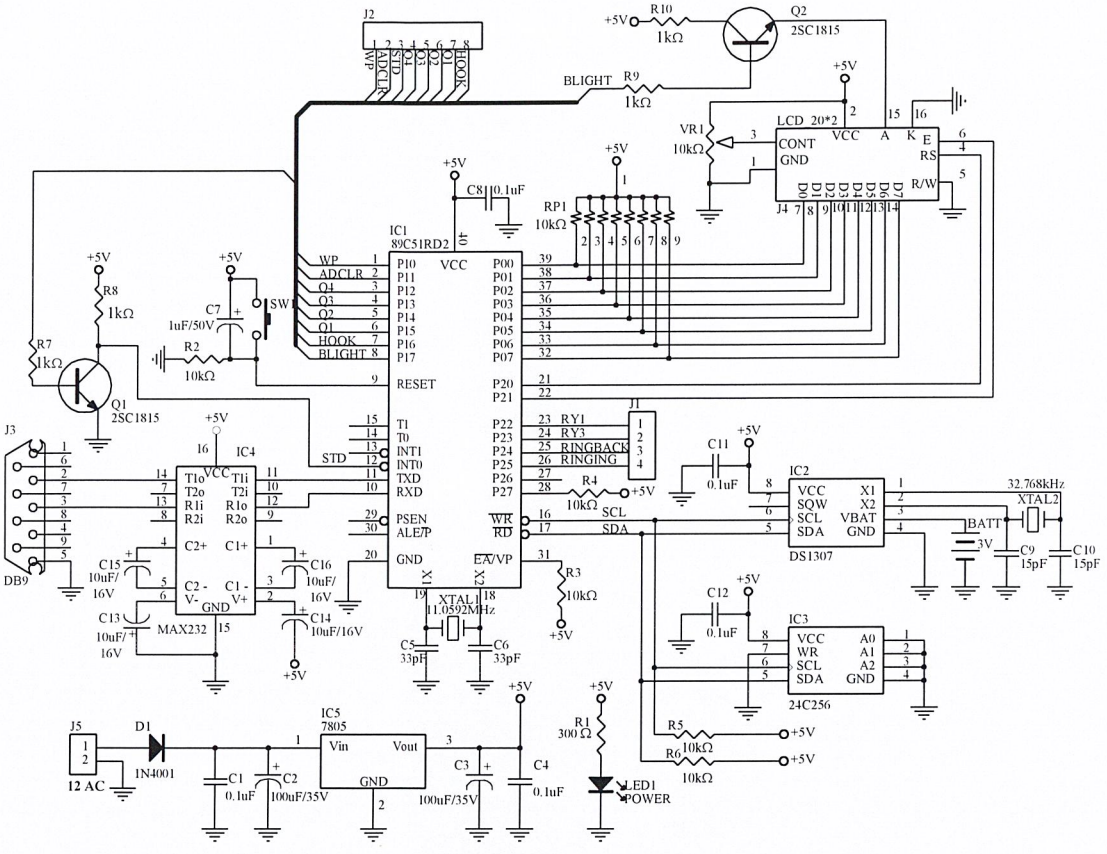
รูปที่ ข.15 วงจรตรวจจับสัญญาณกระดิ่งตอบกลับ



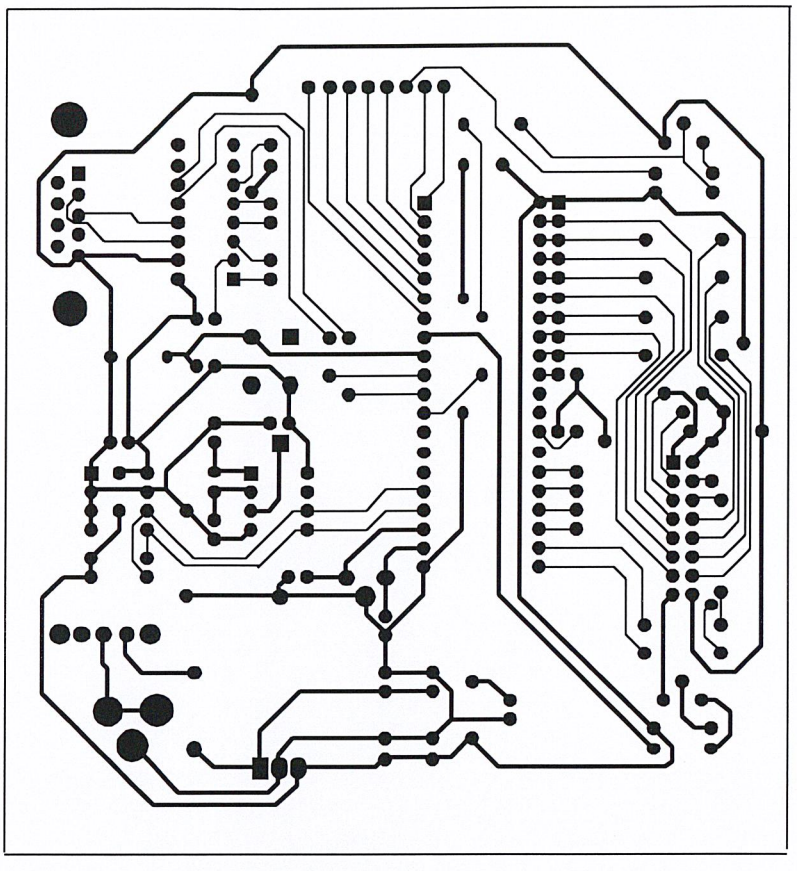
รูปที่ ข.16 ลายวงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ



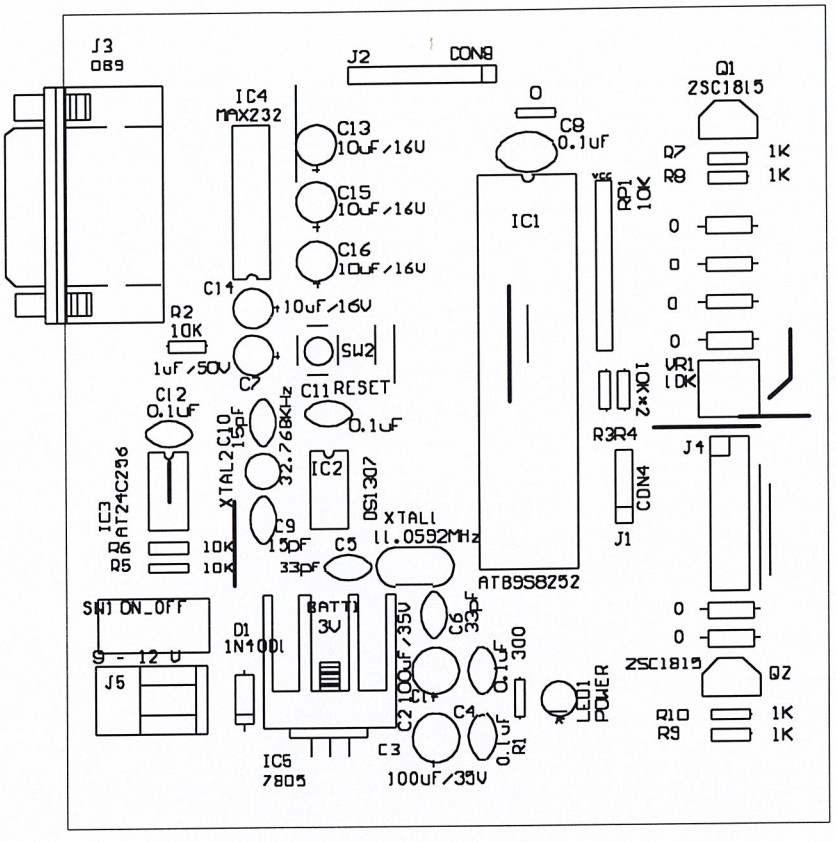
รูปที่ ข.17 วางอุปกรณ์วงจรตรวจจับสัญญาณกระดิ่งและวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ



รูปที่ ข.18 วงจรไมโครคอนโทรลเลอร์



รูปที่ ข.19 ลายวงจรไมโครคอนโทรลเลอร์



รูปที่ ข.20 การวางอุปกรณ์วงจรไมโครคอนโทรลเลอร์

ภาคผนวก ค
รายการอุปกรณ์

ตารางที่ ค.1 รายการอุปกรณ์ของวงจรตรวจจับสัญญาณกระดิ่ง

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
OPT1	4N26	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
ZD1	15 V	1 ตัว
D1	1N4001	1 ตัว
BD1	DB104G	1 ตัว
ตัวเก็บประจุ		
C1	0.1 μF 63 V	1 ตัว
C2	22 μF 50 V	1 ตัว
C3	10 μF 50 V	1 ตัว
ตัวความต้านทาน		
R1	560 Ω 1/4 W	1 ตัว
R2	22 kΩ 1/4 W	1 ตัว
R3	2.7 kΩ 1/4 W	1 ตัว
R4	10 kΩ 1/4 W	1 ตัว
อุปกรณ์อื่นๆ		
J1, J2	Socket 2 pin	2 ตัว
J3	Socket 3 pin	1 ตัว

ตารางที่ ค.2 รายการอุปกรณ์ของวงจรถอดรหัสสัญญาณความถี่คู่

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
IC1	MT8870DE	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
LED1-LED5	สีแดง	5 ตัว

ตารางที่ ค.2 (ต่อ) รายการอุปกรณ์ของวงจรถอดรหัสสัญญาณความถี่คู่

ชื่ออุปกรณ์	รายละเอียด	จำนวน
ตัวเก็บประจุ		
C4-C7	0.1 μF 63 V	4 ตัว
ตัวความต้านทาน		
R5, R6	200 kΩ 1/4 W	2 ตัว
R7	56 kΩ 1/4 W	1 ตัว
R8	68 kΩ 1/4 W	1 ตัว
R9	220 kΩ 1/4 W	1 ตัว
R10	270 kΩ 1/4 W	1 ตัว
R11-R15	2 kΩ 1/4 W	5 ตัว
อุปกรณ์อื่นๆ		
X-TAL	3.579545 MHz	1 ตัว
J1-J3	Socket 2 pin	3 ตัว
J4	Socket 3 pin	1 ตัว

ตารางที่ ค.3 รายการอุปกรณ์ของวงจรตรวจจับการยกหู

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
OPT2	4N26	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
ZD2, ZD3	5.1 V	2 ตัว
Q1	BC 547	1 ตัว
LED6	สีแดง	1 ตัว
BD2	DB104G	1 ตัว

ตารางที่ ค.3 (ต่อ) รายการอุปกรณ์ของวงจรตรวจจับการยกหู

ชื่ออุปกรณ์	รายละเอียด	จำนวน
ตัวความต้านทาน		
R16, R19	10 kΩ 1/4 W	2 ตัว
R17	5 kΩ 1/4 W	1 ตัว
R18	4.7 kΩ 1/4 W	1 ตัว
R20	330 Ω 1/4 W	1 ตัว

ตารางที่ ค.4 รายการอุปกรณ์ของวงจรทวีแรงดัน

ชื่ออุปกรณ์	รายละเอียด	จำนวน
อุปกรณ์สารกึ่งตัวนำ		
D2, D3	1N4001	2 ตัว
ตัวเก็บประจุ		
C8, C9	470 µF 50 V	2 ตัว
C10	0.1 µF	1 ตัว
ตัวความต้านทาน		
R21	4.7 kΩ 1/4 W	1 ตัว
R22	330 kΩ 1/4 W	1 ตัว

ตารางที่ ค.5 รายการอุปกรณ์ของวงจรตั้งเวลาระงับการโทรออกชั่วคราว

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
IC2	NE555N	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
Q2, Q3	BC 549	2 ตัว
SCR1	EC103D	1 ตัว
LED7	สีแดง	1 ตัว

ตารางที่ ค.5 (ต่อ) รายการอุปกรณ์ของวงจรตั้งเวลาระงับการโทรออกชั่วคราว

ชื่ออุปกรณ์	รายละเอียด	จำนวน
ตัวเก็บประจุ		
C11	0.1μF	1 ตัว
C12	1000 μF 50 V	1 ตัว
ตัวความต้านทาน		
R23, R24	1 kΩ 1/4 W	2 ตัว
VR1	1 MΩ 1/4 W	1 ตัว
อุปกรณ์อื่นๆ		
J1-J5	Socket 2 pin	5 ตัว

ตารางที่ ค.6 รายการอุปกรณ์ของวงจรตัดต่อคู่สายโทรศัพท์

ชื่ออุปกรณ์	รายละเอียด	จำนวน
อุปกรณ์สารกึ่งตัวนำ		
D4-D6	1N4001	3 ตัว
Q4-Q6	BC 547	3 ตัว
ตัวความต้านทาน		
R25	220 Ω 1/4 W	1 ตัว
R26, R27	1 kΩ 1/4 W	2 ตัว
R28, R29	4.7 kΩ 1/4 W	2 ตัว
R30	1 kΩ1/4 W	1 ตัว
อุปกรณ์อื่นๆ		
RELAY1-3	แบบ DPDT 5 V	3 ตัว

ตารางที่ ค.7 รายการอุปกรณ์ของวงจรตรวจจับสัญญาณกระดิ่งตอบกลับ

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
IC3	LM324N	1 ตัว
IC4	NE555	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
Q7	BC 549	1 ตัว
Q8	BC 547	1 ตัว
Q9, Q10	BC 549	1 ตัว
LED8	สีแดง	1 ตัว
BD3	DB104G	1 ตัว
ตัวเก็บประจุ		
C13, C14	0.01 μF	2 ตัว
C15-C16	100 μF	2 ตัว
C17	0.1 μF	1 ตัว
ตัวความต้านทาน		
R31-R33	10 kΩ 1/4 W	3 ตัว
R34	220 kΩ 1/4 W	1 ตัว
R35	4.7 kΩ 1/4 W	1 ตัว
R36-R38	1 kΩ 1/4 W	3 ตัว
VR2	500 kΩ 1/4 W	1 ตัว
VR3	50 kΩ 1/4 W	1 ตัว

ตารางที่ ค.8 รายการอุปกรณ์ของวงจรไมโครคอนโทรลเลอร์

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
IC1	89C51RD2	1 ตัว
IC2	DS1307	1 ตัว
IC3	24C256	1 ตัว
IC4	MAX232	1 ตัว
IC5	7805	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
Q1, Q2	2SC1815	2 ตัว
D1	1N4001	1 ตัว
LED1	สีแดง	1 ตัว
ตัวเก็บประจุ		
C1, C4, C8, C11, C12	0.1 µF	5 ตัว
C2, C3	100 µF 35 V	2 ตัว
C5, C6	33 pF	2 ตัว
C7	1 µF 50 V	1 ตัว
C9, C10	15 pF	1 ตัว
C13, C16	10 µF 16 V	5 ตัว
ตัวความต้านทาน		
R1	300 Ω 1/4 W	1 ตัว
R2-R6	10 kΩ 1/4 W	5 ตัว
R7-R10	1 kΩ 1/4 W	4 ตัว
VR1	10 kΩ 1/4 W	1 ตัว
RP1	Rpack 10 kΩ 1/4 W	1 ตัว
อุปกรณ์อื่นๆ		
X-TAL1	11.0592 MHz	1 ตัว
X-TAL2	32.768 kHz	1 ตัว
BATT1	3 V	1 ตัว

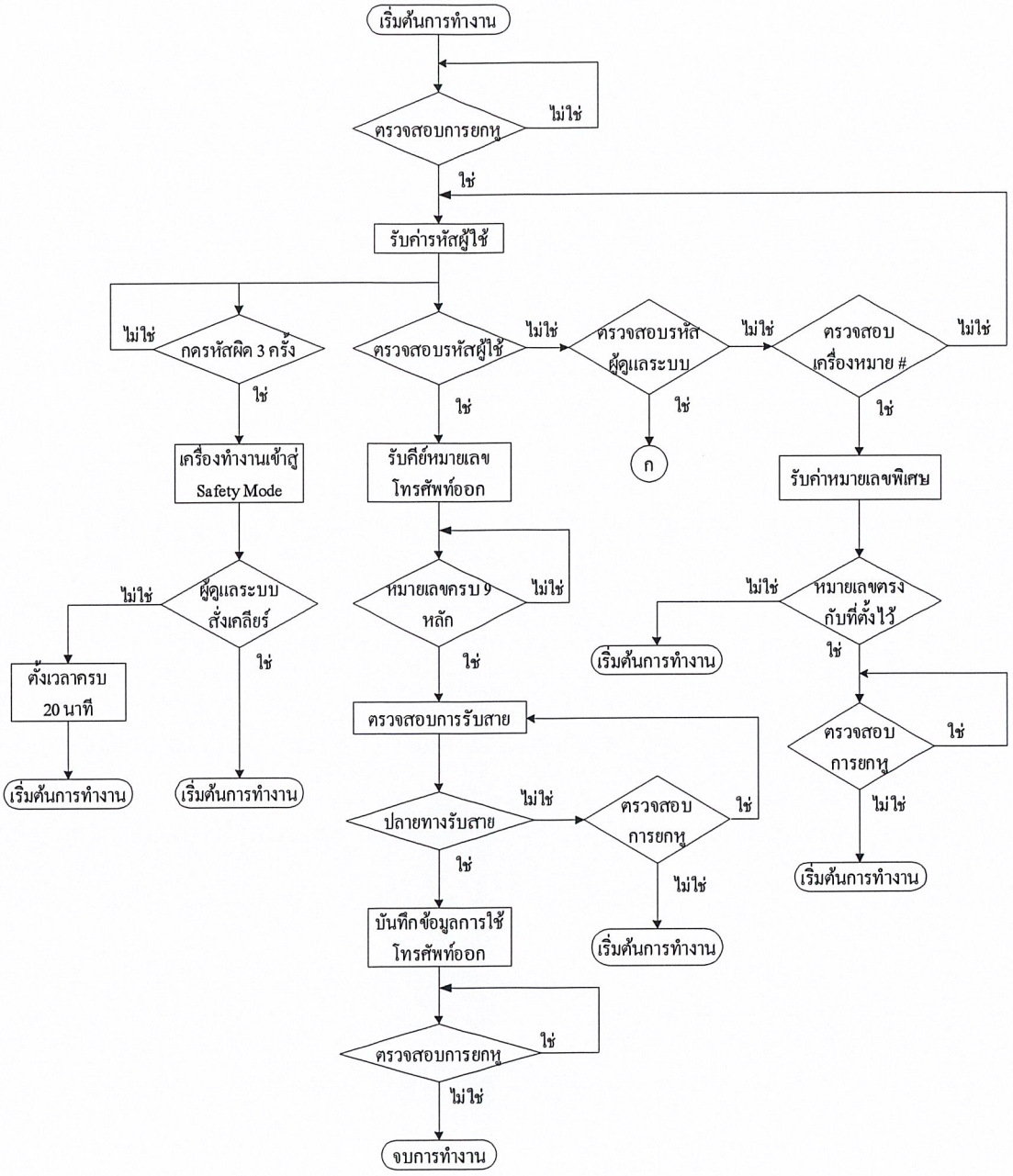
ตารางที่ ค.8 (ต่อ) รายการอุปกรณ์ของวงจรไมโครคอนโทรลเลอร์

ชื่ออุปกรณ์	รายละเอียด	จำนวน
อุปกรณ์อื่นๆ		
SW1	Power	1 ตัว
SW2	ไมโครสวิตซ์	1 ตัว
J1	Socket 4 pin	1 ตัว
J2	Socket 8 pin	1 ตัว
J3	DB-9	1 ตัว
J4	IDC Connector 16 pin	1 ตัว
LCD	20 ตัวอักษร 2 บรรทัด	1 ตัว

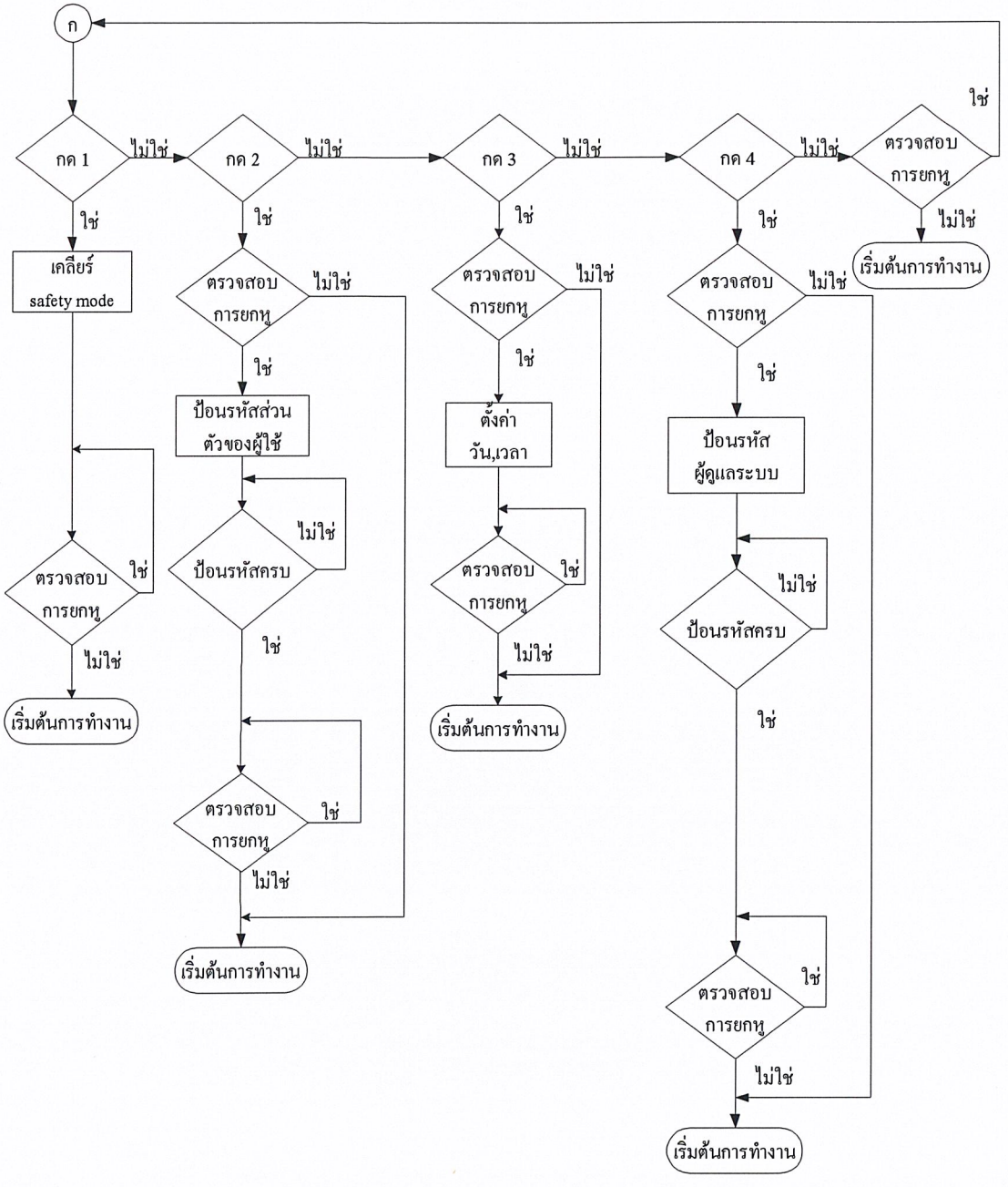
ตารางที่ ค.9 รายการอุปกรณ์ของวงจรแหล่งจ่ายไฟ

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
IC1	7805	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
BRIDGE	CP601 6A	1 ตัว
ตัวเก็บประจุ		
C1	1000 µF 50 V	1 ตัว
C2, C3	0.1 µF	2 ตัว
C4	100 µF 50 V	1 ตัว
อุปกรณ์อื่นๆ		
J1	Socket 3 pin	1 ตัว
J2-J6	Socket 2 pin	5 ตัว

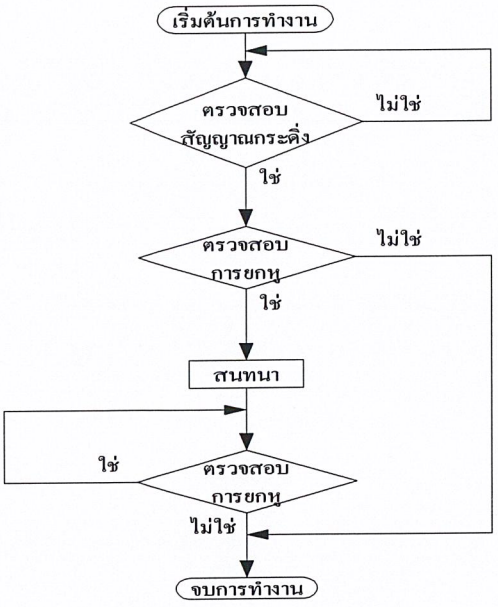
ภาคผนวก ง
แผนผังการทำงานและรหัสต้นฉบับของโปรแกรม



รูปที่ ง.1 แผนผังการทำงานของภาคโทรออก



รูปที่ ง.2 แผนผังการทำงานในส่วนของผู้ดูแลระบบ



รูปที่ ง.3 แผนผังการทำงานภาคโทรเข้า

โปรแกรมควบคุมการทำงาน

```
/*
      Program for DS1307 testing.
      Debugging by serial port 19.2k.
*/
// -------------Description---------------- //
// LCD 16x2 Using 89C51RD2
// Start on 2 Mar. 2004
//
// --------Developer Log Section ---------- //
/*
*/
// ------------ End Section ----------------//
// -------- Hardware Description ---------- //
/*
      P0   -> Data/Command of LCD

      P2.1 -> E (LCD)
      P2.0 -> RS (LCD)

      P3.6 -> SCL
      P3.7 -> SDA

      P1.0 -> Wrong password
      P1.1 -> Admin Clear
      P1.2 -> Q4
      P1.3 -> Q3
      P1.4 -> Q2
      P1.5 -> Q1
      P1.6 -> Hook
      P1.7 -> Back Light

      P2.2 -> RY1
      P2.3 -> RY3
      P2.4 -> RingBack
      P2.5 -> Ringing

      INT0 -> STD

      DS1307 -> Real Time Clock
         -> IIC Address = 1101000 (R/W*)
      24C256 -> Serial EEPROM 256 kBits
               -> IIC Address -> A2=A1=A0=0
                                    ->  (1010) (0) (0) (0) (R/W*)

*/

#include <reg52.h>
#include <absacc.h>
#include <assert.h>
#include <ctype.h>
#include <intrins.h>
#include <math.h>
#include <setjmp.h>
#include <stdarg.h>
#include <stddef.h>
```

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

#define SPNumber1 189
#define SPNumber2 191
#define SPNumber3 195
#define SPNumber4 198
#define SPNumber5 199
#define SPNumber6  1133
#define SPNumber7  1134
#define SPNumber8  1177
#define SPNumber9  1190
#define SPNumber10 1578

#define DataBUS P0
#define LCD_CMD_INI      0x38 //Initial
#define LCD_CMD_CLR      0x01 //Clear LCD
#define LCD_CMD_HOME     0x02 //Cursor Home
#define LCD_CMD_AUTOINC  0x06 //Auto Increment
#define LCD_CMD_OFF      0x08 //Off LCD
#define LCD_CMD_ON       0x0c //On LCD
#define LCD_CMD_BLINK    0x0f //Cursor on & Blink
#define LCD_CMD_LSHF     0x18 //Shift Display to Left
#define LCD_CMD_RSHF     0x1c //Shift Display to Right

sbit En_LCD  = P2^1;
sbit RS_LCD  = P2^0;
sbit SCL  = P3^6;                // I2C I/O Bit
sbit SDA  = P3^7;

sbit Wrong_Password = P1^0;// -> Wrong password
sbit Admin_Clear = P1^1;// -> Admin Clear
sbit Q4 = P1^2;// -> Q4
sbit Q3 = P1^3;// -> Q3
sbit Q2 = P1^4;// -> Q2
sbit Q1 = P1^5;// -> Q1
sbit Hook = P1^6;// -> Hook
sbit BackLight = P1^7;// -> Back Light

sbit RY1 = P2^2;// -> RY1
sbit RY3 = P2^3;// -> RY3
sbit RingBack = P2^4;// -> RingBack
sbit Ringing = P2^5;// -> Ringing
sbit STD = P3^2;// -> STD

bit STD_Flag=0;
bit Hook_Flag=1;

//unsigned char EEPROM_ADDR = 0xA0;
#define EEPROM_ADDR  0xA0

void ExternalInterrupt0(void) interrupt 0
{
      //Ex-Interrupt 1 for adjust offset
      EX0 = 0;//Prevent multiple interrupt

      STD_Flag = 1;
```

```
	EX0=1;//Enable Ex-Interrupr1
}

void dmsec(int icount) //reentrant
{
	//Routine for Delay = count msec
	unsigned int i;
	while(icount) {
		for (i=0;i<335;i++); //for RD2 at 11.059MHz
		icount--;
}
}
void EnableLCD(void)
{
	En_LCD = 1;
	dmsec(1);
	En_LCD = 0;
//	dmsec(1);
}

void SetAddress(unsigned char Address)
{
	RS_LCD = 0;
	DataBUS = Address|0x80;
	EnableLCD();
}
void PutCommand(unsigned char Command)
{
	RS_LCD = 0;
	DataBUS = Command;
	EnableLCD();
}

void PutData(unsigned char Data)
{
	RS_LCD = 1;
	DataBUS = Data;
	EnableLCD();
	PutCommand(LCD_CMD_ON);
}

void DisplayLCD(unsigned char *tmp1,unsigned char Row,unsigned char
Column)
{
	unsigned char i,position;
	switch (Row)
	{
	case 1:
		position = 0x00+Column;
		SetAddress(position);
		break;
	case 2:
		position = 0x40+Column;
		SetAddress(position);
		break;
	}

	i=0;
```

```
	while(tmp1[i]!='\0')
	{
		PutData(tmp1[i]);
		PutCommand(LCD_CMD_ON);
		i=i++;
	}
}

void DisplayLCD1C(unsigned char *tmp1,unsigned char Row,unsigned
char Column)
{
	unsigned char position;
	switch (Row)
	{
	case 1:
		position = 0x00+Column;
		SetAddress(position);
		break;
	case 2:
		position = 0x40+Column;
		SetAddress(position);
		break;
	}

	PutData(tmp1[0]);
	PutCommand(LCD_CMD_ON);
}

void d10usec(unsigned char icount)
{
	//Routine for Delay = count usec
	unsigned char i;
	while(icount) {
		for (i=0;i<2;i++); //for RD2 at 11.059MHz
		icount--;
	}
}

void d5usec()
{
	_nop_();_nop_();_nop_();_nop_();_nop_();
}


void I2C_Delay (void) {                    // I2C delay
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
```

```
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
    _nop_ ();
}

void I2C_high (void) {                        // I2C clock high
    SCL = 1;
    I2C_Delay();
}

void I2C_low (void) {                         // I2C clock low
    SCL = 0;
    I2C_Delay();
}

void I2C_start (void) {                        // start condition
    SDA = 1;
    I2C_high();
    SDA = 0;
```

```
    I2C_Delay();
    I2C_low();
//    SDA = 1;
    I2C_Delay();
}

void I2C_stop (void) {                    // stop condition
    SDA = 0;
    I2C_Delay();
    I2C_high();
    SDA = 1;
    I2C_Delay();
}

bit I2C_wrbyte (unsigned dat) {           // write one byte for
ds1307
    unsigned char i;                     // return 0 = ok
    bit outbit;                          // return 1 = error
    for (i=1;i<=8;i++)
      {
        outbit = dat & 0x80;
        SDA = outbit;
        dat = dat << 1;
        I2C_high ();
        I2C_low ();
    }
    SDA = 1;
    I2C_Delay();
    I2C_high ();
    outbit = SDA;
    I2C_Delay();
    I2C_low ();
    I2C_Delay();
    return (outbit);
}

unsigned char I2C_rdbyte () {             // read last byte for
ds1307 (not ack)
    unsigned char i,dat;
    bit inbit;
    dat = 0;
      SDA=1;
    for (i=1;i<=8;i++) {
        I2C_high ();
        inbit = SDA;
        I2C_low ();
        dat = dat << 1;
        dat = dat | inbit;
    }
    SDA = 1;
    I2C_Delay();
      I2C_high ();
      inbit=SDA;
    I2C_low ();
      if(~inbit)
            dat=0x00;
    return (dat);
}
```

```
/*
//############################ EEPROM Map
############################
// Total 1160 Usage Record -> 32480 Byte

// Header 82 Byte -> Start at 0x00
// Reserved
// First record start at 0x0120

      +-----------------+
      | CurrentAddressL | Current Address (2 Byte) -->When UpLoad
It'll be cleared automatically
      +-----------------+
      | CurrentAddressH |
      +-----------------+
      |   User1_Byte0   | User Record (20x4 Byte) L->H
      +-----------------+
      |        .        |
      +-----------------+
      |        .        |
      +-----------------+
      |  User1_Byte20   |
      +-----------------+
      |        .        |
      +-----------------+
      |        .        |
      +-----------------+
      |        .        |
      +-----------------+
      |   User_Byte0    | Start Usage Data Record 0 (28 Byte)
      +-----------------+
      |   User_Byte1    |
      +-----------------+
      |   User_Byte2    |
      +-----------------+
      |   User_Byte3    |
      +-----------------+
      | Tel. No. Byte0  |
      +-----------------+
      | Tel. No. Byte1  |
      +-----------------+
      | Tel. No. Byte2  |
      +-----------------+
      | Tel. No. Byte3  |
      +-----------------+
      | Tel. No. Byte4  |
      +-----------------+
      | Tel. No. Byte5  |
      +-----------------+
      | Tel. No. Byte6  |
      +-----------------+
      | Tel. No. Byte7  |
      +-----------------+
      | Tel. No. Byte8  |
      +-----------------+
      |   Start Date    |
      +-----------------+
      |   Start Month   |
```

```
      +-----------------+
      |   Start Year    |
      +-----------------+
      |    Start Sec    |
      +-----------------+
      |    Start Min    |
      +-----------------+
      |    Start Hr     |
      +-----------------+
      |     End Date    |
      +-----------------+
      |     End Month   |
      +-----------------+
      |     End Year    |
      +-----------------+
      |      End Sec    |
      +-----------------+
      |      End Min    |
      +-----------------+
      |      End Hr     |
      +-----------------+
      |     Intv Sec    |
      +-----------------+
      |     Intv Min    |
      +-----------------+
      |     Intv Hr     | End Usage Data Record 0
      +-----------------+

//############################ EEPROM Map
############################
*/

//############################## EEPROM
##############################

/*
void WriteEEPROM(unsigned char EEPROM_Addr,AdrressH,AdderssL,Data)
{
      I2C_start();
      I2C_wrbyte(EEPROM_Addr);
      I2C_wrbyte(AdrressH);
      I2C_wrbyte(AdderssL);
      I2C_wrbyte(Data);
      I2C_stop();
      dmsec(5);
}
unsigned char ReadEEPROM(unsigned char
EEPROM_Addr,AdrressH,AdderssL)
{
      unsigned char tmp;
      //Read
    I2C_high ();
      I2C_start();
      I2C_wrbyte(EEPROM_Addr);
      I2C_wrbyte(AdrressH);
      I2C_wrbyte(AdderssL);

   I2C_high ();
```

```
      I2C_start();
      I2C_wrbyte(EEPROM_Addr+1);
      tmp = I2C_rdbyte();
      I2C_stop();
      dmsec(5);
      return tmp;
}
*/

void iWriteEEPROM(unsigned char EEPROM_Addr,unsigned int
ADDR,unsigned char Data)
{
      unsigned char AdrressH,AdderssL;
      AdrressH = (ADDR&0xff00)>>8;
      AdderssL = ADDR&0x00ff;

      I2C_start();
      I2C_wrbyte(EEPROM_Addr);
      I2C_wrbyte(AdrressH);
      I2C_wrbyte(AdderssL);
      I2C_wrbyte(Data);
      I2C_stop();
      dmsec(5);
}

unsigned char iReadEEPROM(unsigned char EEPROM_Addr,unsigned int
ADDR)
{
      unsigned char AdrressH,AdderssL;
      unsigned char tmp;

      AdrressH = (ADDR&0xff00)>>8;
      AdderssL = ADDR&0x00ff;

      //Read
    I2C_high ();
      I2C_start();
      I2C_wrbyte(EEPROM_Addr);
      I2C_wrbyte(AdrressH);
      I2C_wrbyte(AdderssL);

    I2C_high ();
      I2C_start();
      I2C_wrbyte(EEPROM_Addr+1);
      tmp = I2C_rdbyte();
      I2C_stop();
      dmsec(5);
      return tmp;
}

//unsigned char RTC_ADDR = 0xD0;
#define RTC_ADDR 0xD0


typedef struct CLOCK{
            char sec;
            char min;
            char hour;
```

```
            char day;
            char date;
            char month;
            char year;
            }CLOCK;

typedef struct TIME{
            char sec;
            char min;
            char hour;
            char date;
            }TIME;

//unsigned char    TIMBUF[8];            // ss,mm,hh,ww,dd,mm,yy,cc


unsigned int btoi (unsigned char bcd)
{               // change bcd to hex(int)

      unsigned char    STRBUF[3];            // string buffer

    STRBUF[0] = ((bcd & 0xf0) >> 4) | 0x30;           // to string
first
    STRBUF[1] = (bcd & 0x0f) | 0x30;
    STRBUF[2] = 0;
    return (atoi (STRBUF));                           // string to
int
}

void RTC_WRITE(unsigned char rtc_addr, unsigned char ad_in,unsigned
char dat)
{
      I2C_start();
      I2C_wrbyte(rtc_addr);
      I2C_wrbyte(ad_in);
      I2C_wrbyte(dat);
      I2C_stop();
      dmsec(5);
}
void RTC_WRITE_DATA()
{
CLOCK RTC;
      RTC_WRITE(RTC_ADDR,0,RTC.sec);
      RTC_WRITE(RTC_ADDR,1,RTC.min);
      RTC_WRITE(RTC_ADDR,2,RTC.hour);
      RTC_WRITE(RTC_ADDR,3,RTC.day);
      RTC_WRITE(RTC_ADDR,4,RTC.date);
      RTC_WRITE(RTC_ADDR,5,RTC.month);
      RTC_WRITE(RTC_ADDR,6,RTC.year);
}

unsigned char RTC_READ(unsigned char rtc_addr,unsigned char
ad_in)//Return BCD
{
      unsigned char tmp;
    I2C_high ();
      I2C_start();
      I2C_wrbyte(rtc_addr);
```

```
      I2C_wrbyte(ad_in);

    I2C_high ();
      I2C_start();
      I2C_wrbyte(rtc_addr+1);
      tmp = I2C_rdbyte();
      I2C_stop();
      dmsec(1);
      return(tmp);
}
CLOCK RTC_READ_DATA()//Return BCD
{
      CLOCK rtc;
      rtc.sec = RTC_READ(RTC_ADDR,0);
      rtc.min = RTC_READ(RTC_ADDR,1);
      rtc.hour = RTC_READ(RTC_ADDR,2);
      rtc.day = RTC_READ(RTC_ADDR,3);
      rtc.date = RTC_READ(RTC_ADDR,4);
      rtc.month = RTC_READ(RTC_ADDR,5);
      rtc.year = RTC_READ(RTC_ADDR,6);
      return rtc;
}

TIME RTC_READ_TIME()
{
      TIME rtc;
      rtc.sec = RTC_READ(RTC_ADDR,0);
      rtc.min = RTC_READ(RTC_ADDR,1);
      rtc.hour = RTC_READ(RTC_ADDR,2);
      rtc.date = RTC_READ(RTC_ADDR,4);
      return rtc;
}
/*
void BCD2LCD(unsigned char BCD,row,col)
{
      unsigned char tmp[1];
      tmp[0]= BCD&0xf0;
      tmp[0] = tmp[0]>>4;
      tmp[0] = tmp[0]+0x30;
      DisplayLCD(tmp,row,col);

      tmp[0] = BCD&0x0f;
      tmp[0] = tmp[0]+0x30;
      DisplayLCD(tmp,row,col+1);
}
*/
unsigned char Byte2BCD(unsigned char Byte1,unsigned char Byte2)
{
      unsigned char tmp[1],BCD;
      tmp[0] = _crol_(Byte1,4);
      BCD = tmp[0]|Byte2;
      return BCD;
}

TIME ElapsedTime(TIME Time1,TIME Time2)//Return Elapsed Time in
integer form (Not BCD)
{
      TIME tmp1;
```

```
TIME TimeHex1,TimeHex2;
TIME Elapsed;
TIME Elapsed1;
TIME Elapsed2;
TimeHex1.sec = btoi(Time1.sec);
TimeHex1.min = btoi(Time1.min);
TimeHex1.hour = btoi(Time1.hour);
TimeHex1.date = btoi(Time1.date);

TimeHex2.sec = btoi(Time2.sec);
TimeHex2.min = btoi(Time2.min);
TimeHex2.hour = btoi(Time2.hour);
TimeHex2.date = btoi(Time2.date);

if(TimeHex1.date==TimeHex2.date) //Same Date
{
      if(TimeHex2.sec < TimeHex1.sec)
      {
            TimeHex2.sec = TimeHex2.sec+60;
            TimeHex2.min = TimeHex2.min-1;
      }
      if(TimeHex2.min < TimeHex1.min)
      {
            TimeHex2.min = TimeHex2.min+60;
            TimeHex2.hour = TimeHex2.hour-1;
      }
      Elapsed.sec = TimeHex2.sec-TimeHex1.sec;
      Elapsed.min = TimeHex2.min-TimeHex1.min;
      Elapsed.hour = TimeHex2.hour-TimeHex1.hour;
}
else
{
      //If different date we must seperate into 2 Parts.
      //The first part is time before 24.00.
      //The second part is time after 24.00.
      tmp1.sec = 00;
      tmp1.min = 00;
      tmp1.hour = 24;
      if(tmp1.sec < TimeHex1.sec)
      {
            tmp1.sec = tmp1.sec+60;
            tmp1.min = tmp1.min-1;
      }
      if(tmp1.min < TimeHex1.min)
      {
            tmp1.min = tmp1.min+60;
            tmp1.hour = tmp1.hour-1;
      }
      Elapsed1.sec = tmp1.sec-TimeHex1.sec;
      Elapsed1.min = tmp1.min-TimeHex1.min;
      Elapsed1.hour = tmp1.hour-TimeHex1.hour;

      //Second Part
      Elapsed2.sec = TimeHex2.sec;
      Elapsed2.min = TimeHex2.min;
      Elapsed2.hour = TimeHex2.hour;

      //Merge Time
```

```
            Elapsed.sec = Elapsed1.sec + Elapsed2.sec;
            Elapsed.min = Elapsed1.min + Elapsed2.min;
            Elapsed.hour = Elapsed1.hour + Elapsed2.hour;
      }
      return Elapsed;
}

unsigned char  MemADDRH;
unsigned char  MemADDRL;
unsigned int  MemADDR;

void SerialInterrupt(void)interrupt 4
{
unsigned int  i;
unsigned char tmp,tmpdata;
ES=0;
      if(TI == 1)
      {
            TI = 0;
      }
      if(RI == 1)
      {
            tmp=SBUF;
            if(tmp == 161)//Open Serial Comm---->A1
            {

                        MemADDRL = iReadEEPROM(EEPROM_ADDR,0x0000);
                        MemADDRH = iReadEEPROM(EEPROM_ADDR,0x0001);



                        MemADDR = 0;
                        MemADDR = MemADDRH;
                        MemADDR = (MemADDR<<8)|MemADDRL;


                        DisplayLCD("     UPLOADING      ",1,0);
                        DisplayLCD("    PLEASE WAIT     ",2,0);


                        for(i=0x0120;i<=MemADDR;i++)
                        {
                              tmpdata = iReadEEPROM(EEPROM_ADDR,i);
                              SBUF = tmpdata;
                              dmsec(100);
                        }

/*
|   User_Byte0    |   User_Byte1    |   User_Byte2    |
User_Byte3    |
| Tel. No. Byte0  | Tel. No. Byte1  | Tel. No. Byte2  | Tel. No.
Byte3  |
| Tel. No. Byte4  | Tel. No. Byte5  | Tel. No. Byte6  | Tel. No.
Byte7  | Tel. No. Byte8  |
|   Start Date    |   Start Month   |   Start Year    |
|    Start Sec    |    Start Min    |    Start Hr     |
|     End Date    |    End Month    |     End Year    |
|     End Sec     |     End Min     |     End Hr      |
```

```
|     Intv Sec    |    Intv Min     |     Intv Hr     | */

                        //Clear Address
                        MemADDR = 0x120;
                        MemADDRH = (MemADDR&0xff00)>>8;
                        MemADDRL = MemADDR&0x00ff;
                        iWriteEEPROM(EEPROM_ADDR,0x0000,MemADDRL);
                        iWriteEEPROM(EEPROM_ADDR,0x0001,MemADDRH);
                        DisplayLCD("                    ",1,0);
                        DisplayLCD("                    ",2,0);

            }
            if(tmp == 164)//Close Serial Comm------>A4
            {

            }
            RI = 0;
      }

      ES=1;
}
unsigned char DateTime1[3];// Date/Mon/Year
unsigned char DateTime2[3];// Date/Mon/Year
unsigned char DTMF_KEY[9];//for receive DTMF key

void main (void)
{
CLOCK FullTime1;
unsigned char DisplayDTMF_KEY[9];//for Display DTMF key
unsigned char PasswdDTMF_KEY[4];//for Display DTMF key
unsigned char tmp_KEY[4];//for format key

      TIME Time1, Time2;
      unsigned char hr,hr1;
      unsigned char KeyCount;
      unsigned char WrongCount;

//unsigned char  MemADDRH;
//unsigned char  MemADDRL;
//unsigned int  MemADDR;
unsigned char Passwd[4];

      bit bQ4,bQ3,bQ2,bQ1;
      bit Received_Flag;//Flag to indicate receiveing call in.
      bit TelFlag;//
      bit PasswdTRUE;
      bit AdminCLR_Flag;

      TL1 = 0xfa;//9600 baud
      TH1 = 0xfa;
      PCON = 0x80;
      TMOD = 0x21;
      SCON = 0x52;
      TR1=1;
      EA = 1;

      EX0 = 0;
```

```
        ES = 1; //Enable Serial Interrupt
/*
        iWriteEEPROM(EEPROM_ADDR,0x0002,0x01);
        iWriteEEPROM(EEPROM_ADDR,0x0003,0x02);
        iWriteEEPROM(EEPROM_ADDR,0x0004,0x03);
        iWriteEEPROM(EEPROM_ADDR,0x0005,0x04);

        iWriteEEPROM(EEPROM_ADDR,0x0006,0x02);
        iWriteEEPROM(EEPROM_ADDR,0x0007,0x04);
        iWriteEEPROM(EEPROM_ADDR,0x0008,0x06);
        iWriteEEPROM(EEPROM_ADDR,0x0009,0x08);

        iWriteEEPROM(EEPROM_ADDR,0x0053,0x01);//Admin Passwd
        iWriteEEPROM(EEPROM_ADDR,0x0054,0x01);
*/

/*

        Wrong_Password = 0;

        hr = iReadEEPROM(EEPROM_ADDR,0x0002);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0003);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0004);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0005);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0006);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0007);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0008);
        SBUF = hr;
        dmsec(100);
        hr = iReadEEPROM(EEPROM_ADDR,0x0009);
        SBUF = hr;
        dmsec(100);

*/

/*      iWriteEEPROM(EEPROM_ADDR,0x0000,0x2f);
        iWriteEEPROM(EEPROM_ADDR,0x0001,0x01);

        iWriteEEPROM(EEPROM_ADDR,0x0120,0xef);
        iWriteEEPROM(EEPROM_ADDR,0x0121,0xaa);

//      hr = iReadEEPROM(EEPROM_ADDR,0x0120);
//      SBUF = hr;
//      dmsec(100);
//      hr = iReadEEPROM(EEPROM_ADDR,0x0121);
//      SBUF = hr;
```

```
//    dmsec(100);



      WriteEEPROM(EEPROM_ADDR,0x00,0x00,0xac);
      WriteEEPROM(EEPROM_ADDR,0x00,0x01,0xad);
      WriteEEPROM(EEPROM_ADDR,0x00,0x02,0xae);
      WriteEEPROM(EEPROM_ADDR,0x00,0x03,0xaf);

      hr = ReadEEPROM(EEPROM_ADDR,0x00,0x00);
      SBUF = hr;
      dmsec(100);
      hr = ReadEEPROM(EEPROM_ADDR,0x00,0x01);
      SBUF = hr;
      dmsec(100);
      hr = ReadEEPROM(EEPROM_ADDR,0x00,0x02);
      SBUF = hr;
      dmsec(100);
      hr = ReadEEPROM(EEPROM_ADDR,0x00,0x03);
      SBUF = hr;
      dmsec(100);
*/
Wrong_Password=0;
      PutCommand(LCD_CMD_INI);
      PutCommand(LCD_CMD_OFF);
      PutCommand(LCD_CMD_AUTOINC);
      PutCommand(LCD_CMD_HOME);
      PutCommand(LCD_CMD_CLR);

//    SBUF = Byte2BCD(5,9);
//    dmsec(50);

//    SDA=1;
//    I2C_high ();

//    RTC_WRITE(RTC_ADDR, 0,0);
//    RTC_WRITE(RTC_ADDR, 1,0x59);
//    hr = Byte2BCD(2,3);
//    hr = hr&0x3F;//Hr in 24 hr
//    RTC_WRITE(RTC_ADDR, 2,hr);
//    RTC_WRITE(RTC_ADDR, 4,1);

      WrongCount = 0;
      while(1)
      {
main:

            AdminCLR_Flag = 1;

            Hook = 1;//Prepare Input
            while(Hook==1)
            {

                  RY1 = 0;

                  RY3 = 0;
                  DisplayLCD("       KMIT'L       ",1,0);
                  FullTime1 = RTC_READ_DATA();
```

```
//          BCD2LCD(FullTime1.date,2,0);
//          BCD2LCD(FullTime1.month,2,3);
//          BCD2LCD(FullTime1.year,2,6);

hr1= FullTime1.date&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,0);
hr1 = FullTime1.date&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,1);

hr1= FullTime1.month&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,3);
hr1 = FullTime1.month&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,4);

hr1= FullTime1.year&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,6);
hr1 = FullTime1.year&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,7);

hr1= FullTime1.sec&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,18);
hr1 = FullTime1.sec&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,19);

hr1= FullTime1.min&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,15);
hr1 = FullTime1.min&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,16);

hr1= FullTime1.hour&0xf0;
hr1 = hr1>>4;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,12);
hr1 = FullTime1.hour&0x0f;
hr1 = hr1+0x30;
DisplayLCD1C(&hr1,2,13);

      //    BCD2LCD(FullTime1.sec,2,18);
      //    BCD2LCD(FullTime1.min,2,15);
      //    BCD2LCD(FullTime1.hour,2,12);
            DisplayLCD("    ",2,8);
            DisplayLCD("/",2,2);
            DisplayLCD("/",2,5);
```

```
                  DisplayLCD(":",2,17);
                  DisplayLCD(":",2,14);
            //###################### Line In
############################
                  Ringing=1;
                  if(Ringing==1)
                  {
                        Time1 = RTC_READ_TIME();
                        DisplayLCD("      LINE IN       ",1,0);
                        DisplayLCD(" WAIT FOR RECEIVING ",2,0);
                        Received_Flag=0;
                        while(Hook==1)
                        {
                              Time2 = RTC_READ_TIME();
                              //Cal. Interval
                              Time2 = ElapsedTime(Time1,Time2);

                  //          SBUF = Time2.min;
                              dmsec(100);
                              if(Time2.min>=1)//Time out then clear
LCD and exit wait loop
                              {
                                    DisplayLCD("
",1,0);
                                    DisplayLCD("
",2,0);
                                    break;
                              }
                              while(Hook==0)//Receive
                              {
                                    DisplayLCD("      LINE IN
",1,0);
                                    DisplayLCD("     RECEIVING
",2,0);
                                    //Talk
                                    Received_Flag=1;
                              }
                              if(Received_Flag==1)//After receive
Clear LCD
                              {
                                    DisplayLCD("
",1,0);
                                    DisplayLCD("
",2,0);
                                    break;
                              }
                        }//while hook
                  }//ring
            }//while hook
            //###################### Line In
############################

            //###################### Call. Out
############################
            PutCommand(LCD_CMD_CLR);
            RY1 = 1;
            //Goto Get Password Menu
            DisplayLCD("PASSWORD : ",1,0);
```

```
            //Wait for key
            STD = 1;
            Q4 = 1;
            Q3 = 1;
            Q2 = 1;
            Q1 = 1;

            TelFlag = 0;
            for(KeyCount=0;KeyCount<=3;KeyCount++)
            {
                  if(Hook==1) break;

                  DTMF_KEY[KeyCount] = 0;
                  while(STD==0)
                  {
                        if(Hook==1) break;
                  }
                  if(Hook==1) break;

                  STD_Flag = 0;
                  dmsec(300);
                  bQ4 = Q4;
                  bQ3 = Q3;
                  bQ2 = Q2;
                  bQ1 = Q1;
                  tmp_KEY[3] = bQ4&0x01;
                  tmp_KEY[2] = bQ3&0x01;
                  tmp_KEY[1] = bQ2&0x01;
                  tmp_KEY[0] = bQ1&0x01;

                  DTMF_KEY[KeyCount] =
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);

                  if(DTMF_KEY[KeyCount]==0x0a)
                  {
                        PasswdDTMF_KEY[KeyCount] =
DTMF_KEY[KeyCount] + 0x30-10;
//
      DisplayLCD1C(&PasswdDTMF_KEY[KeyCount],1,11+KeyCount);
                        DisplayLCD1C("*",1,11+KeyCount);
                  }
                  else
                  {
                        PasswdDTMF_KEY[KeyCount] =
DTMF_KEY[KeyCount] +0x30;
//
      DisplayLCD1C(&PasswdDTMF_KEY[KeyCount],1,11+KeyCount);
                        DisplayLCD1C("*",1,11+KeyCount);
                  }
            }


            //Search passwd;
            PasswdTRUE = 0;
            for(hr=0;hr<=(20)-1;hr++)
            {
                  Passwd[0] =
iReadEEPROM(EEPROM_ADDR,(hr*4)+0+2);//Add 2 cause start user
```

```
address at 0x02
                  Passwd[1] = iReadEEPROM(EEPROM_ADDR,(hr*4)+1+2);
                  Passwd[2] = iReadEEPROM(EEPROM_ADDR,(hr*4)+2+2);
                  Passwd[3] = iReadEEPROM(EEPROM_ADDR,(hr*4)+3+2);

                  if((DTMF_KEY[0]==Passwd[0])&&
                        (DTMF_KEY[1]==Passwd[1])&&
                        (DTMF_KEY[2]==Passwd[2])&&
                        (DTMF_KEY[3]==Passwd[3]))
                        {
                              PasswdTRUE=1;
                              break;
                        }
            }


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ Admin Menu
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

      if(DTMF_KEY[0]==0x0b)
      {
            WrongCount = 0;

                              Passwd[0] = 0x0b;//*
                              Passwd[1] =
iReadEEPROM(EEPROM_ADDR,0x0053);
                              Passwd[2] =
iReadEEPROM(EEPROM_ADDR,0x0054);
                              Passwd[3] = 0x0c;//#

                              if((DTMF_KEY[0]==Passwd[0])&&
                                    (DTMF_KEY[1]==Passwd[1])&&
                                    (DTMF_KEY[2]==Passwd[2])&&
                                    (DTMF_KEY[3]==Passwd[3]))
                                    {
                              ////Entry Admin Menu
                                          DisplayLCD("1.SAF CLR
2.SET PAS",1,0);
                                          DisplayLCD("3.SET TIM
4.ADM PAS",2,0);
                                    }

////////////////////////////////////////////////////////////////
//////////////

                              //Wait for key
                              STD = 1;
                              Q4 = 1;
                              Q3 = 1;
                              Q2 = 1;
                              Q1 = 1;
                              if(Hook==1) break;//Break Hook
                              DTMF_KEY[0] = 0;
                              while(STD==0)
                              {
                                    if(Hook==1) break;
                              }
                              if(Hook==1) break;//Break Hook
```

```
                                STD_Flag = 0;
                                dmsec(500);
                                bQ4 = Q4;
                                bQ3 = Q3;
                                bQ2 = Q2;
                                bQ1 = Q1;
                                tmp_KEY[3] = bQ4&0x01;
                                tmp_KEY[2] = bQ3&0x01;
                                tmp_KEY[1] = bQ2&0x01;
                                tmp_KEY[0] = bQ1&0x01;

                                DTMF_KEY[0] =
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


                                switch(DTMF_KEY[0])
                                {
                                      case 1:
                                      {
                                            DisplayLCD("     SAFTY
CLEAR    ",1,0);
                                            DisplayLCD("
",2,0);
                                            WrongCount = 0;
                                            AdminCLR_Flag = 1;
                                            Admin_Clear = 0;
                                            dmsec(10);
                                            Admin_Clear = 1;
                                            dmsec(10);
                                            Admin_Clear = 0;
                                            dmsec(10);
                                            goto main;
                                            break;
                                      }
                                      case 2:
                                      {
                                            DisplayLCD("    SET PASS
No.    ",1,0);
                                            DisplayLCD("SELECT 01-
20 :      ",2,0);
                                            //################## Wait
Key ##################


      for(KeyCount=0;KeyCount<=1;KeyCount++)
                                            {
                                                  STD = 1;
                                                  Q4 = 1;
                                                  Q3 = 1;
                                                  Q2 = 1;
                                                  Q1 = 1;
                                                  if(Hook==1)
      break;//Break Hook
                                                  DTMF_KEY[KeyCount]
= 0;
                                                  while(STD==0)
                                                  {
                                                        if(Hook==1)
```

```
      break;
                                                }
                                                if(Hook==1)
      break;//Break Hook

                                                STD_Flag = 0;
                                                dmsec(500);
                                                bQ4 = Q4;
                                                bQ3 = Q3;
                                                bQ2 = Q2;
                                                bQ1 = Q1;
                                                tmp_KEY[3] =
bQ4&0x01;
                                                tmp_KEY[2] =
bQ3&0x01;
                                                tmp_KEY[1] =
bQ2&0x01;
                                                tmp_KEY[0] =
bQ1&0x01;

                                                DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,15+KeyCount);
                                          }//Loop Get Key


                                          hr =
Byte2BCD(DTMF_KEY[0],DTMF_KEY[1]);
                                          DisplayLCD("NEW
PASSWORD :      ",1,0);
                                          DisplayLCD("
",2,0);


      for(KeyCount=0;KeyCount<=3;KeyCount++)
                                          {
                                                STD = 1;
                                                Q4 = 1;
                                                Q3 = 1;
                                                Q2 = 1;
                                                Q1 = 1;
                                                if(Hook==1)
      break;//Break Hook
                                                DTMF_KEY[KeyCount]
= 0;
                                                while(STD==0)
                                                {
                                                      if(Hook==1)
```

```
      break;
                                                }
                                                if(Hook==1)
      break;//Break Hook

                                                STD_Flag = 0;
                                                dmsec(300);
                                                bQ4 = Q4;
                                                bQ3 = Q3;
                                                bQ2 = Q2;
                                                bQ1 = Q1;
                                                tmp_KEY[3] =
bQ4&0x01;
                                                tmp_KEY[2] =
bQ3&0x01;
                                                tmp_KEY[1] =
bQ2&0x01;
                                                tmp_KEY[0] =
bQ1&0x01;

                                                DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;
//
      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],1,15+KeyCount);

      DisplayLCD1C("*",1,15+KeyCount);
                                          }//Loop Get Key

                                          hr = hr-1;//Minus 1 to
decrease offset

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+2,DTMF_KEY[0]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+3,DTMF_KEY[1]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+4,DTMF_KEY[2]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+5,DTMF_KEY[3]);
                                          DisplayLCD("
COMPLETE      ",1,0);
                                          DisplayLCD("
",2,0);

                                          dmsec(1000);

                                          break;
                                    }
                                    case 3:
                                    {
```

```
                                        DisplayLCD("  SET DATE &
TIME   ",1,0);
                                        DisplayLCD("DATE :
",2,0);
                                        hr1 = 0;

      for(KeyCount=0;KeyCount<=11;KeyCount++)
                                        {
                                              if(KeyCount==2)
                                              {
      DisplayLCD("MON  :   ",2,0);
                                              }
                                              if(KeyCount==4)
                                              {
      DisplayLCD("YEAR :   ",2,0);
                                              }
                                              if(KeyCount==6)
                                              {
      DisplayLCD("HOUR :   ",2,0);
                                              }
                                              if(KeyCount==8)
                                              {
      DisplayLCD("MIN  :   ",2,0);
                                              }
                                              if(KeyCount==10)
                                              {
      DisplayLCD("SEC  :   ",2,0);
                                              }

                                              STD = 1;
                                              Q4 = 1;
                                              Q3 = 1;
                                              Q2 = 1;
                                              Q1 = 1;
                                              if(Hook==1)
      break;//Break Hook
                                              DTMF_KEY[KeyCount]
= 0;
                                              while(STD==0)
                                              {
                                                    if(Hook==1)
      break;
                                              }
                                              if(Hook==1)
      break;//Break Hook

                                              STD_Flag = 0;
                                              dmsec(300);
                                              bQ4 = Q4;
                                              bQ3 = Q3;
                                              bQ2 = Q2;
```

```
                                                    bQ1 = Q1;
                                                    tmp_KEY[3] =
bQ4&0x01;
                                                    tmp_KEY[2] =
bQ3&0x01;
                                                    tmp_KEY[1] =
bQ2&0x01;
                                                    tmp_KEY[0] =
bQ1&0x01;

                                                    DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);

                                                    if(KeyCount<2)
                                                    {

      if(DTMF_KEY[KeyCount]==0x0a)
                                                          {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                          }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                          dmsec(500);

      DisplayLCD("DATE : ",2,0);
                                                    }


      if((KeyCount>=2)&&(KeyCount<4))
                                                    {

      DisplayLCD("MON  : ",2,0);

      if(DTMF_KEY[KeyCount]==0x0a)
                                                          {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                          }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                          dmsec(500);
                                                    }

                                    if((KeyCount>=4)&&(KeyCount<6))
                                                    {

      DisplayLCD("YEAR : ",2,0);

      if(DTMF_KEY[KeyCount]==0x0a)
                                                          {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                          }
```

```
        DisplayDTMF_KEY[KeyCount] =
        DTMF_KEY[KeyCount] +0x30;

    DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
                                                }


    if((KeyCount>=6)&&(KeyCount<8))
                                                {

    DisplayLCD("HOUR : ",2,0);

    if(DTMF_KEY[KeyCount]==0x0a)
                                                    {

    DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }

    DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

    DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
                                                }



    if((KeyCount>=8)&&(KeyCount<10))
                                                {

    DisplayLCD("MIN  : ",2,0);

    if(DTMF_KEY[KeyCount]==0x0a)
                                                    {

    DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }

    DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

    DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
                                                }

                                                if(KeyCount>=10)
                                                {

    DisplayLCD("SEC  : ",2,0);

    if(DTMF_KEY[KeyCount]==0x0a)
                                                    {

    DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }

    DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

    DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
```

```
                                                dmsec(500);
                                          }


                                    hr1 = hr1+1;
                                    if(hr1>=2)
                                          hr1 = 0;
                                    }//Loop Get Key


                                    hr =
Byte2BCD(DTMF_KEY[10],DTMF_KEY[11]);
                                    RTC_WRITE(RTC_ADDR,
0,hr);//sec

                                    hr =
Byte2BCD(DTMF_KEY[8],DTMF_KEY[9]);
                                    RTC_WRITE(RTC_ADDR,
1,hr);//sec

                                    hr =
Byte2BCD(DTMF_KEY[6],DTMF_KEY[7]);
                                    hr = hr&0x3F;//Hr in 24
hr
                                    RTC_WRITE(RTC_ADDR,
2,hr);

                                    hr =
Byte2BCD(DTMF_KEY[4],DTMF_KEY[5]);
                                    RTC_WRITE(RTC_ADDR,
6,hr);//Year

                                    hr =
Byte2BCD(DTMF_KEY[2],DTMF_KEY[3]);
                                    RTC_WRITE(RTC_ADDR,
5,hr);//month

                                    hr =
Byte2BCD(DTMF_KEY[0],DTMF_KEY[1]);
                                    RTC_WRITE(RTC_ADDR,
4,hr);//date

                                    break;
                              }
                              case 4:
                              {
                                    DisplayLCD("   SET ADMIN
PASS   ",1,0);
                                    DisplayLCD("NEW
PASSWORD :      ",2,0);

      for(KeyCount=0;KeyCount<=1;KeyCount++)
                                    {
                                          STD = 1;
                                          Q4 = 1;
                                          Q3 = 1;
```

```
                                                    Q2 = 1;
                                                    Q1 = 1;
                                                    if(Hook==1)
      break;//Break Hook
                                                    DTMF_KEY[KeyCount]
= 0;
                                                    while(STD==0)
                                                    {
                                                          if(Hook==1)
      break;
                                                    }
                                                    if(Hook==1)
      break;//Break Hook

                                                    STD_Flag = 0;
                                                    dmsec(500);
                                                    bQ4 = Q4;
                                                    bQ3 = Q3;
                                                    bQ2 = Q2;
                                                    bQ1 = Q1;
                                                    tmp_KEY[3] =
bQ4&0x01;
                                                    tmp_KEY[2] =
bQ3&0x01;
                                                    tmp_KEY[1] =
bQ2&0x01;
                                                    tmp_KEY[0] =
bQ1&0x01;

                                                    DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                    {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;
//
      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,15+KeyCount);

      DisplayLCD1C("*",2,15+KeyCount);
                                              }//Loop Get Key

      iWriteEEPROM(EEPROM_ADDR,0x0053,DTMF_KEY[0]);//Admin Passwd

      iWriteEEPROM(EEPROM_ADDR,0x0054,DTMF_KEY[1]);
                                              DisplayLCD("
COMPLETE      ",1,0);
                                              DisplayLCD("
",2,0);

                                              dmsec(1000);

                                              break;
                                        }
```

```
                              }
        }//if *
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$ Admin Menu
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


            if(PasswdTRUE==0)
            {
                  WrongCount = WrongCount+1;

                  if(WrongCount==3)
                  {
                        //Goto Safty Mode
                        DisplayLCD("     SAFTY MODE      ",1,0);
                        DisplayLCD("                     ",2,0);
                        Wrong_Password = 0;
                        dmsec(1);
                        Wrong_Password = 1;
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
                        dmsec(100000);
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%


                        Wrong_Password = 0;
                        RY1=1;

                        AdminCLR_Flag=0;
                        while(AdminCLR_Flag==0)
                        {
                              //Read Time until min = 20
///////////////////////////////////////////////////////////////////
//////////////

                              while(Hook==1)
                              {
                                    dmsec(1000);
                                    RY1 = 0;
                        //          RY3 = 0;

                                    //###################### No
Line ######################
                                    DisplayLCD("     SAFTY MODE
",1,0);
                                    DisplayLCD("
```

```
",2,0);

                                    //######################## Line
In ############################
                                    Ringing=1;

                                    if(Ringing==1)
                                    {
                                          Time1 = RTC_READ_TIME();
                                          DisplayLCD("      LINE IN
",1,0);
                                          DisplayLCD(" WAIT FOR
RECEIVING ",2,0);
                                          Received_Flag=0;
                                          while(Hook==1)
                                          {
                                                Time2 =
RTC_READ_TIME();
                                                //Cal. Interval
                                                Time2 =
ElapsedTime(Time1,Time2);

      if(Time2.min>=1)//Time out then clear LCD and exit wait loop
                                                {

                                                      DisplayLCD("
",1,0);
                                                      DisplayLCD("
",2,0);
                                                      break;
                                                }

      while(Hook==0)//Receive
                                                {
                                                      DisplayLCD("
LINE IN       ",1,0);
                                                      DisplayLCD("
RECEIVING      ",2,0);
                                                      //Talk

      Received_Flag=1;
                                                }

      if(Received_Flag==1)//After receive Clear LCD
                                                {
                                                      DisplayLCD("
",1,0);
                                                      DisplayLCD("
",2,0);
                                                      break;
                                                }
                                          }//while hook=1
                                    }//ring
                              }//while hook=1
                              //######################## Line In
############################
```

```
                        while(Hook==0)
                        {
                                //###################### Call. Out
############################
                                PutCommand(LCD_CMD_CLR);
                                RY1 = 1;
                                //Goto Get Password Menu
                                DisplayLCD("PASSWORD : ",1,0);
                                DisplayLCD("SAFTY MODE
",2,0);
                                //Wait for key
                                STD = 1;
                                Q4 = 1;
                                Q3 = 1;
                                Q2 = 1;
                                Q1 = 1;

                                TelFlag = 0;

      for(KeyCount=0;KeyCount<=3;KeyCount++)
                                {
                                        if(Hook==1) break;

                                        DTMF_KEY[KeyCount] = 0;
                                        while(STD==0)
                                        {
                                                if(Hook==1) break;
                                        }
                                        if(Hook==1) break;

                                        STD_Flag = 0;
                                        dmsec(500);
                                        bQ4 = Q4;
                                        bQ3 = Q3;
                                        bQ2 = Q2;
                                        bQ1 = Q1;
                                        tmp_KEY[3] = bQ4&0x01;
                                        tmp_KEY[2] = bQ3&0x01;
                                        tmp_KEY[1] = bQ2&0x01;
                                        tmp_KEY[0] = bQ1&0x01;

                                        DTMF_KEY[KeyCount] =
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);

                                        if(DTMF_KEY[KeyCount]==0x0a)
                                        {
                                                PasswdDTMF_KEY[KeyCount]
= DTMF_KEY[KeyCount] + 0x30-10;
                                        //
      DisplayLCD1C(&PasswdDTMF_KEY[KeyCount],1,11+KeyCount);
                                DisplayLCD1C("*",1,11+KeyCount);
                                        }
                                        else
                                        {
                                                PasswdDTMF_KEY[KeyCount]
= DTMF_KEY[KeyCount] +0x30;
//
      DisplayLCD1C(&PasswdDTMF_KEY[KeyCount],1,11+KeyCount);
```

```
                              DisplayLCD1C("*",1,11+KeyCount);
                                    }
                              }
                              Passwd[0] = 0x0b;//*
                              Passwd[1] =
iReadEEPROM(EEPROM_ADDR,0x0053);
                              Passwd[2] =
iReadEEPROM(EEPROM_ADDR,0x0054);
                              Passwd[3] = 0x0c;//#

                              if((DTMF_KEY[0]==Passwd[0])&&
                                    (DTMF_KEY[1]==Passwd[1])&&
                                    (DTMF_KEY[2]==Passwd[2])&&
                                    (DTMF_KEY[3]==Passwd[3]))
                                    {
                              ////Entry Admin Menu
                                          DisplayLCD("1.SAF CLR
2.SET PAS",1,0);
                                          DisplayLCD("3.SET TIM
4.ADM PAS",2,0);
                                    }

////////////////////////////////////////////////////////////////
//////////////

                              //Wait for key
                              STD = 1;
                              Q4 = 1;
                              Q3 = 1;
                              Q2 = 1;
                              Q1 = 1;
                              if(Hook==1) break;//Break Hook
                              DTMF_KEY[0] = 0;
                              while(STD==0)
                              {
                                    if(Hook==1) break;
                              }
                              if(Hook==1) break;//Break Hook

                              STD_Flag = 0;
                              dmsec(500);
                              bQ4 = Q4;
                              bQ3 = Q3;
                              bQ2 = Q2;
                              bQ1 = Q1;
                              tmp_KEY[3] = bQ4&0x01;
                              tmp_KEY[2] = bQ3&0x01;
                              tmp_KEY[1] = bQ2&0x01;
                              tmp_KEY[0] = bQ1&0x01;
                                          DTMF_KEY[0] =
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


                              switch(DTMF_KEY[0])
                              {
                                    case 1:
                                    {
                                          DisplayLCD("     SAFTY
```

```
CLEAR     ",1,0);
                                                DisplayLCD("
",2,0);
                                                WrongCount = 0;
                                                AdminCLR_Flag = 1;
                                                Admin_Clear = 0;
                                                dmsec(10);
                                                Admin_Clear = 1;
                                                dmsec(10);
                                                Admin_Clear = 0;
                                                dmsec(10);
                                                goto main;
                                                break;
                                        }
                                        case 2:
                                        {
                                                DisplayLCD("    SET PASS
No.     ",1,0);
                                                DisplayLCD("SELECT 01-
20 :       ",2,0);
                                                //################## Wait
Key ###################


      for(KeyCount=0;KeyCount<=1;KeyCount++)
                                                {
                                                        STD = 1;
                                                        Q4 = 1;
                                                        Q3 = 1;
                                                        Q2 = 1;
                                                        Q1 = 1;
                                                        if(Hook==1)
      break;//Break Hook
                                                        DTMF_KEY[KeyCount]
= 0;
                                                        while(STD==0)
                                                        {
                                                              if(Hook==1)
      break;
                                                        }
                                                        if(Hook==1)

      break;//Break Hook

                                                        STD_Flag = 0;
                                                        dmsec(500);
                                                        bQ4 = Q4;
                                                        bQ3 = Q3;
                                                        bQ2 = Q2;
                                                        bQ1 = Q1;
                                                        tmp_KEY[3] =
bQ4&0x01;
                                                        tmp_KEY[2] =
bQ3&0x01;
                                                        tmp_KEY[1] =
bQ2&0x01;
                                                        tmp_KEY[0] =
bQ1&0x01;
                                                DTMF_KEY[KeyCount] =
```

```
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,15+KeyCount);
                                              }//Loop Get Key


                                              hr =
Byte2BCD(DTMF_KEY[0],DTMF_KEY[1]);
                                              DisplayLCD("NEW
PASSWORD :      ",1,0);
                                              DisplayLCD("
",2,0);


      for(KeyCount=0;KeyCount<=3;KeyCount++)
                                              {
                                                    STD = 1;
                                                    Q4 = 1;
                                                    Q3 = 1;
                                                    Q2 = 1;
                                                    Q1 = 1;
                                                    if(Hook==1)
      break;//Break Hook
                                                    DTMF_KEY[KeyCount]
= 0;
                                                    while(STD==0)
                                                    {
                                                          if(Hook==1)
      break;
                                                    }
                                                    if(Hook==1)
      break;//Break Hook


                                                    STD_Flag = 0;
                                                    dmsec(500);
                                                    bQ4 = Q4;
                                                    bQ3 = Q3;
                                                    bQ2 = Q2;
                                                    bQ1 = Q1;
                                                    tmp_KEY[3] =
bQ4&0x01;
                                                    tmp_KEY[2] =
bQ3&0x01;
                                                    tmp_KEY[1] =
bQ2&0x01;
                                                    tmp_KEY[0] =
bQ1&0x01;

                                                    DTMF_KEY[KeyCount]
```

```
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;
//
      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],1,15+KeyCount);

      DisplayLCD1C("*",1,15+KeyCount);
                                          }//Loop Get Key

                                          hr = hr-1;//Minus 1 to
decrease offset

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+2,DTMF_KEY[0]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+3,DTMF_KEY[1]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+4,DTMF_KEY[2]);

      iWriteEEPROM(EEPROM_ADDR,(hr*4)+5,DTMF_KEY[3]);
                                          DisplayLCD("
COMPLETE      ",1,0);
                                          DisplayLCD("
",2,0);

                                          dmsec(1000);

                                          break;
                                    }
                                    case 3:
                                    {
                                          DisplayLCD("  SET DATE &
TIME   ",1,0);
                                          DisplayLCD("DATE :
",2,0);
                                          hr1 = 0;


      for(KeyCount=0;KeyCount<=11;KeyCount++)
                                          {

                                                if(KeyCount==2)
                                                {

      DisplayLCD("MON  :   ",2,0);
                                                }
                                                if(KeyCount==4)
                                                {

      DisplayLCD("YEAR :   ",2,0);
                                                }
                                                if(KeyCount==6)
```

```
                                                {

      DisplayLCD("HOUR :    ",2,0);
                                                }
                                                if(KeyCount==8)
                                                {

      DisplayLCD("MIN  :    ",2,0);
                                                }
                                                if(KeyCount==10)
                                                {

      DisplayLCD("SEC  :    ",2,0);
                                                }

                                                STD = 1;
                                                Q4 = 1;
                                                Q3 = 1;
                                                Q2 = 1;
                                                Q1 = 1;
                                                if(Hook==1)
      break;//Break Hook
                                                DTMF_KEY[KeyCount]
= 0;
                                                while(STD==0)
                                                {
                                                      if(Hook==1)
      break;
                                                }
                                                if(Hook==1)
      break;//Break Hook

                                                STD_Flag = 0;
                                                dmsec(300);
                                                bQ4 = Q4;
                                                bQ3 = Q3;
                                                bQ2 = Q2;
                                                bQ1 = Q1;
                                                tmp_KEY[3] =
bQ4&0x01;
                                                tmp_KEY[2] =
bQ3&0x01;
                                                tmp_KEY[1] =
bQ2&0x01;
                                                tmp_KEY[0] =
bQ1&0x01;

                                                DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);

                                                if(KeyCount<2)
                                                {

      if(DTMF_KEY[KeyCount]==0x0a)
                                                      {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                      }
```

```
DisplayDTMF_KEY[KeyCount] =
DTMF_KEY[KeyCount] +0x30;
DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
DisplayLCD("DATE : ",2,0);
                                                }

if((KeyCount>=2)&&(KeyCount<4))
                                                {
DisplayLCD("MON  : ",2,0);
if(DTMF_KEY[KeyCount]==0x0a)
                                                    {
DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }
DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;
DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
                                                }

if((KeyCount>=4)&&(KeyCount<6))
                                                {
DisplayLCD("YEAR : ",2,0);
if(DTMF_KEY[KeyCount]==0x0a)
                                                    {
DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }
DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;
DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                    dmsec(500);
                                                }

if((KeyCount>=6)&&(KeyCount<8))
                                                {
DisplayLCD("HOUR : ",2,0);
if(DTMF_KEY[KeyCount]==0x0a)
                                                    {
DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }
DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;
```

```
                DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                dmsec(500);
                                        }



        if((KeyCount>=8)&&(KeyCount<10))
                                        {

        DisplayLCD("MIN  : ",2,0);

        if(DTMF_KEY[KeyCount]==0x0a)
                                                {

        DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

        DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

        DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                dmsec(500);
                                        }

                                        if(KeyCount>=10)
                                        {

        DisplayLCD("SEC  : ",2,0);

        if(DTMF_KEY[KeyCount]==0x0a)
                                                {

        DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                }

        DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] +0x30;

        DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,hr1+7);
                                                dmsec(500);
                                        }

                                    hr1 = hr1+1;
                                    if(hr1>=2)
                                          hr1 = 0;
                                    }//Loop Get Key


                                    hr =
Byte2BCD(DTMF_KEY[10],DTMF_KEY[11]);
                                    RTC_WRITE(RTC_ADDR,
0,hr);//sec

                                    hr =
Byte2BCD(DTMF_KEY[8],DTMF_KEY[9]);
                                    RTC_WRITE(RTC_ADDR,
1,hr);//sec

                                    hr =
Byte2BCD(DTMF_KEY[6],DTMF_KEY[7]);
```

```
                                        hr = hr&0x3F;//Hr in 24
hr
                                        RTC_WRITE(RTC_ADDR,
2,hr);
                                        hr =
Byte2BCD(DTMF_KEY[4],DTMF_KEY[5]);
                                        RTC_WRITE(RTC_ADDR,
6,hr);//Year
                                        hr =
Byte2BCD(DTMF_KEY[2],DTMF_KEY[3]);
                                        RTC_WRITE(RTC_ADDR,
5,hr);//month
                                        hr =
Byte2BCD(DTMF_KEY[0],DTMF_KEY[1]);
                                        RTC_WRITE(RTC_ADDR,
4,hr);//date
                                        break;
                                    }
                                    case 4:
                                    {
                                        DisplayLCD("   SET ADMIN
PASS   ",1,0);
                                        DisplayLCD("NEW
PASSWORD :      ",2,0);

      for(KeyCount=0;KeyCount<=1;KeyCount++)
                                        {
                                            STD = 1;
                                            Q4 = 1;
                                            Q3 = 1;
                                            Q2 = 1;
                                            Q1 = 1;
                                            if(Hook==1)
      break;//Break Hook
                                            DTMF_KEY[KeyCount]
= 0;
                                            while(STD==0)
                                            {
                                                if(Hook==1)
      break;
                                            }
                                            if(Hook==1)
      break;//Break Hook
                                            STD_Flag = 0;
                                            dmsec(500);
                                            bQ4 = Q4;
                                            bQ3 = Q3;
                                            bQ2 = Q2;
                                            bQ1 = Q1;
                                            tmp_KEY[3] =
bQ4&0x01;
                                            tmp_KEY[2] =
bQ3&0x01;
```

```
                                                    tmp_KEY[1] =
bQ2&0x01;
                                                    tmp_KEY[0] =
bQ1&0x01;

                                                    DTMF_KEY[KeyCount]
= (tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);


      if(DTMF_KEY[KeyCount]==0x0a)
                                                    {

      DTMF_KEY[KeyCount] = DTMF_KEY[KeyCount]-10;
                                                    }

      DisplayDTMF_KEY[KeyCount] = DTMF_KEY[KeyCount] + 0x30;
//
      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,15+KeyCount);

      DisplayLCD1C("*",2,15+KeyCount);
                                                }//Loop Get Key

      iWriteEEPROM(EEPROM_ADDR,0x0053,DTMF_KEY[0]);//Admin Passwd

      iWriteEEPROM(EEPROM_ADDR,0x0054,DTMF_KEY[1]);
                                                DisplayLCD("
COMPLETE      ",1,0);
                                                DisplayLCD("
",2,0);

                                                dmsec(1000);

                                                break;
                                          }
                                    }
//////////////////////////////////////////////////////////////////////
//////////////

                        }//While Hook

                        }//while Admin clr

                  }

                  }

            if(PasswdTRUE==1)
            {
                  WrongCount = 0;//Reset wrong counter
                  RY1 = 0;
                  RY3 = 1;
                  DisplayLCD("TEL. No. : ",2,0);

                  for(KeyCount=0;KeyCount<=8;KeyCount++)
                  {
                        dmsec(50);
                        Hook=1;
                        dmsec(50);
```

```
                        if(Hook==1) break;


                        DTMF_KEY[KeyCount] = 0;
                        STD=1;
                        dmsec(50);
                        while(STD==0)
                        {

                              if(Hook==1) break;
                        }
                        Hook=1;
                        dmsec(50);
                        if(Hook==1) break;
                        SBUF = 0x14;
                        dmsec(50);

                        STD_Flag = 0;
                        bQ4 = Q4;
                        bQ3 = Q3;
                        bQ2 = Q2;
                        bQ1 = Q1;
                        tmp_KEY[3] = bQ4&0x01;
                        tmp_KEY[2] = bQ3&0x01;
                        tmp_KEY[1] = bQ2&0x01;
                        tmp_KEY[0] = bQ1&0x01;

                        dmsec(300);

                        DTMF_KEY[KeyCount] =
(tmp_KEY[3]<<3)|(tmp_KEY[2]<<2)|(tmp_KEY[1]<<1)|(tmp_KEY[0]);
                        if(DTMF_KEY[KeyCount]==0x0a)
                  {
                              DisplayDTMF_KEY[KeyCount] =
DTMF_KEY[KeyCount] + 0x30-10;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,11+KeyCount);
                  }
                  else
                  {
                              DisplayDTMF_KEY[KeyCount] =
DTMF_KEY[KeyCount] +0x30;

      DisplayLCD1C(&DisplayDTMF_KEY[KeyCount],2,11+KeyCount);
                  }

                        dmsec(50);
                  }
                  RY3 = 1;//Connect to receive ringback
                  dmsec(5000);//Hardware Bug for wait ringback

                  RingBack = 1;
                  while(RingBack==0)//Wait while Ring Back
                  {
                        if(Hook==1) break; //Cancel call
                  }
                  if(Hook==0)//Time stamp I
                  {
```

```
                        Time1 = RTC_READ_TIME();
                        DateTime1[0] = RTC_READ(RTC_ADDR,4);
                        DateTime1[1] = RTC_READ(RTC_ADDR,5);
                        DateTime1[2] = RTC_READ(RTC_ADDR,6);
                }
                while(Hook==0)
                {
                        //Record
                        TelFlag = 1;
                        DisplayLCD("     LINE IN USE    ",1,0);
                        FullTime1 = RTC_READ_DATA();

        hr1= FullTime1.date&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,0);
        hr1 = FullTime1.date&0x0f;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,1);

        hr1= FullTime1.month&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,3);
        hr1 = FullTime1.month&0x0f;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,4);

        hr1= FullTime1.year&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,6);
        hr1 = FullTime1.year&0x0f;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,7);

        hr1= FullTime1.sec&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,18);
        hr1 = FullTime1.sec&0x0f;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,19);

        hr1= FullTime1.min&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,15);
        hr1 = FullTime1.min&0x0f;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,16);

        hr1= FullTime1.hour&0xf0;
        hr1 = hr1>>4;
        hr1 = hr1+0x30;
        DisplayLCD1C(&hr1,2,12);
        hr1 = FullTime1.hour&0x0f;
        hr1 = hr1+0x30;
```

```
      DisplayLCD1C(&hr1,2,13);

                  DisplayLCD("    ",2,8);
                  DisplayLCD("/",2,2);
                  DisplayLCD("/",2,5);
                  DisplayLCD(":",2,17);
                  DisplayLCD(":",2,14);                    }
                  if(TelFlag==1)
                  {
                        TelFlag=0;
                        Time2 = RTC_READ_TIME();
                        DateTime2[0] = RTC_READ(RTC_ADDR,4);
                        DateTime2[1] = RTC_READ(RTC_ADDR,5);
                        DateTime2[2] = RTC_READ(RTC_ADDR,6);

                        //Record to rom part I
                        //Write User from PasswdDTMF_KEY,
                        //Tel Number from DisplayDTMF_KEY,
                        //Start time from Time1,
                        //End Time from Time2,
                        //Start Date from DateTime1,
                        //End Date from DateTime2.

                        MemADDRL = iReadEEPROM(EEPROM_ADDR,0x0000);
                        MemADDRH = iReadEEPROM(EEPROM_ADDR,0x0001);

                        MemADDR = 0;
                        MemADDR = MemADDRH;
                        MemADDR = (MemADDR<<8)|MemADDRL;

/*
|   User_Byte0    |   User_Byte1    |   User_Byte2    |
User_Byte3    |
| Tel. No. Byte0  | Tel. No. Byte1  | Tel. No. Byte2  | Tel. No.
Byte3  |
| Tel. No. Byte4  | Tel. No. Byte5  | Tel. No. Byte6  | Tel. No.
Byte7  | Tel. No. Byte8  |
|   Start Date    |   Start Month   |   Start Year    |
|    Start Sec    |    Start Min    |    Start Hr     |
|     End Date    |    End Month    |     End Year    |
|     End Sec     |     End Min     |     End Hr      |
|     Intv Sec    |    Intv Min     |     Intv Hr     | */


                        for(hr=0;hr<=3;hr++)
                        {

      iWriteEEPROM(EEPROM_ADDR,MemADDR,PasswdDTMF_KEY[hr]);
                              MemADDR = MemADDR+1;
                        }
                        for(hr=0;hr<=8;hr++)
                        {

      iWriteEEPROM(EEPROM_ADDR,MemADDR,DisplayDTMF_KEY[hr]);
                              MemADDR = MemADDR+1;
                        }
                        for(hr=0;hr<=2;hr++)
                        {
```

```
        iWriteEEPROM(EEPROM_ADDR,MemADDR,DateTime1[hr]);
                                MemADDR = MemADDR+1;
                        }
        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time1.sec);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time1.min);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time1.hour);
                        MemADDR = MemADDR+1;

                        for(hr=0;hr<=2;hr++)
                        {

        iWriteEEPROM(EEPROM_ADDR,MemADDR,DateTime2[hr]);
                                MemADDR = MemADDR+1;
                        }

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.sec);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.min);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.hour);
                        MemADDR = MemADDR+1;

                        //Cal. Interval
                        Time2 = ElapsedTime(Time1,Time2);

                        //Record to rom part II
                        //Write Elapsed time from Time2

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.sec);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.min);
                        MemADDR = MemADDR+1;

        iWriteEEPROM(EEPROM_ADDR,MemADDR,Time2.hour);
                        MemADDR = MemADDR+1;

                        //Write Current
                        MemADDRH = (MemADDR&0xff00)>>8;
                        MemADDRL = MemADDR&0x00ff;
                        iWriteEEPROM(EEPROM_ADDR,0x0000,MemADDRL);
                        iWriteEEPROM(EEPROM_ADDR,0x0001,MemADDRH);

                        //Clear LCD
                        DisplayLCD("                    ",1,0);
                        DisplayLCD("                    ",2,0);

                }

        }
    }
}
```

โปรแกรมวิชวล เบสิค

```
Dim COMPort As Byte
Dim BundRate As String

Private Sub Combo1_Click()
   Select Case Combo1.ListIndex
      Case 0
         COMPort = 1
      Case 1
         COMPort = 2
      Case 2
         COMPort = 3
      Case 3
         COMPort = 4
      Case 4
         COMPort = 5
      Case 5
         COMPort = 6
      Case 6
         COMPort = 7
      Case 7
         COMPort = 8
      Case Else
         COMPort = 1
   End Select
End Sub

Private Sub Combo2_Click()
   Select Case Combo2.ListIndex
      Case 0
         BundRate = "110"
      Case 1
```

```
        BundRate = "300"
    Case 2
        BundRate = "600"
    Case 3
        BundRate = "1200"
    Case 4
        BundRate = "2400"
    Case 5
        BundRate = "9600"
    Case 6
        BundRate = "14400"
    Case 7
        BundRate = "19200"
    Case 8
        BundRate = "28800"
    Case 9
        BundRate = "38400"
    Case 10
        BundRate = "56000"
    Case 11
        BundRate = "128000"
    Case 12
        BundRate = "256000"
    Case Else
        BundRate = "9600"
  End Select
  Command1.Enabled = True
  Command2.Enabled = True
End Sub

Private Sub Command1_Click()
 If Not MSComm1.PortOpen Then
    MSComm1.CommPort = COMPort
    MSComm1.Settings = CStr(BundRate) + ",n,8,1"
```

```
",n,8,1"
        MSComm1.PortOpen = True
        Command5.Enabled = True
        'MSComm1.Output = "E"
    Else
      MsgBox ("COM Port ÁÕ¡ÒÃãªé§Ò¹ÍÂÙè¡èÍ¹áÅéÇ ¡ÃØ³ÒàÅ×Í¡ COM Port ãËÁè"), , "äÁèÊÒÁÒÃ
¶àª×èÍÁµèÍä´é"
    End If
End Sub

Private Sub Command2_Click()
    MSComm1.PortOpen = False
    StatusBar1.Panels(1).Text = "COM Port Disconnect..."
    Command1.Enabled = False
    Command2.Enabled = False
    Command5.Enabled = False
End Sub

Private Sub Command3_Click()
    CRPT1.DataFiles(0) = App.Path & "\DBTEL.mdb"
    CRPT1.ReportFileName = App.Path & "\DBTel.rpt"
    CRPT1.Destination = 0
    CRPT1.Action = 1
End Sub

Private Sub Command4_Click()
    Unload Me
End Sub

Private Sub Command5_Click()
    MSComm1.Output = Char(161)
End Sub

Private Sub Form_Load()
```

```
Private Sub Form_Load()
    Data1.DatabaseName = App.Path & "\DBTel.mdb"
    Data1.RecordSource = "DBTel"

End Sub
```

ภาคผนวก จ
คู่มือการใช้งาน

คู่มือการใช้งาน
เครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

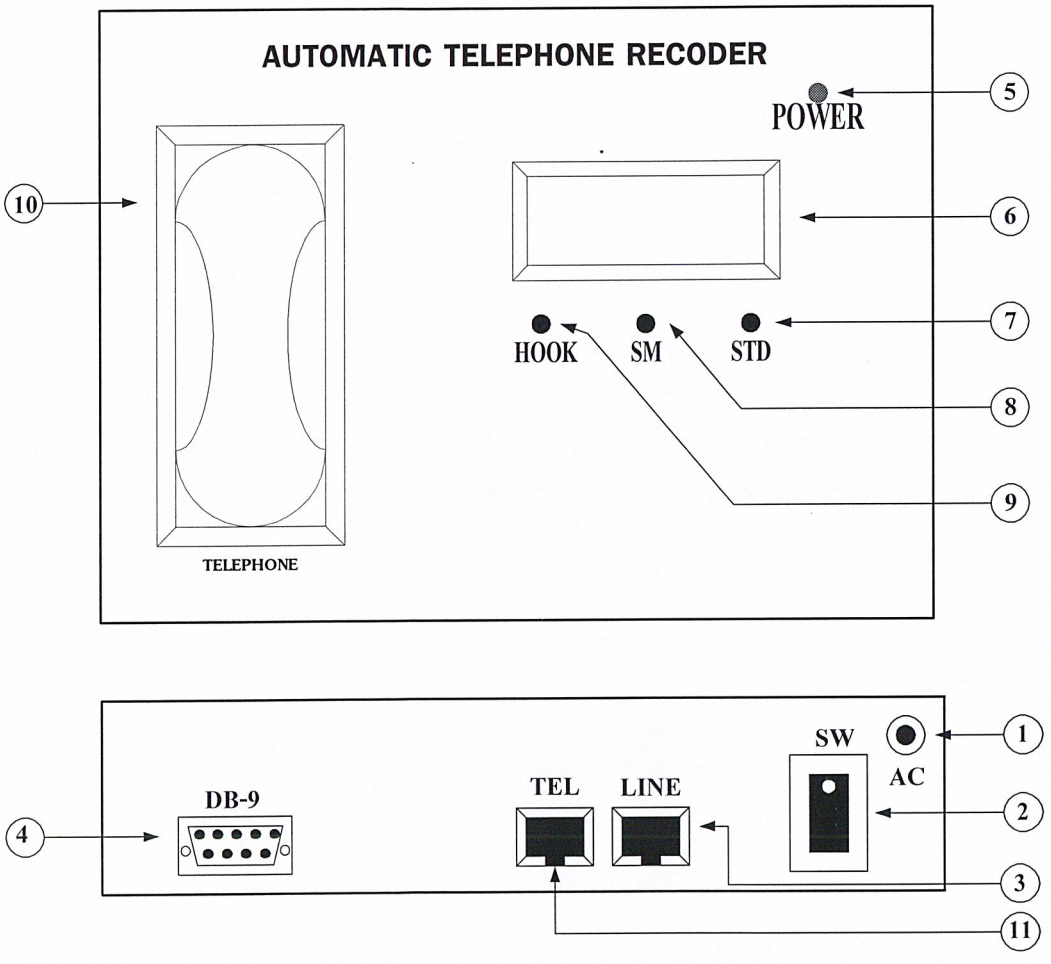
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

1. คำแนะนำเบื้องต้น

ก่อนที่จะใช้งานเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ ควรทำการศึกษาถึงขั้นตอนการใช้งานของเครื่องจากคู่มือหรือการใช้งานเบื้องต้นให้เข้าใจ เพื่อการใช้งานที่ถูกต้อง และเพื่อให้ระบบการทำงานเป็นไปตามกระบวนการได้อย่างถูกต้องมากที่สุด

2. ส่วนประกอบและปุ่มควบคุม



รูปที่ จ.1 ส่วนประกอบและปุ่มควบคุมของเครื่องบันทึกข้อมูลการใช้งานโทรศัพท์อัตโนมัติ

จากรูปที่ จ.1 มีรายละเอียดต่างๆ ดังนี้

① ช่องต่อสายไฟ AC

② สวิตช์เปิด-ปิดเครื่อง

③ ช่องต่อคู่สายโทรศัพท์เข้ากับเครื่อง

④ พอร์ตเชื่อมต่อข้อมูลไปยังคอมพิวเตอร์

⑤ หลอดแอลซีดีแสดงการทำงานเมื่อเปิดเครื่อง

⑥ หลอดแอลซีดีแสดงการกดปุ่มหมายเลขโทรศัพท์

⑦ หลอดแอลซีดีแสดงโหมดการป้องกันการโทรออก

⑧ หลอดแอลซีดีแสดงสภาวะการยกหูโทรศัพท์

⑨ จอแสดงผลแบบผลึกเหลว

⑩ เครื่องโทรศัพท์

⑪ ช่องต่อคู่สายโทรศัพท์เพื่อต่อพ่วง

3. ขั้นตอนการใช้งานเครื่องบันทึกข้อมูลการใช้โทรศัพท์อัตโนมัติ

3.1) เสียบปลั๊กไฟฟ้ากระแสสลับแรงดันไฟฟ้า 220 โวลต์ 50 เฮิรตซ์

3.2) เปิดสวิตช์แหล่งจ่ายไฟของเครื่อง

3.3) ไดโอดแปล่งแสงแสดงว่าเครื่องอยู่ในสภาวะพร้อมทำงาน

3.4) กรณีการโทรออกต้องทำตามขั้นตอนดังนี้

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL 25/01/04 21:35:23

2. เมื่อมีการยกหูโทรศัพท์ จอแสดงผล LCD จะแสดงดังนี้

PASSWORD :

3.เมื่อมีการใส่รหัสผ่านของผู้ใช้จำนวน 4 หลัก จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : ****

4. เมื่อใส่รหัสถูกต้อง จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : **** TEL. No. :

5. หลังจากนั้นจึงกดเลขหมายโทรศัพท์ที่ต้องการโทรออก จอแสดงผล LCD จะแสดงดังนี้

PASSWORD : ****
TEL. No. : 067151826

6. หลังจากกดเลขหมายโทรศัพท์เสร็จและมีคนรับสาย จอแสดงผล LCD จะแสดงดังนี้

LINE IN USE
25/01/04 21:36:01

7.เมื่อสิ้นสุดการสนทนาให้วางหูโทรศัพท์ จอแสดงผล LCD จะแสดงดังนี้

KMITL
25/01/04 21:40:21

8. ถ้าหากมีการกดรหัสผ่านผิด 3 ครั้ง เครื่องจะเข้าสู่โหมดการป้องกันการโทรออกชั่วคราว เป็นเวลา 20 นาที หรือจนกว่าจะได้รับการแก้ไขจากผู้ดูแลระบบ เครื่องก็จะเข้าสู่การทำงานปกติอีกครั้ง

SAFETY MODE

3.5) กรณีการโทรเข้า

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL
25/01/04 21:35:23

2.เมื่อมีสัญญาณกระดิ่งเรียกเข้า จอแสดงผล LCD จะแสดงดังนี้

LINE IN
WAIT FOR RECEIVING

3.เมื่อมีการรับสาย จอแสดงผล LCD จะแสดงดังนี้

LINE IN
RECEIVING

4.เมื่อสิ้นสุดการสนทนา จอแสดงผล LCD จะแสดงดังนี้

KMITL
25/01/04 21:36:02

3.6) กรณีการโทรศัพท์ออกหมายเลขพิเศษ ต้องทำตามขั้นตอนดังนี้

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL
25/01/04 21:35:23

2.เมื่อมีการยกหู ให้กดเครื่องหมาย #

PASSWORD : #

3.กดหมายเลขพิเศษตามที่ผู้ดูแลระบบเซ็ตไว้

PASSWORD : #
TEL. No. : 191

4.เมื่อกดหมายเลขเสร็จแล้ว เครื่องทำการต่อเข้ากับคู่สายองค์การโทรศัพท์

LINE IN USE
25/01/04 21:36:05

5.เมื่อสิ้นสุดการสนทนา

KMITL
25/01/04 21:36:45

3.7) กรณีของผู้ดูแลระบบ

1.แสดงสภาวะปกติเมื่อไม่มีการยกหู

KMITL
25/01/04 21:35:23

2.เมื่อมีการยกหู

PASSWORD :

3.ผู้ดูแลระบบป้อนรหัสส่วนตัว 4 หลัก โดยรหัสจะกดเครื่องหมาย * ตามด้วยตัวเลขสองหลักและหลักสุดท้ายกดเครื่องหมาย # (รหัสผู้ดูแลระบบที่ตั้งไว้ คือ *11#)

PASSWORD : ****

4. เมื่อกดรหัสถูกต้องแล้ว จอแสดงผล LCD จะแสดงเมนูขึ้นมา เพื่อที่ผู้ดูแลระบบจะทำการเซ็ตระบบต่างๆ

1.SAF CLR	2.SET PAS
3.SET TIM	4.ADM PAS

เมื่อกด 1 จะทำการเคลียร์ระบบ SAFETY MODE เพื่อที่จะสามารถใช้งานได้ตามปกติ

เมื่อกด 2 เพื่อที่จะเปลี่ยนรหัสผ่านของผู้ใช้ จอแสดงผล LCD จะแสดงดังนี้

SET PAS No. SELECT 01-20 :

เมื่อต้องการเปลี่ยนรหัสผ่านผู้ใช้คนใด ก็ให้กดเลขหมายโดยกดเลขหมายสองหลัก เช่นถ้าต้องการเปลี่ยนรหัสผ่านผู้ใช้คนที่ 4 ก็ให้กด หมายเลข 04

SET PAS No. SELECT 01-20 : 04

เมื่อกดหมายเลขผู้ใช้เสร็จแล้ว ให้กดรหัสผ่านใหม่ จอแสดงผล LCD จะแสดงดังนี้

NEW PASSWORD : ****

เมื่อกดเปลี่ยนรหัสผ่านเสร็จแล้ว จอแสดงผล LCD จะแสดงดังนี้

COMPLETED

เมื่อกด 3 จะแสดงถึงการตั้งวัน เดือน ปีและเวลา โดยจะเริ่มตั้งวันที่ จอแสดงผล LCD จะแสดงดังนี้

SET DATE & TIME DATE :

เมื่อกดหมายเลข เพื่อเปลี่ยนวันที่เสร็จแล้ว จอแสดงผล LCD จะแสดงให้เปลี่ยนเดือนโดยอัตโนมัติ ซึ่งจะเรียงตั้งแต่ วันที่ เดือน ปี ชั่วโมง นาทีและวินาทีตามลำดับ จอแสดงผล LCD จะแสดงดังนี้

SET DATE & TIME MON :

SET DATE & TIME YEAR :

SET DATE & TIME HOUR :

SET DATE & TIME MIN :

SET DATE & TIME SEC :

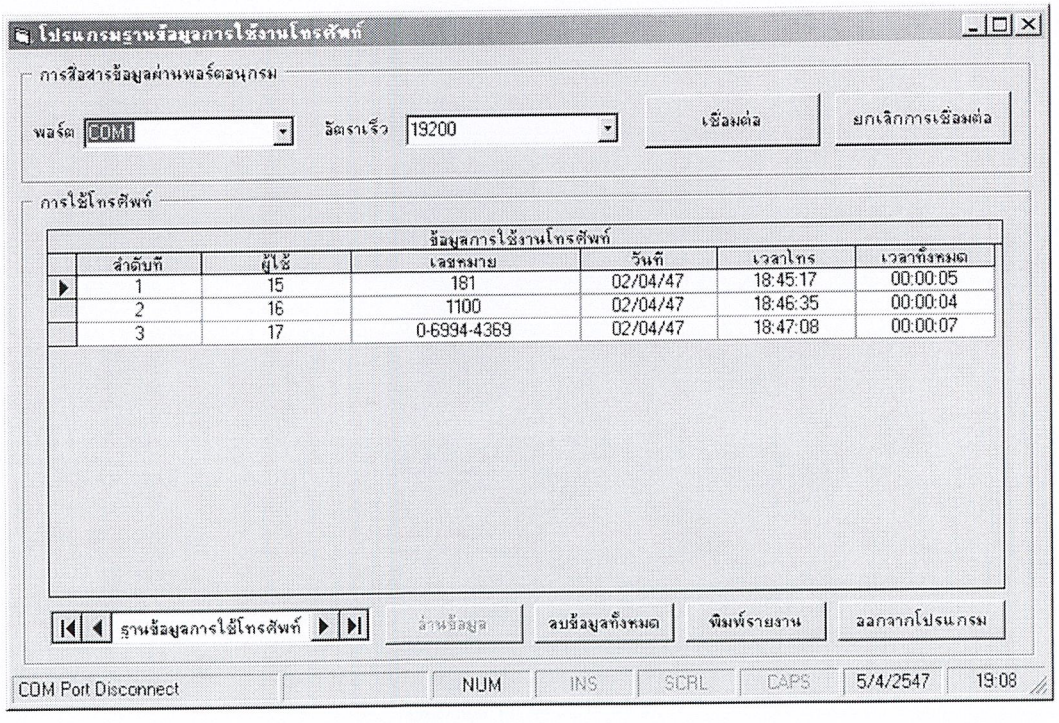
เมื่อกด 4 จะเป็นการเข้าไปเซ็ตระบบของ ผู้ดูแลระบบ จอแสดงผล LCD จะแสดงดังนี้

SET ADMIN PASS NEW PASSWORD :

ขั้นตอนของจอแสดงผล LCD มีการเชื่อมต่อข้อมูลขึ้นสู่คอมพิวเตอร์ เพื่อที่จะนำข้อมูลไปแสดงผลที่คอมพิวเตอร์ จอแสดงผล LCD จะแสดงดังนี้

UPLOADING PLEASE WAIT

3.8) รูปแบบโปรแกรมในขณะใช้งาน



รูปที่ จ.2 รูปแบบโปรแกรมในขณะใช้งาน

4. การแก้ปัญหาเบื้องต้น

เมื่อท่านประสบปัญหาในการใช้งาน สามารถตรวจสอบแนวทางแก้ไขปัญหาเบื้องต้นได้จากตารางข้างล่างนี้

อาการ	สาเหตุและ/หรือวิธีแก้ไข
เครื่องไม่ทำงาน	ตรวจสอบแหล่งจ่ายไฟ สวิตช์เปิด-ปิดเครื่องและการต่อคู่สายโทรศัพท์เข้ากับเครื่อง
เครื่องโทรออกไม่ได้	เครื่องทำงานอยู่ในโหมดการป้องกันการโทรออกชั่วคราว ต้องให้ผู้ดูแลทำการแก้ไขระบบหรือรอจนครบเวลา 20 นาที
วันที่และเวลาไม่เป็นปัจจุบัน	ให้ผู้ดูแลระบบมาทำการปรับตั้งเวลาใหม่

5. การดูแลรักษาและข้อควรปฏิบัติ

5.1 การดูแลรักษา

-ไม่ควรเคลื่อนย้ายบ่อยจนเกินไปเพื่อป้องกันการกระแทก

-ไม่ควรถอดฝาหลังของเครื่องออกเพราะอาจจะทำให้เครื่องได้รับความเสียหายได้

-ระวังอย่าให้เครื่องโดนละอองน้ำหรือเปียกชื้น

5.2 ข้อควรระวัง

-ใช้กับไฟฟ้ากระแสสลับที่มีแรงดัน 220 โวลต์ 50 เฮิรตซ์

-ในการเคลื่อนย้ายในแต่ละครั้งควรปิดสวิตช์และถอดปลั๊กออกก่อน

6. ข้อมูลจำเพาะ

คุณสมบัติ	รายละเอียด
จำนวนผู้ใช้	20 คน 20 รหัสผ่าน
รหัสผ่าน	หมายเลขจำนวน 4 หลัก
ส่วนแสดงผล	แอลซีดี 20 ตัวอักษร 2 บรรทัด
แหล่งจ่ายพลังงาน	ไฟฟ้ากระแสสลับ 220 โวลต์ ความถี่ 50-60 เฮริตซ์

ภาคผนวก ฉ
รายละเอียดและคุณสมบัติของอุปกรณ์

19-0175; Rev 3; 5/96

MAXIM

±15kV ESD-Protected, +5V RS-232 Transceivers

General Description

The MAX202E–MAX213E, MAX232E/MAX241E line drivers/receivers are designed for RS-232 and V.28 communications in harsh environments. Each transmitter output and receiver input is protected against ±15kV electrostatic discharge (ESD) shocks, without latchup. The various combinations of features are outlined in the *Selection Guide*. The drivers and receivers for all ten devices meet all EIA/TIA-232E and CCITT V.28 specifications at data rates up to 120kbps, when loaded in accordance with the EIA/TIA-232E specification.

The MAX211E/MAX213E/MAX241E are available in 28-pin SO packages, as well as a 28-pin SSOP that uses 60% less board space. The MAX202E/MAX232E come in 16-pin narrow SO, wide SO, and DIP packages. The MAX203E comes in a 20-pin DIP/SO package, and needs no external charge-pump capacitors. The MAX205E comes in a 24-pin wide DIP package, and also eliminates external charge-pump capacitors. The MAX206E/MAX207E/MAX208E come in 24-pin SO, SSOP, and narrow DIP packages. The MAX232E/MAX241E operate with four 1µF capacitors, while the MAX202E/MAX206E/MAX207E/MAX208E/MAX211E/MAX213E operate with four 0.1µF capacitors, further reducing cost and board space.

Applications

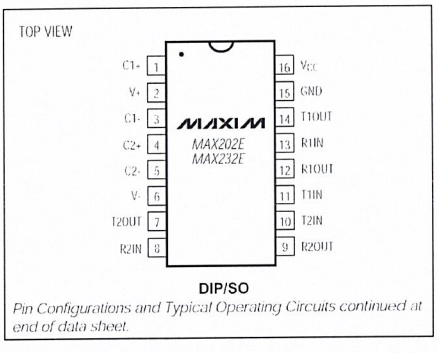
Notebook, Subnotebook, and Palmtop Computers
Battery-Powered Equipment
Hand-Held Equipment

Ordering Information appears at end of data sheet.

Features

- **ESD Protection for RS-232 I/O Pins:**
  ±15kV—Human Body Model
  ±8kV—IEC1000-4-2, Contact Discharge
  ±15kV—IEC1000-4-2, Air-Gap Discharge
- **Latchup Free (unlike bipolar equivalents)**
- **Guaranteed 120kbps Data Rate—LapLink™ Compatible**
- **Guaranteed 3V/µs Min Slew Rate**
- **Operate from a Single +5V Power Supply**

Pin Configurations



DIP/SO

Pin Configurations and Typical Operating Circuits continued at end of data sheet.

Selection Guide

PART	No. of RS-232 DRIVERS	No. of RS-232 RECEIVERS	RECEIVERS ACTIVE IN SHUTDOWN	No. of EXTERNAL CAPACITORS	LOW-POWER SHUTDOWN	TTL THREE-STATE
MAX202E	2	2	0	4 (0.1µF)	No	No
MAX203E	2	2	0	None	No	No
MAX205E	5	5	0	None	Yes	Yes
MAX206E	4	3	0	4 (0.1µF)	Yes	Yes
MAX207E	5	3	0	4 (0.1µF)	No	No
MAX208E	4	4	0	4 (0.1µF)	No	No
MAX211E	4	5	0	4 (0.1µF)	Yes	Yes
MAX213E	4	5	2	4 (0.1µF)	Yes	Yes
MAX232E	2	2	0	4 (1µF)	No	No
MAX241E	4	5	0	4 (1µF)	Yes	Yes

LapLink is a registered trademark of Traveling Software, Inc.

For free samples & the latest literature: http://www.maxim-ic.com, or phone 1-800-998-8800

±15kV ESD-Protected, +5V RS-232 Transceivers

ABSOLUTE MAXIMUM RATINGS

V_{CC} ..... -0.3V to +6V
V+ ..... (V_{CC} - 0.3V) to +14V
V- ..... -14V to +0.3V
Input Voltages
T_IN ..... -0.3V to (V+ + 0.3V)
R_IN ..... ±30V
Output Voltages
T_OUT ..... (V- - 0.3V) to (V+ + 0.3V)
R_OUT ..... -0.3V to (V_{CC} + 0.3V)
Short-Circuit Duration, T_OUT ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) .. 842mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C) .. 889mW
20-Pin SO (derate 10.00mW/°C above +70°C) 800mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Wide Plastic DIP
(derate 14.29mW/°C above +70°C) 1.14W
24-Pin SO (derate 11.76mW/°C above +70°C) 941mW
24-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
28-Pin SO (derate 12.50mW/°C above +70°C) 1W
28-Pin SSOP (derate 9.52mW/°C above +70°C) 762mW
Operating Temperature Ranges
MAX2__EC__ ..... 0°C to +70°C
MAX2__EE__ ..... -40°C to +85°C
Storage Temperature Range ..... -65°C to +165°C
Lead Temperature (soldering, 10sec) ..... +300°C

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

ELECTRICAL CHARACTERISTICS

(V_{CC} = +5V ±10% for MAX202E/206E/208E/211E/213E/232E/241E; V_{CC} = +5V ±5% for MAX203E/205E/207E. C1–C4 = 0.1µF for MAX202E/206E/207E/208E/211E/213E; C1–C4 = 1µF for MAX232E/241E. T_A = T_{MIN} to T_{MAX}; unless otherwise noted. Typical values are at T_A = +25°C.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
DC CHARACTERISTICS							
V_{CC} Supply Current	I_{CC}	No load, T_A = +25°C	MAX202E/203E		8	15	mA
			MAX205E–208E		11	20	
			MAX211E/213E		14	20	
			MAX232E		5	10	
			MAX241E		7	15	
Shutdown Supply Current		T_A = +25°C, Figure 1	MAX205E/206E		1	10	µA
			MAX211E/241E		1	10	
			MAX213E		15	50	
LOGIC							
Input Pull-Up Current		T_IN = 0V (MAX205E–208E/211E/213E/241E)			15	200	µA
Input Leakage Current		T_IN = 0V to V_{CC} (MAX202E/203E/232E)				±10	µA
Input Threshold Low	V_{IL}	T_IN; EN, $\overline{SHDN}$ (MAX213E) or $\overline{EN}$, SHDN (MAX205E–208E/211E/241E)				0.8	V
Input Threshold High	V_{IH}	T_IN		2.0			V
		EN, $\overline{SHDN}$ (MAX213E) or $\overline{EN}$, SHDN (MAX205E–208E/211E/241E)		2.4			
Output Voltage Low	V_{OL}	R_OUT; I_{OUT} = 3.2mA (MAX202E/203E/232E) or I_{OUT} = 1.6mA (MAX205E/208E/211E/213E/241E)				0.4	V
Output Voltage High	V_{OH}	R_OUT; I_{OUT} = -1.0mA		3.5	V_{CC} - 0.4		V
Output Leakage Current		$\overline{EN}$ = V_{CC}, EN = 0V, 0V ≤ R_{OUT} ≤ V_{CC}, MAX205E–208E/211E/213E/241E outputs disabled			±0.05	±10	µA

±15kV ESD-Protected, +5V RS-232 Transceivers

ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = +5V ±10% for MAX202E/206E/208E/211E/213E/232E/241E; V_{CC} = +5V ±5% for MAX203E/205E/207E; C1–C4 = 0.1µF for MAX202E/206E/207E/208E/211E/213E; C1–C4 = 1µF for MAX232E/241E; T_A = T_{MIN} to T_{MAX}; unless otherwise noted. Typical values are at T_A = +25°C.)

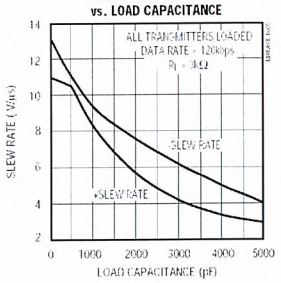
PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
EIA/TIA-232E RECEIVER INPUTS							
Input Voltage Range				-30		30	V
Input Threshold Low		T_A = +25°C, V_{CC} = 5V	All parts, normal operation	0.8	1.2		V
			MAX213E, $\overline{SHDN}$ = 0V, EN = V_{CC}	0.6	1.5		
Input Threshold High		T_A = +25°C, V_{CC} = 5V	All parts, normal operation		1.7	2.4	V
			MAX213E (R4, R5), $\overline{SHDN}$ = 0V, EN = V_{CC}		1.5	2.4	
Input Hysteresis		V_{CC} = 5V, no hysteresis in shutdown		0.2	0.5	1.0	V
Input Resistance		T_A = +25°C, V_{CC} = 5V		3	5	7	kΩ
EIA/TIA-232E TRANSMITTER OUTPUTS							
Output Voltage Swing		All drivers loaded with 3kΩ to ground (Note 1)		±5	±9		V
Output Resistance		V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300			Ω
Output Short-Circuit Current					±10	±60	mA
TIMING CHARACTERISTICS							
Maximum Data Rate		R_L = 3kΩ to 7kΩ, C_L = 50pF to 1000pF, one transmitter switching		120			kbps
Receiver Propagation Delay	t_{PLHR}, t_{PHLR}	C_L = 150pF	All parts, normal operation		0.5	10	µs
			MAX213E (R4, R5), $\overline{SHDN}$ = 0V, EN = V_{CC}		4	40	
Receiver Output Enable Time		MAX205E/206E/211E/213E/241E normal operation, Figure 2			600		ns
Receiver Output Disable Time		MAX205E/206E/211E/213E/241E normal operation, Figure 2			200		ns
Transmitter Propagation Delay	t_{PLHT}, t_{PHLT}	R_L = 3kΩ, C_L = 2500pF, all transmitters loaded			2		µs
Transition-Region Slew Rate		T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 1000pF, measured from -3V to +3V or +3V to -3V, Figure 3		3	6	30	V/µs
ESD PERFORMANCE: TRANSMITTER OUTPUTS, RECEIVER INPUTS							
ESD-Protection Voltage		Human Body Model			±15		kV
		IEC1000-4-2, Contact Discharge			±8		
		IEC1000-4-2, Air-Gap Discharge			±15		

Note 1: MAX211EE_ _ tested with V_{CC} = +5V ±5%.

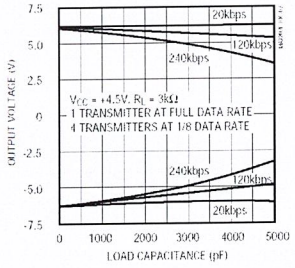
Typical Operating Characteristics (continued)

(Typical Operating Circuits, V_{CC} = +5V, T_A = +25°C, unless otherwise noted.)

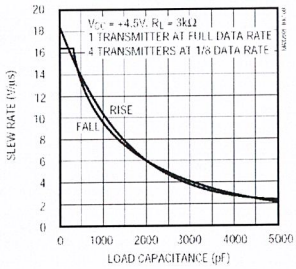
MAX202E/MAX203E/MAX232E
TRANSMITTER SLEW RATE
vs. LOAD CAPACITANCE



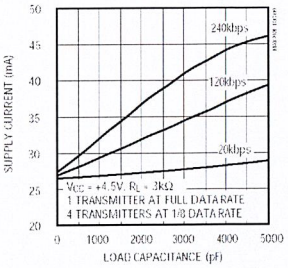
MAX205E–MAX208E
TRANSMITTER OUTPUT VOLTAGE
vs. LOAD CAPACITANCE



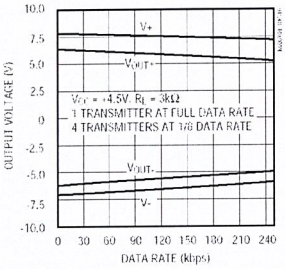
MAX205E–MAX208E
TRANSMITTER SLEW RATE
vs. LOAD CAPACITANCE



MAX205E–MAX208E
SUPPLY CURRENT
vs. LOAD CAPACITANCE



MAX205E–MAX208E
OUTPUT VOLTAGE vs. DATA RATE



Pin Descriptions (continued)

MAX211E/MAX213E/MAX241E

PIN	NAME	FUNCTION
1, 2, 3, 28	T_OUT	RS-232 Driver Outputs
4, 9, 18, 23, 27	R_IN	RS-232 Receiver Inputs
5, 8, 19, 22, 26	R_OUT	TTL/CMOS Receiver Outputs. For the MAX213E, receivers R4 and R5 are active in shutdown mode when EN = 1. For the MAX211E and MAX241E, all receivers are inactive in shutdown.
6, 7, 20, 21	T_IN	TTL/CMOS Driver Inputs. Only the MAX211E, MAX213E, and MAX241E have internal pull-ups to V_{CC}.
10	GND	Ground
11	V_{CC}	+4.5V to +5.5V Supply Voltage
12, 14	C1+, C1-	Terminals for positive charge-pump capacitor
13	V+	$+2V_{CC}$ voltage generated by the charge pump
15, 16	C2+, C2-	Terminals for negative charge-pump capacitor
17	V-	$-2V_{CC}$ voltage generated by the charge pump
24	$\overline{EN}$	Receiver Enable—active low (MAX211E, MAX241E)
	EN	Receiver Enable—active high (MAX213E)
25	SHDN	Shutdown Control—active high (MAX211E, MAX241E)
	$\overline{SHDN}$	Shutdown Control—active low (MAX213E)

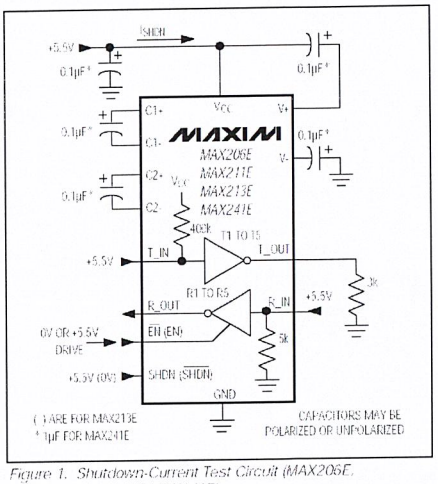


Figure 1. Shutdown-Current Test Circuit (MAX206E, MAX211E/MAX213E/MAX241E)

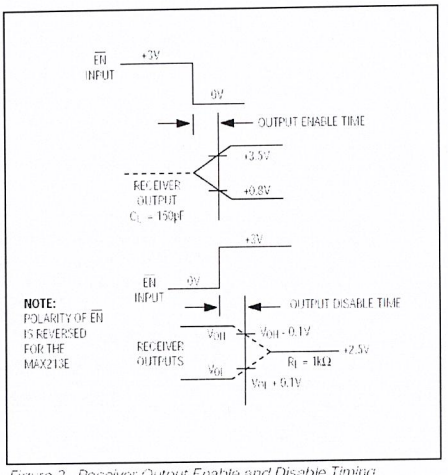


Figure 2. Receiver Output Enable and Disable Timing (MAX205E/MAX206E/MAX211E/MAX213E/MAX241E)

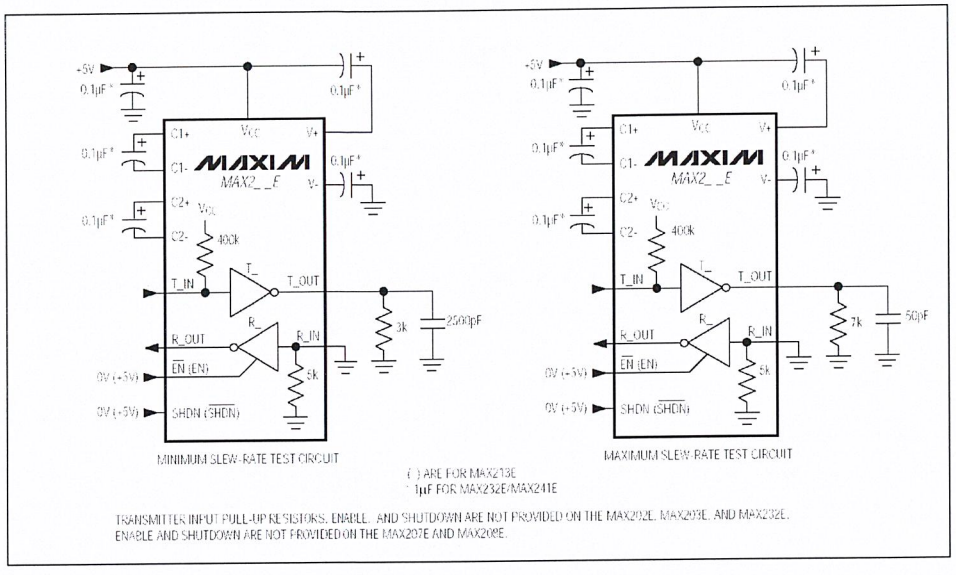


Figure 3. Transition Slew-Rate Circuit

Detailed Description

The MAX202E–MAX213E, MAX232E/MAX241E consist of three sections: charge-pump voltage converters, drivers (transmitters), and receivers. These E versions provide extra protection against ESD. They survive ±15kV discharges to the RS-232 inputs and outputs, tested using the Human Body Model. When tested according to IEC1000-4-2, they survive ±8kV contact-discharges and ±15kV air-gap discharges. The rugged E versions are intended for use in harsh environments or applications where the RS-232 connection is frequently changed (such as notebook computers). The standard (non-"E") MAX202, MAX203, MAX205–MAX208, MAX211, MAX213, MAX232, and MAX241 are recommended for applications where cost is critical.

+5V to ±10V Dual Charge-Pump Voltage Converter

The +5V to ±10V conversion is performed by dual charge-pump voltage converters (Figure 4). The first charge-pump converter uses capacitor C1 to double the +5V into +10V, storing the +10V on the output filter capacitor, C3. The second uses C2 to invert the +10V into -10V, storing the -10V on the V- output filter capacitor, C4.

In shutdown mode, V+ is internally connected to V_{CC} by a 1kΩ pull-down resistor, and V- is internally connected to ground by a 1kΩ pull-up resistor.

RS-232 Drivers

With V_{CC} = 5V, the typical driver output voltage swing is ±8V when loaded with a nominal 5kΩ RS-232 receiver. The output swing is guaranteed to meet EIA/TIA-232E and V.28 specifications that call for ±5V minimum output levels under worst-case conditions. These include a 3kΩ load, minimum V_{CC}, and maximum operating temperature. The open-circuit output voltage swings from (V+ - 0.6V) to V-.

Input thresholds are CMOS/TTL compatible. The unused drivers' inputs on the MAX205E–MAX208E, MAX211E, MAX213E, and MAX241E can be left unconnected because 400kΩ pull-up resistors to V_{CC} are included on-chip. Since all drivers invert, the pull-up resistors force the unused drivers' outputs low. The MAX202E, MAX203E, and MAX232E do not have pull-up resistors on the transmitter inputs.

±15kV ESD-Protected, +5V RS-232 Transceivers

When in low-power shutdown mode, the MAX205E/MAX206E/MAX211E/MAX213E/MAX241E driver outputs are turned off and draw only leakage currents—even if they are back-driven with voltages between 0V and 12V. Below -0.5V in shutdown, the transmitter output is diode-clamped to ground with a 1kΩ series impedance.

RS-232 Receivers

The receivers convert the RS-232 signals to CMOS-logic output levels. The guaranteed 0.8V and 2.4V receiver input thresholds are significantly tighter than the ±3V thresholds required by the EIA/TIA-232E specification. This allows the receiver inputs to respond to TTL/CMOS-logic levels, as well as RS-232 levels.

The guaranteed 0.8V input low threshold ensures that receivers shorted to ground have a logic 1 output. The 5kΩ input resistance to ground ensures that a receiver with its input left open will also have a logic 1 output.

Receiver inputs have approximately 0.5V hysteresis. This provides clean output transitions, even with slow rise/fall-time signals with moderate amounts of noise and ringing.

In shutdown, the MAX213E's R4 and R5 receivers have no hysteresis.

Shutdown and Enable Control (MAX205E/MAX206E/MAX211E/MAX213E/MAX241E)

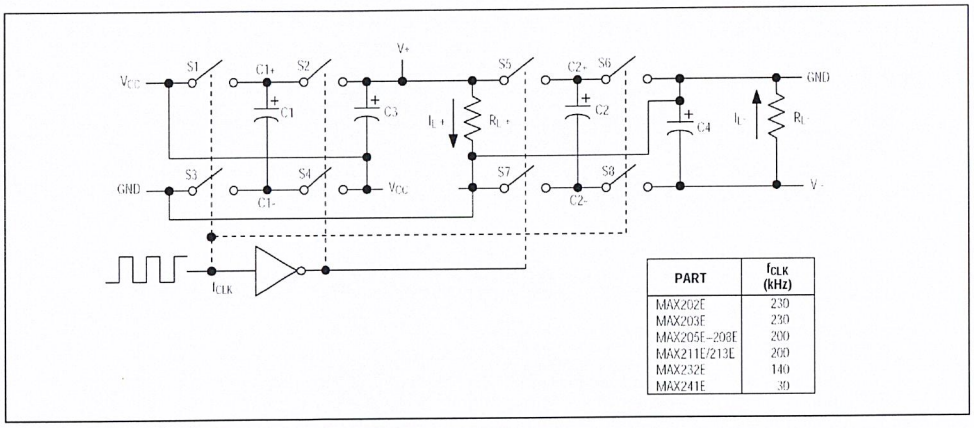
In shutdown mode, the charge pumps are turned off, V+ is pulled down to VCC, V- is pulled to ground, and the transmitter outputs are disabled. This reduces supply current typically to 1µA (15µA for the MAX213E). The time required to exit shutdown is under 1ms, as shown in Figure 5.

Receivers

All MAX213E receivers, except R4 and R5, are put into a high-impedance state in shutdown mode (see Tables 1a and 1b). The MAX213E's R4 and R5 receivers still function in shutdown mode. These two awake-in-shutdown receivers can monitor external activity while maintaining minimal power consumption.

The enable control is used to put the receiver outputs into a high-impedance state, to allow wire-OR connection of two EIA/TIA-232E ports (or ports of different types) at the UART. It has no effect on the RS-232 drivers or the charge pumps.

Note: The enable control pin is active low for the MAX211E/MAX241E ($\overline{EN}$), but is active high for the MAX213E (EN). The shutdown control pin is active high for the MAX205E/MAX206E/MAX211E/MAX241E (SHDN), but is active low for the MAX213E ($\overline{SHDN}$).



PART	f_{CLK} (kHz)
MAX202E	230
MAX203E	230
MAX205E–208E	200
MAX211E/213E	200
MAX232E	140
MAX241E	30

Figure 4. Charge-Pump Diagram

INTEGRATED CIRCUITS

DATA SHEET

P89C51RB2Hxx
P89C51RC2Hxx
P89C51RD2Hxx

80C51 8-bit Flash microcontroller family

16KB/32KB/64KB ISP/IAP Flash with 512B/512B/1KB RAM

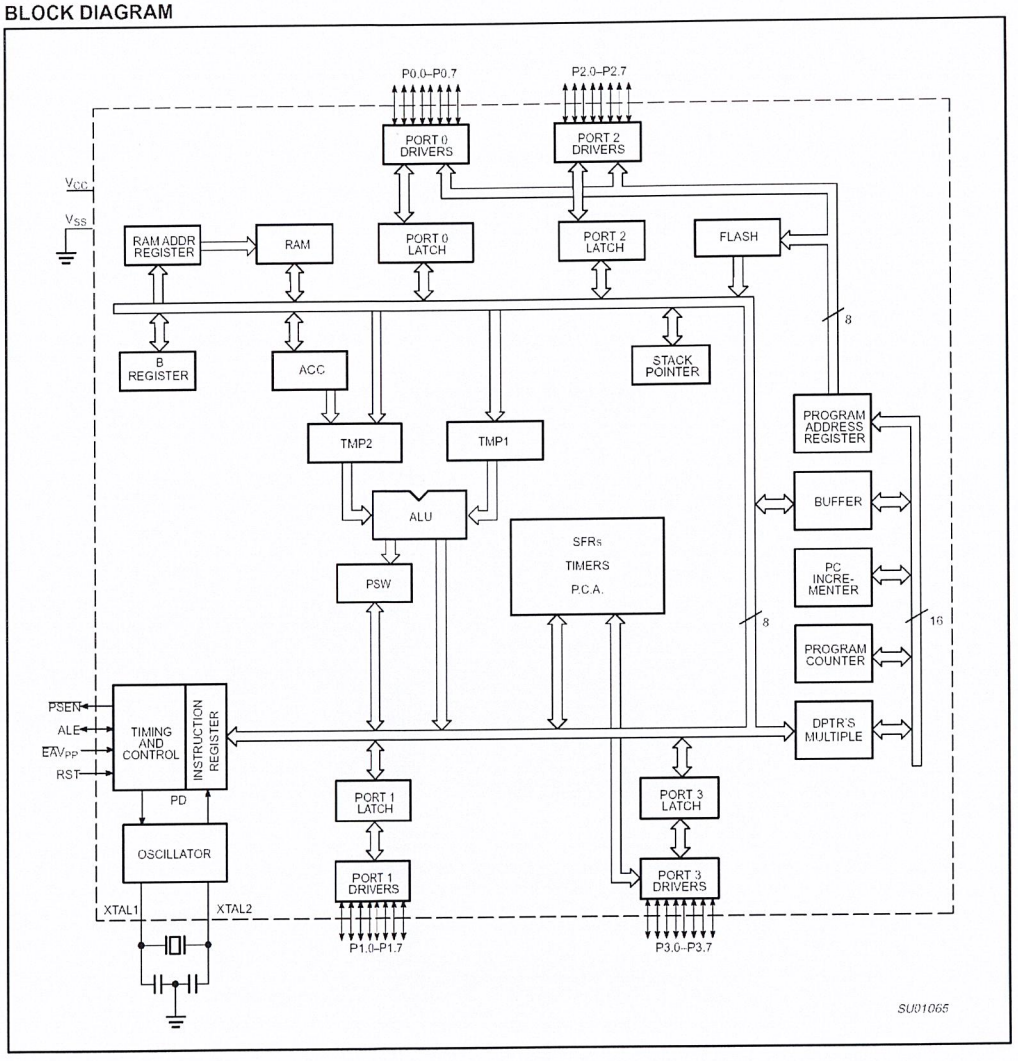
Product data
Supersedes data of 2001 Jun 27

2002 May 24

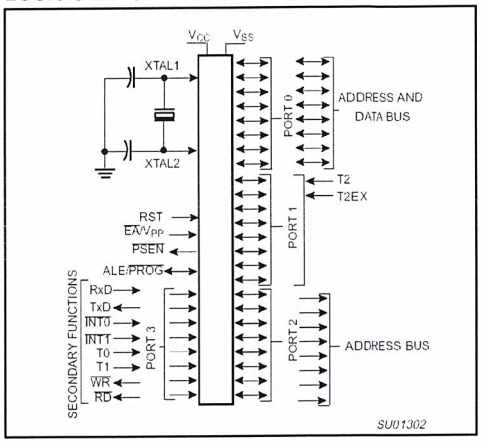
Philips
Semiconductors

PHILIPS

BLOCK DIAGRAM

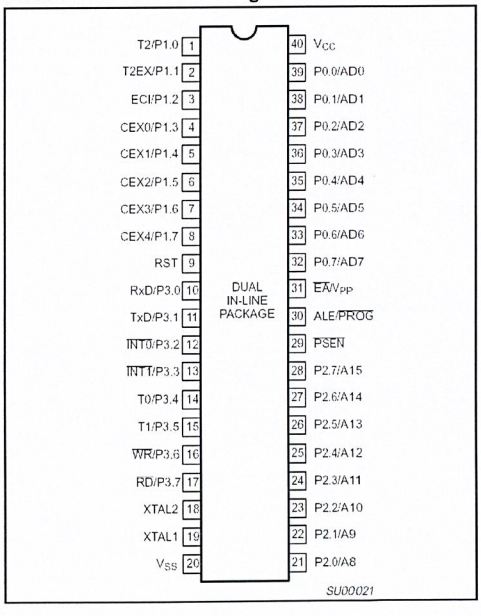


LOGIC SYMBOL

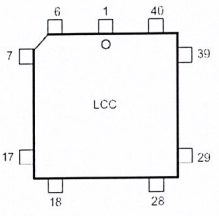


PINNING

Plastic Dual In-Line Package



Plastic Leaded Chip Carrier

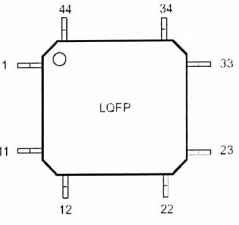


Pin	Function	Pin	Function	Pin	Function
1	NIC*	16	P3.4/T0	31	P2.7/A15
2	P1.0/T2	17	P3.5/T1	32	PSEN
3	P1.1/T2EX	18	P3.6/WR	33	ALE/PROG
4	P1.2/ECI	19	P3.7/RD	34	NIC*
5	P1.3/CEX0	20	XTAL2	35	EA/VPP
6	P1.4/CEX1	21	XTAL1	36	P0.7/AD7
7	P1.5/CEX2	22	VSS	37	P0.6/AD6
8	P1.6/CEX3	23	NIC*	38	P0.5/AD5
9	P1.7/CEX4	24	P2.0/A8	39	P0.4/AD4
10	RST	25	P2.1/A9	40	P0.3/AD3
11	P3.0/RxD	26	P2.2/A10	41	P0.2/AD2
12	NIC*	27	P2.3/A11	42	P0.1/AD1
13	P3.1/TxD	28	P2.4/A12	43	P0.0/AD0
14	P3.2/INT0	29	P2.5/A13	44	VCC
15	P3.3/INT1	30	P2.6/A14		

* NO INTERNAL CONNECTION

SU00023

Plastic Quad Flat Pack



Pin	Function	Pin	Function	Pin	Function
1	P1.5/CEX2	16	VSS	31	P0.6/AD6
2	P1.6/CEX3	17	NIC*	32	P0.5/AD5
3	P1.7/CEX4	18	P2.0/A8	33	P0.4/AD4
4	RST	19	P2.1/A9	34	P0.3/AD3
5	P3.0/RxD	20	P2.2/A10	35	P0.2/AD2
6	NIC*	21	P2.3/A11	36	P0.1/AD1
7	P3.1/TxD	22	P2.4/A12	37	P0.0/AD0
8	P3.2/INT0	23	P2.5/A13	38	VCC
9	P3.3/INT1	24	P2.6/A14	39	NIC*
10	P3.4/T0	25	P2.7/A15	40	P1.0/T2
11	P3.5/T1	26	PSEN	41	P1.1/T2EX
12	P3.6/WR	27	ALE/PROG	42	P1.2/ECI
13	P3.7/RD	28	NIC*	43	P1.3/CEX0
14	XTAL2	29	EA/VPP	44	P1.4/CEX1
15	XTAL1	30	P0.7/AD7		

* NO INTERNAL CONNECTION

SU01400

PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	LQFP		
V_{SS}	20	22	16	I	**Ground:** 0 V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups on all pins. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
					Alternate functions for P89C51RB2/RC2/RD2Hxx Port 1 include:
	1	2	40	I/O	**T2 (P1.0):** Timer/Counter 2 external count input/Clockout (see Programmable Clock-Out)
	2	3	41	I	**T2EX (P1.1):** Timer/Counter 2 Reload/Capture/Direction Control
	3	4	42	I	**ECI (P1.2):** External Clock Input to the PCA
	4	5	43	I/O	**CEX0 (P1.3):** Capture/Compare External I/O for PCA module 0
	5	6	44	I/O	**CEX1 (P1.4):** Capture/Compare External I/O for PCA module 1
	6	7	1	I/O	**CEX2 (P1.5):** Capture/Compare External I/O for PCA module 2
	7	8	2	I/O	**CEX3 (P1.6):** Capture/Compare External I/O for PCA module 3
	8	9	3	I/O	**CEX4 (P1.7):** Capture/Compare External I/O for PCA module 4
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register. P2.7 must be a "1" to program and erase the device.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the P89C51RB2/RC2/RD2Hxx, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**$\overline{INT0}$ (P3.2):** External interrupt
	13	15	9	I	**$\overline{INT1}$ (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	27	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted twice every machine cycle, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. ALE can be disabled by setting SFR auxiliary.0. With this bit set, ALE will be active only during a MOVX instruction.

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	LQFP		
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations. If $\overline{EA}$ is held high, the device executes from internal program memory. The value on the $\overline{EA}$ pin is latched when RST is released and any subsequent changes have no effect. This pin also receives the programming supply voltage (V_{PP}) during Flash programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin (other than V_{PP}) must not be higher than V_{CC} + 0.5 V or less than V_{SS} – 0.5 V.

MICROCHIP 24AA256/24LC256/24FC256

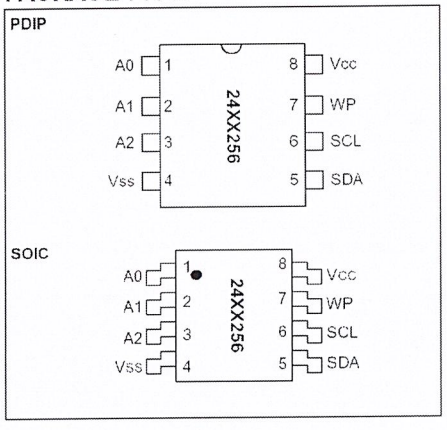
256K Bit I²C™ CMOS Serial EEPROM

DEVICE SELECTION TABLE

Part Number	Vcc Range	Max Clock Frequency	Temp Ranges
24AA256	1.8-5.5V	400 kHz†	I
24LC256	2.5-5.5V	400 kHz‡	I, E
24FC256	2.5-5.5V	1 MHz	I

†100 kHz for Vcc < 2.5V.
‡100 kHz for E temperature range.

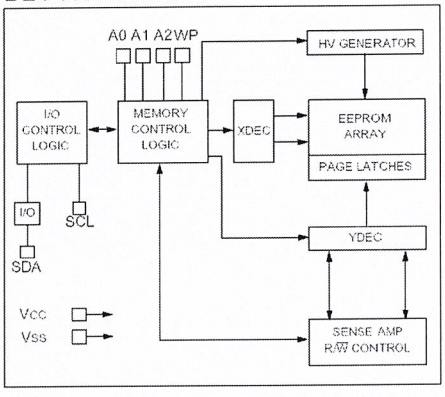
PACKAGE TYPE



FEATURES

- Low power CMOS technology
  - Maximum write current 3 mA at 5.5V
  - Maximum read current 400 µA at 5.5V
  - Standby current 100 nA typical at 5.5V
- 2-wire serial interface bus, I²C compatible
- Cascadable for up to eight devices
- Self-timed ERASE/WRITE cycle
- 64-byte page-write mode available
- 5 ms max write-cycle time
- Hardware write protect for entire array
- Output slope control to eliminate ground bounce
- Schmitt trigger inputs for noise suppression
- 100,000 erase/write cycles guaranteed
- Electrostatic discharge protection > 4000V
- Data retention > 200 years
- 8-pin PDIP and SOIC (208 mil) packages
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Automotive (E): -40°C to +125°C

BLOCK DIAGRAM



DESCRIPTION

The Microchip Technology Inc. 24AA256/24LC256/24FC256 (24XX256*) is a 32K x 8 (256K bit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8V to 5.5V). It has been developed for advanced, low power applications such as personal communications or data acquisition. This device also has a page-write capability of up to 64 bytes of data. This device is capable of both random and sequential reads up to the 256K boundary. Functional address lines allow up to eight devices on the same bus, for up to 2M bit address space. This device is available in the standard 8-pin plastic DIP and 8-pin SOIC (208 mil) packages.

I²C is a trademark of Philips Corporation.
*24XX256 is used in this document as a generic part number for the 24AA256/24LC256/24FC256 devices.

© 1998 Microchip Technology Inc.

MICROCHIP

24AA256/24LC256/24FC256

256K Bit I²C™ CMOS Serial EEPROM

DEVICE SELECTION TABLE

Part Number	Vcc Range	Max Clock Frequency	Temp Ranges
24AA256	1.8-5.5V	400 kHz†	I
24LC256	2.5-5.5V	400 kHz‡	I, E
24FC256	2.5-5.5V	1 MHz	I

†100 kHz for Vcc < 2.5V.
‡100 kHz for E temperature range.

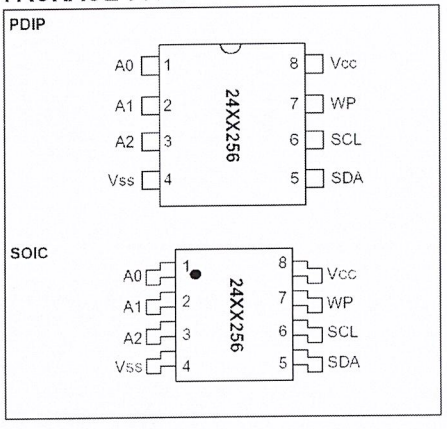
FEATURES

- Low power CMOS technology
  - Maximum write current 3 mA at 5.5V
  - Maximum read current 400 µA at 5.5V
  - Standby current 100 nA typical at 5.5V
- 2-wire serial interface bus, I²C compatible
- Cascadable for up to eight devices
- Self-timed ERASE/WRITE cycle
- 64-byte page-write mode available
- 5 ms max write-cycle time
- Hardware write protect for entire array
- Output slope control to eliminate ground bounce
- Schmitt trigger inputs for noise suppression
- 100,000 erase/write cycles guaranteed
- Electrostatic discharge protection > 4000V
- Data retention > 200 years
- 8-pin PDIP and SOIC (208 mil) packages
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Automotive (E): -40°C to +125°C

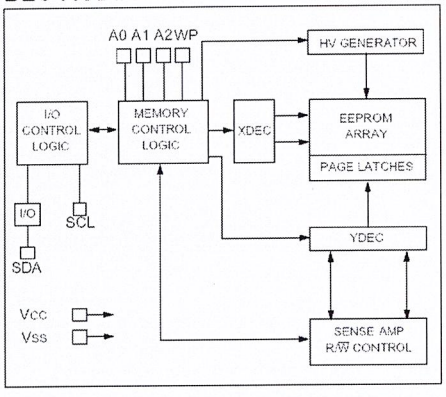
DESCRIPTION

The Microchip Technology Inc. 24AA256/24LC256/24FC256 (24XX256*) is a 32K x 8 (256K bit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8V to 5.5V). It has been developed for advanced, low power applications such as personal communications or data acquisition. This device also has a page-write capability of up to 64 bytes of data. This device is capable of both random and sequential reads up to the 256K boundary. Functional address lines allow up to eight devices on the same bus, for up to 2M bit address space. This device is available in the standard 8-pin plastic DIP and 8-pin SOIC (208 mil) packages.

PACKAGE TYPE



BLOCK DIAGRAM



I²C is a trademark of Philips Corporation.
*24XX256 is used in this document as a generic part number for the 24AA256/24LC256/24FC256 devices.

24AA256/24LC256/24FC256

1.0 ELECTRICAL CHARACTERISTICS

1.1 Maximum Ratings*

VCC.....7.0V
All inputs and outputs w.r.t. VSS ..... -0.6V to VCC +1.0V
Storage temperature .....-65°C to +150°C
Ambient temp. with power applied.....-65°C to +125°C
Soldering temperature of leads (10 seconds) .....+300°C
ESD protection on all pins.....≥ 4 kV

***Notice:** Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

TABLE 1-1 PIN FUNCTION TABLE

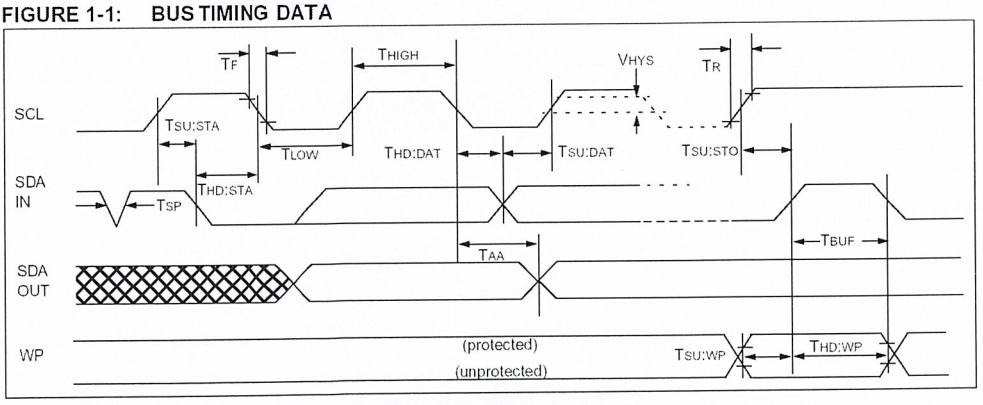
Name	Function
A0, A1, A2	User Configurable Chip Selects
VSS	Ground
SDA	Serial Data
SCL	Serial Clock
WP	Write Protect Input
VCC	+1.8 to 5.5V (24AA256) +2.5 to 5.5V (24LC256,24FC256)

TABLE 1-2 DC CHARACTERISTICS

All parameters apply across the specified operating ranges, unless otherwise noted.	Industrial (I): VCC = +1.8V to 5.5V Tamb = -40°C to +85°C Automotive (E): VCC = +4.5V to 5.5V Tamb = -40°C to 125°C				
Parameter	**Symbol**	**Min**	**Max**	**Units**	**Conditions**
A0, A1, A2, SCL, SDA, and WP pins:					
High level input voltage	VIH	0.7 VCC	—	V	
Low level input voltage	VIL	—	0.3 VCC	V	VCC 2.5V
			0.2 VCC	V	VCC < 2.5V
Hysteresis of Schmitt Trigger inputs (SDA, SCL pins)	VHYS	0.05 VCC	—	V	VCC 2.5V (Note)
Low level output voltage	VOL	—	0.40	V	IOL = 3.0 mA @ VCC = 4.5V IOL = 2.1 mA @ VCC = 2.5V
Input leakage current	ILI	-10	10	µA	VIN = VSS or VCC, WP = VSS VIN = VSS or VCC, WP = VCC
Output leakage current	ILO	-10	10	µA	VOUT = VSS or VCC
Pin capacitance (all inputs/outputs)	CIN, COUT	—	10	pF	VCC = 5.0V (Note) Tamb = 25°C, f_c= 1 MHz
Operating current	ICC Read	—	400	µA	VCC = 5.5V, SCL = 400 kHz
	ICC Write	—	3	mA	VCC = 5.5V
Standby current	ICCS	—	1	µA	SCL = SDA = VCC = 5.5V A0, A1, A2, WP = VSS

Note: This parameter is periodically sampled and not 100% tested.

FIGURE 1-1: BUS TIMING DATA



 © 1998 Microchip Technology Inc.

2.0 PIN DESCRIPTIONS

2.1 A0, A1, A2 Chip Address Inputs

The A0, A1, A2 inputs are used by the 24XX256 for multiple device operations. The levels on these inputs are compared with the corresponding bits in the slave address. The chip is selected if the compare is true.

Up to eight devices may be connected to the same bus by using different chip select bit combinations. If left unconnected, these inputs will be pulled down internally to Vss.

2.2 SDA Serial Data

This is a bi-directional pin used to transfer addresses and data into and data out of the device. It is an open-drain terminal, therefore, the SDA bus requires a pullup resistor to Vcc (typical 10 kΩ for 100 kHz, 2 kΩ for 400 kHz and 1 MHz).

For normal data transfer SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions.

2.3 SCL Serial Clock

This input is used to synchronize the data transfer from and to the device.

2.4 WP

This pin can be connected to either Vss, Vcc or left floating. An internal pull-down resistor on this pin will keep the device in the unprotected state if left floating. If tied to Vss or left floating, normal memory operation is enabled (read/write the entire memory 0000-7FFF).

If tied to Vcc, WRITE operations are inhibited. Read operations are not affected.

3.0 FUNCTIONAL DESCRIPTION

The 24XX256 supports a bi-directional 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter, and a device receiving data as a receiver. The bus must be controlled by a master device which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions while the 24XX256 works as a slave. Both master and slave can operate as a transmitter or receiver, but the master device determines which mode is activated.

4.0 BUS CHARACTERISTICS

The following **bus protocol** has been defined:

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is HIGH will be interpreted as a START or STOP condition.

Accordingly, the following bus conditions have been defined (Figure 4-1).

4.1 Bus not Busy (A)

Both data and clock lines remain HIGH.

4.2 Start Data Transfer (B)

A HIGH to LOW transition of the SDA line while the clock (SCL) is HIGH determines a START condition. All commands must be preceded by a START condition.

4.3 Stop Data Transfer (C)

A LOW to HIGH transition of the SDA line while the clock (SCL) is HIGH determines a STOP condition. All operations must end with a STOP condition.

4.4 Data Valid (D)

The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal.

The data on the line must be changed during the LOW period of the clock signal. There is one bit of data per clock pulse.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of the data bytes transferred between the START and STOP conditions is determined by the master device.

4.5 Acknowledge

Each receiving device, when addressed, is obliged to generate an acknowledge signal after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

Note:	The 24XX256 does not generate any acknowledge bits if an internal programming cycle is in progress.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. During reads, a master must signal an end of data to the slave by NOT generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave (24XX256) will leave the data line HIGH to enable the master to generate the STOP condition.

FIGURE 4-1: DATA TRANSFER SEQUENCE ON THE SERIAL BUS

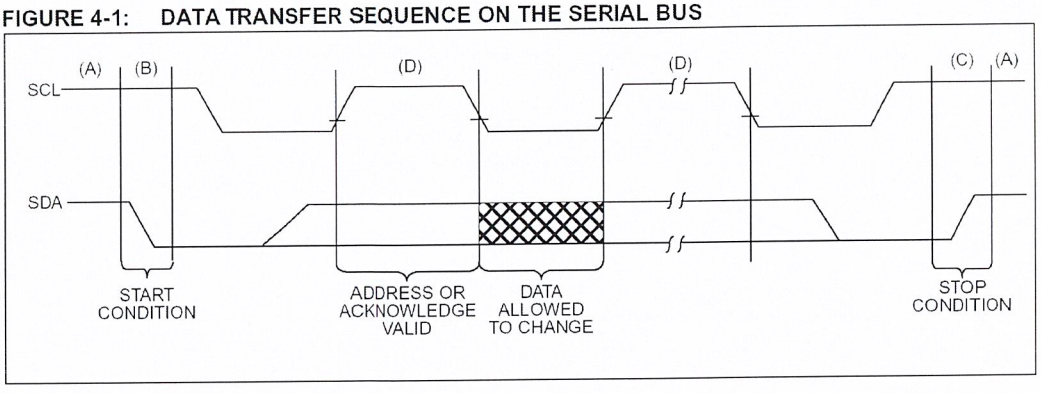
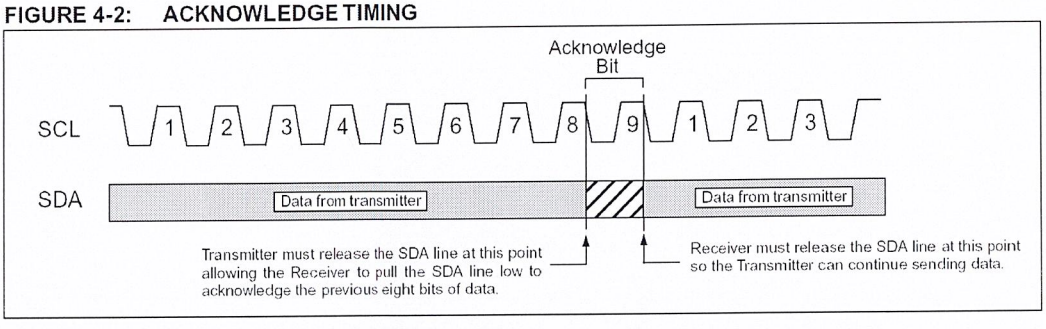


FIGURE 4-2: ACKNOWLEDGE TIMING



 © 1998 Microchip Technology Inc.

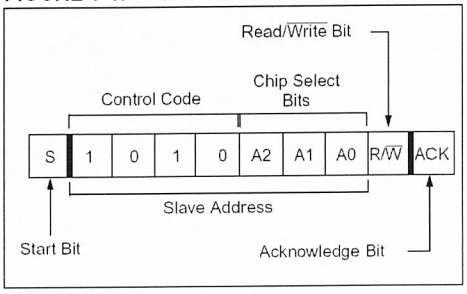
5.0 DEVICE ADDRESSING

A control byte is the first byte received following the start condition from the master device (Figure 5-1). The control byte consists of a 4-bit control code; for the 24XX256 this is set as 1010 binary for read and write operations. The next three bits of the control byte are the chip select bits (A2, A1, A0). The chip select bits allow the use of up to eight 24XX256 devices on the same bus and are used to select which device is accessed. The chip select bits in the control byte must correspond to the logic levels on the corresponding A2, A1, and A0 pins for the device to respond. These bits are in effect the three most significant bits of the word address.

The last bit of the control byte defines the operation to be performed. When set to a one a read operation is selected, and when set to a zero a write operation is selected. The next two bytes received define the address of the first data byte (Figure 5-2). Because only A14...A0 are used, the upper address bits is a don't care. The upper address bits are transferred first, followed by the less significant bits.

Following the start condition, the 24XX256 monitors the SDA bus checking the device type identifier being transmitted. Upon receiving a 1010 code and appropriate device select bits, the slave device outputs an acknowledge signal on the SDA line. Depending on the state of the R/$\overline{W}$ bit, the 24XX256 will select a read or write operation.

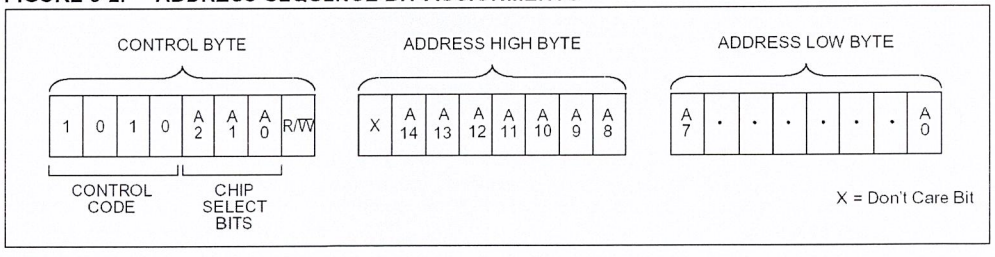
FIGURE 5-1: CONTROL BYTE FORMAT



5.1 Contiguous Addressing Across Multiple Devices

The chip select bits A2, A1, A0 can be used to expand the contiguous address space for up to 2 Mbit by adding up to eight 24XX256's on the same bus. In this case, software can use A0 of the control byte as address bit A15; A1, as address bit A16; and A2, as address bit A17. It is not possible to sequentially read across device boundaries.

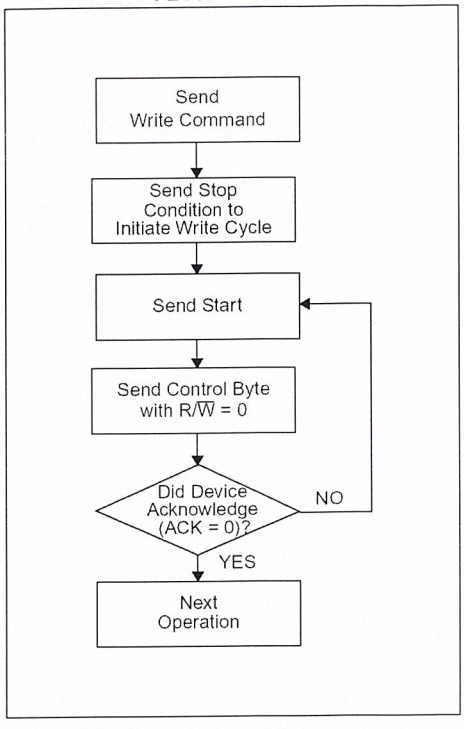
FIGURE 5-2: ADDRESS SEQUENCE BIT ASSIGNMENTS



7.0 ACKNOWLEDGE POLLING

Since the device will not acknowledge during a write cycle, this can be used to determine when the cycle is complete (This feature can be used to maximize bus throughput.) Once the stop condition for a write command has been issued from the master, the device initiates the internally timed write cycle. ACK polling can be initiated immediately. This involves the master sending a start condition, followed by the control byte for a write command (R/$\overline{W}$ = 0). If the device is still busy with the write cycle, then no ACK will be returned. If no ACK is returned, then the start bit and control byte must be resent. If the cycle is complete, then the device will return the ACK, and the master can then proceed with the next read or write command. See Figure 7-1 for flow diagram.

FIGURE 7-1: ACKNOWLEDGE POLLING FLOW



DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

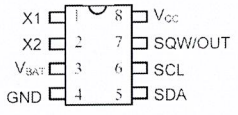
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

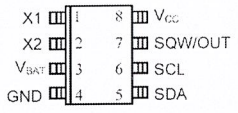
ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

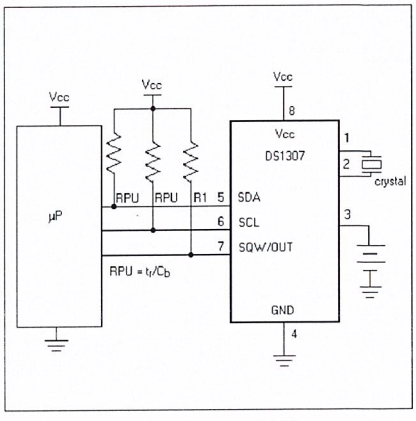
PIN DESCRIPTION

V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

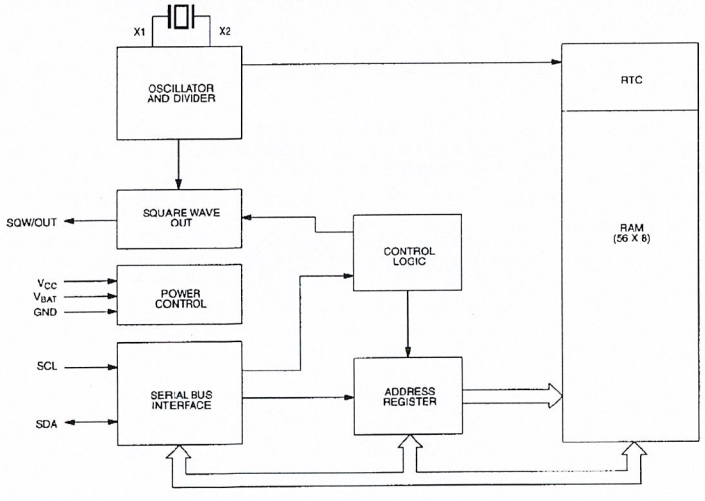
TYPICAL OPERATING CIRCUIT



OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low-current battery backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} + 0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

DS1307 BLOCK DIAGRAM Figure 1



CLOCK ACCURACY

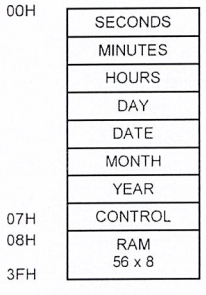
The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. See Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks" for detailed information.

Please review Application Note 95, "Interfacing the DS1307 with a 8051-Compatible Microcontroller" for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307 is shown in Figure 2. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

The time and calendar information is obtained by reading the appropriate register bytes. The RTC registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. Bit 7 of register 0 is the clock halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power-on state of all registers is not defined. Therefore, it is important to enable the oscillator (CH bit = 0) during initial configuration.

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

On a 2-wire START, the current time is transferred to a second set of registers. The time information is read from these secondary registers, while the clock may continue to run. This eliminates the need to re-read the registers in case of an update of the main registers during a read.

ZARLINK™ SEMICONDUCTOR

ISO²-CMOS MT8870D/MT8870D-1
Integrated DTMF Receiver

ISSUE 5 March 1997

Features
- Complete DTMF Receiver
- Low power consumption
- Internal gain setting amplifier
- Adjustable guard time
- Central office quality
- Power-down mode
- Inhibit mode
- Backward compatible with MT8870C/MT8870C-1

Applications
- Receiver system for British Telecom (BT) or CEPT Spec (MT8870D-1)
- Paging systems
- Repeater systems/mobile radio
- Credit card systems
- Remote control
- Personal computers
- Telephone answering machine

Ordering Information

MT8870DE/DE-1	18 Pin Plastic DIP
MT8870DS/DS-1	18 Pin SOIC
MT8870DN/DN-1	20 Pin SSOP

-40 °C to +85 °C

Description

The MT8870D/MT8870D-1 is a complete DTMF receiver integrating both the bandsplit filter and digital decoder functions. The filter section uses switched capacitor techniques for high and low group filters; the decoder uses digital counting techniques to detect and decode all 16 DTMF tone-pairs into a 4-bit code. External component count is minimized by on chip provision of a differential input amplifier, clock oscillator and latched three-state bus interface.

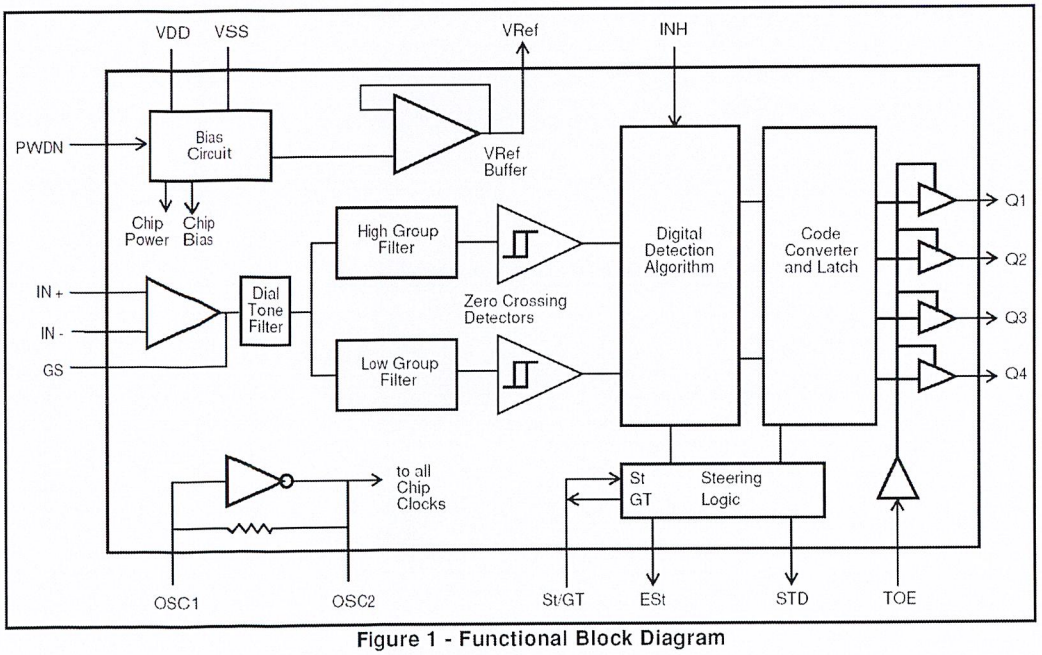


Figure 1 - Functional Block Diagram

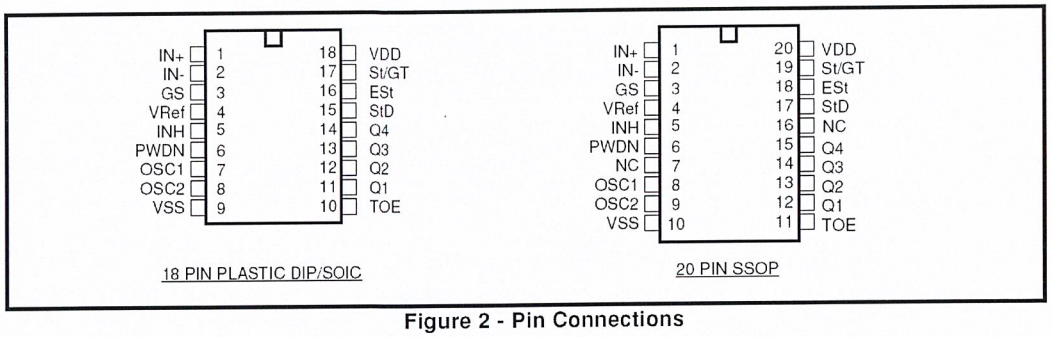


Figure 2 - Pin Connections

Pin Description

Pin #		Name	Description
18	20		
1	1	IN+	**Non-Inverting Op-Amp (Input).**
2	2	IN-	**Inverting Op-Amp (Input).**
3	3	GS	**Gain Select.** Gives access to output of front end differential amplifier for connection of feedback resistor.
4	4	V_{Ref}	**Reference Voltage (Output).** Nominally $V_{DD}/2$ is used to bias inputs at mid-rail (see Fig. 6 and Fig. 10).
5	5	INH	**Inhibit (Input).** Logic high inhibits the detection of tones representing characters A, B, C and D. This pin input is internally pulled down.
6	6	PWDN	**Power Down (Input).** Active high. Powers down the device and inhibits the oscillator. This pin input is internally pulled down.
7	8	OSC1	**Clock (Input).**
8	9	OSC2	**Clock (Output).** A 3.579545 MHz crystal connected between pins OSC1 and OSC2 completes the internal oscillator circuit.
9	10	V_{SS}	**Ground (Input).** 0V typical.
10	11	TOE	**Three State Output Enable (Input).** Logic high enables the outputs Q1-Q4. This pin is pulled up internally.
11-14	12-15	Q1-Q4	**Three State Data (Output).** When enabled by TOE, provide the code corresponding to the last valid tone-pair received (see Table 1). When TOE is logic low, the data outputs are high impedance.
15	17	StD	**Delayed Steering (Output).** Presents a logic high when a received tone-pair has been registered and the output latch updated; returns to logic low when the voltage on St/GT falls below V_{TSt}.
16	18	ESt	**Early Steering (Output).** Presents a logic high once the digital algorithm has detected a valid tone pair (signal condition). Any momentary loss of signal condition will cause ESt to return to a logic low.
17	19	St/GT	**Steering Input/Guard time (Output) Bidirectional.** A voltage greater than V_{TSt} detected at St causes the device to register the detected tone pair and update the output latch. A voltage less than V_{TSt} frees the device to accept a new tone pair. The GT output acts to reset the external steering time-constant; its state is a function of ESt and the voltage on St.
18	20	V_{DD}	**Positive power supply (Input).** +5V typical.
	7, 16	NC	No Connection.

ประวัติผู้แต่ง

ชื่อ-สกุล	นายกอเฉ็ม เตาวะโต
วัน เดือน ปีเกิด	4 มีนาคม พ.ศ. 2524
ภูมิลำเนา	73 หมู่ที่ 1 ตำบลคลองขุด อำเภอเมือง
	จังหวัดสตูล 91000 โทรศัพท์ 0-6994-4369
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านเขาจีน จังหวัดสตูล
มัธยมศึกษาตอนต้น	โรงเรียนพัฒนาการศึกษามูลนิธิ จังหวัดสตูล
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคสตูล
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคสตูล
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม สจล.
ผลงานที่ได้รับรางวัล	รางวัลชนะเลิศการประกวดสิ่งประดิษฐ์ของคนรุ่นใหม่
	ระดับภาค ภาคใต้ ปีการศึกษา 2544
	รางวัลรองชนะเลิศอันดับ 1 การประกวดสิ่งประดิษฐ์
	ของคนรุ่นใหม่ ระดับชาติ ครั้งที่ 12 ปีการศึกษา 2544
คติพจน์	ในโลกนี้ไม่มีความสุขที่แท้จริง มีแต่ความทุกข์ที่เราทนได้
	เราต้องฝ่าฟันมันไป

ประวัติผู้แต่ง

ชื่อ-สกุล	นายเดชชาติ สุขจิต
วัน เดือน ปีเกิด	28 กรกฎาคม พ.ศ. 2524
ภูมิลำเนา	66 หมู่ที่ 3 ตำบลเมืองลีง อำเภอจอมพระ จังหวัดสุรินทร์ 32180 โทรศัพท์ 0-6642-4250
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านทวารไพร จังหวัดสุรินทร์
มัธยมศึกษาตอนต้น	โรงเรียนจอมพระประชาสรรค์ จังหวัดสุรินทร์
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคสุรินทร์
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคสุรินทร์
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
ทุนการศึกษา	ทุนการศึกษาภาษาและวัฒนธรรม ณ เมืองเบปปุ ประเทศญี่ปุ่น
คติพจน์	ฝันให้ไกล ไปให้ถึง

ประวัติผู้แต่ง

ชื่อ-สกุล	นายปรมี แสงเพ็ชร
วัน เดือน ปีเกิด	1 กรกฎาคม พ.ศ. 2524
ภูมิลำเนา	33 หมู่ที่ 2 ตำบลหนองเต็ง อำเภอกระสัง
	จังหวัดบุรีรัมย์ 31160 โทรศัพท์ 0-6715-1826
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านโนนแดง จังหวัดบุรีรัมย์
มัธยมศึกษาตอนต้น	โรงเรียนกระสังพิทยาคม จังหวัดบุรีรัมย์
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคบุรีรัมย์
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคบุรีรัมย์
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม สจล.
คติพจน์	อยู่ให้เขาไว้ใจ จากไปให้เขาคิดถึง

ประวัติผู้แต่ง

ชื่อ-สกุล	นายปรีชา คงแก้ว
วัน เดือน ปีเกิด	29 กรกฎาคม พ.ศ. 2524
ภูมิลำเนา	11 หมู่ที่ 4 ตำบลดอนทราย อำเภอปากพะยูน
	จังหวัดพัทลุง 93120 โทรศัพท์ 0-9661-7173
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านโคกทราย จังหวัดพัทลุง
มัธยมศึกษาตอนต้น	โรงเรียนหารเทา จังหวัดพัทลุง
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคสตูล
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคสตูล
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม สจล.
ทุนการศึกษา	ทุนการศึกษาเรียนดี ปีการศึกษา 2543
คติพจน์	อย่านอนตื่นสาย อย่าอายทำกิน อย่าหมิ่นเงินน้อย
	อย่าคอยวาสนา