การควบคุมระยะไกลด้วยคลื่นวิทยุ

RADIO REMOTE CONTROL

โดย

นายธนชล	บุญคำ	40013331
นายนิรุตต์	ขุนศรี	40013334

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาวิชาเทคโนโลยีโทรคมนาคม ภาควิชาเทคนิคอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์	การควบคุมระยะไกลด้วยคลื่นวิทยุ		
โดย	นาย ธ่นชล	บุญคำ	40013331
	นาย นิรุตต์	ขุนศรี	40013334
อาจารย์ที่ปรึกษา	ผศ. ประดิษฐ์	วัชรพิบูลย์	
ภาควิชา	เทคนิคอุตสาหกรรม		
ปีการศึกษา	2542		

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

---------- ประธานกรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )

การควบคุมระยะไกลด้วยคลื่นวิทยุ

โดย	นาย ธนชล	บุญคำ
	นาย นิรุตต์	ขุนศรี
อาจารย์ที่ปรึกษา	ผศ. ประดิษฐ์	วัชรพิบูลย์
ปีการศึกษา	2542	

บทคัดย่อ

โครงงานนี้นำเสนอการควบคุมอุปกรณ์ไฟฟ้าในระยะทางไกล โดยใช้คลื่นความถี่วิทยุเป็นตัวนำสัญญาณความถี่คู่ (Dual Tone Multi Frequency : DTMF) สำหรับใช้ควบคุมการทำงานของอุปกรณ์ไฟฟ้าในแต่ละช่องการทำงาน ซึ่งในโครงงานนี้สามารถควบคุมได้ถึง 8 ช่องการทำงาน

โดยโครงงานนี้แบ่งการทำงานเป็นสองส่วน คือ ส่วนฮาร์ดแวร์ (Hardware) ซึ่งประกอบด้วยส่วนกำเนิดสัญญาณ DTMF, การเข้ารหัสสัญญาณ DTMF และการถอดรหัสสัญญาณ DTMF และ ส่วนซอฟต์แวร์ (Software) ซึ่งส่วนซอฟต์แวร์นี้เป็นส่วนที่ใช้สำหรับควบคุมการทำงานของไมโครคอนโทรลเลอร์ 8051 เขียนด้วยโปรแกรมภาษาแอสเซมบลี้

RADIO REMOTE CONTROL

BY Mr. THANACHON BUNKAM
Mr. NIRUTE KHUNSRI

ADVISOR Mr. PRADIT VACHRAPIBOOL

ABSTRACT

This project presence the using of radio frequency modulated by DTMF (Dual Tone Multi Frequency) for controlling the electrical equipments, here by controlling of 8 channels are available.

The project is seperated into two parts, hardware and software. The hardware consist of DTMF signal generator, DTMF encoder and DTMF decoder. The software for 8051 microcontroller used the assembly language.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดีในทุก ๆ ด้าน ก็ด้วยความร่วมมือและการช่วยเหลือเป็นอย่างดีจากท่านอาจารย์และเจ้าหน้าที่ทุก ๆ คนในภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยเฉพาะอย่างยิ่งจากท่านผู้ช่วยศาสตราจารย์ ประดิษฐ์ วัชรพิบูลย์ ซึ่งท่านเป็นอาจารย์ที่ปรึกษาทางด้านโครงงานนี้ของผู้จัดทำ ที่ได้ให้คำปรึกษาและคำแนะนำต่าง ๆ ที่มีประโยชน์อย่างยิ่งต่อการทำโครงงานนี้ให้เป็นผลสำเร็จลุล่วงไปด้วยดีตลอดมา และต้องขอขอบคุณทุก ๆ ท่านที่ได้ให้ความช่วยเหลือเป็นอย่างมาก ซึ่งต้องขออภัยที่ไม่ได้กล่าวนามได้ทั้งหมด โดยทางผู้จัดทำหวังว่าคงจะได้รับความอนุเคราะห์จากทุก ๆ ท่านอีกในโอกาสต่อ ๆ ไป

สุดท้ายนี้ต้องขอกราบขอบพระคุณ คุณพ่อและคุณแม่ที่ได้ให้การสนับสนุนทางด้านทุนทรัพย์ทางด้านการศึกษาและกำลังใจด้วยดีตลอดมา

ผู้จัดทำ

นาย ธนชล บุญคำ

นาย นิรุตต์ ขุนศรี

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในสภาวะปัจจุบันเทคโนโลยีต่าง ๆ ได้เจริญก้าวหน้าอย่างรวดเร็ว และมีการนำสัญญาณความถี่ (Frequency Signal) มาใช้อย่างแพร่หลาย ซึ่งสัญญาณความถี่คู่ (Dual Tone Multi Frequency : DTMF) ก็เป็นคุณสมบัติหนึ่งที่ถูกนำมาใช้ให้ก่อเกิดประโยชน์อย่างมากมาย เพื่อสร้างความสะดวกสบายในการดำเนินชีวิตประจำวันของมนุษย์ให้มากขึ้น เช่น ในด้านการสื่อสารแบบไร้สายหรือในด้านสื่อสารแบบสายนำสัญญาณเป็นระยะทางไกลซึ่งต้องสิ้นเปลืองค่าใช้จ่ายด้วยข้อจำกัดทางด้านระยะทาง

นอกจากนั้นในสภาพปัจจุบันการทำงานของมนุษย์มีอัตราเสี่ยงต่ออันตรายที่จะเกิดขึ้น เพราะการทำงานในที่สูงหรือในโรงงานที่มีการควบคุมใกล้กับเครื่องจักร อาจทำให้เกิดอันตรายต่อผู้ใช้งานได้ จากปัญหาที่กล่าวมาจึงทำให้เกิดแนวความคิดที่ว่า ถ้าเรามีการควบคุมอุปกรณ์หรือเครื่องจักรเหล่านั้น โดยที่ผู้ใช้งานอยู่ห่างจากอันตรายที่จะเกิดขึ้นก็จะทำให้ผู้ใช้งานมีความปลอดภัยในการทำงานที่สูงขึ้น จึงได้เกิดแนวความคิดในการสร้างเครื่องควบคุมระยะไกลด้วยคลื่นวิทยุขึ้นมาใช้ประโยชน์

1.2 วัตถุประสงค์ของการศึกษา

1.2.1 เพื่อศึกษาและนำไมโครคอนโทรลเลอร์ตระกูล 8051 มาใช้ประโยชน์

1.2.2 เพื่อเป็นการศึกษาและสามารถใช้วงจรถอดรหัสสัญญาณ DTMF ให้เป็นรหัสเลขฐานสอง (Binary Code Decimal Notation :BCD) โดยใช้วงจรรวม (Integrate Circuit :IC) เบอร์ MT8870

1.2.3 เพื่อศึกษาการใช้โปรแกรมภาษาแอสเซมบลี้ (Assembly) ในการควบคุมและสั่งงาน

1.2.4 ศึกษาถึงการใช้สัญญาณ DTMF ในการสื่อสาร

1.3 ขอบเขตของการทำโครงงานปริญญานิพนธ์

1.3.1 สร้างเครื่องควบคุมที่สามารถเปิด-ปิดอุปกรณ์ไฟฟ้าได้ไม่น้อยกว่า 8 ช่องการทำงาน

1.3.2 สามารควบคุมได้ในรัศมีที่สายตาคนมองเห็นได้

1.3.3 ใช้ความถี่คลื่นวิทยุ (Radio Wave) จากวิทยุรับส่งได้ทุกชนิด

1.3.4 โปรแกรมควบคุมใช้ภาษาแอสเซมบลี้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 เพื่อเป็นการสร้างเครื่องมือสื่อสารที่ก่อให้เกิดความสะดวกสบาย ไว้ใช้ประโยชน์ในชีวิตประจำวัน

1.4.2 ได้ศึกษาถึงการส่งสัญญาณวิทยุ

1.4.3 ได้เรียนรู้ถึงการใช้โปรแกรมคอมพิวเตอร์ในการควบคุมและสั่งงาน

1.4.4 เพื่อให้เกิดแนวคิดในการประยุกต์และพัฒนาในการสร้างเครื่องมือสื่อสาร

1.5 ขั้นตอนการดำเนินงานของโครงงาน

1.5.1 สร้างแนวความคิดโครงงาน

1.5.2 รวบรวมข้อมูลที่จะนำมาใช้ในโครงงาน

1.5.3 ออกแบบวงจรและเขียนโปรแกรมการทำงาน

1.5.4 ทดสอบการทำงาน

1.5.5 ออกแบบและสร้างแผ่นวงจรพิมพ์

1.5.6 ทำการลงอุปกรณ์

1.5.7 ทดสอบการทำงาน ถ้ายังใช้งานไม่ได้ต้องทำการแก้ไขข้อบกพร่อง

1.5.8 ทำเอกสาร หลังจากทดสอบการทำงานแล้ว

บทที่ 2
ทฤษฎีที่ใช้ในโครงงาน

2.1 การนำ ไมโครคอนโทรลเลอร์ 8051 มาใช้ประโยชน์

การเชื่อมต่อกับหน่วยความจำภายนอกนั้น เนื่องจากไมโครคอนโทรลเลอร์ 8051 มีระบบบัสแอสเดรสและบัสข้อมูล เป็นลักษณะแบบใช้การมัลติเพล็กซ์จากพอร์ตเดียวกัน คือ ในระยะเวลาเริ่มต้นเส้นสัญญาณของพอร์ตเหล่านี้จะใช้ในการส่งค่าแอสเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงจะเปลี่ยนเป็นสภาวะอิมพีแดนซ์สูง (High Impedance) เพื่อใช้งานในฐานะของบัสข้อมูล แต่เนื่องจากว่า EEPROM (Electrical Erasable Programable Read Only Memory) ที่ใช้งานทั่วไปนั้น ไม่ใช้การมัลติเพล็กซ์และมีขาสัญญาณบัสแอสเดรสและบัสข้อมูลแยกออกจากกันโดยชัดเจน ดังนั้นการเชื่อมต่อ EEPROM เพื่อทำหน้าที่เป็นหน่วยความจำโปรแกรม จึงจำเป็นต้องมีโปรแกรมประเภทแลทช์ (Latch) ประกอบเพิ่มเติม เพื่อทำการค้างค่าของแอสเดรสที่ส่งออกมาจากไมโครคอนโทรลเลอร์ 8051 ในช่วงเวลาแรกให้กับขาสัญญาณแอสเดรสของ EEPROM ต่อไป

สัญญาณ	คำจำกัดความ	ขาสัญญาณ	หน้าที่
EA	External Access	31	เลือกประเภทหน่วยความจำภายในหรือภายนอก
ALE	Address Enable	30	สัญญาณเอาต์พุต (OUTPUT) สำหรับการแลทช์ข้อมูลแอดเดรสบัส
P2.0 – P2.3	Port 2	21-28	เป็นข้อมูลแอสเดรสไบต์สูงของหน่วยความจำ
P0.0 – P0.7	Port 0	32-39	มัลติเพล็กซ์สัญญาณของบัสแอสเดรสและบัสข้อมูล
PSEN	Program Store Enable	29	สัญญาณระบุการ Read ให้กับหน่วยความจำ EPROM

ตารางที่ 2.1 สัญญาณของ 8051 ที่ใช้ระหว่างการติดต่อ เพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

สัญญาณ EA (External Access) ใช้ในการกำหนดว่าจะอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ซึ่งหากเป็นระดับลอจิกต่ำจะอ่านข้อมูลจากหน่วยความจำภายนอกและในกรณีเป็นลอจิกสูงจะอ่านข้อมูลจากหน่วยความจำภายใน

2.2 คำสั่งที่สำคัญของไมโครคอนโทรลเลอร์ 8051

ACALL ทำหน้าที่เรียกโปรแกรมย่อยในแอดเดรสที่กำหนดและจะเพิ่มค่าใน PC สองครั้ง แล้วส่งผลลัพธ์ใน PC ไปเก็บไว้ในสแตกแล้วเพิ่มค่าตัวชี้สแตกสองครั้ง

ADD ทำหน้าที่บวกตัวแปรจากไบต์ที่กำหนดกับค่าที่อยู่ในแอกคูมิวเลเตอร์ ผลลัพธ์ที่ได้จะเก็บไว้ในแอกคูมิวเลเตอร์และจะมีผลกระทบกับแฟลกที่เกี่ยวข้อง

ADDC ทำหน้าที่บวกตัวแปรที่กำหนดกับค่าแอกคูมิวเลเตอร์พร้อมทั้งแฟลกทด ผลลัพธ์ที่ได้จะเก็บไว้ในแอกคูมิวเลเตอร์ และจะมีผลกับแฟลกตัวทด (carry flag) และแฟลกช่วยเหลือ (auxiliary flag)

AJMP เป็นคำสั่งให้ทำการกระโดดการทำงานของโปรแกรมไปยังแอสเดรสที่กำหนด

ANL จะทำงานตรรก AND ในแต่ละไบต์ระหว่างตัวแปรที่กำหนด

CJNE ทำหน้าที่เปรียบเทียบข้อมูลของโอเปอร์แรนด์สองตัวเลข ถ้าพบว่าไม่เท่ากันก็จะกระโดดไปยังแอสเดรสที่ได้จากการรวมของค่าสัมพันธ์ที่คิดเครื่องหมายของไบต์ที่สามของชุด

CLR bit ค่าบิตที่ถูกกำหนดจะถูกรีเซตให้เป็นศูนย์ โดยที่ไม่มีค่าผลกระทบต่อค่าแฟลกใด ๆ

CPL A จะทำหน้าที่ให้แอกคูมิวเลเตอร์เป็นหนึ่งคอมพลีเมนต์ (one's complement) หรือกลับค่า

CPL bit จะทำหน้าที่หนึ่งคอมพลีเมนต์ที่บิตที่กำหนดหรือบิตที่กำหนดเลขที่โดยตรง

DAA ทำหน้าที่ปรับค่าในแอกคูมิวเลเตอร์เป็นเลขฐานสิบ หลังทำคำสั่ง ADD หรือ ADDC ด้วยการเพิ่มค่า 610 เข้าไปเพื่อให้ได้ค่า BCD ที่เหมาะสมการบวกภายในแฟลกตัวทดจะเซ็ต

DEC ทำหน้าที่ลดค่าตัวแปรที่กำหนดลงไป 1 ค่า โดยไม่มีผลกระทบต่อแฟลกใด ๆ สามารถใช้กับแอกคูมิวเลเตอร์รีจีสเตอร์ค่าตำแหน่งหรือรีจีสเตอร์โดยอ้อม

DIV AB เป็นการหารตัวเลขแบบลงตัว 8 บิต ในแอกคูมิวเลเตอร์ด้วยค่าตัวเลขลงตัวในรีจีสเตอร์ B โดยไม่คิดเครื่องหมาย ผลหารที่ได้จะเก็บไว้ที่แอกคูมิวเลเตอร์และถ้ามีเศษจะเก็บไว้ที่ B แฟลกตัวทดกับ OVERFLOW จะเคลียร์

DJNZ เป็นการลดค่าตัวแปรที่ถูกกำหนดลง 1 ค่า ถ้าผลลัพธ์ไม่เป็นศูนย์ก็จะกระโดดไปยังแอสเดรสที่ถูกกำหนดจากโอเปอร์แรนด์ตัวที่สองหรือรหัสตัวสุดท้ายของคำสั่งที่คิดเครื่องหมายกับค่าในโปรแกรมเคาน์เตอร์ที่ชี้ตำแหน่งของคำสั่งชุดต่อมา แต่ถ้ามีค่าเป็น 00H ก็จะถูกลดเป็น UNDERFLOW มีค่า 0FFH โดยที่ไม่มีผลต่อแฟลกใด ๆ

INC เป็นการเพิ่มค่าของตัวแปรที่ถูกกำหนดให้เป็น 1 ถ้าค่าต้นเป็น 0FFH จะเปลี่ยนเป็น 00H โดยไม่มีการเปลี่ยนแปลงของแฟลกใช้กับรีจีสเตอร์ การบอกตำแหน่งโดยตรงและรีจีสเตอร์โดยอ้อม

INC DPTR เป็นการเพิ่มค่าของรีจีสเตอร์ข้อมูลขนาด 16 บิตให้มีค่าเพิ่มขึ้น 1 ค่า โดย OVERFLOW ที่เกิดจากนัยต่ำที่เพิ่มจาก 0FFH เป็น 00H จะไปเพิ่มค่าในไบต์นัยสูง โดยไม่มีผลต่อแฟลกใด ๆ

JB เป็นคำสั่งให้กระโดดไปยังตำแหน่งที่กำหนด ถ้าตำแหน่งที่กำหนดมีค่าเป็น 1 และไม่มีผลต่อแฟลกใด ๆ

JBC เป็นคำสั่งให้กระโดดไปยังตำแหน่งที่กำหนด ถ้าตำแหน่งที่กำหนดมีค่าเป็น 1 และจะทำการเคลียร์บิตนั้นโดยไม่มีผลต่อแฟลกใด ๆ

JC จะกระโดดถ้าบิตทดเป็น 1 โดยกระโดดไปยังตำแหน่งที่กำหนดในคำสั่ง โดยไม่มีผลต่อแฟลกใด ๆ

JMP เป็นคำสั่งให้กระโดดโดยไม่มีเงื่อนไข โดยกระโดดไปตามค่าผลลัพธ์ที่ได้จากการบวกค่าในแอกคูมิวเลเตอร์กับ DPTR ขนาด 16 บิต โดยที่ค่าของแอกคูมิวเลเตอร์และ DPTR ไม่มีการเปลี่ยนแปลงและไม่มีผลกระทบต่อแฟลกใด ๆ

JNB จะกระโดดไปยังแอดเดรสที่กำหนด ถ้าบิตที่กำหนดไม่เป็น 0 โดยไม่มีผลต่อแฟลกใด ๆ

JNC จะกระโดดไปยังแอดเดรสที่กำหนด ถ้าบิตทดเป็น 0 โดยไม่มีผลกระทบต่อแฟลกใด ๆ

JNZ จะกระโดดไปยังแอดเดรสที่กำหนด ถ้าค่าในแอกคูมิวเลเตอร์ไม่เป็น 0 โดยไม่มีผลต่อแฟลกใด ๆ

JZ จะกระโดดไปยังแอดเดรสที่กำหนด ถ้าค่าในแอกคูมิวเลเตอร์ไม่เป็น 0 หมดโดยไม่มีผลต่อแฟลกใด ๆ

LCALL จะเรียกโปรแกรมย่อยตามตำแหน่งที่กำหนดในคำสั่ง โดยจะเพิ่มค่าโปรแกรมเคาน์เตอร์ขึ้นอีก 3 ค่าเพื่อให้โปรแกรมเคาน์เตอร์ชี้คำสั่งชุดต่อมา แล้วจึงส่งผลลัพธ์ของโปรแกรมเคาน์เตอร์ที่ได้ไปเก็บไว้ที่หน่วยความจำบริเวณสแตก โดยไม่มีผลต่อแฟลกใด ๆ

LJMP เป็นการกระโดดไปยังค่าแอดเดรสที่ถูกกำหนดโดยไม่มีเงื่อนไข โดยการโหลดค่าไบต์สูงและไบต์ต่ำด้วยรหัสคำสั่งของไบต์ที่ 2 และ 3 เข้า PC โดยที่ไม่มีผลต่อแฟลกใด ๆ

MOV เป็นการย้ายตัวแปรที่กำหนดมายังตำแหน่งที่กำหนดไว้ โดยไม่มีผลต่อรีจีสเตอร์ตัวอื่นและค่าแฟลกใด ๆ สามารถกำหนดโหมดเลขที่อยู่ได้ถึง 15 แบบ

MOVC เป็นการโหลดคำสั่งหรือค่าคงที่จากหน่วยความจำโปรแกรมด้วยตำแหน่งที่ชี้ที่อยู่ของรีจีสเตอร์ ซึ่งได้จากการรวมกันของค่าจำนวนเต็มที่อยู่ในแอกคูมิวเลเตอร์กับค่าที่อยู่ในรีจีสเตอร์ 16 บิต

MOVX จะทำการย้ายข้อมูลระหว่าง ACC กับหน่วยความจำภายนอกโดยคำสั่งมี 2 แบบ เพื่อสามารถบอกตำแหน่งได้ทั้ง 8 บิต และ 16 บิต

NOP เป็นคำสั่งให้ทำการ execute ให้ต่อเนื่องกันไปโดยไม่มีแฟลกใด ๆ หรือรีจีสเตอร์ใด ๆ กระทบกระเทือน นอกจาก PC

OLR จะเป็นคำสั่งตรรก OR ระหว่างตัวแปรที่ถูกกำหนดในคำสั่งและเก็บผลลัพธ์ไว้ในที่ๆ กำหนดโดยไม่มีผลกระทบต่อแฟลกใด ๆ

POP จะอ่านค่าข้อมูลของ RAM ภายในที่กำหนดตำแหน่งด้วยตัวชี้สแตก โดยเป็นการกำหนดตำแหน่งโดยตรงและตัวชี้สแตกจะถูกลดค่าลงหนึ่งค่าหลังการอ่านข้อมูล โดยที่ไม่มีผลกระทบต่อค่าแฟลกใด ๆ

PUSH จะเพิ่มค่าตัวชี้สแตกขึ้นหนึ่งค่า แล้วลอกข้อมูลที่ถูกกำหนดด้วยเลขที่อยู่โดยตรงในคำสั่ง เข้าไปยัง RAM ภายในที่ถูกชี้ด้วยค่าตัวชี้สแตกที่ถูกเพิ่มค่าแล้ว โดยไม่มีผลต่อแฟลกใด ๆ

RET จะอ่านค่าข้อมูลของ RAM ภายในที่ไบต์สูงและไบต์ต่ำของ PC ที่เก็บอยู่ในบริเวณสแตกด้วยการลดค่าตัวชี้สแตกลง 2 ค่า โปรแกรมก็จะคืนค่าของ PC เพื่อทำงานคำสั่งที่ต่อจากตัวคำสั่ง ACALL หรือ LCALL ได้โดยทันที โดยไม่มีผลกระทบต่อแฟลก ใด ๆ

RETI จะ POP ไบต์สูงและไบต์ต่ำของ PC ที่เก็บอยู่ในสแตกด้วยการลดค่าตัวชี้สแตก 2 ครั้ง และค่าทางตรรกอินเตอร์รัพต์อีนาเบิลทางระดับความสำคัญจะคืนกลับ ทำให้สามารถทำการอินเตอร์รัพต์ตามความสำคัญที่ตั้งไว้ต่อไปได้ โดยไม่มีผลกระทบใด ๆ

RLA ค่า 8 บิตในแอกคูมิวเลเตอร์จะถูกวนกลับไปทางซ้ายโดยที่ตำแหน่งบิต 7 จะวนกลับมาที่ตำแหน่งบิต 0 โดยไม่มีผลกระทบต่อค่าแฟลกใด ๆ

RLC A จำนวนบิต 8 บิตในแอกคูมิวเลเตอร์รวมทั้งบิตทดใน PSW จะถูกวนกลับไปทางซ้าย โดยที่ตำแหน่งบิต 7 จะวนเข้าบิตทดและบิตทดจะวนเข้าที่ตำแหน่งบิต 0 โดยไม่มีผลกระทบต่อแฟลกใด ๆ

RR A จะเลื่อนแอกคูมิวเลเตอร์จำนวน 8 บิต ไปทางขวา โดยที่ตำแหน่งบิต 0 จะวนเข้าที่ตำแหน่งบิต 7 และไม่มีผลกระทบต่อค่าแฟลกใด ๆ

RRC A จะเลื่อนแอกคูมิวเลเตอร์จำนวน 8 บิต รวมทั้งบิตทดใน PSU จะถูกวนกลับไปทางขวา โดยที่ตำแหน่งบิต 0 จะวนเข้าที่บิตทด และบิตทดจะวนเข้าที่ตำแหน่งบิต 7 โดยไม่มีผลกระทบต่อแฟลกใด ๆ

SETB จะทำการเซตบิตที่กำหนดให้เป็น 1 สามารถใช้คำสั่งนี้กระทำต่อบิตทดหรือบิตที่ถูกกำหนดเลขที่อยู่โดยตรงใด ๆ ได้ โดยไม่มีผลกระทบต่อแฟลกใด ๆ

SJMP เป็นคำสั่งให้กระโดดไปยังตำแหน่งที่กำหนดโดยไม่มีเงื่อนไข โดยจะกระโดดไปตามค่าแอดเดรสที่ได้จากการบวกเอาข้อมูลที่แบบ sign displacement ของรหัสคำสั่งไบต์ที่ 2 กับตัวชี้โปรแกรม PC ที่มีค่าแอดเดรสที่ชี้ไบต์แรกของคำสั่งต่อมา แล้วเก็บไว้ที่ PC ดังนั้นตำแหน่งของคำสั่งที่กระโดดไปจะอยู่ข้างหน้าได้ไม่เกิน 127 ไบต์ หรือข้างหลังไม่เกิน 128 ไบต์

SUBB เป็นคำสั่งลบตัวแปรที่กำหนดและบิตทดออกจากค่าในแอกคูมิวเลเตอร์ และจะเซ็ตค่าบิตทดถ้ามีการยืมค่าจากบิตที่ 7 และจะเคลียร์ถ้าไม่มีการยืมค่า

SWAP A เป็นคำสั่งแลกเปลี่ยนข้อมูลขนาด 4 บิต ภายใน ACC คือบิต 3-0 กับบิต 7-4 โดยไม่มีผลกระทบต่อแฟลกใด ๆ

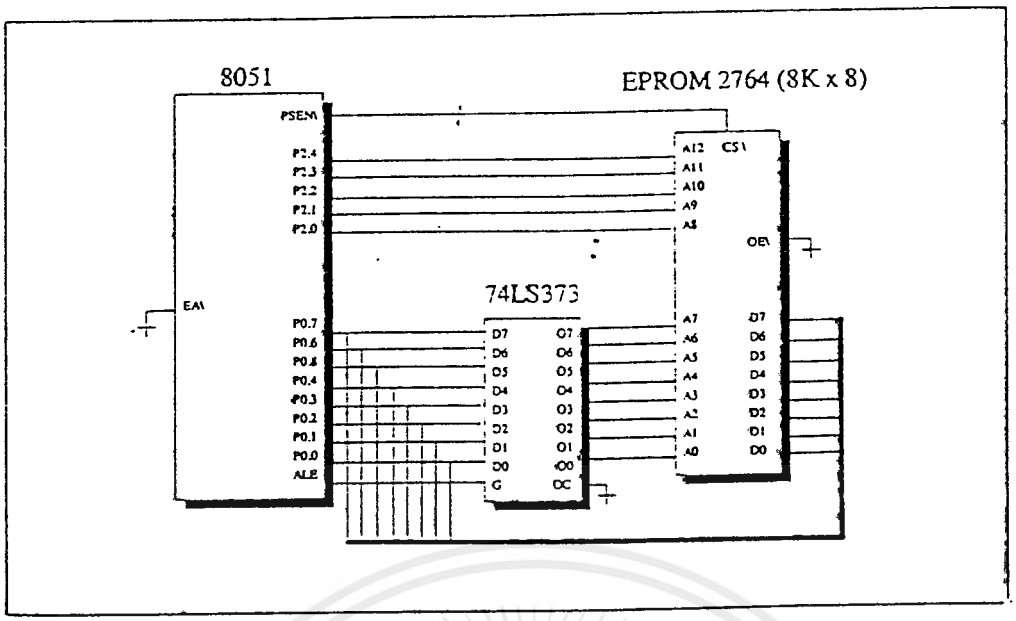
XCH จะโหลดข้อมูลที่กำหนดมาเก็บไว้ใน ACC และขณะเดียวกันก็จะโหลดค่าของ ACC เก็บในตำแหน่งที่ตัวแปรกำหนด สามารถทำงานได้ในโหมดรีจีสเตอร์โดยตรงและรีจีสเตอร์โดยอ้อม

XCHD เป็นการสลับค่า 4 บิต ระหว่างนิบเบิลนัยต่ำของค่าข้อมูลของ ACC โดยทั่วไปจะเป็นการแทนค่าตัวเลข BCD กับค่าข้อมูลนิบเบิลนัยต่ำที่เก็บอยู่ใน RAM ภายใน โดยการกำหนดที่อยู่โดยรีจีสเตอร์

XRL เป็นการแสดงการทำ EX-OR ระหว่างตัวแปรในโหมดโดยอ้อมกับค่าที่เก็บผลลัพธ์ ไว้ที่ ๆ กำหนด โดยไม่มีผลต่อแฟลกใด ๆ

การใช้งานไมโครคอนโทรเลอร์ 8051 ที่ไม่มีหน่วยความจำภายในนั้น จำเป็นต้องเชื่อมต่อเข้ากับหน่วยความจำโปรแกรมซึ่งเป็นไอซี EPROM และจะต้องกำหนดให้เริ่มต้นที่แอสเดรส 0000H เสมอ ทั้งนี้เพราะเมื่อมีการรีเซ็ตหรือเริ่มต้นการจ่ายไฟฟ้าให้กับระบบไมโครคอนโทรลเลอร์ 8051 จะได้เริ่มต้นทำงานตามคำสั่งที่แอสเดรสนี้ทันที

จากในรูปที่ 2.1 เป็นการเชื่อมต่อ 8051 เข้ากับ EPROM ขนาด 8K x 8 บิต เบอร์ 2764 โดยใช้สัญญาณ CS เชื่อมต่อโดยตรงเข้ากับสัญญาณ PSEN ของ 8051 ซึ่งจะทำให้ไม่ว่าจะทำคำสั่งใด ๆ ที่เกี่ยวข้องกับหน่วยความจำโปรแกรมแล้วก็จะเป็นการเลือกให้ EPROM ทำงานตลอดเวลา



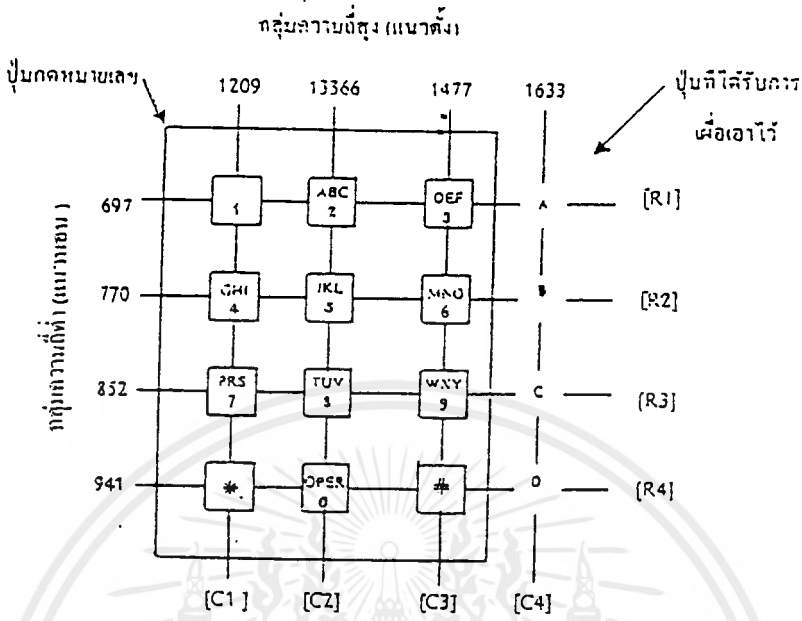
รูปที่ 2.1 การเชื่อมต่อ EPROM เข้ากับไมโครคอนโทรลเลอร์ 8051

2.3 ระบบโทรศัพท์แบบส่งสัญญาณความถี่คู่

เป็นระบบการส่งสัญญาณอีกแบบหนึ่ง ซึ่งใช้มากกว่าระบบการส่งเป็นสัญญาณแบบพัลส์ ระบบนี้หรือเรียกชื่อย่อว่า DTMF มีวิธีการส่งหมายเลขของผู้ที่ต้องการจะติดต่อ โดยการส่งสัญญาณความถี่ 2 ความถี่มอดูเลตกันไป ซึ่งจะเป็นตัวแทนของหมายเลขที่กด ซึ่งความถี่ที่ส่งออกไปจะอยู่ในย่านของเสียงพูด (0.4 กิโลเฮิรตซ์) ซึ่งค่าความถี่ที่ต่ำกว่าจะเป็นความถี่ที่แสดงในแนวนอนและอีกค่าหนึ่งก็จะเป็นความถี่ในแนวตั้ง ซึ่งจะแสดงค่าไว้ในรูปที่ 2.2

2.4 ข้อเปรียบเทียบระหว่างโทรศัพท์แบบหมุนกับระบบ DTMF

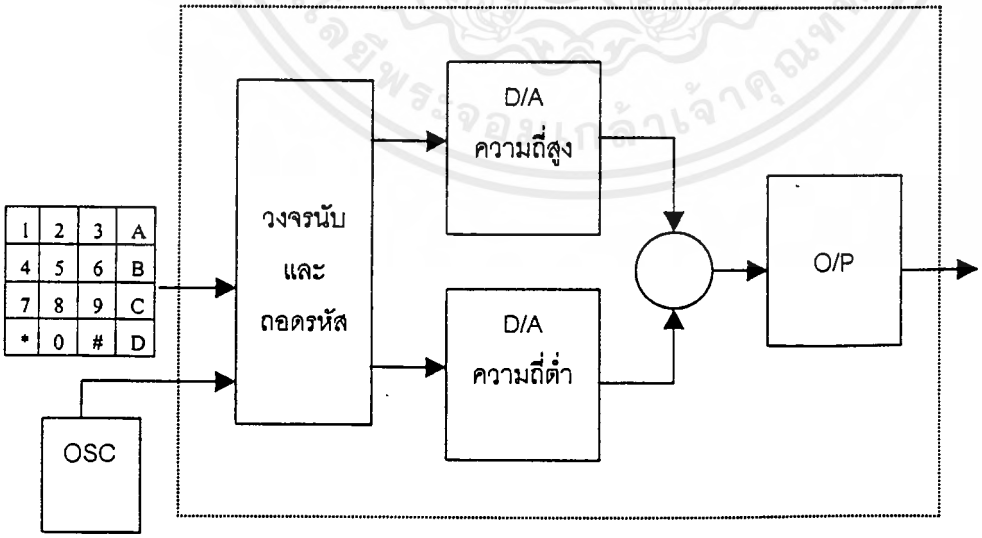
ในการส่งสัญญาณแบบพัลส์ 1 ลูก จะต้องใช้เวลาอย่างน้อย 100 มิลลิวินาที (60 วินาทีสำหรับช่วงการเปิดวงจร และ 40 วินาทีสำหรับช่วงเวลาการปิดวงจร) และยังต้องมีช่วงเวลาที่แยกสัญญาณแต่ละกลุ่มออกอีกอย่างน้อย 700 มิลลิวินาที และยิ่งถ้าหมายเลขที่ต้องการติดต่อด้วยมีค่ามาก และยาวมากขึ้นเท่าใด ย่อมต้องทำให้เสียเวลาในการส่งสัญญาณมากยิ่งขึ้น เช่น หมายเลข 555-5555 จะใช้เวลาในการส่งสัญญาณ = 5 (พัลส์/มิลลิวินาที) x 100 (มิลลิวินาที/พัลส์) x 7 = 3.5 วินาที และระยะเวลาของช่องว่างระหว่างกลุ่มสัญญาณ = 700(มิลลิวินาที) x 6 = 4.2 วินาที ดังนั้นจะใช้เวลาในการส่งทั้งหมด = 3.5 + 4.2 = 7.7วินาที แต่ถ้าเป็นโทรศัพท์ที่ใช้การส่งระบบ DTMF จะสามารถประหยัดเวลาในการส่งหมายเลขไปยังชุมสายได้มากกว่าระบบที่ใช้การส่งสัญญาณพัลส์ ซึ่งเป็นผลให้ชุมสายสามารถใช้อุปกรณ์ประเภทหน่วยความจำได้อย่างมีประสิทธิภาพมากขึ้น



รูปที่ 2.2 แป้นกดหมายเลขและค่าความถี่ในแนวนอนและแนวตั้ง

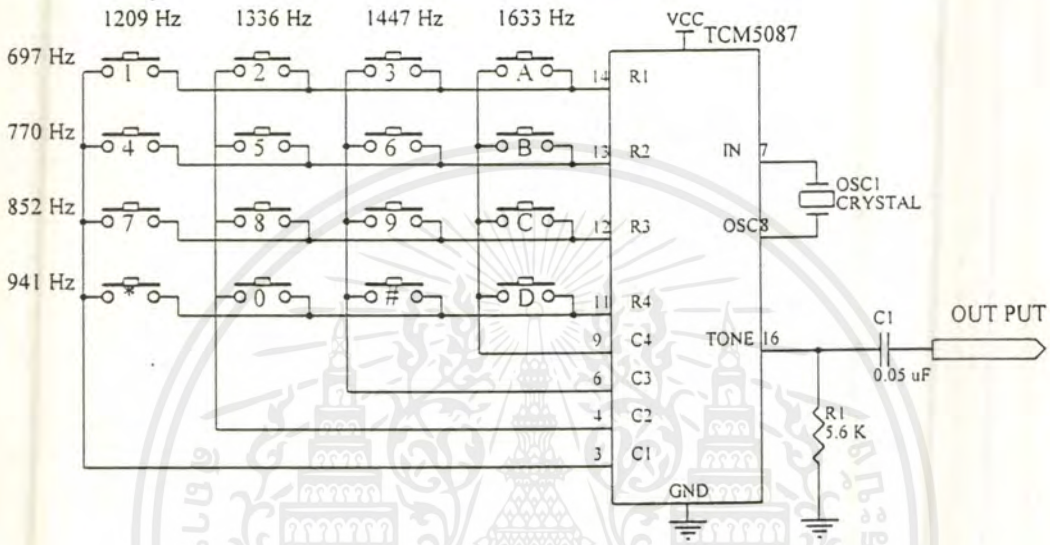
2.5 ระบบการส่งสัญญาณ DTMF

การกำเนิดสัญญาณ DTMF นั้น จะเป็น IC สำเร็จรูป ซึ่งมีใช้งานอยู่หลายเบอร์ เช่น UM9559 หรือ UM91215A ซึ่งเป็นเบอร์ที่เรานำมาใช้งาน โดยมีหลักการทำงานคือ



รูปที่ 2.3 วงจรภายในของไอซีกำเนิดสัญญาณ DTMF เบอร์ UM91215A

วงจรภายในจะประกอบด้วย วงจรนับและถอดรหัส ซึ่งจะแยกแยะการกดหมายเลขในแต่ละครั้งว่าจะตรงกับตำแหน่งใดในแนวแถวและแนวหลัก แล้วจึงนำความถี่ทั้งสองไปหารความถี่หลัก ทำให้ได้สัญญาณที่ออกจากวงจรนับ และถอดรหัสเป็นสัญญาณดิจิตอล 2 สัญญาณ ที่มีความถี่แตกต่างกัน แล้วจึงนำไปแปลงเป็นสัญญาณอะนาล็อก ก่อนที่จะนำสัญญาณทั้งสองมารวมกันเพื่อส่งต่อให้กับภาคเอาท์พุต



รูปที่ 2.4 วงจรกำเนิดสัญญาณ DTMF

หมายเลขแป้นที่กด	ความถี่ด้านต่ำ	ความถี่ด้านสูง	เอาท์พุทไบนารี่
1	697 Hz	1209 Hz	0001
2	697 Hz	1336 Hz	0010
3	697 Hz	1447 Hz	0011
4	770 Hz	1209 Hz	0100
5	770 Hz	1336 Hz	0101
6	770 Hz	1447 Hz	0110
7	852 Hz	1209 Hz	0111
8	852 Hz	1336 Hz	1000
9	852 Hz	1447 Hz	1001
0	914 Hz	1336 Hz	1010

ตารางที่ 2.2 แสดงการเปลี่ยนค่าความถี่ DTMF ไปเป็นเลขฐานสอง

หมายเลขแป้นที่กด	ความถี่ด้านต่ำ	ความถี่ด้านสูง	เอาท์พุทไบนารี่
*	914 Hz	1209 Hz	1011
#	914 Hz	1447 Hz	1100
A	697 Hz	1633 Hz	1101
B	770 Hz	1633 Hz	1110
C	852 Hz	1633 Hz	1111
D	941 Hz	1633 Hz	0000

ตารางที่ 2.2 (ต่อ) แสดงการเปลี่ยนค่าความถี่ DTMF ไปเป็นเลขฐานสอง

2.6 การถอดรหัสความถี่ทางโทรศัพท์ชนิดกดปุ่ม

การถอดรหัสความถี่โทรศัพท์คือการแปลงสัญญาณที่เกิดจากการกดปุ่มตัวเลขของโทรศัพท์ชนิดกดปุ่ม (ชนิด TONE หรือ DTMF) ให้เป็นระบบตัวเลขทางดิจิตอล ซึ่งใช้ไอซี MT8870 แปลงความถี่ทางโทรศัพท์ให้เป็นระบบตัวเลขฐานสองขนาด 4 บิต ซึ่งมีรายละเอียด ดังนี้

2.6.1 คุณสมบัติพิเศษเฉพาะตัวของ MT 8870

1. เป็นตัวรับและถอดรหัสความถี่ (DTMF RECEIVER)
2. กินกระแสไฟน้อย ใช้ไฟเลี้ยงระดับเดียวกับไอซี TTL
3. สามารถตั้งอัตราการขยายภายในตัวไอซีได้
4. สามารถปรับช่วงคาบเวลาของความถี่ได้เป็นไอซีที่มีคุณภาพสูง
5. เป็นไอซีที่มีคุณภาพสูง

2.6.2 โครงสร้างของไอซี MT 8870

โครงสร้างภายในของ MT 8870 ประกอบด้วยวงจรกรองความถี่ และวงจรถอดรหัสทางดิจิตอลเป็นไอซีที่สร้างโดยใช้เทคโนโลยี IOS2-CMOSในส่วนของวงจรกรองความถี่ใช้เทคนิคของสวิตซ์คาปาซิเตอร์ สำหรับกรองความถี่สูงและกรองความถี่ต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอลเพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ออกเป็นเลขฐานขนาด 4 บิต และตรวจสอบช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุตเป็นออปแอมป์ ซึ่งสามารถปรับอัตราการขยายโดยต่ออุปกรณ์ภายนอก ส่วนเอาต์พุตเป็นวงจรค้าง 3 สถานะ

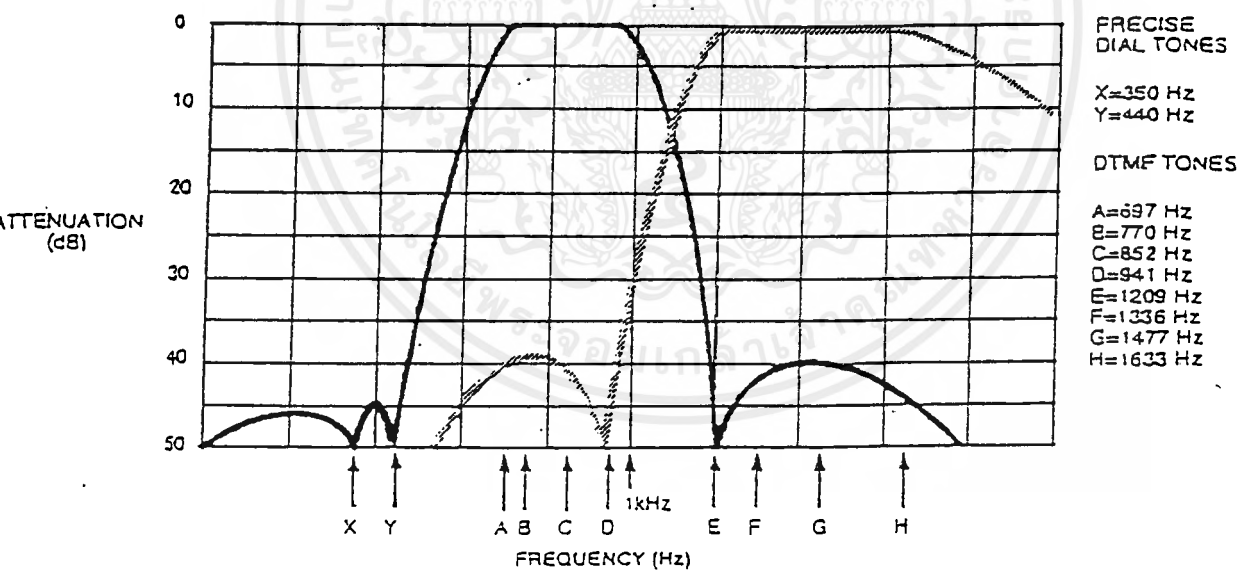
2.6.3 หน้าที่การทำงานภายในไอซี MT 8870

ภายในไอซี MT 8870 ประกอบด้วยส่วนสำคัญ 5 ส่วนคือ

1. ภาคกรองความถี่ (Filter section)
2. ภาคถอดรหัส (Decoder section)
3. ภาคตรวจสอบสัญญาณ (Steerin circuit)
4. ภาคขยายสัญญาณความแตกต่าง (Differential input)
5. ภาคกำเนิดความถี่ (Oscillator)

1. ภาคกรองความถี่

ในส่วนนี้จะแยกสัญญาณ DTMF ที่เข้ามาออกเป็น 2 กลุ่มความถี่ คือ ช่วงความถี่สูง และความถี่ต่ำ โดยใช้วงจรความถี่อันดับ 6 ชนิด สวิตซ์คาปาเตอร์ (six-order switched capacitor band pass filter) ซึ่งความถี่ที่แยกได้มี 2 ช่วง คือ ช่วงความถี่สูง (High Frequency) และช่วงความถี่ต่ำ (Low Frequency)



รูปที่ 2.5 ความถี่ที่ได้จากภาคกรองความถี่

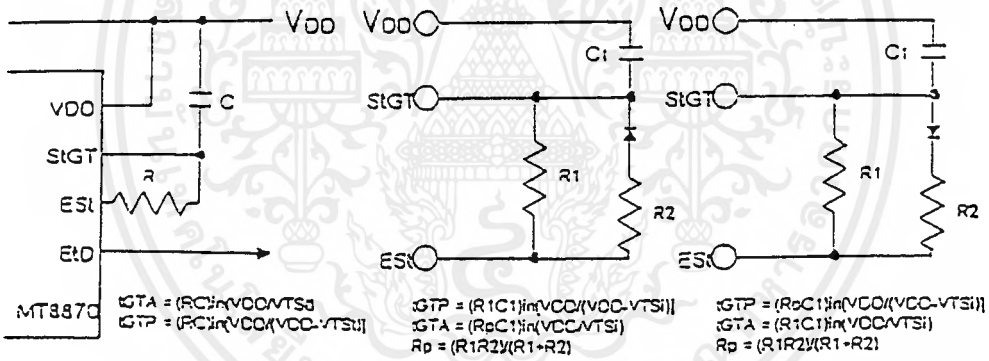
2. ภาคถอดรหัส

ความถี่ DTMF ที่ถูกกรองเรียบร้อยแล้วจะผ่านเข้าวงจรถอดรหัสความถี่ออกเป็นตัวเลข โดยใช้เทคนิคการนับแบบดิจิตอล และมีการตรวจสอบความถี่ที่เข้ามาว่าเป็นความถี่มาตรฐาน DTMF หรือไม่ เพื่อป้องกันความถี่อื่นเข้ามาปน

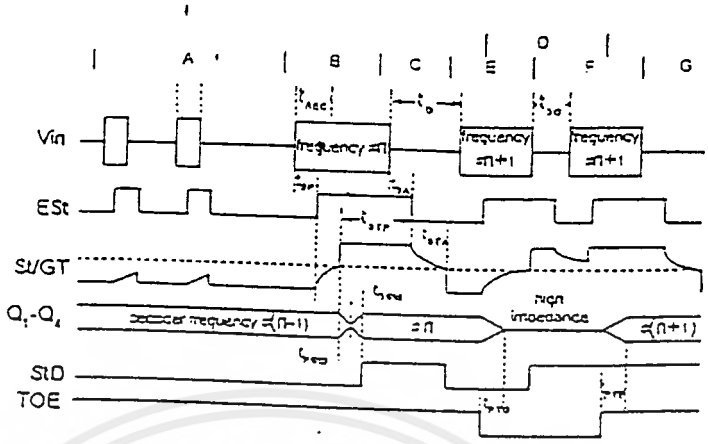
เมื่อตรวจสอบว่าความถี่นั้นถูกต้อง สัญญาณที่ขา (EST ZEARLY STEERING) ก็จะทำงาน

3. ภาคตรวจสอบสัญญาณ

ก่อนที่จะมีการถอดรหัสความถี่ออกไป ที่เฮาต์พุตจะมีการตรวจสอบช่วงความถี่ที่เข้ามามีระยะเวลาตามที่กำหนดหรือไม่ โดยสังเกตจากระยะเวลาการกดปุ่มโทรศัพท์ ซึ่งต้องกดปุ่มให้มีความถี่ออกมาเป็นช่วงเวลาพอสมควร มิฉะนั้นวงจรส่วนนี้จะไม่รับ โดยถือว่าสัญญาณนั้นไม่ถูกต้อง ส่วนช่วงเวลายาวเท่าใดสามารถตั้งได้โดยใช้ RC ภายนอก สัญญาณที่เขา EST จะเป็น 1 นานใกล้เคียงกับระยะเวลาที่มีความถี่ DTMF เข้ามา จากรูปที่ 1.24 เมื่อขา EST เป็น 1 ทำให้ VC สูงขึ้น ตัวเก็บประจุจะจ่ายประจำทำให้แรงดัน VC สูงขึ้นจนถึงค่าเทรสโฮสต์ วงจรถอดรหัสถึงจะถอดรหัสออกเป็นตัวเลขขนาด 4 บิต รายละเอียดการทำงานพิจารณาได้จากแผนภูมิเวลาในรูปที่ 2.6



รูปที่ 2.6 วงจรตรวจสอบสัญญาณอย่างง่าย



รูปที่ 2.7 ผังเวลาของไอซี MT 8870

อธิบายขั้นตอนการทำงาน

A - ตรวจสอบความถี่ที่เข้ามา แต่คาบเวลาไม่ถูกต้อง เอาต์พุตไม่เปลี่ยนแปลง

B - ความถี่ # n ถูกตรวจพบ และมีคาบเวลาที่ถูกต้อง ความถี่ที่ถูกถอดรหัสและค้างไว้ที่เอาต์พุต

C - จบความถี่ # n ช่วงห่างถูกต้อง เอาต์พุตยังคงค้างจนกว่าจะได้รับความถี่ที่ถูกต้องใหม่

D - เอาต์พุตเปลี่ยนเป็นอิมพีแดนซ์สูง

E - ความถี่ # n+1 ถูกตรวจพบคาบเวลาถูกต้อง ความถี่ถูกถอดรหัสและค้างไว้

F - ความถี่ # n+1 หายไป ช่วงห่างไม่ถูกต้อง เอาต์พุตยังคงค้างอยู่

G - ความถี่ # n+1 ช่วงห่างถูกต้อง เอาต์พุตยังค้างอยู่จนถึงความถี่ใหม่

อธิบายศัพท์

Vin - สัญญาณความถี่ DTMF ที่เข้ามา

Est - Easly Sterring output ให้แสดงความถี่ที่ถูกต้อง

St/Gt - Sterring input / Guard Time output สำหรับต่อกับ RC ภายนอก

Q1-Q4 - เอาต์พุต BCD ขนาด 4 บิต

StD - Deleyed Sterring output ใช้แสดงว่าความถี่ที่ได้รับหรือที่หายไปมีคาบเวลาตามที่กำหนด เพื่อแสดงความถูกต้องของสัญญาณ

TOE - Tone Output Enable (input) ใช้ควบคุม Q1-Q4 ให้เป็นอิมพีแดนซ์สูง

tID - เวลาที่สั้นที่สุดระหว่างสัญญาณ DTMF ที่ถูกต้อง 2 สัญญาณ

tDO - เวลาที่นานที่สุดที่ยอมให้สัญญาณหายไป ในคาบเวลาความถี่ที่ถูกต้อง

tDP - เวลาที่ใช้ในการตรวจพบความถี่ DTMF ที่ถูกต้อง

tDA - เวลาที่ใช้ในการตรวจหายไปของสัญญาณความถี่ DTMF ที่ถูกต้อง

tGTP - ช่วงคาบเวลาของความถี่ของการปรากฏความถี่ DTMF

tGTA - ช่วงคาบเวลาของความถี่ของการหายไปของความถี่ DTMF

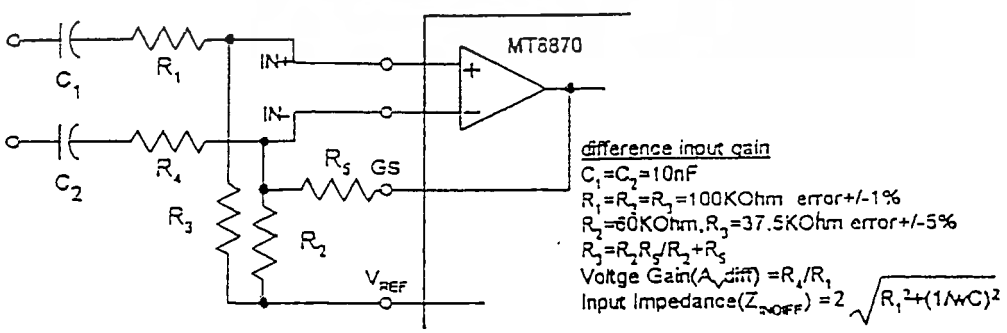
สำหรับคำว่า Guard time หมายถึง ช่วงคาบเวลาความถี่ที่เข้ามาซึ่งจะต้องนานเท่ากับหรือกว่าช่วงเวลาที่เราตั้งไว้ จึงจะได้รับการยอมรับว่าสัญญาณความถี่นั้นถูกต้อง หรือพูดได้ว่าเวลาที่เราตั้งไว้โดย RC ก็คือ Grand time นั่นเอง เมื่อสัญญาณความถี่เข้ามานานเท่ากับหรือมากกว่าเวลาที่ตั้งไว้จึงสามารถแปลงเป็นตัวเลขได้ ถ้าสัญญาณความถี่ที่เข้ามาสั้นกว่าก็จะไม่มีการถอดรหัสเป็นตัวเลขออกไป การตั้งเวลาและคำนวณดูได้จากรูปที่ 2.6

4. ภาคขยายสัญญาณความแตกต่าง

วงจรส่วนอินพุตของ MT 8870 เป็นภาคขยายสัญญาณแบบออปแอมป์ ที่สามารถปรับอัตราการขยายโดยต่อวงจรภายนอกเพิ่มเข้าไป และแสดงการต่อวงจรภายนอกเข้ากับอินพุต ซึ่งสามารถคำนวณอัตราการขยายความแตกต่างของอินพุต และอิมพีแดนซ์ได้ ดังนี้

อัตราขยาย (Av diff = R5/R1)

อินพุตอิมพีแดนซ์ (Zin diff = 2/ RI EXP 2 +(1/We)EXP 2



รูปที่ 2.8 การต่อวงจรภาคอินพุต

5. ภาคกำเนิดความถี่

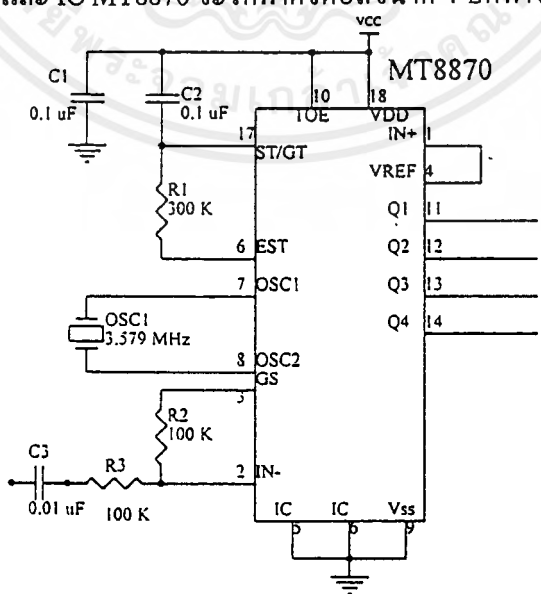
ในภาคนี้จะมีวงจรเวลาอยู่ภายในเพียงแต่ต่อแร่ X-TAL ที่มีความถี่ 3.579545 MHz. ก็จะสามารถใช้งานได้เลย

2.6.3 การนำ MT 8870 ไปใช้งาน

1. นำไปใช้งานด้านการควบคุมระยะไกล
2. เครื่องป้องกันโทรศัพท์ทางไกล
3. ใช้ในงานด้านเครดิตการ์ด
4. ใช้งานร่วมกับคอมพิวเตอร์
5. ใช้ในเครื่องชุมสายขนาดย่อยๆ หรือ PABX
6. ใช้กับงานทางด้านโทรศัพท์ทั่วไป
7. เครื่องกันขโมย
8. การควบคุมอุปกรณ์ทางโทรศัพท์
9. ใช้ทำเครื่องสอบถามทางโทรศัพท์

2.7 การถอดรหัสสัญญาณ DTMF

หลังจากที่เรารับสัญญาณ DTMF แล้ว จะต้องมีวงจรถอดรหัสสัญญาณ DTMF เป็นสัญญาณดิจิตอลขนาด 4 บิต โดยใช้ IC MT8870 เป็นตัวถอดรหัสและแปลงสัญญาณ ซึ่งมีลักษณะการต่อวงจรดังรูปที่ 2.9 และ IC MT8870 จะให้ค่าดิจิตอลขนาด 4 บิตทางเอาท์พุทดังตารางที่ 2.3



รูปที่ 2.9 วงจรถอดรหัส สัญญาณ DTMF

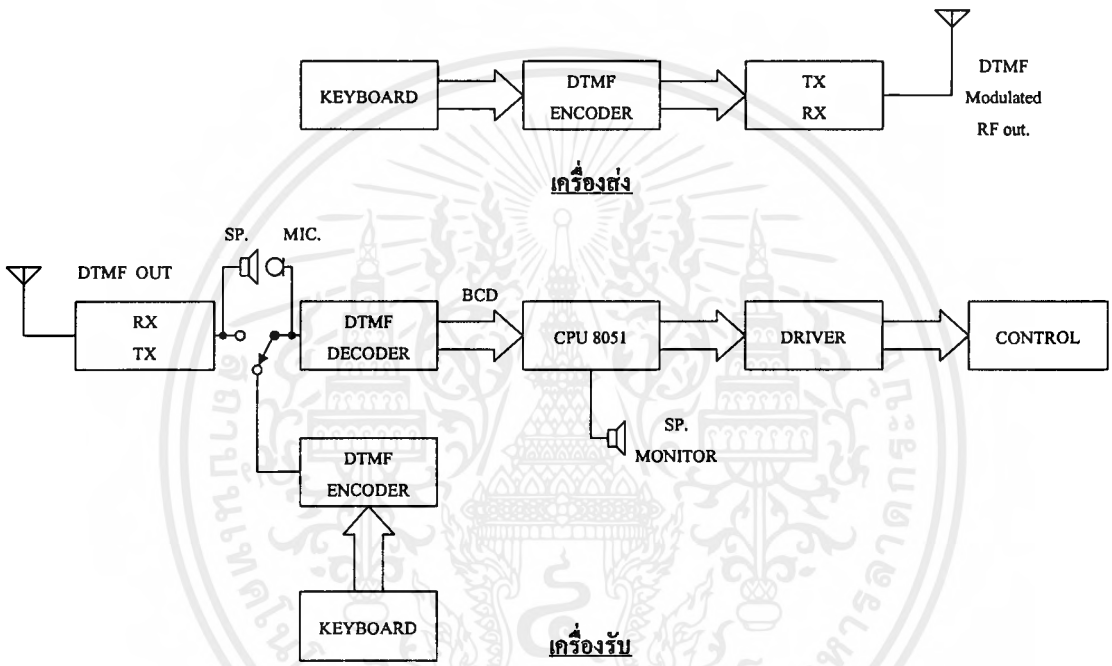
DIGITAL	TOE	INX	Est	Q4	Q3	Q2	Q1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	X	H	1	1	0	1
B	H	X	H	1	1	1	0
C	H	X	H	1	1	1	1
D	H	X	H	0	0	0	0

ตารางที่ 2.3 แสดงการถอดรหัสสัญญาณ DTMF เป็นเลขฐานสอง

บทที่ 3
โครงสร้างของระบบโครงงาน

3.1 โครงสร้างทางฮาร์ดแวร์

3.1.1 บล็อคไดอะแกรมการทำงานของโครงงาน



รูปที่ 3.1 บล็อคไดอะแกรมการทำงานของโครงงาน

เมื่อทำการกด KEYBOARD จะผลิตสัญญาณ DTMF ออกมาเป็นความถี่คู่ ป้อนเข้าสู่ภาค DTMF ENCODER ทำการเข้ารหัสสัญญาณ DTMF ได้สัญญาณความถี่ที่แตกต่างกัน จากนั้นภาค DTMF DECODER จะทำการถอดรหัสสัญญาณ DTMF ให้ได้เป็นสัญญาณเลขไบนารีขนาด 4 บิต ซึ่งเลขไบนารีขนาด 4 บิตจากภาค DTMF DECODER นี้จะถูกนำไปเปรียบเทียบกับเลขไบนารีขนาด 4 บิตในตัวไมโครคอนโทรลเลอร์ 8051 แล้วจึงนำค่าที่ได้ผ่านภาค DRIVER ทำการขยายสัญญาณไปควบคุมอุปกรณ์ไฟฟ้าที่ภาค CONTROL ต่อไป

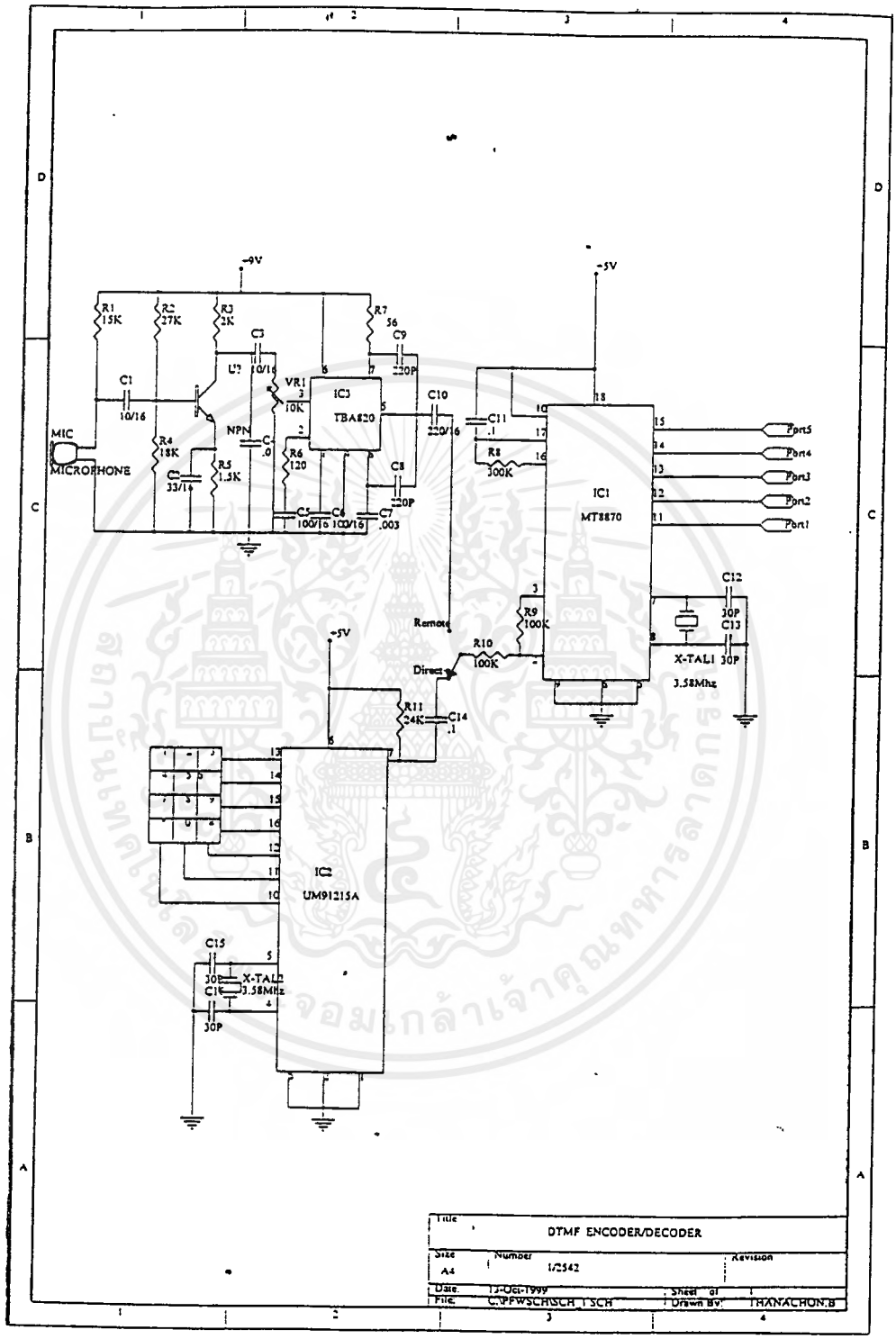
ซึ่งจุดเด่นของโครงงานนี้ ที่ภาค DTMF DECODER นอกจากจะรับสัญญาณ DTMF จากภาค DTMF ENCODER ได้โดยตรงแล้ว ยังสามารถที่จะรับสัญญาณ DTMF จากเครื่องวิทยุรับส่ง

ได้ทุกชนิดอีกด้วย กล่าวคือที่ภาค DTMF DECODER จะใช้ไมโครโฟนในการรับสัญญาณ DTMF ของวิทยุรับส่งที่ผ่านลำโพงออกมาเป็นความถี่ของสัญญาณเสียงได้อีกด้วย

3.1.2 การทำงานของวงจรส่วนเข้ารหัสและถอดรหัสสัญญาณ DTMF

จากรูปที่ 3.2 เมื่อทำการกดคีย์บอร์ดจะทำให้ไอซี 2 เบอร์UM91215A ผลิตสัญญาณ DTMF ซึ่งมีค่าความถี่เปลี่ยนแปลงตามตัวเลขที่กด ผ่านสวิตช์ที่ทำการเลือกการต่อโดยตรงหรือรีโมท เข้าสู่ไอซี1 เบอร์ MT8870 ทำหน้าที่ถอดรหัสสัญญาณ DTMF ให้ได้เป็นสัญญาณเลขไบนารีขนาด 4 บิต ซึ่งเลขไบนารีขนาด 4 บิตจากไอซี 1 นี้จะถูกนำไปเปรียบเทียบกับเลขไบนารีขนาด 4 บิตในตัวของไมโครคอนโทรลเลอร์ 8051 แล้วจึงนำค่าที่ได้ไปควบคุมอุปกรณ์ไฟฟ้าที่ภาค DRIVER ต่อไป

ส่วนทางด้านสวิตช์ที่เลือกรีโมท จะใช้ไมโครโฟนในการรับสัญญาณ DTMF จากอุปกรณ์สื่อสารทุกชนิดที่สามารถส่งสัญญาน DTMF ได้ ผ่านวงจรแอมปลิไฟล์ทำการขยายสัญญาณ DTMF ให้แรงพอที่จะส่งให้ไอซี 1 สามารถทำการถอดรหัสสัญญาณ DTMF ให้เป็นเลขไบนารี 4 บิตทางเอาท์พุตได้



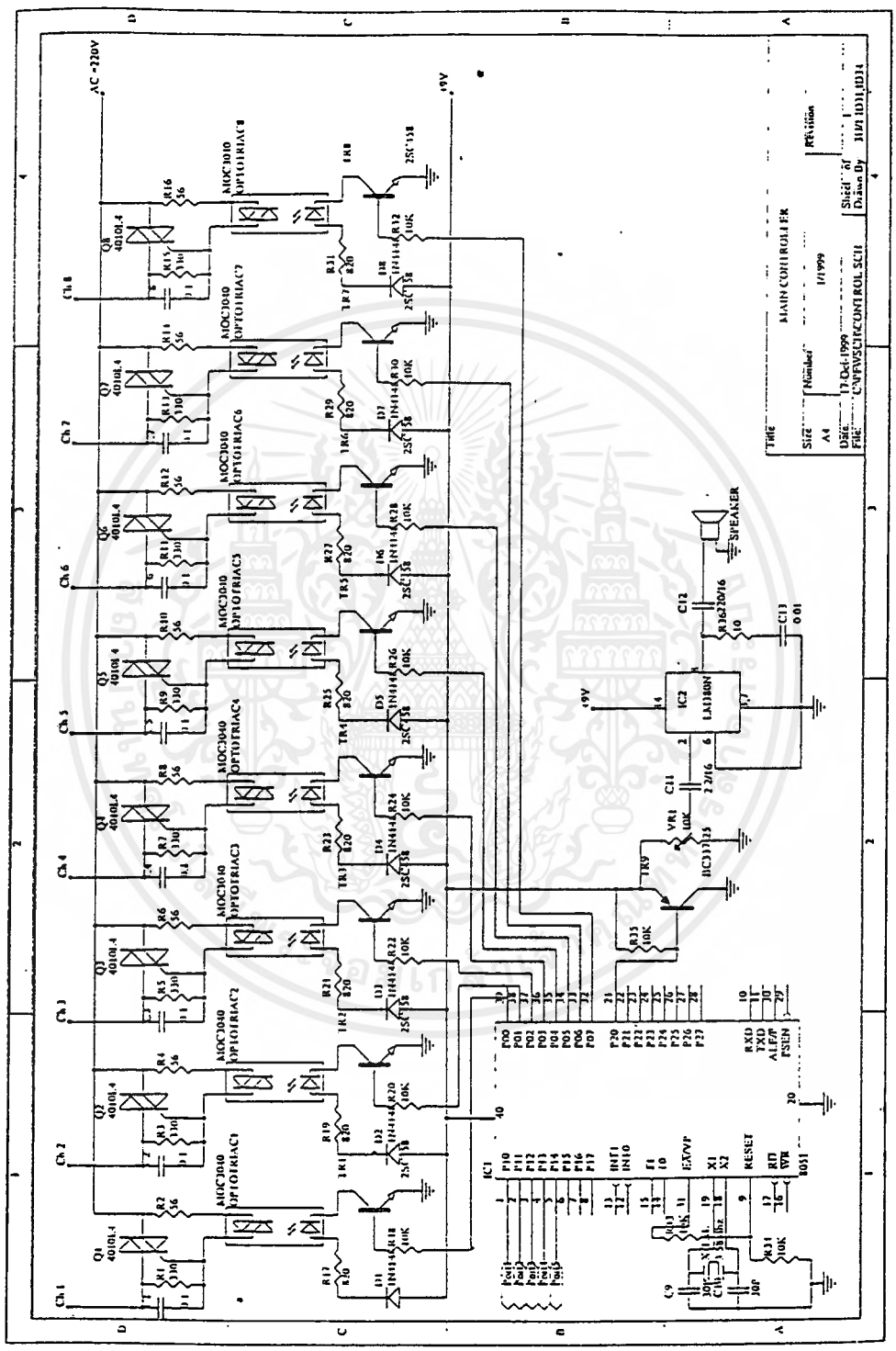
รูปที่ 3.2 วงจรการทำงานส่วนการเข้ารหัสและถอดรหัสสัญญาณ DTMF

3.1.3 การทำงานของวงจรส่วนการควบคุม

จากรูปที่ 3.3 ประกอบด้วยส่วนที่ทำหน้าที่ควบคุมกับส่วนที่ทำหน้าที่ไดรเวอร์ โดยส่วนควบคุมจะรับสัญญาณไบนารี 4 บิตเข้ามาทางพอร์ต 1 ของไอซี 1 ที่เป็นไมโครคอนโทรลเลอร์ 8051 ซึ่งมีหน่วยความจำภายในขนาด 4 Kbit เพื่อใช้เป็นหน่วยความจำในการเขียนโปรแกรมควบคุมการทำงานของส่วนไดรเวอร์ โดยสั่งการทำงานออกมาทางพอร์ต 0 ของไมโครคอนโทรลเลอร์ 8051

ที่ขา 21 ของไมโครคอนโทรลเลอร์ 8051 นั้น เป็นการเขียนโปรแกรมกำเนิดสัญญาณความถี่เสียงผ่านแอมปลิไฟล์ทำการขยายสัญญาณออกลำโพง เพื่อบอกสภาวะการทำงานของวงจร

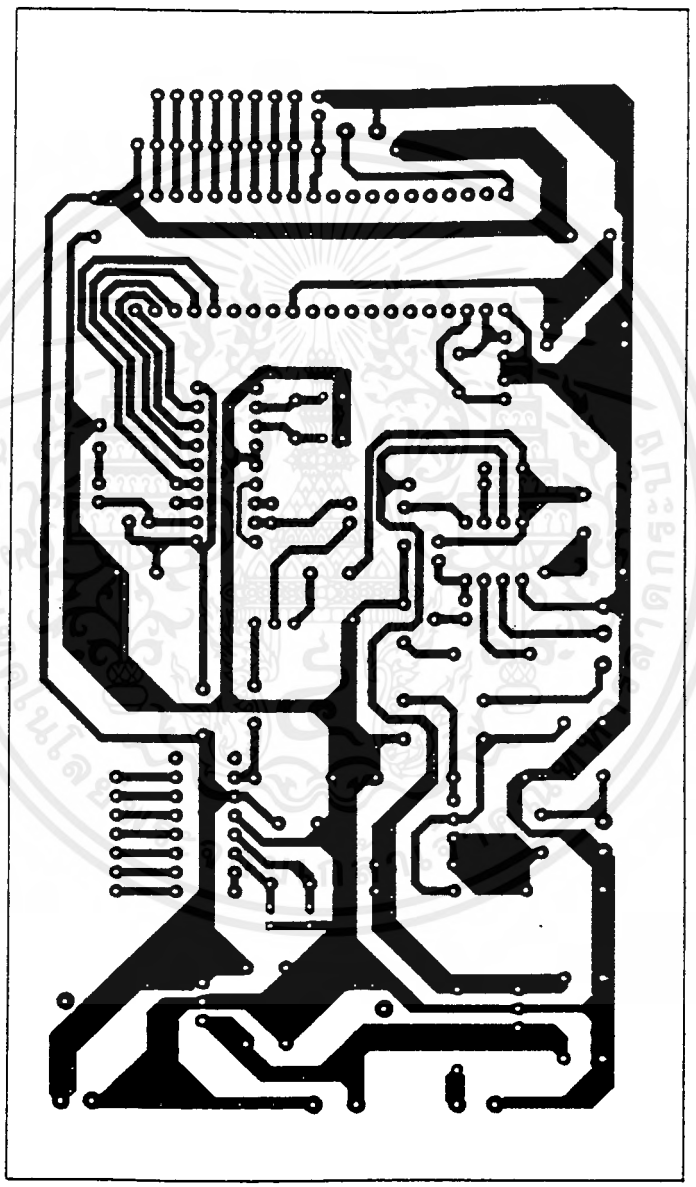
ส่วนที่ทำหน้าที่ไดรเวอร์จะรับสัญญาณการควบคุมจากไมโครคอนโทรลเลอร์ 8051 ผ่านพอร์ต 0 ในแต่ละขา โดยสามารถทำการควบคุมได้ในแต่ละช่องสัญญาณ โดยที่พอร์ต 0 ของไมโครคอนโทรลเลอร์ 8051 จะส่งสัญญาณ HIGH ออกมาทำการไดรเวอร์ทรานซิสเตอร์ 2SC458 ที่ทำหน้าที่เป็นสวิทช์ ON-OFF เพื่อทำให้ OPTO TRIAC เบอร์ MOC3041 ทำงานหรือหยุดทำงาน เมื่อ TRIAC ทำงานจะยอมให้ไฟ AC ไหลผ่านไปสู่โหลดทางเอาท์พุตได้



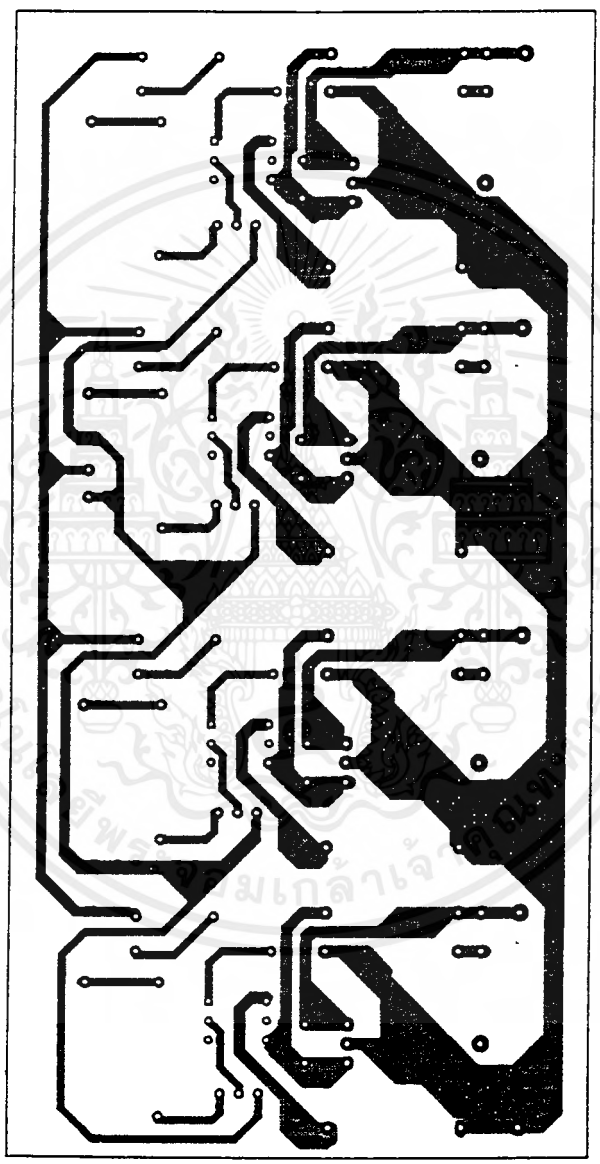
รูปที่ 3.3 วงจรการทำงานส่วนควบคุม

3.1.4 การสร้างแผ่นวงจรพิมพ์

หลังจากที่กำหนดขอบเขตของวงจรและได้ทำการทดลองบนแผ่นโฟโต้บอร์ดแล้วจึงดำเนินการออกแบบลายวงจร โดยจัดวางอุปกรณ์ต่าง ๆ ให้ง่ายต่อการเชื่อมต่อ และวางตำแหน่งของคอนเน็คเตอร์เพื่อความสะดวกและความสวยงาม โดยได้แยกแผ่นลายวงจรออกเป็นสองแผ่นทำให้เกิดความสะดวกในการวางอุปกรณ์และการต่อคอนเน็คเตอร์ การเชื่อมต่อไฟเลี้ยงให้แผ่นวงจรทั้งสองผ่านทางคอนเน็คเตอร์ด้วย ซึ่งในการดำเนินการจะมีการปรับเปลี่ยนหลายครั้งเพื่อให้เกิดความเหมาะสมและเกิดความสะดวกมากขึ้นจนแน่ใจว่าได้ตำแหน่งที่เหมาะสมแล้ว จึงดำเนินการกัดลายวงจรให้เป็นแผ่นวงจรพิมพ์ที่สมบูรณ์ต่อไป



รูปที่ 3.4 ลายวงจรของแผ่นควบคุมหลัก



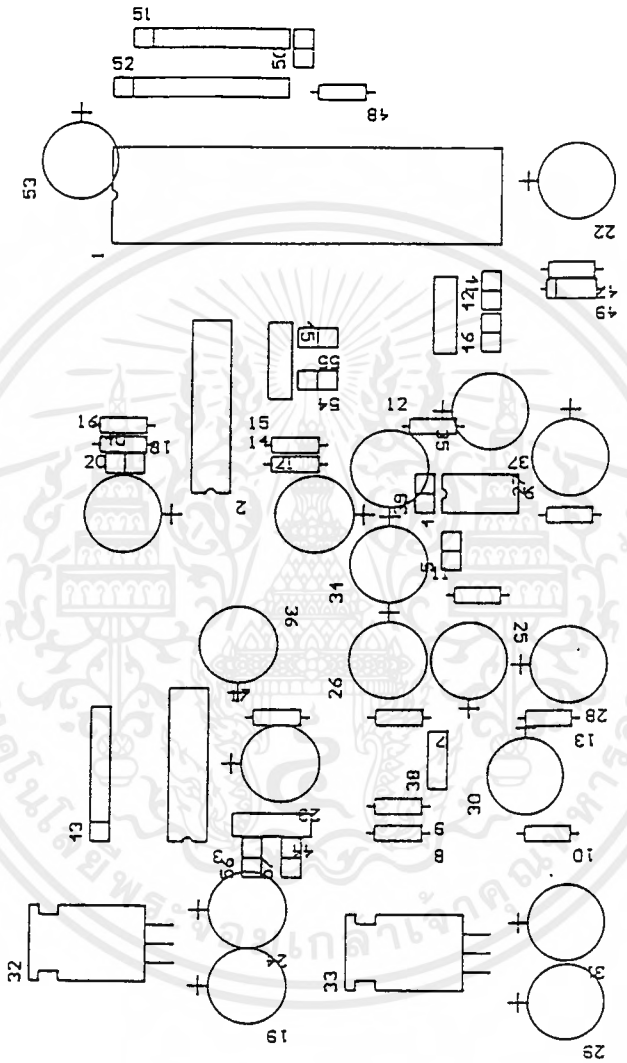
รูปที่ 3.5 ลายวงจรของ TRIAC ควบคุมอุปกรณ์

3.1.5 การลงอุปกรณ์

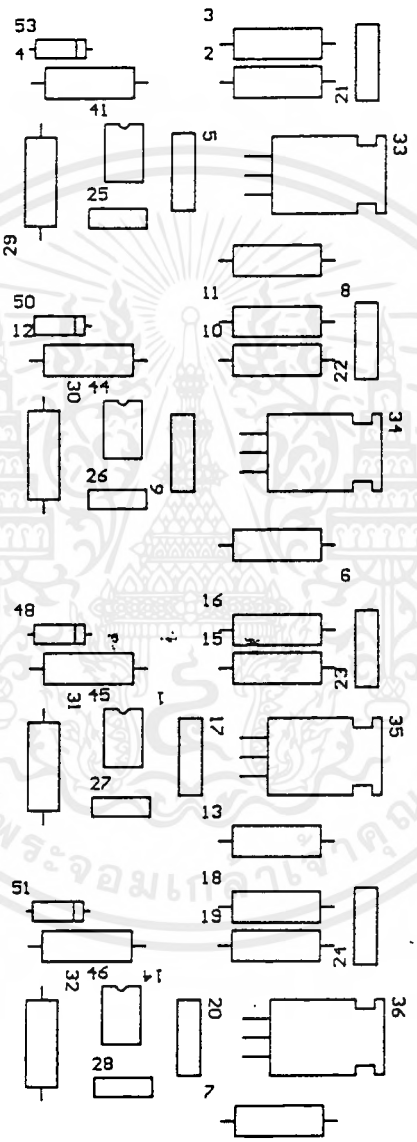
หลังจากที่สร้างลายวงจรเสร็จแล้วจะเป็นขั้นตอนการลงอุปกรณ์ โดยมีขั้นตอนการดำเนินการ ดังนี้

1. ลงอุปกรณ์ที่มีขนาดเตี้ย ๆ ก่อน เช่น จุดเชื่อมต่อ ตัวความต้านทาน เป็นต้น
2. ควรตรวจสอบค่าของอุปกรณ์และเบอร์ของอุปกรณ์ให้ถูกต้อง
3. ตำแหน่งอุปกรณ์จะต้องถูกต้องไม่ผิดพลาด
4. วางตำแหน่งของอุปกรณ์ให้ถูกต้อง เพราะถ้าไม่ถูกต้องอาจเกิดความเสียหายได้
5. อุปกรณ์ประเภทไอซีควรใช้ SOCKET
6. ระมัดระวังเรื่องไฟฟ้าสถิต ซึ่งอาจจะทำความเสียหายให้แก่อุปกรณ์ เช่น ทรานซิสเตอร์ และไอซีได้
7. การบัดกรีในแต่ละจุดต้องมั่นใจว่าขาของอุปกรณ์ติดแน่นและไม่ลัดวงจรกับขาอื่น ๆ
8. ตำแหน่งของคอนเน็คเตอร์ในแต่ละจุดจะต้องถูกต้องตามตำแหน่ง

หลังจากที่ลงอุปกรณ์ต่าง ๆ แล้วต้องตรวจสอบความถูกต้องและความเรียบร้อยอีกครั้ง โดยเฉพาะในส่วนของแรงดันไฟสลับ 220 โวลท์ จนแน่ใจว่าจะไม่เกิดการลัดวงจรขึ้น เพราะถ้าเกิดการลัดวงจรจะทำให้อุปกรณ์เกิดความเสียหายอย่างรุนแรงได้

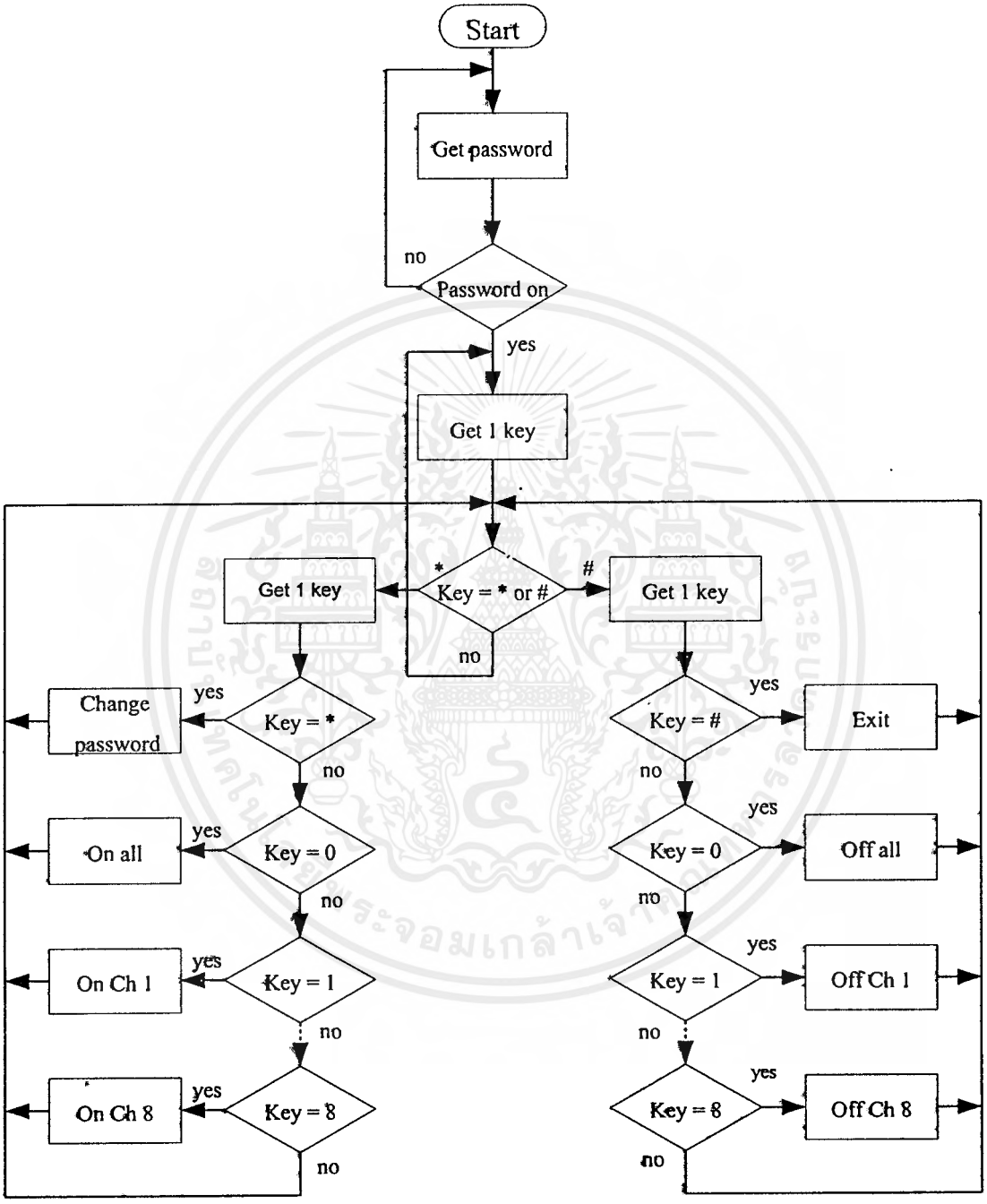


รูปที่ 3.6 การลงอุปกรณ์ของวงจรควบคุมหลัก



รูปที่ 3.7 การลงอุปกรณ์ของวงจร TRIAC

3.2.1 ขั้นตอนการทำงานของโปรแกรมควบคุม



รูปที่ 3.8 โฟลว์ชาร์ทแสดงการทำงานของโปรแกรม

3.2.2 คำสั่งที่ใช้ในการควบคุม

รอรับรหัสผ่าน 4 ตัว เพื่อผ่านเข้าไปควบคุมอุปกรณ์ (มีสัญญาณบี๊บตอบรับ 2 ครั้ง)

** ตั้งรหัสผ่านใหม่ (มีสัญญาณบี๊บตอบรับ 2 ครั้ง)

#1 ควบคุมให้อุปกรณ์ที่ ช่องที่ 1 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*1 ควบคุมให้อุปกรณ์ที่ ช่องที่ 1 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#2 ควบคุมให้อุปกรณ์ที่ ช่องที่ 2 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*2 ควบคุมให้อุปกรณ์ที่ ช่องที่ 2 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#3 ควบคุมให้อุปกรณ์ที่ ช่องที่ 3 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*3 ควบคุมให้อุปกรณ์ที่ ช่องที่ 3 ทำงาน(มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#4 ควบคุมให้อุปกรณ์ที่ ช่องที่ 4 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*4 ควบคุมให้อุปกรณ์ที่ ช่องที่ 4 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#5 ควบคุมให้อุปกรณ์ที่ ช่องที่ 5 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*5 ควบคุมให้อุปกรณ์ที่ ช่องที่ 5 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#6 ควบคุมให้อุปกรณ์ที่ ช่องที่ 6 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*6 ควบคุมให้อุปกรณ์ที่ ช่องที่ 6 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#7 ควบคุมให้อุปกรณ์ที่ ช่องที่ 7 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*7 ควบคุมให้อุปกรณ์ที่ ช่องที่ 7 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#8 ควบคุมให้อุปกรณ์ที่ ช่องที่ 8 หยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*8 ควบคุมให้อุปกรณ์ที่ ช่องที่ 8 ทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

#0 ควบคุมให้อุปกรณ์ทุกช่องหยุดทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้ง)

*0 ควบคุมให้อุปกรณ์ทุกช่องทำงาน (มีสัญญาณบี๊บตอบรับ 1 ครั้ง)

ออกจากโปรแกรมการทำงาน (มีสัญญาณบี๊บยาวตอบรับ 1 ครั้งและสั้น 1 ครั้ง)

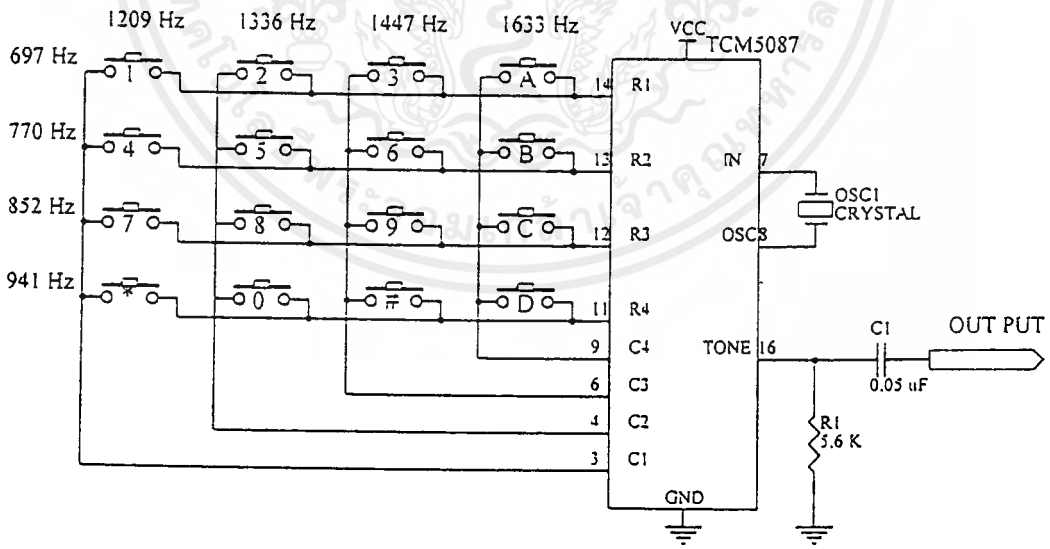
บทที่ 4
ผลการดำเนินงาน

4.1 เครื่องมือและอุปกรณ์ที่ใช้

- แผงวงจร ชุดฝึกไมโครคอนโทรลเลอร์ 8051
- ชุดแผงต่อวงจรทดลอง (Protoboard)
- แหล่งจ่ายไฟ 5 VDC. และ 12 VDC.
- แป้นกดกำเนิดความถี่ DTMF (Key Pad)

ในการดำเนินการ จะแยกการทำงานของวงจรออกเป็นส่วน ๆ เพื่อให้ได้ข้อมูลที่ชัดเจนและปัญหาที่เกิดขึ้นในแต่ละส่วน เพื่อให้สามารถวิเคราะห์ปัญหาต่าง ๆ ได้อย่างถูกต้อง ซึ่งในการทดลองจะกระทำโดยการจำลองอินพุตเพื่อดูเอาท์พุตที่เกิดขึ้น

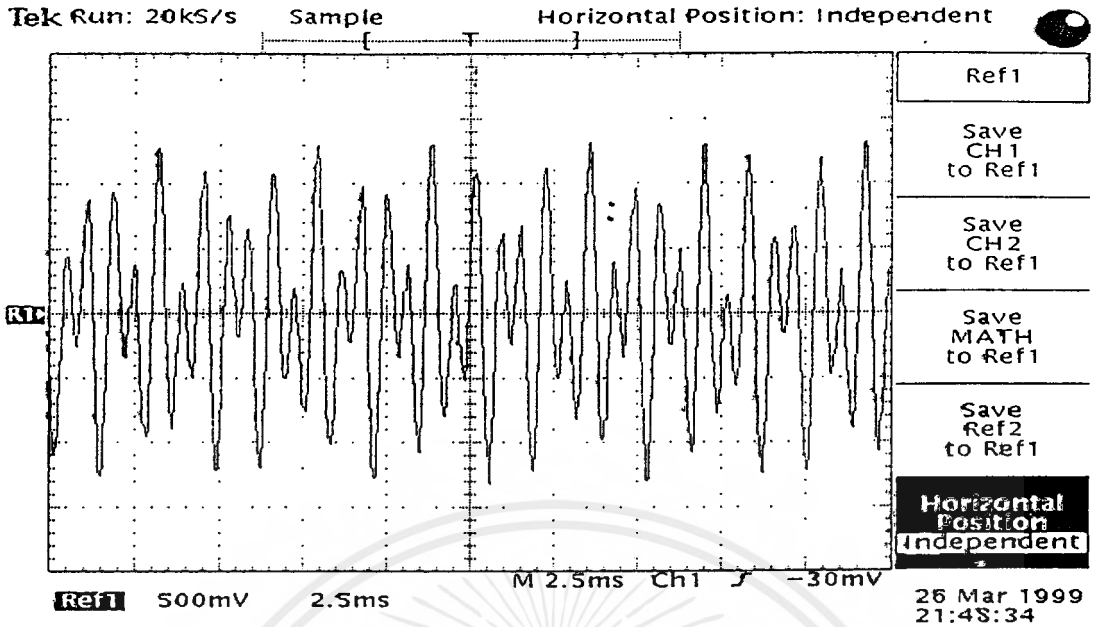
4.2 ส่วนการเข้ารหัสสัญญาณ DTMF



รูปที่ 4.1 วงจรกำเนิดสัญญาณ DTMF

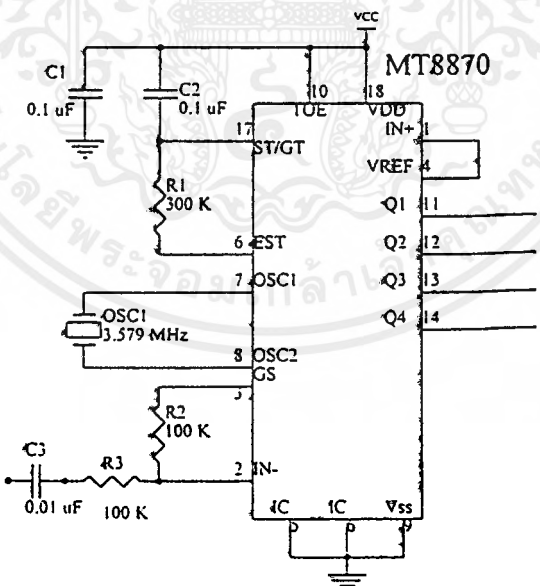
หมายเลขแป้นที่กด	ความถี่ด้านต่ำ	ความถี่ด้านสูง	เอาท์พุตไบนารี่
1	697 Hz	1209 Hz	0001
2	697 Hz	1336 Hz	0010
3	697 Hz	1447 Hz	0011
4	770 Hz	1209 Hz	0100
5	770 Hz	1336 Hz	0101
6	770 Hz	1447 Hz	0110
7	852 Hz	1209 Hz	0111
8	852 Hz	1336 Hz	1000
9	852 Hz	1447 Hz	1001
0	914 Hz	1336 Hz	1010
*	914 Hz	1209 Hz	1011
#	914 Hz	1447 Hz	1100
A	697 Hz	1633 Hz	1101
B	770 Hz	1633 Hz	1110
C	852 Hz	1633 Hz	1111
D	941 Hz	1633 Hz	0000

ตารางที่ 4.1 แสดงการเปลี่ยนค่าความถี่ DTMF ไปเป็นเลขฐานสอง



รูปที่ 4.2 ลักษณะพัลส์ของการเข้ารหัสสัญญาณ DTMF

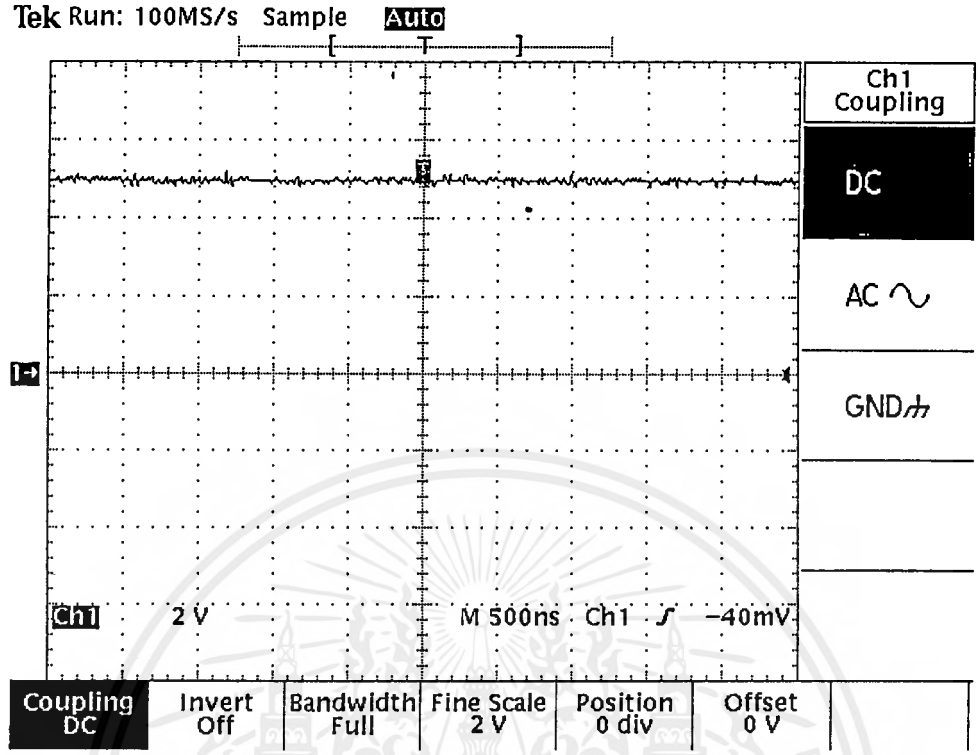
4.3 ส่วนการถอดรหัสสัญญาณ DTMF



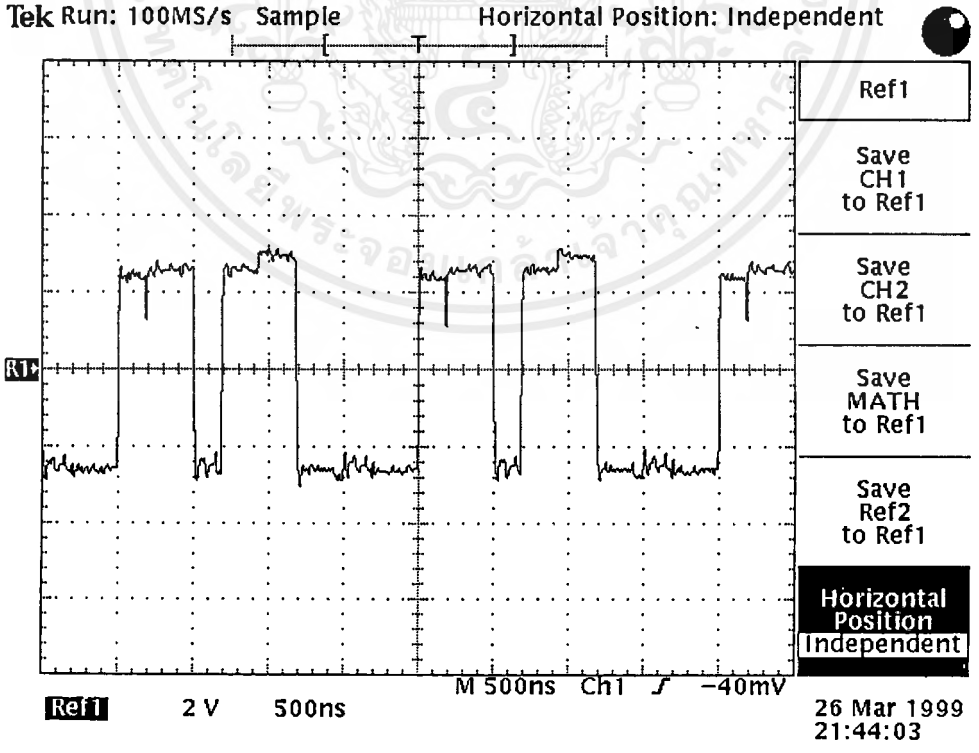
รูปที่ 4.3 วงจรถอดรหัส สัญญาณ DTMF

DIGITAL	TON	INX	Est	Q4	Q3	Q2	Q1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	X	H	1	1	0	1
B	H	X	H	1	1	1	0
C	H	X	H	1	1	1	1
D	H	X	H	0	0	0	0

ตารางที่ 4.2 แสดงการถอดรหัสสัญญาณ DTMF เป็นเลขฐานสอง



รูปที่ 4.4 สัญญาณ DTMF ที่ส่งออกมาที่เอาท์พุท Q ขณะมีสถาวะเป็น "1"

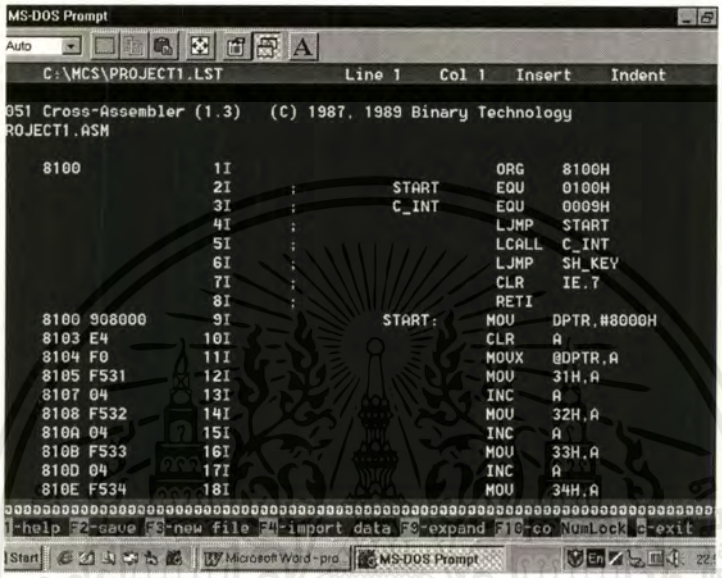


รูปที่ 4.5 สัญญาณที่ส่งไปควบคุม TRIAC ในส่วนของการควบคุมอุปกรณ์

4.4 โปรแกรมการทำงาน

4.4.1 เขียนโปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคล (PC)

การเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งใช้อิดิเตอร์ของไซคิด (SK) เพราะจะทำให้เกิดความสะดวกในการแก้ไข



รูปที่ 4.6 การเขียนโปรแกรมบน SK

4.4.2 การคอมไพล์โปรแกรม

ใช้โปรแกรมคอมไพล์ (SXA51) เพื่อทำการคอมไพล์ไฟล์ .ASM (Project1.asm) ที่เขียนด้วยโปรแกรม SK ให้เป็นนามสกุล .HEX (โดยใช้คำสั่ง sxa51 project1.asm) เพื่อให้ได้ไฟล์ในรูปแบบของเลขฐาน 16 หรือจะคอมไพล์เป็นไฟล์ .LST เพื่อใช้ในการตรวจสอบโปรแกรมโดยละเอียด ด้วยก็ได้ (โดยใช้คำสั่ง sxa51 –l project1.asm)

4.4.3 การทดสอบการทำงานของโปรแกรม

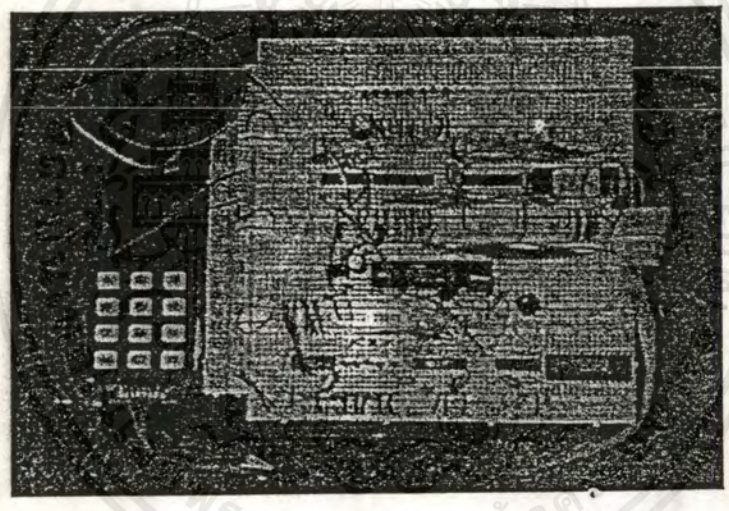
การทดสอบการทำงานของโปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคล สามารถที่จะทำการตรวจสอบการทำงานของโปรแกรมที่เขียนขึ้น โดยใช้โปรแกรม 8051sim ซึ่งจะเป็นการจำลองการทำงานเพื่อตรวจสอบสภาวะต่าง ๆ ของรีจีสเตอร์ และขั้นตอนการทำงานของโปรแกรมในแต่ละขั้นตอนการทำงาน จะเป็นประโยชน์อย่างยิ่งในการตรวจสอบโดยละเอียด

4.4.4 การทดสอบการทำงานของโปรแกรมบนชุดฝึกไมโครคอนโทรลเลอร์

ทำการทดสอบการทำงานของโปรแกรมบนชุดฝึกไมโครคอนโทรลเลอร์ของ 8051 โดยการโหลดไฟล์ในรูปของเลขฐาน 16 ลงบนชุดฝึกไมโครคอนโทรลเลอร์ของ 8051 ผ่านทางพอร์ตอนุกรม 1 (COM PORT 1) เพื่อทำการทดสอบการทำงานของโปรแกรมบนเครื่องชุดฝึกไมโครคอนโทรลเลอร์ของ 8051 ได้เช่นกัน และจะเป็นการทำงานที่ใกล้เคียงกับลักษณะของความจริงมาก

4.4.5 การทดสอบบนแผงโปรโตบอร์ด

เนื่องจากที่แผงโปรโตบอร์ดจะสามารถต่อวงจรในส่วนต่าง ๆ ซึ่งการทดลองในขั้นตอนนี้จะเป็นการจำลองที่เหมือนจริงมากจนแน่ใจได้ว่า ทั้งวงจรและการทำงานของโปรแกรมถูกต้องดีทุกหน้าที่การทำงาน



รูปที่ 4.7 การต่อวงจรเพื่อทดลองการทำงานบนแผงโปรโตบอร์ด

4.4.6 การทดสอบการทำงานบนแผ่นวงจรจริง

เพื่อเป็นการทดสอบลายวงจรและโปรแกรมร่วมกัน โดยการตรวจสอบในแต่ละจุดจะทำการเปรียบเทียบกับผลที่ได้ในขณะที่ทดลองในแต่ละภาค ซึ่งจะเป็นการทดสอบในขั้นตอนเกือบจะสุดท้าย

4.4.7 นำแผ่นวงจรบรรจุลงกล่อง

การนำแผ่นวงจรบรรจุลงกล่องรวมทั้งการเดินสายต่าง ๆ ให้เกิดความสวยงามและสะดวกในการใช้งาน โดยทำการทดสอบการใช้งานจริงอีกครั้งจนแน่ใจว่ามีความถูกต้องครบทุกหน้าที่การทำงาน ก่อนที่จะนำมาใช้งานจริง ๆ ได้ตามที่ต้องการ

บทที่ 5
สรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากแนวความคิดที่ต้องการควบคุมอุปกรณ์ต่าง ๆ จากการนำสัญญาณ DTMF มาใช้ประโยชน์จึงได้สร้างปริญญานิพนธ์นี้ขึ้นมา ซึ่งปรากฏว่าสามารถนำมาใช้ประโยชน์ได้จริงในชีวิตประจำวัน สามารถส่งสัญญาณควบคุมได้ในระยะที่สายตามองเห็นได้ โดยสามารถควบคุมให้อุปกรณ์ทำงาน (ปิดวงจร) หรือหยุดการทำงาน (เปิดวงจร) ได้ทั้ง 8 ช่องตามที่ต้องการ นอกจากนั้นยังมีสัญญาณตอบกลับเพื่อเป็นการแสดงสถานะของการควบคุมให้ทราบขณะที่ทำการสั่งงาน ดังนั้นสัญญาณ DTMF จึงถือว่าเป็นสัญญาณที่เป็นประโยชน์อย่างยิ่งที่เรานิยมนำมาใช้งานกันอย่างแพร่หลาย

5.2 ประโยชน์ที่ได้รับจากปริญญานิพนธ์

เนื่องจากปริญญานิพนธ์นี้ประกอบไปด้วยสองส่วนหลักคือ การทำงานของวงจรและส่วนโปรแกรมควบคุมการทำงาน จึงทำให้ได้เรียนรู้ทั้งสองด้าน คือ

5.2.1 ในด้านของวงจรการทำงาน

ได้ศึกษาถึงการสร้างแผ่นวงจรพิมพ์ การลงอุปกรณ์อิเล็กทรอนิกส์ รวมทั้งได้ศึกษาถึงคุณสมบัติประจำตัวของไมโครคอนโทรลเลอร์ในตระกูล 8051, การเข้ารหัสสัญญาณ DTMF โดยใช้ไอซีเบอร์ UM91215A และการถอดรหัสสัญญาณ DTMF โดยใช้ไอซีเบอร์ MT8870 รวมทั้งการนำอุปกรณ์เหล่านี้มาใช้งานจริงในการเชื่อมต่อวงจร

5.2.2 ในด้านของโปรแกรมการทำงาน

ได้ศึกษาถึงการใช้คำสั่งในการเขียนโปรแกรมควบคุมการทำงาน การเขียนโปรแกรมบนอิดิเตอร์ (SK) การแปลงไฟล์ให้เป็นไฟล์ในรูปแบบต่าง ๆ เช่น ไฟล์ในรูปแบบของเลขฐาน 16 (File.HEX) ไฟล์ในรูปแบบของการแสดงรายละเอียด (File.LST) การทดสอบการทำงานของโปรแกรมบนโปรแกรมจำลอง (8051sim) ทำให้ได้ศึกษาถึงการทำงานโดยละเอียด รวมทั้งการทำการทดสอบบนเครื่อง SINGLE BOARD ของ 8051 ตลอดจนการนำโปรแกรมมาใช้งานจริงมาติดตั้งบนแผงวงจร ซึ่งในแต่ละขั้นตอนก็ได้มีการปรับเปลี่ยนและแก้ไขสิ่งต่าง ๆ อยู่หลายต่อหลายครั้งเพื่อให้ได้การทำงานที่ถูกต้อง จนสามารถนำมาใช้งานจริงได้เป็นผลสำเร็จ

5.3 การดำเนินงานบรรลุตามวัตถุประสงค์ของปริญญานิพนธ์หรือไม่

จากผลที่ได้จากการทดลองและการนำมาใช้งานจริง พบว่าสามารถทำงานได้ตามเป้าหมายที่วางไว้ ทั้งในด้านของการทำงานของตัวโปรแกรมควบคุมการทำงานและการส่งสัญญาณ DTMF มากับคลื่นวิทยุ โดยสามารถควบคุมได้ในระยะที่ตั้งเป้าหมายไว้คือ ในระยะที่สายตาคนเรามองเห็น จึงสรุปได้ว่าการดำเนินงานทั้งหมดบรรลุตามวัตถุประสงค์ของปริญญานิพนธ์ฉบับนี้

5.4 ปัญหาที่พบและการแก้ไข

ในการดำเนินงานปรากฏว่าจะมีปัญหาต่าง ๆ เกิดขึ้นอยู่ตลอดเวลาทั้งปัญหาที่ได้คาดการณ์ไว้ว่าจะเกิดขึ้นและปัญหาที่เกิดขึ้นโดยไม่ได้คาดคิด เช่น

5.4.1 ปัญหาในด้านของอุปกรณ์

ปัญหาในด้านของอุปกรณ์คือ คุณภาพของอุปกรณ์ทุก ๆ ตัวที่จะนำมาต่อใช้งานทั้งในแผงโปรโตบอร์ดและแผ่นวงจรจริง ซึ่งควรทำการเช็คอุปกรณ์ทุก ๆ ตัวที่จะนำมาใช้งาน และการบันทึกโปรแกรมด้วยชุดฝึกของไมโครคอนโทรลเลอร์8051 จะให้ความถูกต้องและมีความสะดวกมากขึ้น

5.4.2 ปัญหาในเรื่องของความถี่

เนื่องจากวงจรจะต้องมีความถี่ไว้เพื่อเป็นสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์ และสร้างแหล่งกำเนิดความถี่ให้กับวงจรส่วนต่าง ๆ ซึ่งส่วนใหญ่แล้วเป็นความถี่สูงที่ได้จากคริสตอล จึงอาจทำให้เกิดการสูญเสียได้ง่ายในกรณีที่ต่อวงจรทดลองบนแผงโปรโตบอร์ด เช่น การเสียบที่ไม่แน่นเพียงพอ การหลวมที่เกิดจากการเคลื่อนย้ายในการเดินทางต่าง ๆ ซึ่งพบว่าเป็นปัญหาที่เกิดขึ้นบ่อย ดังนั้นในการทำงานจึงต้องตรวจสอบทุกครั้ง

5.5 ข้อเสนอแนะ

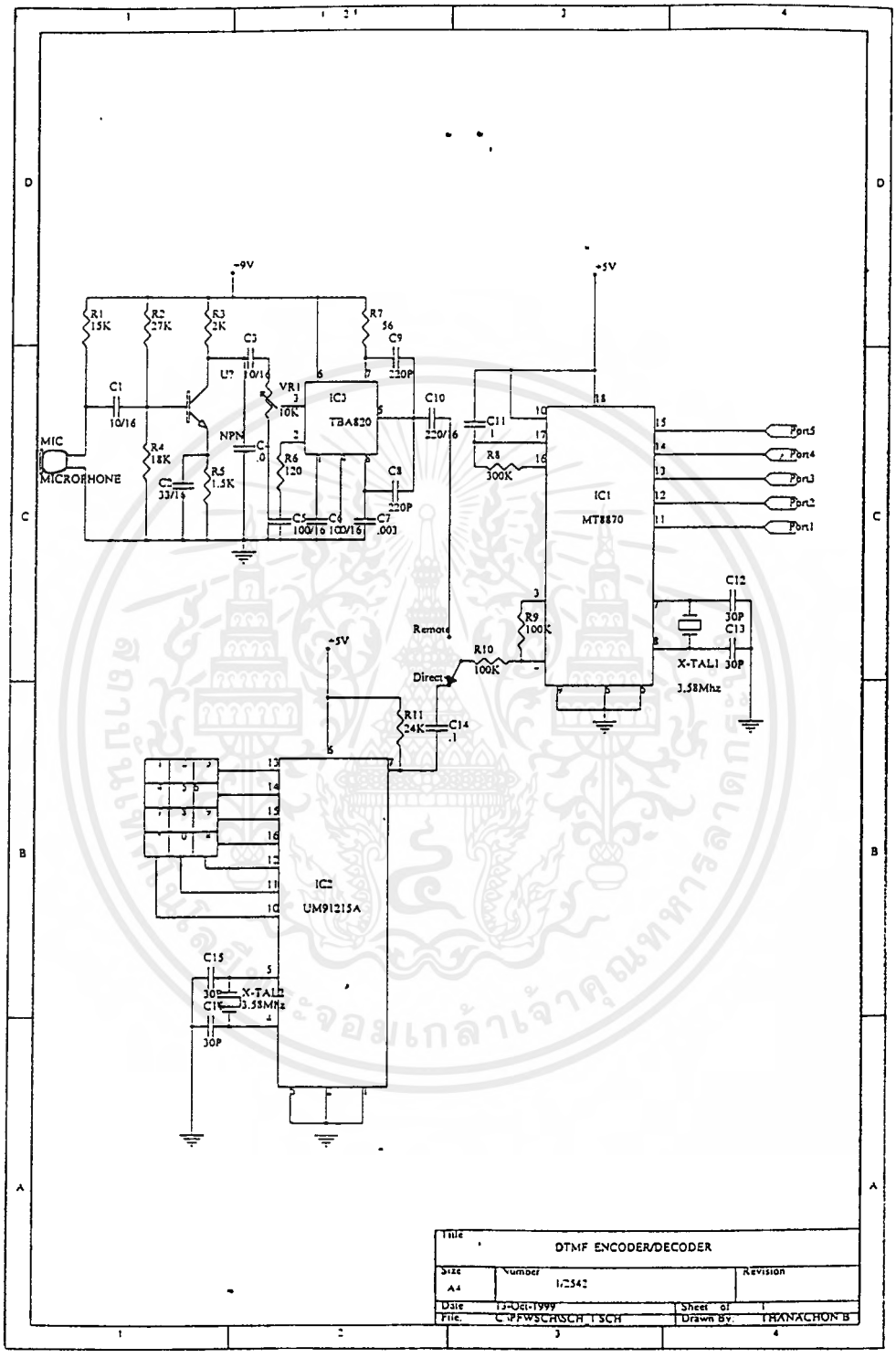
การดำเนินงานในการสร้างปริญญานิพนธ์นี้ พบว่ายังสามารถที่จะประยุกต์และดัดแปลงในส่วนต่าง ๆ ของโครงงานนี้ เพื่อนำมาใช้ประโยชน์อื่น ๆ ตามความประสงค์ได้อย่างกว้างขวางและจะก่อให้เกิดการพัฒนาให้ได้เครื่องมือหรืออุปกรณ์ต่าง ๆ เพื่อนำมาใช้ให้เกิดประโยชน์ในชีวิตประจำวัน ทางผู้จัดทำหวังเป็นอย่างยิ่งว่าจะมีผู้ที่นำโครงงานนี้มาพัฒนาและปรับปรุงให้สามารถใช้งานได้อย่างสะดวก รวมทั้งเพิ่มความสามารถในการควบคุมของโปรแกรมให้มากขึ้น

เอกสารอ้างอิง

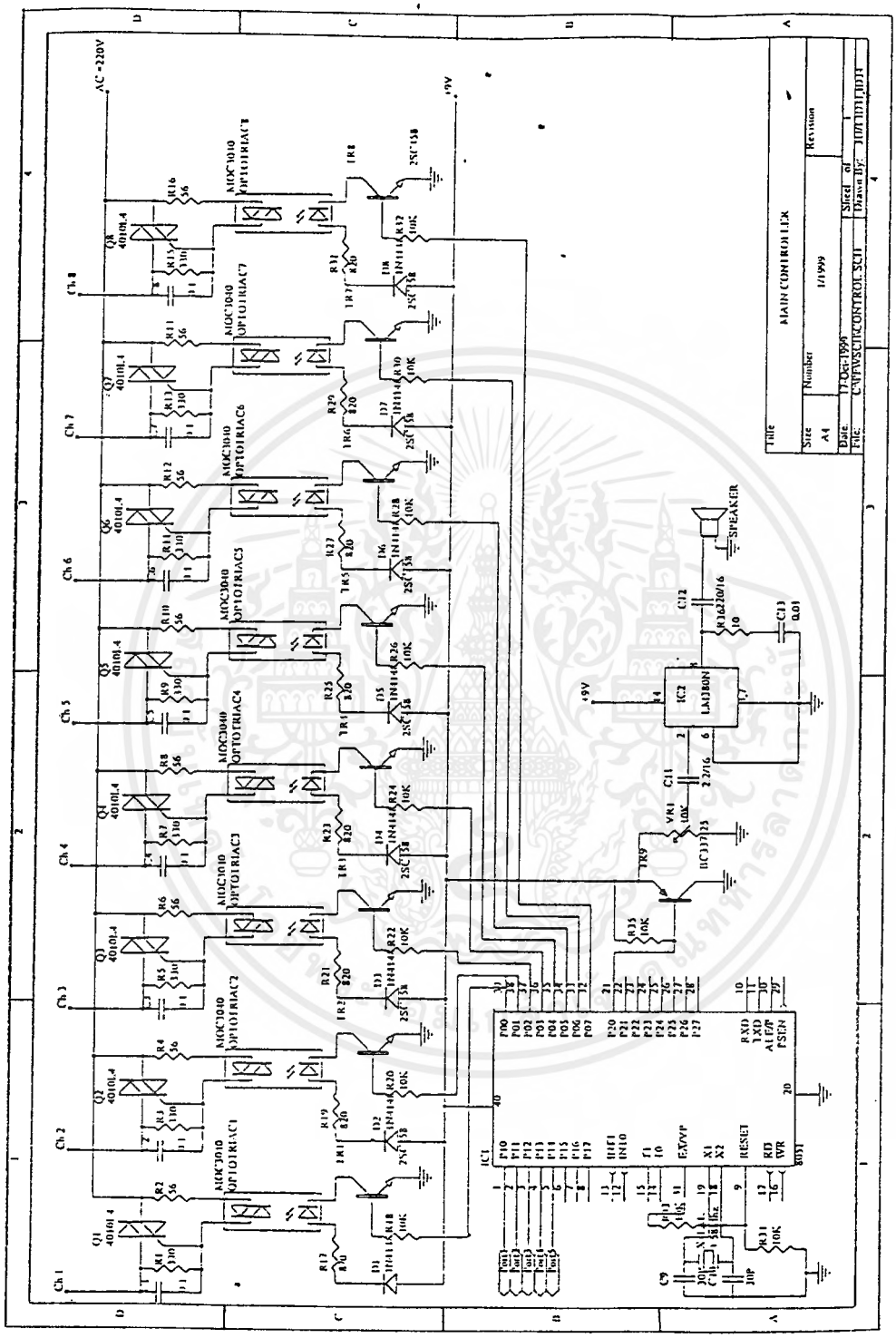
1. สุนทร วิทูสุรพจน์, การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051, บริษัท เอช.เอ็น.กรุ๊ป จำกัด, 2537
2. อุดม จีนประดับ, ไมโครคอนโทรลเลอร์, สำนักพิมพ์พระจอมเกล้าพระนครเหนือ, 2541
3. น.อ. ธวัชชัย เลื่อนฉวี, เทคโนโลยีโทรศัพท์, โรงพิมพ์ศุภาลัย, 2533

ภาคผนวก ก

วงจรการทำงาน

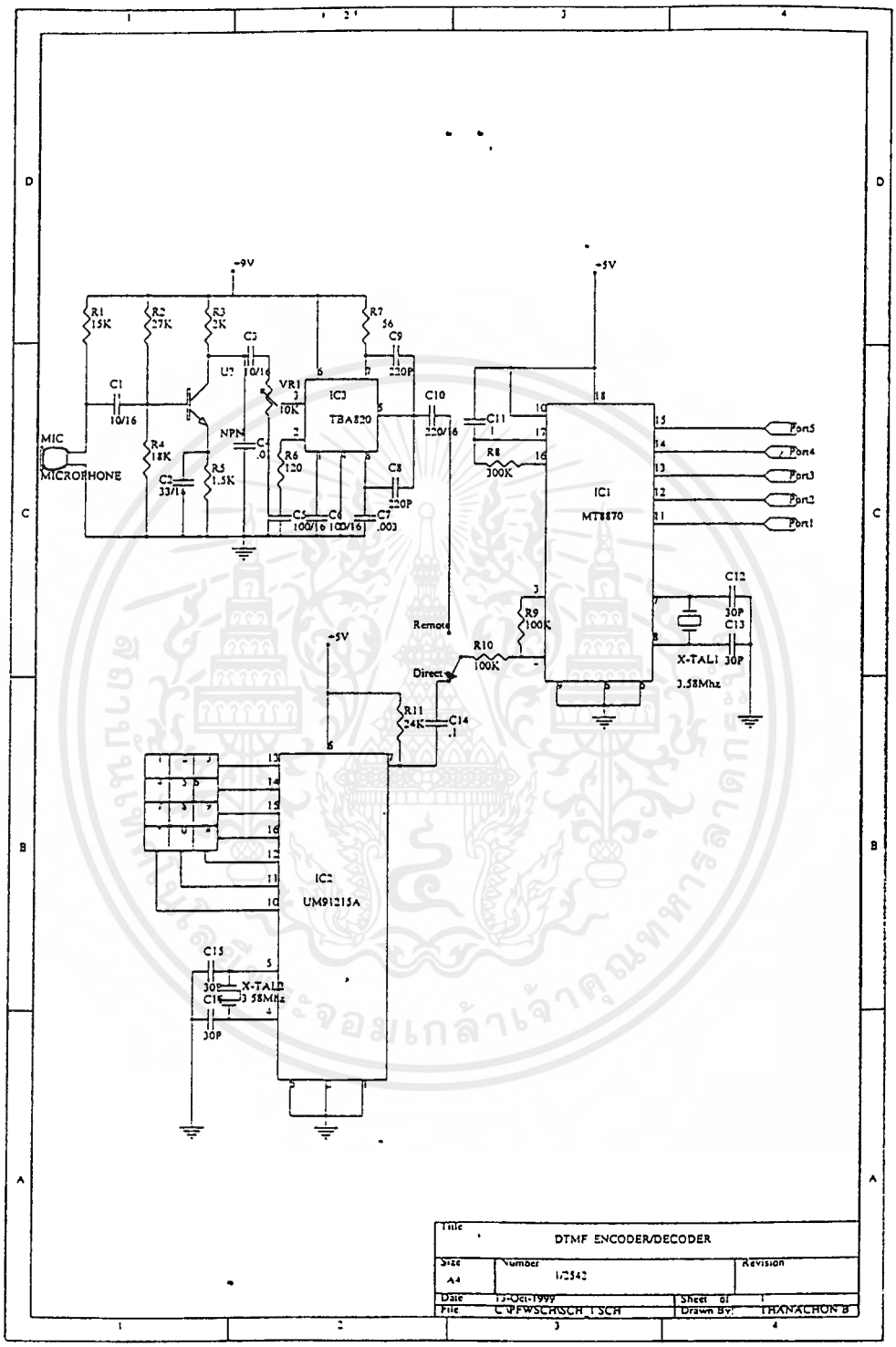


รูปที่ ก.1 วงจรการทำงานส่วนการเข้ารหัสและถอดรหัสสัญญาณ DTMF

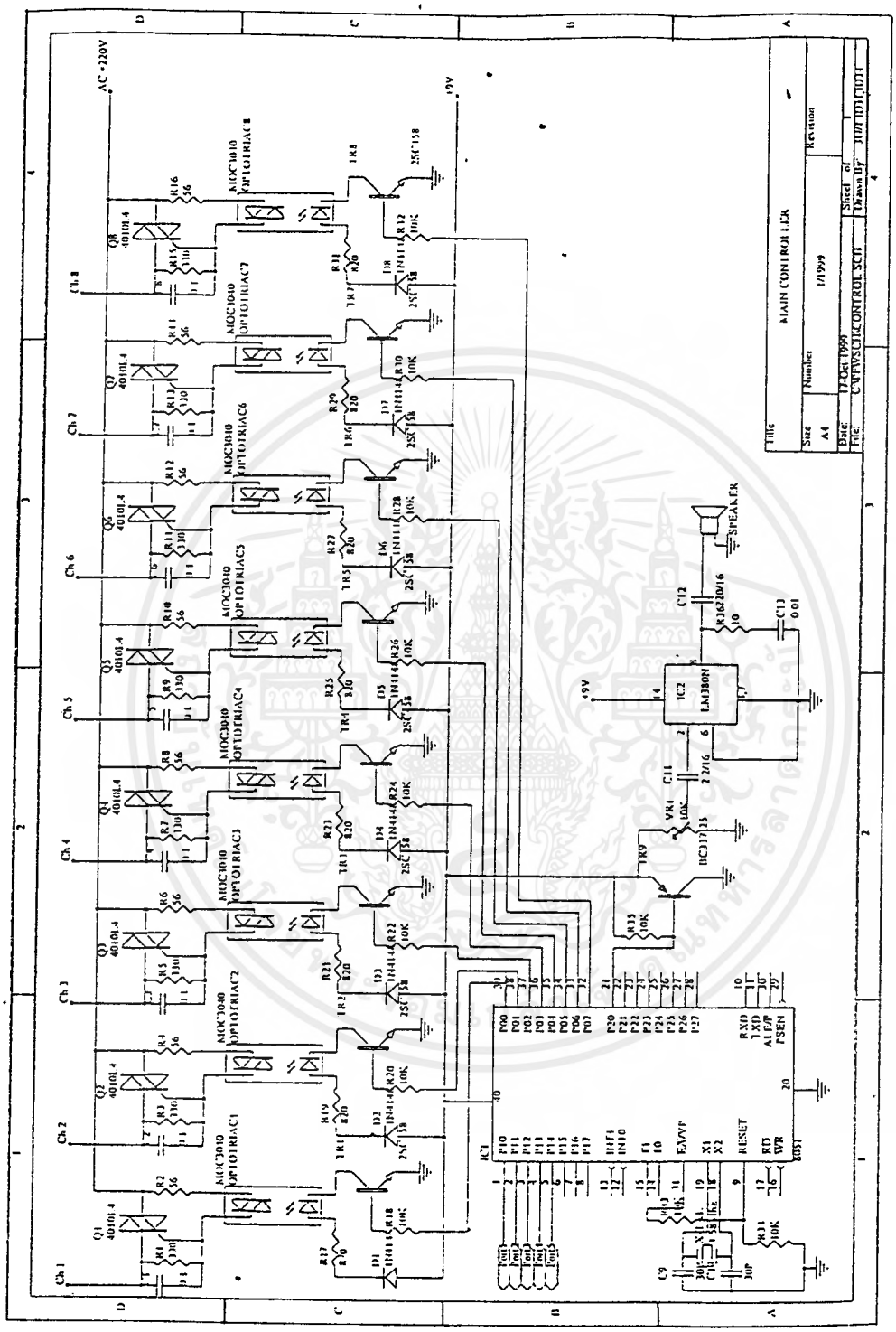


รูปที่ ก.2 วงจรการทำงานส่วนควบคุม

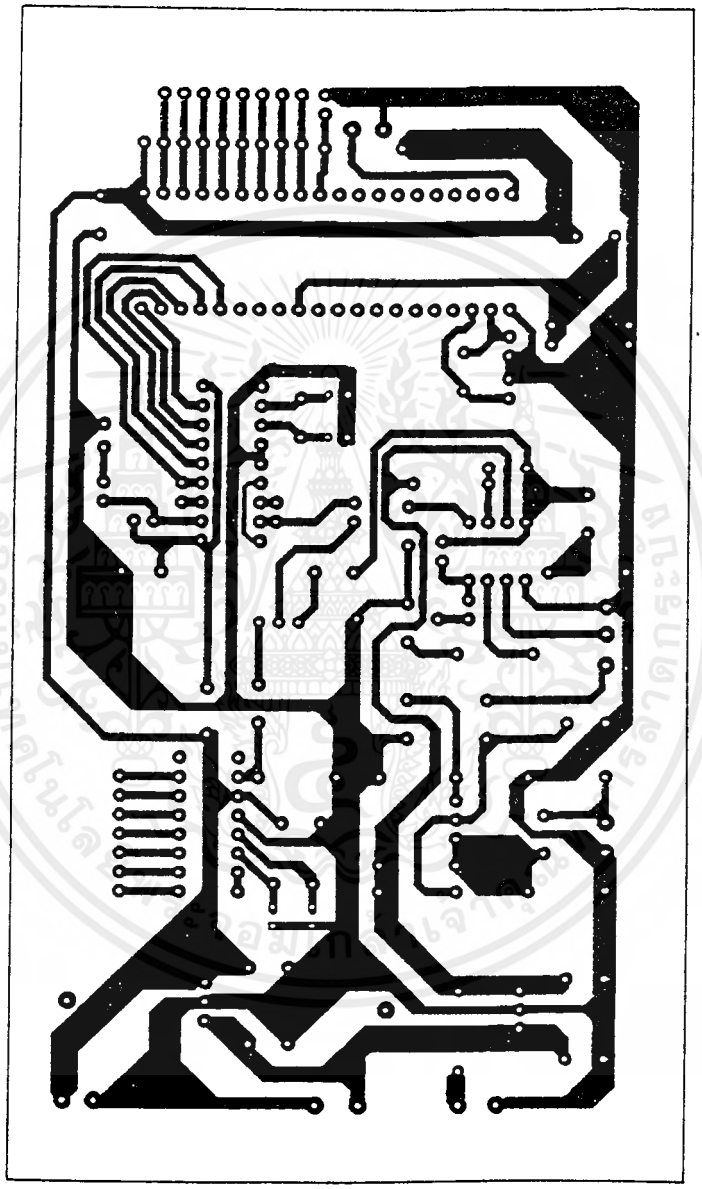
ภาคผนวก ก
วงจรการทำงาน



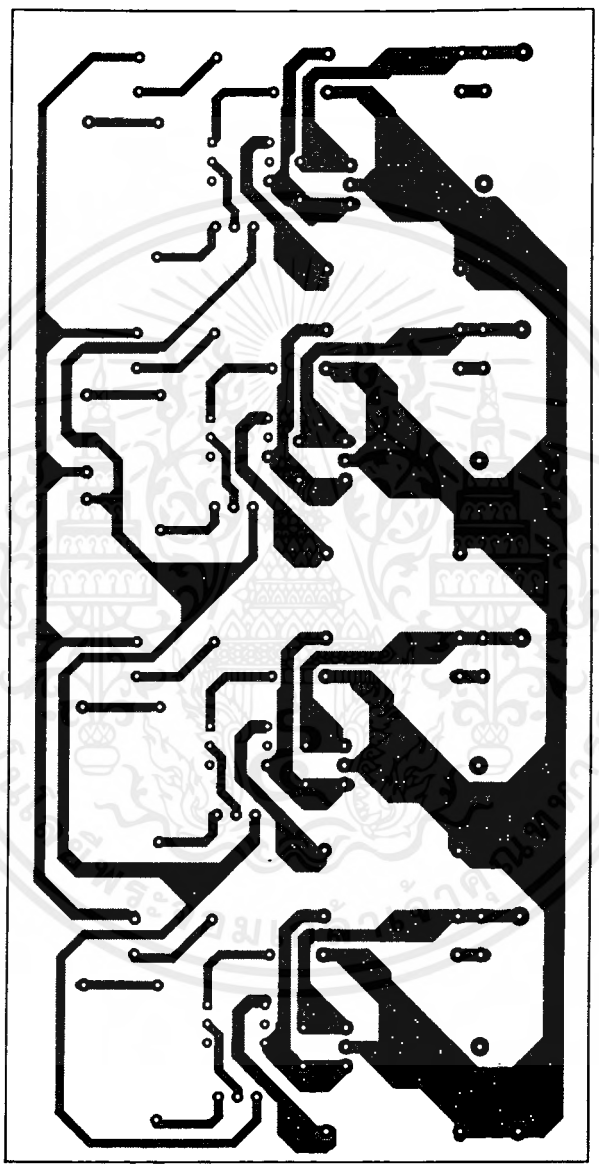
รูปที่ ก.1 วงจรการทำงานส่วนการเข้ารหัสและถอดรหัสสัญญาณ DTMF



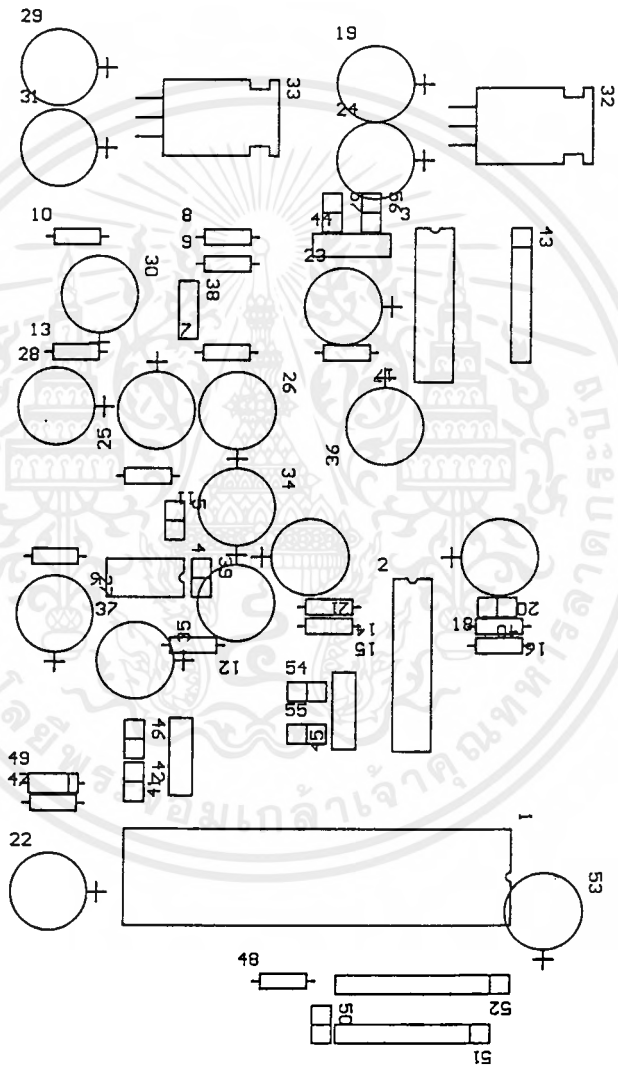
รูปที่ ก.2 วงจรการทำงานส่วนควบคุม



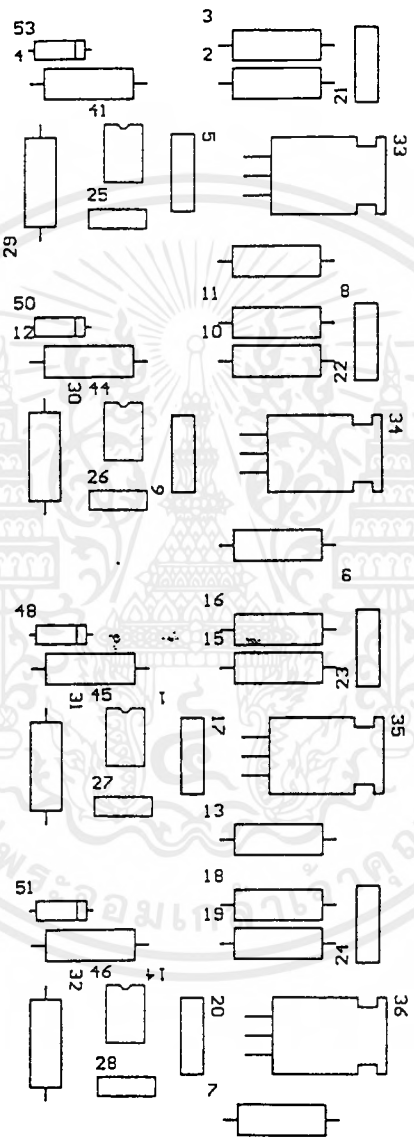
รูปที่ ก.3 ลายวงจรของแผ่นควบคุมหลัก



รูปที่ ก.4 ลายวงจรของ TRIAC ควบคุมอุปกรณ์



รูปที่ ก.5 การลงอุปกรณ์ของวงจรควบคุมหลัก



รูปที่ ก.6 การลงอุปกรณ์ของวงจร TRIAC

ภาคผนวก ข

โปรแกรมการทำงาน

โปรแกรมการทำงาน

```
OFF     EQU 0CH
ON      EQU 0BH
KEY     EQU P1
PORT    EQU P0
CK      EQU P1

TMP     EQU 2EH ;BYTE BIT

KEY1            EQU 30H                 ; PASSWORD KEY BUFFER
BKEY1           EQU 31H
BKEY2           EQU 32H
BKEY3           EQU 33H
BKEY4           EQU 34H

BPASS1          EQU 35H
BPASS2          EQU 36H
BPASS3          EQU 37H
BPASS4          EQU 38H

SAV_PASSW   EQU 00H ;BIT 00         ; PASSWORD  SAVED IS TRUE
PASSW_OK    EQU 01H ;BIT 01         ; PASSWORD CORRECT IS TRUE
PASS1       EQU 2                   ;  DEFAULT PASSWORD
PASS2       EQU 5
PASS3       EQU 8
PASS4       EQU 10   ;0

CK_BIT      EQU 74H  ; BIT  74H IN 2EH  ; CHECK BIT KEYPAD
SOUND       EQU P2.0                    ; PORT SOUND
```

```
                ORG  0000H

                MOV  A,#10                  ; INITIAL DELAY TIME 1 SEC
                CLR  SAV_PASSW              ; CLEAR FLAG PASSWORD SAVE
                CALL DELAY
                MOV  PORT,0H                ; CLEAR OUTPUT
RESTART:        MOV  R7,#2                  ; SOUND
BSTART:         CALL TONE1
                MOV  A,#3
                CALL DELAY
                DJNZ R7,BSTART

                CALL IN4KEY                 ; GET 4 KEY
                CALL CHK_PASSW              ; COMPARE KEY TO  PASSWORD
                JNB  PASSW_OK,RESTART       ; RESET

ISNUM:          CALL IN_KEY                 ; GET 1 KEY
ISNUM1:         CLR C
                MOV A,KEY1                  ; IF KEY IS NUMBER GOTO
                SUBB A,#0AH                  ISNUM FOR GETNEW KEY FIRST
                JC   ISNUM                   KEY MUST BE # OR *

CHK_ON:         MOV  A,KEY1                 ; IF KEY IS NOT * GOTO
                CJNE A,#ON,GO_CHK_OFF        GO_CHK_OFF
                CALL IN_KEY                 ; GET 1 KEY AND BEEP 1 TIME
                CALL TONE1 ;SOUND ON
                MOV  A,KEY1
                CJNE A,#0BH,CHK_ONALL       ; IF KEY #, *  GOTO CHK_ON ALL
```

```
NEWPASS:    MOV  R7,#2                    ; SOUND
NEWPASS1:   CALL TONE1
            MOV  A,#1                     ; TWO TONE  SOUND
            CALL DELAY
            CALL TONE2
            MOV  A,#1
            CALL DELAY
            DJNZ R7,NEWPASS1
            CALL IN4KEY                   ; CHECK OLD PASSWORD
            CALL CHK_PASSW
            JNB  PASSW_OK,RESTART         ; IF PASSWORD INCORRECT
            CALL IN4KEY                   ; GET NEW PASSWORD
            MOV  BPASS1,BKEY1             ; STORE PASSWORD TO BUFFER
            MOV  BPASS2,BKEY2
            MOV  BPASS3,BKEY3
            MOV  BPASS4,BKEY4
            SETB SAV_PASSW                ; SET FLAG SAVE PASSWORD
            CALL BPASSWOK                 ; SOUND
            SJMP ISNUM                    ; GOTO NEW CONTROL

GO_CHK_OFF: LJMP CHK_OFF
CHK_ONALL:  CJNE A,#0AH,CHK_ON1           ; IF KEY IS NOT 0
            MOV  PORT,#0FFH               CHECK ON ALL
            SJMP OK
CHK_ON1:    CJNE A,#01,CHK_ON2            ; CHECK ON BIT 0
            SETB PORT.0
            SJMP OK
CHK_ON2:    CJNE A,#02,CHK_ON3            ; CHECK ON BIT 1
            SETB PORT.1
            SJMP OK
```

```
CHK_ON3:    CJNE A,#03,CHK_ON4          ; CHECK ON BIT 2
            SETB PORT.2
            SJMP OK
CHK_ON4:    CJNE A,#04,CHK_ON5          ; CHECK ON BIT 3
            SETB PORT.3
            SJMP OK
CHK_ON5:    CJNE A,#05,CHK_ON6          ; CHECK ON BIT 4
            SETB PORT.4
            SJMP OK
CHK_ON6:    CJNE A,#06,CHK_ON7          ; CHECK ON BIT 5
            SETB PORT.5
            SJMP OK
CHK_ON7:    CJNE A,#07,CHK_ON8          ; CHECK ON BIT 6
            SETB PORT.6
            SJMP OK
CHK_ON8:    CJNE A,#08,OK1              ; CHECK ON BIT 7
            SETB PORT.7
OK:         LJMP ISNUM
OK1:        LJMP ISNUM1

CHK_OFF:    MOV  A,KEY1                 ; CHECK OFF
            CJNE A,#OFF,OK
            CALL IN_KEY
            CALL TONE2   ;SOUND OFF
            MOV  A,KEY1
            CJNE A,#0CH,CHK_OFFALL
            LJMP RESTART
```

```
CHK_OFFALL: CJNE A,#0AH,CHK_OFF1        ; CHECK OFF BIT 0
            MOV  PORT,#0
            SJMP OK
CHK_OFF1:   CJNE A,#01,CHK_OFF2         ; CHECK OFF BIT 1
            CLR  PORT.0
            SJMP OK
CHK_OFF2:   CJNE A,#02,CHK_OFF3         ; CHECK OFF BIT 2
            CLR  PORT.1
            SJMP OK
CHK_OFF3:   CJNE A,#03,CHK_OFF4         ; CHECK OFF BIT 3
            CLR  PORT.2
            SJMP OK
CHK_OFF4:   CJNE A,#04,CHK_OFF5         ; CHECK OFF BIT 4
            CLR  PORT.3
            SJMP OK
CHK_OFF5:   CJNE A,#05,CHK_OFF6         ; CHECK OFF BIT 5
            CLR  PORT.4
            SJMP OK;
CHK_OFF6:   CJNE A,#06,CHK_OFF7         ; CHECK OFF BIT 6
            CLR  PORT.5
            SJMP OK
CHK_OFF7:   CJNE A,#07,CHK_OFF8         ; CHECK OFF BIT 7
            CLR  PORT.6
            SJMP OK
CHK_OFF8:   CJNE A,#08,OK1              ; CHECK OFF BIT 8
            CLR  PORT.7
            SJMP OK
```

```
IN4KEY:     CALL IN_KEY                 ; GET 4 KEY AND
            MOV  BKEY1,KEY1             STORE TO KEY BUFFER
            CALL IN_KEY
            MOV  BKEY2,KEY1
            CALL IN_KEY
            MOV  BKEY3,KEY1
            CALL IN_KEY
            MOV  BKEY4,KEY1
            RET

CHK_PASSW: CLR PASSW_OK                 ; CHECK DEFAULT PASSWORD
            JB   SAV_PASSW,CK_NEW_PW
            MOV  A,BKEY1
            CJNE A,#PASS1,FALSE
            MOV  A,BKEY2
            CJNE A,#PASS2,FALSE
            MOV  A,BKEY3
            CJNE A,#PASS3,FALSE
            MOV  A,BKEY4
            CJNE A,#PASS4,FALSE
            CALL BPASSWOK
            SJMP END_CHK

CK_NEW_PW: MOV  A,BKEY1                 ; CHECK USER PASSWORD
            CJNE A,BPASS1,FALSE
            MOV  A,BKEY2
            CJNE A,BPASS2,FALSE
            MOV  A,BKEY3
            CJNE A,BPASS3,FALSE
            MOV  A,BKEY4
```

```
                CJNE A,BPASS4,FALS
                CALL BPASSWOK
                SJMP END_CHK
FALSE:          CALL BFALSE
END_CHK:        RET

BFALSE:         MOV  R7,#2                      ; TONE FALSE
PWFALSE:        CALL TONE2
                MOV  A,#3
                CALL DELAY
                DJNZ R7,PWFALSE
                RET

BPASSWOK:       MOV  R7,#2                      ; TONE PASSWORD OK
BPASSWOK1:      CALL TONE1
                MOV  A,#3
                CALL DELAY
                DJNZ R7,BPASSWOK1
                SETB PASSW_OK
                SJMP END_CHK
                RET

IN_KEY:         MOV  TMP,#0FFH
NO_PRESS:       MOV  TMP,KEY
                JNB  CK_BIT,NO_PRESS            ; LOOP WHILE NOT PRESS KEY
                MOV  A,#0FH
                ANL  A,TMP
                MOV  KEY1,A
```

```
NO_REL:   MOV  TMP,CK
          CPL  SOUND
          MOV A,#3
          CALL BDELAY
          JB  CK_BIT,NO_REL          ; LOOP WHILE RELEASE KEY
          RET

BDELAY:   MOV  R3,A                  ; SOUND
BDELAY3:  MOV  R4,#180
BDELAY1:  DJNZ R4,BDELAY1
          DJNZ R3,BDELAY3
          RET

DELAY:    MOV  R0,A                  ; DELAY
DELAY3:   MOV  R1,#180
DELAY2:   MOV  R2,#255
DELAY1:   DJNZ R2,DELAY1
          DJNZ R1,DELAY2
          DJNZ R0,DELAY3
          RET

TONE1:    MOV  A,#1
          MOV  R0,#4
TONE12:   MOV  R2,#255
TONE11:   CPL  SOUND
          CALL BDELAY
          DJNZ R2,TONE11
          DJNZ R0,TONE12
          RET
```

```
TONE2:      MOV  A,#4
            MOV  R0,#1
TONE22:     MOV  R2,#255
TONE21:     CPL  SOUND
            CALL BDELAY
            DJNZ R2,TONE21
            DJNZ R0,TONE22
            RET

            END
```

ภาคผนวก ค

ข้อมูลรายละเอียดของไมโครคอนโทรลเลอร์ 8051

ข้อมูลรายละเอียดของไอซีเบอร์ MT8870

ข้อมูลรายละเอียดของไอซีเบอร์ UM95087

MHS C51 INSTRUCTION SET

Interrupt Response time : Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings (1)

INSTRUC.	FLAG			INSTRUC.	FLAG		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLRC	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C,/ bit	X		
DIV	0	X		ORL C, bit		X	
DA	X			ORL C, bit		X	
RRC	X			MOV C, bit		X	
RLC	X			CJNE	X		
SETB C	1						

(1) note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on Instruction set and addressing modes :

Rn	– Register R7-R0 of the currently selected Register Bank
direct	– 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR (i.e., I/O port, control register, status register, etc. (128-255)).
@ Ri	– 8-bit internal data RAM location (0-255) addresses indirectly through register R1 or R0.
# data	– 8-bit constant included in instruction.
# data 16	– 16-bit constant included in instruction.
addr 16	– 16-bit destination address. Used by LCALL & LJMP. Abranch can be anywhere within the 64K-byte Program memory address space
addr 11	– 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction
rel	– Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditionnal jumps. Range is–128 to + 127 bytes relative to first byte of the following instruction.
bit	– Direct Addressed bit in internal Data RAM or special Function Register.

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
ARITHMETIC OPERATIONS			
ADD A, Rn	Add register to Accumulator	1	12
ADD A, direct	Add direct byte to Accumulator	2	12
ADD A, @Ri	Add indirect RAM to Accumulator	1	12
ADD A, #data	Add immediate data to Accumulator	2	12
ADDCA, Rn	Add register to Accumulator with Carry	1	12
ADDCA, direct	Add direct byte to Accumulator with Carry	2	12
ADDCA, @Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDCA, #data	Add immediate data to Acc with Carry	2	12
SUBB A, Rn	Subtract Register from Acc with borrow	1	12
SUBB A, direct	Subtract direct byte from Acc with borrow	2	12
SUBB A, @Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB A, #data	Subtract immediate data from Acc with borrow	2	12
INC A	Increment Accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment direct RAM	1	12
DEC A	Decrement Accumulator	1	12
DEC Rn	Decrement Register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12

Table 10 : 80C51 Instruction Set Summary.

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
ARITHMETIC OPERATIONS (continued)			
INC DPTR	Increment Data Pointer	1	24
MUL AB	Multiply A & B	1	48
DIV AB	Divide A by B	1	48
DA A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS			
ANL A, Rn	AND Register to Accumulator	1	12
ANL A, direct	AND direct byte to Accumulator	2	12
ANL A, @Ri	AND indirect RAM to Accumulator	1	12
ANL A, #data	AND immediate data to Accumulator	2	12
ANL direct, A	AND Accumulator to direct byte	2	12
ANL direct, #data	AND immediate data to direct byte	3	24
ORL A, Rn	OR register to Accumulator	1	12
ORL A, direct	OR direct byte to Accumulator	2	12
ORL A, @Ri	OR indirect RAM to Accumulator	1	12
ORL A, #data	OR immediate data to Accumulator	2	12
ORL direct, A	OR Accumulator to direct byte	2	12
ORL direct, #data	OR immediate data to direct byte	3	24
XRL A, Rn	Exclusive-OR register to Accumulator	1	12
XRL A, direct	Exclusive-OR direct byte to accumulator	2	12
XRL A, @Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL A, #data	Exclusive-OR immediate data to Accumulator	2	12
XRL direct, A	Exclusive-OR Accumulator to direct byte	2	12
XRL direct, #data	Exclusive-OR immediate data to direct byte	3	24
CLR A	Clear Accumulator	1	12
CPL A	Complement Accumulator	1	12

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
LOGICAL OPERATIONS (continued)			
RL A	Rotate Accumulator Left	1	12
RLC A	Rotate Accumulator Left through the Carry	1	12
RR A	Rotate Accumulator Right	1	12
RRC A	Rotate Accumulator Right through the Carry	1	12
SWAP A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER			
MOV A, Rn	Move Register to Accumulator	1	12
MOV A, direct	Move direct byte to Accumulator	2	12
MOV A, @Ri	Move indirect RAM to Accumulator	1	12
MOV A, #data	Move immediate data to Accumulator	2	12
MOV Rn, A	Move Accumulator to register	1	12
MOV Rn, direct	Move direct byte to register	2	24
MOV Rn, #data	Move immediate data to register	2	12
MOV direct, A	Move Accumulator to direct byte	2	12
MOV direct, Rn	Move register to direct byte	2	24
MOV direct, direct	Move direct byte to direct	3	24
MOV direct, @Ri	Move indirect RAM to direct byte	2	24
MOV direct, #data	Move immediate data to direct byte	3	24
MOV @Ri, A	Move Accumulator to indirect RAM	1	12

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
DATA TRANSFER (continued)			
MOV @Ri, direct	Move direct by to indirect RAM	2	24
MOV @Ri, #data	Move immediate data to indirect RAM	2	12
MOV DPTR, #data16	Load Data Pointer with a 16-bit constant	3	24
MOVCA, @A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVCA, @A+PC	Move Code byte relative to PC to Acc	1	24
MOVXA, @Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX A, @DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX@Ri, A	Move Acc to External RAM (8-bit addr)	1	24
MOVX @DPTR, A	Move Acc to External RAM (16-bit addr)	1	24
PUSH direct	Push direct byte only stack	2	24
POP direct	Pop direct byte from stack	2	24
XCH A, Rn	Exchange register with Accumulator	1	12
XCH A, direct	Exchange direct byte with Accumulator	2	12
XCH A, @Ri	Exchange indirect RAM with Accumulator	1	12
XCHD A, @Ri	Exchange loworder Digit indirect RAM with Acc	1	12

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry	1	12
CLR bit	Clear direct bit	2	12
SETB C	Set Carry	1	12
SETB bit	Set direct bit	2	12
CPL C	Complement Carry	1	12
CPL bit	Complement direct bit	2	12
ANL C, bit	AND direct bit to Carry	2	24
ANL C, /bit	AND complement of direct bit to Carry	2	24
ORL C, bit	OR direct bit to Carry	2	24
ORL C, /bit	OR complement of direct bit to Carry	2	24
MOV C, bit	Move direct bit to Carry	2	12
MOV bit, C	Move Carry to direct bit	2	24
JC rel	Jump if Carry is set	2	24
JNC rel	Jump if Carry not set	2	24
JB bit, rel	Jump if direct Bit is set	3	24
JNB bit, rel	Jump if direct Bit is Not set	3	24
JBC bit, rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING			
ACALLK addr11	Absolute Subroutine Call	2	24
LCALL addr16	Long Subroutine Call	3	24
RET	Return from Subroutine	1	24
RETI	Return from interrupt	1	24
AJMPaddr11	Absolute Jump	2	24
LJMPaddr16	Long Jump	3	24
SJMP rel	Short Jump (relative addr)	2	24

MNEMONIC	DESCRIPTION	BYTE	OSCIL. PERIOD
PROGRAM BRANCHING (continued)			
JMP @A+DPTR	Jump direct relative to the DPTR	1	24
JZ rel	Jump if Accumulator is zero	2	24
JNZ rel	Jump if Accumulator is not Zero	2	24
CNJE A, direct, rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE A, #data, rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE Rn, #data, rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE @Ri, #data, rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ Rn, rel	Decrement register and Jump if Not Zero	2	24
DJNZ direct, rel	Decrement direct byte and Jump if Not Zero	3	24
NOP	No Operation	1	12

INSTRUCTION DEFINITIONS

ACALL addr 11

Function : Absolute Call

Description : ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2 K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example : Initially SP equals 07H. The labs " SUBRTN " is at program memory location 0345 H. After executing the instruction,

ACALL SUBRTN

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

Bytes : 2

Cycles : 2

Encoding :

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation :

ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
[(SP)] ← (PC_{7-0})
(SP) ← (SP) + 1
[(SP)] ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD a, <src-byte>

Function : Add

Description : ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6 ; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed : register, direct, register-indirect, or immediate.

Example : The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A, R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A, Rn

Byte : 1

Cycle : 1

Encoding :

0 0 1 0	1 r r r

Operation : ADD
(A) ← (A) + (Rn)

ADD A, direct

Bytes : 2

Cycle : 1

Encoding :

0 0 1 0	0 1 0 1	direct address

Operation : ADD
(A) ← (A) + (direct)

ADD A, @RI

Byte : 1

Cycle : 1

Encoding :

0 0 1 0	1 1 1 i

Operation : ADD
(A) ← (A) + ((RI))

ADD A, # data

Bytes : 2

Cycle : 1

Encoding :

0 0 1 0	0 1 0 0	Immediate data

Operation : ADD
(A) ← (A) + # data

ADDC A, <src-byte>

Function : Add with Carry

Description : ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry or bit flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occured.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6 ; otherwise OV is cleared. When adding signed intergers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing mode are allowed ; register, direct, register-indirect, or immediate.

Example : The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A, R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A, RN

Byte : 1

Cycle : 1

Encoding :

0	0	1	1	1	r	r	r

Operation : ADDC
$(A) \leftarrow (A) + (C) + (R_n)$

ADDC A, direct

Bytes : 2

Cycle : 1

Encoding :

0	0	1	1	0	1	0	1	direct address

Operation : ADDC
$(A) \leftarrow (A) + (C) + (direct)$

ADDC A, @ RI

Byte : 1

Cycle : 1

Encoding :

0	0	1	1	0	1	1	i

Operation : ADDC
$(A) \leftarrow (A) + (C) + ((R_i))$

ADDC A, #data

Bytes : 2

Cycle : 1

Encoding :

0	0	1	1	0	1	0	0	immediate data

Operation : ADDC
$(A) \leftarrow (A) + (C) + \#\ data$

AJMP addr11

Function : Absolute Jump

Description : AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (*after* incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2 K block of program memory as the first byte of the instruction following AJMP.

Example : The label " JMPADR " is at program memory location 0123H. The instruction,

AJMP JMPADR

is a location 0345H and will load the PC with 0123H.

ADD A, direct

Bytes : 2

Cycles : 2

Encoding :

a10 a9 a8 0	0 0 0 1

a7 a6 a5 a4	a3 a2 a1 a0

Operation : AJMP
(PC) ← (PC) + 2
($PC_{10\text{-}0}$) ← page address

ANL <dest-byte>, <src-byte>

Function : Logical-AND for byte variables

Description : ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.
The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing ; when the destination is a direct address, the source can be the Accumulator or immediate data. *Note* : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example : If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,
ANL A, R0
will leave 41H (01000001B) in the Accumulator.
When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,
ANL P1, #01110011B
will clear bits 7, 3, and 2 of output port 1.

ANL A, Rn

Bytes : 1

Cycles : 1

Encoding :

0 1 0 1	1 r r r

Operation : ANL
(A) ← (A) ∧ (Rn)

ANL A, direct

Bytes : 2

Cycles : 1

Encoding :

0 1 0 1	0 1 0 1

direct address

Operation : ANL
(A) ← (A) ∧ (direct)

ANL A, @ RI

Byte : 1

Cycle : 1

Encoding :

0	1	0	1	0	1	1	i

Opération : ANL
(A) ← (A) ∧ ((Ri))

ANL A, #DATA

Bytes : 2

Cycle : 1

Encoding :

0	1	0	1	0	1	0	0	immediate data

Operation : ANL
(A) ← (A) ∧ # data

ANL direct, A

Bytes : 2

Cycle : 1

Encoding :

0	1	0	1	0	0	1	0	direct address

Operation : ANL
(direct) ← (direct) ∧ (A)

ANL direct, # data

Bytes : 3

Cycles : 2

Encoding :

0	1	0	1	0	0	1	1	direct address	immediate data

Operation : ANL
(direct) ← (direct) ∧ # data

ANL C, <src-bit>

Function : Logical-AND for bit variables

Description : If the Boolean value of the source bit is logical 0 then clear the carry flag ; otherwise leave the carry flag in its current state. A slash (" / ") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, *but the source bit itself is not affected.* No other flags are affected.

Only direct addressing is allowed for the source operand.

Example : Set the carry flag if, P1.0 = 1, ACC.7 = 1, and OV = 0 :

```
MOV C, P1.0          ; LOAD CARRY WITH INPUT PIN STATE
ANL C, ACC.7         ; AND CARRY WITH ACCUM. BIT 7
ANL C,/OV            ; AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C, bit

Bytes : 2

Cycles : 2

Encoding :

Operation :

ANL
(C) ← (C) ∧ (bit)

ANL C, /bit

Bytes : 2

Cycles : 2

Encoding :

Operation :

ANL
(C) ← (C) ∧ $\overline{(bit)}$

CJNE <dest-byte>, <src-byte>, rel

Function : Compare and Jump if Not Equal

Description : CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte> ; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations : the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example : The Accumulator contains 34H, register 7 contains 56H. The first instruction in the sequence,

```
                CJNE   R7, #60H, NOT_EQ
;               ...    ...               ; R7 = 60H
NOT_EQ :        JC     REQ_LOW           ; IF R7 < 60H
;               ...    ...               ; R7 > 60H
```

sets the carry flag and branches to the instruction at label NOT-EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

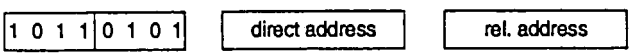
WAIT : CJNE A, P1, WAIT

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H).

CJNE A, direct, rel

Bytes : 3
Cycles : 2

Encoding :



Operation :

```
(PC) ← (PC) + 3
IF (A) <> (direct)
THEN
        (PC) ← (PC) + relative offset1
IF (A) < (direct)
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE A, # data, rel

Bytes : 3
Cycles : 2

Encoding :

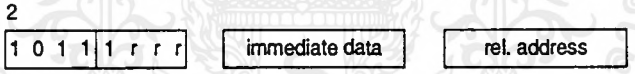
Operation :

```
(PC) ← (PC) + 3
IF (A) <> (data)
THEN
        (PC) ← (PC) + relative offset
IF (A) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE Rn, # data, rel

Bytes : 3
Cycles : 2

Encoding :



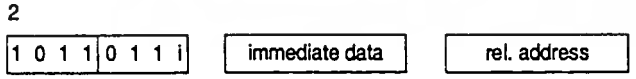
Operation :

```
(PC) ← (PC) + 3
IF (Rn) <> data
THEN
        (PC) ← (PC) + relative offset
IF (Rn) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CJNE @Ri, # data, rel

Bytes : 3
Cycles : 2

Encoding :



Operation :

```
(PC) ← (PC) + 3
IF (Ri) <> data
THEN
        (PC) ← (PC) + relative offset
IF ((Ri)) < data
THEN
        (C) ← 1
ELSE
        (C) ← 0
```

CLR A

Function : Clear Accumulator

Description : The Accumulator is cleared (all bits set on zero). No flags are affected.

Example : The Accumulator contains 5CH (01011100B). The instruction,
CLRA
Will leave the Accumulator set to 00H (00000000B).

Bytes : 1

Cycles : 1

Encoding :

1 1 1 1	0 1 0 0

Operation :
CLR
(A) ← 0

CLR bit

Function : Clear bit

Description : The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example : Port 1 has previously been written with 5DH (01011101B). The instruction,
CLR P1.2
will leave the port set to 59H (01011001B).

CLR C

Bytes : 1

Cycles : 1

Encoding :

1 1 0 0	0 0 1 1

Operation :
CLR
(C) ← 0

CLR bit

Bytes : 2

Cycles : 1

Encoding :

1 1 0 0	0 0 1 0	bit address

Operation :
CLR
(bit) ← 0

CPLA

Function : Complement Accumulator

Descritpion : Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example : The accumulator contains 5CH (01011100B). The instruction,
CPLA
will leave the Accumulator set to 0A3H (10100011B).

Bytes : 1

Cycles : 1

Encoding :

1 1 1 1	0 1 0 0

Operation :
CPL
$(A) \leftarrow \overline{(A)}$

CPL bit

Function : Complement bit

Description : The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.
Note : When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example : Port 1 has previously been written with 5BH (01011101B). The instruction sequence.

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011011B).

CPL C

Bytes : 1

Cycles : 1

Encoding :

1 0 1 1	0 0 1 1

Operation : CPL
$(C) \leftarrow \overline{(C)}$

CPL bit

Bytes : 2

Cycles : 1

Encoding :

1 0 1 1	0 0 1 0	bit address

Operation : CPL
$(bit) \leftarrow \overline{(bit)}$

DA A

Function : Decimal-adjust Accumulator for Addition .

Description : DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits, Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx - 111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note : DA A *cannot* simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal substraction.

Example : The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC  A, R3
DA    A
```

will first perform a standard twos-complement binary addition, resulting in the value 0BEH (10111110), in the Accumulator. The carry and auxillary carry flags will be cleared.

The decimal Adjust instruction will then after the Accumulator to the value 24H (00100100B) indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56,67, and the carry-in. The carry flag will set by the Decimal Adjust instruction, indicating that a decimal overflow occured. The true sum 56,67, and 1 is 124.
BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD  A, #99H
DA   A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 - 1 = 29.

Bytes : 1

Cycles : 1

Encoding :

1 1 0 1	0 1 0 0

Operation :

```
DA
- contents of Accumulator are BCD
IF [[(A3 - 0) > 9] V [(AC) = 1]]
        THEN (A3 - 0) ← (A3 - 0) + 6
                AND
IF [[(A7 - 4) > 9] V [(C) = 1]]
        THEN (A7 - 4) ← (A7 - 4) + 6
```

DEC byte

Function : Decrement

Description : The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed : accumulator, register, direct, or register-indirect.

Note : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example : Register 0 contains 7FH (01111111B). Internal RAM locations 7 EH and 7FH contain 00H and 40H, respectively. The instruction sequence.

DEC @ R0

DEC R0

DEC @ R0

will leave register 0 set to 7EH internal RAM locations 7EH and 7FH to 0FFH and 3FH.

DEC A

Bytes : 1

Cycles : 1

Encoding :

0 0 0 1	0 1 0 0

Operation : DEC
(A) ← (A) - 1

DEC Rn

Bytes : 1

Cycles : 1

Encoding :

0 0 0 1	1 r r r

Operation : DEC
(Rn) ← (Rn) - 1

DEC direct

Bytes : 2

Cycles : 1

Encoding :

0 0 0 1	0 1 0 1

direct address

Operation : DEC
(direct) ← (direct) - 1

DEC @ RI

Bytes : 1

Cycles : 1

Encoding :

0 0 0 1	0 1 1 i

Operation : DEC
((Ri)) ← ((Ri)) - 1

DIV AB

Function : Divide

Description : DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient ; register B receives the integer remainder. The carry and OV flags will be cleared.
Exception : If B had originally contained 00H ; the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example : The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 x 18) +.17. Carry and OV will both be cleared.

Bytes : 1

Cycles : 4

Encoding :

1 0 0 0	0 1 0 0

Operation :

DIV
$(A)_{15-8}$
$\leftarrow$ (A)/(B)
$(B)_{7-0}$

DJNZ <byte>, <rel-addr>

Function : Decrement and Jump if Not Zero

Description : DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example : Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H, respectively. the instruction sequence,

DJNZ 40H, LABEL_1

DJNZ 50H, LABEL_2

DJNZ 60H, LABEL_3

will cause a jump to the instruction at label LABEL2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was *not* taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
          MOV   R2, #8
TOGGLE :  CPL   P1.7
          DJNZ  R2, TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles ; two for DJNZ and one to after the pin.

DJNZ Rn, rel

Bytes : 2

Cycles : 2

Encoding :

1 1 0 1	1 r r r	rel. address

Operation :

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) - 1
IF (Rn) > 0 or (Rn) < 0
        THEN
                (PC) ← (PC) + rel
```

DJNZ direct, rel

Bytes : 3

Cycles : 2

Encoding :

1 1 0 1	0 1 0 1	direct address	rel. address

Operation :

```
DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) - 1
IF (direct) > 0 or (direct) < 0
        THEN
                (PC) ← (PC) + rel
```

INC <byte>

Function : Increment

Description : INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. There addressing modes are allowed : register, direct, or register-indirect.

Note : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, *not* the input pins.

Example : Register 0 contains 7EH (01111110B). Internal locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

INC @R0

INC R0

INC @R0

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes : 1

Cycles : 1

Encoding :

0 0 0 0	0 1 0 0

Operation :

```
INC
(A) ← (A) + 1
```

INC Rn

Bytes : 1

Cycles : 1

Encoding :

0 0 0 0	1 r r r

Operation :

```
INC
(Rn) ← (Rn) + 1
```

INC direct

Bytes : 2

Cycles : 1

Encoding :

0 0 0 0	0 1 0 1	direct address

Operation : INC
(direct) ← (direct) + 1

INC @ RI

Bytes : 1

Cycles : 1

Encoding :

0 0 0 0	0 1 1 i

Operation : INC
((Ri)) ← ((Ri)) + 1

INC DPTR

Function : Increment Data Pointer

Description : Increment the 16-bit data pointer by 1.A 16-bit increment (modulo 2^{16}) is performed ; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example : Registers DPH and DPL contain 12H and 0FEH, respectively. The instruction sequence,

INC DPTR

INC DPTR

INC DPTR

will change DPH and DPL to 13H and 01H.

Bytes : 1

Cycles : 2

Encoding :

1 0 1 0	0 0 1 1

Operation : INC
(DPTR) ← (DPTR) + 1

JB bit,rel

Function : Jump if Bit set

Descritpion : If the indicated bit is a one, jump to the address indicated ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Note : When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example : The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence.

JB P1.2, LABEL 1
JB ACC.2, LABEL 2

will cause program execution to branch to the instruction at label LABEL 2.

Bytes : 3

Cycles : 2

Encoding :

Operation :

```
JB
(PC) ← (PC) + 3
IF (bit) = 1
        THEN
                (PC) ← (PC) + rel
```

JBC bit, rel

Function : Jump if Bit is set and Clear bit

Description : If the indicated bit is a one, branch to the address indicated ; otherwise proceed with the next instruction. *The bit will not be cleared if it is already a zero.* The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note : When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, *not* the input pin.

Example : The Accumulator holds 56H (01010110B). The instruction sequence,

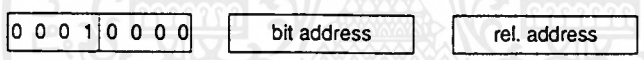
```
JBC ACC.3, LABEL 1
JBC ACC.2, LABEL 2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes : 3

Cycles : 2

Encoding :



Operation :

```
JBC
(PC) ← (PC) + 3
IF (bit) = 1
        THEN
                (bit) ← 0
                (PC) ← (PC) + rel
```

JC rel

Function : Jump if Carry is set

Description : If the carry flag is set, branch to the address indicated ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example : The carry flag is cleared. The instruction sequence,

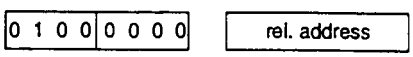
```
JC      LABEL 1
CPL     C
JC      LABEL 2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes : 2

Cycles : 2

Encoding :



Operation :

```
JC
(PC) ← (PC) + 2
IF (C) = 1
        THEN
                (PC) ← (PC) + rel
```

JMP @A + DPTR

Function : Jump indirect

Description : Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}) : a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example : An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP-TBL :

```
                MOV   DPTR, #JMP_TBL
                JMP   @ A + DPTR
JMP_TBL :       AJMP  LABEL0
                AJMP  LABEL1
                AJMP  LABEL2
                AJMP  LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remembers that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes : 1

Cycles : 2

Encoding :

0 1 1 1	0 0 1 1

Operation :
JMP
(PC) ← (A) + (DPTR)

JNB bit, rel

Function : Jump if Bit not set

Description : If the indicated bit is a zero, branch to the indicated address ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. *The bit tested is not modified.* No flags are affected.

Example : The data present at input port 1 is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB P1.3, LABEL1
JNB ACC3, LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes : 3

Cycles : 2

Encoding :

0 0 1 1	0 0 0 0	bit address	rel. address

Operation :
JNB
(PC) ← (PC) + 3
IF (bit) = 0
THEN (PC) ← (PC) + rel

JNC rel

Function : Jump if Carry not set

Description : If the carry flag is a zero, branch to the address indicated ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example : The carry flag is set. The instruction sequence,

```
JNC LABEL1
CPLC
JNC LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes : 2

Cycles : 2

Encoding :

0 1 0 1	0 0 0 0	rel. address

Operation :

```
JNC
(PC) ← (PC) + 2
IF (C) = 0
        THEN (PC) ← (PC) + rel
```

JNZ rel

Function Jump if Accumulator Not Zero

Description : If any bit of the Accumulator is a one, branch to the indicated address ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example : The Accumulator originally holds 00H. The instruction sequence,

```
JNZ LABEL1
INC A
JNZ LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes : 2

Cycles : 2

Encoding :

0 1 1 1	0 0 0 0	rel. address

Operation :

```
JNZ
(PC) ← (PC) + 2
IF (A) ≠ 0
        THEN (PC) ← (PC) + rel
```

JZ rel

Function : Jump if Accumulator Zero

Description : If all bits of the Accumulator are zero, branch to the address indicated ; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example : The Accumulator originally contains 01H. The instruction sequence.

```
JZ      LABEL1
DEC     A
JZ      LABEL2
```

will change the Accumulator to 00H and cause program execution at the instruction identified by the label LABEL2.

Bytes : 2

Cycles : 2

Encoding :

0 1 1 0 0 0 0 0	rel. address

Operation :

JZ
(PC) ← (PC) + 2
IF (A) = 0
THEN (PC) ← (PC) + rel

LCALL addr16

Function : Long call

Description : LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space. No flags are affected.

Example : Initially the Stack Pointer equals 07H. The label " SUBRTN " is assigned to program memory location 1234H. After executing the instruction,

LCALL SUBRTN

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1235H.

Bytes : 3

Cycles : 2

Encoding :

0 0 0 1 0 0 1 0	addr15-addr8	addr7-addr0

Operation :

LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (SP_{15-8})
(PC) ← $addr_{15-0}$

LJMP addr16

Function : Long Jump

Description : LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example : The label " JMPADR " is assigned to the instruction at program memory location 1234H. The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes : 3

Cycles : 2

Encoding :

0 0 0 0 0 0 1 0	addr15-addr8	addr7-addr0

Operation : LJMP
(PC) ← $addr_{15-0}$

MOV <dest-byte>, <src-byte>

Function : Move byte variable

Description : The byte variable indicated the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinaisons of source and destination addressing modes are allowed.

Example : Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port 1 is 11001010B (0CAH).

```
MOV   R0, #30H      ; R0 <= 30h
MOV   A, @ R0       ; A <= 40H
MOV   R1, A         ; R1 <= 40h
MOV   R, @ R1       ; B <= 10h
MOV   @ R1, P1      ; RAM (40H) <= OCAH
MOV   P2, P1        ; P2 # 0CAH
```

leaves the value 30H in register 0,40H in both the Accumulator and register 1,10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A, Rn

Bytes : 1

Cycles : 1

Encoding :

1 1 1 0 1 r r r

Operation : MOV
(A) ← (Rn)

***MOV A,direct**

Bytes : 2

Cycles : 1

Encoding :

1 1 1 0 0 1 0 1	direct address

Operation : MOV
(A) ← (direct)

*MOV A, ACC is not valid instruction.

MOV A,@ RI

Bytes : 1

Cycles : 1

Encoding :

1 1 1 0	0 1 1 i

Operation : MOV
(A) ← (Ri)

MOV A, # data

Bytes : 2

Cycles : 1

Encoding :

0 1 1 1	0 1 0 0	immediate data

Operation : MOV
(A) ← # data

MOV Rn, A

Bytes : 1

Cycles : 1

Encoding :

1 1 1 1	1 r r r

Operation : MOV
(Rn) ← (A)

MOV Rn, direct

Bytes : 2

Cycles : 2

Encoding :

1 0 1 0	1 r r r	direct addr.

Operation : MOV
(Rn) ← (direct)

MOV Rn, # data

Bytes : 2

Cycles : 1

Encoding :

0 1 1 1	1 r r r	immediate data

Operation : MOV
(Rn) ← # data

MOV direct, A

Bytes : 2

Cycles : 1

Encoding :

1 1 1 1	0 1 0 1	direct address

Operation : MOV
(direct) ← (A)

MOV direct, Rn

Bytes : 2

Cycles : 2

Encoding :

1 0 0 0	1 r r r	direct address

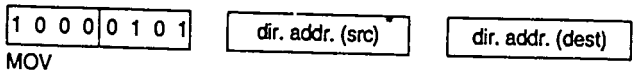
Operation : MOV
(direct) ← (Rn)

MOV direct, direct

Bytes : 3

Cycles : 2

Encoding :



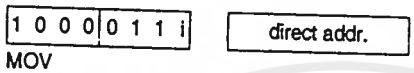
Operation : MOV
(direct) ← (direct)

MOV direct, @ Ri

Bytes : 2

Cycles : 2

Encoding :



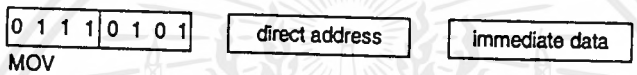
Operation : MOV
(direct) ← (Ri)

MOV direct, # data

Bytes : 3

Cycles : 2

Encoding :



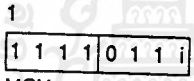
Operation : MOV
(direct) ← # data

MOV @ Ri, A

Bytes : 1

Cycles : 1

Encoding :



Operation : MOV
((Ri)) ← (A)

MOV @ Ri, direct

Bytes : 2

Cycles : 2

Encoding :

Operation : MOV
((Ri)) ← (direct)

MOV @ Ri+, data

Bytes : 2

Cycles : 1

Encoding :

Operation : MOV
((Ri)) ← # data

MOV <dest-bit>, <src-bit>

Function : More bit data

Description : The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag ; the other may be any directly addressable bit. No other register or flag is affected.

Example : The carry flag is originally set. The data present at input Port 3 is 11000101B. The data previously written to output Port 1 is 35H (00110101B).

```
MOV     P1.3, C
MOV     C, P3.3
MOV     P1.2, C
```

will leave the carry cleared and change Port 1 to 39H (00111001B).

MOV C, bit

Bytes : 2

Cycles : 1

Encoding :

1 0 1 0	0 0 1 0	bit address

Operation :
MOV
(C) ← (bit)

MOV bit, C

Bytes : 2

Cycles : 2

Encoding :

1 0 0 1	0 0 1 0	bit address

Operation :
MOV
(bit) ← (C)

MOV DPTR, # data 16

Function : Load Data Pointer with a 16-bit constant

Description : The Data Pointer is loaded with the 16-bit constant indicated. the 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction which moves 16-bits of data at once.

Example : The instruction,

MOV DPTR, 1234H

will load the value 1234H into the Data Pointer : DPH will hold 12H and DPL will hold 34H.

Bytes : 3

Cycles : 2

Encoding :

1 0 0 1	0 0 0 0	immed. data 15-8	immed. data 7-0

Operation :
MOV
(DPTR) ← # $data_{15-0}$
DPH DPL ← # $data_{15-8}$ # $data_{7-0}$

MOVC A, @ A + <base-reg>

Function : Move Code byte

Description : The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, PC is incremented to the address of the following instruction before being added with the Accumulator ; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example : A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL PC :    INC     A
            MOVC    A, @ A + PC
            RET
            DB      66H
            CB      77H
            CB      88H
            DB      99H
```

If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to " get around " the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A, @ A + DPTR

Bytes : 1

Cycles : 2

Encoding :

1	0	0	1	0	0	1	1

Operation : MOVC
(A) ← ((A) + (DPTR))

MOVC A, @ A + PC

Bytes : 1

Cycles : 2

Encoding :

1	0	0	0	0	0	1	1

Operation : MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

MOVX <dest-byte>, <src-byte>

Function : Move External

Description : The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the " X " appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.
In the first type, the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.
In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64K bytes), since no additional instructions are needed to set up the output ports.
It is possible in some situation to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example : An external 256 byte RAM using multiplexed address/data lines is connected to the 80C51 Port 0. Port 3 provides control lines for the external RAM. Ports 0 and 2 are used for normal I/O. Registers 0 and 1 contain 12H and 34H. Location 34H of the external RAM holds the value 56H. The instruction sequence

```
MOVX    A, @ R1
MOVX    @ R0, A
```

copies the value 56H into both the Accumulator and external RAM location 12H.

MOVX A, @ Ri

Bytes : 1

Cycles : 2

Encoding :

1 1 1 0	0 0 1 i

Operation : MOVX
(A) ← ((Ri))

MOVX @ Ri, A

Bytes : 1

Cycles : 2

Encoding :

1 1 1 1	0 0 1 i

Operation : MOVX
((Ri)) ← (A)

MOVX A, @ DPTR

Bytes : 1

Cycles : 2

Encoding :

1 1 1 0	0 0 0 0

Operation : MOVX
(A) ← ((DPTR))

MOVX @ DPTR, A

Bytes : 1

Cycles : 2

Encoding :

1 1 1 1	0 0 0 0

Operation : MOVX
(DPTR) ← (A)

MUL AB

Function : Multiply

Description : MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (OFFH) the overflow flag is set ; otherwise it is cleared. The carry flag is always cleared.

Example : Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (OAOH). The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes : 1

Cycles : 4

Encoding :

1 0 1 0	0 1 0 0

Operation :

MUL
$(A)_{7-0} \leftarrow (A) \times (B)$
$(B)_{15-8}$

NOP

Function : No Operation

Description : Execution continue at the following instruction. Other than the PC, no registers or flags are effected.

Example : It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enable) with the instruction sequence.

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETP    P2.7
```

Bytes : 1

Cycles : 1

Encoding :

0 0 0 0	0 0 0 0

Operation :

NOP
$(PC) \leftarrow (PC) + 1$

ORL <dest-byte> <src-byte>

Function : Logical-OR for byte variables

Description : ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte, No flags are affected.
The two operands allow six addressing mode combinaisons. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing ; when the destination is a direct address, the source can be the Accumulator or immediate data.
Note : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example : If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction,
ORL A, R0
will leave the Accumulator holding the value 0D7H (11010111B).
When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction.,
ORL P1, # 00110010b
will set bits 5, 4, and 1 of output Port 1.

ORL A, Rn

Bytes : 1

Cycles : 1

Encoding :

0 1 0 0	1 r r r

Operation ORL
(A) ← (A) V (Rn)

ORL A, direct

Bytes : 2

Cycles : 1

Encoding :

0 1 0 0	0 1 0 1	direct address

Operation : ORL
(A) ← (A) V (direct)

ORL A, @ Ri

Bytes : 1

Cycles : 1

Encoding :

0 1 0 0	0 1 1 i

Operation : ORL
(A) ← (A) V ((Ri))

ORL A, # data

Bytes : 2

Cycles : 1

Encoding :

0 1 0 0	0 1 0 0	immediate data

Operation ORL
(A) ← (A) V # data

ORL direct, A

Bytes : 2

Cycles : 1

Encoding :

0 1 0 0	0 0 1 0	direct address

Operation : ORL
(direct) ← (direct) V (A)

ORL direct, # data

Bytes : 3

Cycles : 2

Encoding : 0 1 0 0 | 0 0 1 1 | direct address | immediate data

Operation : ORL
(direct) ← (direct) V # data

ORL C, <src-bit>

Function : Logical-OR for bit variable

Description : Set the carry flag if the Boolean value is a logical 1 ; leave the carry in its current state otherwise. A slash (" / ") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit it self is not affected. No other flags are affected.

Example : Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0 :

```
MOV   C, P1.0      ; LOAD CARRY WITH INPUT PIN P10
ORL   C, ACC.7     ; OR CARRY WITH THE ACC. BIT7
ORL   C,/OV        ; OR CARRY WITH THE INVERSE OF OV
```

ORL C, bit

Bytes : 2

Cycles : 2

Encoding : 0 1 1 1 | 0 0 1 0 | bit address

Operation : ORL
(C) ← (C) V (bit)

ORL C, /bit

Bytes : 2

Cycles : 2

Encoding : 1 0 1 0 | 0 0 0 0 | bit address

Operation : ORL
(C) ← (C) V $\overline{(bit)}$

POP direct

Function : Pop from stack.

Description : The contents of internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example : The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP DPH
POP DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H)

Bytes : 2

Cycles : 2

Encoding : 1 1 0 1 | 0 0 0 0 | direct address

Operation : POP
(direct) ← ((SP))
(SP) ← (SP) – 1

PUSH direct

Function : push onto stack.

Description : The Stack Pointer is incremented by one. The contents fo the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example : On entering interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH DPL
PUSH DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM location 0AH and 0BH, respectively.

Bytes : 2

Cycles : 2

Encoding :

1 1 0 0	0 0 0 0	direct address

Operation :

```
PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)
```

RET

Function : Return from subroutine

Description : RET pops the high-and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following en ACALL or LCALL. No flags are affected.

Example : The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H, and 01H, respectively. The instruction,

```
RET
```

will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.

Bytes : 1

Cycles : 2

Encoding :

0 0 1 0	0 0 1 0

Operation :

```
RET
(PC15 - 8) ← ((SP))
(SP) ← (SP) – 1
(PC7-0) ← ((SP))
(SP) ← (SP) – 1
```

RETI

Function : Return from interrupt

Description : RETI pops the high-and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected ;the PSW is *not* automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower-or-same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example : The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction,

```
RETI
```

will leave the Stack Pointer equal to 09H and return program execution to location 0123H.

Bytes : 1
Cycles : 2
Encoding :

0	0	1	1	0	0	1	0

Operation : RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RL A

Function : Rotate Accumulator Left
Description : The eight bits in the Accumulator are rotated one bit to the left. Bit 7 rotated into the bit 0 position. No flags are affected.
Example : The Accumulator holds the value 0C5H (11000101B). The instruction,
RL A
leaves the Accumulator holding the value 8BH (100001011B) with the carry unaffected.
Bytes : 1
Cycles : 1
Encoding :

0	0	1	0	0	0	1	1

Operation : RL
$(An + 1) \leftarrow (An)$ n = 0 - 6
$(A0) \leftarrow (A7)$

RLC A

Function : Rotate Accumulator Left through the Carry flag
Description : The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag ; the original state of the carry flag moves into the bit 0 position. No other flags are affected.
Example : The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,
RCL A
leaves the Accumulator holding the value 8BH (10001010B) with the carry set.
Bytes : 1
Cycles : 1
Encoding :

0	0	1	1	0	0	1	1

Operation : RLC
$(An + 1) \leftarrow (An)$ n = 0 - 6
$(A0) \leftarrow (C)$
$(C) \leftarrow (A7)$

RR A

Function : Rotate Accumulator Right

Description : The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example : The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (11100010B) with the carry unaffected.

Bytes : 1

Cycles : 1

Encoding :

0 0 0 0	0 0 1 1

Operation :

RR
(An) ← (An + 1) n = 0 - 6
(A7) ← (A0)

RRC A

Function : Rotate Accumulator Right through Carry flag

Description : The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag ; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example : The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes : 1

Cycles : 1

Encoding :

0 0 0 1	0 0 1 1

Operation :

RRC
(An) ⇐ (A_{n+1})n = 0 - 6
(A7) ← (C)
(C) ← (A0)

SETB <bit>

Function : Set bit

Description : SETB sets the indicated bit to one. SETB can operate on the carry flag or any direct addressable bit. No other flags are affected.

Example : The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

```
SETB    C
SETB    P1.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35H (00110101B).

SETB C

Bytes : 1

Cycles : 1

Encoding :

1 1 0 1	0 0 1 1

Operation :
SETB
(C) ← 1

SETB bit

Bytes : 2

Cycles : 1

Encoding :

1 1 0 1	0 0 1 0	bit address

Operation :
SETB
(bit) ← 1

SJMP rel

Function : Short Jump

Description : Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example : The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

```
SJMP RELADR
```

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(*Note*: Under the above conditions the instruction following SJMP will be at 102H. therefore, the displacement byte of the instruction will be the relative offset (0123H - 0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be an one-instruction infinite loop).

Bytes : 2

Cycles : 2

Encoding :

1 0 0 0	0 0 0 0	rel. address

Operation :
SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A, <src-byte>

Function : Subtract with borrow

Description : SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit7, and clears C otherwise. (If C was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision substraction so the carry is subtracted from the Accumulator along with the source operand). AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.
When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.
The source operand allows four addressing modes : register, direct, register-indirect, or immediate.

Example : The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. the instruction,

SUBB A, R2

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision substraction, it should not be explicity cleared by a CLRC instruction.

SUBB A, Rn

Bytes : 1

Cycles : 1

Encoding :

1	0	0	1	1	r	r	r

Operation : SUBB
(A) ← (A) – (C) – (Rn)

SUBB A, direct

Bytes : 2

Cycles : 1

Encoding :

1	0	0	1	0	1	0	1

direct address

Operation : SUBB
(A) ← (A) – (C) – (direct)

SUBB A, @ Ri

Bytes : 1

Cycles : 1

Encoding :

1	0	0	1	0	1	1	i

Operation : SUBB
(A) ← (A) – (C) – (Ri)

SUBB A, # data

Bytes : 2

Cycles : 1

Encoding :

1	0	0	1	0	1	0	0

immediate data

Operation : SUBB
(A) ← (A) – (C) – # data

SWAP A

Function : Swap nibbles within the Accumulator

Description : SWAP A interchanges the low-and high-order nibbles (four-bit fields) of the Accumulator (bits 3 - 0 and bits 7 - 4). The operation can also be thought of a four-bit rotate instruction. No flag are affected.

Example : The Accumulator holds the value 0C5H (11000101B). The instruction,

SWAP A

leave the Accumulator holding the value 5CH (01011100B).

Bytes : 1

Cycles : 1

Encoding :

1	1	0	0	0	1	0	0

Operation : SWAP
$(A_{3-0}) \rightleftarrows (A_{7-4})$

XCH A, <byte>

Function : Exchange Accumulator with byte variable

Description : XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example : R0 contains the addres 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCH A, @R0

will leave RAM location 20H holding the values 3FH (00111111B) and 75H (01110101B) in the Accumulator.

XCH A, Rn

Bytes : 1

Cycles : 1

Encoding :

1	1	0	0	1	r	r	r

Operation : XCH
$(A) \rightleftarrows (Rn)$

XCH A, direct

Bytes : 2

Cycles : 1

Encoding :

1	1	0	0	0	1	0	1

direct address

Operation : XCH
$(A) \rightleftarrows$ (direct)

XCH A, @RI

Bytes : 1

Cycles : 1

Encoding :

1	1	0	0	0	1	1	i

Operation : XCH
$(A) \rightleftarrows ((Ri))$

XCHD A, @ RI

Function : Exchange Digit

Description : XCHD exchanges the low-order nibble of the Accumulator (bits 3 - 0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (7 - 4) of each register are not affected. No flags are affected.

Example : R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCHD A, @ R0

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes : 1

Cycles : 1

Encoding :

1 1 0 1	0 1 1 i

Operation : XCHD

$(A_{3-0}) \rightleftarrows ((Ri_{3-0}))$

XRL <dest-byte>, <src-byte>

Function : Logical Exclusive-OR for byte variable

Description : XRL performs the bitwise logical Exclusive- OR operation between the indicated variables, storing the results in the destination. No flags are affected.
The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing ; when the destination is a direct address, the source can be the accumulator or immediate data.

(*Note* : When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins).

Example : If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

XRL A, R0

will leave the Accumulator holding the value 69H (01101001B).
When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL P1, # 00110001B

will complement bits 5, 4, and 0 of output Port 1.

XRL A, Rn

Bytes : 1

Cycles : 1

Encoding :

0 1 1 0	1 r r r

Operation : XRL

(A) ← (A) ∀ (Rn)

XRL A, direct

Bytes : 2

Cycles : 1

Encoding :

0 1 1 0	0 1 0 1	direct address

Operation : XRL

(A) ← (A) ∀ (direct)

XRL A, @ RI

Bytes : 1

Cycles : 1

Encoding :

0	1	1	0	0	1	1	i

Operation : XRL
(A) ← (A) ∀ ((Ri))

XRL A, #data

Bytes : 2

Cycles : 1

Encoding :

0	1	1	0	0	1	0	0	immediate data

Operation : XRL
(A) ← (A) ∀ # data

XRL direct, A

Bytes : 2

Cycles : 1

Encoding :

0	1	1	0	0	0	1	0	direct address

Operation : XRL
(direct) ← (direct) ∀ (A)

XRL direct, # data

Bytes : 3

Cycles : 2

Encoding :

0	1	1	0	0	0	1	1	direct address	immediate data

Operation : XRL
(direct) ← (direct) ∀ # data

DATA SHEET

80C31/80C51

CMOS SINGLE-CHIP 8 BIT MICROCONTROLLER

- 80C31/80C51 : 0 TO 12 MHz
  80C31/80C51-1 : 0 TO 16 MHz
  80C31-S/80C51-S : 0 TO 20 MHz
  80C31/80C51-L : 0 TO 6 MHz WITH 2.7 V < Vcc < 6 V
  80C51F : 80C51 WITH PROTECTED ROM

FEATURES

- POWER CONTROL MODES
- 128 x 8 BIT RAM
- 4 K BYTES OF ROM (80C51)
- 32 PROGRAMMABLE I/O LINES
- TWO 16 BIT TIMER/COUNTER
- 64 K PROGRAM MEMORY SPACE
- FULLY STATIC DESIGN
- BOOLEAN PROCESSOR
- 5 INTERRUPT SOURCES
- PROGRAMMABLE SERIAL PORT
- 64 K DATA MEMORY SPACE
- TEMPERATURE RANGE : COMMERCIAL, INDUSTRIAL, AUTOMOTIVE AND MILITARY

INTRODUCTION

MHS's 80C31 and 80C51 are high performance CMOS versions of the 8031/8051 NMOS single chip 8 bit µC and are manufactured using a self-aligned silicon gate CMOS process (SAJI VI).

The fully static design of the MHS 80C31/80C51 allows to reduce system power consumption by bringing the clock frequency down to any value, even DC, without loss of data.

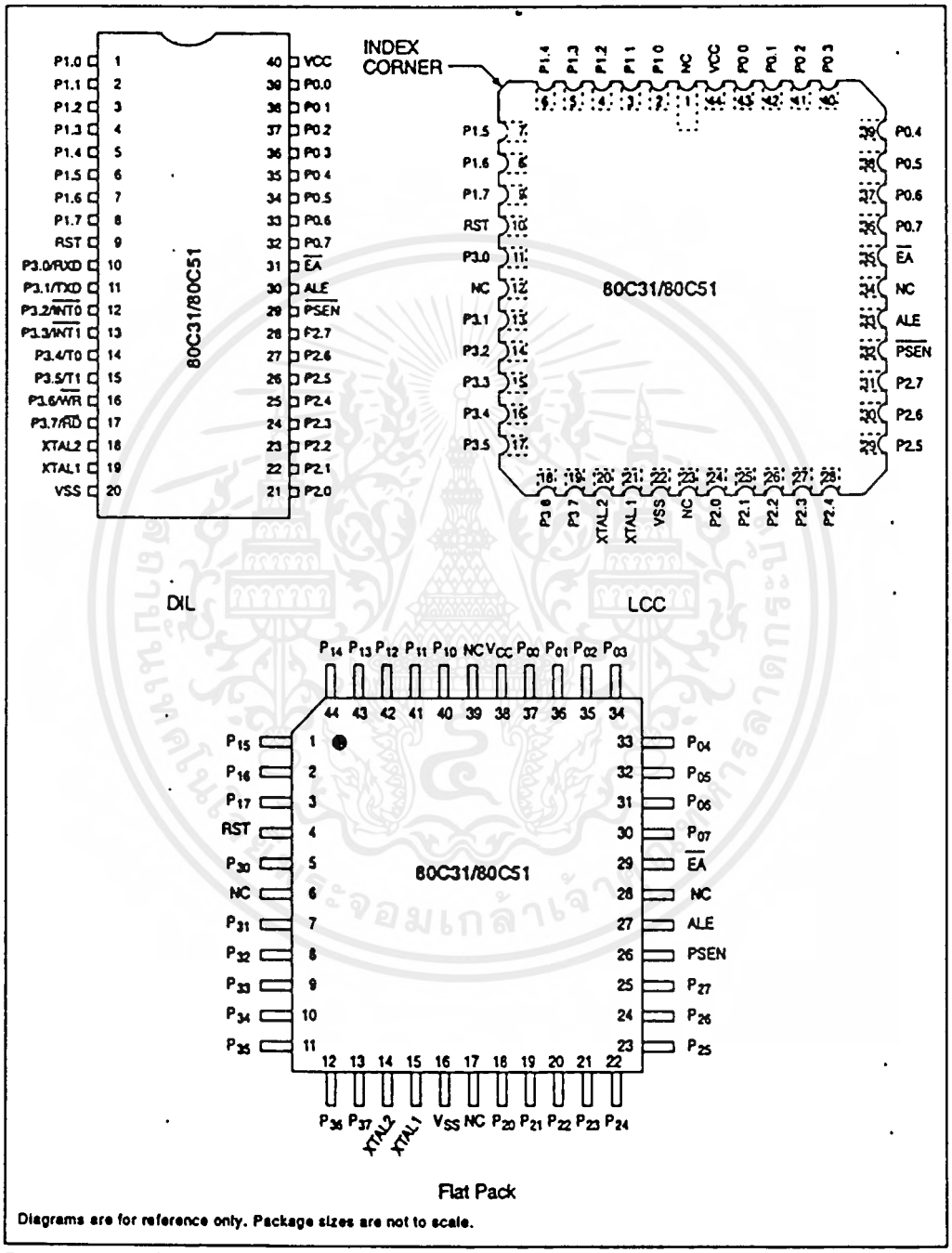
The 80C51 retains all the features of the 8051 : 4 K bytes of ROM ; 128 bytes of RAM ; 32 I/O lines ; two 16 bit timers ; a 5-source, 2-level interrupt structure ; a full duplex serial port ; and on-chip oscillator and clock circuits.

In addition, the 80C51 has two software-selectable modes of reduced activity for further reduction in power consumption. In the Idle Mode the CPU is frozen while the RAM, the timers, the serial port, and the interrupt system continue to function. In the Power Down Mode the RAM is saved and all other functions are inoperative.

The 80C31 is identical to the 80C51 except that it has no on-chip ROM.

INTERFACE

PIN CONFIGURATION



Diagrams are for reference only. Package sizes are not to scale.

Figure 1.

PIN DESCRIPTION

Vss

Circuit ground potential

Vcc

Supply voltage during normal, Idle, and Power Down operation.

Port 0

Port 0 is an 8 bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in that state 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 pullups when emitting 1's. Port 0 also outputs the code bytes during program verification in the 80C51. External pullups are required during program verification. Port 0 can sink eight LS TTL inputs.

Port 1

Port 1 is an 8 bit bi-directional I/O port with internal pullups. Port 1 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address byte during program verification. In the 80C51, Port 1 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

Port 2

Port 2 is an 8 bit bi-directional I/O port with internal pullups. Port 2 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups. 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 pullups when emitting 1's. During accesses to external Data Memory that use 8 bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

It also receives the high-order address bits and control signals during program verification in the 80C51. Port 2 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

Port 3

Port 3 is an 8 bit bi-directional I/O port with internal pullups. Port 3 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the pullups. It also serves the function of various special features of the MHS 51 Family, as listed below.

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external Data Memory write strobe)
P3.7	RD (external Data Memory read strobe)

Port 3 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

RST

A high level on this for two machine cycles while the oscillator is running resets the device. An internal pull-down resistor permits Power-On reset using only a capacitor connected to Vcc.

ALE

Address Latch Enable output for latching the low byte of the address during accesses to external memory. ALE is activated as though for this purpose at a constant rate of 1/6 the oscillator frequency except during an external data memory access at which time on ALE pulse is skipped. ALE can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pullup.

$\overline{PSEN}$

Program Store Enable output is the read strobe to external Program Memory. PSEN is activated twice each machine cycle during fetches from external Program Memory. (However, when executing out of external Program Memory, two activations of PSEN are skipped during each access to external Data Memory). PSEN is not activated during fetches from internal Program Memory. PSEN can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pullup.

$\overline{EA}$

When EA is held high, the CPU executed out of internal Program Memory (unless the Program Counter exceeds 0FFFH). When EA is held low, the CPU executes only out of external Program Memory. EA must not be floated.

XTAL1

Input to the inverting amplifier that forms the oscillator. Receives the external oscillator signal when an external oscillator is used.

XTAL2

Output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. This pin should be floated when an external oscillator is used.

FUNCTIONAL DESCRIPTION

BLOCK DIAGRAM

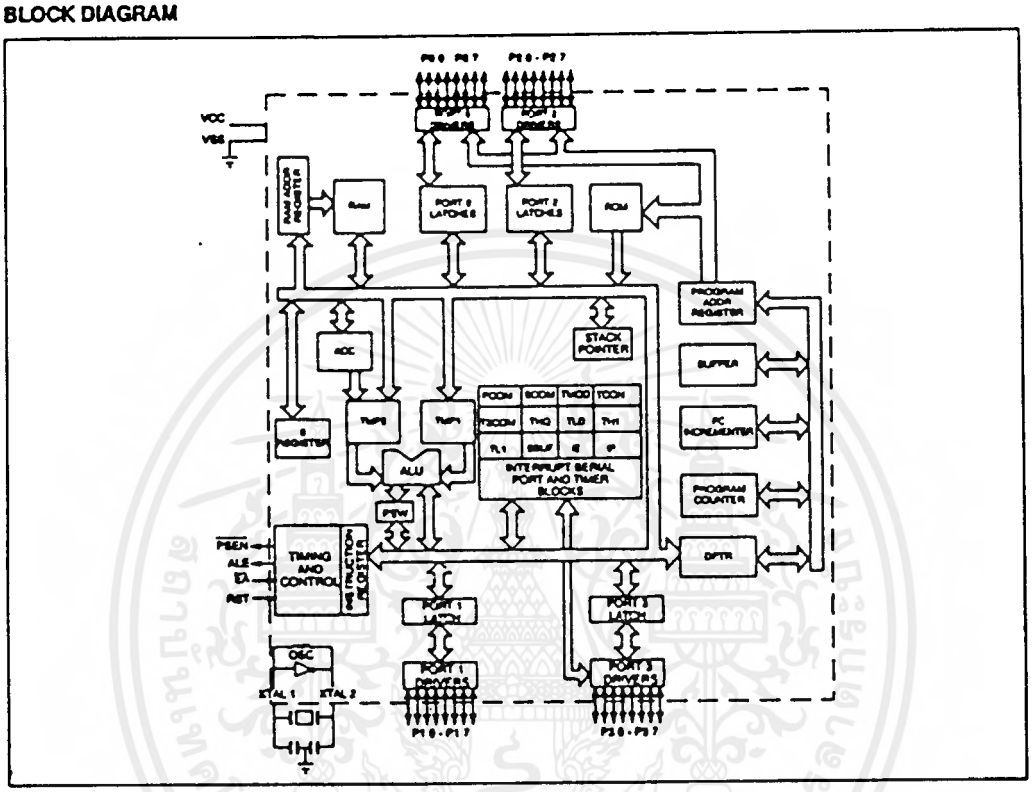


Figure 2.

IDLE AND POWER DOWN OPERATION

Figure 3 shows the internal Idle and Power Down clock configuration. As illustrated, Power Down operation stops the oscillator. Idle mode operation allows the interrupt, serial port, and timer blocks to continue to function while the clock to the CPU is gated off.

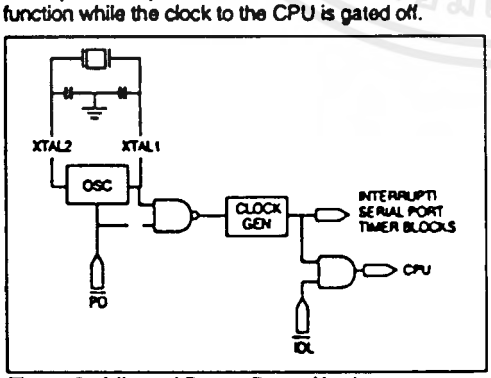


Figure 3 : Idle and Power Down Hardware.

PCON : Power Control Register

(MSB)							(LSB)
SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1, the baud rate is doubled when the serial port is being used in either modes 1, 2 or 3.
-	PCON.6	(Reserved)
-	PCON.5	(Reserved)
-	PCON.4	(Reserved)
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

These special modes are activated by software via the Special Function Register, its hardware address is 87H. PCON is not bit addressable.

If 1's are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).

IDLE MODE

The instruction that sets PCON.0 is the last instruction executed before the Idle mode is activated. Once in the Idle mode the CPU status is preserved in its entirety : the Stack Pointer, Program Counter, Program Status Word, Accumulator, RAM, and all other register maintain their data during Idle. Table 1 describes the status of the external pins during Idle mode.

There are two ways to terminate the Idle mode. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating Idle mode. The interrupt is serviced, and following RETI, the next instruction to be executed will be the one following the instruction that wrote 1 to PCON.0.

The flag bits GF0 and GF1 may be used to determine whether the interrupt was received during normal execution or during the Idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When Idle mode is terminated by an enabled interrupt, the service routine can examine the status of the flag bits.

The second way of terminating the Idle is with a hardware reset. Since the oscillator is still running, the hardware reset needs to be active for only 2 machine cycles (24 oscillator periods) to complete the reset operation.

POWER DOWN MODE

The instruction that sets PCON.1 is the last executed prior to entering power down. Once in power down, the oscillator is stopped. The contents of the onchip RAM and the Special Function Register is saved during power down mode. A hardware reset is the only way of exiting the power down mode. The hardware reset initiates the Special Function Register (see *Table 1*).

In the Power Down mode, V_{CC} may be lowered to minimize circuit power consumption. Care must be taken to ensure the voltage is not reduced until the power down mode is entered, and that the voltage is restored before the hardware reset is applied which frees the oscillator. Reset should not be released until the oscillator has restarted and stabilized.

Table 1 describes the status of the external pins while in the power down mode. It should be noted that if the power down mode is activated while in external program memory, the port data that is held in the Special Function Register P2 is restored to Port 2. If the data is a 1, the port pin is held high during the power down mode by the strong pullup, T1, shown in *Figure 4*.

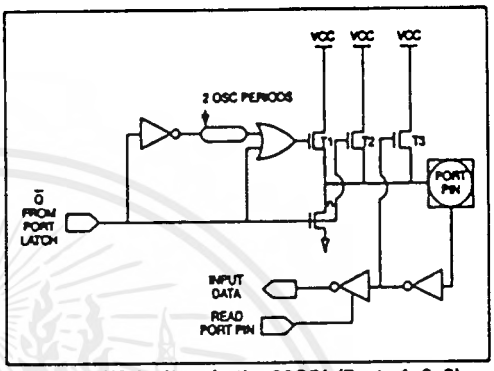


Figure 4 : I/O Buffers in the 80C51 (Ports 1, 2, 3).

STOP CLOCK MODE

Due to static design, the MHS 80C31/C51 clock speed can be reduced until 0 MHz without any data loss in memory or registers. This mode allows step by step utilization, and permits to reduce system power consumption by bringing the clock frequency down to any value. At 0 MHz, the power consumption is the same as in the Power Down Mode.

I/O PORTS

The I/O port drive of the 80C51 is similar to the 8051. The I/O buffers for Ports 1, 2 and 3 are implemented as shown in figure 4.

When the port latch contains a 0, all pFETS in *figure 4* are off while the nFET is turned on. When the port latch makes a 0-to-1 transition, the nFET turns off. The strong pullup pFET, T1, turns on for two oscillator periods, pulling the output high very rapidly. As the output line is drawn high, pFET T3 turns on through the inverter to supply the I_{OH} source current. This inverter and T3 form a latch which holds the 1 and is supported by T2.

When Port 2 is used as an address port, for access to external program of data memory, any address bit that contains a 1 will have his strong pullup turned on for the entire duration of the external memory access.

MODE	PROGRAM MEMORY	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Port Data	Port Data	Port Data	Port Data
Idle	External	1	1	Floating	Port Data	Address	Port Data
Power Down	Internal	0	0	Port Data	Port Data	Port Data	Port Data
Power Down	External	0	0	Floating	Port Data	Port Data	Port Data

Table 1 : Status of the external pins during Idle and Power Down modes.

When an I/O pin on Ports 1, 2 or 3 is used as an input, the user should be aware that the external circuit must sink current during the logical 1-to-0 transition. The maximum sink current is specified as ITL under the D.C. Specifications. When the input goes below approximately 2 V, T3 turns off to save ICC current. Note, when returning to a logical 1, T2 is the only internal pullup that is on. This will result in a slow rise time if the user's circuit does not force the input line high.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output respectively, of an inverting amplifier which is configured for use as an on-chip oscillator, as shown in figure 5. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected as shown in *figure 6*. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

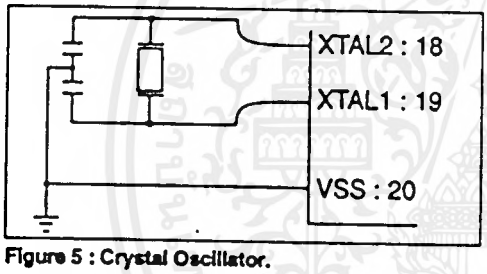


Figure 5 : Crystal Oscillator.

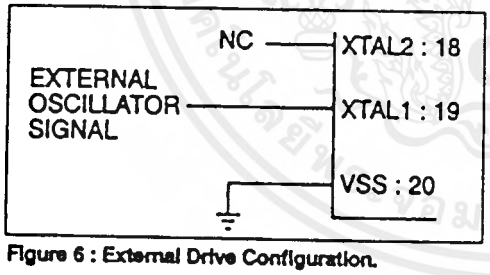


Figure 6 : External Drive Configuration.

80C51 WITH PROTECTED ROM

MHS provides a new member in the 80C51 Family named "80C51F" which permits full protection of the internal ROM contents.

With a non protected 80C51, it is very easy to read out the contents of the internal 4 K bytes of ROM.

Three methods exist, two of them are special test modes and the last one is by means of MOVC instructions.

- **Test mode "VER" :** Using this special test mode, the internal ROM contents are output on port P0 ; the address being applied on ports P2 (AD15...AD8) and P1 (AD7...AD0).
- **Test mode "TMB" :** With this second test mode, the contents of the 80C51 internal bus is presented on port P1 during the PH2 clock phases.
- **Using MOVC instructions :** If EA = 0, and following a reset, the 80C51 fetches its instructions from external program memory. It is then possible to write a small program whose purpose is to dump the internal ROM contents by means of MOVC A, @A + DPTR and MOVC A, @A + PC instructions.

80C51F with program protection features

This version adds ROM protection features in some strategic points of the 80C51F in order to eliminate the possibility of reading the ROM contents (once the protection has been programmed) by one if the three forementioned methods (VER and TMB test modes, or MOVC instructions).

Nevertheless the customer must note the following :

- Once the protection has been programmed, the 80C51F program always starts at address 0 in the internal ROM.
- The application program must be self contained in the internal 4 K of ROM, otherwise it would be possible to trap the program counter address in the external PROM/EPROM (beyond 4 K) and then to dump the internal ROM contents by means of a patch using MOVC instructions.

Thus, if an extra EPROM is necessary, it is advised to ensure that it will contain only constants or tables.

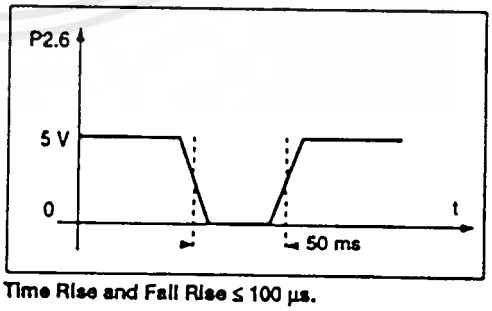
Test of the on-chip program memory

- **Before protection is activated :** The 80C51F can be tested as any normal 80C51 (using test equipment or any other methods).
- **After protection is activated :** It is then no longer possible to dump the internal ROM contents.

How to program the protection mechanism

- To burn correctly the fuse a specific configuration of inputs must be settled as below :
  - RST = ALE = 1
  - P2.7 = 1

Furthermore PSEN signal must be tied at + 9 V ± 5 % level voltage and a pulse must be applied on P2.6 input Port. The timing on P2.6 is shown below :



Time Rise and Fall Rise ≤ 100 µs.

- The electrical schematic shows a typical application to deliver P2.6 signal.

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Ambiant Temperature Under Bias :
C = commercial ..... : 0°C to 70°C
I = industrial .... - 40°C to +85°C
Storage Temperature..... .. - 65°C to + 150°C
Voltage on Vcc to Vss.... . . . . - 0.5 V to + 7 V
Voltage on Any Pin to Vss .. - 0.5 V to Vcc + 0.5 V
Power Dissipation ..... 1 W**

** This value is based on the maximum allowable die temperature and the thermal resistance of the package

* Notice

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

DC CHARACTERISTICS

T_A = 0 to + 70°C ; Vcc = 5 V ± 20 % ; Vss = 0 V ; F = 0 to 16 MHz.
Vcc = 5 V ± 10 % ; Vss = 0 V ; F = 16 to 20 MHz
T_A = - 40 to 85°C ; Vcc = 5 V ± 10 % ; Vss = 0 V ; F = 0 to 16 MHz.

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
VIL	Input Low Voltage	- 0.5 V	0.2 VCC - 0.1	V	
VIH	Input High Voltage (Except XTAL and RST)	0.2 VCC + 0.9	VCC + 0.5	V	
VIH1	Input High Voltage (RST and XTAL1)	0.7 VCC	VCC + 0.5	V	
VOL	Output Low Voltage (Ports 1, 2, and 3)		0.3 0.45 1.0	V V V	IOL = 100 µA IOL = 1.6 mA (note 3) IOL = 3.5 mA
VOL1	Output Low Voltage (Port 0, ALE, PSEN)		0.3 0.45 1.0	V V V	IOL = 200 µA IOL = 3.2 mA (note 3) IOL = 7.0 mA
VOH	Output High Voltage Ports 1, 2, 3	Vcc - 0.3 Vcc - 0.7 Vcc - 1.5		V V V	IOH = - 10 µA IOH = - 30 µA IOH = - 60 µA VCC = 5 V ± 10 %
VOH1	Output High Voltage (Port 0, ALE, PSEN)	Vcc - 0.3 Vcc - 0.7 Vcc - 1,5		V V V	IOH = - 200 µA IOH = - 3.2 mA IOH = - 7.0 mA VCC = 5 V ± 10 %
IIL	Logical 0 Input Current (Ports 1, 2, 3)		C - 50 I - 60	µA	Vin = 0.45 V
ILI	Input Leakage Current (Port 0, EA)		± 10	µA	0.45 < Vin < VCC
ITL	Logical 1 to 0 Transition Current (Ports 1, 2, 3)		- 650	µA	Vin = 2.0 V
IPD	Power Down Current		50	µA	VCC = 2.0 V to 6 V (note 2)
RRST	RST Pulldown Resistor	50	150	kΩ	
CIO	Capacitance of I/O Buffer		10	pF	f_c = 1 MHz, T_A = 25°C
ICC	Power Supply Current Active Mode 12 MHz 16 MHz 20 MHz Idle Mode 12 MHz 16 MHz 20 MHz		 20 26 32 5 6 8	 mA mA mA mA mA mA	(notes 1, 2)

ABSOLUTE MAXIMUM RATINGS*

Ambiant Temperature Under Bias :
A = Automotive ..... – 40°C to + 125°C
Storage Temperature ..... – 65°C to + 150°C
Voltage on Any Pin to Vss ..... – 0.5 V to VCC + 0.5 V
Voltage on Vcc to Vss ..... – 0.5 V to 6.5 V
Power Dissipation ..... 1 W

* Notice

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

DC CHARACTERISTICS

T_A = – 40° to 125°C ; Vcc = 5 V ± 10 % ; Vss = 0 V ; F = 0 to 12 MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
VIL	Input Low Voltage	– 0.5 V	0.2 VCC – 0.1	V	
VIH	Input High Voltage (Except XTAL1, RST)	0.2 VCC + 0.9	VCC + 0.5	V	
VIH1	Input High Voltage (XTAL1, RST)	0.7 VCC	VCC + 0.5	V	
VOL	Output Low Voltage (Ports 1, 2, and 3)		0.3 0.45 1.0	V V V	IOL = 100 µA IOL = 1.6 mA (note 3) IOL = 3.5 mA
VOL1	Output Low Voltage (Port 0, ALE, PSEN)		0.3 0.45 1.0	V	IOL = 200 µA IOL = 3.2 mA (note 3) IOL = 7.0 mA
VOH	Output High Voltage (Ports 1, 2, 3)	Vcc – 0.3		V	IOH = – 10 µA
		Vcc – 0.7		V	IOH = – 30 µA
		Vcc – 1.5		V	IOH = – 60 µA
VOH1	Output High Voltage (Port 0 in External Bus Mode, ALE, PSEN)	Vcc – 0.3		V	IOH = – 200 µA
		Vcc – 0.7		V	IOH = – 3.2 mA
		Vcc – 1,5		V	IOH = – 7.0 mA
IIL	Logical 0 Input Current Ports 1, 2, 3		– 75	µA	Vin = 0.45 V
ITL	Logical 1 to 0 Transition Current (Ports 1, 2, 3)		– 750	µA	Vin = 2.0 V
ILI	Input Leakage Current (Port 0, EA)		± 10	µA	0.45 < Vin < VCC
RRST	RST Pulldown Resistor	50	150	kΩ	
CIO	Pin Capacitance		10	pF	Test Freq = 1 MHz, T_A = 25°C
IPD	Power Down Current		75	µA	VCC = 2 V to 5.5 V (note 2)
ICC	Power supply current Active mode 12 MHz Idle mode 12 MHz		 21 7	 mA mA	 VCC = 5.5 V (notes 1, 2) VCC = 5.5 V (notes 1, 2)

ABSOLUTE MAXIMUM RATINGS*

Ambiant Temperature Under Bias :
M =Military ..... – 55°C to + 125°C
Storage Temperature ..... – 65°C to + 150°C
Voltage on Any Pin to V_{SS} ..... – 0.5 V to VCC + 0.5 V
Voltage on V_{CC} to V_{SS} ..... – 0.5 V to 6.5 V
Power Dissipation ..... 1 W

*** Notice**

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

DC CHARACTERISTICS

T_A = – 55° to 125°C ; VSS = 0 V ; VCC = 5 V ± 10 % ; F = 0 to 12 MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
VIL	Input Low Voltage	– 0.5 V	0.2 VCC – 0.1	V	
VIH	Input High Voltage (Except XTAL1, RST)	0.2 VCC + 0.9	VCC + 0.5	V	
VIH1	Input High Voltage (XTAL1, RST)	0.7 VCC	VCC + 0.5	V	
VOL	Output Low Voltage (Ports 1, 2, 3)		0.45	V	IOL = 1.6 mA (note 3)
VOL1	Output Low Voltage (Port 0, ALE, PSEN)		0.45	V	IOL = 3.2 mA (note 3)
VOH	Output High Voltage (Ports 1, 2, 3)	2.4		V	IOH = – 60 µA VCC = 5 V ± 10 %
		0.75 VCC		V	IOH = – 25 µA
		0.9 VCC		V	IOH = – 10 µA
VOH1	Output High Voltage (Port 0 in External Bus Mode, ALE, PEN)	2.4		V	IOH = – 800 µA VCC = 5 V ± 10 %
		0.75 VCC		V	IOH = – 300 µA
		0.9 VCC		V	IOH = – 80 µA
IIL	Logical 0 Input Current Ports 1, 2, 3		– 75	µA	Vin = 0.45 V
ITL	Logical 1 to 0 Transition Current (Ports 1, 2, 3)		– 750	µA	Vin = 2 V
ILI	Input Leakage Current (Port 0, EA)		± 10	µA	0.45 < Vin < VCC
RRST	RST Pulldown Resistor	50	150	kΩ	
CIO	Pin Capacitance		10	pF	Test Freq = 1 MHz, T_A = 25°C
IPD	Power Down Current		75	µA	VCC = 2 V to 5.5 V (note 2)
ICC	Power supply current Active mode 12 MHz Idle mode 12 MHz		 21 7	 mA mA	 VCC = 5.5 V (notes 1, 2) VCC = 5.5 V (notes 1, 2)

ABSOLUTE MAXIMUM RATINGS*

Ambiant Temperature Under Bias :
C = commercial ..... 0°C to 70°C
I = industrial ..... – 40°C to +85°C
Storage Temperature ..... – 65°C to + 150°C
Voltage on V_{CC} to V_{SS} ..... – 0.5 V to + 7 V
Voltage on Any Pin to V_{SS} – 0.5 V to V_{CC} + 0.5 V
Power Dissipation ..... 1 W**
** This value is based on the maximum allowable die temperature and the thermal resistance of the package

* Notice

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

DC CHARACTERISTICS

T_A = – 40° to 85°C ; V_{CC} = 2.7 V to 6 V ; VSS = 0 V ; F = 0 to 6 MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
VIL	Input Low Voltage	– 0.5 V	0.2 V_{CC} – 0.1	V	
VIH	Input High Voltage (Except XTALs and RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	
VIH1	Input High Voltage to RST for Reset	0.7 V_{CC}	V_{CC} + 0.5	V	
VIH2	Input High Voltage to XTAL1	0.7 V_{CC}	V_{CC} + 0.5	V	
VPD	Power Down Voltage to V_{CC} in PD Mode	2.0	6.0	V	
VOL	Output Low Voltage (Ports 1, 2, and 3)		0.45		IOL = 800 µA (note 3)
VOL1	Output Low Voltage (Port 0, ALE, PSEN)		0.45	V	IOL = 1.6 mA (note 3)
VOH	Output High Voltage Ports 1, 2, 3	0.9 V_{CC}		V	IOH = – 10 µA
VOH1	Output High Voltage (Port 0 in External Bus Mode), ALE, PEN	0.9 V_{CC}		V	IOH = – 80 µA
IIL	Logical 0 Input Current Ports 1, 2, 3		C – 50 I – 60	µA	Vin = 0.45 V
ILI	Input Leakage Current		± 10	µA	0.45 < Vin < VCC
ITL	Logical 1 to 0 Transition Current (Ports 1, 2, 3)		– 650	µA	Vin = 2.0 V
IPD	Power Down Current		50	µA	V_{CC} = 2.0 V to 6 V (note 2)
RRST	RST Pulldown Resistor	50	150	kΩ	
CIO	Capacitance of I/O Buffer		10	pF	f_c = 1 MHz, T_A = 25°C

MAXIMUM Icc (mA)

	OPERATING (NOTE 3)			IDLE (NOTE 4)		
FREQ. VCC	2.7 V	5 V	6 V	2.7 V	5 V	6 V
1 MHz	0.8 mA	1.5 mA	1.8 mA	400 µA	800 µA	1 mA
6 MHz	4 mA	8 mA	10 mA	1.2 mA	3.5 mA	3.8 mA

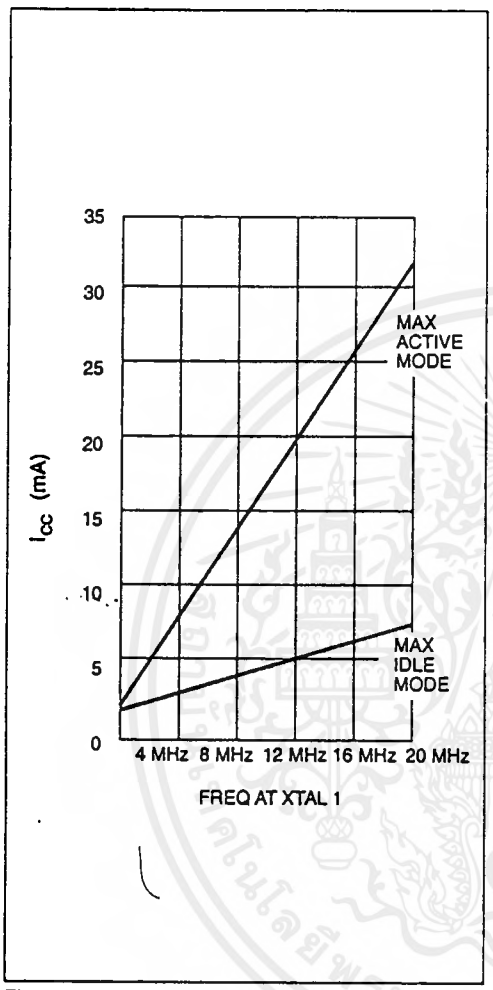


Figure 1 : ICC vs. Frequency. Valid only within frequency specifications of the device under test.

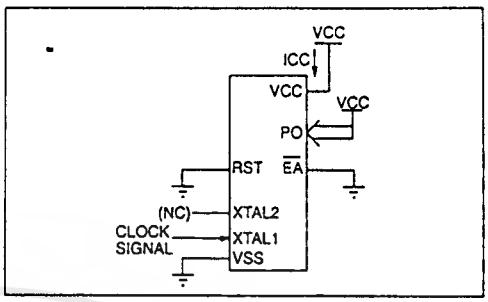


Figure 12 : ICC Test Condition, Idle Mode. All other pins are disconnected.

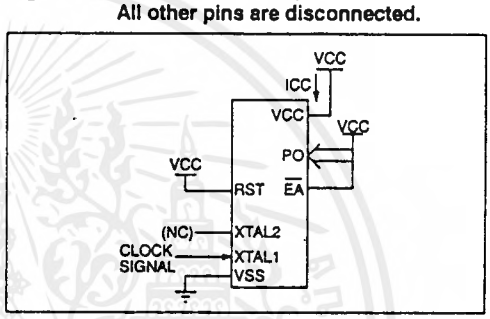


Figure 13 : ICC Test Condition, Active Mode. All other pins are disconnected.

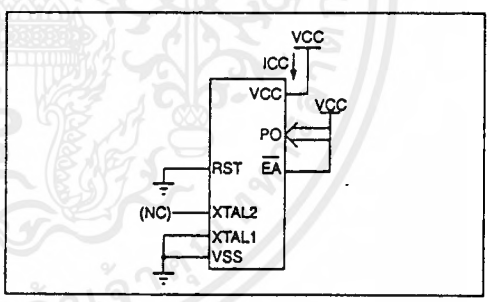


Figure 15 : ICC Test Condition, Power Down Mode. All other pins are disconnected.

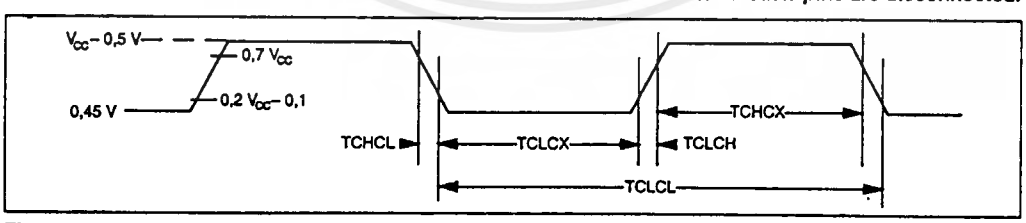


Figure 14 : Clock Signal Waveform for ICC Tests in Active and Idle Modes. TCLCH = TCHCL = 5 ns.

Note 1 : ICC max is given by :

Active Mode : ICCMAX = 1.47 x FREQ + 2.35

Idle Mode : ICCMAX = 0.33 x FREQ + 1.05

where FREQ is the external oscillator frequency in MHz. ICCMAX is given in mA. See figures 1 through 5 for ICC test conditions.

Note 2 : ICC is measured with all output pins disconnected ; XTAL1 driven with TCLCH, TCHCL = 5 ns, VIL = VSS + .5 V, VIH = VCC – .5 V ; XTAL2 N.C. ; EA = RST = Port 0 = VCC. ICC would be slightly higher if a crystal oscillator used.

Idle ICC is measured with all output pins disconnected ; XTAL1 driven with TCLCH, TCHCL = 5 ns, VIL = VSS + .5 V, VIH = VCC – .5 V ; XTAL2 N.C ; Port 0 = VCC ; EA = RST = VSS.

Power Down ICC is measured with all output pins disconnected ; EA = PORT 0 = VCC ; XTAL2 N.C. ; RST = VSS.

Note 3 : Capacitance loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the VOLs of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1 to 0 transitions during bus operations. In the worst cases (capacitive loading 100 pF), the noise pulse on the ALE line may exceed 0.45 V may exceed 0,45 V with maxi VOL peak 0.6 V. A Schmitt Trigger use is not necessary.

EXPLANATION OF THE AC SYMBOL

Each timing symbol has 5 characters. The first character is always a "T" (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for.

Example :

TAVLL = Time for Address Valid to ALE low.

TLLPL = Time for ALE low to $\overline{\text{PSEN}}$ low.

A : Address.
C : Clock.
D : Input data.
H : Logic level HIGH.
I : Instruction (program memory contents).
L : Logic level LOW, or ALE.
P : PSEN.
Q : Output data.
R : READ signal.
T : Time.
V : Valid.
W : WRITE signal.
X : No longer a valid logic level.
Z : Float.

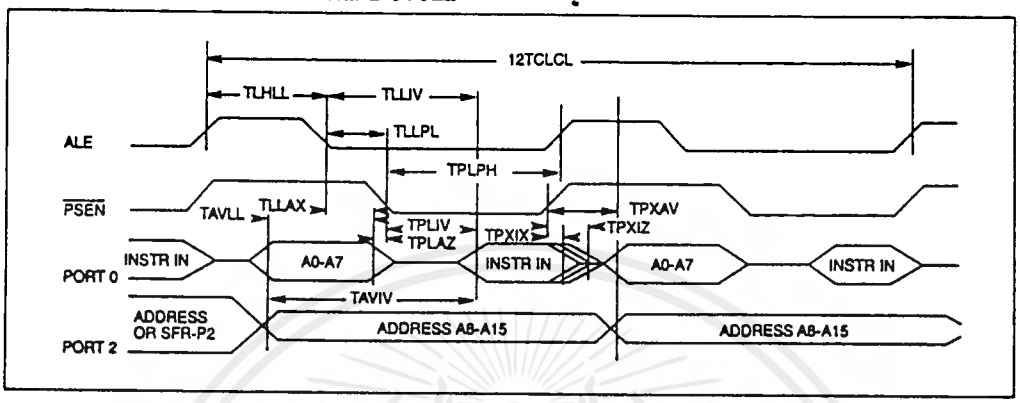
AC PARAMETERS

T_A = 0 to + 70 °C ; VSS = 0 V ; VCC = 5 V ± 20 % ; 0 to 16 MHz
T_A = 0 to + 70 °C ; VSS = 0 V ; VCC = 5 V ± 10 % ; 16 to 20 MHz
T_A = – 40 to + 85 °C ; VSS = 0 V ; VCC = 5 V ± 10 % ; 0 to 16 MHz
T_A = – 55 to + 125 °C ; VSS = 0 V ; VCC = 5 V ± 10 % ; 0 to 12 MHz
(Load Capacitance for PORT0, ALE and PSEN = 100 pf ; Load Capacitance for all other outputs = 80 pf.)

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

SYMBOL	PARAMETER	0 TO 12 MHz		16 MHz		20 MHz	
		MIN	MAX	MIN	MAX	MIN	MAX
TLHLL	ALE pulse width	2TCLCL-40		110		70	
TAVLL	Address Valid to ALE	TCLCL-40		30		25	
TLLAX	Address Hold After ALE	TCLCL-30		35		25	
TLLIV	ALE to Valid Instr In		4TCLCL-100		185		140
TLLPL	ALE to PSEN	TCLCL-30		45		30	
TPLPH	PSEN Pulse Width	3TCLCL-45		165		130	
TPLIV	PSEN to Valid Instr IN		3TCLCL-105		125		80
TPXIX	Input Instr Hold After PSEN	0		0		0	
TPXIZ	Input Instr Float After PSEN		TCLCL-25		22		10
TPXAV	PSEN to Address Valid	TCLCL-8		55		45	
TAVIV	Address to Valid Instr In		5TCLCL-105		230		180
TPLAZ	PSEN Low to Address Float		10		10		10

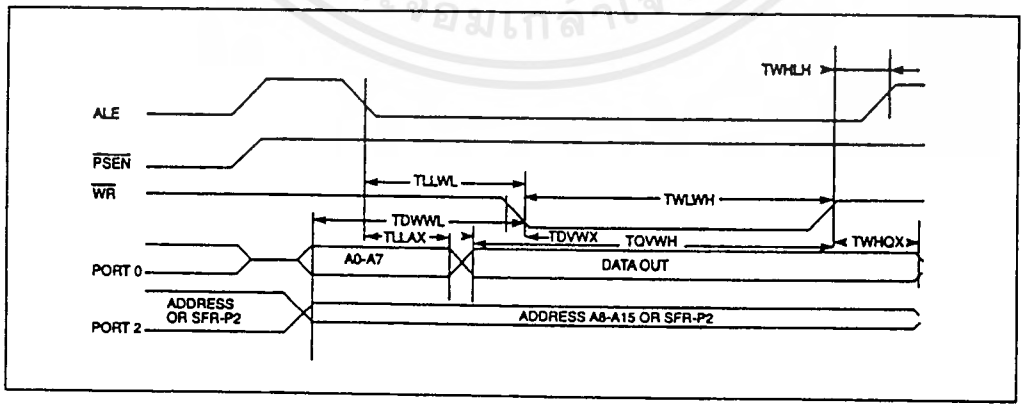
EXTERNAL PROGRAM MEMORY READ CYCLE



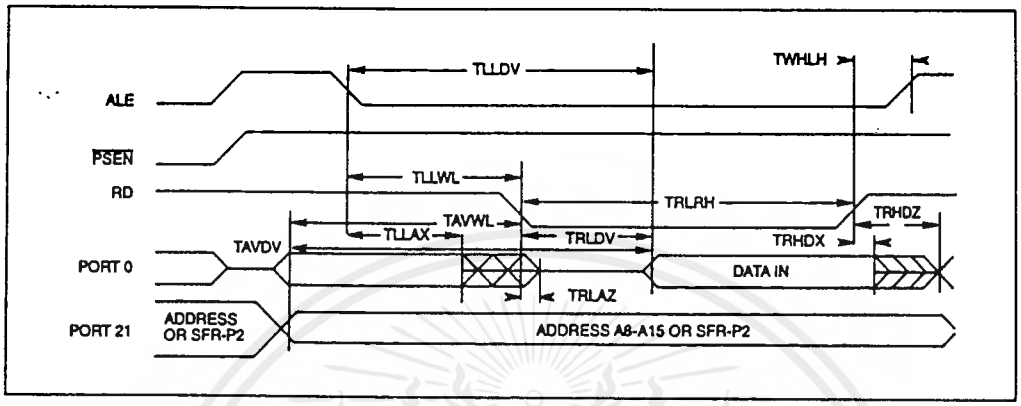
EXTERNAL DATA MEMORY CHARACTERISTICS

SYMBOL	PARAMETER	0 TO 12 MHz		16 MHz		20 MHz	
		MIN	MAX	MIN	MAX	MIN	MAX
TRLRH	RD Pulse Width	6TCLCL-100		340		260	
TWLWH	WR Pulse Width	6TCLCL-100		340		260	
TRLDV	RD to Valid Data In		5TCLCL-165		240		180
TRHDX	Data Hold After RD	0		0		0	
TRHDZ	Data Float After RD		2TCLCL-60		90		70
TLLDV	ALE to Valid Data In		8TCLCL-150		435		360
TAVDV	Address to Valid Data In		9TCLCL-165		480		400
TLLWL	ALE to WR or RD	3TCLCL-50	3TCLCL+50	150	250	125	185
TAVWL	Address to WR or RD	4TCLCL-130		180		170	
TQVWX	Data Valid to WR Transition	TCLCL-50		15		10	
TQVWH	Data Set-Up to WR High	7TCLCL-150		380		310	
TWHQX	Data Hold After WR	TCLCL-50		40		30	
TRLAZ	RD Low to Address Float		0		0		0
TWHLH	RD or WR High to ALE High	TCLCL-40	TCLCL+40	35	90	30	65

EXTERNAL DATA MEMORY WRITE CYCLE



EXTERNAL DATA MEMORY READ CYCLE

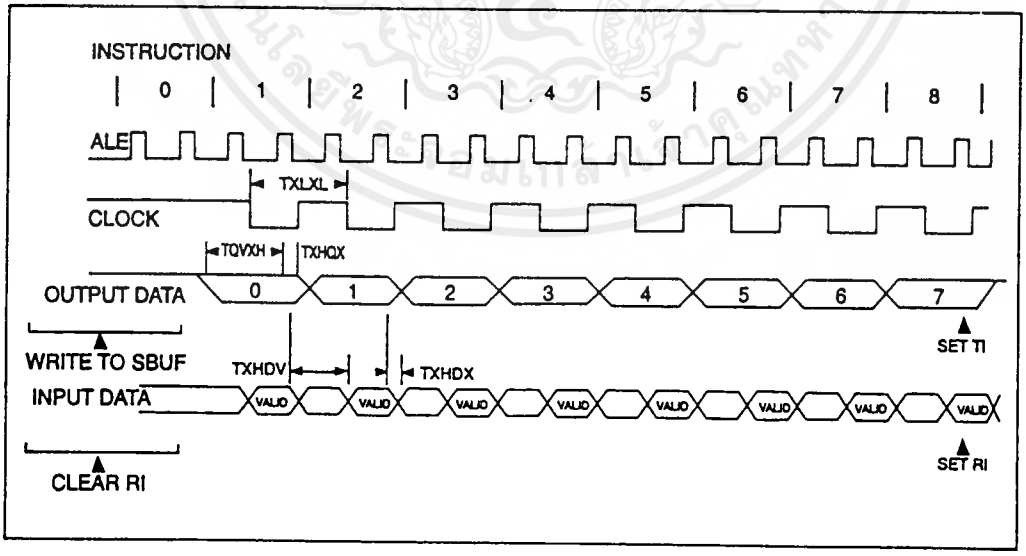


SERIAL PORT TIMING – SHIFT REGISTER MODE

T_A = 0 to + 70 °C ; VSS = 0 V ; VCC = 5 V ± 20 % ; 0 to 16 MHz
T_A = 0 to + 70 °C ; VSS = 0 V ; VCC = 5 V ± 10 % ; 16 to 20 MHz
T_A = – 40 to + 85 °C ; VSS = 0 V ; VCC = 5 V ± 20 % ; 0 to 16 MHz
T_A = – 55 to + 125 °C ; VSS = 0 V ; VCC = 5 V ± 10 % ; 0 to 12 MHz

SYMBOL	PARAMETER	0 TO 12 MHz		16 MHz		20 MHz	
		MIN	MAX	MIN	MAX	MIN	MAX
TXLXL	Serial port clock cycle time	12TCLCL		750		600	
TQVHX	Output data setup to clock rising edge	10TCLCL-133		563		450	
TXHQX	Output data hold after clock rising edge	2TCLCL-117		63		50	
TXHDX	Input data hold after clock rising edge	0		0		0	
TXHDV	Clock rising edge to input data valid		10TCLCL-133		563		450

SHIFT REGISTER TIMING WAVEFORMS

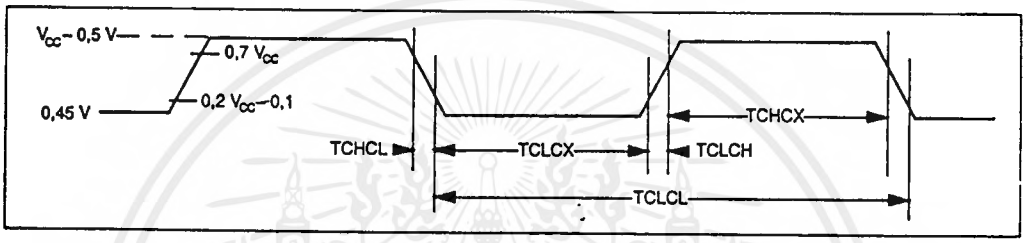


EXTERNAL CLOCK DRIVE CHARACTERISTICS (XTAL1)

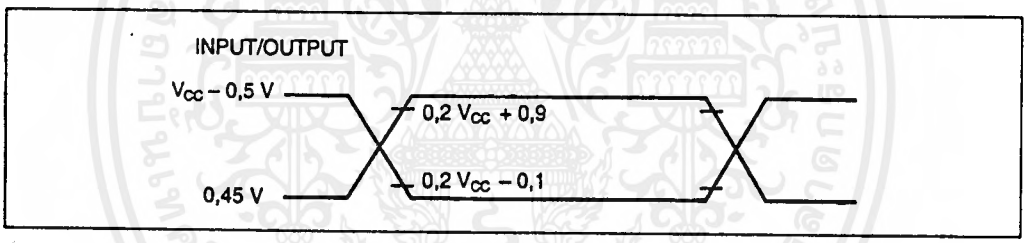
SYMBOL	PARAMETER	MIN	MAX	UNIT
TCLCL	Oscillator Period	50 (5)		ns
TCHCX	High Time	20 (5)		ns
TCLCX	Low Time	20 (5)		ns
TCLCH	Rise Time		20 (5)	ns
TCHCL	Fall Time		20 (5)	ns

(5) AT 20 MHz

EXTERNAL CLOCK DRIVE WAVEFORMS

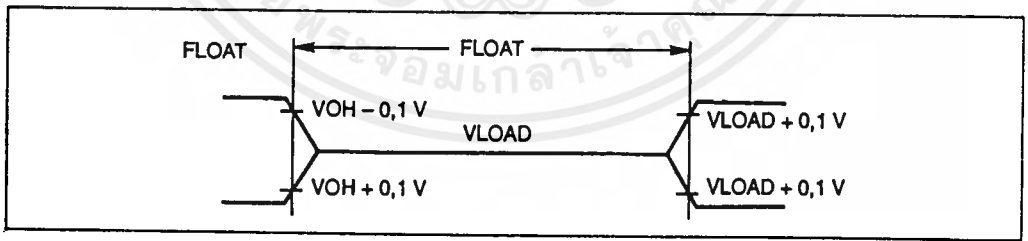


AC TESTING INPUT/OUTPUT WAVEFORMS



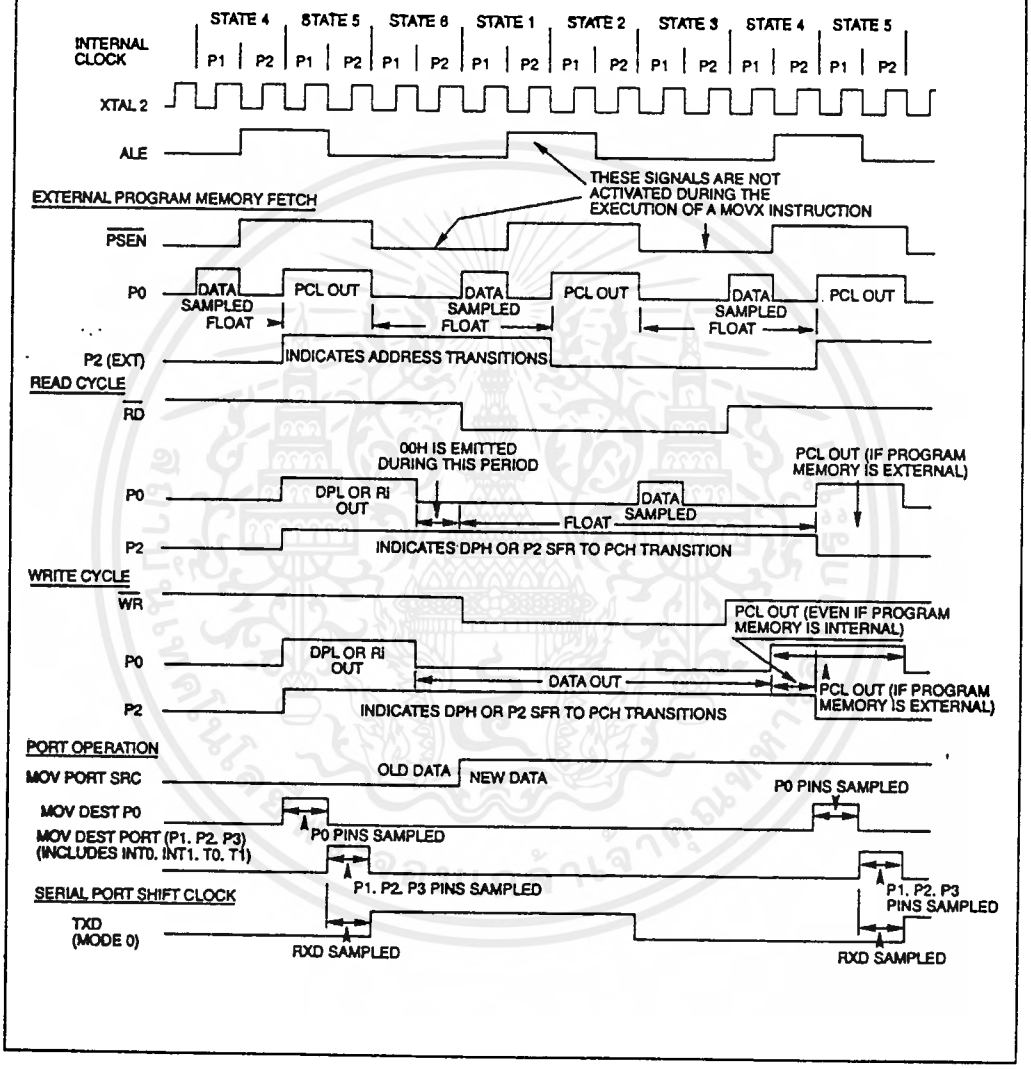
AC inputs during testing are driven at V_{CC} – 0.5 for a logic "1" and 0.45 V for a logic "0". Timing measurements are made at VIH min for a logic "1" and VIL max for a logic "0".

FLOAT WAVEFORMS



For timing purposes as port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV change from the loaded VOH/VOL level occurs. Iol/IoH ≥ ± 20 mA.

CLOCK WAVEFORMS



This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component. Typically though (T_A = 25°C fully loaded) RD and WR propagation delays are approximately 50 ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

INSTRUCTION OPCODES

MHS C51 INSTRUCTION SET DESCRIPTION

ARITHMETIC OPERATIONS MNEMONIC		DESCRIPTION	BYTE	CYC
ADD	A, Rn	Add registrer to Accumulator	1	1
ADD	A, direct	Add direct bytes to Accumulator	2	1
ADD	A, @Ri	Add indirect RAM to Accumulator	1	1
ADD	A, #data	Add immediate data ot Accumulator	2	1
ADDC	A, Rn	Add register to Accumulator with Carry	1	1
ADDC	A, direct	Add direct byte to A with Carry flag	2	1
ADDC	A, @Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A, #data	Add immediate data to A with Carry flag	2	1
SUBB	A, Rn	Subtract register from A with borrow	1	1
SUBB	A, direct	Subtract direct byte from A with Borrow	2	1
SUBB	A, @Ri	Subtract indirect RAM from A with Borrow	1	1
SUBB	A, data	Subtract immed. data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
MUL	AR	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1
LOGICAL OPERATIONS MNEMONIC		**DESTINATION**	**BYTE**	**CYC**
ANL	A, Rn	AND register to Accumulator	1	1
ANL	A, direct	AND direct byte to Accumulator	2	1
ANL	A, @Ri	AND indirect RAM to Accumulator	1	1
ANL	A, #data	AND immediate data to Accumulator	2	1
ANL	direct, A	AND Accumultaor to direct byte	2	1
ANL	direct, #data	AND immediate data to direct byte	3	2
ORL	A, Rn	OR register to Accumulator	1	1
ORL	A, direct	OR direct byte to Accumulator	2	1
ORL	A, @Ri	OR indirect RAM to Accumulator	1	1
ORL	A, #data	OR immediate data to Accumulator	2	1
ORL	direct A	OR Accumulator to direct byte	2	1
ORL	direct, #data	OR immediate data to direct byte	3	2
XRL	A, Rn	Exclusive-OR register to Accumulator	1	1
XRL	A, direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A, @Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A, #data	Exclusive-OR immediate data to A	2	1
XRL	direct, A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct, #data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Flight through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

ARITHMETIC TRANSFER

MNEMONIC		DESCRIPTION	BYTE	CYC
MOV	A, Rn	Move register to Accumulator	1	1
MOV	A, direct	Move direct byte to Accumulator	2	1
MOV	A, @Ri	Move indirect RAM to Accumulator	1	1
MOV	A, #data	Move immediate data to Accumulator	2	1
MOV	Rn, A	Move Accumulator to register	1	1
MOV	Rn, direct	Move direct byte to register	2	2
MOV	Rn, #data	Move immediate data to register	2	1
MOV	direct, A	Move Accumulator to direct byte	2	1
MOV	direct, Rn	Move register to direct byte	2	2
MOV	direct, direct	Move direct byte to direct	3	2
MOV	direct, @Ri	Move indirect RAM to direct byte	2	2
MOV	direct, #data	Move immediate data to direct byte	3	2
MOV	@Ri, A	Move Accumulator to indirect RAM	1	1
MOV	@Ri, direct	Move direct byte to indirect RAM	2	2
MOV	@Ri, #data	Move immediate data to indirect RAM	2	1
MOV	DPTR, #data 16	Load Data Pointer with a 16-bit constant	3	2
MOVC	A, @A + DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A, @A + PC	Move Code byte relative to PC to A	1	2
MOVX	A, @Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A, @DPTR	Move external RAM (16-bit addr) to A	1	2
MOVX	@Ri, A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR, A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A, Rn	Exchange register with Accumulator	1	1
XCH	A, direct	Exchange direct byte with Accumulator	2	1
XCH	A, @Ri	Exchange indirect RAM with A	1	1
XCHD	A, @Ri	Exchange low-order nibble ind RAM with A	1	1

BOOLEAN VARIABLE MANIPULATION

MNEMONIC		DESCRIPTION	BYTE	CYC
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C, bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit, C	Move Carry flag to direct bit	2	2

PROGRAM AND MACHINE CONTROL

MNEMONIC		DESCRIPTION	BYTE	CYC
ACALL	addr 11	Absolute subroutine Call	2	2
LCALL	addr 16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr 11	Absolute Jump	2	2
LJMP	addr 16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2

PROGRAM AND MACHINE CONTROL (cont.) MNEMONIC		DESCRIPTION	BYTE	CYC
JB	bit, rel	Jump if direct Bit set	3	2
JNB	bit, rel	Jump if direct Bit Not sef	3	2
JBC	bit, rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A, direct, rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A, #data, ref	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn, #data, rel	Comp. immed. to reg & Jump if Not Equal	3	2
CJNE	@Ri, #data, rel	Comp. immed. to ind. & jump if Not Equal	3	2
DJNZ	Rn, rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct, rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes :

Rn – Working register R0-R7
direct – 128 internal RAM locations, any I/O port, control or status register
@Ri – Indirect internal RAM location addressed by register R0 or R1
#data – 8-bit constant included in instruction
#data 16 – 16-bit constant included as bytes 2 & 3 of instruction
bit – 128 software flags, any I/O pin, control or status bit

Notes on program addressing modes :

addr 16 – Destination address for LCALL & LJMP may be anywhere within the 64-k program memory address space
Addr 11 – Destination address for ACALL & AJMP will be within the same 2-k page of program memory as the first byte of the following instruction
rel – SJMP and all conditional jumps include an 8-bit offset byte. Range is + 127 – 128 bytes relative to the first byte of the following instruction.

INSTRUCTION OPCODES IN HEXADECIMAL ORDER

HEX CODE	NUMB. OF BYTES	MNEM.	OPERANDS
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALLR	code addr
13	1	RC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, data
25	2	ADD	A, data addr
26	1	ADD	A, @R0
27	1	ADD	A, @R1
28	1	ADD	A, R0
29	1	ADD	A, R1
2A	1	ADD	A, R2
2B	1	ADD	A, R3
2C	1	ADD	A, R4
2D	1	ADD	A, R5
2E	1	ADD	A, R6
2F	1	ADD	A, R7
30	3	JNB	bit, addr, code
31	2	ACALL	addr
32	1	RETI	code addr

HEX CODE	NUMB. OF BYTES	MNEM.	OPERANDS
33	1	RLC	A
34	2	ADDC	A, #data
35	2	ADDC	A, data addr
36	1	ADDC	A, @RD
37	1	ADDC	A, @R1
38	1	ADDC	A, R0
39	1	ADDC	A, R1
3A	1	ADDC	A, R2
3B	1	ADDC	A, R3
3C	1	ADDC	A, R4
3D	1	ADDC	A, R5
3E	1	ADDC	A, R6
3F	1	ADDC	A, R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr A
43	3	ORL	data addr, #data
44	2	ORL	A, #data
45	2	ORL	A, data addr
46	1	ORL	A, @R0
47	1	ORL	A, @R1
48	1	ORL	A, R0
49	1	ORL	A, R1
4A	1	ORL	A, R2
4B	1	ORL	A, R3
4C	1	ORL	A, R4
4D	1	ORL	A, R5
4E	1	ORL	A, R6
4F	1	ORL	A, R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr, A
53	3	ANL	data addr, #data
54	2	ANL	A, #data
55	2	ANL	A, data addr
56	1	ANL	A, @R0
57	1	ANL	A, @R1
58	1	ANL	A, R0
59	1	ANL	A, R1
5A	1	ANL	A, R2
5B	1	ANL	A, R3
5C	1	ANL	A, R4
5D	1	ANL	A, R5
5E	1	ANL	A, R6
5F	1	ANL	A, R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr A
63	3	XRL	data addr, #data
64	2	XRL	A, #data
65	2	XRL	A, data addr

HEX CODE	NUMB. OF BYTES	MNEM.	OPERANDS
66	1	XRL	A, @R0
67	1	XRL	A, @R1
68	1	XRL	A, R0
69	1	XRL	A, R1
6A	1	XRL	A, R2
6B	1	XRL	A, R3
6C	1	XRL	A, R4
6D	1	XRL	A, R5
6E	1	XRL	A, R6
6F	1	XRL	A, R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C, bit addr
73	1	JMP	@A + DPTR
74	2	MOV	A, #data
75	3	MOV	data addr, #data
76	2	MOV	@R0, #data
77	2	MOV	@R1, #data
78	2	MOV	R0, #data
79	2	MOV	R1, #data
7A	2	MOV	R2, #data
7B	2	MOV	R3, #data
7C	2	MOV	R4, #data
7D	2	MOV	R5, #data
7E	2	MOV	R6, #data
7F	2	MOV	R7, #data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C, bit addr
83	1	MOVC	A, @A + PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr, @R0
87	2	MOV	data addr, @R1
88	2	MOV	data addr, R0
89	2	MOV	data addr, R1
8A	2	MOV	data addr, R2
8B	2	MOV	data addr, R3
8C	2	MOV	data addr, R4
8D	2	MOV	data addr, R5
8E	2	MOV	data addr, R6
8F	2	MOV	data addr, R7
90	3	MOV	DPTR, #data
91	2	ACALL	code adddr
92	2	MOV	bit addr, C
93	1	MOVC	A, @A + DPTR
94	2	SUBB	A, #data
95	2	SUBB	A, data addr
96	1	SUBB	A, @R0
97	1	SUBB	A, @R1
98	1	SUBB	A, R0

HEX CODE	NUMB. OF BYTES	MNEM.	OPERANDS
99	1	SUBB	A, R1
9A	1	SUBB	A, R2
9B	1	SUBB	A, R3
9C	1	SUBB	A, R4
9D	1	SUBB	A, R5
9E	1	SUBB	A, R6
9F	1	SUBB	A, R7
A0	2	ORL	C, bit addr
A1	2	AJMP	code addr
A2	2	MOV	C, bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0, data addr
A7	2	MOV	@R1, data addr
A8	2	MOV	R0, data addr
A9	2	MOV	R1, data addr
AA	2	MOV	R2, data addr
AB	2	MOV	R3, data addr
AC	2	MOV	R4, data addr
AD	2	MOV	R5, data addr
AE	2	MOV	R6, data addr
AF	2	MOV	R7, data addr
B0	2	ANL	C, bit addr
B1	2	ACALL	code addr
B2	2	CPL	Bit addr
B3	1	CPL	C
B4	3	CJNE	A, #data, code addr
B5	3	CJNE	A, data addr, code addr
B6	3	CJNE	@R0, #data, code addr
B7	3	CJNE	@R1, #data, code addr
B8	3	CJNE	R0, #data, code addr
B9	3	CJNE	R1, #data, code addr
BA	3	CJNE	R2, #data, code addr
BB	3	CJNE	R3, #data, code addr
BC	3	CJNE	R5, #data, code addr
BD	3	CJNE	R4, #data, code addr
BE	3	CJNE	R6, #data, code addr
BF	3	CJNE	R7, #data, code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A, data addr
C6	1	XCH	A, @R0
C7	1	XCH	A, @R1
C8	1	XCH	A, R0
C9	1	XCH	A, R1
CA	1	XCH	A, R2
CB	1	XCH	A, R3

ISO²-CMOS MT8870D/MT8870D-1
Integrated DTMF Receiver

ISSUE 3 May 1995

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
MT8870DC/DC-1	18 Pin Ceramic DIP
MT8870DS/DS-1	18 Pin SOIC
MT8870DN/DN-1	20 Pin SSOP
MT8870DT/DT-1	20 Pin TSSOP

-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.

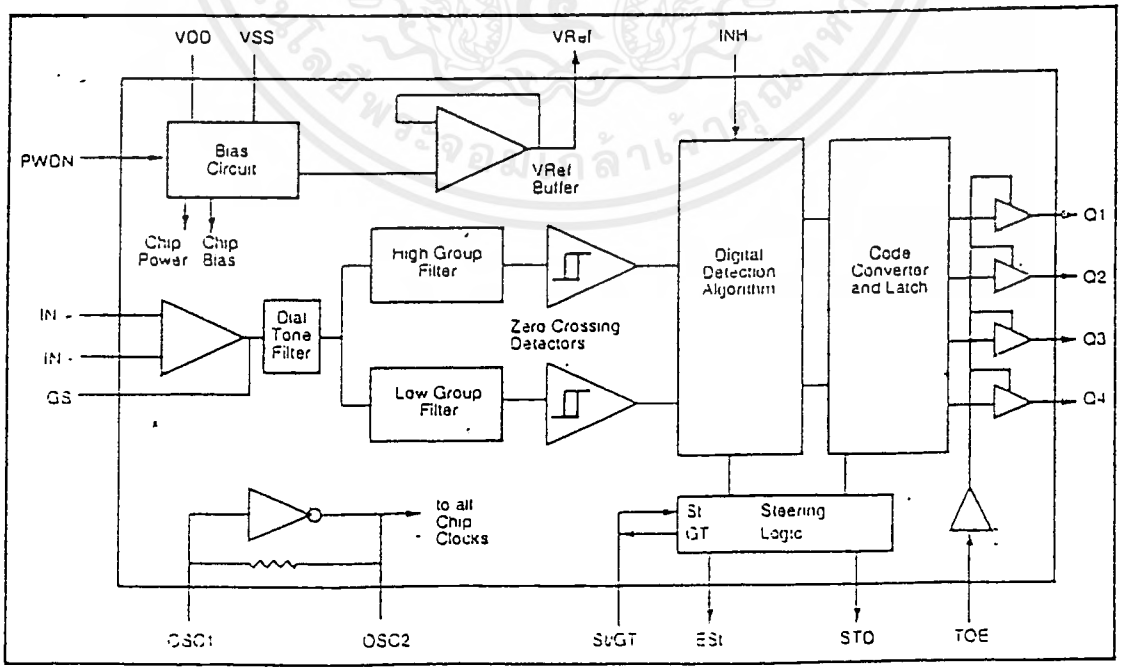


Figure 1 - Functional Block Diagram

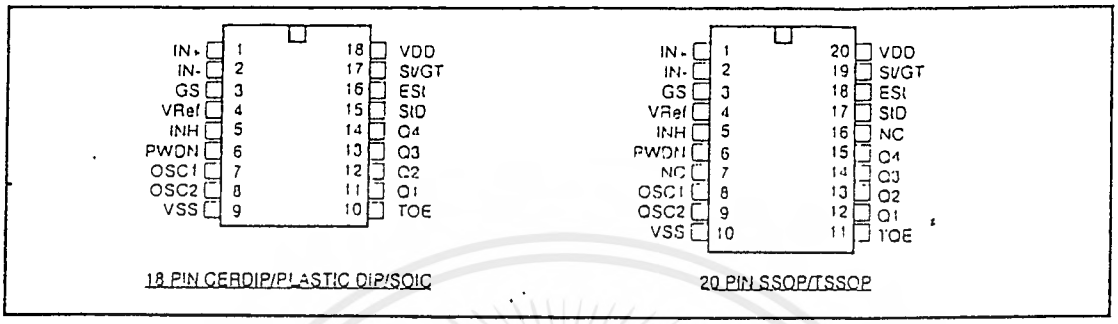


Figure 2 - Pin Connections

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.

Functional Description

The MT8870D/MT8870D-1 monolithic DTMF receiver offers small size, low power consumption and high performance. Its architecture consists of a bandsplit filter section, which separates the high and low group tones, followed by a digital counting section which verifies the frequency and duration of the received tones before passing the corresponding code to the output bus.

Filter Section

Separation of the low-group and high group tones is achieved by applying the DTMF signal to the inputs of two sixth-order switched capacitor bandpass filters, the bandwidths of which correspond to the low and high group frequencies. The filter section also incorporates notches at 350 and 440 Hz for exceptional dial tone rejection (see Figure 3). Each filter output is followed by a single order switched capacitor filter section which smooths the signals prior to limiting. Limiting is performed by high-gain comparators which are provided with hysteresis to prevent detection of unwanted low-level signals. The outputs of the comparators provide full rail logic swings at the frequencies of the incoming DTMF signals.

Decoder Section

Following the filter section is a decoder employing digital counting techniques to determine the frequencies of the incoming tones and to verify that they correspond to standard DTMF frequencies. A complex averaging algorithm protects against tone simulation by extraneous signals such as voice while

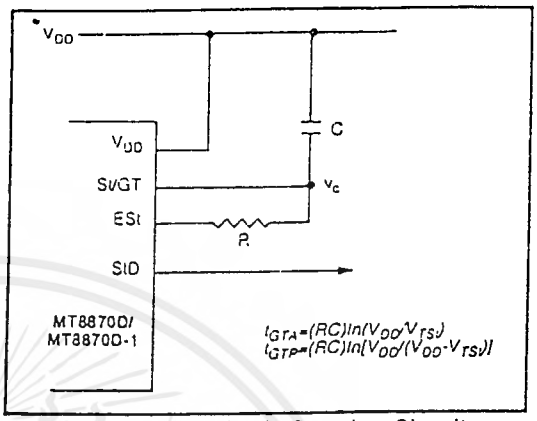


Figure 4 - Basic Steering Circuit

providing tolerance to small frequency deviations and variations. This averaging algorithm has been developed to ensure an optimum combination of immunity to talk-off and tolerance to the presence of interfering frequencies (third tones) and noise. When the detector recognizes the presence of two valid tones (this is referred to as the "signal condition" in some industry specifications) the "Early Steering" (ESt) output will go to an active state. Any subsequent loss of signal condition will cause ESt to assume an inactive state (see "Steering Circuit").

Steering Circuit

Before registration of a decoded tone pair, the receiver checks for a valid signal duration (referred to as character recognition condition). This check is performed by an external RC time constant driven by ESt. A logic high on ESt causes v_c (see Figure 4) to rise as the capacitor discharges. Provided signal

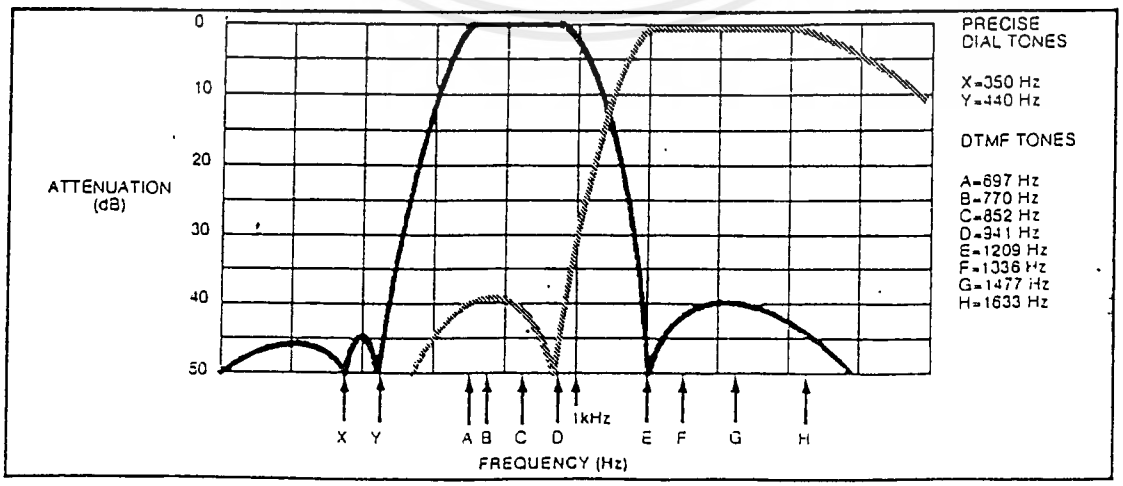


Figure 3 - Filter Response

condition is maintained (ESt remains high) for the validation period (t_{GTP}). v_c reaches the threshold (V_{TSt}) of the steering logic to register the tone pair, latching its corresponding 4-bit code (see Table 1) into the output latch. At this point the GT output is activated and drives v_c to V_{DD}. GT continues to drive high as long as ESt remains high. Finally, after a short delay to allow the output latch to settle, the delayed steering output flag (StD) goes high, signalling that a received tone pair has been registered. The contents of the output latch are made available on the 4-bit output bus by raising the three state control input (TOE) to a logic high. The steering circuit works in reverse to validate the interdigit pause between signals. Thus, as well as rejecting signals too short to be considered valid, the receiver will tolerate signal interruptions (dropout) too short to be considered a valid pause. This facility, together with the capability of selecting the steering time constants externally, allows the designer to tailor performance to meet a wide variety of system requirements.

Guard Time Adjustment

In many situations not requiring selection of tone duration and interdigital pause, the simple steering circuit shown in Figure 4 is applicable. Component values are chosen according to the formula:

$$t_{REC}=t_{DP}+t_{GTP}$$
$$t_{ID}=t_{DA}+t_{GTA}$$

The value of t_{DP} is a device parameter (see Figure 11) and t_{REC} is the minimum signal duration to be recognized by the receiver. A value for C of 0.1 µF is recommended for most applications, leaving R to be selected by the designer.

Digit	TOE	INH	ESt	Q_4	Q_3	Q_2	Q_1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	L	H	1	1	0	1
B	H	L	H	1	1	1	0
C	H	L	H	1	1	1	1
D	H	L	H	0	0	0	0
A	H	H	L	undetected, the output code will remain the same as the previous detected code			
B	H	H	L				
C	H	H	L				
D	H	H	L				

Table 1. Functional Decode Table

L=LOGIC LOW, H=LOGIC HIGH, Z=HIGH IMPEDANCE
X = DON'T CARE

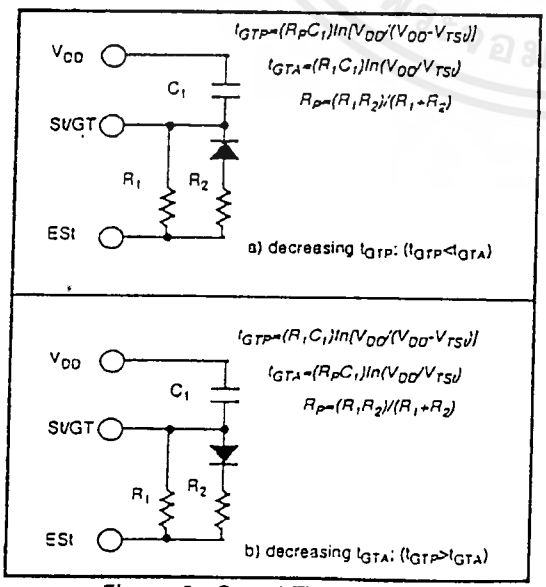


Figure 5 - Guard Time Adjustment

Different steering arrangements may be used to select independently the guard times for tone present (t_{GTP}) and tone absent (t_{GTA}). This may be necessary to meet system specifications which place both accept and reject limits on both tone duration and interdigital pause. Guard time adjustment also allows the designer to tailor system parameters such as talk off and noise immunity. Increasing t_{REC} improves talk-off performance since it reduces the probability that tones simulated by speech will maintain signal condition long enough to be registered. Alternatively, a relatively short t_{REC} with a long t_{DO} would be appropriate for extremely noisy environments where fast acquisition time and immunity to tone drop-outs are required. Design information for guard time adjustment is shown in Figure 5.

Power-down and Inhibit Mode

A logic high applied to pin 6 (PWDN) will power down the device to minimize the power consumption in a standby mode. It stops the oscillator and the functions of the filters.

Inhibit mode is enabled by a logic high input to the pin 5 (INH). It inhibits the detection of tones representing characters A, B, C, and D. The output code will remain the same as the previous detected code (see Table 1).

Differential Input Configuration

The input arrangement of the MT8870D/MT8870D-1 provides a differential-input operational amplifier as well as a bias source (V_{Ref}) which is used to bias the inputs at mid-rail. Provision is made for connection of a feedback resistor to the op-amp output (GS) for adjustment of gain. In a single-ended configuration, the input pins are connected as shown in Figure 10 with the op-amp connected for unity gain and V_{Ref} biasing the input at $^1/_2V_{DD}$. Figure 6 shows the differential configuration, which permits the adjustment of gain with the feedback resistor R_5.

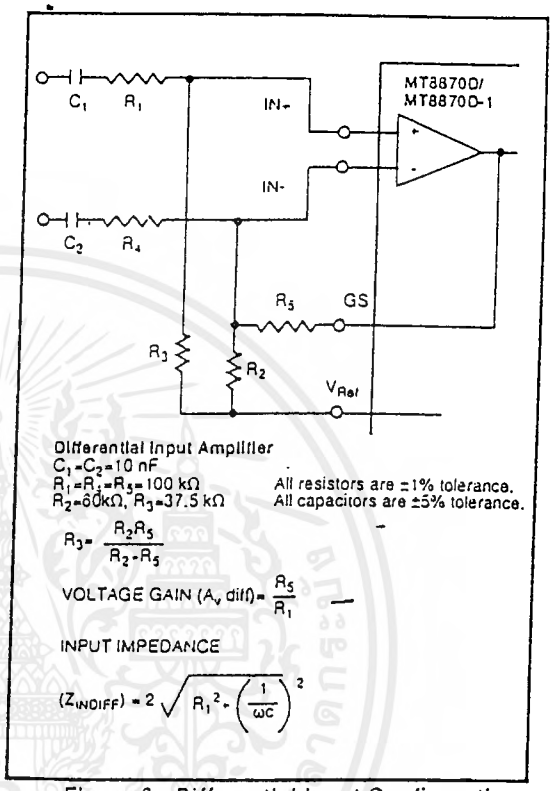


Figure 6 - Differential Input Configuration

Crystal Oscillator

The internal clock circuit is completed with the addition of an external 3.579545 MHz crystal and is normally connected as shown in Figure 10 (Single-Ended Input Configuration). However, it is possible to configure several MT8870D/MT8870D-1 devices employing only a single oscillator crystal. The oscillator output of the first device in the chain is coupled through a 30 pF capacitor to the oscillator input (OSC1) of the next device. Subsequent devices are connected in a similar fashion. Refer to Figure 7 for details. The problems associated with unbalanced loading are not a concern with the arrangement shown, i.e., precision balancing capacitors are not required.

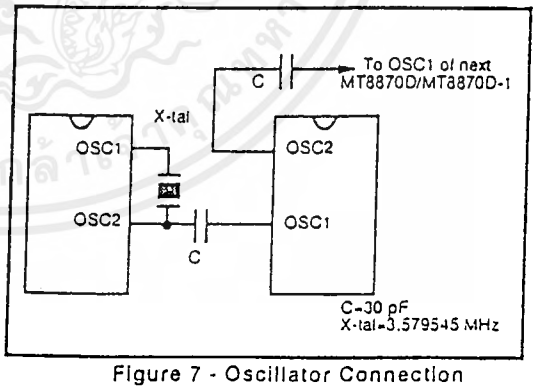


Figure 7 - Oscillator Connection

Parameter	Unit	Resonator
R1	Ohms	10.752
L1	mH	.432
C1	pF	4.984
C0	pF	37.915
Qm	-	896.37
Δf	%	±0.2%

Table 2. Recommended Resonator Specifications

Note: Qm=quality factor of RLC model, i.e., 1/2Π/R1C1.

Applications

RECEIVER SYSTEM FOR BRITISH TELECOM SPEC POR 1151

The circuit shown in Fig. 9 illustrates the use of MT8870D-1 device in a typical receiver system. BT Spec defines the input signals less than -34 dBm as the non-operate level. This condition can be attained by choosing a suitable values of R_1 and R_2 to provide 3 dB attenuation, such that -34 dBm input signal will correspond to -37 dBm at the gain setting pin GS of MT8870D-1. As shown in the diagram, the component values of R_3 and C_2 are the guard time requirements when the total component tolerance is 6%. For better performance, it is recommended to use the non-symmetric guard time circuit in Fig. 8.

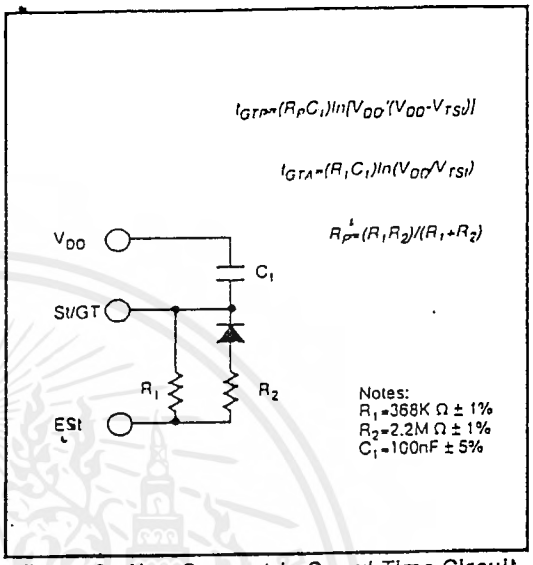


Figure 8 - Non-Symmetric Guard Time Circuit

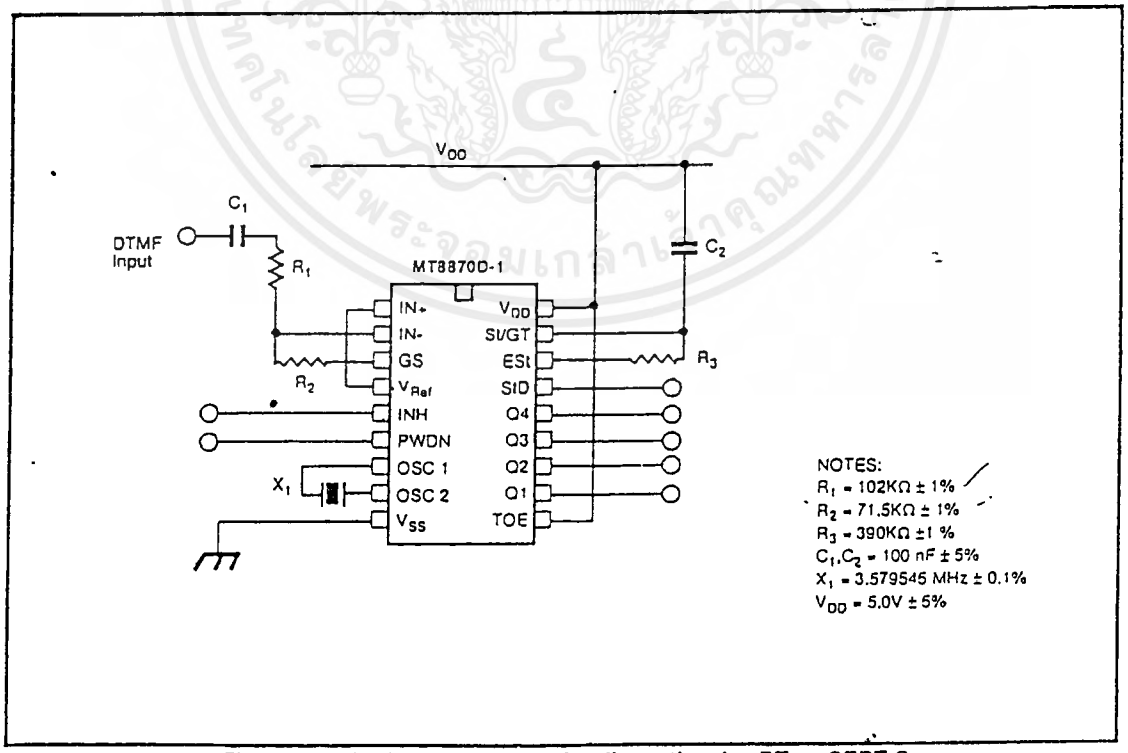


Figure 9 - Single-Ended Input Configuration for BT or CEPT Spec

Absolute Maximum Ratings†

	Parameter	Symbol	Min	Max	Units
1	DC Power Supply Voltage	V_{DD}		7	V
2	Voltage on any pin	V_I	V_{SS}-0.3	V_{DD}+0.3	V
3	Current at any pin (other than supply)	I_I		10	mA
4	Storage temperature	T_{STG}	-65	+150	°C
5	Package power dissipation	P_D		500	mW

† Exceeding these values may cause permanent damage. Functional operation under these conditions is not implied.
Derate above 75 °C at 16 mW / °C. All leads soldered to board.

Recommended Operating Conditions - Voltages are with respect to ground (V_{SS}) unless otherwise stated.

	Parameter	Sym	Min	Typ‡	Max	Units	Test Conditions
1	DC Power Supply Voltage	V_{DD}	4.75	5.0	5.25	V	
2	Operating Temperature	T_O	-40		+85	°C	
3	Crystal/Clock Frequency	fc		3.579545		MHz	
4	Crystal/Clock Freq.Tolerance	Δfc		±0.1		%	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

DC Electrical Characteristics - V_{DD}=5.0V± 5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C, unless otherwise stated.

		Characteristics	Sym	Min	Typ‡	Max	Units	Test Conditions
1	SUPPLY	Standby supply current	I_{DDQ}		10	25	μA	PWDN=V_{DD}
2		Operating supply current	I_{DD}		3.0	9.0	mA	
3		Power consumption	P_O		15		mW	fc=3.579545 MHz
4	INPUTS	High level input	V_{IH}	3.5			V	V_{DD}=5.0V
5		Low level input voltage	V_{IL}			1.5	V	V_{DD}=5.0V
6		Input leakage current	I_{IH}/I_{IL}		0.1		μA	V_{IN}=V_{SS} or V_{DD}
7		Pull up (source) current	I_{SO}		7.5	20	μA	TOE (pin 10)=0, V_{DD}=5.0V
8		Pull down (sink) current	I_{SI}		15	45	μA	INH=5.0V, PWDN=5.0V, V_{DD}=5.0V
9		Input impedance (IN+, IN-)	R_{IN}		10		MΩ	@ 1 kHz
10		Steering threshold voltage	V_{TSt}	2.2	2.4	2.5	V	V_{DD} = 5.0V
11	OUTPUTS	Low level output voltage	V_{OL}			V_{SS}+0.03	V	No load
12		High level output voltage	V_{OH}	V_{DD}-0.03			V	No load
13		Output low (sink) current	I_{OL}	1.0	2.5		mA	V_{OUT}=0.4 V
14		Output high (source) current	I_{OH}	0.4	0.8		mA	V_{OUT}=4.6 V
15		V_{Ref} output voltage	V_{Ref}	2.3	2.5	2.7	V	No load, V_{DD} = 5.0V
16		V_{Ref} output resistance	R_{OR}		1		kΩ	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

Operating Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C, unless otherwise stated.

Gain Setting Amplifier

	Characteristics	Sym	Min	Typ‡	Max	Units	Test Conditions
1	Input leakage current	I_{IN}			100	nA	$V_{SS} \le V_{IN} \le V_{DD}$
2	Input resistance	R_{IN}	10			MΩ	
3	Input offset voltage	V_{OS}			25	mV	
4	Power supply rejection	PSRR	50			dB	1 kHz
5	Common mode rejection	CMRR	40			dB	0.75 V ≤ V_{IN} ≤ 4.25 V biased at V_{Ref}=2.5 V
6	DC open loop voltage gain	A_{VOL}	32			dB	
7	Unity gain bandwidth	f_C	0.30			MHz	
8	Output voltage swing	V_O	4.0			V_{pp}	Load ≥ 100 kΩ to V_{SS} @ GS
9	Maximum capacitive load (GS)	C_L			100	pF	
10	Resistive load (GS)	R_L			50	kΩ	
11	Common mode range	V_{CM}	2.5			V_{pp}	No Load

MT8870D AC Electrical Characteristics - V_{DD}=5.0V ±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C, using Test Circuit shown in Figure 10.

	Characteristics	Sym	Min	Typ‡	Max	Units	Notes*
1	Valid input signal levels (each tone of composite signal)		-29		+1	dBm	1,2,3,5,6,9
			27.5		869	mV_{RMS}	1,2,3,5,6,9
2	Negative twist accept				8	dB	2,3,6,9,12
3	Positive twist accept				8	dB	2,3,6,9,12
4	Frequency deviation accept		±1.5% ± 2 Hz				2,3,5,9
5	Frequency deviation reject		±3.5%				2,3,5,9
6	Third tone tolerance			-16		dB	2,3,4,5,9,10
7	Noise tolerance			-12		dB	2,3,4,5,7,9,10
8	Dial tone tolerance			+22		dB	2,3,4,5,8,9,11

‡ Typical figures are at 25 °C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES
1. dBm= decibels above or below a reference power of 1 mW into a 600 ohm load.
2. Digit sequence consists of all DTMF tones.
3. Tone duration= 40 ms, tone pause= 40 ms.
4. Signal condition consists of nominal DTMF frequencies.
5. Both tones in composite signal have an equal amplitude.
6. Tone pair is deviated by ±1.5 %± 2 Hz.
7. Bandwidth limited (3 kHz) Gaussian noise.
8. The precise dial tone frequencies are (350 Hz and 440 Hz) ± 2 %.
9. For an error rate of better than 1 in 10,000.
10. Referenced to lowest level frequency component in DTMF signal.
11. Referenced to the minimum valid accept level.
12. Guaranteed by design and characterization.

MT8870D-1 AC Electrical Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ T_O ≤ +85°C , using Test Circuit shown in Figure 10.

	Characteristics	Sym	Min	Typ‡	Max	Units	Notes*
1	Valid input signal levels (each tone of composite signal)		-31		+1	dBm	Tested at V_{DD}=5.0V 1,2,3,5,6,9
			21.8		869	mV_{RMS}	
2	Input Signal Level Reject		-37			dBm	Tested at V_{DD}=5.0V 1,2,3,5,6,9
			10.9			mV_{RMS}	
3	Negative twist accept				8	dB	2,3,6,9,13
4	Positive twist accept				8	dB	2,3,6,9,13
5	Frequency deviation accept		±1.5%± 2 Hz				2,3,5,9
6	Frequency deviation reject		±3.5%				2,3,5,9
7	Third zone tolerance			-18.5		dB	2,3,4,5,9,12
8	Noise tolerance			-12		dB	2,3,4,5,7,9,10
9	Dial tone tolerance			+22		dB	2,3,4,5,8,9,11

‡ Typical figures are at 25 °C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES
1. dBm= decibels above or below a reference power of 1 mW into a 600 ohm load.
2. Digit sequence consists of all DTMF tones.
3. Tone duration= 40 ms, tone pause= 40 ms.
4. Signal condition consists of nominal DTMF frequencies.
5. Both tones in composite signal have an equal amplitude.
6. Tone pair is deviated by ±1.5 %± 2 Hz.
7. Bandwidth limited (3 kHz) Gaussian noise.
8. The precise dial tone frequencies are (350 Hz and 440 Hz) ± 2 %.
9. For an error rate of better than 1 in 10,000.
10. Referenced to lowest level frequency component in DTMF signal.
11. Referenced to the minimum valid accept level.
12. Referenced to Fig. 10 input DTMF tone level at -25dBm (-28dBm at GS Pin) interference frequency range between 480-3400Hz.
13. Guaranteed by design and characterization.

AC Electrical Characteristics - V_{DD}=5.0V±5%, V_{SS}=0V, -40°C ≤ To ≤ +85°C, using Test Circuit shown in Figure 10.

		Characteristics	Sym	Min	Typ‡	Max	Units	Conditions
1	TIMING	Tone present detect time	t_{DP}	5	11	14	ms	Note 1
2		Tone absent detect time	t_{DA}	0.5	4	8.5	ms	Note 1
3		Tone duration accept	t_{REC}			40	ms	Note 2
4		Tone duration reject	$\overline{t_{REC}}$	20			ms	Note 2
5		Interdigit pause accept	t_{ID}			40	ms	Note 2
6		Interdigit pause reject	t_{DO}	20			ms	Note 2
7	OUTPUTS	Propagation delay (St to Q)	t_{PQ}		8	11	µs	TOE=V_{DD}
8		Propagation delay (St to StD)	t_{PStD}		12	16	µs	TOE=V_{DD}
9		Output data set up (Q to StD)	t_{QStD}		3.4		µs	TOE=V_{DD}
10		Propagation delay (TOE to Q ENABLE)	t_{PTE}		50		ns	load of 10 kΩ, 50 pF
11		Propagation delay (TOE to Q DISABLE)	t_{PTD}		300		ns	load of 10 kΩ, 50 pF
12	PDWN	Power-up time	t_{PU}		30		ms	Note 3
13		Power-down time	t_{PD}		20		ms	
14	CLOCK	Crystal/clock frequency	f_C	3.5759	3.5795	3.5831	MHz	
15		Clock input rise time	t_{LHCL}			110	ns	Ext. clock
16		Clock input fall time	t_{HLCL}			110	ns	Ext. clock
17		Clock input duty cycle	DC_{CL}	40	50	60	%	Ext. clock
18		Capacitive load (OSC2)	C_{LO}			30	pF	

‡ Typical figures are at 25°C and are for design aid only: not guaranteed and not subject to production testing.

*NOTES:
1. Used for guard-time calculation purposes only.
2. These, user adjustable parameters, are not device specifications. The adjustable settings of these minimums and maximums are recommendations based upon network requirements.
3. With valid tone present at input, t_{PU} equals time from PDWN going low until ESt going high.

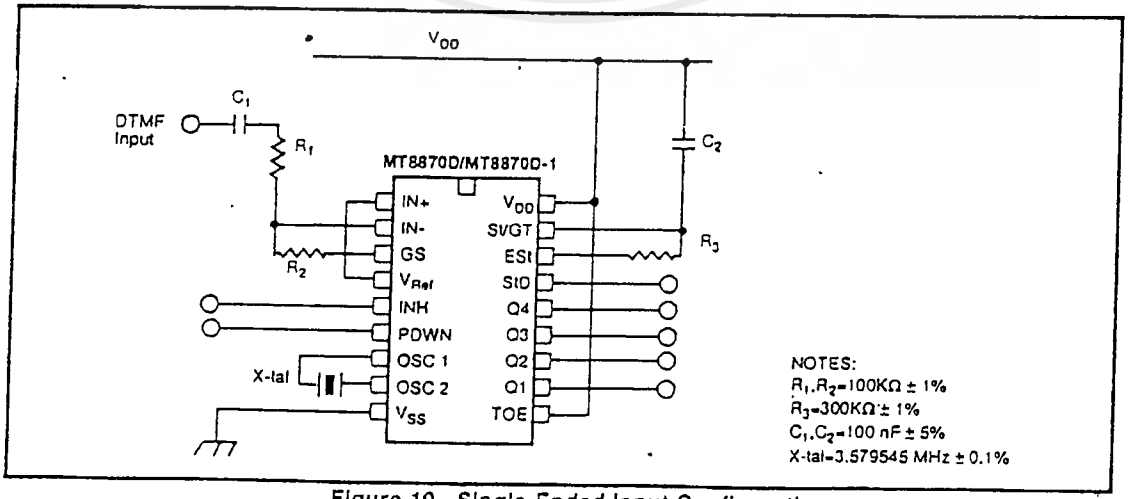
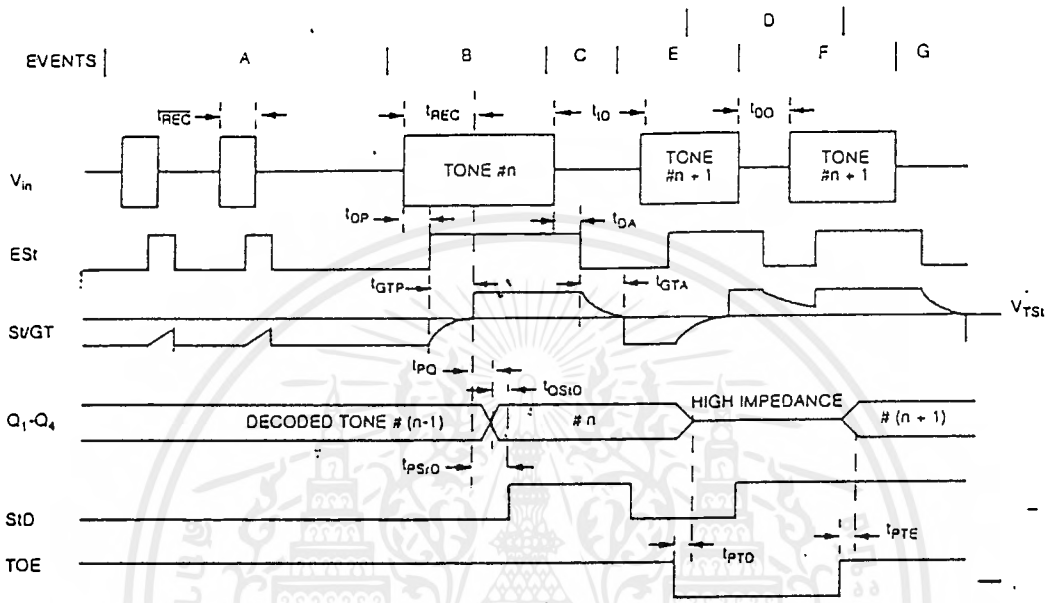


Figure 10 - Single-Ended Input Configuration



EXPLANATION OF EVENTS

A) TONE BURSTS DETECTED, TONE DURATION INVALID, OUTPUTS NOT UPDATED.

B) TONE #n DETECTED, TONE DURATION VALID, TONE DECODED AND LATCHED IN OUTPUTS

C) END OF TONE #n DETECTED, TONE ABSENT DURATION VALID, OUTPUTS REMIAN LATCHED UNTIL NEXT VALID TONE.

D) OUTPUTS SWITCHED TO HIGH IMPEDANCE STATE.

E) TONE #n + 1 DETECTED, TONE DURATION VALID, TONE DECODED AND LATCHED IN OUTPUTS (CURRENTLY HIGH IMPEDANCE).

F) ACCEPTABLE DROPOUT OF TONE #n + 1, TONE ABSENT DURATION INVALID, OUTPUTS REMAIN LATCHED.

G) END OF TONE #n + 1 DETECTED, TONE ABSENT DURATION VALID, OUTPUTS REMAIN LATCHED UNTIL NEXT VALID TONE.

EXPLANATION OF SYMBOLS

V_{in} DTMF COMPOSITE INPUT SIGNAL.

ESt EARLY STEERING OUTPUT. INDICATES DETECTION OF VALID TONE FREQUENCIES.

St/GT STEERING INPUT/GUARD TIME OUTPUT. DRIVES EXTERNAL RC TIMING CIRCUIT.

Q_1-Q_4 4-BIT DECODED TONE OUTPUT.

StD DELAYED STEERING OUTPUT. INDICATES THAT VALID FREQUENCIES HAVE BEEN PRESENT/ABSENT FOR THE REQUIRED GUARD TIME THUS CONSTITUTING A VALID SIGNAL.

TOE TONE OUTPUT ENABLE (INPUT). A LOW LEVEL SHIFTS Q_1-Q_4 TO ITS HIGH IMPEDANCE STATE.

$t_{\overline{REC}}$ MAXIMUM DTMF SIGNAL DURATION NOT DETECED AS VALID

t_{REC} MINIMUM DTMF SIGNAL DURATION REQUIRED FOR VALID RECOGNITION

t_{ID} MAXIMUM TIME BETWEEN VALID DTMF SIGNALS.

t_{DO} MAXIMUM ALLOWABLE DROP OUT DURING VALID DTMF SIGNAL.

t_{DP} TIME TO DETECT THE PRESENCE OF VALID DTMF SIGNALS.

t_{DA} TIME TO DETECT THE ABSENCE OF VALID DTMF SIGNALS.

t_{GTP} GUARD TIME, TONE PRESENT.

t_{GTA} GUARD TIME, TONE ABSENT.

Figure 11 - Timing Diagram

UMC

UMC

UM95087

Tone Dialer

Features

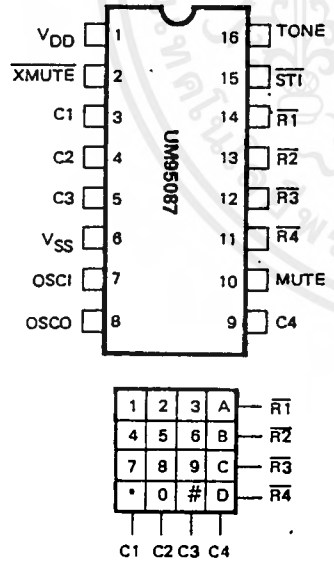
- Operating voltage range: 3.5 to 10.0 Volts
- Uses 3.58 MHz TV crystal to derive all frequencies, providing high accuracy and stability
- On-chip regulation of dual and single-tone amplitudes
- Built-in auxiliary switching functions
- Built-in mute driver
- Minimum external parts count
- Multiple key entry is pin selectable for either single tone or no tone

General Description

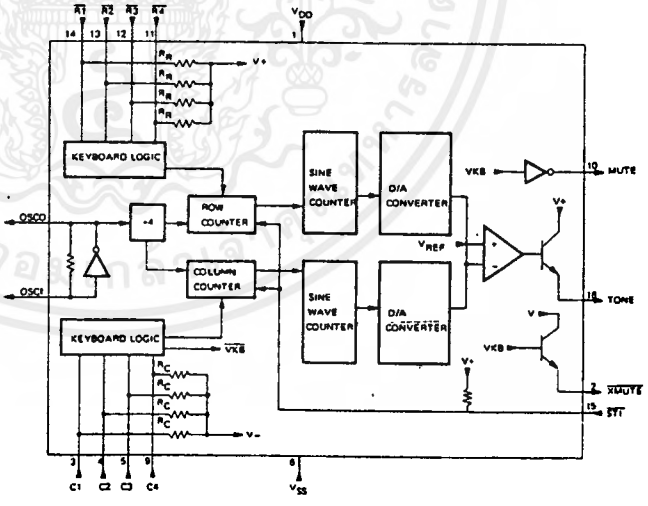
The UM95087 is a monolithic CMOS integrated circuit designed for Dual-Tone-Multi-Frequency (DTMF) telephone dialing.

Its features include single contact static keyboard input; single-tone inhibit ($\overline{STI}$) option; wide supply voltage operation with regulated output. The UM95087 provides high performance with low output tone distortion: THD < −20d8.

Pin Configuration & Keyboard Assignments



Block Diagram



Absolute Maximum Ratings *

Power Supply Voltage . . . . . –0.3V to V
Input Voltage . . . . . –0.3V to V_{DD} + 0.3V
Maximum Power Dissipation . . . . . 500 mW
Operating Temperature . . . . . –20°C to +70°C
Storage Temperature . . . . . –55°C to +150°C

***Comments**

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

DC Electrical Characteristics (T_{OP} = 25°C, F_{OSC} = 3.579 MHz)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions		Test Ckt.
Operating Voltage	V_{DD}	3.5		10	V			B
Supply Operating Current	I_{DD1}		1.0	2.0	mA	V_{DD} = 3.5V	All outputs unloaded	B
	I_{DD2}		13	20	mA	V_{DD} = 10.0V	Oscillator running	
Standby Current	I_{SD1}		0.2	100	μA	V_{DD} = 3.5V	All outputs unloaded	A
	I_{SD2}		0.3	200	μA	V_{DD} = 10.0V	Oscillator not running	
$\overline{XMUTE}$ Output Current	I_{OH1}	–15	–25		mA	V_{DD} = 3.5V, V_{OH} = 2.5V, No key entry		C
	I_{OH2}	–20	–31		mA	V_{DD} = 10.0V, V_{OH} = 8.0V, No key entry		
XMTR Output Leakage Current	I_{OL1}		0.1	100	μA	V_{DD} = 10.0V, V_{OL} = 0.0V, with key entry		C
MUTE Output Drive Current	I_{OH1}	–0.5	–0.8		mA	V_{DD} = 3.5V, V_{OH} = 3V, with key entry		C
	I_{OH2}	–1	–4.5		mA	V_{DD} = 10.0V, V_{OH} = 9.5V, with key entry		
MUTE Output Sink Current	I_{OL1}		3.0		mA	V_{DD} = 3.5V, V_{OL} = 0.5V, No key entry		C
	I_{OL2}		7.5		mA	V_{DD} = 10.0V, V_{OL} = 0.5V, No key entry		
Row Input Current	I_R	100	175		μA	V_{IN} = 0V, All outputs unloaded		–
Column Input Current	I_C	100	140		μA	V_{IN} = 3.5V, All outputs unloaded		–
Input Voltage Range	V_{IH}	0.8	V_{DD}		V_{DD}			–
	V_{IL}		0	0.2	V_{DD}			
Single Column Tone Output Amplitude	V_{OC}	396	500	630	mV_{P-P}	R_L = 1 KΩ, V_{DD} = 3.5V		B
		396	500	630		R_L = 1 KΩ, V_{DD} = 10.0V		
Single Row Tone Output Amplitude	V_{OR}	317	400	504	mV_{P-P}	R_L = 1 KΩ, V_{DD} = 3.5V		B
		317	400	504		R_L = 1 KΩ, V_{DD} = 10.0V		
Pre-Emphasis	T_{WIST}	1.0	2.0	3.0	dB			B
Valley of Single Tone	V_V	0.23	3.0		V_{DD}			B
Distortion	DIS%			10	%	V_{DD} = 5 * Note		B
Tone Output External Load Impedance	R_L	620			Ω	V_{DD} = 3.5V		–
		330			Ω	V_{DD} = V		
$\overline{STI}$ Input Resistance	R_{IN}	20	40	100	KΩ	25°C		–

*Note: $DIS(\%) = \frac{100(V_1^2 + V_2^2 + \ldots + V_n^2)^{1/2}}{(V_{IL}^2 + V_{IH}^2)^{1/2}}$

(a) $V_1 \ldots V_n$ are the intermodulation or harmonic frequencies in the 500 Hz to 3400 Hz band.

(b) V_{IL} and V_{IH} are the individual frequency components of the DTMF signal.

AC Characteristics

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions
Tone Output Rise Time	T_R		8.5		ms	
OSC Startup Timing	T_{START}		8		ms	
Predigital Pause	T_{TPDP}		5		ms	
Mute Delay	T_{MOT}		5		ms	

R/C	Spec.	Actual	Error (%)	Unit	Conditions
$\overline{R1}$	697	701.3	+0.62	Hz	F_{OSC} = 3.579 MHz
$\overline{R2}$	770	771.4	+0.19	Hz	
$\overline{R3}$	852	857.2	+0.61	Hz	
$\overline{R4}$	941	935.1	−0.63	Hz	
C1	1,209	1,215.9	+0.57	Hz	
C2	1,336	1,331.7	−0.32	Hz	
C3	1,477	1,471.9	−0.35	Hz	
C4	1,633	1,645.0	+0.73	Hz	

Note: % Error does not include oscillator drift.

Pin Description

Pin No.	Designation	Description
1 6	V_{DD} V_{SS}	Power supply. These are the power supply input pins. The UM95087 is designed to operate from 3.5 to 10.0 volts.
2	$\overline{XMUTE}$	$\overline{XMUTE}$ switch output. This pin is connected to the emitter of an on-chip bipolar transistor whose collector is connected to V_{DD}. With no keyboard input, this transistor is turned on and pulls the pin up to within V_{BE} of the V_{DD} supply. When a keyboard entry is sensed, this output goes open circuit (high impedance). The XMTR switch output switches regardless of the state of the $\overline{STI}$ pin input.
3 4 5 9 14 13 12 11	C1 C2 C3 C4 $\overline{R1}$ $\overline{R2}$ $\overline{R3}$ $\overline{R4}$	Keyboard inputs. The UM95087 features inputs compatible with the standard 2-of-8 keyboard (the inexpensive single-contact Form A keyboard). No noise is generated by scanned or dynamic input. Normal operation with a keyboard produces dual tone generation when a single button is pushed. A single tone is generated when two or more buttons in the same row are pushed. Pushing two buttons diagonal to each other will result in no tone being generated. When the push of a button produces input to a single row and column, the tone for a digit is generated. Input to two buttons in a single column will produce a column tone. Input to more than one column will not produce any tone. The internal circuit of the UM95087 will not sense the activation of a single row. If a single-row tone is desired, two columns in the desired row must be activated.

Pin Description (Continued)

Pin No.	Designation	Description
7 8	OSCI OSCO	Oscillator input/output. The UM95087 contains an on-chip inverter with sufficient loop gain to provide oscillation when working with a low cost television color-burst crystal. The circuit is designed to work with a 3.579 MHz crystal. The oscillator is disabled whenever a keyboard input is absent. Any crystal frequency deviation from 3.579 MHz will be reflected in the tone output frequency. Most crystals do not vary more than ±0.02%.
10	MUTE	MUTE output. The MUTE output is a conventional CMOS gate that pulls to V_{SS} with no keyboard input and pulls to V_{DD} supply when a keyboard entry is sensed. This output is used to control auxiliary switching functions that are required to actuate upon keyboard input. The MUTE output switches regardless of the state of $\overline{STI}$ pin input.
15	$\overline{STI}$	Single tone inhibit input. The $\overline{STI}$ input is used to inhibit the generation of other than dual tones. It has a pull-up to V_{DD} and when left floating or tied to V_{DD}, single or dual tones may be generated. When forced to V_{SS}, any input situation that would result in a single tone will now result in no tone, but all other chip functions will operate normally.
16	TONE	DTMF signal output. The TONE OUT is connected internally in the UM95087 to the emitter of an NPN transistor and is the on-chip operational amplifier which mixes the row and column tones together.

Functional Description

1. Keyboard

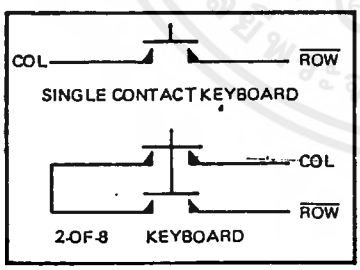


Figure 1 Keyboard Configurations

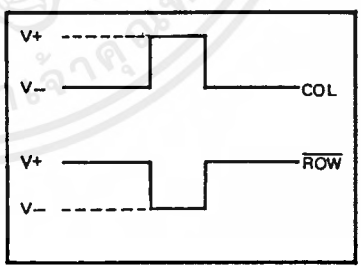


Figure 2 Electrical Inputs

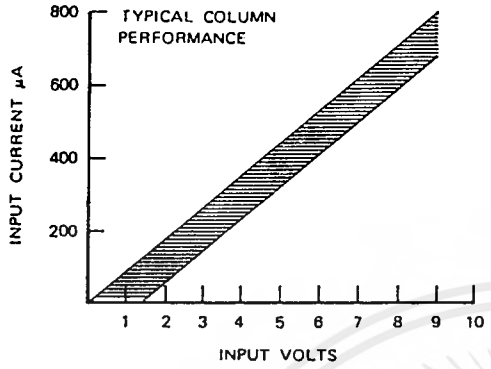


Figure 3 Typical Input Operating Conditions for Pins 3, 4, 5 and 9 with Voltage Reference V_{SS} @ 25°C.

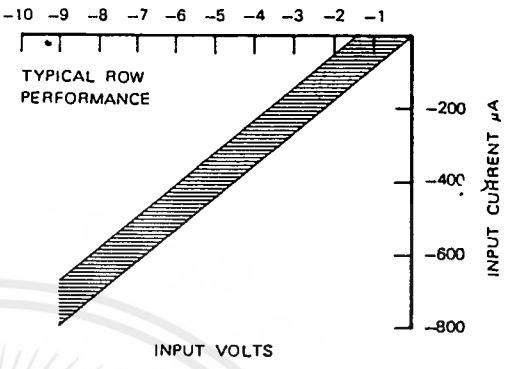


Figure 4 Typical Input Operating Conditions for Pins 11, 12, 13, & 14 with Voltage Reference V_{DD} @ 25°C.

2. Crystal Specification

A standard television color burst crystal is specified to have much tighter tolerance than necessary for the generation of tones. The tolerance specification can be relaxed as follows:

Frequency: 3.58 MHz ± 0.02%.

$R_S < 100\ \Omega$, $L_M = 96mH$, $C_M = 0.25pF$, $C_H = 5pF$, $C_L = 18pF$.

3. The DTMF Generator

The Row and Column output waveforms are shown in Figures 5 and 6 below. In the UM95087, these waveforms are digitally synthesized using on-chip D to A converters, and are provided with output level regulation. The maximum THD of the dual tone is -20dB.

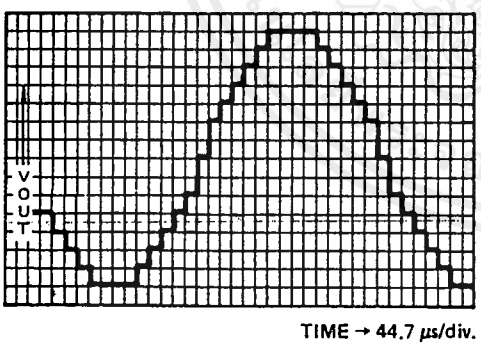


Figure 5 Single Row Output Waveform

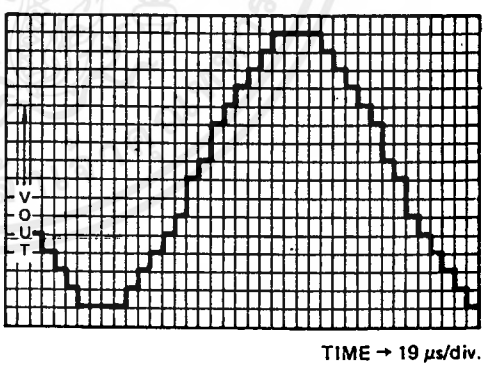
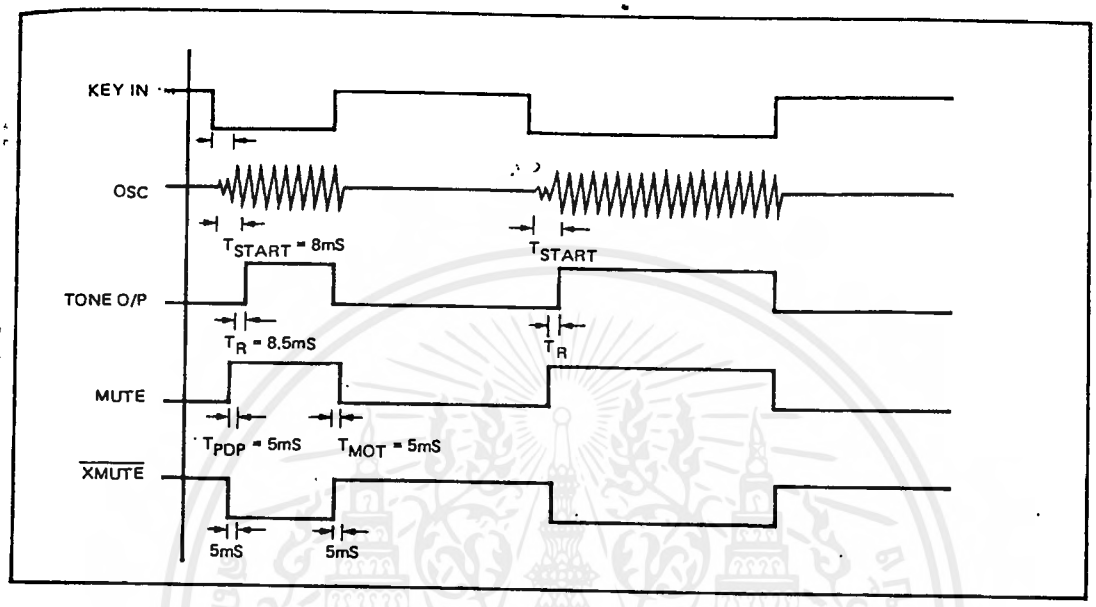
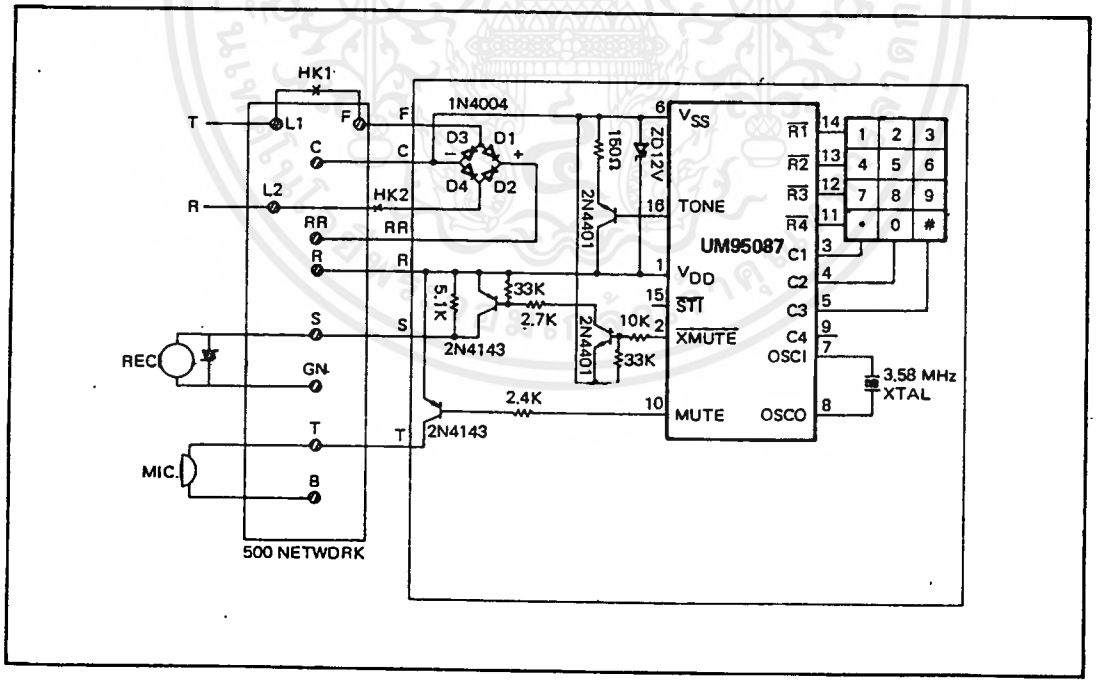


Figure 6 Single Column Output Waveform

Timing Waveform

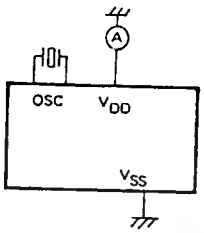


Application Circuit (for reference only)

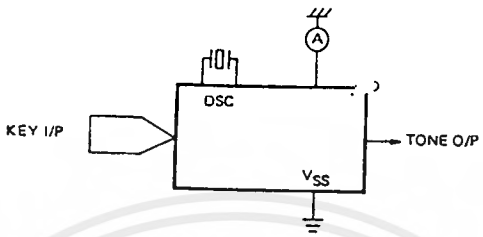


Test Circuits

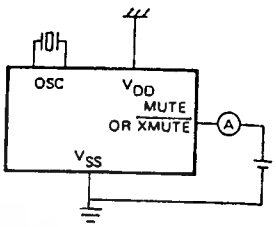
(A)



(B)



(C)



Ordering Information

Part No.	Package
UM95087	16L-DIP