เครื่องควบคุมอุปกรณ์ทางโทรศัพท์

(TELEPHONE COMMANDER)

ผู้จัดทำ

นาย กุศ	สองทิศ	321025
นาย คมกฤช	แก้ววิเศษ	321043
นาย บัณฑิต	วิจิตรประไพ	321158

อาจารย์ที่ปรึกษา

ผศ. พิพัฒน์ เลาหสงคราม

ปริญญานิพนธ์นี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมเทคโนโลยีการวัดคุมทางอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

เรื่อง เครื่องควบคุมอุปกรณ์ทางโทรศัพท์
TELEPHONE COMMANDER

ผู้จัดทำ

นาย กุศ	สองทิศ	321025
นาย คมกฤช	แก้ววิเศษ	321043
นาย บัณฑิต	วิจิตรประไพ	321158

.....อาจารย์ที่ปรึกษา
(ผศ. พิพัฒน์ เลาหสงคราม)

สารบัญ

บทคัดย่อ

เครื่องควบคุมอุปกรณ์ทางโทรศัพท์ เป็นการนำไมโครโปรสเซสเซอร์ มาประยุกต์ใช้งานใช้การควบคุมการทำงานของอุปกรณ์ โดยผู้สั่งงานสามารถสั่งงานได้โดยการกดปุ่มหมายเลขโทรศัพท์ วงจร DTMF ถึงจะทำการถอดรหัสจากสัญญาณความถี่เป็นดิจิตอลสัญญาณดิจิตอลที่ได้ก็จะถูกไมโครโปรเซสเซอร์ MCS - 51 ทำการประมวลผล โดยมีวงจรโซลิดสเตทรีเลย์เป็นตัว เปิด-ปิดอุปกรณ์ พร้อมทั้งมีการตั้งเวลาการเปิด-ปิด อุปกรณ์ด้วย มี 7-Segment เป็นตัวแสดงค่าเวลาที่ใช้วงจร RTC เป็นฐานเวลา

Abstract

Telephone Commander is the device that bring the microprocessor to contorl electrical Appliance workings. The User can Command the device by pushing the buttons. DTMF decoder will decode frequency signal to digital Signal. And the digital signal is calculated by The mcs - 51 Microprocessor. The main parts of the device consist of Solid State circuit is the unit for the electrical appliances,the seven-segment for display the time which is based on RTC circuit.

บทนำ

ในปัจจุบันระบบโทรศัพท์ถือได้ว่าเป็นเครื่องมือสื่อสารที่สำคัญที่สุดโดยมีเครือข่ายการติดต่อที่กว้างขวางด้วยเหตุว่าโทรศัพท์มือเครือข่ายที่กว้างขวางนี้เอง ทำให้มีการหันมาใช้ประโยชน์จากเครือข่ายสายโทรศัพท์หลายรูปแบบ ไม่ว่าจะเป็น FAX ระบบ Computer Online ฯลฯในโครงงานชิ้นนี้ก็ได้นำประโยชน์จากสายโทรศัพท์มาประยุกต์ใช้ในการควบคุมสั่งงานอุปกรณ์ ต่าง ๆ โดยผู้ใช้สามารถสั่งงานได้โดยการกดรหัสผ่านที่ผู้ใช้กำหนดขึ้นเอง เมื่อทำการกดรหัสผ่านแล้ว ก็จะได้เข้าสู่การสั่งงานอุปกรณ์ไฟฟ้า ซึ่งการทำงานของเครื่องควบคุมอุปกรณ์ทางโทรศัพท์นี้สามารถทำงานได้

การใช้งาน TELEPHONE COMANDER

ใช้โทรศัพท์แบบกดปุ่ม -เมื่อโทรศัพท์ดัง 10 ครั้ง โดยไม่มีคนรับก็จะเข้าการทำงานของ

โปรแกรมโดยเราจะต้องกดตัวเลข 9 ตัวซึ่งเป็นรหัสดังนี้

ตัวที่ 1 รหัสของเครื่องไฟฟ้า (1-8)

ตัวที่ 2 เวลาเปิด (ชั่วโมง) หลักแรก (0-2)

ตัวที่ 3 เวลาเปิด (ชั่วโมง) หลักที่ 2 (0-9)

ตัวที่ 4 เวลาเปิด (นาที) หลักแรก (0-5)

ตัวที่ 5 เวลาเปิด (นาที) หลักที่ 2 (0-9)

ตัวที่ 6 เวลาปิด (ชั่วโมง) หลักแรก (0-2)

ตัวที่ 7 เวลาปิด (ชั่วโมง) หลักที่ 2 (0-9)

ตัวที่ 8 เวลาปิด (นาที) หลักแรก (0-5)

ตัวที่ 9 เวลาปิด (นาที) หลักที่ 2 (0-9)

เมื่อกดครบทั้ง 9 ตัวแล้วก็วางสายโทรศัพท์ได้เลย

ตัวอย่างเช่น

208300930

หมายความว่า ให้เครื่องไฟฟ้าตัวที่ 2 เปิดเวลา 8.30 ปิดเวลา 9.30

บทที่ 1

ทฤษฎีและหลักการทำงาน

ระบบโทรศัพท์

ส่วนประกอบของระบบโทรศัพท์

โทรศัพท์ คือ เครื่องมือสื่อสารเชื่อโยงนำเสียงพูดระหว่างผู้ใช้ที่อยู่ ณ สถานที่แห่งหนึ่งกับบุคคลที่ต้องการติดต่อด้วย ณ สถานที่อีกแห่งหนึ่ง ให้สามารถพูดจาติดต่อกันได้เหมือนบุคคลทั้งสองนั่งสนทนาอยู่ด้วยกัน

ระบบโทรศัพท์มีส่วนประกอบที่สำคัญ 3 ส่วนด้วยกัน คือ

1. เครื่องรับโทรศัพท์

เครื่องรับโทรศัพท์เป็นอุปกรณ์ที่ผู้ใช้จะใช้ในการติดต่อระหว่างกันประกอบด้วย เครื่องส่ง(transmitter) เครื่องรับ(resiever) กระดิ่ง(ringer) อุคสวิทช์ (hook switch) หน้าปัทม์ สำหรับหมุนหรือกดหมายเลข

เครื่องส่งและเครื่องรับรวมกันเรียกว่า Hand set

2. สายโทรศัพท์

เครื่องรับโทรศัพท์แต่ละเครื่อง จะมีสายโทรศัพท์ 1 คู่ เพื่อเชื่อโยงและเป็นสื่อนำสัญญาณต่าง ๆ จากชุมสายมายังตัวเครื่องรับโทรศัพท์ในขนาดเดียวกัน ก็ทำหน้าที่เป็นสื่อในการส่งสัญญาณไฟฟ้าที่แปลงมาจากสัญญาณเสียงระหว่างเครื่องรับโทรศัพท์

สายโทรศัพท์ ที่ต่อเชื่อมโยงระหว่างชุมสายเพื่อให้บริการระหว่างชุมสายเรียกว่า ทรังค์ (trumk)

3. ชุมสายโทรศัพท์

ชุมสายโทรศัพท์ เป็นสถานที่ที่รวมคู่สายของเครื่องรับโทรศัพท์แต่ละเครื่องในระแวก ใกล้เคียงกัน และทำหน้าที่เชื่อมคู่สายให้กับผู้ใช้โทรศัพท์ พร้อมกับส่งสัญญาณแจ้งภาวะการใช้ต่าง ๆ ให้ผู้ใช้ทราบ

ชุมสายโทรศัพท์ระบบเก่าจะเป็นระบบใช้พนักงานต่อ(manual telephone) ซึ่งในปัจจุบันได้รับการพัฒนามาเป็นระบบอัตโนมัติ (automatic telephone)

สัญญาณพื้นฐานที่ใช้ในระบบโทรศัพท์

สัญญาณพื้นฐานที่ใช้ในระบบโทรศัพท์ คือ สัญญาณที่แจ้งภาวะการใช้โทรศัพท์ ซึ่งแบ่งออกเป็น 4 ประเภท คือ

1. สัญญาณพร้อมให้หมุน (dial tone) เป็นสัญญาณที่ทางชุมสายใช้แจ้งไปยังผู้เรียกใช้โทรศัพท์ว่า อุปกรณ์ต่าง ๆ ภายในชุมสายพร้อมที่จะทำการต่อโทรศัพท์ให้แก่ผู้ใช้โทรศัพท์

2. สัญญาณเรียกกลับ (ringback tone) เป็นสัญญาณที่บอกให้ผู้เรียกทราบว่าทางสายของผู้ถูกเรียกว่าง และกำลังเรียกอยู่

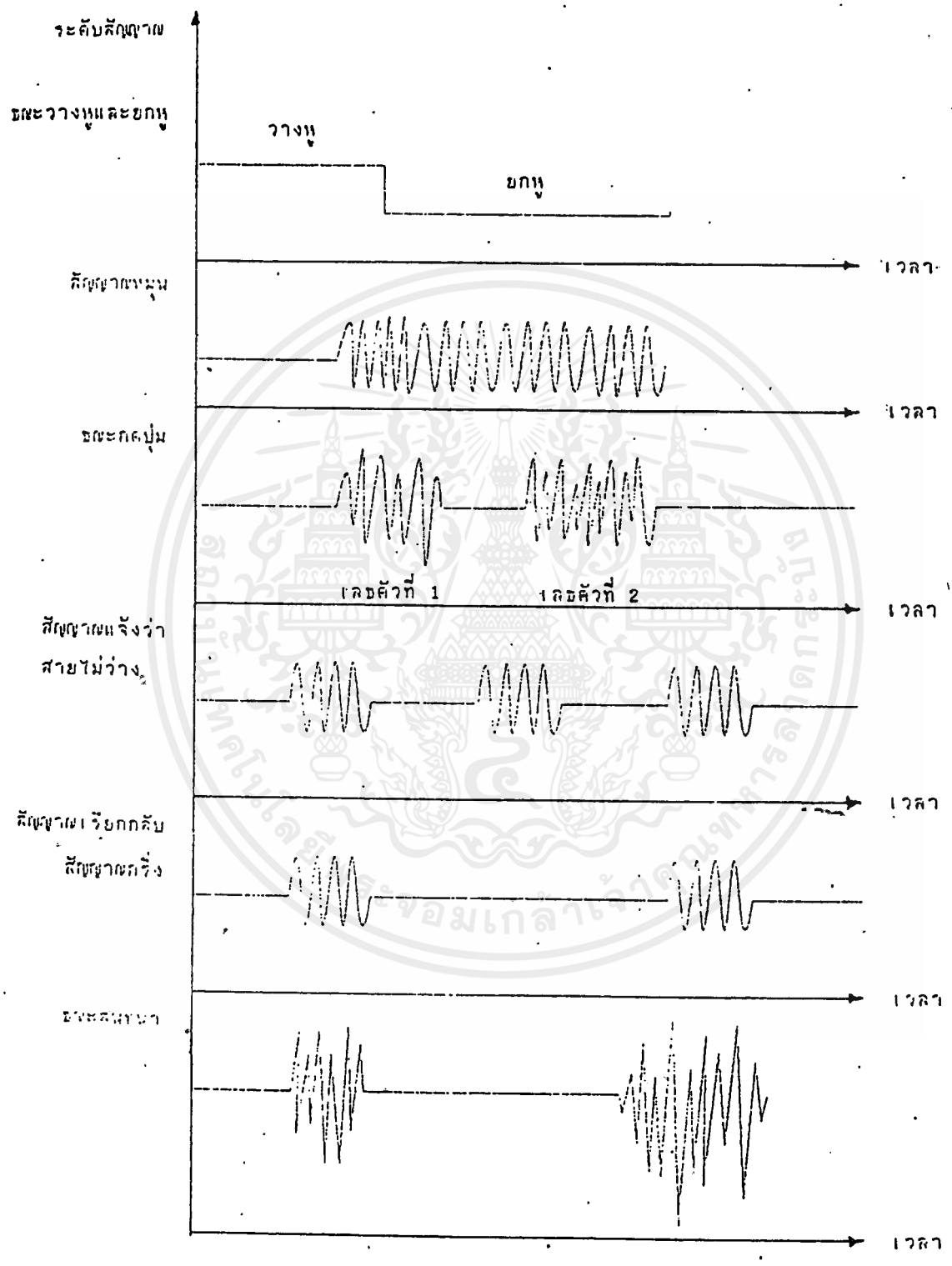
3. สัญญาณกริ่ง (ringing tone) เป็นสัญญาณที่ชุมสายส่งไปยังเครื่องผู้รับบอกให้รู้ว่ามีการติดต่อกัน

4. สัญญาณแจ้งว่าสายไม่ว่าง (busy tone) เป็นสัญญาณที่บอกให้ผู้เรียกทราบว่ายังไม่สามารถติดต่อกับเครื่องรับโทรศัพท์ หมายเลขนั้นในเวลานั้นได้

คุณสมบัติของสัญญาณ ต่าง ๆ เหล่านี้ มีแสดงในตาราง และรูป

ตารางคุณสมบัติของสัญญาณต่าง ๆ ที่ใช้ในการแจ้งภาวะการใช้โทรศัพท์

ชนิดของสัญญาณ	การส่งสัญญาณ	ความถี่ (เฮิร์ทช)
สัญญาณพร้อมให้หมุน	ต่อเนื่องไม่ขาดหาย	480
สัญญาณเรียกกลับ	1 วินาที ขาดหาย 4 วินาที	25
สัญญาณกริ่ง	1 วินาที ขาดหาย 4 วินาที	440 มอดูเลทกับ 480
สัญญ่าณแจ้งว่าสายไม่ว่าง	- ขาดหาย 30 ครั้งต่อนาทีเมื่อสายในชุมสายไม่ว่าง - ขาดหาย 60 ครั้งต่อนาทีเมื่อเครื่องรับโทรศัพท์ที่ต้องการติดต่อถูกใช้อยู่ - ขาดหาย 120 ครั้งต่อนาที เมื่อทรังค์ไม่ว่าง	480 มอดูเลทกับ 620



รูปที่ 1 คุณสมบัติของสัญญาณต่าง ๆ ที่ใช้ในระบบโทรศัพท์

ลักษณะทางไฟฟ้าของสัญญาณ

สัญญาณระหว่างคู่ สายโทรศัพท์มีทั้งสัญญาณไฟกระแสตรง (DC) และสัญญาณไฟกระแสสลับ (AC) ซึ่งระดับสัญญาณระหว่างคู่สายโทรศัพท์จะแตกต่างกันไปดังที่แสดงไว้ในตารางและรูปที่ 1

ตารางระดับสัญญาณระหว่างคู่สายโทศัพท์ในช่วงการใช้งานต่าง ๆ

ช่วงเวลาการใช้งาน	ระดับสัญญาณไฟกระแสตรง	ระดับสัญญาณไฟกระแสสลับ
ไม่ได้ใช้งาน ไม่ได้ยกหูฟังขึ้น	48 โวลท์	-
ยกหูฟังขึ้นมีสัญญาณหมุน	10 "	600 มิลลิโวลท์
กดหมายเลข	10 "	ไม่เกิน 0.5 โวลท์
มีสัญญาณแจ้งว่าสายไม่ว่าง	10 "	400 มิลลิโวลท์
มีสัญญาณเรียกกลับ	10 "	400 "
มัสัญญาณกริ่ง(สำหรับเครื่องผู้รับ)	48 "	100 โวลท์
มีการพูดระหว่างสาย	10 "	ไม่เกิน 1 โวลท์ (สัญญาณเสียง)

ระบบ DTMF

ในยุคแรกเริ่มของการใช้โทรศัพท์ เครื่องรับโทรศัพท์ที่ใช้จะเป็นแบบหน้าปัทม์หมุน ซึ่งการหมุนหมายเลขจะทำให้เกิดพัลส์ (pulse) ของกระแสในจำนวนเท่ากับหมายเลขที่หมุน จะพัลส์ที่เกิดขึ้นจะถูกส่งไปยังชุมสายด้วยความเร็ว 10 พัลส์ ต่อวินาที (pulse per second, pps)หรือ 20 พัลส์ต่อวินาที

เนื่องจากโทรศัพท์ที่ใช้ระบบหน้าปัทม์หมุนสำหรับการติดต่อผ่านชุมสายไม่ค่อยจะอำนวยความสะดวกให้กับผู้ใช้โทรศัพท์เท่าใดนัก เพราะเป็นระบบเชิงกล ทำงานค่อนข้างช้า ดังนั้น จึงได้มีการคิดสร้างโทรศัพท์ชนิดกดปุ่มขึ้นระบบโทรศัพท์ชนิดกดปุ่มนี้เรียกว่า ระบบ dtmf (dual tone multifrequency) เนื่องจากการกดปุ่มหมายเลขแต่ละปุ่มบนหน้าปัทม์เครื่องรับโทรศัพท์นั้น ทำให้เกิดสัญญาณที่ประกอบขึ้นจากความถี่ 2 ความถี่ ส่งออกไปตามสายโทรศัพท์ไปยังชุมสาย เพื่อเรียกให้ชุมสายรู้ว่าผู้ใช้โทรศัพท์ต้องการติดต่อกับโทรศัพท์เครื่องใดแทนการส่งพัลส์ของกระแสการหมุนหน้าปัทม์ของเครื่องรับโทรศัพท์หน้าปัทม์หมุน

ข้อดีของการใช้โทรศัพท์แบบกดปุ่ม

- ลดเวลาในการเรียกหมายเลขลง
- การเรียกเลขหมายทำได้ง่ายขึ้น
- สามารถใช้วงจรทางโซลิตสเตทอิเทคทรอนิคส์แทนอุปกรณ์เชิงกล
- มีความผิดพลาดในการส่งหมานเลขน้อย
- ใช้สัญญาณระบบความถี่เสียง ซึ่งสามารถส่งระหว่างสถานีได้และสามารถนำไปใช้งานได้หลายอย่าง
- สามารที่จะเพิ่มปุ่มได้อีก 4 ปุ่มในคอลัมน์ที่ 4 เพื่อการใช้งานอย่างอื่น

สิ่งที่สำคัญทีสุดสำหรับโทรศัพท์แบบกดปุ่ม คือ ระบบ DTMF

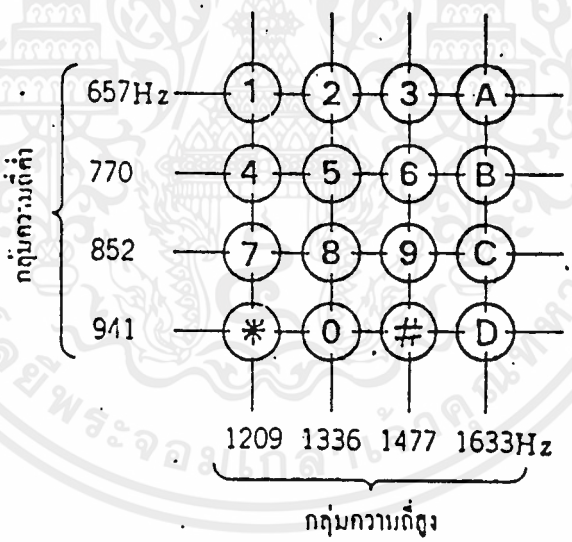
<u>ระบบ DTMF</u>

ระบบ DTMF นี้จะมีความถี่มาตรฐานในย่านความถี่เสียงที่แตกต่างกัน 8 ความถี่โดยจะแบ่งความถี่เหล่านี้ออกเป็น 2 กลุ่ม คือ กลุ่มความถี่ต่ำ 4 ความถี่ และกลุ่มความถีสูง 4 ความถี่และสัญญาณ DTMF นี้ จะมาจากการรวมสัญญาณความถี่จากกลุ่มความถี่ต่ำ 1 ความถี่ และสัญญาณความถี่จากกลุ่มความถี่สูงอีก 1 ความถี่ ดังนั้น สัญญาณ DTMF จึงมีได้ทั้งหมด 16 สัญญาณ (4 ความถี่ต่ำ x 4 ความถีสูง)

การเลือกความถี่มาตรฐานของระบบ DTMF นี้ ผู้ออกแบบระบบได้ใช้ความพยายามอย่างมากในการเลือกความถี่ที่จะใช้เพราะเนื่องจากต้องระวังไม่ให้สัญญาณเสียงต่าง ๆ ที่

เกิดขึ้นในสายโทรศัพท์ เช่น สัญญาณแจ้งสถานะการใช้ต่างๆ สัญญาณลบกวนภายในสายโทรศัพท์ มีความถี่อยู่ในช่วงความถี่ DTMF และยังต้องระวังความถี่ที่อาจเกิดขึ้นจากการรวมตัวกันแบบฮาร์โมนิกของค่าความถี่ใดความถี่หนึ่งของความถี่ DTMF และในที่สุดก็ได้ความถี่มาตรฐานทั้งแปด ดังนี้.

- ความถี่มาตรฐานในกลุ่มความถี่ต่ำ 4 ความถี่ คือ 697, 770, 852 และ 941 เฮิร์ทช
- ความถี่มาตรฐานในกลุ่มความถี่สูง 4 ความถี่ คือ 1209,1336,1477และ 1633 "
- ระบบ DTMF นี้เรียกอีกอย่างหนึ่งว่า ระบบ 4x4 เนื่องจากการใช้แป้นกดขนาด4x4 ในการสร้างสัญญาณ DTMF และได้กำหนดปุ่มแต่ละปุ่มเหล่านั้นได้ด้วยตัวเลข 0-9,* (star หรือ saterisk), # (pound หรือ octophorpe), A,B,C,และ D ซึ่งในการกดปุ่ม ๆ หนึ่งจะให้สัญญาณความถี่คู่หนึ่งออกมาดังแสดงไว้ในรูปที่ 2



รูปที่ 2 การจัดปุ่มและระบบสัญญาณ

ในการใช้งานทั่วไป จะใช้เฉพาะปุ่มตัวเลข 0-9 เท่านั้น ส่วนปุ่ม *,#,A,B,C และ D สามารถนำมาใช้งานอื่น ๆ ได้

โทรศัพท์ที่ใช้ระบบ DTMF ผู้ใช้สามารถฟังเสียงสัญญาณ DTMF ได้โดยการยกหูโทรศัพท์ขึ้นฟัง แล้วกดปุ่มได้ปุ่มหนึ่งบนหน้าปัทม์ ตัวอย่างเช่นการกดปุ่มหมายเลข 8 จะเกิดสัญญาณความถี่ 852 เฮิร์ทช และ 1336 เฮิร์ทช ขึ้นพร้อมกัน สัญญาณจะถูกส่งผ่านคู่สายไปยัง

ชุมสาย แลถูกถอดรหัสโดยตัวรับ DTMF ที่ชุมสายโทรศัพท์จะประกอบด้วยอุปกรณ์ควบคุมที่ทำหน้าที่ในการจัดการติดต่อโทรศัพท์ภายในท้องถิ่น ในบริเวณหนึ่งภายในพื้นที่หนึ่ง ๆ ซึ่งถูกกำหนดด้วยตัวเลข 3 ตัวแรกของเลขหมายหลังจากที่ชุมสายทำการเชื่อมคู่สายระหว่างผู้เรียกและผู้ถูกเรียกเรียบร้อยแล้ว ตัวรับสัญญาณ DTMF ของชุมสายจะหยุดทำงานเพราะในการกดปุ่มบนหน้าปัทม์ โทรศัพท์ครั้งต่อไปจะเป็นการติดต่อกันโดยตรงระหว่างผู้เรียกและผู้ถูกเรียก

วงจรถอดรหัสความถี่ DTMF

การถอดรหัสความถี่ DTMF คือ การแปลงสัญญาณความถี่คู่ที่เกิดจากการกดปุ่มบนหน้าปัทม์โทรศัพท์ ให้เป็นระบบตัวเลขทางดิจิคอล หรือเลขฐานสองขนาด 4 บิท ซึ่งค่าเลขฐานสองที่ได้จากการถอดรหัส แสดงไว้ในตาราง

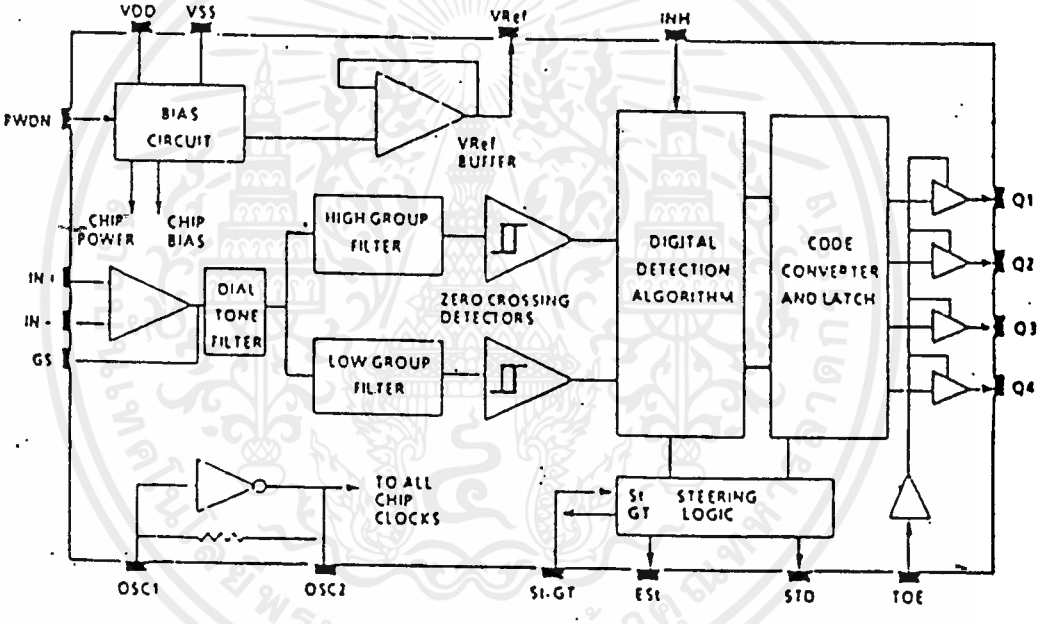
ค่าเลขฐานสองที่ได้จากการถอดรหัสความถี่ DTMF

ความถี่ต่ำ	ความถี่สูง	หมายเลข	Q_4	Q_3	Q_2	Q_1
697	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1477	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1477	9	1	0	0	1
941	1209	0	1	0	1	0
941	1336	*	1	0	1	1
941	1477	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

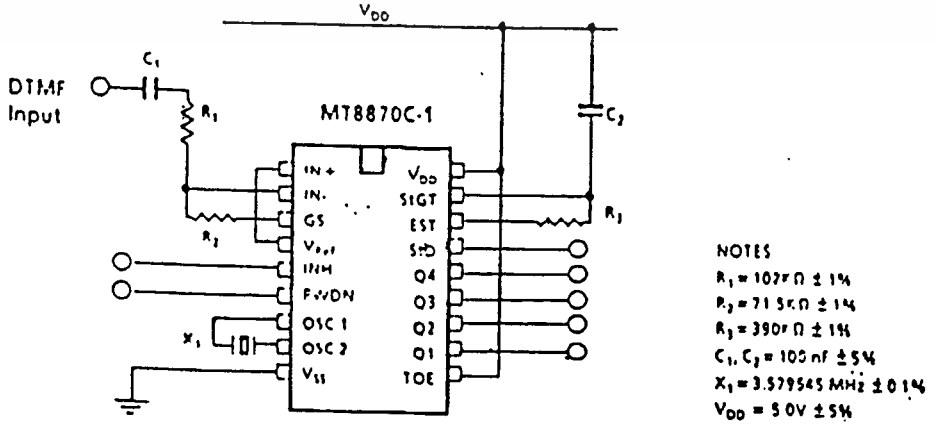
การถอกรหัสความถี่ DTMF ในยุคก่อนจะใช้ซี จำพวกเฟสลอกลูปซึ่งก่อปัญหามากมาย เนื่องจากทำให้ความถี่เปลี่ยนไป ตัววงจรมีขนาดใหญ่เพราะให้ไอซีจำนวนมาก ซึ่งในปัจจุบันนี้การถอดรหัสความถี่ DTMF สามารถทำได้สะดวกโดยใช้ไอซี MT8870 ซึ่งเป็นไอซีถอดรหัสความถี่ DTMFโดยตรง

โครงสร้างภายในของ MT8870

โครงสร้างภายในของ MT8870 ประกอบด้วยวงจรกรองความถี่และวงจรถอดรหัสฟังก์ชันทางดิจิตอล ในส่วนของวงจรกรองความถี่ ใช้เทคนิคของสวิทช์คาปาซิเตอร์ฟิลเตอร์กรองความถี่สูงและต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอลเพื่อตรวจจับและถอดรหัสใช้เทคนิคการนับทางดิจิตอล เพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ ออกเป็นเลขฐานสองขนาด 14 บิท และเช็คช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุทเป็นออปแอมป์ ซึ่งสามารถปรับอัตราขยายได้โดยต่าอุปกรณ์ภายนอก ภายเอาท์พุทเป็นวงจรแลทช์ 3 สถานะ โครงสร้างภายในของ MT8870 แสดงดังรูปที่ 3



รูปที่ 3 โครงสร้างภายในของ MT8870



รูปที่ 4 การต่อใช้งานเบื้องต้นของ MT8870

ฟังก์ชันการทำงานภายในของ MT8870

ภายใน MT8870 ประกอบด้วยส่วนสำคัญ 5 ส่วนคือ

1. ภาคกรองความถี่ (filter section)
2. ภาคถอดรหัส (decoder section)
3. ภาคตรวจสอบสัญญาณ (steering section)
4. ภาคขยายสัญญาณความแตกต่าง (differential section)
5. ภาคกำเนิดความถี่ (oscillator)

ภาคกรองความถี่

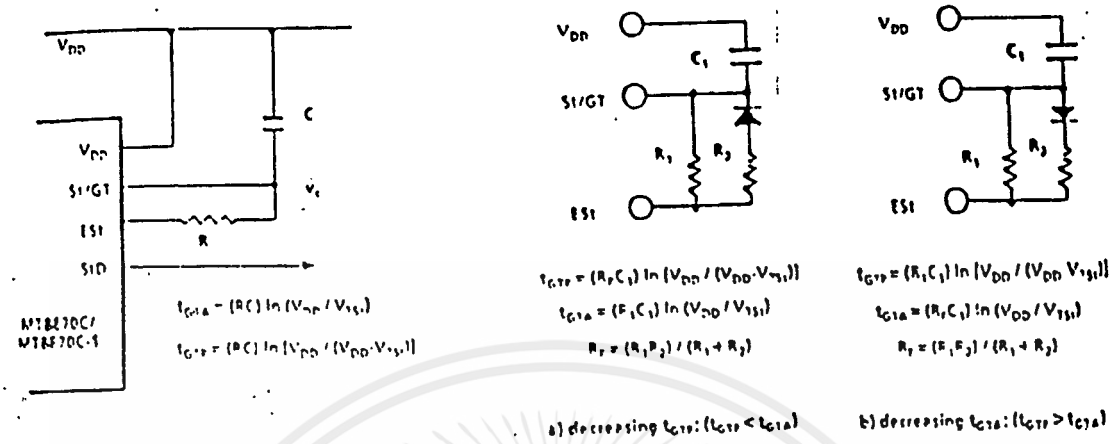
ในส่วนนี้จะแยกสัญญาณ DTMF ที่เข้ามาออกเป็น 2 กลุ่มความถี่ คือ ช่วงความถี่สูง และช่วงความถี่ต่ำ โดยใช้วงจรกรองความถี่อันดับ 6 ชนิดสวิทช์ คาปาซิเตอร์ (six-order swtiched capacitor band pass filter)

ภาคถอดรหัส

ความถี่ DTMF ที่ถูกกรองเรียบร้อยแล้ว จะผ่านเข้าวงจรถอดรหัสความถี่ออกเป็นตัวเลขโดยใช้เทคนิคการนับแบบดิจิตอล และจะมีการตรวจสอบความถี่ที่เข้ามาว่าเป็นความถี่มาตรฐาน DTMF หรือไม่ เพื่อป้องกันความถี่อื่นเข้ามาผสมเมื่อตรวจสอบแล้วว่าความถี่นั้นถูกต้อง สัญญาณที่ขาES (early sterring) ก็จะแอคตีฟ โดยค่าที่ถอดรหัสได้จากความถี่ต่าง ๆ แสดงไว้ในตาราง

ภาคตรวจสอบสัญญาณ

ก่อนที่จะมีการถอดระหัสความถี่ออกไปที่เอาท์พุท จะต้องมีการตรวจสอบช่วงความถี่ที่เข้ามาว่า มีระยะเวลาตามที่กำหนดหรือไม่ โดยสังเกตุจากระยะเวลาการกดปุ่มโทรศัพท์ซึ่งต้องกดปุ่มให้มีความถี่ออกมาเป็นช่วงเวลานานพอสมควรมิฉะนั้นวงจรส่วนนี้จะไม่รับ โดยถือว่าสัญญาณนั้นไม่ถูกต้อง ส่วนช่วงเวลานานเท่าใด สามารถตั้งได้โดยใช้ RC ต่อภายนอก ขา ES จะเป็น "high" นานใกล้เคียงกับระยะเวลาที่มีความถี่ DTMF เข้ามา จากรูปเมื่อขา ES เป็น "high" ทำให้ V สูงขึ้น ตัวเก็บประจุ C จะคายประจุ เพราะทำให้แรงดัน V สูงขึ้นจนถึงค่าเทรชโฮลด์ วงจรถอดรหัสจึงจะถอดรหัสออกเป็นตัวเลขฐานสอง ขนาด 4 บิท



รูปที่ 5 วงจรตรวจสอบสัญญาณอย่างง่าย และการคำนวณการ์ดไทม์

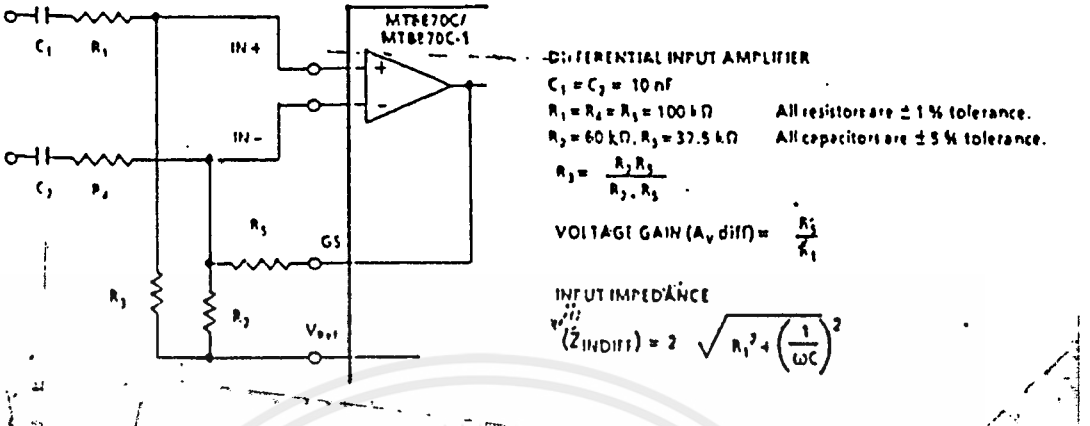
สำหรับช่วงเวลาที่ตั้งไว้โดยการต่อ RC ภายนอกนี้ เรียกว่า การ์ดไทม์ (gard time) ซึ่งเป็นตัวกำหนดช่วงเวลาในการกดปุ่ม โดยในการกดปุ่มแต่ละครั้งจะต้องกดอยู่นานอย่างน้อยเท่ากับการ์ดไทม์ จึงจะทำการถอดรหัสได้

ภาคขยายสัญญาณความแตกต่าง

วงจรส่วนอินพุทของ MT8870 เป็นภาคขยายออปแอมป์ที่สามารถปรับอัตราขยายโดยต่อวงจรภายนอกเพิ่มเข้าไปดังรูปที่6 แสดงการต่อวงจรภายนอกเข้ากับอินพุท ซึ่งสามารถคำนวนอัตราขยายความแตกต่างอินพุท และอิมพีแดนซ์ ได้ดังนี้

อัตราขยาย (A_v diff) $= R_s/R_i$

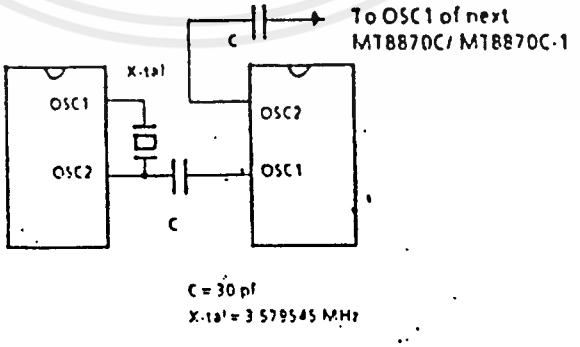
อินพุทอิมพีแดนซ์ (Z_{in} diff) $= 2\sqrt{R_i^2 + (1/wc)^2}$



รูปที่ 6 การต่อวงจรภาคอินพุท

ภาคกำเนิดความถี่

ในภาคกำเนิดความถี่นี้ ภายในไอซีจะมีวงจรเวลาอยู่เพียงแต่แร่คริสตอลขนาด 3.579545 เมกะเฮิร์ทช ก็สามารถใช้งานได้ทันที การต่อวงจรกำเนิดความถี่แสดงไว้ในรูปที่ 7



รูปที่ 7 การต่อวงจรผลิตความถี่

วงจรนาฬิกา

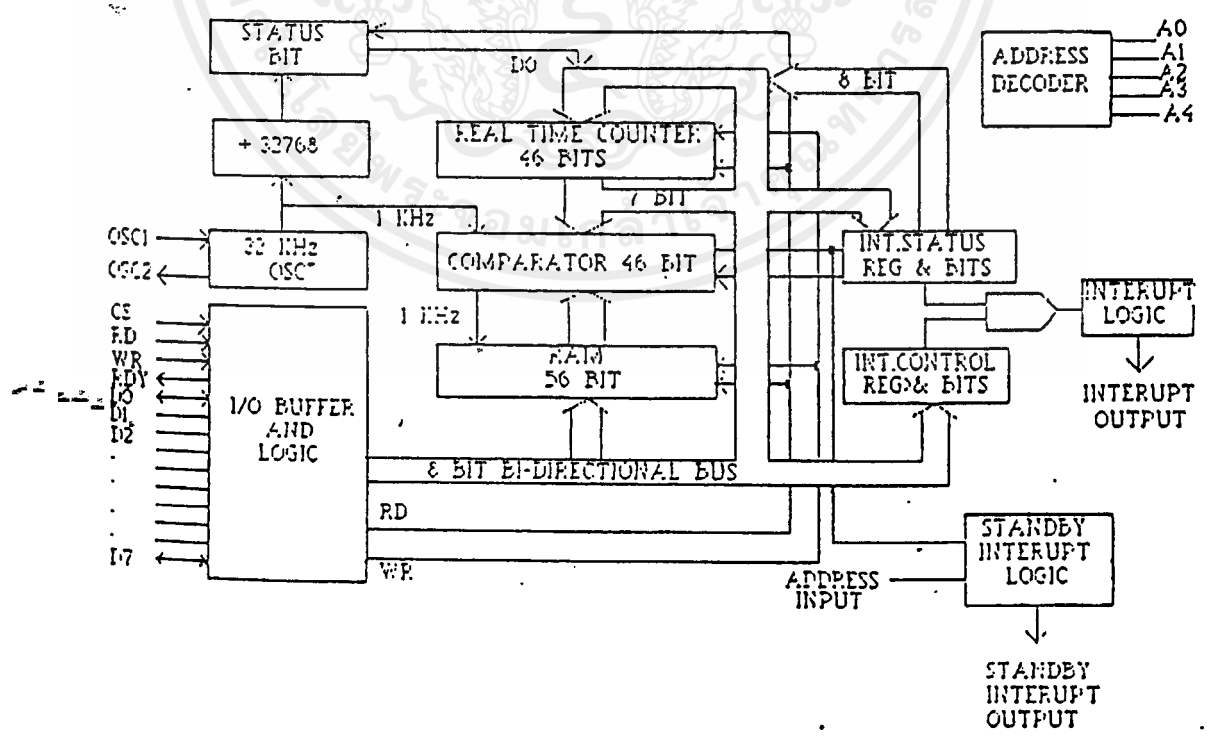
(Real Time Clock , RTC)

การคิดค้นเครื่องมือบอกเวลาได้มีการพัฒนาขึ้นเรื่อยๆ จนถึงการคิดค้นนาฬิกาที่สามารถบอกทั้งเวลา วัน เดือน ปี ซึ่งเรียกว่า วงจรนาฬิกา (Real Timd Clock) โดยอยู่ในรูปของชิพไอซี ซึ่งใช้งานร่วมกับไมโครโปรเซสเซอร์

RTC มีอยู่ด้วยกันหลายเบอร์ แต่บางเบอร์ก็มีปัญหาเกี่ยวกับข้อมูลของเวลาผิดพลาดเมื่อทำการปิดไฟของระบบที่ทำงานร่วมกันอยู่จากการทดลอง MM58167 เป็น RTC ที่ดี ข้อมูลไม่สูญหายเมื่อหยุดจ่ายไฟแก่ระบบเพราะทั้งนี้มีแบตเตอรี่สำรอง

MM58167

ชิพ MM58167 นี้เปรียบเสมือนพอร์ทอินพุท/เอาท์พุท ซึ่งสามารถอ่านและเขียนไปที่พอร์ทนั้นได้ ดูจากบล็อกไดอะแกรมในรูปที่ 8 จะเห็นว่ามีส่วนประกอบที่สำคัญคือ ตัวกำเนิดสัญญาณนาฬิกา (oscillator) ตัวนับเวลา (real time counter) ตัวเปรียบเทียบ (comparator) และ RAM



รูปที่ 8 บล็อกไดอะแกรมของ MM58167

การใช้งานนั้นก็เพียงแต่เขียนข้อมูลแบบ BCD ไปที่ตัวนับเวลา ตามแอดเดรสของเคาน์เตอร์แต่ละตัว เช่าของวัน ชั่วโมง หรือนาที ตัวนับเวลาก็จะเริ่มนับเวลาที่ตั้งให้และในทางตรงกันข้ามเราก็สามารถอ่านค่าของเวลาจากตัวนับเวลาได้เช่นกัน นอกจากนั้นผู้ใช้ยังสามารถตั้งเวลาในการให้สัญญาณเตือนได้ โดยสามารถตั้งให้ RTC ให้สัญญาณอินเทอร์รัพท์เอาท์พุท ในวันเวลาใดก็ได้โดยที่ตั้งเวลาไปที่ RAM เมื่อเวลาของตัวนับเวลาเดินมาตรงเวลากับใน RAM จะทำให้เกิดการเปรียบเทียบขึ้นและให้สัญญาณอินเทอร์รัพท์เอาท์พุท จากรูปที่ 8 มีส่วนประกอบที่สำคัญ ที่จะกล่าวถึงมีดังนี้

<u>ตัวนับเวลา</u>

ตัวนับเวลา เป็นตัวนับและจัดการเกี่ยวกันเวลา ถูกแบ่งเป็นดิจิตละ 4 บิทการเข้าถึงตัวนับเวลาจะกระทำครั้งละสองดิจิต (ในขณะ READ และ WRITE) ซึ่งแต่ละดิจิตจะให้ค่า BCD ดังแสดงในตารางข้างล่าง บิทที่ไม่ใช่จะถูกโฮรด์โดยลอจิก " 0 " ซึ่งเราไม่ต้องสนใจในขณะที่ทำการเขียนข้อมูลลงบนบรัสข้อมูล เหตุที่บางบิตไม่ใช้ก็เนื่องจากว่าไม่จำเป็นต้องใช้การในข้อมูลแบบ BCD ของบางหลัก

<u>รายละเอียดของข้อมูลเป็นดิจิตในแต่ละเคาน์เตอร์แอดเดรส</u>

เคาน์เตอร์ แอดเดรส	หลักหน่วย $D_0\ D_1\ D_2\ D_3$	BCD โคด มากสุด	หลักสิบ $D_4\ D_5\ D_6\ D_7$	BCD โคด มากสุด
1/1000 วินาที (00H)	- - - -		$D_4\ D_5\ D_6\ D_7$	9
1/100 และ 1/10 วินาที (01H)	$D_0\ D_1\ D_2\ D_3$	9	$D_4\ D_5\ D_6\ D_7$	9
วินาที (02H)	$D_0\ D_1\ D_2\ D_3$	9	$D_4\ D_5\ D_6$ -	5
นาที (03H)	$D_0\ D_1\ D_2\ D_3$	9	$D_4\ D_5\ D_6$ -	5
ชั่วโมง (04H)	$D_0\ D_1\ D_2\ D_3$	9	$D_4\ D_5$ - -	2
วันในสัปดาห์ (05H)	$D_0\ D_1\ D_2$ -	7	- - - -	0
วันที่ (06H)	$D_0\ D_1\ D_2\ D_3$	9	$D_4\ D_5$ - -	3
เดือน (07H)	$D_0\ D_1\ D_2\ D_3$	9	D_4 - - -	1

RAM

MM88167 มี RAM ขนาด 16 บิท ซึ่งใช้ในการเก็บข้อมูลเมื่อไฟดับหรือใช้เก็บข้อมูลการตั้งปลุกเพื่อที่จะเปรียบเทียบกับตัวนับเวลา ข้อมูลใน RAM จะสามารถเปรียบเทียบกับตัวนับเวลา จะมีดิจิตที่ไม่ได้ใช้คือ หลักหน่วยของ1/1000 ของ วินาที และหลักสิบของวันในสัปดาห์ (ไม่ใช้ในตัวนับเวลาดูตารางประกอบ)

RAM จะถูกกำหนดให้มีรูปแบบเหมือนกับตัวนับเวลาอย่างไรก็ตามก็ยังมีบิทที่ยังไม่ได้ใช้อยู่ซึ่งบิตที่ไม่ใช้ในตัวนับเวลานี้ จะเปรียบเทียบกับ " 0 " ใน RAM

อินเทอร์รัพท์และตัวเปรียบเทียบ

มีสัญญาณอินเทอร์รัพท์อยู่สองอย่าง อย่างแรกคืออินเทอร์รัพท์เอาท์พุท(แอคตีฟ "1") เอาท์พุทนี้สามารถโปรแกรมให้เกิดสัญญาณทางออกได้ 8 อย่าง คือ 10 เฮิร์ทช,1 เฮิร์ทช ,1 นาทีต่อครั้ง , 1ชั่วโมงต่อครั้ง , 1 วันต่อครั้ง, 1 สัปดาห์ต่อครั้ง, 1 เดือนต่อครั้ง เมื่อรวมกับตัวนับเวลาจะเกิดการเปรียบเทียบขึ้น

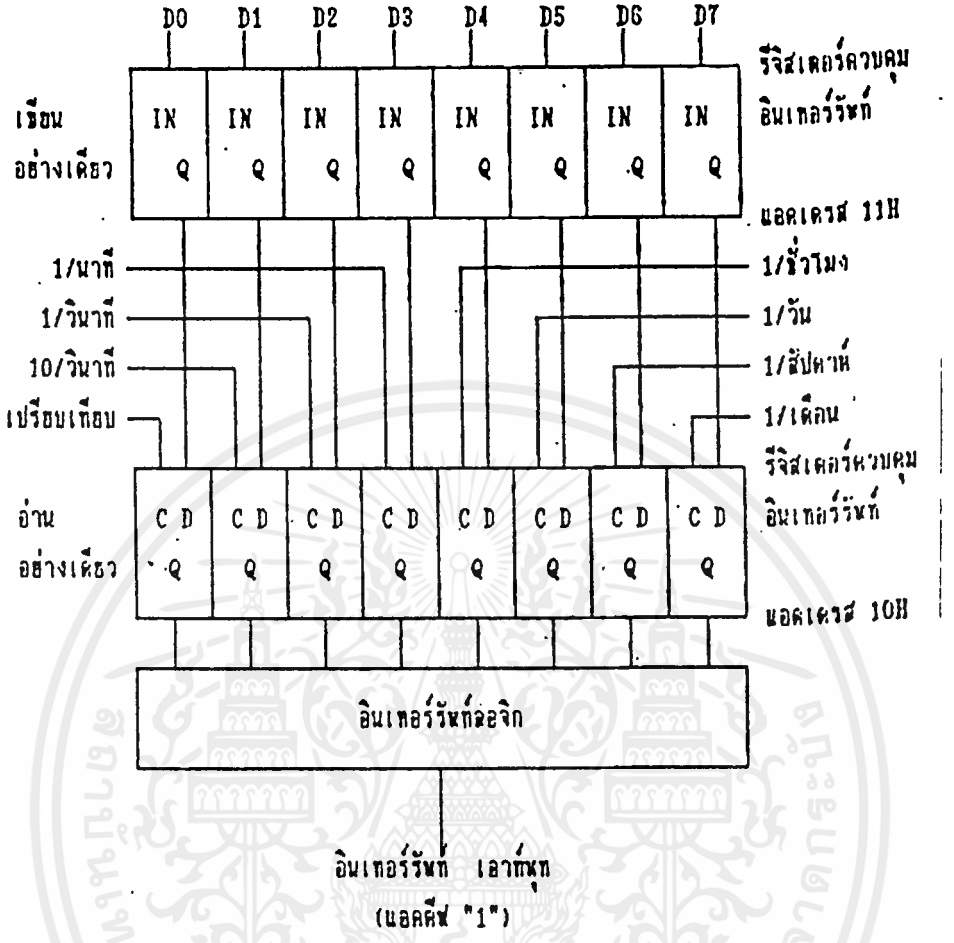
วิธีการที่จะอีนาเบิลสัญญาณอินเทอร์รัพท์ คือ ให้ลอจิก " 1 " แก่รีจิสเตอร์ควบคุมการอินเทอร์รัพท์ (interrupt control register)ในบิที่ตรงกับความถี่ที่เราต้องการจะให้เกิดสัญญาณอินเทอร์รัพท์ ดูรูปประกอบเช่น ต้องการให้สัญญาณ อินเทอร์รัพท์ทุก ๆ 1 นาที ให้ D_2 เป็น"1" เขียนไปที่ รีจิสเตอร์ควบคุมการอินเทอร์รัพท์ โดยดูเป็นตาราง แอทเดรส ได้จากตาราง

เราสามารถตรีจิสเตอร์ ควบคุมอินเทอร์รัพท์ ทุกวินาที และทุก ๆ ชั่วโมง เซต บิทที่ 3 (D_3)กับบิทที่ 2 (D_2) ในที่นี้คือ OCH โดยเขียนไปที่ แอทเดรสของรีจิสเทอร์ควบคุมอินเทอร์รัพท์

เมื่อเวลานับมาถึงค่าสูงสุดของแต่ละค่าจะทำให้เกิดคล็อคให้กับรีจิสเทอร์ควบคุมอิน-เทอร์รัพท์ ซึ่งจะทำให้อินเทอร์ลัพท์เอาท์พุทเป็น "1"(บิทใดบิทหนึ่งต้องถูกอีนาเบิลด้วย) การอ่านรีจิสเตอร์ควบคุมอินเทอร์รัพท์ ทำให้เราทราบว่าสัญญาณอินเทอร์ลัพเป็นสัญญาณของบิทใด อีกทั้งยังเป็นรีเซตรีจิสเตอร์ควบคุมอินเทอร์รัพท์อีกด้วย

แอดเดรสของฟังชั่นต่าง ๆ

A_4	A_3	A_2	A_1	A_0	ฟังก์ชัน
0	0	0	0	0	นับ 1/1000 วินาที
0	0	0	0	1	นับ 1/100 และ 1/10 วินาที
0	0	0	1	0	นับวินาที
0	0	0	1	1	นับนาที
0	0	1	0	0	นับชั่วโมง
0	0	1	0	1	นับวันในสัปดาห์
0	0	1	1	0	นับวันที่
0	0	1	1	1	นับเดือน
0	1	0	0	0	RAM 1/1000 วินาที
0	1	0	0	1	RAM 1/100 และ 1/10 วินาที
0	1	0	1	0	RAM วินาที
0	1	0	1	1	RAM นาที
0	1	1	0	0	RAM ชั่วโมง
0	1	1	0	1	RAM วันในสัปดาห์
0	1	1	1	0	RAM วันที่
0	1	1	1	1	RAM เดือน
1	0	0	0	0	รีจิสเตอร์สถานะอินเทอร์รัปท์
1	0	0	0	1	รีจิสเตอร์ควบคุมอินเทอร์รัปท์
1	0	0	1	0	รีเซตตัวนับเวลา
1	0	0	1	1	รีเซต RAM
1	0	1	0	0	สถานะบิท
1	0	1	0	1	ค่าสั่ง "GO"
1	0	1	1	0	สแตนด์บายอินเทอร์รัปท์
1	0	1	1	1	โหมดทดสอบ



รูปที่ 9 การอีนาเบิลสัญญาณอินเทอร์รัพท์ต่าง ๆ

การอ่านรีจิเตอร์ควบคุมอินเทอร์รัพท์นี้จะได้ข้อมูลบนบัสข้อมูล ซึ่งประกอบด้วยบิทที่ทำให้เกิดการอินเทอร์รัพท์ โดยจะให้ค่าเป็น "1" ที่บิทนั้น (ดูรูปที่ 9 ประกอบ)หลังจากรอบของการอ่านแล้วจะทำให้รีจิสเตอร์ควบคุมอินเทอร์รัพท์ถูกรีเซท

อินเทอร์รัพท์อีกอย่างหนึ่ง คือ สแตนด์บายอินเทอร์รัพท์ (เอาท์พุทเป็นแบบ โอเพนเดรนแอคทีฟ " o ") อินเทอร์รัพท์ตัวนี้จะเกิดขึ้นเมื่อเราได้ทำการอีนาเบิล ไว้และเกิดการเปรียบเทียบใน RAM กับตัวนับเวลาการอีนาเบิลทำได้โดยเขียน 01H ไปที่แอดเดรส 16H และในทางตรงกันข้ามถ้าให้ 00H ที่แอดเดรส 16H จะเป็นการดิสเอเบิล

<u>เพาเวอร์ดาวน์โหมด</u>

ขาพาวเวอร์ดาวน์ (power down) เป็นตัวเลือก ชิพที่สำคัญที่สอง มันจะดิส

เอเบิลสัญญาณออกมาทั้งหมด ยกเว้นสัญญาณสแตนด์บายอินเทอร์รัพท์เมื่อขานี้ได้รับลอจิก"0" MM58167 จะไม่ตอบสนองแก่สัญญาณจากภายนอกแต่นาฬิกายังเดินตามปกติและยังให้สัญญาณสแตนด์บายอินเทอร์รัพท์ (ขา14) ถ้าได้มีการโปรแกรมให้ขานี้ทำงานไว้ก่อนแล้ว

เมื่อต้องการเปลี่ยนจากโหมดการทำงานปกติมาเป็นสแตนด์บายโหมด ควรจะให้ขาเพาเวอร์ดาวน์เป็นลอจิก "0" อย่างน้อยที่สุด 10^{-6} วินาที ก่อนที่จะทำการลดระดับลงมาเป็นสแตนด์บายโหมด

เมื่อต้องการเปลี่ยนกับเข้าสู่การทำงานปกติ ผู้ใช้ต้องมั่นใจว่าขาอินพุทอื่น ๆต้องเป็นสัญญาณที่ถูกต้อง ก่อนที่จะกลับมาสู่โหมดการทำงานปกติ ทั้งนี้เพื่อป้องกันข้อมูลของนาฬิกาเสียไป ซึ่งจะทำให้นาฬิกาเดินผิด ตัวอย่างนี้ได้แก่การที่ขา CS,RD,WR ของ MM58167 มีสัญญาณเปลี่ยนแปลงในขณะที่กลับสู่โหมดปกติจะทำให้มีการเขียนข้อมูลไปที่ตัวนับเวลาหรือ-ใน RAM

ตัวนับเวลาและ RAM รีเซต : คำสั่ง GO

ตัวนับเวลา (CUONTER)และ RAM สามารถรีเซตได้โดยเขียน FFH ที่แอดเดรส 12H,13H ตามลำดับ การให้พัลส์ของการเขียนไปที่แอดเดรส 15H (คำสั่ง GO) จะรีเซทตัวนับของวินาที ขณะทำการเขียนไปที่แอดเดรส 15H นี้ MM58167 จะไม่สนใจข้อมูลบนบัสข้อมูลแต่ผลของ คำสั่ง GO มีดังนี้

ถ้าตัวนับของวินาทีนับได้มากกว่า 39 เมื่อเราใช้คำสั่ง GO (แอดเดรส 15H) จะทำให้หลักของนาทีเพิ่มขึ้นในกรณีอื่น ๆ จะไม่มีผลต่อหลักนาที

บิทสถานะ

บิทสถานะจะบอกผู้ใช้ว่าถ้าขณะที่ทำการอ่านตัวนับนั้น ตัวนับกำลังอยู่ในช่วงของการอัพเดท เวลา ข้อมูลที่อ่านได้ อาจมีการผิดพลาดเกิดขึ้น บิทสถานะนี้จะอ่านได้จากแอดเดรส 14H ของ RTC โดยจะให้ลอจิก"1" ที่บิท 0 ของบัสข้อมูลในขณะที่บิทอื่นๆ เป็น "0" หมดถ้าสัญญาณนี้ปรากฏขึ้นภายหลังการอ่านตัวนับควรมีการอ่านตัวนับใหม่ ที่ขอบขาลงของสัญญาณ READ ที่แอดเดรส 14H จะรีเซทบิทสถานะด้วย

<u>ตัวกำเนิดสัญญาณนาฬิกา</u>

เป็นตัวกำเนิดสัญญาณนาฬิกาแบบ เรโซแนนซ์ ขนาด โดยใช้อุปกรณ์ภายนอกเพียงตัวเก็บประจุ 1 ตัว ตัวต้านทาน 1 เมกะโอหม์ 1 ตัว แร่กำเนิดความถี่ 1 ตัว โดยตัวต้านทานจะต่ออยู่ระหว่างขั้ว OSC in (ขา10) และ OSC out(ขา11) เพื่อที่จะไปไบ-อัสตัวอินเวอร์เตอร์ที่อยู่ภายในให้ทำงานอยู่ในช่วงที่เป็นเชลล์เชสสำหรับแร่แบบไมโครเพาเวอร์คริสตรอลจะใช้ตัวต้านทานต่ออนุกรมกับขา OSC out โดยใช้ตัวต้านทานมีค่าโดยประมาณ 200 กิโลโอห์ม ส่วนตัวเก็ยประจุโดยปกติจะมีค่าอยู่ในช่วง 20-25 พิโคฟารัด แร่ที่ใช้มีความถี่ 32768 เฮิร์ทช

<u>คอนโทรลไทม์</u>

สัญญาณ Read write และ chip select เป็นสัญญาณอินพุท ทำงานที่ลอจิก "0" และสัญญาณ Ready เป็นสัญญาณออก(โอเพนเดรนเอาท์พุท) ที่จุดเริ่มต้นของการอ่านหรือการเขียนขา Ready จะให้สัญญาณเอาท์พุทเป็น " 0 " อยู่จนกระทั่งข้อมูลค่าปรากฏอยู่บนบัสข้อมูลเรียบร้อยหรือข้อมูลได้ถูกแลตซ์ไว้แล้วในขณะช่วงของการเขียน

<u>โหมดทดสอบ</u>

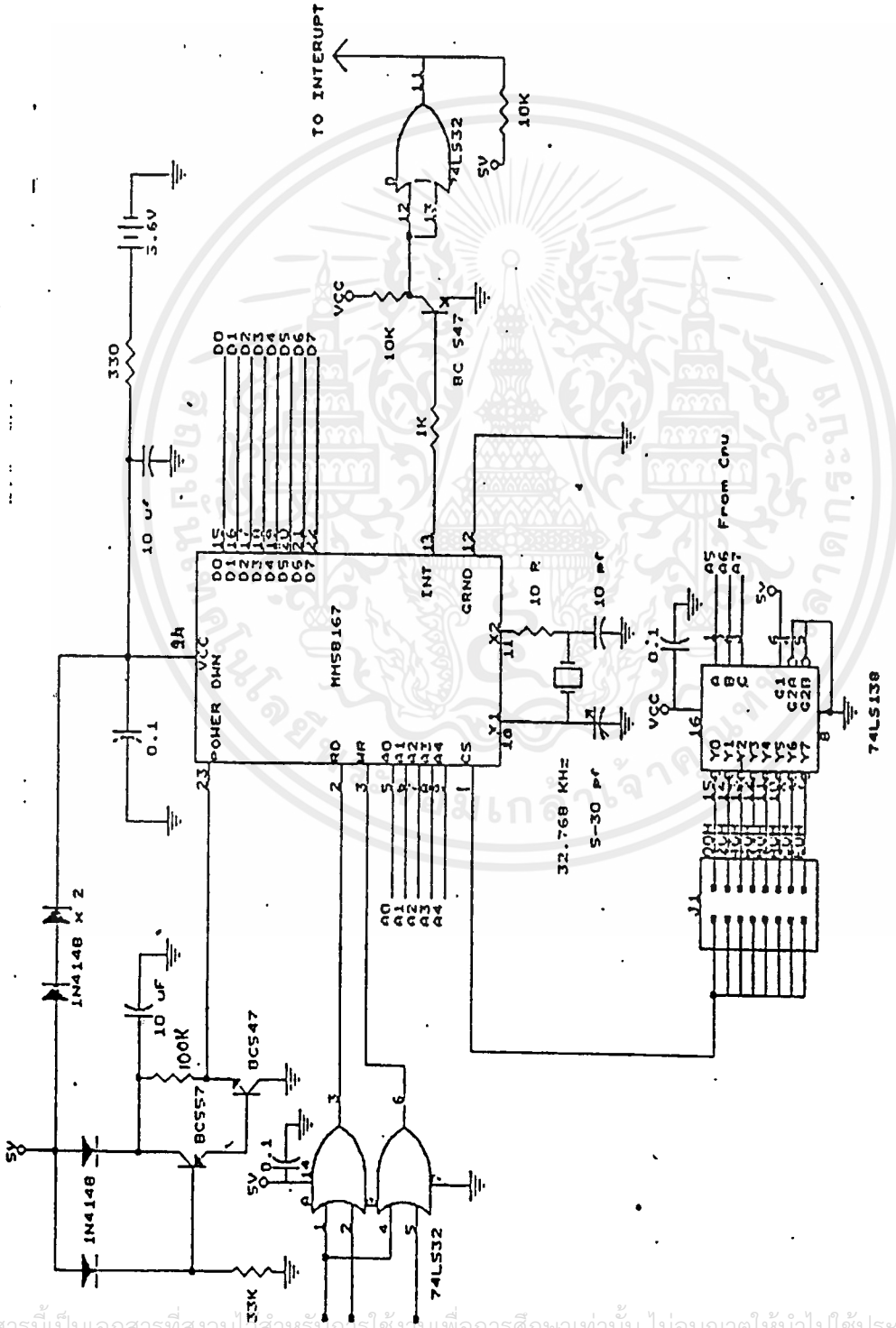
ในโหมดนี้ใช้เป็นเพียงการทดสอบ RTCชิพ ให้ทำงานที่ความถี่สูงกว่าการทำงานปกติ ในโหมดทดสอบนี้ความถี่ 32 กิโลเฮิร์ทชจะถูกต่อตรงเข้าที่ 1/1000 วินาทีขา CS และ WR ต้องเป็น" 0 "และให้แอดเดรส เป็น 1FH

<u>ลักษณะของบอร์ด RTC</u>

บอร์ด RTC มีขนาด 2.7 x2.2 นิ้ว มีคอนเน็กเตอร์ 40 ขา สำหรับเสียบลงบนบอร์ดควบคุมและมีคอนเน็กเตอร์สำรองอีกชุด เพื่อขยายระบบออกไปใช้งานรวมกับระบบอื่นได้ บนบอร์ดมีแบตเตอร์รี่ 3.6 โวลท์ ไฟกระแสตรงทำการแบคอัพข้อมูลของนาฬิกาในกรณีที่ไฟดับและยังทำให้นาฬิกาเดินตามปกติ ไม่มีผลกระทบเมื่อไฟดับ

การดีโคดพอร์ท

บนบอร์ด RTC ใช้ 74LS138 ดีโคดพอร์ทโดยอิสระและยังมีจัมพ์เปอร์ให้เลือกเบอร์ของพอร์ท ได้อีก 8 เบอร์ คือ 00H,20H,40H,60H,80H,A0H,C0H,และ E0H ผู้ใช้สามารถเลือกจัมพ์เปอร์ เพื่อเลือกเบอร์พอร์ทได้โดยอิสระ



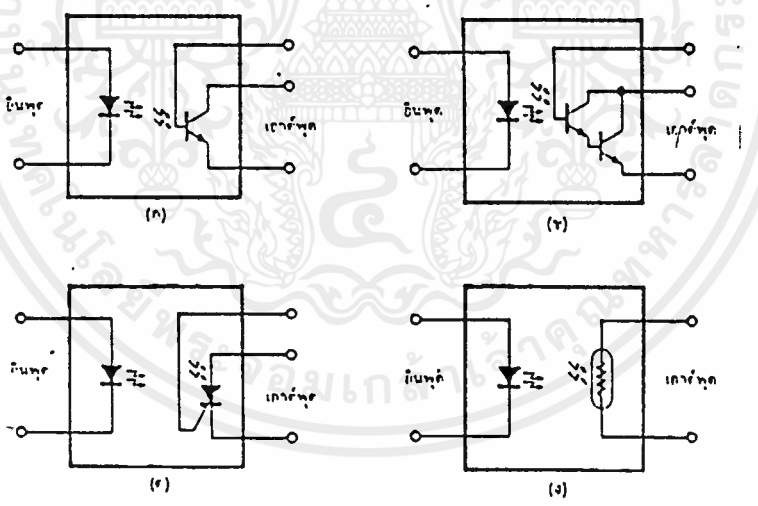
รูปที่ 10 วงจรภายในของ RTC

วงจรโซลิดสเตทรีเลย์

(Solid State Relay, SSR)

โซลิคสเตทรีเลย์ คือ อุปกรณ์ทางอิเลคทรอนิกส์อย่างหนึ่ง ที่ทำหน้าที่เหมือนกับรีเลย์ไฟฟ้ากลโดยปราศจากชิ้นส่วนไฟฟ้ากลที่เคลื่อนไหว โดยพื้นฐานแล้วโซลิดสเตทรีเลย์จะมีขั้วอินพุทและเอาท์พุทและเอาท์พุทอย่างละ 2 ขั้ว ขั้วอินพุทเป็นขั้วสำหรับป้อนสัญญาณควบคุมหรือสัญญาณกรุตุ้น เพื่อบังคับให้สวิทช์ทางด้านขั้วเอาท์พุทปิดหรือเปิดทั้งนี้จะมีการแยกทางไฟฟ้าระหว่างขั้วอินพุทกับขั้วเอาท์ พุทมักจะใช้การเชื่อมโยงด้วยแสง(optocoupling) ซึ่งต่างจากในรีเลย์ไฟฟ้ากลที่ใช้การเชื่อมโยงด้วยแม่เหล็กไฟฟ้า (electromagnetic coupling)

ในโซลิดสเตทรีเลย์ที่ใช้การเชื่อมโยงด้วยแสง วงจรทางด้านอินพุทจะเป็น LED ชนิดแกลเลียมอาร์เซไนด์ (GaAs) และตัวรับแสงด้านเอาท์พุทอาจเป็นโฟโต้าทรานซิสเตอร์ โฟโตเออาร์ โฟโตไตรแอก หรือ โฟโตรีซิสเตอร์ วงจรภายในของโซลิคสเตทรีเลย์ แสดงไว้ในรูปที่ 11



รูปที่ 11 ตัวรับแสงแบบต่างๆในโซลิดสเตทรีเลย์

ก) โฟโตทรานซิสเตอร์

ข) โฟโตดาลิงตันทรานซิสเตอร์

ค) โฟโตเอสซีอาร์

ง) โฟโตรีซิสเตอร์

ข้อดีของแผ่นวงจรโซลิดสเตทรีเลย์คือ

สามารถขับได้โดยตรงด้วยไอซีหลายตระกูล ซึ่งรวมทั้งไมโครโปรเซสเซอร์ด้วย

มีอายุการใช้งานยาวนานมาก

ปลอดภัยจากเสียงรบกวน

ไม่มีอาการ bounce เกิดขึ้นที่หน้าสัมผัส ซึ่งจะเป็นตัวก่อให้เกิดสัญญาณรบกวน

ไม่ก่อให้เกิดสัญญาณรบกวนต่อระบบไฟฟ้าที่ตัวมันต่ออยู่

ส่วนข้อเสียคือ

ไม่สามารถใช้กับวงจรที่มีโวลเตจสูงได้

มีการสูญเสียค่อนข้างสูง

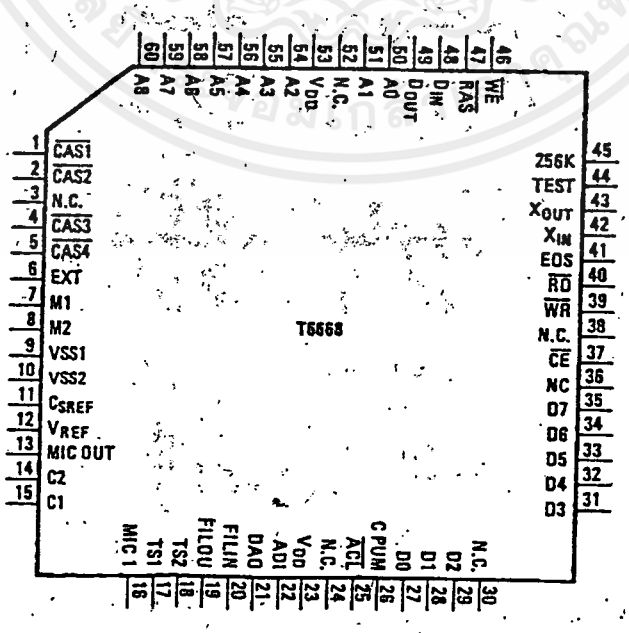
ต้องใช้แผ่นระบายความร้อน

มีกระแสรั่วไหลในภาวะหยุดนำกระแส

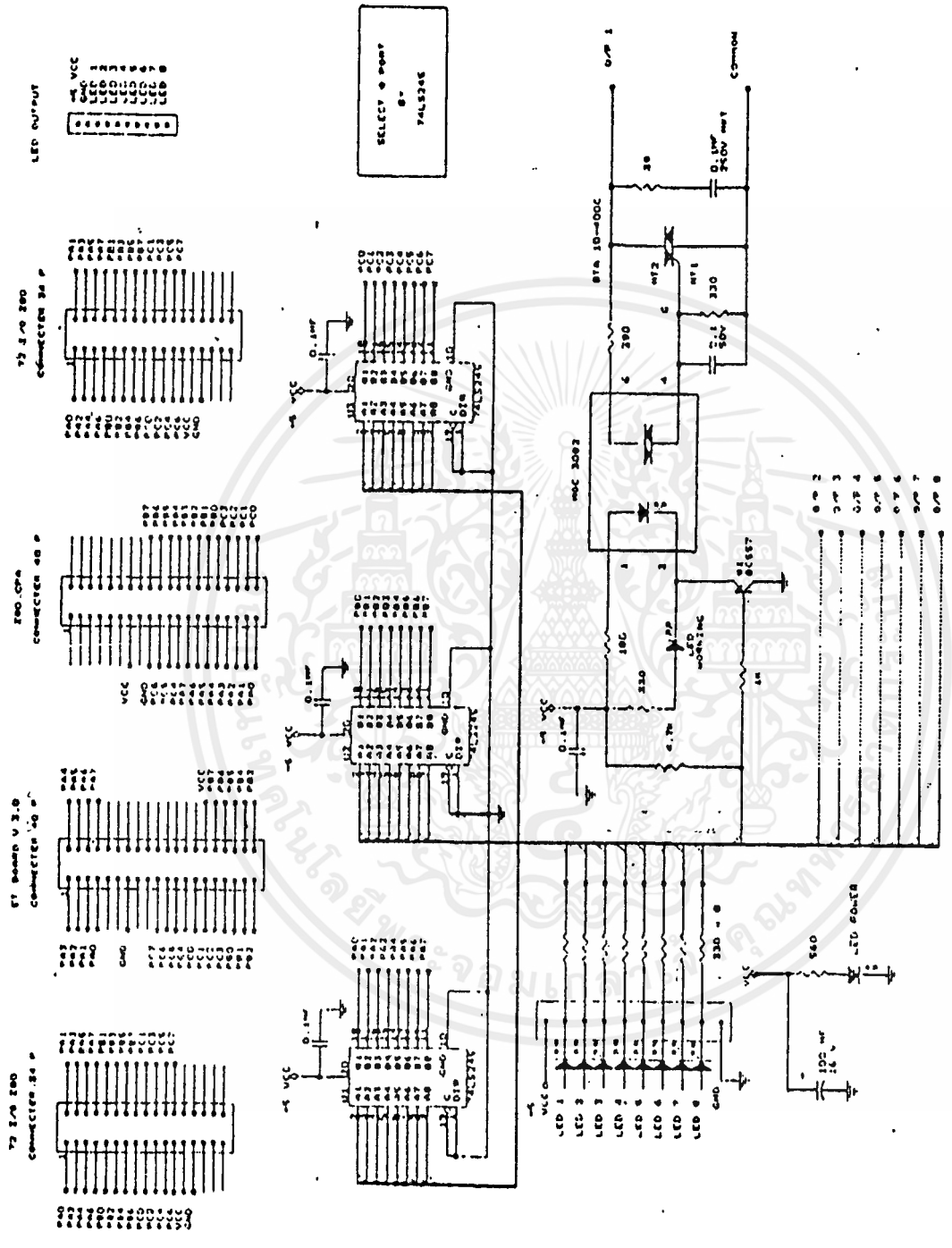
ตัวโซลิดสเตทรีเลย์จะเสียได้ง่าย ถ้าเกิดการรัดวงจรขึ้น

วงจรวิเคราะห์เสียงพูด

จากวงจรในรูป 2 เป็นวงจรสมบรูณ์ของชุดวิเคราะห์เสียงพูดหัวใจสำคัญของวงจรอยู่ที่ IC1 และIC2ซึ่งเป็นไอซีไมโครโปรเซสเซอร์และหน่วยความจำ ตัวIC1 เองถูกออกแบบขึ้นมาเพื่อใช้งานด้านวิเคราะห์เสียงพูดโดยเฉพาะ ซึ่งเป็นผลิตผลการประดิษฐ์ของบริษัทโตชิบาแห่งประเทศญี่ปุ่น เป็นไอซีชนิด CMOS LSI ลักษณะโครงสร้างภายนอกและตำแหน่งขาต่างๆแสดงไว้ในรูปที่ 12



รูปที่ 12 แสดงตำแหน่งขาต่างๆของไอซี T6668



วงจรภายในของแผ่นวงจรไฟฟ้าอิเล็กทรอนิกส์

MCS-51

คุณสมบัติของ MCS-51

- CPU 8 บิตที่ควบคุมได้ง่าย
- เพิ่มการทำงานลอจิกครั้งละ 1 บิตได้
- สายอินพุทและเอาท์พุทมีจำนวน 32 เส้น ใช้เลือกแอดเดรสแยกต่างหากจากกันได้
- มีแรมบรรจุไว้ภายในขนาด 128 ไบท์ หรือ 256 ไบท์
- วงจรตั้งเวลา วงจรนับมีขนาด 2 3 หรือ 16 บิต
- กำหนดเป็น UART(Universal Synchronous Asynchronous Receiver Transmitter ที่รับ

ส่งข้อมูลได้สองทิศทาง

- อินเทอร์รับแบ่งออกเป็น 2 ระดับ จาก 5 หรือ 6 แหล่ง
- มีสัญญาณนาฬิกาภายในตัว
- มีหน่วยความจำสำหรับเก็บข้อมูลภายในขนาด 4 หรือ 8 กิโลไบต์
- มีแอดเดรสของหน่วยความจำสำหรับเก็บโปรแกรมจำนวนทั้งหมด 64 กิโลไบต์
- มีแอดเดรสของหน่วยความจำสำหรับเก็บข้อมูลจำนวนทั้งหมด 64 กิโลไบต์
- คำสั่งทั้งหมดมี 111 คำสั่ง

  8051,8051AH,8052AH เป็นอุปกรณ์ที่ทำงานโดยการควบคุมจากโปรแกรมในหน่วยความจำ(รอม) โดยเหมาะกับการผลิตเพื่อใช้งานควบคุมจำนวนมากๆ (8051 มีหน่วยความจำ 4 กิโลไบต์ ส่วน 8052AH มีหน่วยความจำเพิ่มขึ้น)มีอีกสองเบอร์ที่ทำงานคล้ายกันคือ 8031AH และ 8032AH ที่ใช้แทน 8051AH และ 8052AH ตามลำดับ

ภายในหน่วยความจำ

หน่วยความจำแบ่งเป็น 2 กลุ่ม คือ หน่วยความจำสำหรับเก็บข้อมูล และหน่วยความจำสำหรับเก็บโปรแกรม mcs จะอ่านหน่วยความจำสำหรับเก็บโปรแกรมเข้ามาเป็นภาษาเครื่องตามลำดับ ส่วนหน่วยความจำสำหรับเก็บข้อมูลจะใช้เป็นที่เก็บตัวแปรการคำนวนหาผลลัพธ์ทันที

เบอร์	โปรแกรม	ข้อมูล	ตั้งเวลานับ	หมายเลข
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16 bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16 bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16 bit	5
8032AH		256 x 8 RAM	3 x 16 bit	6
8031AH		128 x 8 RAM	2 x 16 bit	5
8031		128 x 8 RAM	2 x 16 bit	5

รีจิสเตอร์ภายใน mcs

mcs มีรีจิสเตอร์ที่อำนวยความสะดวกในการใช้งานตามคำสั่งต่างๆประกอบด้วยแอคคิวมูเลเตอร์ รีจิสเตอร์B ที่ใช้ในการคูณและหาร รีจิสเตอร์สถานะ สแต็กพอยน์เตอร์ ดาต้าพอยน์เตอร์ พอร์ตหมายเลขศูนย์ถึงหมายเลขสาม รีจิสเตอร์แบบคู่ซึ่งใช้ส่งและรับข้อมูลชนิดอนุกรม รีจิสเตอร์ 16 บิตที่เป็นวงจรตั้งเวลาและวงจรนับ รีจิสเตอร์ซึ่งจองเอาไว้สำหรับใช้สำหรับนับตัวที่ 3 รีจิสเตอร์คำสั่งสำหรับหน้าที่พิเศษ

ความเร็วของ mcs เทียบกับ 6502 และ z80

คำสั่งทั้งหมดของ mcs ทำงานด้วยรอบคำสั่งเดียวของภาษาเครื่องคือภายในระยะเวลาของสัญญาณนาฬิกา 12 ลูก เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์ ดังนั้นใน 1 รอบคำสั่งภาษาเครื่องเท่ากับ 1 ไมโครวินาที ดังนั้นความเร็วของ mcs-51เท่ากับ 6502 ที่ใช้ความถี่ 2 เมกะเฮิรตซ์ หรือเท่ากับ z80 ที่ความถี่ 8 เมกะเฮิรตซ์ นอกจากนี้ mcs ยังมีความสามารถจัดการกับข้อมูลได้น้อยกว่า 8 บิตต่อ 1 ครั้ง เพื่อช่วยให้ความสะดวกในการเขียนโปรแกรมโดยไม่ต้องลำบากในการปิดกั้นบิตที่ไม่ต้องการ

วงจรนับ วงจรตั้งเวลา

เมื่อทำงานเป็นวงจรตั้งเวลา รีจิสเตอร์ที่ทำหน้าที่ตั้งเวลาจะเพิ่มค่าขึ้นหนึ่ง

ทุกๆรอบคำสั่งของเครื่อง และจะนับด้วยอัตราสูงสุดที่ 1/12 ของความเร็วสัญญาณนาฬิกาของโปรเซสเซอร์เมื่อทำงานเป็นวงจรนับ รีจิสเตอร์วงจรนับจะเพิ่มขึ้นหนึ่งเมื่อมีสัญญาณป้อนให้อินพุต t0 t1 หรือ t2(เฉพาะ 8052) เป็นขอบสัญญาณขาลง อัตราการนับสูงสุดคือ 1/24 ของความเร็วสัญญาณนาฬิกาของโปรเซสเซอร์ วงจรนับและวงจรตั้งเวลา 0 และ 1 มีวิธีโปรแกรมให้ทำงานได้ต่างกันถึง 4 แบบ รวมทั้งการทำงานเป็น 8 บิต หรือ 16 บิต และการบรรจุค่าพรีเซตหนึ่งค่าได้เองอย่างอัตโนมัติ วงจรนับและวงจรตั้งเวลาที่ 1 เลือกโปรแกรมให้ทำหน้าที่เป็นตัวกำเนิดสัญญาณของอัตราการส่งบิตออกไปอย่างอนุกรมสำหรับใช้ในการอินเตอร์เฟสได้ วงจรนับและวงจรตั้งเวลาที่ 2 (เฉพาะ 8052 เท่านั้น) มีการทำงานย่อยๆ อีก 3 ชนิด

1.วงจรนับ 16 บิตที่สามารถโหลดค่ากลับคืนเองอย่างอัตโนมัติ

2.วงจรนับที่จองไว้ขนาด 16 บิต

3.วงจรกำเนิดสัญญาณของการส่งบิตเพื่อใช้อินเตอร์เฟส

พอร์ตชนิดอนุกรมอยู่ภายใน

ไมโครคอนโทรลเลอร์ในตระกูลmcs-51 เป็นชิพที่มีอินเตอร์เฟสอนุกรมชนิด 2 ทาง ทำให้รับและส่งข้อมูลพร้อมกันตัวรับข้อมูลชนิดอะซิงโครนัส มีบัฟเฟอร์สำหรับข้อมูลเป็นพิเศษเพื่อเพิ่มความเร็วในการสื่อสาร พอร์ตชนิดอนุกรมนั้น เราสามารถเลือกโปรแกรมเพื่อเลือกใช้การทำงานแบบใดแบบหนึ่งใน 4 แบบด้วยการใช้โปรแกรมควบคุมอัตราการส่งข้อมูล

อัตราการส่งข้อมูลที่เลือกใช้ได้จะสูงถึง 19200 บิตต่อวินาที ด้วยความเร็วของสัญญาณนาฬิกา 1 เมกะเฮิรตซ์ สำหรับใช้ในระบบเครือข่าย และระบบการสื่อสารของโปรเซสเซอร์หลายตัวร่วมกัน เราจะเลือกความเร็วของสัญญาณนาฬิกาด้วยวงจรนับและวงจรตั้งเวลา

อินเตอร์รับและชุดคำสั่ง

8051 รับการอินเตอร์รับได้ 5 แหล่ง ส่วน 8052 รับได้ 6 แหล่ง คือขา int0 int1 วงจรนับและวงจรตั้งเวลาที่ 0 และ 1 (8052 เพิ่มวงจรนับและตั้งเวลาที่ 2) และจากพอร์ตอนุกรม

เราสามารถกำหนดลำดับความสำคัญของอินเตอร์รับได้ 2 ระดับ โดยไม่ต้องอาศัยวงจรภายนอกช่วย

คำสั่ง

ชุดคำสั่งของ mcs-51 แบ่งเป็น 5 กลุ่ม

- กลุ่มการถ่ายเทข้อมูล
- กลุ่มคณิตศาสตร์
- กลุ่มตรรกศาสตร์
- กลุ่มของบูลลีน
- กลุ่มของการกระโดด

กลุ่มการถ่ายเทข้อมูล

การถ่ายเทข้อมูลใน RAM ภายใน ได้แก่คำสั่ง mov push pop xch xchd

การถ่ายเทข้อมูลใน RAM ภายนอก และหน่วยความจำโปรแกรม ได้แก่คำสั่ง jmp call ret reti nop

กลุ่มคณิตศาสตร์ ได้แก่ inc dec add adc subb mul div da

กลุ่มตรรกศาสตร์ ได้แก่ cpl rl rlc rr rrc swap anl orl xrl

กลุ่มของบูลลีน

กลุ่มของการกระโดด

CP-32

CPU

1. 8052 AH BASIC
   1.1 ใช้ภาษาเบสิคใน 8052 AH BASIC
   1.2 ใช้ร่วมกับ CP32-BASIC (EPROM 27256)

2. 8032
   2.1 ใช่ร่วมกับ CP32-BASIC (EPROM 27256) พัฒนาระบบด้วยภาษาเบสิค
   2.2 เขียน MONITOR โปรแกรมควบคุมบอร์ดเองใช้เป็นคอนโทรล

3. 8051 , 8031 , 8751
   3.1 เขียน MONITOR โปรแกรมควบคุมบอร์ดเอง
   3.2 ใช้เป็นบอร์ดควบคุม

MEMOR

CP32 สามารถใส่หน่วยความจำได้สูงสุด 96KB โดยแบ่งเป็น

1. CODE MEMORY (U3)
   ขนาดหน่วยความจำ 32KB สัญญาณที่ใช้ในการติดต่อกับ U3 ใช้ PSEN ต่อเข้าที่ขา OE สำหรับการอ่านข้อมูล U3 ระบุให้เป็น ROM โดย DECODE หน่วนความจำไว้ที่ตำแหน่ง 0000-7FFFH

2. DATA MEMORY (U4)
   ขนาดหน่วนความจำ 8-32KB สัญญาณที่ใช้ในการติดต่อกับ U4 ให้ RD และ WR ต่อเข้ากับ RD และ WR ของหน่วนความจำตามลำดับ U4 นี้กำหนดให้เป็น RAM เลือกได้ 2 ขนาด คือ 8KB และ 32KB โดยการเลือกที่ JUMPER J2 (ดูการเซทุ DIP SW และ JUMPER) U4 ถูก DECODE ไว้ที่ตำแหน่ง 0000h- 7FFFh

3. CODE & DATA MEMORY (U5)
ขนาดหน่วนความจำ 8KB - 32KB เลือกใช้ได้ทั้ง ROM และ RAM สัญญาณที่ใช้ติดต่อกับ U5 คือ WR , RD , PSEN นั้น หมายความว่า U5 เป็นได้ทั้ง CODE MEMORY และ DATA MEMORY การเลือกขนาดของหน่วยความจำเลือกโดยการเซท DIP SW2 (ดูการเซท DIP SW และ JUMPER) U5 ถูก DECODE ไว้ที่ตำแหน่ง 8000h - FFFFh แต่หน่วยความจำช่วง E00h - FFFFh (8KB สุดท้าย) ถูกกำหนดให้เป็นแอดเดรสของพอร์ท จึงไม่สามารถใช้งานหน่วยความจำในช่วงนี้ได้

ส่วนประกอบบนบอร์ด

CONNECTOR

1. EXP1 เป็น INPUT/OUTPUT ของตัว CPU (ดูรายละเอียดขาได้จากวงจร)

2. EXP2 เป็นคอนเนคเตอร์ที่ใช้ขยายระบบโดยมีลักษณะขาเหมือนกับคอนเนคเตอร์ของ Z80 ยกเว้นสัญญาบางอย่างที่ไม่มี เช่น REFRESH BUSRQ , BUSAK, WAIT , ฯลฯ EXP1 นี้สามารถนำอุปกรณ์ต่าง ๆ ที่เคยใช้กับ ET-BOARD 72IO , RTC ฯลฯ มาใช้ได้ (ถ้าใช้บอร์ด 72IO ต้องเซทบอร์ด 72IO ให้เป็น POWRE ON RESET)

3. I/O CONNECTOR เป็น I/O ของพอร์ต 8255 โดยมีลักษณะขาเหมือนกับขาอินพุท/เอ๊าท์พุทของ 72IO สามารถนำสัญญาณไปใช้ในการควบคุมได้ สามารถต่อกับบอร์ดสนับสนุน SOLID STARE RELAY (SSRAC)

4. RS232 ใช้ต่อกับ SERIAL PORT (RS232) ของเครื่อง PC เพื่อทำการพัฒนาโปรแกรมโดยใช้ PC เป็น TERMINAL

JUMPER

J1 EX/IN ใช้สำหรับเลือกว่าจะใช้โปรแกรมภายใน CPU (8052 , 8751)หรือภายนอก โดยถ้าใส่ JUMPER จะเป็นการใช้โปรแกรมจากภายนอก

J2 เลือกหน่วนความจำ U4 สามารถเลือกได้ 2 ขนาดคือ 8KB และ 32KB ในกรณีที่เลือก U4 มีขนาด 32K ให้ดูคำเตือนในคู่มือ CP32-BASIC (เมื่อใช้ร่วมกับ CP32- BASIC)

SWITCH

SW1 RESER SW เป็นปุ่มรีเซททั้ง CPU และ พอร์ต 8255 ที่ขา RESE นี้ถูกต่อไปที่ I/O XONNECTOR และ EXP2 ด้วย

DIP SW2 เป็นการเลือกหน่วยความจำ U5 ซึ่งสามารถใช้ได้ทั้ง ROM , RAM , EPROM, E^2 PROM (ดูการเซท DIP Sw และ JUMPER)

SW3 TURN ON เมื่อ U5 เป็น EPROM และ TURN OFF เมื่อ U5 เป็น RAM

JACK

DC JACK เป็น JACK สำหรับไฟ DC+10V

VPROG เป็นขั้วสำหรับจ่ายไฟ VPP ในการเขียน EPROM

LED

LED POWER ใช้แสดงว่ามีการจ่ายไฟเข้าระบบ

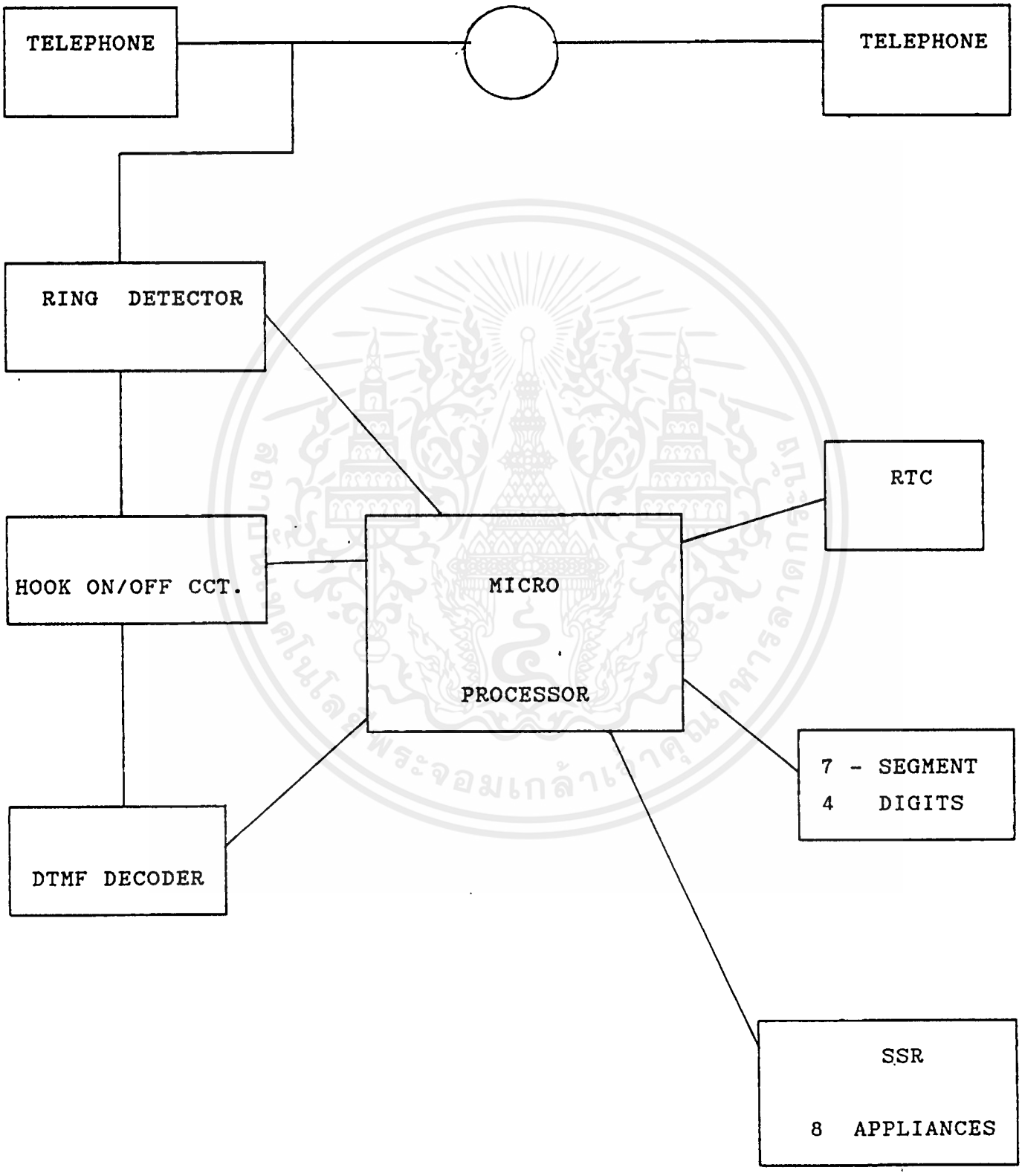
LED BURN จะติดเมื่อเซท DIP SW2 จุดที่ 8 ให้ ON เพื่อทำการเขียน EPROM (ใช้ CPU 8052 AH BASIC)

X'TAL

เป็นคริสตอลที่จ่ายฐานเวลาให้กับ CPU เป็นCRYSTAL ขนาด 11.0592 MHz

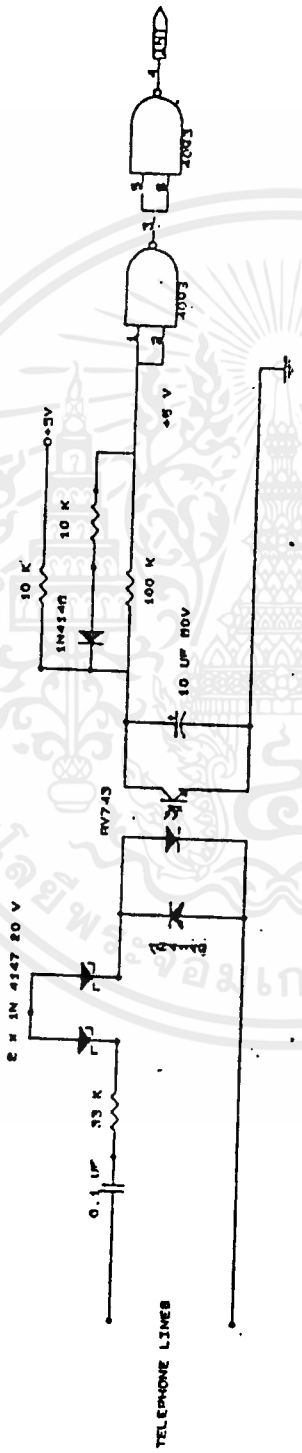
บทที่ 2

หลักการทำงานของส่วนต่าง ๆ ของเครื่องควบคุมอุปกรณ์ทางโทรศัพท์



บล็อคไดอแกรมของเครื่องควบคุมอุปกรณ์ทางโทรศัพท์

วงจร RING DETECTOR



รูปที่ 2.5 วงจรตรวจสอบสัญญาณกริ่ง

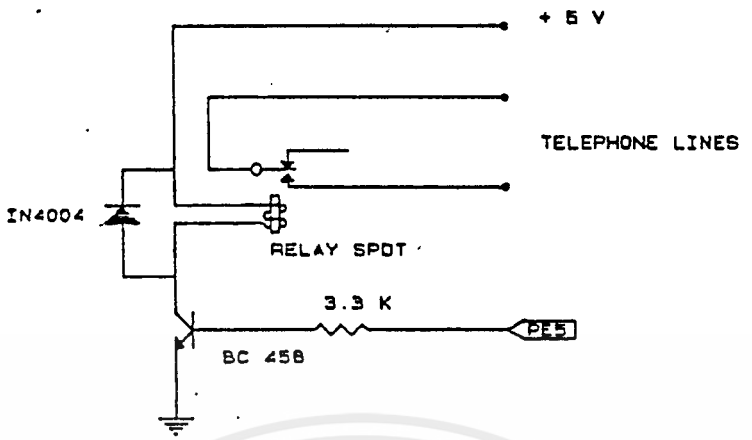
หน้าที่การทำงาน

ตรวจสอบว่ามีสัญญาณกริ่งเข้ามาหรือไม่ โดยถ้ามีสัญญาณกริ่งซึ่งเป็นสัญญาณไฟสลับระดับ 100 โวลท์ เรียกเข้ามาที่คู่สายโทรศัพท์ที่มีเครื่องควบคุมอุปกรณ์ทางโทรศัพท์ต่อพ่วงอยู่ วงจรตรวจสอบสัญญาณกริ่งก็จะทำงาน โดยจะแปลงสัญญาณกริ่งให้เป็นสัญญาณพัลส์ที่อยู่ในช่วง 0-5 โวลท์ แล้วนำสัญญาณพัลส์นี้ไปเชื่อมต่อกับขาอินเทอรัพท์INT ของไมโครโปรเซสเซอร์

หลักการทำงาน

เมื่อมีสัญญาณกริ่งซึ่งเป็นสัญญาณไฟสลับขนาด 100 โวลท์ 25 เฮิรท์ซคร่อมอยู่บนสัญญาณไฟตรง 48 โวลท์ ผ่านส่วนแรกของวงจรตรวจสอบสัญญาณกริ่งซึ่งมีตัวเก็บประจุต่ออยู่นั้น จะทำให้สัญญาณไฟตรงไม่สามารถที่จะผ่านไปได้ และมีตัวต้านทานเป็นตัวจำกัดกระแสที่ป้อนเข้ามา ส่วนซีเนอร์ไดโอด 2 ตัวที่ต่อติดกันนั้นจะทำหน้าที่ลดแรงดันไฟฟ้าที่คร่อมตัวมัน ให้มีค่าเท่ากับแรงดันไบอัสย้อนกลับสูงสุดของซีเนอร์ไดโอดเอง สัญญาณที่ได้ออกมาจะมีทั้งด้านบวกและลบ แรงดันไฟฟ้าที่ได้จะต่อเข้ากับขา 1 และ 2 ของออปโตคัปเปิล โดยมีไดโอดต่อคร่อมแบบย้อนกลับไว้ที่ขา 1 และ 2 ส่วนเอาท์พุทของออปโตคัปเปิลนั้นจะต่อคร่อมเข้ากับวงจรดังรูปที่ 2.5 ในกรณีที่ไม่มีสัญญาณกริ่งเรียกเข้ามา ออปโตคัปเปิลจะยังไม่ทำงาน และที่ด้านเอาท์พุทของวงจรจะมีค่าสัญญาณไฟตรงเท่ากับ 5 โวลท์ ถ้ามีสัญญาณกริ่งเรียกเข้ามา จะทำให้มีแรงดันตกคร่อมที่ออปโตคัปเปิลด้านอินพุท ออปโตคัปเปิลจะทำงานโดยขา 4 และ 5 จะต่อเชื่อมกัน ได้เอาท์พุทของวงจรนี้มีแรงดันเป็น 0 โวลท์ โดยมีตัวต้านทานกับตัวเก็บประจุเป็นวงจรกรองทางด้านเอาท์พุท เอาท์พุทที่ได้จะนำมาเป็นอินพุทของแนนด์เกทสองตัวที่ต่อกันเป็นบัฟเฟอร์ที่เป็นชมิททริกเกอร์ ทำให้เวลาที่มีสัญญาณกริ่งเข้ามา ช่วงที่มีเสียงกริ่งดังจะได้สัญญาณเอาท์พุทของวงจรตรวจสอบสัญญาณกร่งเป็นสัญญาณลอจิกต่ำ ส่วนกรณีที่ไม่มีสัญญาณกริ่ง ที่ด้านเอาท์พุทก็จะได้สัญญาณลอจิกสูง ดังนั้นเมื่อมีสัญญาณกริ่งเข้ามา สัญญาณเอาท์พุทของวงจรตรวจสอบสัญญาณกริ่งก็จะมีลักษณะของสัญญาณพัลส์ตามลักษณะของช่วงเสียงกริ่งที่ดังขึ้น สัญญาณพัลส์ที่ได้จะส่งไปยังขาอินเทอร์รัพ INT ของไมโครโปรเซสเซอร์ต่อไป

วงจร HOOK ON/OFF



วงจรตัด-ต่อคู่สายโทรศัพท์

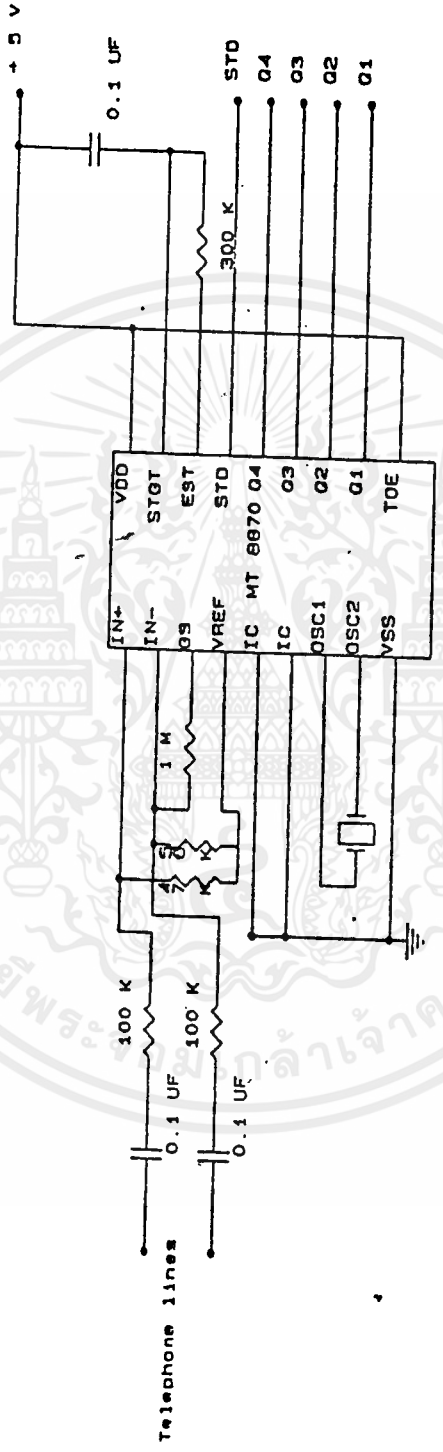
หน้าที่การทำงาน

ใช้ตัด-ต่อตัวต้านทานขนาด 560 โอห์มคร่อมคู่สาย เพื่อทำหน้าที่เสมือนมีการยกและวางหูโทรศัพท์ตามลำดับ

หลักการทำงาน

วงจรตัด-ต่อคู่สายโทรศัพท์นั้นเป็นวงจรขับรีเลย์ให้ทำงานเป็นสวิทช์ตัดต่อตัวต้านทาน 560 โอหม์คร่อมคู่สาย เพื่อให้แรงดันคร่อมคู่สายลดลงตามเกณฑ์ทำให้เสมือนมีการยกหูโทรศัพท์ โดยรีเลย์นั้นสามารถทำงานได้ โดยให้แรงดันไฟฟ้าคร่อมรีเลย์มีค่าอยู่ในช่วง +,- 10% ของค่าแรงดันไฟฟ้าของตัวรีเลย์นั้นและกระแสที่ไหลผ่านต้องมีค่าตามการทำงานของรีเลย์ โดยเราใช้ทรานซิสเตอร์เป็นตัวขับให้เกิดการทำงานของรีเลย์ ทำให้รีเลย์ต่อครบวงจร โดยทรานซิสเตอร์จะเริ่มทำงาน ก็ต่อเมื่อเราให้สัญญานลอจิกสูงแก่ทรานซิสเตอร์เท่านั้น และเมื่อทรานซิสเตอร์เริ่มทำงาน จะมีกระแสไหลจากขาคอลเลคเตอร์ไปยังขาอิมิตเตอร์ โดยที่รีเลย์คอยล์จะมีไดโอสต่อไบอสกลับอยู่ เพื่อให้เกิดกระแสไหลได้ในกรณีที่มีเปลี่ยนแปลงกระแสผ่านขดลวดอย่างกะทันหัน เพื่อป้องกันไม่ให้กระแสไปทำให้ทรานซิสเตอร์พัง

วงจร DTMF DECODER



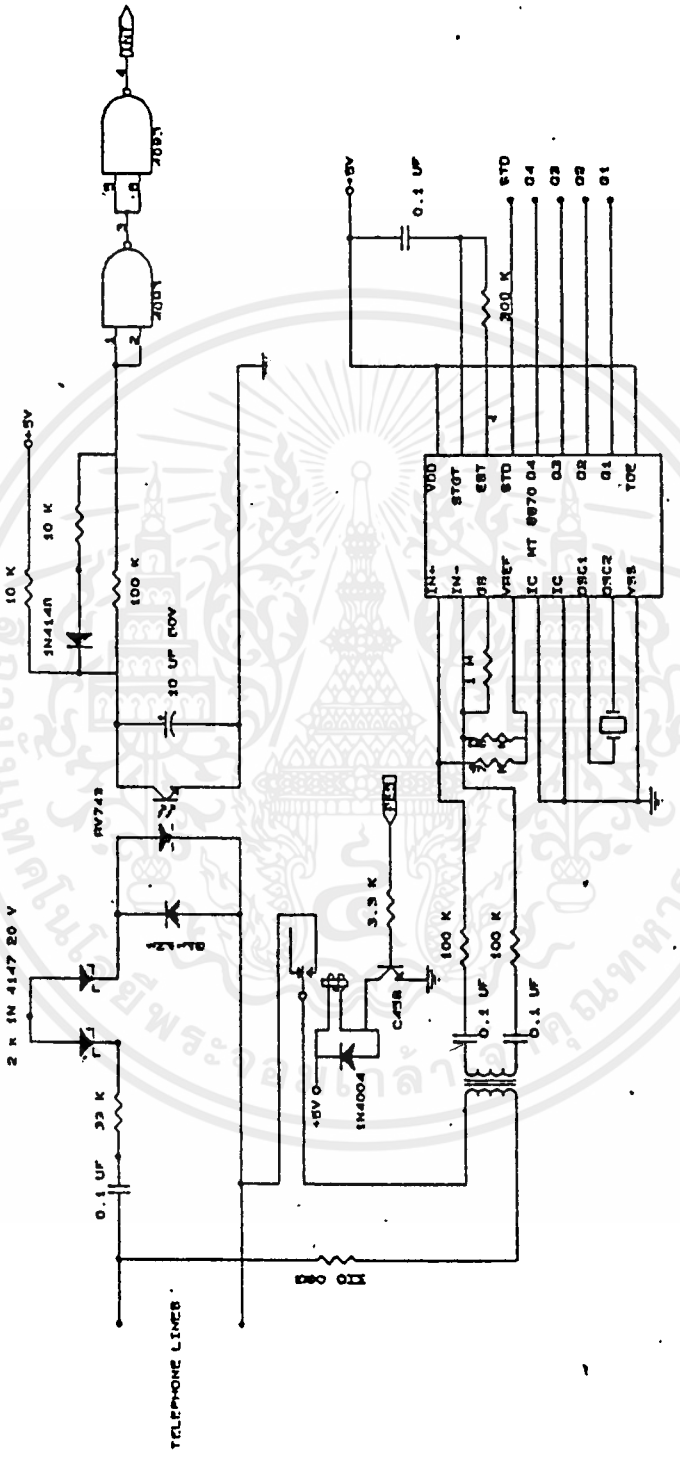
วงจรถอดรหัส DTMF

หน้าที่การทำงาน

รับสัญญาณ DTMF จากคู่สายแล้ว นำมาถอดรหัส DTMF ให้เป็นสัญญาณดิจิตอลที่เป็นข้อมูล 4 บิท พร้อมสัญญาณดิจิตอลที่เป็นสโตรบอีก 1 บิท เพื่อใช้เป็นข้อมูลส่งให้แก่ส่วนการทำงานของไมโครโปรเซสเซอร์

หลักการทำงาน

หลังจากต่อคู่สายโทรศัพท์แบบอัตโนมัติแล้ว มีสัญญาณ DTMF ถูกส่งมาตามคู่สาย สัญญาณนั้นจะมาเข้าทางอินพุทของไอซี MT8870 ซึ่งจะทำหน้าที่แปลงรหัสDTMF ให้เป็นสัญญาณดิจิตอลที่เป็นข้อมูล 4 บิท พร้อมทั้งสัญญาณสโตรบอีก 1 บิท ตามจำนวนรหัสDTMF ซึ่งมีทั้งหมด 16 รหัส สัญญาณทั้งหมดนี้จะถูกส่งต่อไปยังส่วนการทำงานของไมโครโปรเซสเซอร์เพื่อเป็นข้อมูลในการประมวลผลและสั่งงานต่อไป



วงจรส่วนตรวจสอบสัญญาณกริ่งโทรศัพท์และถอดรหัส DTMF

วงจร RTC

หน้าที่การทำงาน

เพื่อเชื่อมต่อวงจรนาฬิกากับไมโครโปรเซสเซอร์

หลักการทำงาน

ไมโครโปรเซสเซอร์จะอ่านข้อมูลจากวงจรนาฬิกา โดยผ่านทางพอร์ทรับข้อมูล ซึ่งข้อมูลที่อ่านมาจาก RTC จะเป็นข้อมูลของ นาทีและช.ม ข้อมูลเหล่านี้จะนำไปใช้ในการเปรียบเทียบเวลากับเวลาของการเปิด-ปิดอุปกรณ์ที่ผู้สั่งงานตั้งเอาไว้

7-SEGMENT 4 DIGITS

หน้าที่การทำงาน

เพื่อแสดงเวลาเป็นชั่วโมงและนาที

หลักการทำงาน

ใช้พอร์ทที่ส่งข้อมูลจากไมโครโปรเซสเซอร์มาควบคุม7-SEGMENT 2 พอร์ท โดยทำงานในลักษณะของการมัลติเพล็กโดยพอร์ทหนึ่งต่อเข้าที่เซกเมนต์ของทุกตัวแสดงผลเพื่อควบคุมให้เซกเมนต์เปล่งแสง อีกพอร์ทหนึ่งต่อกับขาร่วมของแต่ละตัวแสดงผล และการที่จะกำหนดให้ตัวแสดงผลตัวใดติด ก็ทำการควบคุมที่ขาร่วมของตัวแสดงลแต่ละตัวนั้น

บทที่ 3

โปรแกรมการทำงาน

ขั้นตอนการทำงานในส่วนโปรแกรม

ส่วนโปรแกรมหลัก

เริ่มต้นโปรมแกรมจะทำการเรียกเวลาจากวงจร RTC มาเก็บไว้ในเมมโมรี่ แล้วทำการแสดงเวลาที่เก็บไว้ออกทาง 7-SEGMENT 4 DIGITS ในหน่วยชั่วโมงและนาที การแสดงเวลาใช้วิธีการมัลติเพล็กซ์ เมื่อทำการแสดงเวลาครบ 4 DIGITS แล้ว ก็จะทำการเปรียบเทียบเวลาปัจจุบันกับเวลาที่ตั้งไว้ในการเปิด-ปิดอุปกรณ์ ถ้าพบว่าเวลาตรงกันก็จะทำการเปิด-ปิดอุปกรณ์ทันที และจะทำการเปรียบเทียบเวลาจนครบทั้ง 8 อุปกรณ์ แล้วจะวนกลับไปเริ่มต้นทำงานใหม่ เป็นเช่นนี้เรื่อยไปจนกว่าจะมีสัญญาณอินเทอร์รัพท์เข้ามา

ส่วนโปรแกรมอินเทอร์รัพท์

โปรแกรมอินเทอร์รัพท์เริ่มจากการรับสัญญาณกริ่งโทรศัพท์ที่เข้ามาทางขา int0โดยใช้ r0ของbank ที่ 3 เป็นตัวนับจำนวนสัญญาณ เมื่อมีเสียงกริ่งเข้ามา r0 จะ inc แล้วจะหน่วงเวลา 10 วินาที ถ้าก่อนหมดเวลา มีเสียงกริ่งเข้ามาอีกก็จะ inc r0 เมื่อค่าใน r0 = 0a(10) ก็จะมีการรับสายแล้วเข้าสู่การทำงานของโปรแกรมต่อไป ต่ถ้าหน่วงเวลา 10 วินาทีแล้วยังไม่ มีเสียงกริ่งเข้ามาก็จะออกจากโปรแกรมอินเทอรรัพท์

โปรแกรม intt

เป็น int0 interrupt ทำหน้าที่รับสัญญาณเสียงกริ่งเข้ามาทางขา int0 เมื่อมีเสียงกริ่งเข้ามาจะเพิ่มค่าที่ r0 ของ bank3 แล้วจะเรียกใช้โปรแกรม waitt เพื่อหน่วงเวลา 10 วินาที เมื่อครบ 10 วินาทีจะออกจาก interrupt ถ้ามีเสียงกริ่งเข้ามาระหว่างหน่วงเวลาจะเพิ่มค่าที่ r0 จนกระทั่งครบ 10 ครั้ง จะรับสายโทรศัพท์ แล้วเรียกใช้โปรแกรม userover getdata program ตามลำดับ เสร็จแล้วจะวางสาย reset r0 แล้วออกจาก interrupt

โปรแกรม userover

เมื่อมีการรับสายโทรศัพท์ จะเริ่มให้ timer 1 จับเวลา 1 นาที เมื่อครบ 1 นาที แล้วจะวางสาย

โปรแกรม getdata

จะรับ key ที่กดเข้ามา 9 ตัว โดยเรียกใช้โปรแกรม scankey แล้วเก็บค่า data1-data9

โปรแกรม program

เป็นการนำ data1-data9 มาตรวจสอบและเก็บค่าเวลาเปิด ปิดของอุปกรณ์ต่างๆ

ในขั้นตอนแรกเป็นการตรวจสอบ data1 ซึ่งต้องมีค่าระหว่าง 1-8 ถ้าไม่ถูกต้องจะออกจากโปรแกรม

ต่อมาจะเช็คค่า data2 ต้องอยู่ระหว่าง 0-2 โดยเรียกโปรแกรม check2 ถ้า data 2=2 จะต้องเช็คค่า data3 โดยใช้โปรแกรม check3 เสร็จแล้วจะเก็บค่า data2 และ data3 นี้เป็นค่าชั่วโมงที่จะเปิดอุปกรณ์

ต่อมาจะเช็คค่า data4 ต้องอยู่ระหว่าง 0-5 โดยเรียกใช้โปรแกรม check5 แล้วเก็บ data4 และ data5 เป็นค่านาทีที่จะเปิดอุปกรณ์

ส่วนต่อมาจะเป็นส่วนของเวลาปิด โดยนอกจากจะต้องเช็คค่าด้วยโปรแกรม check2 check3 check5 ดังข้างต้นแล้วยังต้องมีการตรวจสอบว่าเวลาปิดต้องมีค่ามากกว่าเวลาเปิดอีกด้วยโดยส่วนของเวลาปิดนี้จะเรียกใช้โปรแกรม offtime

โปรแกรม offtime

ทำหน้าที่ตรวจสอบว่าเวลาปิดมีค่ามากกว่าเวลาเปิดหรือไม่ ถ้าไม่ จะออกจากโปรแกรม ถ้าใช่จะเก็บค่าเวลาปิดไว้

โปรแกรม check2

ทำหน้าที่เช็คค่าของตัวแปร key ถ้ามีค่ามากกว่า 2 จะเซ็ตบิต errflg

โปรแกรม check3

ทำหน้าที่เช็คค่าของตัวแปร key ถ้ามีค่ามากกว่า 3 จะเซ็ตบิต errflg

โปรแกรม check5

ทำหน้าที่เช็คค่าของตัวแปร key ถ้ามีค่ามากกว่า 5 จะเซ็ตบิต errflg

โปรแกรม softtime

เป็น software delay โดยจะ delay รอบละ 1 ms โดยเราจะตั้งค่าที่ต้องการได้ที่ a(low byte) และ b(high byte)

โปรแกรม scankey

จะรับ key ที่กดเข้ามาจากโทรศัพท์ชนิดกดปุ่มเข้ามาทาง port1 บิต 0-3 โดยบิต 4 เป็น strobe โดยในที่นี้จะอนุญาตให้ รับค่า 0-9 เท่านั้น

```
8051  Cross-Assembler(1.3) (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

                    1;**********************************
                    2;        PORT  ADDRESS
                    3;**********************************
                    4;          PORT  8255
E0E3=               5CNTRWD     EQU    0E0E3H
E0E0=               6PORTA      EQU    0E0E0H
E0E1=               7PORTB      EQU    0E0E1H
E0E2=               8PORTC      EQU    0E0E2H
                    9
                    10;         PORT  RTC
E040=               11RTC       EQU    0E040H
E044=               12HOR1      EQU    RTC+4
E045=               13HOR10     EQU    RTC+5
E042=               14MIN1      EQU    RTC+2
E043=               15MIN10     EQU    RTC+3
E040=               16SEC1      EQU    RTC
E041=               17SEC10     EQU    RTC+1
E04F=               18CREG_F    EQU    RTC+0FH
E04D=               19CREG_D    EQU    RTC+0DH
                    20
                    21;**********************************
                    22;          BIT  ADDRESS
                    23;**********************************
0000=               24EQP1      EQU    00H
0001=               25EQP2      EQU    01H
0002=               26EQP3      EQU    02H
0003=               27EQP4      EQU    03H
0004=               28EQP5      EQU    04H
0005=               29EQP6      EQU    05H
```

```
0006=           30EQP7      EQU     06H
0007=           31EQP8      EQU     07H
0008=           32FR        EQU     08H
0009=           33FS        EQU     09H
                34
                35;*********************************
                36;         INTERNAL  RAM
                37;*********************************
0040=           38TIME_ON   EQU     40H
0050=           39TIME_OFF  EQU     50H
0037=           40RING      EQU     37H
0030=           41MINUTE    EQU     30H
0031=           42HOUR      EQU     31H
0032=           43DISBUF    EQU     32H
0020=           44CTRL_EQP  EQU     20H
                45
                46;*********************************
                47;          CONSTANT  VALUE
                48;*********************************
00FE=           49PATT      EQU     0FEH
                50                                  51
0041=           52onh1      equ     41h
0042=           53onm2      equ     42h
0043=           54onh2      equ     43h
0044=           55onm3      equ     44h
0045=           56onh3      equ     45h
0046=           57onm4      equ     46h
0047=           58onh4      equ     47h

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm
```

```
0048=           59onm5    equ     48h
0049=           60onh5    equ     49h
004A=           61onm6    equ     4ah
004B=           62onh6    equ     4bh
004C=           63onm7    equ     4ch
004D=           64onh7    equ     4dh
004E=           65onm8    equ     4eh
004F=           66onh8    equ     4fh
                67
0051=           68offh1   equ     51h
0052=           69offm2   equ     52h
0053=           70offh2   equ     53h
0054=           71offm3   equ     54h
0055=           72offh3   equ     55h
0056=           73offm4   equ     56h
0057=           74offh4   equ     57h
0058=           75offm5   equ     58h
0059=           76offh5   equ     59h
005A=           77offm6   equ     5ah
005B=           78offh6   equ     5bh
005C=           79offm7   equ     5ch
005D=           80offh7   equ     5dh
005E=           81offm8   equ     5eh
005F=           82offh8   equ     5fh
0060=           83newkey  equ     60h
0061=           84key     equ     61h
0011=           85errflg  equ     11h
0012=           86flg     equ     12h
0070=           87data1   equ     70h
0071=           88data2   equ     71h
0072=           89data3   equ     72h
0073=           90data4   equ     73h
0074=           91data5   equ     74h
```

```
0075=              92data6   equ     75h
0076=              93data7   equ     76h
0077=              94data8   equ     77h
0078=              95data9   equ     78h
0010=              96savetime equ    0010h
                   97
                   98
                   99
                  100;*********************************
                  101;*      TELEPHONE COMMANDER       *
                  102;*             BY                 *
                  103;*      COMKRICH KAEWWISET        *
                  104;*      BUNDIT WICHITPRAPAI       *
                  105;*        KUSA SONGTID            *
                  106;*********************************
                  107
                  108
0000 020025       109ljmp ladd
0003              110org 0003h
0003 0201EE       111ljmp intt
001B              112org 001bh
001B 02019D       113ljmp hardtime
001E 1201E4       114userdly:       lcall usertime
0021 1201EA       115lcall stoptime
0024 32           116reti
                  117

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0025 78FF         118LADD:          MOV R0,#0FFH
0027 D802         119LAG:           DJNZ R0,LAG1
0029 8006         120SJMP  SET
```

```
002B 79FF       121LAG1:            MOV R1,#0FFH
002D D9FE       122LAG2:            DJNZ R1,LAG2
002F 80F6       123SJMP LAG
                124
0031 751800     125SET:             mov 18h,#00h
0034 75A8FF     126mov ie,#0ffh
0037 90E04F     127MOV  DPTR,#CREG_F
003A 7401       128MOV  A,#01H
003C F0         129MOVX @DPTR,A
003D 7404       130MOV  A,#04H
003F F0         131MOVX @DPTR,A
0040 90E04D     132MOV  DPTR,#CREG_D
0043 7400       133MOV  A,#00H
0045 F0         134MOVX @DPTR,A
                135
0046 90E0E3     136MOV  DPTR,#CNTRWD
0049 7480       137MOV  A,#10000000B
004B F0         138MOVX @DPTR,A
                139
004C 7520FF     140MAIN:            MOV   CTRL_EQP,#0FFH
004F 90E0E0     141MOV   DPTR,#PORTA
0052 E520       142MOV   A,CTRL_EQP
0054 F0         143MOVX  @DPTR,A
0055 C3         144CLR   C
0056 9295       145MOV   P1.5,C
                146
0058 120063     147laggg:           LCALL   FIRST
005B 1200DF     148LCALL   COMPARE
005E 120196     149LCALL   CONTROL
0061 80F5       150SJMP    laggg
                151
0063 7932       152FIRST:           MOV  R1,#DISBUF
0065 7CFE       153MOV  R4,#PATT
```

```
0067 7830        154MOV  R0,#MINUTE
                 155
0069 90E044      156MAKE:          MOV  DPTR,#HOR1
006C E0          157MOVX A,@DPTR
006D 540F        158ANL  A,#0FH
006F FE          159MOV  R6,A
0070 90E045      160MOV  DPTR,#HOR10
0073 E0          161MOVX A,@DPTR
0074 540F        162ANL  A,#0FH
0076 23          163RL   A
0077 23          164RL   A
0078 23          165RL   A
0079 23          166RL   A
007A 4E          167ORL  A,R6
007B F531        168MOV  HOUR,A
                 169
007D 90E042      170MOV  DPTR,#MIN1
0080 E0          171MOVX A,@DPTR
0081 540F        172ANL  A,#0FH
0083 FE          173MOV  R6,A
0084 90E043      174MOV  DPTR,#MIN10
0087 E0          175MOVX A,@DPTR
0088 540F        176ANL  A,#0FH

8051   Cross-Assembler   (1.3)   (C) 1987,  1989 Binary Technology
c\et-em\project.asm

008A 23          177RL   A
008B 23          178RL   A
008C 23          179RL   A
008D 23          180RL   A
008E 4E          181ORL  A,R6
008F F530        182MOV  MINUTE,A
```

```
                        183
0091 1200C4             184LCALL ADJUST1
0094 7932               185MOV R1,#DISBUF
0096 7D04               186MOV R5,#04H
0098                    187DISPLY:
0098 E7                 188MOV  A,@R1
0099 90E0E1             189MOV  DPTR,#PORTB
009C F0                 190MOVX @DPTR,A
009D 90E0E2             191MOV  DPTR,#PORTC
00A0 EC                 192MOV   A,R4
00A1 F0                 193MOVX @DPTR,A
                        194
00A2 7E04               195MOV R6,#04H
00A4 7FFF               196DELAYY:          MOV R7,#0FFH
00A6 00                 197DLY:             NOP
00A7 00                 198NOP
00A8 00                 199NOP
00A9 DFFB               200DJNZ R7,DLY
00AB DEF7               201DJNZ R6,DELAYY
00AD EC                 202MOV A,R4
00AE 23                 203RL A
00AF FC                 204MOV R4,A
00B0 09                 205INC R1
00B1 DDE5               206DJNZ R5,DISPLY
00B3 22                 207RET   ;SJMP  FIRST
                    208
00B4 3F                 209TAB:             DB   3FH
00B5 065B4F             210DB   06H,5BH,4FH
00B8 666D7D             211DB   66H,6DH,7DH
00BB 077F6F             212DB   07H,7FH,6FH
00BE 777C39             213DB   77H,7CH,39H
00C1 5E7971             214DB   5EH,79H,71H
                        215
```

```
00C4 7A02       216ADJUST1:         MOV R2,#02H
00C6 7D02       217ADJUST:          MOV R5,#02H
00C8 E6         218MOV  A,@R0
00C9 FB         219MOV R3,A
00CA 540F       220ADJT1:           ANL A,#0FH
00CC 9000B4     221MOV DPTR,#TAB
00CF 93         222MOVC A,@A+DPTR
00D0 F7         223MOV @R1,A
00D1 09         224INC R1
00D2 DD04       225DJNZ R5,ADJT2
00D4 08         226INC R0
00D5 DAEF       227DJNZ R2,ADJUST
00D7 22         228RET
00D8 EB         229ADJT2:           MOV A,R3
00D9 03         230RR A
00DA 03         231RR A
00DB 03         232RR A
00DC 03         233RR A
00DD 80EB       234SJMP ADJT1
                235

8051 Cross-Assembler (1.3)   (C) 1987, 1989  Binary   Technology
c:\et-em\project.asm

                236
00DF            237COMPARE:
00DF AE30       238MOV    R6,MINUTE
00E1 AF31       239MOV    R7,HOUR
00E3 1200EB     240LCALL  CMP_ON
00E6 120136     241LCALL  CMP_OFF
00E9 C3         242CLR    C
00EA 22         243RET
                244
```

```
00EB 7840      245CMP_ON:          MOV    R0,#TIME_ON
00ED C3        246CHEQ1_ON:        CLR    C
00EE 120182    247LCALL  CMP
00F1 BD0102    248CJNE   R5,#01H,CHEQ2_ON
00F4 C200      249CLR    EQP1
00F6 C3        250CHEQ2_ON:        CLR    C
00F7 120182    251LCALL  CMP
00FA BD0102    252CJNE   R5,#01H,CHEQ3_ON
00FD C201      253CLR    EQP2
00FF C3        254CHEQ3_ON:        CLR    C
0100 120182    255LCALL  CMP
0103 BD0102    256CJNE   R5,#01H,CHEQ4_ON
0106 C202      257CLR    EQP3
0108 C3        258CHEQ4_ON:        CLR    C
0109 120182    259LCALL  CMP
010C BD0102    260CJNE   R5,#01H,CHEQ5_ON
010F C203      261CLR    EQP4
0111 C3        262CHEQ5_ON:        CLR    C
0112 120182    263LCALL  CMP
0115 BD0102    264CJNE   R5,#01H,CHEQ6_ON
0118 C204      265CLR    EQP5
011A C3        266CHEQ6_ON:        CLR    C
011B 120182    267LCALL  CMP
011E BD0102    268CJNE   R5,#01H,CHEQ7_ON
0121 C205      269CLR    EQP6
0123 C3        270CHEQ7_ON:        CLR    C
0124 120182    271LCALL  CMP
0127 BD0102    272CJNE   R5,#01H,CHEQ8_ON
012A C206      273CLR    EQP7
012C C3        274CHEQ8_ON:        CLR    C
012D 120182    275LCALL  CMP
0130 BD014E    276CJNE   R5,#01H,END
0133 C207      277CLR    EQP8
```

```
0135 22          278RET
                 279
0136 7850        280CMP_OFF:          MOV     R0,#TIME_OFF
0138 C3          281CHEQ1_OFF:        CLR     C
0139 120182      282LCALL   CMP
013C BD0102      283CJNE    R5,#01H,CHEQ2_OFF
013F D200        284SETB     EQP1
0141 C3          285CHEQ2_OFF:        CLR     C
0142 120182      286LCALL   CMP
0145 BD0102      287CJNE    R5,#01H,CHEQ3_OFF
0148 D201        288SETB     EQP2
014A C3          289CHEQ3_OFF:        CLR     C
014B 120182      290LCALL   CMP
014E BD0102      291CJNE    R5,#01H,CHEQ4_OFF
0151 D202        292SETB     EQP3
0153 C3          293CHEQ4_OFF:        CLR     C
0154 120182      294LCALL   CMP

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0157 BD0102      295CJNE    R5,#01H,CHEQ5_OFF
015A D203        296SETB     EQP4
015C C3          297CHEQ5_OFF:        CLR     C
015D 120182      298LCALL   CMP
0160 BD0102      299CJNE    R5,#01H,CHEQ6_OFF
0163 D204        300SETB     EQP5
0165 C3          301CHEQ6_OFF:        CLR     C
0166 120182      302LCALL   CMP
0169 BD0102      303CJNE    R5,#01H,CHEQ7_OFF
016C D205        304SETB     EQP6
016E C3          305CHEQ7_OFF:        CLR     C
016F 120182      306LCALL   CMP
```

```
0172 BD0102     307CJNE   R5,#01H,CHEQ8_OFF
0175 D206       308SETB    EQP7
0177 C3         309CHEQ8_OFF:      CLR    C
0178 120182     310LCALL  CMP
017B BD0103     311CJNE   R5,#01H,END
017E D207       312SETB    EQP8
0180 22         313RET
                314
0181 22         315END:            RET
                316
0182 7D00       317CMP:            MOV    R5,#00H
0184 EE         318MOV    A,R6
0185 96         319SUBB   A,@R0
0186 C3         320CLR    C
0187 700A       321JNZ    CMP2
0189 08         322INC    R0
018A EF         323MOV    A,R7
018B 96         324SUBB   A,@R0
018C C3         325CLR    C
018D 7005       326JNZ    CMP1
018F 7D01       327MOV    R5,#01H
0191 8001       328SJMP   CMP1
0193 08         329CMP2:           INC    R0
0194 08         330CMP1:           INC    R0
0195 22         331RET
                332
                333
0196            334CONTROL:
0196 90E0E0     335MOV  DPTR,#PORTA
0199 E520       336MOV   A,CTRL_EQP
019B F0         337MOVX  @DPTR,A
019C 22         338RET
                339
```

```
019D C0E0       340     hardtime:       push acc
019F C083       341     push dph
01A1 C082       342     push dpl
01A3 900010     343     mov dptr,#savetime
01A6 E0         344     movx a,@dptr
01A7 14         345     dec a
01A8 B40007     346     cjne a,#00h,aff
01AB F0         347     movx @dptr,a
01AC A3         348     inc dptr
01AD E0         349     movx a,@dptr
01AE 600B       350     jz don
01B0 8012       351     sjmp sava
01B2 B4FF0F     352     aff:            cjne a,#0ffh,sava
01B5 F0         353     movx @dptr,a

 8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\ project.asm

01B6 A3         354     inc dptr
01B7 E0         355     movx a,@dptr
01B8 14         356     dec a
01B9 8009       357     sjmp sava
01BB D082       358     don:            pop dpl
01BD D083       359     pop dph
01BF D0E0       360     pop acc
                361
01C1 02001E     362     ljmp userdly
01C4 F0         363     sava:           movx @dptr,a
01C5 D082       364     pop dpl
01C7 D083       365     pop dph
01C9 D0E0       366     pop acc
01CB 31CE       367     acall starttim
01CD 32         368     reti
```

```
                    369
01CE 758DFA         370         starttim:       mov th1,#0fah
01D1 758BCB         371         mov tl1,#0cbh
01D4 758DFF         372         mov th1,#0ffh
01D7 758B00         373         mov tl1,#00h
01DA 53890F         374         anl tmod,#0fh
01DD 438940         375         orl tmod,#40h
01E0 438840         376         orl tcon,#40h
01E3 22             377         ret
                    378
01E4 C295           379         usertime:       clr p1.5
01E6 751800         380         mov 18h,#00h
01E9 22             381         ret
                    382
01EA 5388BF         383         stoptime:       anl tcon,#0bfh
01ED 22             384         ret
                    385
                    386;interrupt program start here
                    387
01EE                388intt:
01EE C0D0           389push psw
01F0 C0E0           390push acc
01F2 C0F0           391push b
01F4 C083           392push dph
01F6 C082           393push dpl
01F8 0518           394inc 18h
01FA 75A8FF         395mov ie,#0ffh
01FD E518           396mov a,18h
01FF B40A1B         397cjne a,#0ah,waitt
0202 D295           398setb p1.5
0204 120235         399lcall userover
0207 120255         400lcall getdata
020A 12028C         401lcall program
```

```
020D C295       402clr p1.5
020F 751800     403mov 18h,#00h
0212 D082       404pop dpl
0214 D083       405pop dph
0216 D0F0       406pop b
0218 D0E0       407pop acc
021A D0D0       408pop psw
021C 32         409reti
021D 7410       410waitt:        mov a,#10h       ;delay for 10s
021F 75F027     411mov b,#27h
0222 1206DA     412lcall softtime

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0225 751800     413mov 18h,#00h
0228 D083       414pop dph
022A D082       415pop dpl
022C D0F0       416pop b
022E D0E0       417pop acc
0230 D0D0       418pop psw
0232 D0A8       419pop ie
0234 32         420reti
                421
0235 900010     422userover:      mov dptr,#savetime
0238 7460       423mov a,#60h
023A F0         424movx @dptr,a
023B A3         425inc dptr
023C 74FF       426mov a,#0ffh
023E F0         427movx @dptr,a
023F 43A888     428orl ie,#88h
0242 120245     429lcall starttime
                430
```

```
                    431
                    432
0245 758DFA         433starttime:       mov th1,#0fah
0248 758BCB         434mov tl1,#0cbh
024B 53890F         435anl tmod,#0fh
024E 438940         436orl tmod,#40h
0251 438840         437orl tcon,#40h
0254 22             438ret
                    439
0255 120701         440getdata:         lcall scankey
0258 856070         441mov data1,newkey
025B 120701         442lcall scankey
025E 856071         443mov data2,newkey
0261 120701         444lcall scankey
0264 856072         445mov data3,newkey
0267 120701         446lcall scankey
026A 856073         447mov data4,newkey
026D 120701         448lcall scankey
0270 856074         449mov data5,newkey
0273 120701         450lcall scankey
0276 856075         451mov data6,newkey
0279 120701         452lcall scankey
027C 856076         453mov data7,newkey
027F 120701         454lcall scankey
0282 856077         455mov data8,newkey
0285 120701         456lcall scankey
0288 856078         457mov data9,newkey
028B 22             458ret
                    459
028C E570           460program:         mov a,data1
028E B40147         461cjne a,#01h,chk2
0291 857161         462oh1:             mov key,data2
0294 1206AD         463lcall check2
```

```
0297 101137      464jbc errflg,er1
029A E571        465mov a,data2
029C B40214      466cjne a,#02h,con1
029F E571        467mov a,data2
02A1 C4          468swap a
02A2 F541        469mov onh1,a
02A4 857261      470mov key,data3
02A7 1206BC      471lcall check3

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

02AA 101124      472jbc errflg,er1
02AD E572        473mov a,data3
02AF 4241        474orl onh1,a
02B1 8009        475sjmp om1
02B3 E571        476con1:          mov a,data2
02B5 C4          477swap a
02B6 F541        478mov onh1,a
02B8 E572        479mov a,data3
02BA 4241        480orl onh1,a
02BC 857361      481om1:           mov key,data4
02BF 1206CB      482lcall check5
02C2 10110C      483jbc errflg,er1
02C5 E573        484mov a,data4
02C7 C4          485swap a
02C8 F540        486mov time_on,a
02CA E574        487mov a,data5
02CC 4240        488orl time_on,a
02CE 0204ED      489ljmp off1
02D1 754100      490er1:           mov onh1,#00h
02D4 754000      491mov time_on,#00h
02D7 22          492ret
```

```
02D8 E570       493chk2:            mov a,data1
02DA B40247     494cjne a,#02h,chk3
02DD 857161     495oh2:             mov key,data2
02E0 1206AD     496lcall check2
02E3 101137     497jbc errflg,er2
02E6 E571       498mov a,data2
02E8 B40214     499cjne a,#02h,con2
02EB E571       500mov a,data2
02ED C4         501swap a
02EE F543       502mov onh2,a
02F0 857261     503mov key,data3
02F3 1206BC     504lcall check3
02F6 101124     505jbc errflg,er2
02F9 E572       506mov a,data3
02FB 4243       507orl onh2,a
02FD 8009       508sjmp om2
02FF E571       509con2:            mov a,data2
0301 C4         510swap a
0302 F543       511mov onh2,a
0304 E572       512mov a,data3
0306 4243       513orl onh2,a
0308 857361     514om2:             mov key,data4
030B 1206CB     515lcall check5
030E 10110C     516jbc errflg,er2
0311 E573       517mov a,data4
0313 C4         518swap a
0314 F542       519mov onm2,a
0316 E574       520mov a,data5
0318 4242       521orl onm2,a
031A 020509     522ljmp off2
031D 754300     523er2:             mov onh2,#00h
0320 754200     524mov onm2,#00h
0323 22         525ret
```

```
0324 E570       526chk3:            mov a,data1
0326 B40347     527cjne a,#03h,chk4
0329 857161     528oh3:             mov key,data2
032C 1206AD     529lcall check2
032F 101137     530jbc errflg,er3

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0332 E571       531mov a,data2
0334 B40214     532cjne a,#02h,con3
0337 E571       533mov a,data2
0339 C4         534swap a
033A F545       535mov onh3,a
033C 857261     536mov key,data3
033F 1206BC     537lcall check3
0342 101124     538jbc errflg,er3
0345 E572       539mov a,data3
0347 4245       540orl onh3,a
0349 8009       541sjmp om3
034B E571       542con3:            mov a,data2
034D C4         543swap a
034E F545       544mov onh3,a
0350 E572       545mov a,data3
0352 4245       546orl onh3,a
0354 857361     547om3:             mov key,data4
0357 1206CB     548lcall check5
035A 10110C     549jbc errflg,er3
035D E573       550mov a,data4
035F C4         551swap a
0360 F544       552mov onm3,a
0362 E574       553mov a,data5
0364 4244       554orl onm3,a
```

```
0366 020525     555ljmp off3
0369 754500     556er3:          mov onh3,#00h
036C 754400     557mov onm3,#00h
036F 22         558ret
0370 E570       559chk4:         mov a,data1
0372 B40447     560cjne a,#04h,chk5
0375 857161     561oh4:          mov key,data2
0378 1206AD     562lcall check2
037B 101137     563jbc errflg,er4
037E E571       564mov a,data2
0380 B40214     565cjne a,#02h,con4
0383 E571       566mov a,data2
0385 C4         567swap a
0386 F547       568mov onh4,a
0388 857261     569mov key,data3
038B 1206BC     570lcall check3
038E 101124     571jbc errflg,er4
0391 E572       572mov a,data3
0393 4247       573orl onh4,a
0395 8009       574sjmp om4
0397 E571       575con4:         mov a,data2
0399 C4         576swap a
039A F547       577mov onh4,a
039C E572       578mov a,data3
039E 4247       579orl onh4,a
03A0 857361     580om4:          mov key,data4
03A3 1206CB     581lcall check5
03A6 10110C     582jbc errflg,er4
03A9 E573       583mov a,data4
03AB C4         584swap a
03AC F546       585mov onm4,a
03AE E574       586mov a,data5
03B0 4246       587orl onm4,a
```

```
03B2 020541     588ljmp off4
03B5 754700     589er4:             mov onh4,#00h

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

03B8 754600     590mov onm4,#00h
03BB 22         591ret
03BC E570       592chk5:            mov a,data1
03BE B40547     593cjne a,#05h,chk6
03C1 857161     594oh5:             mov key,data2
03C4 1206AD     595lcall check2
03C7 101137     596jbc errflg,er5
03CA E571       597mov a,data2
03CC B40214     598cjne a,#02h,con5
03CF E571       599mov a,data2
03D1 C4         600swap a
03D2 F549       601mov onh5,a
03D4 857261     602mov key,data3
03D7 1206BC     603lcall check3
03DA 101124     604jbc errflg,er5
03DD E572       605mov a,data3
03DF 4249       606orl onh5,a
03E1 8009       607sjmp om5
03E3 E571       608con5:            mov a,data2
03E5 C4         609swap a
03E6 F549       610mov onh5,a
03E8 E572       611mov a,data3
03EA 4249       612orl onh5,a
03EC 857361     613om5:             mov key,data4
03EF 1206CB     614lcall check5
03F2 10110C     615jbc errflg,er5
03F5 E573       616mov a,data4
```

```
03F7 C4          617swap a
03F8 F548        618mov onm5,a
03FA E574        619mov a,data5
03FC 4248        620orl onm5,a
03FE 02055D      621ljmp off5
0401 754900      622er5:          mov onh5,#00h
0404 754800      623mov onm5,#00h
0407 22          624ret
0408 E570        625chk6:         mov a,data1
040A B40647      626cjne a,#06h,chk7
040D 857161      627oh6:          mov key,data2
0410 1206AD      628lcall check2
0413 101137      629jbc errflg,er6
0416 E571        630mov a,data2
0418 B40214      631cjne a,#02h,con6
041B E571        632mov a,data2
041D C4          633swap a
041E F54B        634mov onh6,a
0420 857261      635mov key,data3
0423 1206BC      636lcall check3
0426 101124      637jbc errflg,er6
0429 E572        638mov a,data3
042B 424B        639orl onh6,a
042D 8009        640sjmp om6
042F E571        641con6:         mov a,data2
0431 C4          642swap a
0432 F54B        643mov onh6,a
0434 E572        644mov a,data3
0436 424B        645orl onh6,a
0438 857361      646om6:          mov key,data4
043B 1206CB      647lcall check5
043E 10110C      648jbc errflg,er6
```

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology

c:\et-em\project.asm

```
0441 E573       649mov a,data4
0443 C4         650swap a
0444 F54A       651mov onm6,a
0446 E574       652mov a,data5
0448 424A       653orl onm6,a
044A 020579     654ljmp off6
044D 754B00     655er6:          mov onh6,#00h
0450 754A00     656mov onm6,#00h
0453 22         657ret
0454 E570       658chk7:         mov a,data1
0456 B40747     659cjne a,#07h,chk8
0459 857161     660oh7:          mov key,data2
045C 1206AD     661lcall check2
045F 101137     662jbc errflg,er7
0462 E571       663mov a,data2
0464 B40214     664cjne a,#02h,con7
0467 E571       665mov a,data2
0469 C4         666swap a
046A F54D       667mov onh7,a
046C 857261     668mov key,data3
046F 1206BC     669lcall check3
0472 101124     670jbc errflg,er7
0475 E572       671mov a,data3
0477 424D       672orl onh7,a
0479 8009       673sjmp om7
047B E571       674con7:          mov a,data2
047D C4         675swap a
047E F54D       676mov onh7,a
0480 E572       677mov a,data3
0482 424D       678orl onh7,a
```

```
0484 857361     679om7:             mov key,data4
0487 1206CB     680lcall check5
048A 10110C     681jbc errflg,er7
048D E573       682mov a,data4
048F C4         683swap a
0490 F54C       684mov onm7,a
0492 E574       685mov a,data5
0494 424C       686orl onm7,a
0496 020595     687ljmp off7
0499 754D00     688er7:             mov onh7,#00h
049C 754C00     689mov onm7,#00h
049F 22         690ret
04A0 E570       691chk8:             mov a,data1
04A2 B40847     692cjne a,#08h,wrong
04A5 857161     693oh8:             mov key,data2
04A8 1206AD     694lcall check2
04AB 101137     695jbc errflg,er8
04AE E571       696mov a,data2
04B0 B40214     697cjne a,#02h,con8
04B3 E571       698mov a,data2
04B5 C4         699swap a
04B6 F54F       700mov onh8,a
04B8 857261     701mov key,data3
04BB 1206BC     702lcall check3
04BE 101124     703jbc errflg,er8
04C1 E572       704mov a,data3
04C3 424F       705orl onh8,a
04C5 8009       706sjmp om8
04C7 E571       707con8:            mov a,data2
```

```
8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm
```

```
04C9 C4          708swap a
04CA F54F        709mov onh8,a
04CC E572        710mov a,data3
04CE 424F        711orl onh8,a
04D0 857361      712om8:          mov key,data4
04D3 1206CB      713lcall check5
04D6 10110C      714jbc errflg,er8
04D9 E573        715mov a,data4
04DB C4          716swap a
04DC F54E        717mov onm8,a
04DE E574        718mov a,data5
04E0 424E        719orl onm8,a
04E2 0205B1      720ljmp off8
04E5 754F00      721er8:          mov onh8,#00h
04E8 754E00      722mov onm8,#00h
04EB 22          723ret
04EC 22          724wrong:        ret
                 725
04ED C004        726off1:         push 04h
04EF C003        727push 03h
04F1 C002        728push 02h
04F3 C001        729push 01h
04F5 A941        730mov r1,onh1
04F7 AA40        731mov r2,time_on
04F9 1205CD      732lcall offtime
04FC 8B51        733mov offh1,r3
04FE 8C50        734mov time_off,r4
0500 D001        735pop 01h
0502 D002        736pop 02h
0504 D003        737pop 03h
0506 D004        738pop 04h
0508 22          739ret
0509 C004        740off2:         push 04h
```

```
050B C003       741push 03h
050D C002       742push 02h
050F C001       743push 01h
0511 A943       744mov r1,onh2
0513 AA42       745mov r2,onm2
0515 1205CD     746lcall offtime
0518 8B53       747mov offh2,r3
051A 8C52       748mov offm2,r4
051C D001       749pop 01h
051E D002       750pop 02h
0520 D003       751pop 03h
0522 D004       752pop 04h
0524 22         753ret
0525 C004       754off3:        push 04h
0527 C003       755push 03h
0529 C002       756push 02h
052B C001       757push 01h
052D A945       758mov r1,onh3
052F AA44       759mov r2,onm3
0531 1205CD     760lcall offtime
0534 8B55       761mov offh3,r3
0536 8C54       762mov offm3,r4
0538 D001       763pop 01h
053A D002       764pop 02h
053C D003       765pop 03h
053E D004       766pop 04h

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0540 22         767ret
0541 C004       768off4:        push 04h
0543 C003       769push 03h
```

```
0545 C002       770push 02h
0547 C001       771push 01h
0549 A947       772mov r1,onh4
054B AA46       773mov r2,onm4
054D 1205CD     774lcall offtime
0550 8B57       775mov offh4,r3
0552 8C56       776mov offm4,r4
0554 D001       777pop 01h
0556 D002       778pop 02h
0558 D003       779pop 03h
055A D004       780pop 04h
055C 22         781ret
055D C004       782off5:        push 04h
055F C003       783push 03h
0561 C002       784push 02h
0563 C001       785push 01h
0565 A949       786mov r1,onh5
0567 AA48       787mov r2,onm5
0569 1205CD     788lcall offtime
056C 8B59       789mov offh5,r3
056E 8C58       790mov offm5,r4
0570 D001       791pop 01h
0572 D002       792pop 02h
0574 D003       793pop 03h
0576 D004       794pop 04h
0578 22         795ret
0579 C004       796off6:        push 04h
057B C003       797push 03h
057D C002       798push 02h
057F C001       799push 01h
0581 A94B       800mov r1,onh6
0583 AA4A       801mov r2,onm6
0585 1205CD     802lcall offtime
```

```
0588 8B5B       803mov offh6,r3
058A 8C5A       804mov offm6,r4
058C D001       805pop 01h
058E D002       806pop 02h
0590 D003       807pop 03h
0592 D004       808pop 04h
0594 22         809ret
0595 C004       810off7:          push 04h
0597 C003       811push 03h
0599 C002       812push 02h
059B C001       813push 01h
059D A94D       814mov r1,onh7
059F AA4C       815mov r2,onm7
05A1 1205CD     816lcall offtime
05A4 8B5D       817mov offh7,r3
05A6 8C5C       818mov offm7,r4
05A8 D001       819pop 01h
05AA D002       820pop 02h
05AC D003       821pop 03h
05AE D004       822pop 04h
05B0 22         823ret
05B1 C004       824off8:          push 04h
05B3 C003       825push 03h

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

05B5 C002       826push 02h
05B7 C001       827push 01h
05B9 A94F       828mov r1,onh8
05BB AA4E       829mov r2,onm8
05BD 1205CD     830lcall offtime
05C0 8B5F       831mov offh8,r3
```

```
05C2 8C5E       832mov offm8,r4
05C4 D001       833pop 01h
05C6 D002       834pop 02h
05C8 D003       835pop 03h
05CA D004       836pop 04h
05CC 22         837ret
                838
05CD 857561     839offtime:     mov key,data6
05D0 1206AD     840lcall check2          ;check if more than 2
05D3 101103     841jbc errflg,e1
05D6 0205DC     842ljmp ne1
05D9 0206A8     843e1:          ljmp err
05DC E9         844ne1:        mov a,r1
05DD C4         845swap a
05DE 540F       846anl a,#0fh           ;first 4 bit
05E0 B57503     847cjne a,data6,more    ;check if off >< on
05E3 020623     848ljmp chklow          ;off = in check last 4 bit
05E6 C3         849more:       clr c
05E7 9575       850subb a,data6
05E9 10D603     851jbc ac,ok            ;off > on ok
05EC 0206A8     852ljmp err             ;off < on error
05EF E575       853ok:     mov a,data6  ;ok1 save offh1 and offm1
05F1 C4         854swap a
05F2 FB         855mov r3,a
05F3 E575       856mov a,data6
05F5 B4020F     857cjne a,#02h,ne2
05F8 857661     858mov key,data7
05FB 1206BC     859lcall check3         ;check if more than 3
05FE 101103     860jbc errflg,e2
0601 020607     861ljmp ne2
0604 0206A8     862e2:          ljmp err
0607 E576       863ne2:         mov a,data7
0609 4203       864orl 03,a
```

```
060B 857761     865mov key,data8
060E 1206CB     866lcall check5         ;check if more than 5
0611 101103     867jbc errflg,e3
0614 02061A     868ljmp ne3
0617 0206A8     869e3:              ljmp err
061A E577       870ne3:             mov a,data8
061C C4         871swap a
061D FC         872mov r4,a
061E E578       873mov a,data9
0620 4204       874orl 04,a
0622 22         875ret
0623 E575       876chklow:          mov a,data6
0625 C4         877swap a
0626 FB         878mov r3,a
0627 E575       879mov a,data6
0629 B4020F     880cjne a,#02h,ne4
062C 857661     881mov key,data7
062F 1206BC     882lcall check3          ;check if more than 3
0632 101103     883jbc errflg,e4
0635 02063B     884ljmp ne4

8051 Cross-Assembler (1.3)  (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

0638 0206A8     885e4:              ljmp err
063B E9         886ne4:             mov a,r1
063C 540F       887anl a,#0fh           ;last 4 bit
063E B57603     888cjne a,data7,mor     ;check if off >< on
0641 020666     889ljmp chkmin          ;offh1 = onh1 check minute
0644 C3         890mor:             clr c
0645 E9         891mov a,r1
0646 540F       892anl a,#0fh
0648 9576       893subb a,data7
```

```
064A 10D603     894jbc ac,okk              ;off > on ok
064D 0206A8     895ljmp err                ;off < on error
0650 E576       896okk:   mov a,data7      ;save offh1 and offm1
0652 4203       897orl 03,a
0654 857761     898mov key,data8
0657 1206CB     899lcall check5            ;check if more than 5
065A 10114B     900jbc errflg,err
065D E577       901mov a,data8
065F C4         902swap a
0660 FC         903mov r4,a
0661 E578       904mov a,data9
0663 4204       905orl 04,a
0665 22         906ret
0666 E576       907chkmin:         mov a,data7
0668 4203       908orl 03,a
066A 857761     909mov key,data8
066D 1206CB     910lcall check5            ;check if more than 5
0670 101135     911jbc errflg,err
0673 EA         912mov a,r2
0674 C4         913swap a
0675 540F       914anl a,#0fh              ;first 4 bit
0677 B57703     915cjne a,data8,morr       ;check if off >< on
067A 020693     916ljmp chklm              ;off = on check last 4 bit
067D C3         917morr:           clr c
067E EA         918mov a,r2
067F C4         919swap a
0680 540F       920anl a,#0fh              ;first 4 bit
0682 9577       921subb a,data8
0684 10D603     922jbc ac,okkk             ;off > on ok
0687 0206A8     923ljmp err                ;off < on error
068A E577       924okkk:           mov a,data8    ;save offm1
068C C4         925swap a
068D FC         926mov r4,a
```

```
068E E578       927mov a,data9
0690 4204       928orl 04,a
0692 22         929ret
0693 E577       930chklm:          mov a,data8
0695 C4         931swap a
0696 FC         932mov r4,a
0697 C3         933clr c
0698 EA         934mov a,r2
0699 540F       935anl a,#0fh              ;last 4 bit
069B 9578       936subb a,data9
069D 10D603     937jbc ac,okkkk            ;off > on ok
06A0 0206A8     938ljmp err                ;off <= on error
06A3 E578       939okkkk:          mov a,data9    ;save offm1
06A5 4204       940orl 04,a
06A7 22         941ret
06A8 7B23       942err:            mov r3,#23h
06AA 7C59       943mov r4,#59h

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

06AC 22         944ret                     ;return to interrupt
                945
                946
                947
                948;check2 check if more than 2
06AD C3         949check2:          clr c
06AE C211       950clr errflg
06B0 7402       951mov a,#02h
06B2 9561       952subb a,key
06B4 10D602     953jbc ac,error2
06B7 8002       954sjmp ov2
06B9 D211       955error2:          setb errflg
```

```
06BB 22          956ov2:          ret
                 957
                 958;check3 check if more than 3
06BC C3          959check3:       clr c
06BD C211        960clr errflg
06BF 7403        961mov a,#03h
06C1 9561        962subb a,key
06C3 10D602      963jbc ac,error3
06C6 8002        964sjmp ov3
06C8 D211        965error3:        setb errflg
06CA 22          966ov3:          ret
                 967
                 968;check5 check if more than 5
06CB C3          969check5:       clr c
06CC C211        970clr errflg
06CE 7405        971mov a,#05h
06D0 9561        972subb a,key
06D2 10D602      973jbc ac,error5
06D5 8002        974sjmp ov5
06D7 D211        975error5:       setb errflg
06D9 22          976ov5:          ret
                 977
                 978;softtime will delay for 1ms
                 979;put high desire byte in b
                 980;put low desire byte in a
00EC=            981softtime:     delay equ 0ech
06DA C007        982push 07h                ;save r7
06DC C0E0        983push acc                ;save a for a=b=0 test
06DE 45F0        984orl a,b                 ;will be 00 if both 00
06E0 B40004      985cjne a,#00h,ook         ;retern if all 00
06E3 D0E0        986pop acc                 ;keep stack balanced
06E5 8017        987sjmp done
06E7 D0E0        988ook:          pop acc;not all zeroes,proceed
```

```
06E9 7FEC       989timer:    mov r7,#delay    ;initialize r7
06EB 00         990onemil:   nop      ;tune the loop for 6 cycles
06EC 00         991nop
06ED 00         992nop
06EE 00         993nop
06EF DFFA       994djnz r7,onemil
06F1 00         995nop
06F2 D5E0F4     996djnz acc,timer   ;count a and b down as one
06F5 B5F002     997cjne a,b,bdown   ;a=00 count b down until 00
06F8 8004       998sjmp done        ;if so then delay is done
06FA 15F0       999bdown:          dec b
06FC 80EB      1000sjmp timer
06FE D007      1001done: pop 07h ;restore r7 to original value
0700 22        1002ret

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

               1003
               1004;scankey bit 0-3 of port1 are data bit 4 is
                  a strobe
               1005;only 0-9 are valid
               1006
0701 309536    1007scankey:     jnb p1.5,go
               1008
0704 C001      1009scan:        push    01h
0706 7590FF    1010      MOV    P1,#0FFH
0709 A294      1011      MOV    C,P1.4
070B 50F7      1012      JNC    scan
070D 7590FF    1013      MOV    P1,#0FFH
0710 A990      1014      MOV    r1,P1
0712 53010F    1015      ANL    01,#00001111B
0715 B90A04    1016      cjne   r1,#0ah,test
```

```
0718 7400      1017      mov     a,#00h
071A 8008      1018      sjmp    one
071C C3        1019test:         clr     c
071D 7409      1020      mov     a,#09h
071F 99        1021      subb    a,r1
0720 10D6E1    1022      jbc     ac,scan
0723 E9        1023      mov     a,r1
0724 F560      1024one:          mov     newkey,a
0726 90E0E0    1025      MOV     DPTR,#PORTA
0729 F0        1026      MOVX    @DPTR,A
072A D3        1027        WAIT:           SETB    C
072B 9294      1028      MOV     P1.4,C
072D A294      1029      MOV     C,P1.4
072F 40F9      1030      JC      WAIT
0731 90E0E0    1031      MOV     DPTR,#PORTA
0734 7400      1032      MOV     A,#00H
0736 F0        1033      MOVX    @DPTR,A
0737 D001      1034      pop 01h
0739 22        1035      ret
               1036
073A 751800    1037go:           mov  18h,#00h
073D D082      1038pop dpl
073F D083      1039pop dph
0741 D0F0      1040pop b
0743 D0E0      1041pop acc
0745 D0D0      1042pop psw
0747 D0A8      1043pop ie
0749 32        1044reti
               1045
               1046
               1047
               1048
               1049
```

```
                1050
                1051
                1052
                1053
                1054
                1055
                1056
                1057
                1058
                1059

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

      adjt1 = 00CA    220   234
      adjt2 = 00D8    229   225
     adjust = 00C6    217   227
    adjust1 = 00C4    216   184
        aff = 01B2    352   346
      bdown = 06FA    999   997
     check2 = 06AD    949   463   496   529   562
                      595   628   661   694   840
     check3 = 06BC    959   471   504   537   570
                      603   636   669   702   859
                      882
     check5 = 06CB    969   482   515   548   581
                      614   647   680   713   866
                      899   910
 cheq1_off = 0138     281
  cheq1_on = 00ED     246
 cheq2_off = 0141     285   283
  cheq2_on = 00F6     250   248
 cheq3_off = 014A     289   287
```

cheq3_on = 00FF	254	252				
cheq4_off = 0153	293	291				
cheq4_on = 0108	258	256				
cheq5_off = 015C	297	295				
cheq5_on = 0111	262	260				
cheq6_off = 0165	301	299				
cheq6_on = 011A	266	264				
cheq7_off = 016E	305	303				
cheq7_on = 0123	270	268				
cheq8_off = 0177	309	307				
cheq8_on = 012C	274	272				
chk2 = 02D8	493	461				
chk3 = 0324	526	494				
chk4 = 0370	559	527				
chk5 = 03BC	592	560				
chk6 = 0408	625	593				
chk7 = 0454	658	626				
chk8 = 04A0	691	659				
chklm = 0693	930	916				
chklow = 0623	876	848				
chkmin = 0666	907	889				
cmp = 0182	317	247	251	255	259	263
	267	271	275	282	286	290
	294	298	302	306	310	
cmp1 = 0194	330	326	328			
cmp2 = 0193	329	321				
cmp_off = 0136	280	241				
cmp_on = 00EB	245	240				
cntrwd = E0E3	5	136				
compare = 00DF	237	148				
con1 = 02B3	476	466				
con2 = 02FF	509	499				
con3 = 034B	542	532				

```
      con4 = 0397      575   565
      con5 = 03E3      608   598
      con6 = 042F      641   631
      con7 = 047B      674   664
      con8 = 04C7      707   697
   control = 0196      334   149
    creg_d = E04D       19   132
    creg_f = E04F       18   127
  ctrl_eqp = 0020       44   140   142   336
     data1 = 0070       87   441   460   493   526   559
                       592   625
                       658   691
     data2 = 0071       88   443   462   465   467   476
                       495   498   500   509   528   531
                       533   542   561   564   566
                       575   594   597   599   608   627
                       630   632
                       641   660   663   665   674   693
                       696   698   707
     data3 = 0072       89   445   470   473   479   503
                       506   512
                       536   539   545   569   572   578
                       602   605   611
                       635   638   644   668   671   677
                       701   704   710
```

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

```
     data4 = 0073      90   447   481   484   514   517
                      547   550   580   583   613   616
                      646   649   679   682   712
                      715
```

```
 data5 = 0074      91   449   487   520   553   586
                   619   652   685   718
 data6 = 0075      92   451   839   847   850   853
                   856   876   879
 data7 = 0076      93   453   858   863   881   888
                   893   896   907
 data8 = 0077      94   455   865   870   898   901
                   909   915   921   924   930
 data9 = 0078      95   457   873   904   927   936   939
 delay = 00EC     981   989
delayy = 00A4     196   201
disbuf = 0032      43   152   185
disply = 0098     187   206
   dly = 00A6     197   200
   don = 01BB     358   350
  done = 06FE    1001   987   998
    e1 = 05D9     843   841
    e2 = 0604     862   860
    e3 = 0617     869   867
    e4 = 0638     885   883
   end = 0181     315   276   311
  eqp1 = 0000      24   249   284
  eqp2 = 0001      25   253   288
  eqp3 = 0002      26   257   292
  eqp4 = 0003      27   261   296
  eqp5 = 0004      28   265   300
  eqp6 = 0005      29   269   304
  eqp7 = 0006      30   273   308
  eqp8 = 0007      31   277   312
   er1 = 02D1     490   464   472   483
   er2 = 031D     523   497   505   516
   er3 = 0369     556   530   538   549
   er4 = 03B5     589   563   571   582
```

er5 = 0401	622	596	604	615		
er6 = 044D	655	629	637	648		
er7 = 0499	688	662	670	681		
er8 = 04E5	721	695	703	714		
err = 06A8	942	843	852	862	869	885
	895	900	911	923	938	
errflg = 0011	85	464	472	483	497	505
	516	530	538	549	563	571
	582	596	604	615	629	
	637	648	662	670	681	695
	703	714	841	860	867	883
	900	911	950	955	960	
	965	970	975			
error2 = 06B9	955	953				
error3 = 06C8	965	963				
error5 = 06D7	975	973				
first = 0063	152	147				
flg = 0012	86					
fr = 0008	32					
fs = 0009	33					
getdata = 0255	440	400				
go = 073A	1037	1007				
hardtime = 019D	340	113				
hor1 = E044	12	156				
hor10 = E045	13	160				
hour = 0031	42	168	239			
intt = 01EE	388	111				
key = 0061	84	462	470	481	495	503
	514	528	536	547	561	569
	580	594	602	613	627	
	635	646	660	668	679	693
	701	712	839	858	865	881
	898	909	952	962	972	

```
   ladd = 0025      118   109
    lag = 0027      119   123
   lag1 = 002B      121   119
   lag2 = 002D      122

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

  laggg = 0058      147   150
   main = 004C      140
   make = 0069      156
   min1 = E042       14   170
  min10 = E043       15   174
 minute = 0030       41   154   182   238
    mor = 0644      890   888
   more = 05E6      849   847
   morr = 067D      917   915
    ne1 = 05DC      844   842
    ne2 = 0607      863   857   861
    ne3 = 061A      870   868
    ne4 = 063B      886   880   884
 newkey = 0060       83   441   443   445   447   449   451
                    453   455   457   1024
   off1 = 04ED      726   489
   off2 = 0509      740   522
   off3 = 0525      754   555
   off4 = 0541      768   588
   off5 = 055D      782   621
   off6 = 0579      796   654
   off7 = 0595      810   687
   off8 = 05B1      824   720
  offh1 = 0051       68   733
  offh2 = 0053       70   747
```

offh3 = 0055	72	761						
offh4 = 0057	74	775						
offh5 = 0059	76	789						
offh6 = 005B	78	803						
offh7 = 005D	80	817						
offh8 = 005F	82	831						
offm2 = 0052	69	748						
offm3 = 0054	71	762						
offm4 = 0056	73	776						
offm5 = 0058	75	790						
offm6 = 005A	77	804						
offm7 = 005C	79	818						
offm8 = 005E	81	832						
offtime = 05CD	839	732	746	760	774	788	802	
	816	830						
oh1 = 0291	462							
oh2 = 02DD	495							
oh3 = 0329	528							
oh4 = 0375	561							
oh5 = 03C1	594							
oh6 = 040D	627							
oh7 = 0459	660							
oh8 = 04A5	693							
ok = 05EF	853	851						
okk = 0650	896	894						
okkk = 068A	924	922						
okkkk = 06A3	939	937						
om1 = 02BC	481	475						
om2 = 0308	514	508						
om3 = 0354	547	541						
om4 = 03A0	580	574						
om5 = 03EC	613	607						
om6 = 0438	646	640						

```
    om7 = 0484      679   673
    om8 = 04D0      712   706
    one = 0724     1024  1018

8051 Cross-Assembler (1.3)   (C) 1987, 1989 Binary Technology
c:\et-em\project.asm

 onemil = 06EB      990   994
   onh1 = 0041       52   469   474   478   480   490   730
   onh2 = 0043       54   502   507   511   513   523   744
   onh3 = 0045       56   535   540   544   546   556   758
   onh4 = 0047       58   568   573   577   579   589   772
   onh5 = 0049       60   601   606   610   612   622   786
   onh6 = 004B       62   634   639   643   645   655   800
   onh7 = 004D       64   667   672   676   678   688   814
   onh8 = 004F       66   700   705   709   711   721   828
   onm2 = 0042       53   519   521   524   745
   onm3 = 0044       55   552   554   557   759
   onm4 = 0046       57   585   587   590   773
   onm5 = 0048       59   618   620   623   787
   onm6 = 004A       61   651   653   656   801
   onm7 = 004C       63   684   686   689   815
   onm8 = 004E       65   717   719   722   829
    ook = 06E7      988   985
    ov2 = 06BB      956   954
    ov3 = 06CA      966   964
    ov5 = 06D9      976   974
   patt = 00FE       49   153
  porta = E0E0        6   141   335  1025  1031
  portb = E0E1        7   189
  portc = E0E2        8   191
program = 028C      460   401
   ring = 0037       40
```

บรรณานุกรม

กิตติศักดิ์ ผู้พัฒน์ , เปรมจิต วิสุทธิศิริ " เครื่องควบคุมอุปกรณ์ไฟฟ้าโดยโทรศัพท์ STMF " วิทยานิพนธ์ปริญญาตรี ภาควิชาวัดคุมทางอุตสาหกรรม สาขาคอมพิวเตอร์อุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง พ.ศ. 2533

ไอซีน่าสนใจ "MT8870 ไอซีถอดรหัสความถี่โทรศัพท์" เซมิคอลดัคเตอร์ อีเล็คทรอนิกส์ เล่มที่ 88 : 54-60

ทีมงาน อีทีที "RTC ระบบรีลไทม์คล็อก " เซมิคอลดัคเตอร์ อีเล็คทรอนิกส์ เล่มที่98 : หน้า 190-197

ปราโมทย์ จูฑากร : "เข้าใจและใช้งานโซลิดสเตทเลย์" เซมิคอนดัคเตอร์ อีเล็ค-ทรอนิกส์ เล่มที่ 107 : หน้าที่ 104-114

น.ต. ธวัชชัย เลื่อนฉวี : "เทคโนโลยีโทรศัพท์ " ห.จ.ก. ภาพพิมพ์, ครั้งที่ 2 : พ.ศ. 2531