ชุดบันทึกเสียงพูด และตรวจสอบข้อมูลทางโทรศัพท์

สมชาย ธรรรองพาณิชย์

สุทธินันธ์ แคงสกุล

ปริญญานิพนธ์ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขา อิเล็คทรอนิคส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยี พระจอมเกล้า วิทยาเขตเจ้าคุณทหารลาดกระบัง

2533

ชุดบันทึกเสียงพูด และ ตรวจสอบข้อมูลทางโทรศัพท์

สมชาย ธรรรองพาณิชย์

สุกรินันธ์ แดงสกุล

ได้รับพิจารณาอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชา อิเล็คทรอนิคส์

คณะกรรมการตรวจสอบ ปริญญานิพนธ์

_____ประธานกรรมการ

(ผ.ศ. นิกร สุขุ.คมตันติ)

_____กรรมการ

( )

_____กรรมการ

( )

_____กรรมการ

( )

วันที่ 18 เดือน เมษายน พ.ศ. 2534

ชุดบันทึกเสียงพูด และตรวจสอบข้อมูลทางโทรศัพท์

สมชาย ธรรรองพาณิชย์

สุกธินันธ์ แดงสกุล

ผ.ศ. นิกร สุขุตมคันติ
ปีการศึกษา 2533

บทคัดย่อ

ปัจจุบันนี้ การติดต่อสื่อสาร นับเป็นสิ่งสำคัญในชีวิตประจำวัน ทำให้เรารับทราบข่าวสารข้อมูลที่ถูกต้อง โทรศัพท์นับว่าเป็นวิธีการติดต่อสื่อสารที่รวดเร็ว สะดวกทั้งยังประหยัด เมื่อเทียบกับการสื่อสารชนิดอื่น ด้วยข้อดีเหล่านี้ จึงทำให้ผู้จัดทำเกิดแนวความคิด พัฒนาปรับปรุงการสื่อสารด้วยโทรศัพท์ เพื่อใช้ให้เกิดประโยชน์สูงสุดชุดบันทึกเสียงพูดและตรวจสอบข้อมูลทางโทรศัพท์ ถูกสร้างขึ้นโดยนำวงจรหลายชนิดมาทำงานร่วมกัน อาทิ ภาคบันทึกเสียงพูด , ภาคถอดรหัสความถี่โทรศัพท์ โดยอาศัย ไมโครโปรเซสเซอร์ เป็นตัวควบคุมการทำงาน ผู้จัดทำหวังว่า ปริญญานิพนธ์นี้ จะก่อประโยชน์และเป็นแนวทางในการพัฒนา ระบบสื่อสารทางโทรศัพท์ต่อไป

ผู้จัดทำ

สารบัญ

คำนำ

ตั้งแต่เทคโนโลยีทางด้านสารกึ่งตัวนำได้รับการพัฒนาขึ้นเรื่อย ๆ ได้มีการผลิตไอซีเฉพาะงานขึ้นอย่างมากมาย ทุกแขนง ทั้งด้านการสื่อสาร ด้านคอมพิวเตอร์และด้านอุตสาหกรรม ในการทำงานบางอย่างเราอาจต้องใช้องค์ประกอบหลายอย่าง เช่น นำคอมพิวเตอร์ประยุกต์เข้ากับสื่อสาร ดังนั้นหากเราสามารถนำไอซีเฉพาะงานด้านต่างกันประยุกต์เข้าด้วยกัน ก็จะเกิดประสิทธิภาพในการใช้งาน

วิทยานิพนธ์ฉบับนี้ จะประกอบด้วยส่วนสำคัญแบ่งได้ดังนี้

1. ส่วนควบคุม อาศัยไมโครโปรเซสเซอร์มาใช้ เพื่อควบคุมการลำดับการทำของเครื่อง และตรวจสอบการโทรเข้ามายังเครื่อง
2. ส่วนบันทึกเสียงพูด อาศัยไอซี T6668 เป็นเปลี่ยนสัญญาณเพื่อเก็บเข้าหน่วยความจำไดนามิกแรม
3. ส่วน DECODER สัญญาณโทรศัพท์ เพื่อนำไปควบคุมอุปกรณ์ไฟฟ้าตามต้องการ

ซึ่งในเล่มนี้จะกล่าวถึง ส่วนบันทึกเสียงพูดกับส่วนถอดรหัสสัญญาณความถี่โทรศัพท์ชนิดกดปุ่ม

บทที่ 1

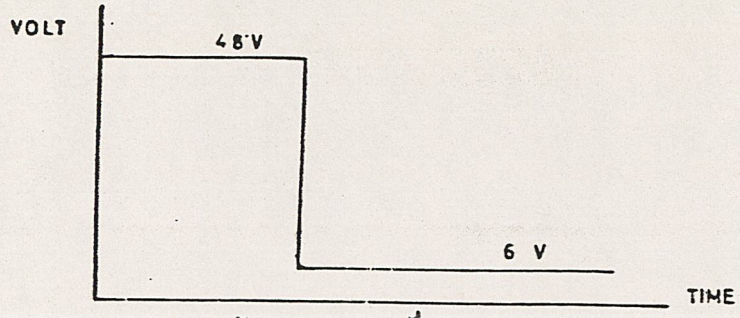
การทำงานของโทรศัพท์

1.1 แนะนำวิธีการใช้และสำรวจการทำงานของโทรศัพท์

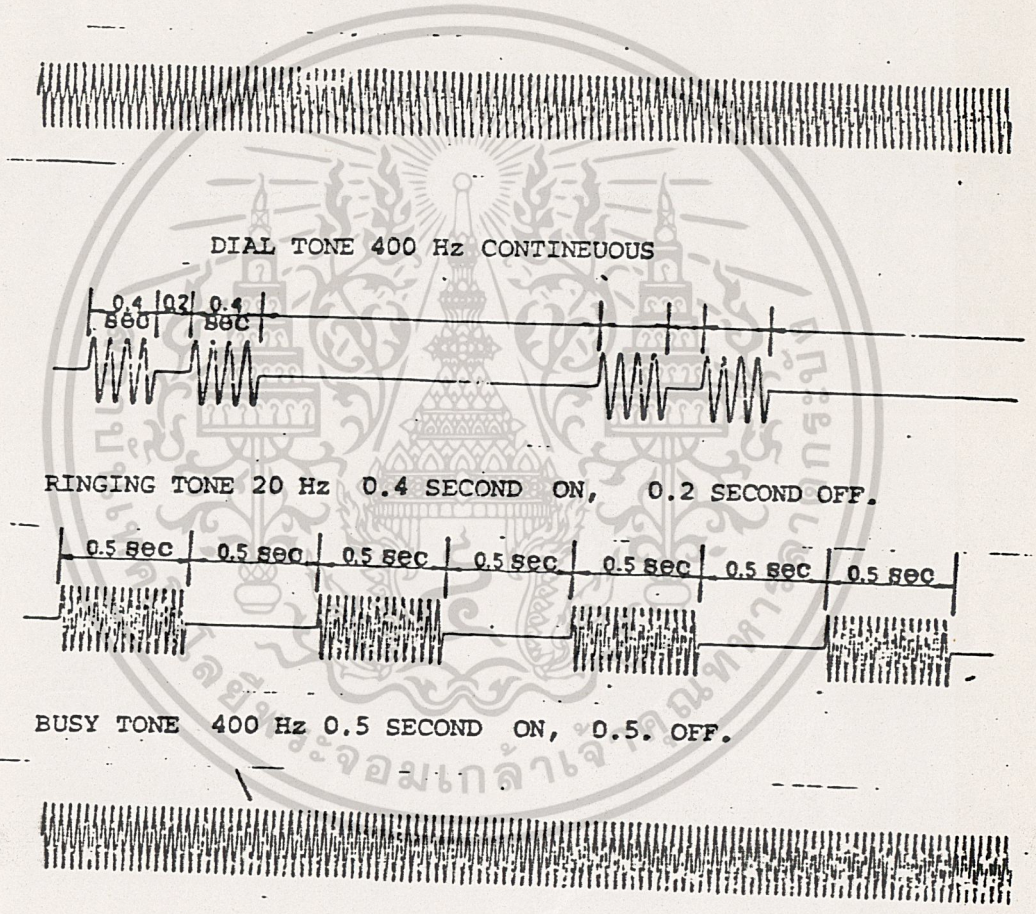
การหมุนหน้าปัทม์โทรศัพท์ ตามเลขหมายที่ประสงค์จะติดต่อควรใช้นิ้วชี้ขวาสอดไปในช่องหมายเลขที่ต้องการ แล้วหมุนมาทางขวาให้สุดจนถึงที่กั้นแล้วดึงนิ้วออก ให้หน้าปัทม์หมุนกลับเองด้วยความเร็วขนาดมาตรฐานที่ปรับแต่งไว้ให้ได้ 20 แรงกระตุ้นต่อ 1 วินาที (20 IMPULSE PER SECOND 20 IPS) อย่าได้ใช้นิ้วช่วยหมุนกลับเป็นอันขาด จะทำให้เกิดการผิดพลาดคลาดเคลื่อนในการต่อโทรศัพท์ เพราะการส่งสัญญาณแรงกระตุ้นให้เครื่องอุปกรณ์ชุมสายทำงานนั้น เครื่องรับโทรศัพท์จะทำงานในตอนหน้าปัทม์หมุนกลับมาทางซ้ายเท่านั้น อีกประการหนึ่งนอกจากนิ้วชี้ขวาที่ใช้หมุนหน้าปัทม์แล้วอย่าใช้ดินสอปากกา หรือของแข็งอื่นใดสอดใส่ในช่องเลขหมายเพื่อหมุนหน้าปัทม์เป็นอันขาด การหมุนหน้าปัทม์ให้หมุนตัวเลขแต่ละตัวมาทางขวาจนสุดที่กั้นอย่างสม่ำเสมอจนครบทุกตัวเลขอย่าปล่อยทิ้งจังหวะนานเกินไป และอย่ารีบเร่งหมุนเร็วเกินไป

การฟังสัญญาณ เมื่อยกหูโทรศัพท์ขึ้นก่อนจะหมุนหน้าปัทม์ให้ฟังสัญญาณว่าง หรือเสียงดายล์ (DIAL TONE) ซึ่งมีเสียงดังยาวติดต่อกันไปอย่างชัดเจน ไม่มีเสียงอื่นใดมารบกวน เมื่อได้ยินสัญญาณดังกล่าวแล้วจึงลงมือหมุนเลขหมาย ถ้าไม่ได้ยินเสียงสัญญาณให้วางหูรอไว้สักระยะหนึ่ง เพราะไม่ใช่ของแปลกในการยกหูโทรศัพท์ขึ้นมาแล้วไม่มีสัญญาณว่างทันที โดยเฉพาะในระหว่างชั่วโมงเร่งรัดของวันทำงาน ทั้งนี้เพราะอุปกรณ์เครื่องชุมสายโทรศัพท์ได้กำหนดขีดความสามารถ ในการจราจรผ่านอุปกรณ์ชุมสายและเคเบิลสายผ่านรวมกันในอัตราสูงสุด 120 ราย ต่อ 1000 เลขหมาย ยกเว้นจะมีเหตุขัดข้องผิดปกติที่ยกหูฟังเท่าไรก็ไม่ได้ยินเสียงสัญญาณว่าง แต่มีลักษณะอย่างอื่นมาแทนตลอดเวลา ก็ขอให้หาโทรศัพท์เลขหมายอื่นหมุนแจ้งเหตุขัดข้อง เพื่อช่างจะได้ตรวจซ่อมต่อไป

เมื่อหมุนเลขที่ต้องการเรียกแล้ว จะได้ยินเสียงเป็นจังหวะดัง 0.5 วินาที แล้วเงียบ 0.5 วินาที ดังสลับกันไป สัญญาณนี้เรียกว่าสัญญาณไม่ว่าง (BUSY TONE) แสดงว่าเลข



แสดงระดับ VOLT เมื่อวางหูและยกหู



รูปที่ 1.1 แสดงให้เห็น TONE ชนิดต่าง ๆ ที่ใช้ในระบบโทรศัพท์

หมายที่เรียกไปนั้นยังไม่ว่าง ให้วางหูโทรศัพท์ไว้ที่เดิมก่อน รอสักครู่กะว่าเลขหมายที่ประสงค์จะติดต่อได้ใช้โทรศัพท์เสร็จแล้วจึงหมุนเรียกไปใหม่

เมื่อหมุนเลขหมายที่ต้องการเรียกแล้ว ได้ยินเสียงเป็นจังหวะดัง 1 วินาที แล้วเงียบ 4 วินาที สัญญาณนี้เรียกว่า RING TONE หรือ CALLING - TONE แสดงว่าเลขหมายที่เรียกไปว่างพร้อมที่จะพูดได้ให้คอยจนกว่าฝ่ายถูกเรียกจะยกหูรับ

เมื่อหมุนเลขหมายที่ต้องการเรียกแล้วเป็นจังหวะถี่ ๆ สั้น 3 ครั้ง ยาว 1 ครั้งสลับกันไป สัญญาณนี้เรียกว่า NU TONE แสดงว่าเลขหมายที่เรียกไปเป็นเลขหมายว่าง ยังไม่มีการติดตั้ง หรือถูกยกเลิกไป

1.2 ชนิดต่าง ๆ ของระบบชุมสาย

1.2.1 MANUAL EXCHANGE SYSTEM.

โทรศัพท์ระบบนี้เป็นระบบง่าย ๆ ใช้หลักการเพื่อบริการผู้ใช้ เครื่องมืออุปกรณ์ก็ไม่ยุ่งยากเท่าใดนักและที่เรียกว่า MAMUAL นั้นเพราะว่าต้องใช้ OPERATOR คอยต่อสายให้

ก. MAGNETO SWITCHBOARD เป็นระบบที่ไม่ยุ่งยากอะไรมากนัก ที่ชุมสายโทรศัพท์ (TELEPHON EXCHANGE) เป็นผู้จ่ายทั้ง TRANSMISSION และ SIGNALLING CURRENT ให้กับเครื่องของผู้เช่า (SUBSCRIBER) เครื่องโทรศัพท์ต้องมี D.C. 3V สำหรับเลี้ยงวงจรพร้อมกันนั้นที่ตัวเครื่องโทรศัพท์เองก็ยังมีเครื่องกำเนิดสำหรับส่งสัญญาณไปยังชุมสาย (TELEPHONE EXCHANGE)

ข. COMMON BATTERY SWITCHBOARD เป็นระบบซึ่งใช้กระแสไฟจากชุมสายโทรศัพท์จ่ายไปยังเครื่องโทรศัพท์ โดยตรงทั้ง TRANSMISSION และสัญญาณ

1.2.2 AUTOMATIC SYSTEM

ในระบบโทรศัพท์อัตโนมัติผู้ใช้โทรศัพท์ไม่ต้องการ OPERATOR คอยต่อให้หมายถึงพอยกหูฟังขึ้นผู้ใช้ก็เพียงแต่หมุนหน้าปัทม์ตามหมายเลขที่ต้อง เสียเวลาหมุนหน้าปัทม์เพียงไม่ถึง 1 นาที ก็จะต่อโทรศัพท์ได้ ซึ่งผิดกันกับแบบ MANUAL เมื่อยกหูฟังขึ้นแล้วต้องรอให้ OPERATOR ก็อาจจะว่างนอน ซึ่งอย่างน้อยก็เสียเวลา 2-3 นาที ซึ่งเป็นการเสียเวลา

โดยใช่เหตุ แต่ระบบอัตโนมัติแบบ STEP BY STEP และครอสบาร์นี้เสียเวลาน้อยมาก เมื่อเรายกหูฟังขึ้น ควรจะฟังดูเสียก่อนว่ามีเสียง DIAL TONE (แมวกรน) หรือไม่ถ้าไม่ได้ยินเสียง DIAL TONE ก็ควรวางหูฟังลงก่อนแล้วยกขึ้นมาฟังใหม่ เมื่อได้ยินเสียง DIAL TONE แล้วก็เริ่มหมุนเลขหมายตามที่เราต้องการได้ ขณะที่เราหมุนนั้น SWITCH และ RELAY ต่าง ๆ ก็จะทำงานเพื่อที่จะทำการต่อไปยังเลขหมายที่เราต้องการจะพูดด้วย เมื่อติดต่อกับผู้ที่เราต้องการจะพูดด้วยแล้ว เราจะได้ยินเสียงกระดิ่งดังเป็นจังหวะ ๆ (20 Hz) ซึ่งเรียกว่า RING BACK TONE และถ้าหากต่อไม่ติดเราก็จะได้รับสัญญาณ BUSY TONE (400 Hz) ซึ่งจังหวะจะเร็วกว่าหรือถี่กว่า เราก็จะต้องวางหูฟังลงแล้วยกขึ้นมาเพื่อที่จะหมุนหน้าปัทม์ใหม่

1.2.3 COMMON CONTROL SYSTEM.

โทรศัพท์ระบบนี้ หรือจะเรียกอีกอย่างหนึ่งว่าระบบครอสบาร์ (X-BAR) ระบบนี้เป็นระบบซึ่งใหม่สำหรับบ้านเรา เป็นระบบซึ่งค่อนข้างยุ่งยาก ซับซ้อนมากกว่าระบบที่กล่าวมาข้างต้น แต่ว่าระบบการทำงานมีประสิทธิภาพดีกว่า เร็วกว่า สะดวกกว่า ราคาก็สูงกว่า และเสียค่ารักษาน้อยกว่า ระบบนี้ทำงานเหมือนสมองกล ทำงานด้วย SWITCH ของ X-BAR และระบบ COMMON CONTROL ซึ่งประกอบด้วย RELAY เป็นจำนวนมากทำหน้าที่แทน OPERATOR เป็นระบบซึ่งปัจจุบันกำลังนิยมใช้กันทั่วโลกทั้ง LOCAL CONNECTION & TOLL CONNECTION.

1.2.4 POWER DRIVEN SYSTEM.

โทรศัพท์ระบบนี้ที่บ้านเราไม่มีใช้ ส่วนมากจะมีใช้ในอเมริกา และยุโรปบางประเทศ และระบบนี้ก็ยังคงใช้ระบบ COMMON CONTROL เช่นเดียวกับระบบ X-BAR ผิดกันแต่สวิสซึ่งใช้มอเตอร์จะหมุนอยู่ตลอดเวลาที่ทำงาน

1.2.5 ELECTRONIC SWITCH

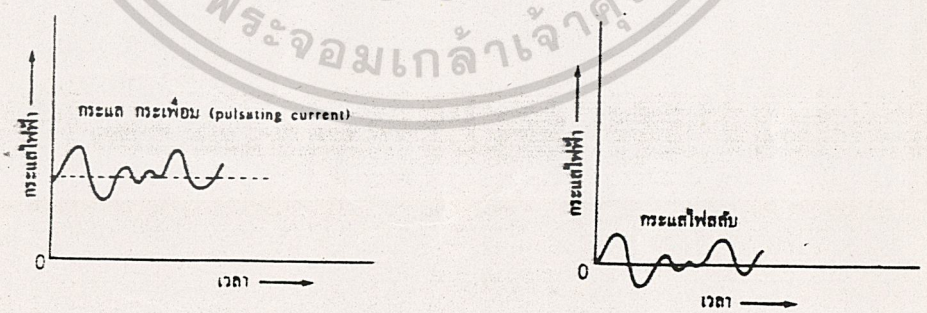
โทรศัพท์ระบบนี้เป็นระบบใหม่สุดในโลก ซึ่งเพิ่งจะผลิตขึ้นมาใช้โดยประเทศเยอรมันเป็นประเทศแรกที่ติดตั้งและใช้งานแล้ว สำหรับประเทศญี่ปุ่นผึ่งจะติดตั้งไปประมาณ 5-15 ชุมสาย นับได้ว่าเป็นประดิษฐ์กรรมที่ใหม่

1.3 อุปกรณ์โทรศัพท์ (ELEPHONE INSTRUMENTS)

1.3.1 TRANSMITTER

TRANSMITTER ที่ใช้ในเครื่องโทรศัพท์มีหลายชนิดแต่ละชนิดจะทำหน้าที่อย่างเดียวกันคือ เปลี่ยนคลื่นเสียงให้เป็นพลังไฟฟ้า เช่น CONDENSER TRANSMITTER , MAGNETIC TYPE TRANSMITTER และ CARBON TRANSMITTER ปัจจุบันนี้ CARBON TRANSMITTER เป็นที่นิยมใช้เพราะมีประสิทธิภาพในการส่งคลื่นเสียงได้ดีกว่าชนิดอื่น

รูปที่ 1.2 หลักการทำงานของ TRANSMITTER



รูปที่ 1.3 แสดงรูปคลื่นขณะที่มีเสียงพูดใน TRANMITTER

หลักการทำงานของ CARBON TRANSMITTER

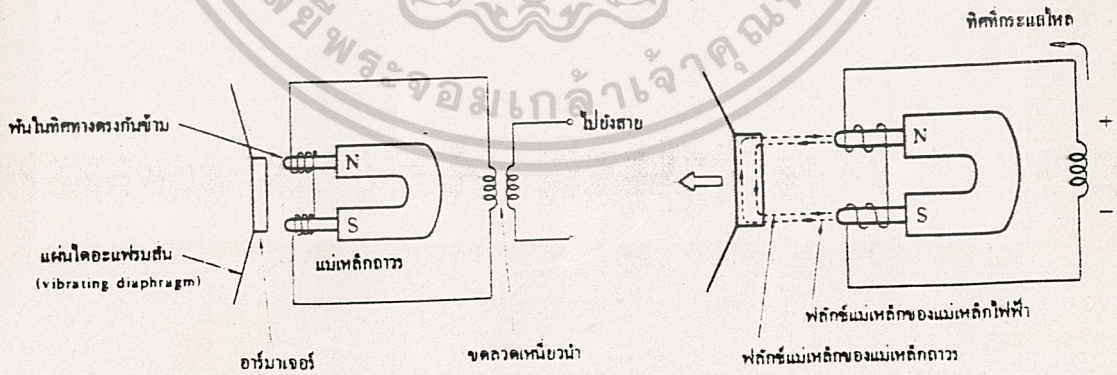
TRANSMITTER ชนิดนี้เปรียบเสมือน A.C. GENERATOR สำหรับคลื่นเสียงที่มี IMPEDANCE ภายในเครื่อง และผลิตแรงเคลื่อนขึ้นมาจำนวนหนึ่งส่งไปในสายซึ่งมีไฟตรงเลี้ยงอยู่ในวงจร พลังงานของคำพูดที่ปรากฏบน TRANSMITTER นี้นับว่าน้อยมาก การที่จะทำให้ได้พลังงานสูงสุดก็คือ ต้อง MACTHING TRANSMITTER กับ LINE IMPEDANCE

อุปกรณ์ที่ประกอบขึ้นเป็น TRANSMITTER ชนิดนี้จะประกอบด้วยผงถ่าน (CARBON GRANUA) หนัก 0.32 กรัม บรรจุอยู่ในกล่องที่มีปริมาตร 1 ม.ม.^{2} และมีแผ่น DIAPHRAGM หนาประมาณ 0.5 ม.ม. วางอยู่หน้ากล่องผงถ่าน ความต้านทานของผงถ่าน ความต้านทานของผงถ่านประมาณ 20-30 โอห์ม

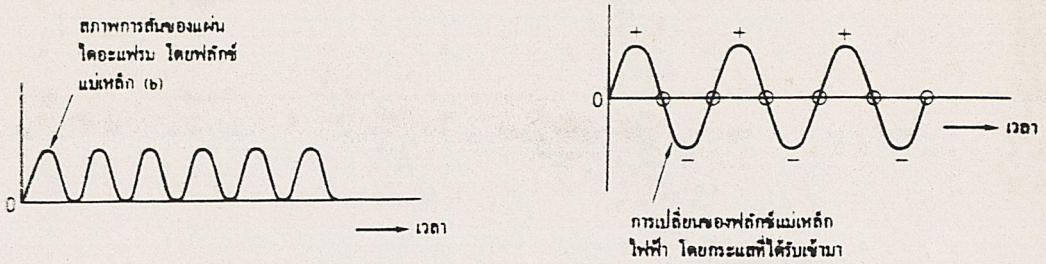
เมื่อคลื่นเสียงกระทบแผ่น DIAPHRAGM จะทำให้เกิดการสั่นสะเทือน (VIBRATE) จะทำให้ผงถ่านเปลี่ยนค่าความต้านทานตามความหนักเบาของคลื่นเสียง จะเป็นผลทำให้กระแสในวงจร OSCILLATE ตามคลื่นเสียงนั้นผ่านออกไปทาง INDUCTION COLL ออกไปตาม LINE ไปยังเครื่องรับ (RECEIVER)

1.3.2 RECEIVER

หลักการทำงานของหูฟังเป็นการเปลี่ยนพลังงานไฟฟ้าเป็นพลังงานเสียง เหมือนการทำงานของลำโพงสำหรับหูฟังที่ใช้ในเครื่องโทรศัพท์จะเป็นแบบกระทัดรัด



รูปที่ 1.4 หลักการทำงานของ RECEIVER

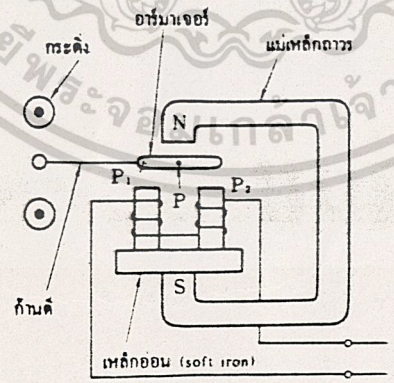


รูปที่ 1.5 แสดงรูคลื่นการทำงานของ RECEVER

เมื่อ SPEECH CURRENT ไหลผ่านขดลวดที่พันอยู่บนแท่งแม่เหล็กอ่อนซึ่งวางติดอยู่บนแม่เหล็กอ่อนซึ่งวางติดอยู่บนแม่เหล็กถาวรการมีอำนาจแม่เหล็กนี้จะกระทำต่อ DIAPHRAGM ตาม SPEECH CURRENT ที่ส่งมาจากด้านส่ง การสั่นสะเทือน (VIBRATE) นี้ จะออกมาในรูปคลื่นเสียง และการที่เสียงชัดเจนหรือไม่ขึ้นอยู่กับ MAGNETIC FORCE ของแม่เหล็กถาวร ขนาดและความหนาของแผ่น DIAPHRAGM และ AIR GAP ระหว่างแผ่น DIAPHRAGM กับแท่งเหล็กอ่อน ซึ่งห่างประมาณ Ø.3-Ø.4 ม.ม.

1.3.3 MAGNETO BELL.

กริ่งโทรศัพท์ที่ใช้ปัจจุบันได้พัฒนาให้เล็กกระทัดรัด และบางชนิดสามารถปรับเสียงได้ กริ่งโทรศัพท์ทำงานด้วยสัญญาณ A.C. (20 Hz)



รูปที่ 1.6 แสดงการทำงานของกระดิ่ง

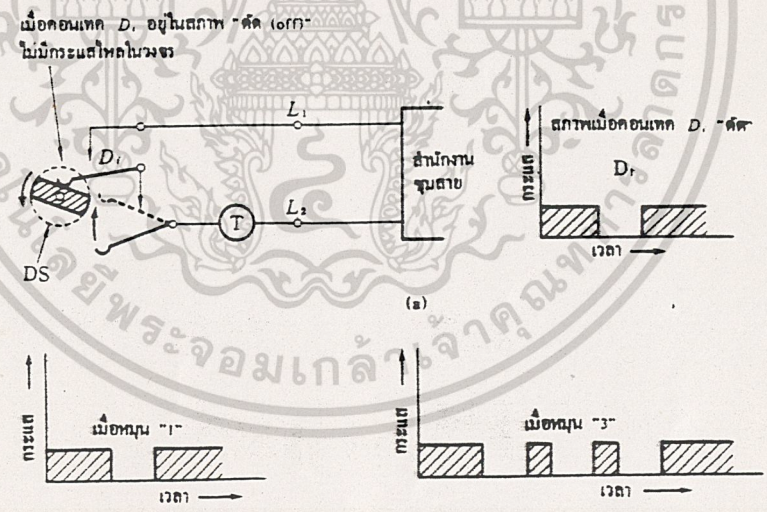
1.4 หน้าปัทม์โทรศัพท์ (DIAL)

1.4.1 หลักการโดยทั่วไปของหน้าปัทม์

หน้าปัทม์โทรศัพท์เป็นอุปกรณ์สำหรับตัดกระแสไฟในคู่สาย เพื่อเรียกหมายเลขที่ต้องการ เมื่อผู้เช่ายกหูฟัง เพื่อหมุนหมายเลขที่ต้องการจะทำให้เกิดครบวงจรด้วย HOOK SWITCH กระแสไฟตรง 48 VOLT สามารถไหลผ่านวงจรเครื่องโทรศัพท์ได้และผ่าน CONTACT DIAL IMPULSE จะทำหน้าที่ตัดและต่อกระแสไฟตรงที่ส่งมาจากชุมสาย

เมื่อผู้เช่าได้ยินสัญญาณ DIAL TONE เริ่มหมุนหน้าปัทม์ไม่ว่าจะหมุนด้วย DIAL IMPULSE CONTACT (Di) จะทำการตัดกระแสไฟจำนวนเท่านั้นครั้ง เพื่อส่งไปยังชุมสาย อุปกรณ์ภายในชุมสายก็จะทำการต่อไปยังเลขหมายที่ต้องการ

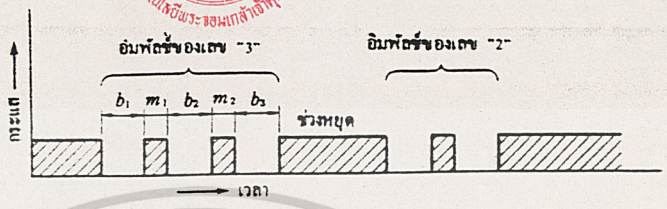
หน้าปัทม์โทรศัพท์โดยทั่วไปจะใช้ขดสปริงแบบลานนาฬิกา ทำหน้าที่ให้หน้าปัทม์หมุนกลับหรือบางแบบใช้ลวดสปริงขดเป็นรูปทรงกระบอก อุปกรณ์ที่นับว่าสำคัญคือ ลูกเบี้ยว (IMPULSECAM) PULSE จะเกิดขึ้นจากการทำงานของลูกเบี้ยว ดังแสดงในรูป 1.7



รูปที่ 1.7 แสดงการทำงานของลูกเบี้ยว

ความสัมพันธ์ระหว่างลูกเบี้ยวกับ คือเมื่อหมุนหน้าปัทม์ลูกเบี้ยวจะทำหน้าที่ดันให้ เปิดและปิดวงจรอยู่ตลอดเวลา นั่นคือไฟกระแสตรงที่เลี้ยงอยู่ในคู่สายนั้นจะถูกตัดให้ขาดเป็น

ห้วงฯตามเวลาที่กำหนด เช่น ถ้าหมุนหน้าปัทม์ด้วยเลข 3 Di จะเปิดและปิดวงจร 3 ครั้ง ดังแสดงในรูป 1.8



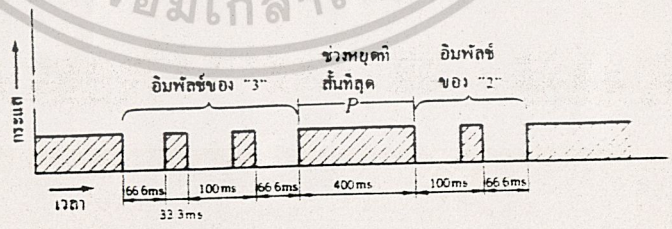
รูปที่ 1.8 แสดงการทำงานของ Di

1.4.2 IMPULSE SPEED

ความเร็วของ IMPULSE หมายถึง จำนวนครั้งของ CONTACT Di ตัดหรือต่อตามเวลาที่กำหนดมาตรฐานในหนึ่งวินาทีหน้าปัทม์จะหมุนกลับได้ด้วยสปริงแรงดึงสม่ำเสมอเวลาที่ใช้ในการหมุนกลับที่เดิมของแต่ละตัวเลขไม่เท่ากัน ความเร็วมาตรฐานจะกำหนดจากความเร็วเฉลี่ยของ IMPULSE โดยทั่ว ๆ ไปนั้น ความเร็วของหน้าปัทม์มีอยู่ 2 แบบ คือ 10 และ 20 พัลซ์ต่อวินาที

1.4.3 MAKE BREAK RATIL

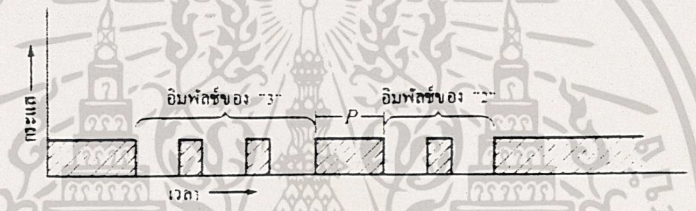
อัตราส่วนของการตัดและต่อของ CONTACT Di คืออัตราส่วนของเวลาที่ CONTACT Di ตัดและต่อวงจรกระแสจะไม่ไหลในขณะที่ Di ตัด และจะมีกระแสไหลในขณะที่ Di ต่อ อัตราที่ใช้เป็นมาตรฐานคือ 2 / 1



รูปที่ 1.9 แสดงการตัดต่อของ CONTACK Di

1.4.4 MINIMUM PAUSE

การหมุนหน้าปัทม์นั้น จะต้องหมุนตัวเลขทีละตัว เช่น ถ้าหากหมุนหมายเลข "3" และ "2" ตามลำดับจะเห็นได้ว่ามีช่วงระยะเวลาอยู่ช่วงหนึ่งระหว่างการหมุนกลับของหน้าปัทม์ ช่วงระยะเวลานั้นเรียกว่า MINIMUM PAUSE และช่วงระยะของการหยุดที่น้อยที่สุดนี้เองที่ขบวนของพัลซ์ (IMPULSE TRAIN) ถูกส่งไปยังชุมสาย MINIMUM PAUSE แบบเก่าจะประมาณ 600 mS แบบใหม่ประมาณ 400 mS และถ้าหากว่าช่วงนี้สั้นมากเกินไปจะทำให้อุปกรณ์สำหรับรับพัลซ์ไม่สามารถที่จะรับรู้ได้ว่าเป็นพัลซ์ของเลขอะไร ดังแสดงในรูป 1.10



รูปที่ 1.10 แสดงให้เห็นค่ามาตรฐานของพัลซ์ และช่วงระยะเวลาที่หยุดน้อยที่สุด (MINIMUM PAUSE)

1.4.5 การ BALANCING NETWORK (Z_n)

วงจรจะประกอบด้วยความต้านทาน 25 โอห์ม ต่ออนุกรมกับคาพาซิเตอร์ 1.24 ไมโครฟารัด และมีความต้านทาน 150 โอห์ม ต่อขนานกับสองตัวแรก เพื่อช่วยทำให้ด้าน SIDE TONE ดีขึ้น

1.4.6 การป้องกัน SIDE TONE หรือเสียงหอน

ความไวของ TRANSMITTER หรือ RECEIVER จะทำให้ได้เสียงออกมาไม่ชัดเจน เพราะมีเสียงหอน SIDE TONE หรือเสียงหอนดังเกินไป เมื่อผู้เช่าให้ LINE สั้น ดังนั้นแผ่นรองดังในรูปจึงถูกเพิ่มเข้าไปในวงจรโทรศัพท์ สำหรับเพิ่ม LINE LOSS อีก 3 dB

1.5.1 อุปกรณ์โทรศัพท์แบบปุ่ม (key telephone equipment)

อุปกรณ์โทรศัพท์แบบปุ่ม ต่อชุมสายโทรศัพท์ (telephone station) กับสายผู้เช่าได้หมด และระหว่างชุมสายด้วยกัน ด้วยการต่อแบบหลายสาย (multiple connection) และมีหน้าที่ 3 อย่าง ดังต่อไปนี้

1) ภายในชุมสายเราสามารถโทรศัพท์ถึงชุมสายอื่นภายในหรือภายนอกเขตด้วยการกดปุ่มต่อผ่านสายระหว่างชุมสาย (interstation line) หรือสายของผู้เช่า (subsriber's line)

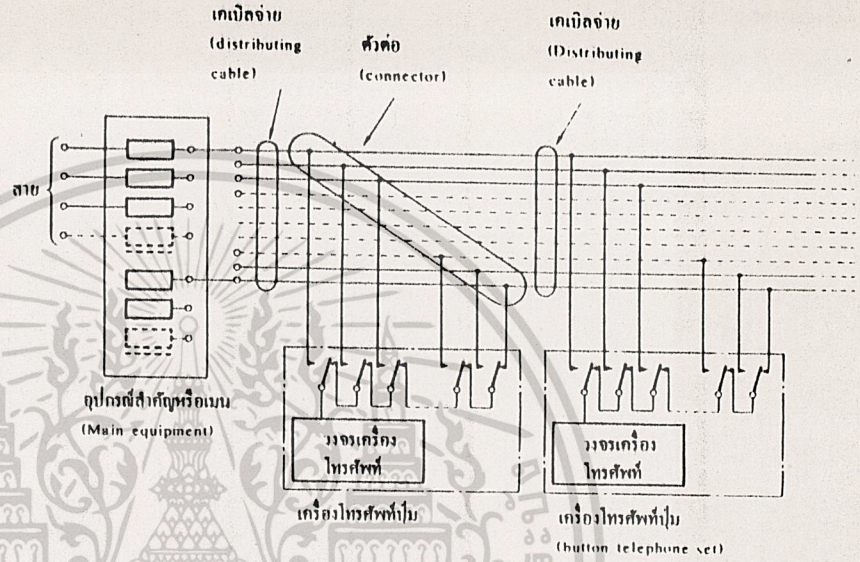
2) ภายในชุมสายที่ถูกเรียกผ่านสายผู้เช่า หรือสายระหว่างชุมสาย สามารถโทรศัพท์ถึงชุมสายอื่นภายในเขตหรือชุมสายนอกเขต ด้วยการรักษาให้สายอยู่ในสภาพที่ไม่ว่าง (busy)

3) ชุมสายภายในเขตทั้งหมด สามารถย้ายโทรศัพท์ที่ได้รับเข้า ให้กับชุมสายอื่นภายในเขตได้

ตัวอย่างเช่น ในการโทรศัพท์ต่อพ่วงแบบ P.B.X. สายโทรศัพท์อยู่รวมกันที่ตำแหน่งเดียว (คือที่แผงสวิช) เพื่อสับเปลี่ยนการเรียก โดยการทำงานของพนักงานองค์การโทรศัพท์ ส่วนในระบบโทรศัพท์แบบปุ่ม (key telephone) ผู้ใช้สับเปลี่ยนการเรียกด้วยตนเอง เพราะสายโทรศัพท์ทั้งหมดต่อไปยังเครื่องโทรศัพท์นั้น การต่อแบบหลายสาย (multiple connection) ไปยังเครื่องโทรศัพท์ อาจจะเสียค่าใช้จ่ายแพงมาก เมื่อจำนวนสายของผู้เช่าหรือจำนวน เครื่องโทรศัพท์มีมากดังนั้นขอบเขตของการใช้โทรศัพท์แบบปุ่มจึงจำกัด แต่สำนักงานที่ใช้เครื่องโทรศัพท์มากกว่า 5-6 และต่ำกว่า 30 เครื่อง นิยมใช้อุปกรณ์แบบนี้เพราะใช้งานง่ายกว่า P.B.X. และไม่ต้องการพนักงานโทรศัพท์

1) ชนิดต่าง ๆ ของระบบโทรศัพท์แบบปุ่ม

ระบบโทรศัพท์แบบปุ่ม ที่ติดตั้งโดยตรง คืออุปกรณ์โทรศัพท์แบบปุ่มเบอร์ 3 ซึ่งแยกออกเป็นชนิดต่าง ๆ ตามจำนวนสายของผู้เช่าและเครื่องพ่วงดังในตารางที่ 1.1



ตารางที่ 1.6.1 ชนิดต่าง ๆ ของระบบโทรศัพท์คีย์หรือปุ่ม (key or button telephone system)

การแยกชนิด / ชนิด	จำนวนเครื่องโทรศัพท์ที่ติดตั้งภายในสถานี (station)	อุปกรณ์หลักหรือเมน (main equipment)				เครื่องโทรศัพท์ (telephone set)				ระบบการเรียก (หน้าปัทม์)
		สายผู้เช่า (สายภายนอก) (subscriber line)		สายระหว่างสถานี (interstation line)		จำนวนปุ่ม (Number of Button)				
		จำนวนที่ใช้ได้	จำนวนติดตั้งจริง	จำนวนที่ใช้ได้	จำนวนติดตั้งจริง	สายผู้เช่า (สายภายนอก) (subscriber line)	สายระหว่างสถานี	รักษาสภาพต่อไว้ (hold)	เรียก (หน้าปัทม์)	
Type K206	6	2	2	1*	1	2	0	1	3	ปุ่ม
Type K410	10	5	4	1*	1	5	0*	1	5	"
Type K620	20	8	6	2*	1	8	2*	1	0	หมุนเลข

เครื่องหมาย *จำนวนที่ใช้ได้ของสายผู้เช่าหนึ่งชุด (ปุ่มกดของผู้เช่าอาจจะเปลี่ยนเป็นจำนวนที่ใช้ได้ของสายระหว่างสถานีหนึ่งชุด (ปุ่มกดของสายระหว่างสถานี) ได้

รูปที่ 1.11 ผังการต่อสายของอุปกรณ์โทรศัพท์ปุ่ม

2) **การทำงานของระบบโทรศัพท์แบบปุ่มเบอร์ 3**

ระบบนี้มีหลักการทำงานดังต่อไปนี้

1) ผู้ใช้สามารถเลือกได้โดยตรง (หมุนหมายเลข (dial) รับการเรียกเข้า และต่อเครื่องพ่วง) รักษาสภาพต่อ (hold) ไว้เพื่อสับเปลี่ยนสาย

2) ผู้ใช้สามารถเรียกระห่ [illegible] รใช้เครื่องโทรศัพท์แบบปุ่มผ่านการต่อภายใน (

3) สายที่ถูกเ [illegible] ต่อไว้ และสายที่ไม่ว่างจะมีหลอดไฟแสดงไว้สำ

4) ในกรณีของโ [illegible] ามารถเรียกเครื่องโทรศัพท์แบบปุ่มได้สองเครื่องเ

5) การเรียกโทรศั [illegible] ทำได้ด้วยการกดปุ่มสำหรับเครื่องชนิด K206 , [illegible] ยเลข (เลขจาก 0 ถึง 9) สำหรับเครื่องชนิด "

6) ในกรณีที่ระบบไฟฟ [illegible] ลขานุการและสัญญาณสายไม่ว่างจะไม่ทำงาน [illegible] บโทรศัพท์ทุกเครื่องได้โดยผ่านสายของผู้เช่า

7) การเรียกโทรศัพท์จะ [illegible] ที่กำลังเรียกโทรศัพท์ผ่านสายของผู้เช่า

8) ผู้ใช้สามารถตอบโทรศัพท์จากสายของผู้เช่าไ [illegible] ดเมื่อเขากำลังเรียกโทรศัพท์เลขานุการ (secretary telephone)

9) ความเร็วของหน้าปัทม์โทรศัพท์ (dial speed) ของเครื่องโทรศัพท์แบบปุ่มสามารถเปลี่ยนแปลง 10 pps หรือ 20 pps (pps = pulse per second)

1.5.2 เครื่องโทรศัพท์หน้าปัทม์แบบปุ่มกด (Push-button dial telephone set)

1) ระบบหน้าปัทม์แบบปุ่มกด

ในระบบสวิชชิ่ง S x S หรือ X.B. ที่ได้กล่าวมาแล้วสัญญาณเรียกของผู้เช่า (subscriber's address sognal) เป็นสัญญาณจังหวะไฟตรงที่เท่ากับจำนวนครั้งของการหมุนหน้าปัทม์เพื่อให้แผงสวิชทำงาน

จากการพัฒนาด้านแผงสวิชอีเลคตรอนิค เราจึงมีระบบสัญญาณหลายความถี่ (multi-frequency signal system) ขึ้น ระบบนี้มีลักษณะดังนี้

1) เวลาของการหมุนหมายเลขได้ลดลงมาก

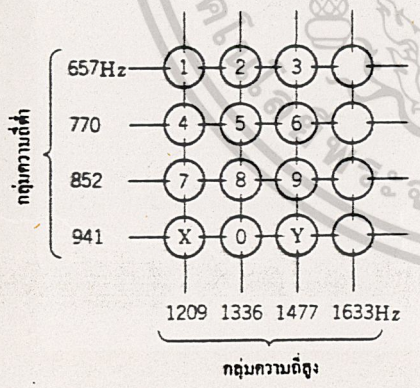
2) การหมุนหมายเลขง่ายลง

3) สามารถเพิ่มปุ่มกดอื่นนอกจากปุ่มหมายเลข เพื่อส่งสัญญาณบริการประเภทอื่นได้ด้วย

4) เราใช้สัญญาณความถี่ของเสียง (voice frequency signal) ซึ่งสา มารถส่งระหว่างสถานีได้ และสามารถนำไปใช้งานได้หลายอย่าง

2) ระบบสัญญาณ

ระบบนี้เรียกว่าระบบ 4 x 4 ในระบบนี้สัญญาณประกอบด้วย การรวมความถี่ 2 ค่าที่เลือกมาจากความถี่ 2 กลุ่ม (กลุ่มความถี่สูงและกลุ่มความถี่ต่ำ) ซึ่งในแต่ละกลุ่มมีความถี่ 4 ค่า และตรงกับตัวเลข (และโค๊ต) ดังแสดงในรูปที่ 1.12 ในระบบนี้ต้ำแหน่ง X,y และแถวตั้งที่สี่ (ตรงกับความถี่ 1,633Hz) สามารถใช้โค๊ตอื่นที่ไม่ใช่ตัวเลข สำหรับงานอื่นได้



รูปที่ 1.12 การจัดปุ่มและระบบสัญญาณ

ความถี่ (Hz)		กลุ่มความถี่สูง (K_2)			
		H_1 1,209 (11-15)	H_2 1,336 (11-14)	H_3 1,477 (11-13)	H_4 1,633
กลุ่มความถี่ต่ำ (K_1)	L_1 697 (1-5)	1	2	3	
	L_2 770 (1-4)	4	5	6	
	L_3 852 (1-3)	7	8	9	
	L_4 941 (1-2)		0		

ตารางที่ 2 ส่วนประกอบของสัญญาณและระบบปุ่ม

3) **เครื่องโทรศัพท์แบบ 600-P**

เครื่องโทรศัพท์หน้าปัทม์แบบปุ่มกดนี้ ขณะนี้ใช้อยู่ในซึ่งเหมือนกับเครื่องโทรศัพท์แบบ 600 ต่างกันแต่ที่หน้าปัทม์เท่านั้น คุณสมบัติของเสียงก็เหมือนกับของโทรศัพท์ชนิด 600

(a) **วงจรโทรศัพท์**

1) วงจรและการทำงานของสวิช

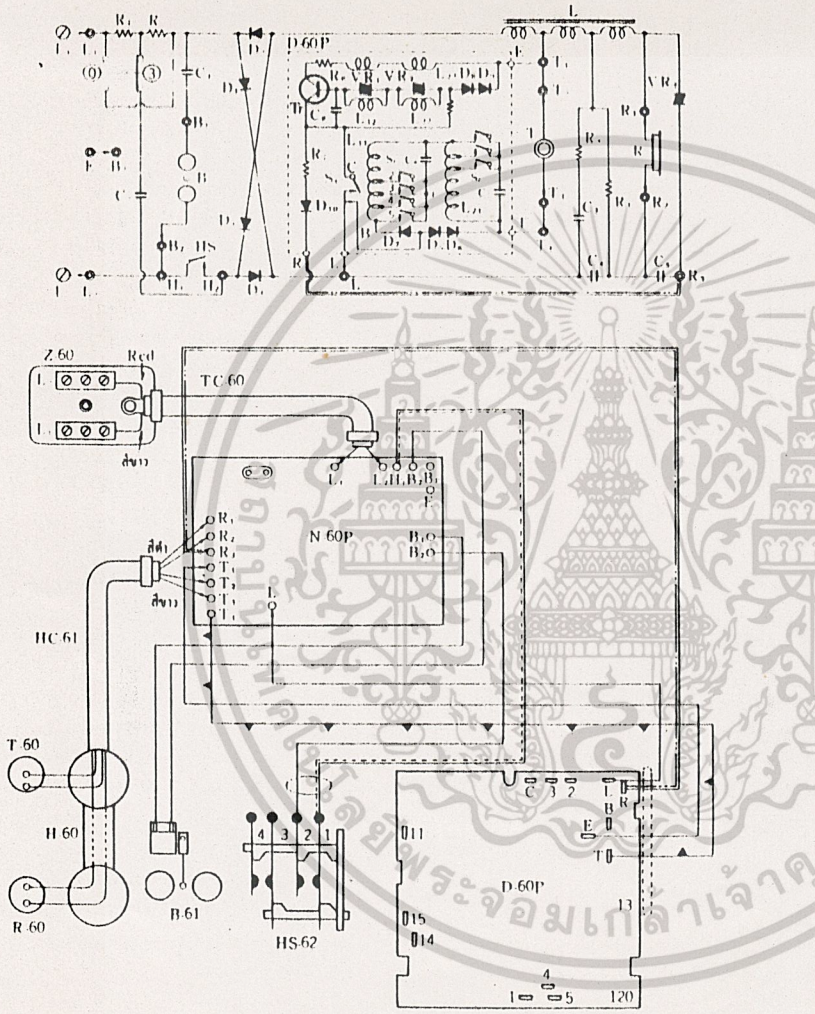
รูปที่ 1.13 แสดงวงจรของเครื่องโทรศัพท์ และส่วนที่อยู่ภายในเส้นปรุเป็นวงจรของหน้าปัทม์ Dial circuit) K_1 , K_2 และ K_3 เป็นสวิชที่ทำงานด้วยการกดปุ่มบนหน้าปัทม์และประกอบเข้าด้วยกัน ดังต่อไปนี้

การจัดปุ่มกดมีลักษณะดังแสดงในตารางที่ 2 สวิช 4 อันของ K_1 จัดไว้ตรงกับแถวตั้ง 4 แถวและสวิช 3 ตัวของ K_2 จัดให้ตรงกับแนวนอน 3 แถว ตัวอย่างเช่นเมื่อกดปุ่ม 1 คอนเทคระหว่าง 1-5 และระหว่าง 11-15 จะสัมผัส และเมื่อกดปุ่ม 0 คอนเทคระหว่าง 1-2 และระหว่าง 11-14 จะสัมผัสส่วน K_3 จะทำงานด้วยการกดปุ่มอื่น

เมื่อกดปุ่มให้ปุ่มเคลื่อนที่ประมาณ 1 มม. สวิช K_1 และ K_2 จะสัมผัสตามลำดับ (สัมผัสที่ละอันดังกล่าวข้างบน) และเมื่อเคลื่อนที่เป็นระยะทางประมาณ 1.7 มม. คอนเทคระหว่าง $C-P_3$ จะแยกจากกัน และเมื่อเคลื่อนที่ประมาณ 1.9 มม. คอนเทคระหว่าง $C-P_3$ จะแยกจากกัน และเมื่อเคลื่อนที่ประมาณ 2.7 มม. ปุ่มกดจะชนแผ่นรอง และกลับที่เดิม เมื่อเลิกกด

2) การทำงานของวงจา

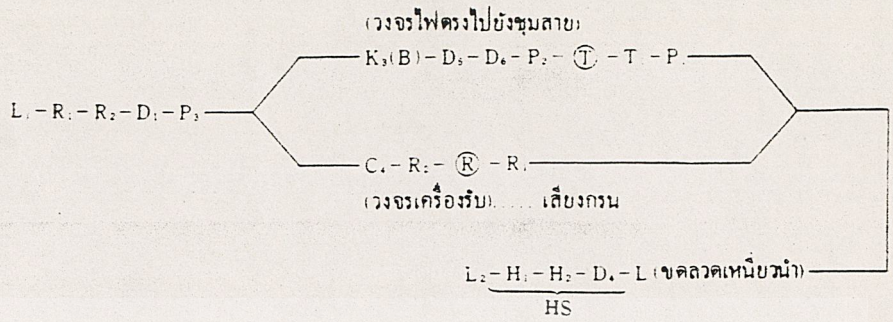
วงจรของโทรศัพท์แบบนี้ต่างกับของโทรศัพท์แบบ 600-A บ้าง คือ โทรศัพท์แบบนี้มีวงจรเรคติไฟเออร์คลื่นเต็ม (full-wave rectifier) (D_1-D_4) และมีส่วนประกอบของวงจรระหว่าง P_3-P_2 รวมอยู่ด้วยซึ่งวงจรเหล่านี้ไม่ทำให้คุณสมบัติของเสียงเปลี่ยนแปลง



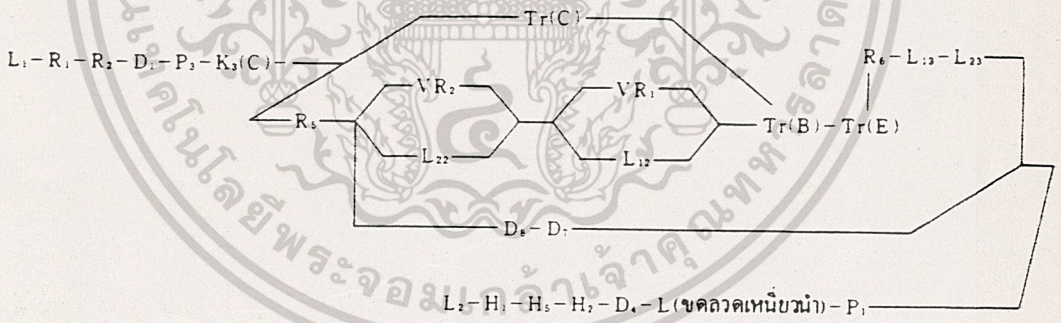
สัญญลักษณ์	ชื่อของชิ้นส่วน	สัญญลักษณ์	ชื่อของชิ้นส่วน
R_1	150Ω(¼W) ตัวต้านทาน	D_1	SR-1 ไดโอด
R_2	150Ω(¼W) -	D_2	-
R_3	25Ω(¼W) -	D_3	-
R_4	150Ω(¼W) -	D_4	-
R_5	1kΩ(¼W) -	D_5	-
R_6	62Ω(¼W) -	D_6	-
R_7	2kΩ(¼W) -	D_7	-
C_1	0.9μF คอนเดนเซอร์	D_8	-
C_2	0.084μF -	D_9	-
C_3	1.24μF -	D_{10}	-
C_4	2μF -	VR_1	VR-60 วาริสเตอร์
C_5	10μF -	VR_2	-
C_6	0.01μF -	VR_3	-
C_7	0.03μF -	Tr	ทรานซิสเตอร์
C_8	0.01μF -	K_1	คอนเทคของความถี่ (กลุ่มความถี่ต่ำ)
L_{11}	การจัดขดลวด	K_2	คอนเทคของความถี่ (กลุ่มความถี่สูง)
L_{12}		K_3	คอนเทคร่วม
L_{13}		T	เครื่องส่งสัญญาณ T-60
L_{14}		R	เครื่องรับสัญญาณ R-60
L_{15}	การจัดขดลวด	B	กระดิ่งแบตเน็ต B-61
L_{16}		HS	สวิทช์ขอเกี่ยว HS-62
L	ขดลวดเหนี่ยวนำ		

หมายเหตุ 1 T_1, T_2 สายสีขาว
R_1, R_2 สายสีดำ
2 B_1 โค้ดสีน้ำเงิน
B_2 โค้ดสีม่วง
3 H_1 สายบิดสีแดงเป็นเครื่องหมาย
4 R_1 สายบิดสีดำต่อเนื่องเป็นเครื่องหมาย
5 T_1 สายบิดสีดำขาวเป็นเครื่องหมาย
6 ขั้วต่อด้วยการเสียบปลั๊ก

รูปที่ 1.13 วงจรโทรศัพท์แบบ 600-P และผังการต่อสาย



เมื่อหมุนหมายเลข สวิช K_1 และ K_2 จะปิดด้วยการกดปุ่มเพื่อทำงานเป็นวงจรทูนนิ่งออสซิลเลเตอร์ (oscillator tuning circuit) ด้วย L_n , L_2 และ L_{21} , C_7 ดังได้กล่าวข้างบน การสัมผัสของ L_n และ L_{21} เปลี่ยนตามปุ่มที่ถูกกด การรวมความถี่ทูนนิ่ง (tuning fuquency) สองค่าก็จะเปลี่ยนไปด้วยในลำดับต่อไป เมื่อคอนเทคระหว่าง $C-P_3$ ถูกปิดกระแสไฟตรงจะไหลผ่านทรานซิสเตอร์ Tr ในวงจรข้างบน



ในวงจรทรานซิสเตอร์นี้มีขดลวด L_{12} และ L_{13} พันอยู่บนแกนวร่วมกัน และมีวงจรป้อนกลับอย่างบวก (positive feedback) จากขาอีมิตเตอร์ (emitter) ไปยังขาเบส (base) ขดลวด L_n ก็อยู่บนแกนเดียวกับข้างบน และกระแสไฟตรงผ่านไหล D_8-D_7 เนื่องด้วยความต้านทานมีค่าต่ำ L_n (รวมทั้ง L_{21}) จึงอยู่ในสภาพของการลัดวงจร ดังนั้นจึงไม่มีออสซิลเลชั่น (oscillation) เกิดขึ้น เนื่องจากมีการป้อนกลับเพียงเล็กน้อยแต่ เมื่อคอนเทคระหว่าง $B-P_3$ แยกจากกัน

มีกระแสไฟฟ้าตรงค่าสูงไหลผ่าน D_6-D_7 ทำให้ความต้านทานมีค่าสูงมาก ดังนั้นปลายทั้งสองของวงจรทูนนิ่ง (tuning circuit) จึงถือได้ว่าเป็นวงจรเปิด นั่นคือ Q ของวงจรทูนนิ่งมีค่าสูงและเกิดออสซิลเลชั่นขึ้นในวงจรนี้

ชั่วขณะก่อนที่คอนเทคระหว่าง B-P_3 จะแยกจากกัน กระแสไฟตรงไหลผ่าน L_{11} และ L_{12} ตามความแตกต่างของแรงเคลื่อนไฟฟ้า ที่เกิดจากความแตกต่างของแรงเคลื่อนตกใน D_5 , D_6 และใน D_9 เนื่องจากกระแสไฟตรงที่ผ่านคอนเทคเหล่านี้เมื่อคอนเทคระหว่าง B-P_3 แยกจากกันวงจรจะอยู่ในสภาพที่เกิดออสซิลเลชั่นขึ้นได้ กระแสไฟตรงนี้จะเปลี่ยนเป็นกระแสออสซิลเลติง (oscillating current) ในวงจรทรานซิสเตอร์ แต่การออสซิลเลชั่นมีขนาดจำกัด และถูกรักษาให้อยู่ที่ค่าหนึ่ง โดยวาริสเตอร์ VR_1 และ VR_2 ที่ต่อขนานอยู่กับ L_{12} และ L_{22}

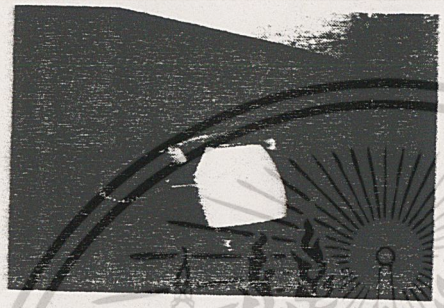
ดังนั้นสัญญาณที่ระดับค่าหนึ่ง จะถูกส่งออกขณะที่ผู้เรียกกดปุ่มเปิดคอนเทคระหว่าง B-P_3 จนถึงขณะที่ผู้เรียกปล่อยปุ่มเพื่อปิดคอนเทค D_8 , D_9 , R_5 และ R_6 เป็นส่วนหนึ่งของวงจรสำหรับกระแสไบอัส (bias current) ของทรานซิสเตอร์และ C_8 มีไว้เพื่อป้องกันการเกิดพาราซิติคออสซิลเลชั่น (parasitic oscillation)

ดังนั้นจะเห็นในคำอธิบายข้างบนว่า ถ้าคอนเทค 2 ตัวต่อในเวลาเดียวกันใน K_1 หรือ K_2 Q ของวงจรทูนนิ่ง จะไม่มีค่าสูงนัก ดังนั้นการออสซิลเลชั่นที่ความถี่ใด ๆ ของกลุ่ม จะไม่เกิดขึ้น ตัวอย่างเช่นเมื่อกดปุ่ม 1 และ 2 พร้อม ๆ กันจะมีออสซิลเลชั่นเกิดขึ้นที่ความถี่ 697 H_z เท่านั้น และจะไม่มีออสซิลเลชั่น เกิดขึ้นเมื่อกดปุ่ม 1 และ 5 พร้อม ๆ กัน เมื่อคอนเทคระหว่าง C-P_3 ของ K_3 แยกจากกันโดยการกดปุ่ม P_3-K_3-R_7-R_{10}-P_4-P_4-VR_3-L (ขดลวดเหนี่ยวนำ) เมื่อได้วงจรดังข้างบนส่วนหนึ่งของกระแสไฟตรงจะไหลผ่าน VR_3 ดังนั้นอิมพีดันซ์ของ VR_3 จะลดลงเพื่อเป็นวงจรเบี่ยง (by - pass circuit) และเสียงกรนหรือสัญญาณให้หมุน (dial tone) จากเครื่องรับจะถูกควบคุมให้มีระดับพอสมควร เมื่อไม่มีกระแสไฟตรงไหลผ่านวงจรอิมพีดันซ์ของ D_{10} จะมีค่าสูงมาก ซึ่งจะไม่มีผลกระทบกระเทือนต่อเสียงพูด (speech)

จุดประสงค์ตอนแรกของการต่อทรานซิสเตอร์ อย่างขนานกับเครื่องส่ง ก็เพื่อป้องกันไม่ให้เครื่องส่งถูกทำลาย ด้วยแรงเคลื่อนค่าสูง จากปรากฏการณ์ทรานเซี้ยน - (transient) เมื่อต่อทรานซิสเตอร์กับแหล่งจ่ายกระแสไฟตรงผ่านสายของผู้เช่า

(c) การตอบรับของผู้เรียก

เมื่อผู้ถูกเรียกตอบโทรศัพท์ กระแสที่จ่ายจะกลับทาง จึงเพิ่มเรคติไฟเออร์คลื่นเต็ม (full-wave rectifier) ที่ประกอบด้วยไดโอด D_1 , D_2 , D_3 และ D_4 เข้าในวงจรเพื่อป้องกันทรานซิสเตอร์จากกระแสกลับทิศ



รูปที่ 1.14 โทรศัพท์หน้าปัทม์แบบปุ่มกด (Push-button dial - telephone)

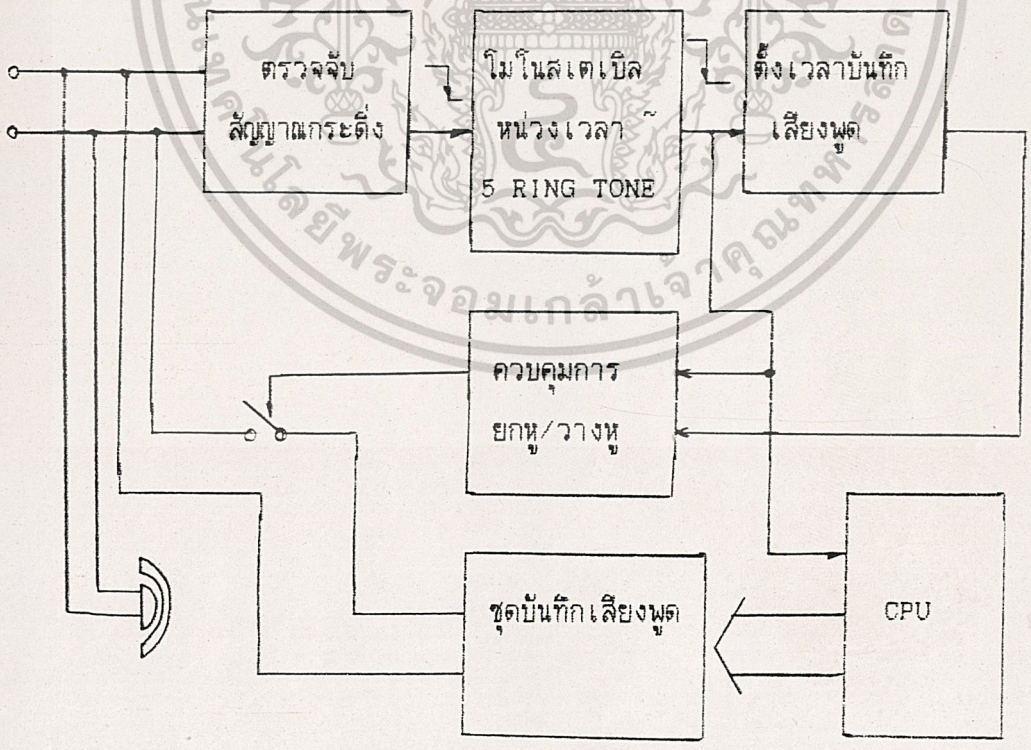
รูปที่ 1.15 ระบบโทรศัพท์คีย์ แบบ K-620

อุปกรณ์เมนและโทรศัพท์คีย์ชนิด K-620

การตรวจจับสัญญาณกระดิ่ง

หลักการเบื้องต้นของการสื่อสารข้อมูลทางโทรศัพท์ จะมีสัญญาณที่สำคัญคือ ในสภาวะสายว่าง คู่สายโทรศัพท์จะมีแรงดันประมาณ 48 โวลท์ซึ่งจ่ายมาจากชุมสายโทรศัพท์ เมื่อมีผู้เรียกเลขหมายเข้ามา ทางชุมสายจะจ่ายสัญญาณกระดิ่ง (RING TONE) เป็นแรงดันกระแสสลับ ซึ่งมีแรงดันประมาณ 100 V_{P-P} เป็นเวลา 2 วินาที และหยุดเป็นเวลา 4 วินาที เป็นจังหวะเช่นนี้ซ้ำไปเรื่อย ๆ ซึ่งแรงดันนี้จะทำให้สัญญาณกระดิ่งดังที่เครื่องรับ เมื่อมีการยกหูโทรศัพท์ขึ้นสวิทช์โทรศัพท์จะทำการต่อคู่สาย เข้ากับวงจร ทำให้แรงดันตกเหลือ ประมาณ 5-10 โวลท์

จากหลักการที่กล่าวมาแล้วเราสามารถนำมา สร้างวงจรตรวจสอบเรียกหมายโดยทำให้เกิดสภาวะความต้านทางต่ำขึ้นระหว่างคู่สาย ก็จะเสมือนการยกหูโทรศัพท์แล้วต่อส่วนถ่ายทอดสัญญาณ เข้ากับคู่สายโทรศัพท์ จากนั้นก็สามารถใช้คู่สายโทรศัพท์ถ่ายทอดข่าวสารกับชุดบันทึกเสียงพูดได้



รูปที่ 1.16 แสดงหลักการตรวจ RING TONE

จากรูปที่ 1.16 ถ้าเป็นบล๊อกไดอะแกรมข้อชุดตรวจสอบสัญญาณกระดิ่ง ซึ่งมีส่วนสำคัญ ดังนี้

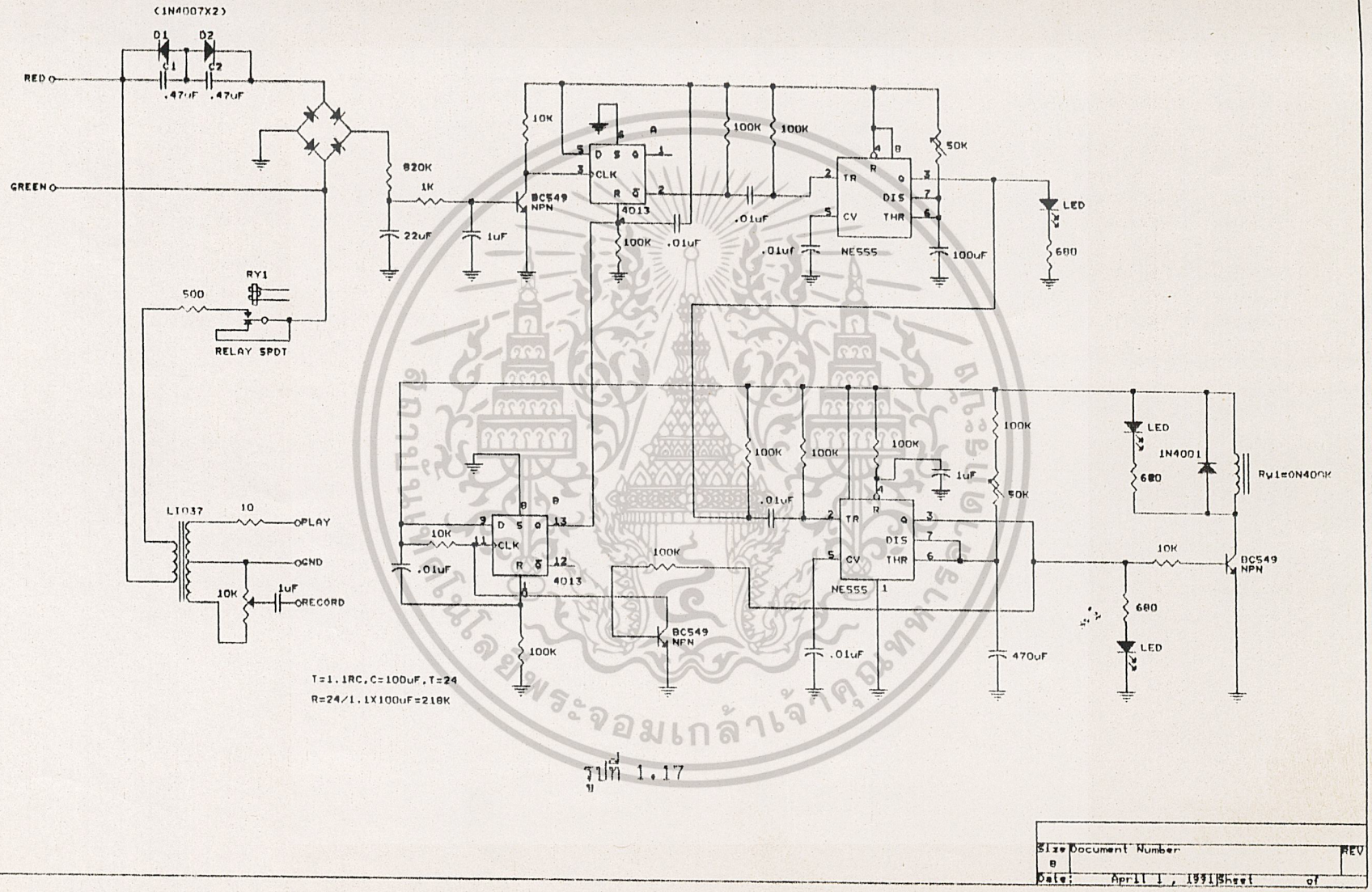
- ตัวตรวจจับสัญญากระดิ่ง ใช้ตรวจสอบว่ามีการเรียกใช้คู่สายหรือไม่ถ้ามีจะสร้าง พัลซ์ขึ้น
- ตัวหน่วงเวลา ใช้หน่วงเวลาเพื่อให้มีสัญญาณกระดิ่ง 5 ครั้ง แล้วจึงสั่งให้มีการยกหู ใช้วงจรโมโนสเตเบิลในการตั้งเวลา
- ตัวตั้งเวลาบันทึกเสียงพูด ใช้ตั้งเวลาประมาณ 1 นาที สำหรับส่งข้อความออกและบันทึกข้อมูลจากคู่สาย เข้าชุดบันทึกเสียงพูด
- ควบคุมการยกหู อาศัยรีเลย์ในการต่อคู่สายโทรศัพท์เข้ากับชุดบันทึกเสียงพูด

จากบล๊อกไดอะแกรมรูปที่ 1.16 ต่อไปจะเป็นวงจรตรวจสอบสัญญาณกระดิ่ง ดังรูปที่ 1.17 ซึ่งจะได้อธิบายดังต่อไปนี้

ตัวตรวจสอบสัญญาณกระดิ่ง

เป็นวงจรส่วนหน้า โดยที่ C_1,C_2 และ D_1,D_2 ต่อกันเพื่อเป็นตัวถ่ายทอดสัญญาณกระดิ่ง และยอมให้กระแสสลับผ่านเท่านั้น เมื่อสัญญาณกระดิ่งเข้ามา จะถูกเรคติฟรายโดย D_3-D_6 ซึ่งต่อเป็นวงจรบริดจ์เรคติฟายเจอร์สัญญาณกระดิ่ง 16 H_z เออโวลท์ จะถูกเรคติฟายเป็นสัญญาณพัลซ์ 32 H_z 100 V จะเห็นว่าแรงดันสูงมากจึงต้องลดทอนด้วยความต้านทาน R_1 ให้มีแรงดัน เหมาะสม

ตัวเก็บประจุ C_3,C_4 และตัวต้านทาน R_2 ต่อเข้าด้วยกันเป็นวงจร LOW - PASS FILTOR ให้แรงดันขาเบส ของทรานซิสเตอร์ Q_1 ค่อย ๆ สูงเพิ่มขึ้นเมื่อเพิ่มถึง 0.7 โวลท์ ทรานซิสเตอร์ Q_1 และจะนำกระแส สร้างสัญญาณนาฬิกาที่เป็นขอบขาลง จึงยังไม่มีผลต่อ $IC_{1/1}$ เนื่องจาก $IC_{1/1}$ ทำงานเมื่อ CK เป็นขอบขาขึ้นเมื่อสัญญาณกระดิ่งเงียบ $IC_{1/1}$ จะทำงาน ทำให้ที่ขา Q เป็นลอจิกเปลี่ยนจาก ไลจิก "0"ไปเป็น "1" ไปเป็นสัญญาณทริกให้กับ IC_2 ตัวเก็บประ C_5 และตัวต้านทาน R_4 ต่อเป็นวงจรรีเซ็ท ในช่วงเปิดเครื่องครั้งแรก

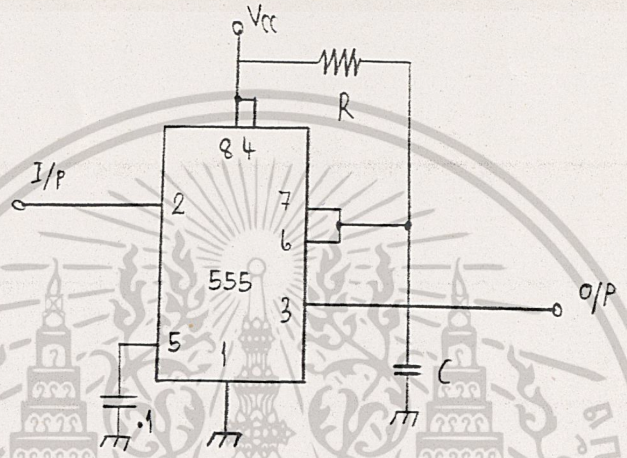


รูปที่ 1.17

Size	Document Number	REV
B		
Date:	April 1 , 1991 Sheet	of

ตัวหน่วงเวลา

ตัวหน่วงเวลาใช้หน่วงเวลาให้มีสัญญาณกระดิ่งดัง 5 ครั้ง โดยใช้วงจรโมโนสเตเบิล ไอซีเบอร์ 555 เมื่อได้สัญญาณทริกเกอร์จาก $IC_{1/1}$ ก็จะทำให้ลอจิกที่ขา 3 เปลี่ยนจาก "0" ไปเป็น "1" ตามเวลาที่กำหนดด้วยค่า RC ที่ขา 6,7



รูปที่ 1.18 วงจรโมโนสเตเบิล

ในที่นี้เราต้องการ หน่วงเวลาสัญญาณกระดิ่ง 5 ครั้ง แต่ละครั้งนาน 6 sec. จะได้ T=30 sec. เลือกใช้ C=100 MF และช่วงเวลา T คำนวณได้จากสูตร

$$T = 1.1 \times R \times C$$

โดยที่ T มีหน่วยเป็นวินาที

R มีหน่วยเป็นโอห์ม

C มีหน่วยเป็นฟาริด

จากสมการเราจะได้ค่า R = 330 K

ตัวตั้งเวลาบันทึกเสียงพูด

ตัวตั้งเวลาบันทึกเสียงพูด ใช้ตั้งเวลาในการส่งข้อความออก และเวลาในการบันทึกข้อมูลโดยใช้เวลาตามชุดบันทึกเสียงพูดคือ 93 sec. และไอซีที่ใช้คือเบอร์ 555 IC_3 จะทำงานเมื่อได้รับสัญญาณทริกขอบขาลงจากตัวหน่วงเวลาตัวแรก เอ้าพุทของไอซีนี้จะนำไปเป็นแรงดันไบอัสให้กับทรานซิสเตอร์ Q_2 R_{15} และC_{10} ต่อเป็นวงจร RESET ให้กับ IC_3 เป็นการ RESET ให้กับ IC_3 ในช่วงเปิดเครื่องครั้งแรก.

ตัวควบคุมการยกหู

ตัวควบคุมการยกหูประกอบด้วย Q_2,$RELAY_1$,T_1 เมื่อขา 2 ได้รับสัญญาณทริกจาก IC_2 ทำให้ขาเอ้าพุทขา 3 เป็นไลจิก "1" เป็นแรงดันไปอัสให้กับขาเบสของทรานซิสเตอร์ Q_2 รีเลย์จะต่อคู่สายโทรศัพท์เข้ากับความต้านทาน 500 โอหม์ และทรานฟอร์เมอร์ ทำให้แรงดันของคู่สายโทรศัพท์ลดลงเหลือประมาณ 5-10 โวลท์ เป็นการยกหูโทรศัพท์ สัญญาณเสียงที่มากับคู่สายจะดันสปิงผ่านทรานฟอร์เบอร์ T_1 เพื่อนำไปเข้าชุดบันทึกเสียงพูด ทรานซิสเตอร์จะนำกระแสอยู่นานประมาณ 90 วินาทีโดยตั้งเวลาด้วยค่า RC เมื่อครบเวลากำหนด รีเลย์จะหยุดทำงานตัดคู่สายโทรศัพท์ออกจาก ทรานฟอร์เมอร์

บทที่ 2

การถอดรหัสความถี่โทรศัพท์ชนิดกดปุ่ม (DTMF)

ในปัจจุบันนี้โทรศัพท์มีส่วนสำคัญกับชีวิตประจำวันของเราอย่างมากมาย และนับวันยิ่งทวีความสำคัญเพิ่มมากขึ้นทุกที และอุตสาหกรรมด้านโทรศัพท์สื่อสารก็ขยายตัวขึ้นอย่างรวดเร็ว ดังนั้นเราจึงน่าจะศึกษาเอาไว้ เพื่อจะได้นำมาใช้ประโยชน์ได้ตามต้องการ

ก่อนอื่นขอให้ความหมายของคำว่า **ถอดรหัสความถี่โทรศัพท์** อันหมายถึงแปลงสัญญาณความถี่ซึ่งเกิดจากการกดปุ่มตัวเลขของโทรศัพท์ชนิดกดปุ่ม (ชนิด Tone หรือ DTMF) ให้เป็นระบบตัวเลขทางดิจิตอล ซึ่งไอซี MT8870 ใช้แปลงความถี่โทรศัพท์ให้เป็นตัวเลขฐานสองขนาด 4 บิต

ในยุคก่อน การออกแบบวงจรถอดรหัสความถี่ของโทรศัพท์ มักใช้ไอซีจำพวกสล็อกลูปซึ่งสร้างปัญหาสารพัด ไม่ว่าเรื่องของความถี่ที่เปลี่ยนแปลงไป การปรับแต่งวงจรขนาดของวงจรที่ใหญ่ เพราะต้องใช้ไอซีจำนวนมาก

คุณสมบัติของ MT8870

เป็นตัวรับและถอดรหัสความถี่ (DTMF receiver)
กินไฟน้อย ใช้ไฟเลี้ยงระดับเดียวกันกับ TTL
สามารถตั้งอัตราขยายภายในตัวไอซีได้
สามารถปรับการ์ดไทม์ (Guard time) ได้
เป็นไอซีคุณภาพสูง

การนำ MT8870 ไปใช้งาน

นำไปใช้ในงานด้านรีโมตคอนโทรล
เครื่องป้องกันโทรศัพท์ทางไกล
ใช้ในงานเกี่ยวกับเครดิตการ์ด
ใช้งานร่วมกับคอมพิวเตอร์

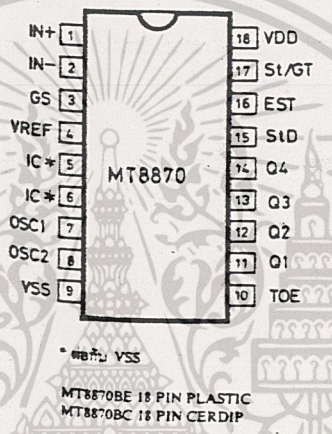
ใช้ในเครื่องชุมสายขนาดย่อย หรือ PABX

ใช้กับงานทางด้านโทรศัพท์ทั่วไป

เครื่องกันขโมย

การควบคุมอุปกรณ์ทางโทรศัพท์

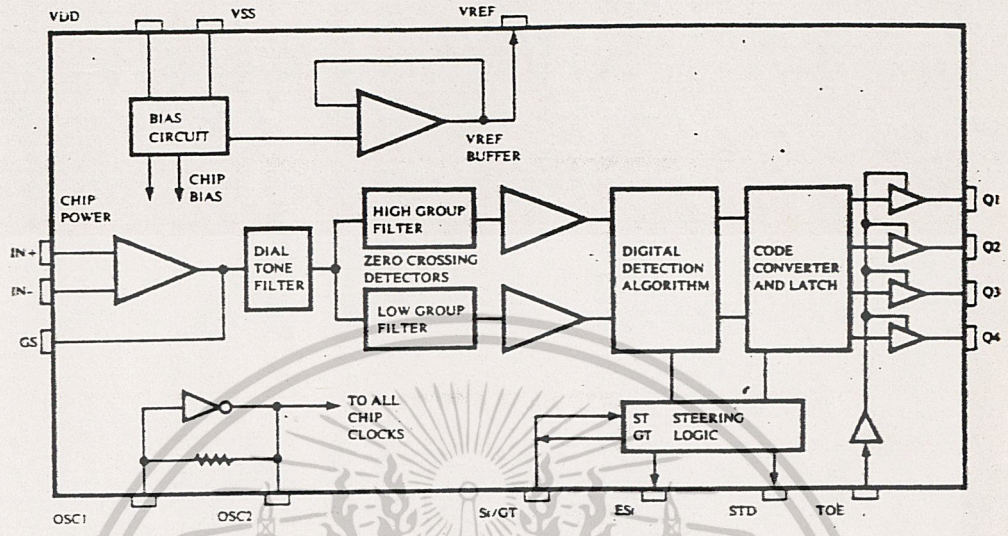
ใช้ทำเครื่องสอบถามทางโทรศัพท์



รูปที่ 2.1 แสดงรายละเอียดขาของ MT8870

โครงสร้างของ MT8870

โครงสร้างภายในของ MT8870 ประกอบด้วยวงจรกรองความถี่และวงจรถอดรหัสฟังก์ชันทางด้านดิจิตอล เป็นไอซีที่สร้างโดยใช้เทคโนโลยี ISO^2 - CMOS ในส่วนของวงจรกรองความถี่ใช้เทคนิคของสวิตช์คาปาซิเตอร์ฟิลเตอร์สำหรับกรองความถี่สูง และต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอล เพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ ออกเป็นเลขฐานสองขนาด 4 บิต และเช็คช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุตเป็นออปแอมป์ ซึ่งสามารถปรับอัตราขยายได้โดยต่ออุปกรณ์ภายนอกเอาต์พุตเป็น วงจรแลตช์ 3 สถานะ รูปที่ 2.1 แสดงขาของ MT8870 และรูปที่ 2.2 แสดงโครงสร้างภายในของ MT8870

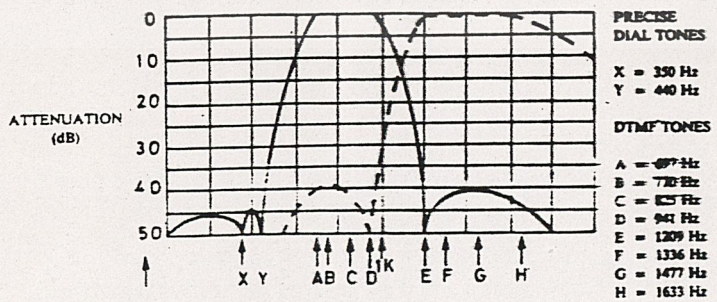


รูปที่ 2.2 แสดงโครงสร้างภายในของ MT8870

ฟังก์ชันการทำงานภายใน MT8870

ภายใน MT8870 ประกอบด้วยส่วนสำคัญ 5 ส่วน คือ

- ภาคกรองความถี่ (filter section)
- ภาคถอดรหัส (decoder section)
- ภาคตรวจสอบสัญญาณ (steering circuit)
- ภาคขยายสัญญาณความแตกต่าง (diferential input)
- ภาคกำเนินความถี่ (oscillator



รูปที่ 2.3 แสดงความถี่ที่ได้จากภาคกรองความถี่

ภาคกรองสัญญาณความถี่

ในส่วนนี้จะแยกสัญญาณ DTMF ที่เข้ามาออกเป็น 2 กลุ่มความถี่คือ ช่วงความถี่สูง และความถี่ต่ำ โดยใช้วงจรแถบความถี่อันดับ 6 ชนิดสวิตช์คาปาซิเตอร์ (six- order switched capacitor band pass filter) ซึ่งความถี่ที่แยกได้มี 2 ช่วง คือ ช่วงความถี่สูงและช่วงความถี่ต่ำ

F_{LOW}	F_{HIGH}	NO	TOE	Q_4	Q_3	Q_2	Q_1
697	1209	1	H	Ø	Ø	Ø	1
697	1336	2	H	Ø	Ø	1	Ø
697	1477	3	H	Ø	Ø	1	1
770	1209	4	H	Ø	1	Ø	Ø
770	1336	5	H	Ø	1	Ø	1
770	1477	6	H	Ø	1	1	Ø
852	1209	7	H	Ø	1	1	1
852	1336	8	H	1	Ø	Ø	Ø
852	1477	9	H	1	Ø	Ø	1
941	1336	Ø	H	1	Ø	1	Ø
941	1209	*	H	1	Ø	1	1
941	1477	#	H	1	1	Ø	Ø
697	1633	A	H	1	1	Ø	1
770	1633	B	H	1	1	1	Ø
852	1633	C	H	1	1	1	1
941	1633	D	H	Ø	Ø	Ø	Ø
-	-	ANY	L	Z	Z	Z	Z

รูปที่ 2.4 แสดงค่าที่ถอดรหัสได้จากความถี่ต่าง ๆ

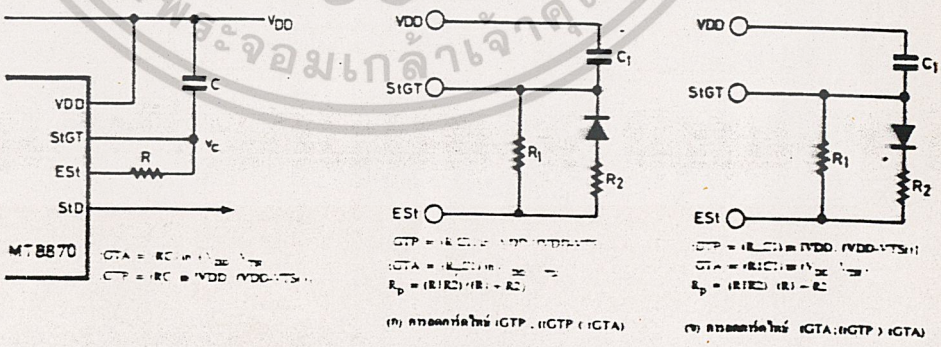
ภาคถอดรหัส

ความถี่ DTMF ที่ถูกกรองเรียบร้อยแล้วจะผ่านเข้าวงจรถอดรหัสความถี่ออกเป็นตัวเลข โดยใช้เทคนิคการนับแบบดิจิตอล และมีการตรวจสอบความถี่ที่เข้ามาว่าเป็นความถี่มาตรฐาน DTMF หรือไม่ เพื่อป้องกันความถี่อื่นเข้ามาผสม

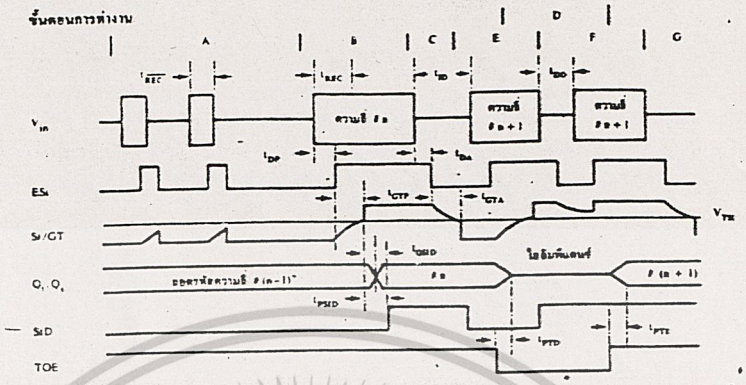
เมื่อตรวจสอบว่าความถี่นั้นถูกต้อง สัญญาณที่ขา ESt (early steering) ก็จะแอคทีฟ สำหรับค่าที่ถอดรหัสได้จากความถี่ต่าง ๆ นั้น แสดงในรูปที่ 2.4

ภาคตรวจสอบสัญญาณ

ก่อนที่จะมีการถอดรหัสความถี่ออกไปที่เอาต์พุต จะมีการตรวจสอบช่วงความถี่ที่เข้ามามีระยะเวลาตามที่กำหนดหรือไม่ โดยสังเกตุจากระยะเวลาการกดปุ่มโทรศัพท์ ซึ่งต้องกดปุ่มให้มีความถี่ออกมาเป็นช่วงเวลาพอสมควร มิฉะนั้นวงจรส่วนนี้จะไม่รับ โดยถือว่าสัญญาณนั้นไม่ถูกต้อง ส่วนช่วงเวลายาวเท่าใดสามารถตั้งได้โดยใช้ RC ต่อภายนอก สัญญาณที่ขา ESt จะเป็น "High" นานใกล้เคียงกับระยะเวลาที่มีความถี่ DTMF เข้ามา จากรูปที่ 2.5 เมื่อขา ESt เป็น "High" ทำให้ V_c สูงขึ้น ตัวเก็บประจุ C จะคายประจุทำให้แรงดัน V_c สูงขึ้นจนถึงค่าเทรชโฮลด์ วงจรถอดรหัสจึงจะถอดรหัสออกเป็นตัวเลขขนาด 4 บิต รายละเอียดการทำงานของให้ดูจากแผนภูมิเวลาหรือไทมิ่งไดอะแกรม (timing diegram) ในรูปที่ 2.8 จะเข้าใจได้ง่ายกว่า



รูปที่ 2.5 แสดงวงจรตรวจสอบสัญญาณอย่างง่ายและการแสดงการกำหนดเวลาการดีไบน์ (gard time) พร้อมวิธีคำนวณ



รูปที่ 2.9 แสดงแผนภูมิเวลา (timing diagram) ของ MT8870

อธิบายขั้นตอนการทำงาน

A - ตรวจสอบความถี่เข้ามา แต่คาบเวลาไม่ถูกต้อง เอาต์พุตไม่เปลี่ยน

B - ความถี่ #n ถูกตรวจพบและมีคาบเวลาที่ถูกต้อง ความถี่ถูกถอดรหัส และแลตช์ไว้ที่เอาต์พุต

C - จบความถี่ #n ช่วงห่างถูกต้อง เอาต์พุตยังคงแลตช์อยู่จนกว่าจะได้รับความถี่ที่ถูกต้องใหม่

D - เอาต์พุตเปลี่ยนเป็นไอโอมีดีแคนซ์

E - ความถี่ #n + 1 ถูกตรวจพบคาบเวลาถูกต้อง ความถี่ถูกถอดรหัสและแลตช์ไว้

F - ความถี่ #n + 1 หายไป ช่วงห่างไม่ถูกต้อง เอาต์พุตยังคงแลตช์อยู่

C - จบความถี่ #n + 1 ช่วงห่างถูกต้อง เอาต์พุตยังแลตช์อยู่จนถึงความถี่ใหม่ที่ถูกต้อง

อธิบายคำศัพท์

V_{in} - สัญญาณความถี่ DTMF ที่เข้ามา

ESt - Early Sterring output ให้แสดงความถี่ที่ถูกต้อง

St/GT - Steering input/Guard Time output สำหรับต่อกับ RC ภายนอก

Q_1-Q_4 - เอาต์พุต BCD ขนาด 4 บิต

StD - Delayed Steering output ใช้แสดงว่าความถี่ที่ได้รับหรือหายไป มีคาบเวลาตามที่กำหนด เพื่อแสดงความถูกต้องของสัญญาณ

TOE - Tone Output Enable (input) ใช้ควบคุม Q_1-Q_4 ให้เป็นไฮอิมพีแดนซ์

$t_{\_\_}$ - คาบเวลานานสุดที่ตรวจพบความถี่ DTMF แล้วยังไม่ถูกต้อง
$_{REC}$

t_{REC} - คาบเวลาสั้นสุดที่ต้องการเพื่อแสดงสัญญาณถูกต้อง

t_{ID} - เวลาสั้นสุดระหว่างสัญญาณ DTMF ที่ถูกต้อง 2 สัญญาณ

t_{DO} - เวลานานสุดที่ยอมให้สัญญาณหายไปได้ในคาบเวลาความถี่ที่ถูกต้อง

t_{DP} - เวลาที่ใช้ในการตรวจพบสัญญาณความถี่ DTMF ที่ถูกต้อง

t_{DA} - เวลาที่ใช้ในการตรวจการหายไปของสัญญาณความถี่ DTMF ที่ถูกต้อง

t_{DIP} - การ์ดไทม์ของการปรากฏความถี่ DTMF

t_{GTA} - การ์ดไทม์ของการหายไปของความถี่ DTMF

สำหรับคำว่าการ์ดไทม์ (gard time) นั้นหมายถึง ช่วงคาบเวลาของความถี่ที่เข้ามา ซึ่งจะต้องนานเท่ากับหรือมากกว่าช่วงเวลาที่เราตั้งไว้ จึงจะได้รับการยอมรับว่าสัญญาณความถี่นั้นถูกต้อง หรือพูดได้ว่าเวลาที่เราตั้งไว้โดย RC ก็คือ การ์ดไทม์นั้นเอง เมื่อสัญญาณความถี่เข้ามานานเท่าหรือมากกว่าเวลาที่ตั้งไว้จึงสามารถแปลงเป็นตัวเลขได้ ถ้าสัญญาณความถี่เข้ามาสั้นกว่าก็จะไม่มีการถอดรหัสเป็นตัวเลขออกไป การตั้งเวลาและคำนวณเวลาดูได้จากรูปที่ 2.5

ภาคขยายสัญญาณความแตกต่าง

วงจรส่วนอินพุตของ MT8870 เป็นภาคขยายออปแอมป์ที่สามารถปรับอัตราขยายโดยต่อวงจรภายนอกเพิ่มเข้าไปรูปที่ 2.6 แสดงการต่อวงจรภายนอกเข้ากับอินพุตซึ่งสามารถคำนวณอัตราการขยายความแตกต่างของอินพุตและอิมพีแดนซ์ได้ ดังนี้

อัตราขยาย (A_v diff) = R_s/R_i

อินพุตอิมพีแดนซ์ (Z_{in} diff)

$$= 2\sqrt{R_i^2 + (1/wc)^2}$$

ภาคกำเนิดความถี่

ในภาคนี้ภายในไอซีจะมีวงจรเวลาอยู่ภายใน เพียงแต่ต่อคริสตอลขนาด 3.58 MHz ก็สามารถใช้งานได้ทันที การต่อวงจรกำเนิดความถี่แสดงในรูปที่ 2.7

การประยุกต์ใช้งานการถอดรหัสความถี่โทรศัพท์

จากหลักการเบื้องต้นของ MT 8870 ไอซี ถอดรหัสสัญญาณโทรศัพท์ เราสามารถนำมาประยุกต์ใช้งาน โดยนำมาสร้างการควบคุมอุปกรณ์ทางไฟฟ้าโดยผ่านทางคู่สายโทรศัพท์ เช่น เราต้องการเปิดแอร์ภายในห้องนอนให้เย็นไว้ก่อนที่เราจะกลับถึงบ้าน หรืออาจจะสั่งให้เปิดไฟหน้าบ้าน

จากรูปที่ 2.4 จะเห็นว่าที่เอ้าพุท Q_1 ถึง Q_4 ของ MT 8870 จะให้โลจิกไบนารีตามการกดเลขหมายโทรศัพท์ เอ้าพุทที่ได้เราจะนำไปทำการดีโค้ดเดอร์แล้วนำไปควบคุม ฟลิมฟลอม ดังจะกล่าวต่อไป

การนำการถอดระหัสความถี่โทรศัพท์มาใช้งาน

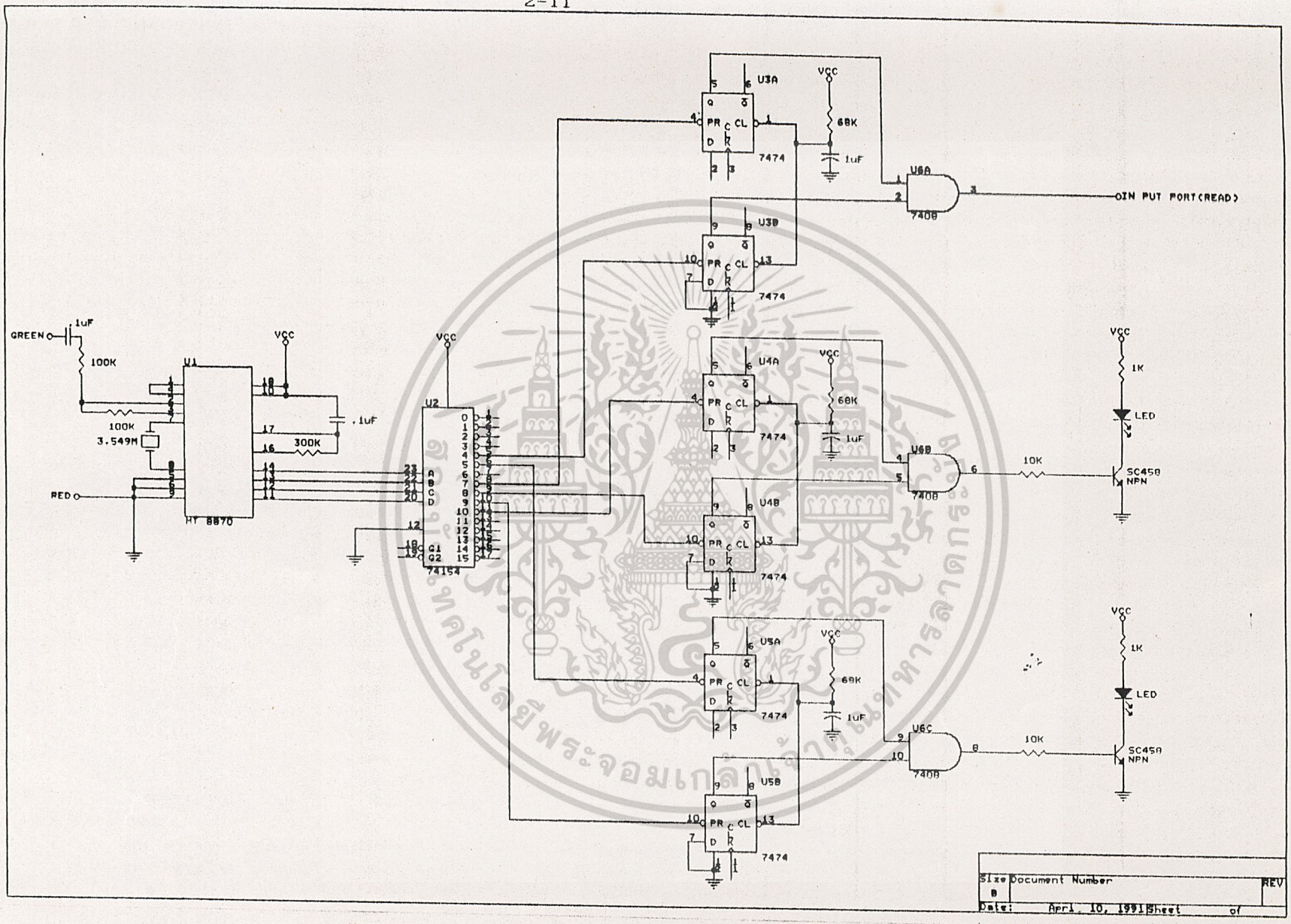
จากวงจรเอ้าพุทจาก MT 8870 ซึ่งเป็น BCD ไบนารี่โค้ด จะถูกต่อเข้ากับอินพุทของ 74LS154 ซึ่งเป็นตัวดีโค้ดเดอร์ 4 TO 16 การทำงานของ 74LS154 แสดงดังรูปที่ 2.11

FUNCTION TABLE

INPUTS						OUTPUTS															
E_0	E_1	A_3	A_2	A_1	A_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH voltage level
L = LOW voltage level
X = Don't care

รูปที่ 2.11 ตารางความจริงของ 74LS154



รูปที่ 2.10 วงจรการควบคุมอุปกรณ์จากโทรศัพท์

2-11

ดังนั้นเมื่อเรากดคีย์โทรศัพท์หมายเลขใดหมายเลขหนึ่ง ก็จะทำให้เอ้าพุท ของ 74LS154 เป็น LOW เพียงขาเดียว เช่นถ้าเรากดคีย์หมายเลข 1 จะทำให้ขา 2 ของ 74LS154 เป็น LOW ขาอื่นจากเป็น HIGH หมด เอ้าพุทจากการดีโค๊ดนี้เราจะนำไปต่อเข้ากับขา PRESET ของ ดี-ฟลิมฟลอบ 74LS74 จะทำให้ที่ขาเอ้าพุท Q ของ ดี-ฟลิบฟลอมเป็น HIGH

ที่อินพุทของ 74LSØ8 ซึ่งเป็นแอนด์เกท 2 อินพุท จะถูกต้องเข้ากับเอ้าพุท Q ของดีโค๊ดเดอร์ ดังนั้นเอ้าพุทของ 78LSØ8 จะเป็น HIGH ได่จะต้องมีการกดคีย์โทรศัพท์ 2 หมายเลข ทั้งนี้เพื่อป้องกันเหตุบังเอิญจากการที่ผู้ใช้ไปกดหมายเลขใดหมายเลขหนึ่งเข้า และเอ้าพุทจาก 74LSØ8 นี่จะนำไปใช้ควบคุมอุปกรณ์ไฟฟ้าต้องไป.

เมื่อเราต้องการจะปิดอุปกรณ์ไฟฟ้าที่เราสั่งให้เปิดไปแล้วก็สามารถทำได้ โดยกดหมายเลขโทรศัพท์ที่เรากำหนดไว้ในที่นี้เป็นหมายเลข Ø จะถูกดีโค๊ดเดอร์ที่ขา 1 เป็นโลจิก LOW จะไปทำการเคลียร์ ฟลิบฟลอบทำให้อินพุทของแอนด์เป็น LOW ทำให้เอ้าพุทเป็น LOW ด้วย

รูปที่ 2.12 เป็นหมายเลขคีย์และหน้าที่การใช้งาน

KEY	FUNTION
4,7	เปิดอุปกรณ์ไฟฟ้า 1
5,8	เปิดอุปกรณ์ไฟฟ้า 2
Ø.	ปิดอุปกรณ์ไฟฟ้า 1
1.	ปิดอุปกรณ์ไฟฟ้า 2

ที่ขา CLEAR ของ ดี-ฟลิปฟลอม จะมีความต้าน 68 กิโลโอหม์ต่ออยู่กับคาปาซิเตอร์ 1 ไมโครฟาริดต่อลงกราวด์ เพื่อจะทำการ CLEAR ให้เอ้าพุท Q ของดี-ฟลิปฟลอมเป็นโลจิก LOW ทุกครั้งที่เปิดเครื่อง

บทที่ 3

การวิเคราะห์เสียงพูด

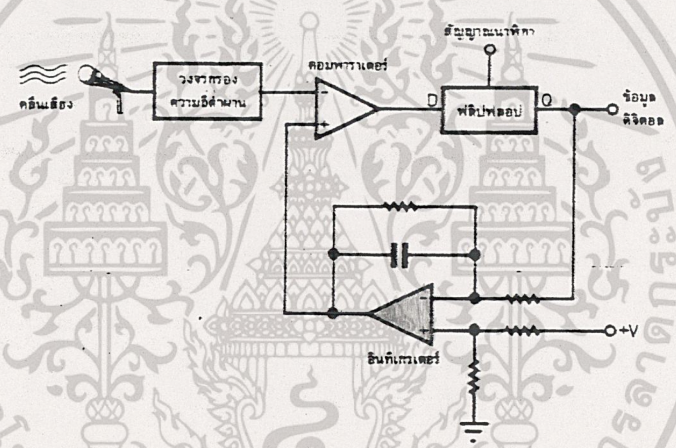
ตั้งแต่เทคโนโลยีทางด้านสารกึ่งตัวนำ ได้รับการพัฒนาขึ้นมา จนสามารถสร้างเครื่องจักรที่ใช้ในการคำนวณที่มีประสิทธิภาพสูงที่สุดเท่าที่มนุษย์เคยรู้จักกันซึ่งเรียกว่า -ดิจิตอลคอมพิวเตอร์ (digital computer) ดิจิตอลคอมพิวเตอร์ได้รับการวิวัฒนาการไปมากและมีแนวโน้มถึงการเปลี่ยนแปลงใหม่ ๆ ในอนาคต ดิจิตอลคอมพิวเตอร์ได้นำไปใช้งานในด้านต่าง ๆ อย่างมากมาย เช่น การนำไปใช้งานทางด้านธุรกิจ ทางอุตสาหกรรม เป็นต้น ซึ่งดิจิตอลคอมพิวเตอร์จำเป็นจะต้องควบคุมด้วยการป้อนข้อมูลเข้าทางคีย์บอร์ดหรือบัตรเจาะรู แล้วแสดงเอาท์พุทด้วยตัวอักษรบนจอ , บนกระดาษหรือทำให้อุปกรณ์อื่น ๆ ที่ควบคุมด้วยคอมพิวเตอร์ทำงาน แต่ในช่วงไม่กี่ปีมานี้คอมพิวเตอร์ได้รับการพัฒนาไปใช้ในการสร้างเสียง ซึ่งจะให้เอาท์พุทเป็นเสียงพูดของมนุษย์แทนที่จะเป็นตัวอักษรบนจอภาพหรือกระดาษในระยะแรก ๆ นั้นทำได้แต่เพียงทำให้คอมพิวเตอร์พูดได้เพียงคำพูดบางคำที่จำเป็นเท่านั้น และกำลังได้รับการพัฒนาขึ้นไปเรื่อย ๆ จนถึงขนาดที่สามารถจะพูดได้อย่างมนุษย์ ซึ่งต้องใช้หน่วยความจำในการเก็บข้อมูลคำพูดอย่างมากมาย ซึ่งต่อไปในอนาคตด้วยเทคโนโลยีขั้นสูงจะสามารถที่จะสร้างหน่วยความจำที่มีขนาดเล็ก แต่เก็บข้อมูลได้อย่างมากมาย ในปัจจุบันนี้ผู้เชี่ยวชาญทางด้านคอมพิวเตอร์ได้มุ่งความสนใจไปที่การวิเคราะห์และสังเคราะห์เสียงพูดของมนุษย์ โดยทำให้มนุษย์สามารถที่จะสั่งให้คอมพิวเตอร์ทำงานด้วยเสียงพูดแทนที่จะป้อนข้อมูลเข้าทางคีย์บอร์ด และสามารถทำให้มนุษย์พูดจาโต้ตอบกับคอมพิวเตอร์ได้ ซึ่งจะนำเทคนิคอันนี้ไปใช้ในหุ่นยนต์ ต่อไปในโลกอนาคตมนุษย์จะได้รับความสะดวกสบายอย่างมากมาย ในที่นี้จะกล่าวถึงการนำไอซีสำเร็จรูปมาใช้ ในการสร้างเสียงพูด ซึ่งพอจะเป็นแนวทางให้ผู้สนใจรับประโยชน์บ้างไม่มากก็น้อย

ในการวิเคราะห์เสียงพูด จะต้องมีการเปลี่ยนสัญญาณ จากอนาล็อกไปเป็นดิจิตอล (Analog to digital converter) ซึ่งสามารถทำได้ หลายวิธี ในที่นี้ใช้ CVSD (continuous variable slope delta modulation) ทั้งนี้เนื่องจากจะประหยัดหน่วยความจำในการเก็บข้อมูล

เดลต้ามอดูเลชั่น

เทคนิคของเดลต้ามอดูเลชั่นจะไม่ใช้การสุ่มสัญญาณหนึ่งจุดแล้วแปลงเป็นข้อมูลดิจิตอลหนึ่งเวิร์ด ที่มีความละเอียดเป็นจำนวนบิตที่ต้องการ แต่จะใช้วิธีเปรียบเทียบความสูงหรือการเปลี่ยนแปลงของสัญญาณเสียงแทน

ข้อมูลที่ได้ก็คือทิศทางของการเปลี่ยนแปลง ซึ่งก็มีเพียง ขึ้น หรือ ลง เท่านั้น ดังนั้นความกว้างของข้อมูลดิจิตอลจึงใช้เพียงบิตเดียวก็เพียงพอ ข้อดีของวิธีการเดลต้ามอดูเลชั่นก็คือใช้หน่วยความจำน้อยกว่าวิธีการแบบอื่น ๆ

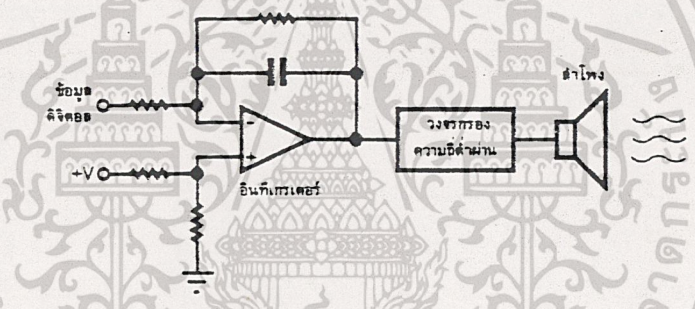


รูปที่ 1 วงจรเบื้องต้นของเดลต้ามอดูเลชั่นในส่วนของการแปลงจากสัญญาณเสียงเป็นดิจิตอล

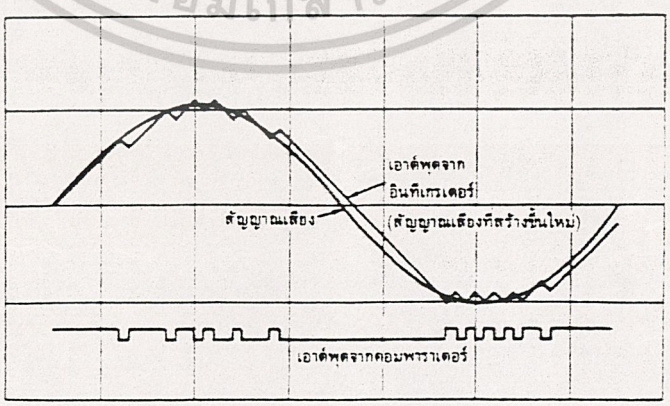
รูปที่ 3.1 เป็นวงจรเบื้องต้นของเดลต้ามอดูเลชั่น คอมพาราเตอร์จะทำหน้าที่เปรียบเทียบสัญญาณอินพุตปัจจุบันกับสัญญาณอินพุตก่อนหน้า ซึ่งได้จากการป้อนกลับมายังอินทิเกรเตอร์เอาต์พุตจากการเปรียบเทียบถูกป้อนผ่านฟลิปฟลอปที่ควบคุมด้วยสัญญาณนาฬิกาเพื่อให้ได้เป็นข้อมูลดิจิตอล ซึ่งก็คือการกำหนดอัตราการสุ่มสัญญาณนั่นเอง

สัญญาณที่ได้จากตัวเปรียบเทียบและจากอินทิเกรเตอร์ เปรียบเทียบกับสัญญาณอินพุตแสดงในรูปที่ 3.3 ลักษณะเช่นนี้จะพบว่า ยิ่งความถี่ของสัญญาณนาฬิกามีค่าสูงก็ยิ่งสามารถ

บันทึกการเปลี่ยนแปลงที่แคบได้มากขึ้น ทำให้ได้คุณภาพเสียงที่ดีขึ้น แต่ก็สิ้นเปลืองหน่วยความจำมากขึ้นตามไปด้วย ความถี่เท่าใดจึงจะเพียงพอคงต้องใช้การทดลองโดยการนำเอาต์พุตสุดท้ายที่เป็นข้อมูลดิจิตอลผ่านวงจรแปลงกลับในรูปที่ 3.2 แล้วฟังเสียงก็ได้ หากฟังเป็นภาษามนุษย์รู้เรื่องก็ใช้ที่ค่านั้น สำหรับเสียงพูดคุณภาพเทียบเท่าเสียงจากโทรศัพท์ซึ่งมีแถบกว้างประมาณ 4 kHz ก็ใช้เพียง 16 kHz แต่ที่ความถี่ต่ำถึง 9.6 kHz ก็ยังฟังรู้เรื่อง ความถี่นี้จะเป็นตัวกำหนดอัตราเร็วข้อมูล (bit rate) ซึ่งที่ 16 kHz ก็เท่ากับ 1,600 บิตต่อวินาที



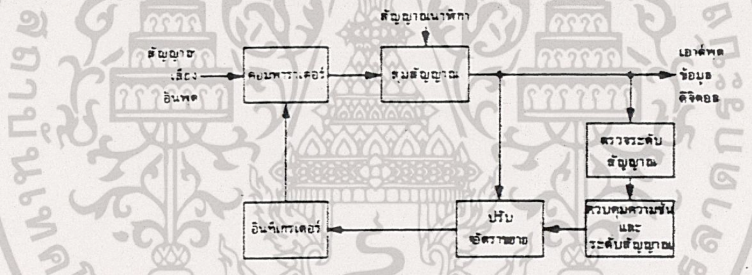
รูปที่ 2 วงจรที่ใช้แปลงกลับจากข้อมูลดิจิตอลเป็นสัญญาณเสียง



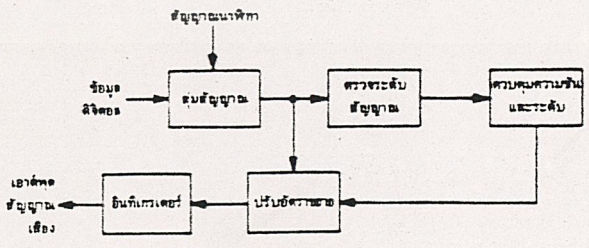
รูปที่ 3 เปรียบเทียบสัญญาณอินพุตกับข้อมูลที่ได้สัญญาณอะนาลอกจากอินทิเกรเตอร์

CVSD

ข้อจำกัดของวิธีการเดลต้ามอดูเลชั่นก็คือ แถบกว้างความถี่ใช้งาน ซึ่งถูกจำกัดโดยความถี่สัญญาณนาฬิกา และจะสูงกว่าความถี่สูงสุดของสัญญาณอินพุตมากกว่า 2 เท่าขึ้นไปอีก อันหนึ่ง คือความเร็วของการเปลี่ยนแปลงความสูงของสัญญาณ หรือ ไดนามิกเรนจ์ ระบบเดลต้ามอดูเลชั่นธรรมดามีค่าไดนามิกเรนจ์ที่แคบ จำเป็นต้องมีส่วนเพิ่มเติมทำหน้าที่ขยายไดนามิกเรนจ์ให้กว้าง โดยการควบคุมอัตราขยายของอินทีเกรเตอร์ เพื่อให้ตอบสนองต่อสัญญาณที่มีความชันมาก ๆ ได้ทัน ระบบนี้มีชื่อเรียกใหม่ว่า ระบบเดลต้ามอดูเลชั่นแบบเปลี่ยนแปลงความชันต่อเนื่อง หรือ CVSD (continuous variable slope delta modulation)



รูปที่ 3.4 แผนผังการทำงานของระบบ CVSD ในส่วนของการเปลี่ยนแปลงจากสัญญาณเสียงเป็นข้อมูลดิจิตอล



รูปที่ 3.5 แผนผังการทำงานของระบบ CVSD ในส่วนเปลี่ยนแปลงสัญญาณเสียง

ระบบ CVSD ทั้งส่วนแปลงจากอะนาลอกเป็นดิจิตอลและส่วนแปลงกลับจากดิจิตอลเป็นอะนาลอก แสดงในรูปที่ 4 และ 5 ตามลำดับ วิธีการ CVSD ก็คือมีการตรวจระดับสัญญาณ โดยอาจใช้วิธีการจัดให้มีรีจิสเตอร์สำหรับเก็บข้อมูลดิจิตอลล่าสุดจำนวน 3 ถึง 4 บิต แล้วจรวจดูว่าเป็น "0" หมดหรือ "1" หมอหรือไม่ ถ้าใช้แสดงว่าขณะนี้อัตราขยายของอินทีเกรเตอร์ต่ำเกินไป ตอบสนองต่อความชันของสัญญาณไม่ทัน ก็จะทำการเพิ่มอัตราขยายให้สูงขึ้นเฉพาะในช่วงนั้น

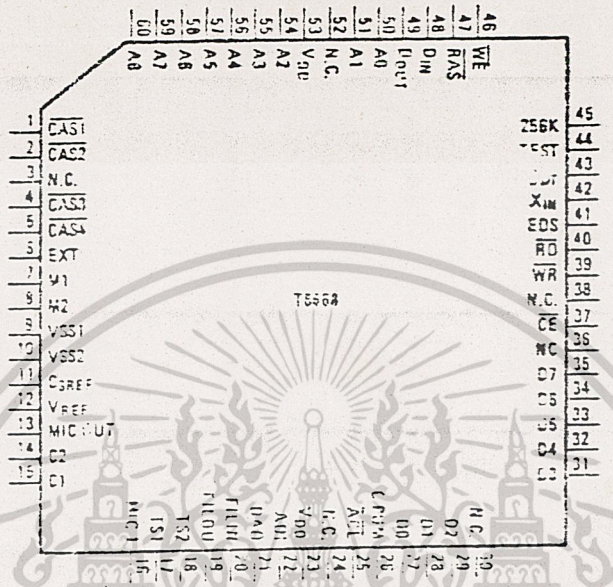
ในส่วนของการแปลงกลับก็จะต้องมีการทำงานในลักษณะเดียวกัน คือมีรีจิสเตอร์ตรวจดูข้อมูบว่าเป็น "0" หมดหรือ "1" หมดหรือไม่ แล้วจัดการควบคุมอัตราขยายของอินทีเกรเตอร์ให้สอดคล้องกัน

จากที่ได้กล่าวมา บิตเรตหรืออัตราเร็วของข้อมูลสำหรับวิธีการเดลต้ามอดูเลชั่นมีค่าเท่ากับความถี่สัญญาณนาฬิกา เสียงพูดคุณภาพเท่าระบบโทรศัพท์ที่แถบกว้างความถี่ 4 kHz ต้องใช้ความถี่นาฬิกา 16 kHz ได้ข้อมูลดิจิตอลที่บิตเรต 16 K บิตต่อวินาที หรือพูดอีกอย่างได้ว่า การบันทึกเสียง 1 วินาที ต้องใช้หน่วยความจำ 16 K บิต ถ้าต้องการเวลาบันทึกยาวนานขึ้นก็ต้องใช้หน่วยความจำมากขึ้นเป็นทวีคูณ จึงหลีกไม่พ้นที่จะต้องใช้ไดนามิกแรมซึ่งมีความจุสูง

ปัจจุบันนี้มีการคิดค้นและออกแบบวงจรอิเลคทรอนิคส์ขึ้นมาทำหน้าที่บันทึก และถ่ายทอดเสียงแทนเทป ซึ่งให้ความสะดวก และคล่องตัวกว่าถึงแม้ วงจรอิเลคทรอนิคส์จะบันทึกเสียงไว้ได้ไม่มากเท่ากับเทป แต่เมื่อเทียบประโยชน์ในการใช้งานบางอย่างแล้ว นับว่าคุ้มค่ามากกว่าในที่นี้ เราใช้ไอซีสำเร็จรูป เบอร์ T6668 ซึ่งมีรายละเอียดดังต่อไปนี้

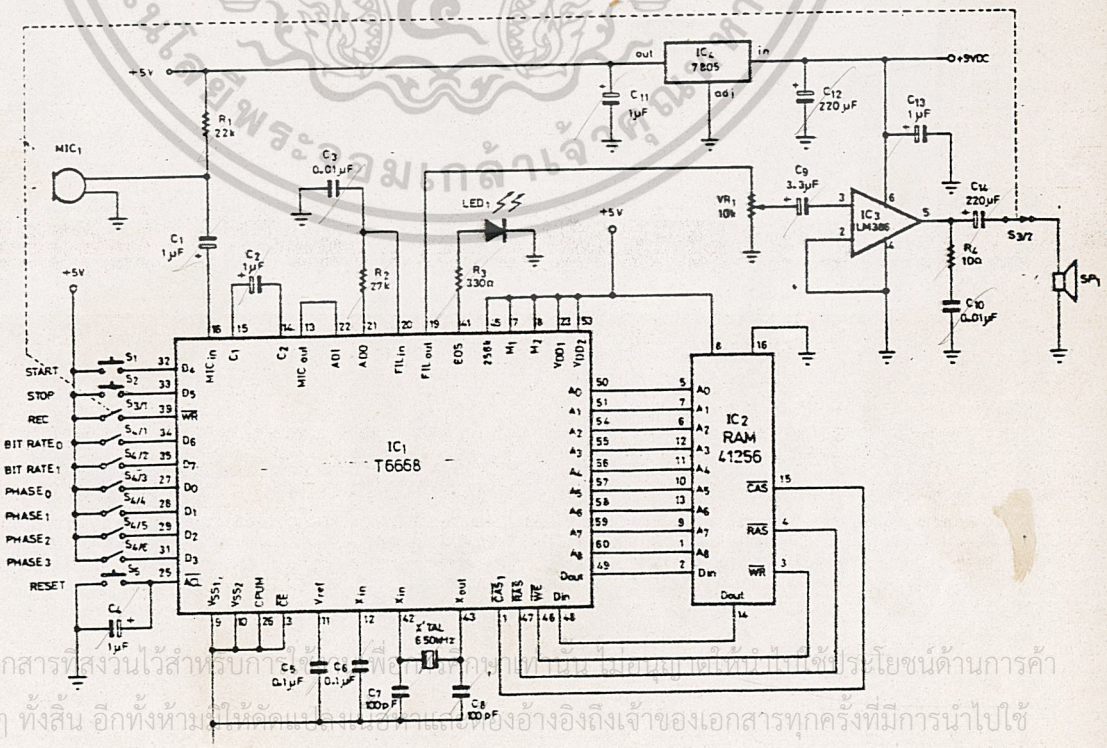
วงจรการทำงาน

จากวงจรในรูปที่ 3.7 เป็นวงจรของชุดวิเคราะห์เสียงพูด หัวใจสำคัญของวงจรอยู่ที่ IC_1 และ IC_2 ซึ่งเป็นไอซีไมโครโปรเซสเซอร์และหน่วยความจำ ตัว IC_1 เองถูดออกแบบขึ้นมาเพื่อใช้งานด้านวิเคราะห์เสียงโดยเฉพาะ ซึ่งเป็นผลิตผลการประดิษฐ์ของบริษัทโตชิบาแห่งประเทศญี่ปุ่น เป็นไดซีชนิด CMOS LSI ลักษณะโครงสร้างภายนอกและตำแหน่งขาต่างๆ แสดงไว้ในรูปที่ 3.6



รูปที่ 6 แสดงตำแหน่งขาต่าง ๆ ของไอซี T6668

รูปที่ 7 วงจรชุดวิเคราะห์เสียงพูด



KBPS	D_7	D_6
8	Ø	Ø
11	Ø	1
16	1	Ø
32	1	1

ตารางที่ 1

การทำงาน IC จะทำการรับสัญญาณเสียงพูดเข้ามาจากนั้นทำการขยาย แล้วเปลี่ยนจากสัญญาณอนาล็อกไปเป็นข้อมูลดิจิตอล แล้วไปเก็บไว้ที่ไดนามิคแรม (DRAM) IC_2 โดย CPU ภายในจะทำการเลื่อนแอดเดรส ที่จะนำเข้าไปเก็บเองโดยอัตโนมัติ เมื่อทำการแปลงข้อมูลจาก D/A จะใช้อัตรา 10 BIT D/A เพื่อเปลี่ยนกลับมาเป็นเสียงเช่นเดิมการอัดเข้าไป เราจะสามารถเลือก speed ได้ 4 speed โดยเลือกที่ D_6 - D_7

จากตารางที่ 1

1. ถ้าเราเลื่อนสวิทซ์ D_7 - D_6 ไปที่ Ø. Ø จะทำให้อัตราความเร็วของการแปลงข้อมูลเป็น 8K bit ต่อวินาที ทำให้อัดหรือเล่นได้นาน 128 วินาที

2. D_1 - D_6 เป็น Ø.1 จะทำให้อัตราการแปลงข้อมูลเป็น 11K bit ต่อวินาทีทำให้อัดหรือเล่นได้นาน 93 วินาที

3. D_7 - D_6 เป็น 1.Ø ทำให้อัตราการแปลงข้อมูลเป็น 16K bit ทำให้อัดหรือเล่นได้นาน 64 วินาที

4. D_7 - D_6 เป็น 1.1 ทำให้อัตราการแปลงข้อมูลเป็น 32K bit ทำให้อัดหรือเล่นได้นาน 32 วินาที

การทดลองใช้ X-TAL 650 KHz เป็นฐานความถี่และต่อกับ RAM 256K จำนวน 4 ตัว ทำให้ความจุของ memory เพิ่มขึ้นเป็น 1M bit ดังวงจรรูปที่ 3.8 การอัดเมื่อเรา

ชนิดของ RAM	256K	M_2	M_1	ADDRESS ที่หยุด
64KRAM ตัวที่1	Ø	Ø	Ø	ØFFFFH
64KRAM ตัวที่2	Ø	Ø	1	1FFFFH
64KRAM ตัวที่3	Ø	1	Ø	2FFFFH
64KRAM ตัวที่4	Ø	1	1	3FFFFH
256KRAM ตัวที่1	1	Ø	Ø	3FFFFH
256KRAM ตัวที่2	1	Ø	1	7FFFFH
256KRAM ตัวที่3	1	1	Ø	BFFFFH
256KRAM ตัวที่4	1	1	1	FFFFFH

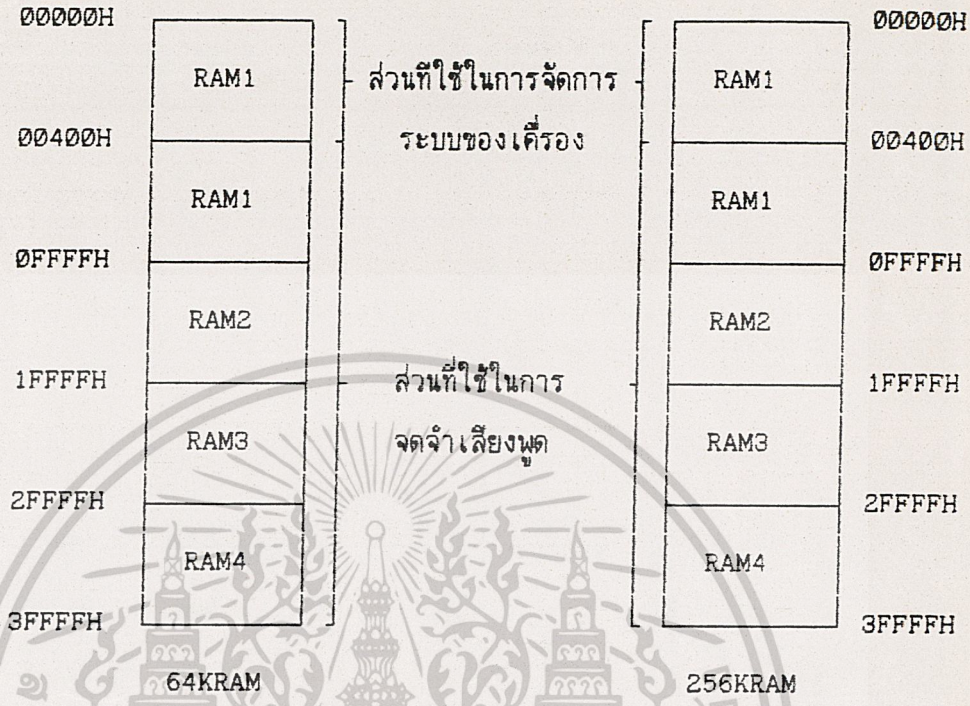
ตารางที่ 2

การเพิ่มเมมโมรี่ ให้กับ IC T6668 สามารถกำหนดได้โดยการต่อขา M_2 (ขา 8) M_1 (ขา 7) ตามตารางข้างล่างนี้ คือ ถ้าเราต่อ M_2 M_1 ลงกราวด์ T6668 จะทำการเขียนหรืออ่านข้อมูลจาก 00000H ไปจนถึง 0FFFFH แล้วตัวมันเองก็จะเลิกการอ่านหรือการเขียนมารอการเริ่มต้นใหม่

ดังนั้นเราจึงกำหนดขนาดของเมมโมรี่ได้ตามต้องการเพื่อการประหยัดในการนำไปใช้งานที่ต้องการขนาดเมมโมรี่ต่างกันได้

แผนภูมิของเมมโมรี่ที่ใช้ในการทำงานทั้ง 2 แบบ ตามรูปที่ 3.10

การต่อเมมโมรี่เพิ่มเติมทำโดยการง้างขา CAS (ขา 15 ของ 41256) ออกมาแล้วซ้อนทับไปดังรูปที่ 6 จากนั้นก็ต่อขา CAS ไปยัง CAS2, CAS3 และ CAS4 ของไอซี T6668



รูปที่ 3.5 แผนภูมิของ Memory ที่ใช้ในการทำงานทั้งสองแบบ

ระยะเวลา ช่อง	D_7	D_6
128 วินาที	0	0
93 วินาที	0	1
64 วินาที	1	0
32 วินาที	1	1

ตารางที่ 4

การใช้งานในแบบธรรมดา

1. เปิดเครื่องจะเห็น LED ติดอยู่
2. กดสวิทช์ไปที่อัดค้างไว้ (CE จะต้องต่อกับกราวด์ด้วย)
3. เลือกช่องที่จะอัดเข้าไปโดยช่องที่จะอัดมีอยู่ 4 ตัว สวิทช์นี้เป็นไบนาริโค้ด ดังตารางที่ 3

D_0	D_1	D_2	D_3	ช่องที่
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

ตารางที่ 3

4. เลือกสปีดโดยตั้ง D_7 - D_6 ได้ตามต้องการ (เวลาที่แสดงนี้ใช้เมมโมรี่ 1M bit)

5. กดปุ่มสวิทช์ START แล้วไฟที่ LED จะดับ แสดงว่าเครื่องกำลังอัดคำพูดเข้าไปเก็บ เมื่อพูดจนพอใจแล้วจึงกดสวิทช์ STOP อีกครั้งหนึ่งไฟที่ LED จะสว่างในกรณีที่เราพูดนานเกินกว่าเวลาที่กำหนดในข้อที่ 4 เมื่อถึงกำหนดเวลาเครื่องจะหยุดการอัดโดยอัตโนมัติ ไฟที่ LED จะสว่างขึ้นมาเพื่อบอกให้เรารู้เป็นการสิ้นสุดขั้นตอนการอัดใน 1 ช่อง

6. ถ้าเราต้องการอัดในช่องอื่น ๆ อีก ก็ทำเช่นเดียวกัน ตั้งแต่ต้นจนถึงข้อ 5 (เวลารวมของแต่ละช่องต้องไม่เกินเวลาที่ได้กำหนดไว้)

7. การอ่านทำโดยการยกเลิกสวิทช์ WR ขึ้น (CE ต่อกราวด์เหมือนเดิม)

8. เลือกช่องที่จะอ่านและสปีด

9. กดสวิทช์ START เครื่องจะพูดตามที่อัดไว้ ถ้าเรากดสวิทช์ซ้ำกันหลายครั้งในระหว่างพูดเครื่องจะจำได้ว่ามีการกดสวิทช์ START ขึ้นเพียงครั้งเดียว และจะพูดซ้ำอีกเมื่อพูดจบ

10. เมื่อต้องการให้เครื่องพูดติดต่อกันทำโดยเลือกช่องแรก กด START เสร็จแล้วเปลี่ยนช่องแล้วกดสวิทช์ซ้ำอีกตามต้องการได้

จากที่กล่าวมาข้างต้นจึงทำให้เครื่องนี้สามารถตัดต่อคำพูดได้ พูดซ้ำได้เร่งหรือลดสปีดคำพูดได้ ทำให้เกิดเป็นเสียงแปลก ๆ ในชาวด์แทรกภาพยนตร์ได้ เครื่องนี้สามารถควบคุมได้จาก CPU โดยตรง ซึ่งทำให้สามารถไปประยุกต์ใช้งานต่าง ๆ ได้ตามต้องการ

การสร้าง

วงจรวิเคราะห์เสียงพูดชุดนี้ ถึงแม้จะประกอบด้วยอุปกรณ์ไม่มากนักแต่การประกอบก็ต้องอาศัยความละเอียดมากทีเดียว โดยเฉพาะการบัดกรีที่ขาของไอซี T6668 ต้องระวังเป็นพิเศษ เพราะแต่ละขาชิดกันมาก อาจจะช็อตถึงกันได้ และเพื่อให้ขนาดของปริ้นท์เล็กลงจำเป็นต้องใช้ปริ้นท์แบบสองหน้าแบบของลายปริ้นท์ด้านบนได้แสดงไว้ในรูปที่ 3.12 ส่วนลายปริ้นท์ด้านล่างตามรูปที่ 3.13 และการวางอุปกรณ์ต่าง ๆ ในรูปที่ 3.14

เมื่อทำปริ้นท์เสร็จเรียบร้อยแล้วก่อนใส่อุปกรณ์ต่าง ๆ ลงไปคุณจะต้องทำการต่อสายจั๊มระหว่างปริ้นท์ด้านบนและด้านล่างถึงกันเสียก่อน ซึ่งมีทั้งหมด 14 จุด จุดที่จะต้องจั๊มนั้นจะทำเครื่องหมายกากะบาดไว้ให้แล้วตามรูปที่ 3.15

เมื่อจั๊มสายระหว่างปริ้นท์ทั้งสองด้านแล้ว ต่อมาก็เริ่มประกอบอุปกรณ์ต่าง ๆ เริ่มของแผ่นปริ้นท์เป็นส่วนใหญ่และมีบางจุดที่บัดกรีทางด้านล่างของแผ่นปริ้นท์

ในกรณีที่คุณใช้ไมค์ชนิดคอนเดนลายปริ้นท์ ด้านบนกับด้านล่าง เซอร์ไมค์ คุณจะต้องต่อตัวต้านทาง R_1 เพิ่มที่บนแผ่นปริ้นท์ตามรายละเอียดในรูปที่ 3.14

จากวงจรในรูปที่ 3.8 เราออกแบบแผ่นให้ใช้กับ RAM เพียงตัวเดียวตามปกติ เราสามารถต่อ RAM ขนานกันได้ถึง 4 ตัว เพื่อให้สามารถเก็บข้อมูลได้มากขึ้นการจะเพิ่ม RAM อีกก็สามารถทำได้โดยเอา RAM ตัวที่ 2, 3 และ 4 คร่อมลงไปบน RAM ตัวแรก จากนั้นต่อขา CAS (ขา 15) ของ RAM แต่ละตัว ไปเข้าขา CAS ของไอซี T6668 ได้แก่ขา 2, 4 และขา 5

การทดสอบ

เมื่อประกอบอุปกรณ์เรียบร้อยดีแล้ว ตรวจดูไม่มีอะไรผิดพลาดแล้ว สายต่าง ๆ ต่อไว้ถูกต้องดีแล้ว เริ่มแรกป้อนไฟเข้าวงจรวัดไฟที่ขาเอาท์พุทของ IC จะต้องได้บวก 5 โวลท์ ขณะนี้ LED จะติดสว่างด้วย จากนั้นปรับ VR_1 ไว้ที่ตำแหน่งกึ่งกลาง ตอนนี้จะได้ยินเสียงซ่าเล็กน้อยที่ลำโพง ต่อมาทดลองบันทึกข้อความดู ขั้นแรกให้เลื่อนสวิทช์ SW_3 (Start) ไปที่ LED จะดับลง ลองพูดข้อความอะไรก็ได้เข้าไปที่ไมค์กะประมาณเวลาสัก 35 วินาที จากนั้นกดสวิทว์ SW_2 เพื่อหยุดการบันทึก LED จะติดสว่าง จากนั้นเลื่อนสวิทช์ SW_3 (REC) ไปที่ตำแหน่งปิด ตอนนี้ก็เป็นการเสร็จสิ้นการบันทึก

จากนั้นให้กดสวิทช์ SW_1 เพื่อฟังข้อความที่บันทึกไว้ ขณะที่กดสวิทช์ SW_1 นี้ LED_1 จะดับลงพร้อมกับมีเสียงที่เราบันทึกเข้าไปดังออกมาที่ลำโพง เมื่อข้อความที่ถูกบันทึกไว้หมดแล้ว (จะได้ความยาวประมาณ 30 วินาที) LED_1 จะสว่างขึ้นมาอีกและถ้าเราต้องการฟังข้อความที่บันทึกไว้อีก ก็กดสวิทช์ SW_1 คุณจะสามารถเล่นกลับบได้ตลอดเวลาจนกว่าคุณจะบันทึกข้อมูลใหม่ลงไป

การแก้ปัญหาเสียงหวีดขณะบันทึก

ปัญหาหนึ่งที่เกิดกับชุดวิเคราะห์เสียงพูด คือเกิดเสียงหวีดขึ้นที่ลำโพงทำให้เสียงที่บันทึกถูกรบกวน สาเหตุของเสียงหวีด เกิดจากการป้อนกลับของสัญญาณระหว่างไมค์กับลำโพง วิธีแก้ไขให้ต่อสวิทช์สำหรับตัดลำโพงออกขณะที่ทำการบันทึกโดยจะใช้สวิทช์ตัวเดียวกับ SW_3 (REC) หรือต่อสวิทช์เพิ่มขึ้นต่างหากก็ได้

บทที่ 4
ไดนามิกแรม (DRAM)

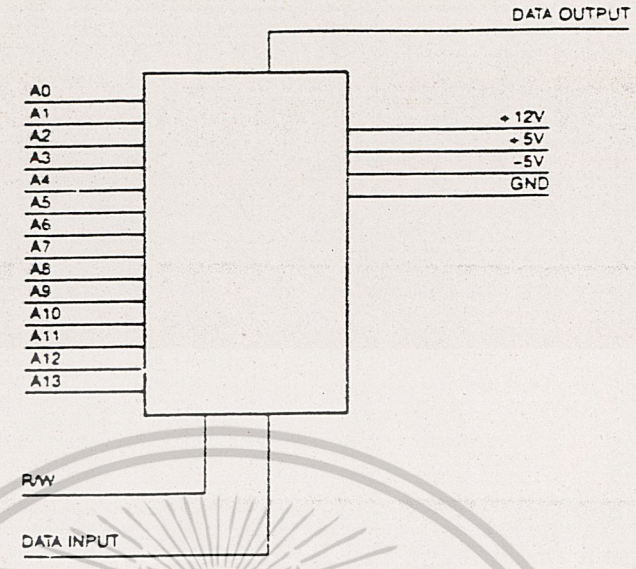
4.1 หน่วยความจำแบบ ไดนามิกแรม (DRAM)

หน่วยความจำแบบไดนามิก ที่สามารถให้ความจุที่สูงมากต่อตัวชิป ซ้ำตัวมันยังมีราคาที่ถูกมากด้วย ในกรณีที่ระบบไมโครโปรเซสเซอร์ของเราต้องการความจุของหน่วยความจำที่มาก ๆ แล้ว หน่วยความจำที่น่าจะเอามาใช้งานคือ ไดนามิกแรม (DRAM) ในลักษณะของการต่อใช้งานนั้น อาจมีข้อยุ่งยากบ้างเมื่อเทียบกับการต่อหน่วยความจำแบบสแตติกแรมที่กล่าวมา ฉะนั้นจึงขอยกตัวอย่างของ DRAM ที่ใช้เป็นพื้นฐานการศึกษาคือ เบอร์ 4116 หรือหากเป็นของบริษัท NEC จะใช้ชื่อเป็น uPD416 ซึ่งเป็นเบอร์ที่มีการใช้มานาน สามารถหาซื้อได้ง่าย มีราคาถูกเป็น DRAM ที่ใช้ในเครื่องไมโครคอมพิวเตอร์ของ APPLE โครงสร้างของ 4116 จะมีความจุขนาด 16KBx1 ใช้สายข้อมืออินพุทและเอ้าพุทที่แยกจากกันแต่ระบบไฟเลี้ยงจะใช้ถึง 3 ระดับคือ +12v,+5v,-5v และหากจะพิจารณาจากขนาดของหน่วยความจำโดยวิธีการนับจำนวนขาแอดเดรสของหน่วยความจำดังที่ได้กล่าวมาในเรื่องของ SRAM นั้น 4116 ก็ควรจะมีขาสัญญาณดังต่อไปนี้

- สัญญาณแอดเดรสจำนวน 14 ขา
- สัญญาณข้อมูลออก 1 ขา
- สัญญาณข้อมูลเข้า 1 ขา
- สัญญาณการบอกอ่านเขียน 1 ขา

ซึ่งสามารถเขียนเป็นบล็อกได้ดังรูปที่ 4.1.1
แต่จำนวนขาแอดเดรสของหน่วยความจำไม่ได้เป็นไปอย่างรูปที่ผ่านมา หากแต่จะมีเพียง 7 เส้นเท่านั้น แสดงดังรูปที่ 4.1.2

เพราะการกำหนดแอดเดรสของหน่วยความจำนี้เป็นแบบ มัลติเพลกซ์ (multiplex) โดยจะแบ่งสัญญาณแอดเดรสนี้ออกเป็น 2 กลุ่มๆ ละ 7 เส้น กลุ่มแรกเป็น A0 A6 เราเรียกแอดเดรสกลุ่มนี้ว่า แอดเดรสทางแนวนอน (ROW ADDRESS) และกลุ่มที่สอง (A7-A13)



รูปที่ 4.1.1 แสดงถึงขาสัญญาณต่างของไดนามิกแรม 4116 ที่ควรจะเป็น

คือแอดเดรสทางแนวตั้ง (COLUMN ADDRESS) ในการต่อสายแอดเดรสสู่ตัวหน่วยความจำไดนามิกแรมเราจะต้องสร้างส่วนของการมัลติเพล็กสัญญาณการอ้างแอดเดรสจาก CPU ดังแสดงในรูปที่ 4.1.3 เพื่อส่งสัญญาณแอดเดรสให้หน่วยความจำทีละกลุ่มโดยการส่งกลุ่มของแอดเดรสทางแนวนอนไปก่อนและตามด้วยแอดเดรสทางแนวตั้ง

ในการส่งแอดเดรสไปสู่หน่วยความจำในแต่ละแนวนั้นเราจะต้องส่งสัญญาณอันหนึ่งเพื่อบอกให้ตัวหน่วยความจำไดนามิกแรมรู้ว่าตอนนี้เราส่งแอดเดรสทางแนวใดไปให้เพื่อมันจะได้แลทช์ (LATCH) ค่าแอดเดรสได้อย่างถูกต้อง ฉะนั้นขาอินพุทของ DRAM จึงต้องมีขา RAS (Row Address Strobe) ,CAS (Column Address Strobe) เพิ่มเข้ามาเพื่อใช้ในการรับสัญญาณที่บอกให้แลทช์แอดเดรสในแนวนอนและแนวตั้งตามลำดับ เมื่อ DRAM ได้แอดเดรสครบ (A0 A6,A7 A13) แล้วก็สามารถทำการอ่านเขียนได้เหมือนหน่วยความจำทั่วไปโดยการใช้สัญญาณ WE เข้าที่ Write (ขา 3 ขา 4116) ดังนั้นไดอะแกรมเวลาของสัญญาณการอ้างแอดเดรสจะเป็นดังรูปที่ 4.1.4 ส่วนในรูปที่ 4.1.5 เป็นรูปที่แสดงถึงวงจรที่ใช้งานการมัลติเพล็กแอดเดรสจาก CPU เข้าสู่ DRAM โดยสัญญาณที่ใช้ในการกำหนดตัว มัลติเพล็กนั้นเราเรียกว่า MUX ซึ่งจากรูปจะเห็นเมื่อ MUX มีค่าเป็น "1" ตัวเลือกข้อมูล

NEC Microcomputers, Inc.

NEC
μPD416
μPD416-1
μPD416-2
μPD416-3
μPD416-5

16384 x 1 BIT DYNAMIC MOS RANDOM ACCESS MEMORY

DESCRIPTION

The NEC μPD416 is a 16384 words by 1 bit Dynamic MOS RAM. It is designed for memory applications where very low cost and large bit storage are important design objectives.

The μPD416 is fabricated using a double-poly-layer N-channel silicon gate process which affords high storage cell density and high performance. The use of dynamic circuitry throughout, including the sense amplifiers, assures minimal power dissipation.

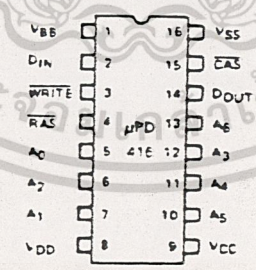
Multiplexed address inputs permit the μPD416 to be packaged in the standard 16 pin dual-in-line package. The 16 pin package provides the highest system bit densities and is available in either ceramic or plastic. Noncritical clock timing requirements allow use of the multiplexing technique while maintaining high performance.

FEATURES

- 16384 Words x 1 Bit Organization
- High Memory Density – 16 Pin Ceramic and Plastic Packages
- Multiplexed Address Inputs
- Standard Power Supplies +12V, -5V, +5V
- Low Power Dissipation: 462 mW Active (MAX), 40 mW Standby (MAX)
- Output Data Controlled by $\overline{CAS}$ and Unlatched at End of Cycle
- Read-Modify-Write, $\overline{RAS}$-only Refresh, and Page Mode Capability
- All Inputs TTL Compatible, and Low Capacitance
- 128 Refresh Cycles
- 5 Performance Ranges:

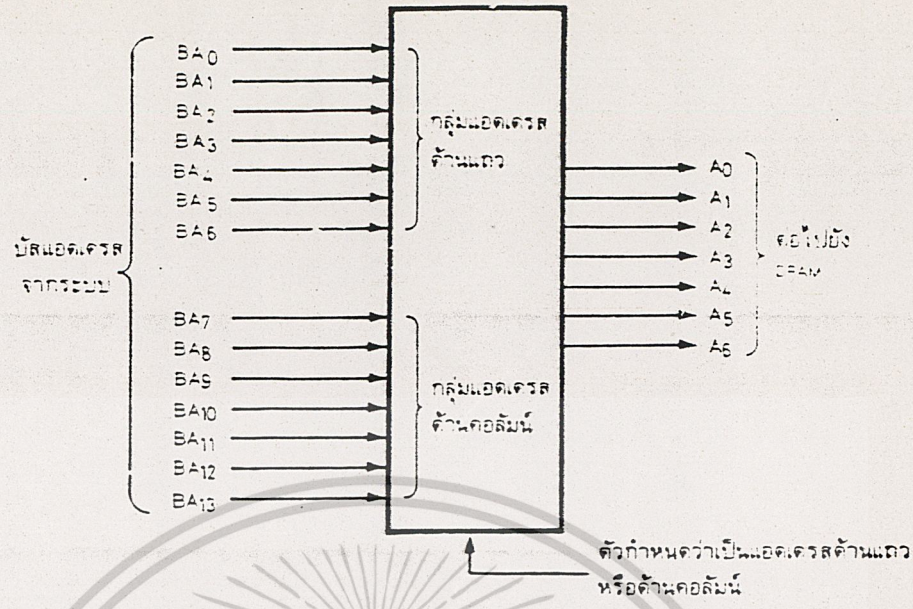
	ACCESS TIME	RAN CYCLE	RMW CYCLE
μPD416	300 ns	510 ns	575 ns
μPD416-1	250 ns	410 ns	465 ns
μPD416-2	200 ns	375 ns	375 ns
μPD416-3	150 ns	375 ns	375 ns
μPD416-5	120 ns	320 ns	320 ns

PIN CONFIGURATION

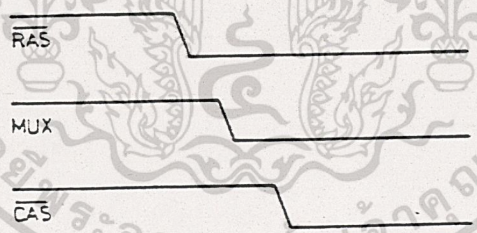


A_0-A_6	Address Inputs
$\overline{CAS}$	Column Address Strobe
D_{IN}	Data In
D_{OUT}	Data Out
$\overline{RAS}$	Row Address Strobe
$\overline{WRITE}$	Read/Write
V_{BB}	Power (-5V)
V_{CC}	Power (+5V)
V_{DD}	Power (+12V)
V_{SS}	Ground

รูปที่ 4.1.2 แสดงข้อมูลบางประการและลักษณะขาของ 4116

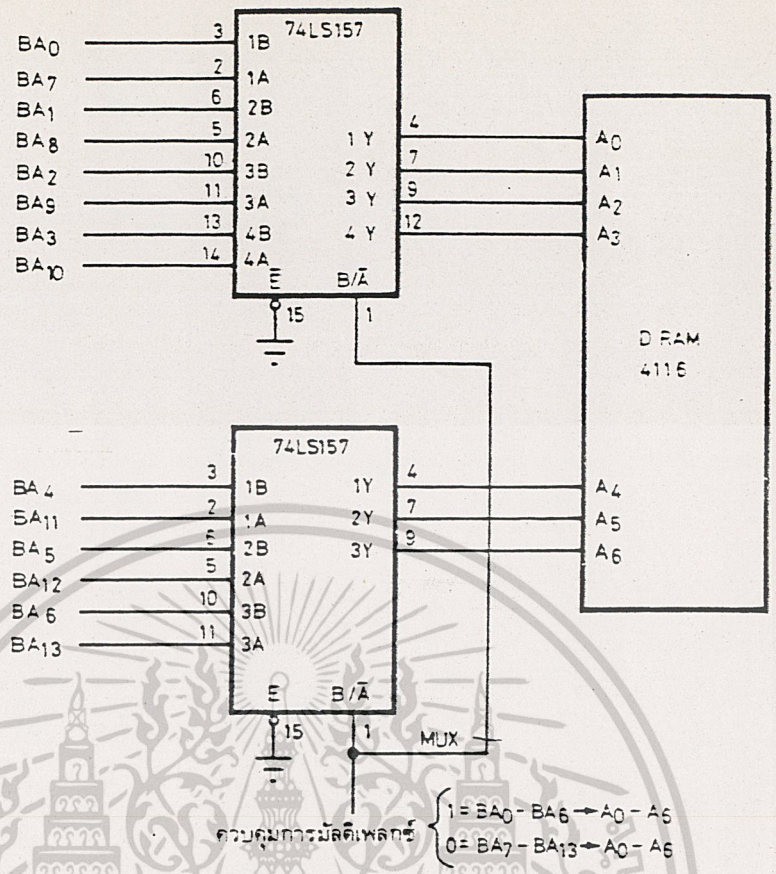


รูปที่ 4.1.3 แสดงถึงส่วนของการมัลติเพล็กสัญญาณการอ้างแอดเดรสเพื่อให้สัญญาณการอ้างแอดเดรสเข้าสู่ขาแอดเดรสของ DRAM ทีละกลุ่มซึ่งส่วนนี้สามารถได้ง่ายมากโดยใช้อุปกรณ์โลจิกพวก TTL ซึ่งมีฟังก์ชั่นของการมัลติเพล็กอยู่แล้วเช่น เบอร์ 74LS257 เป็นต้น

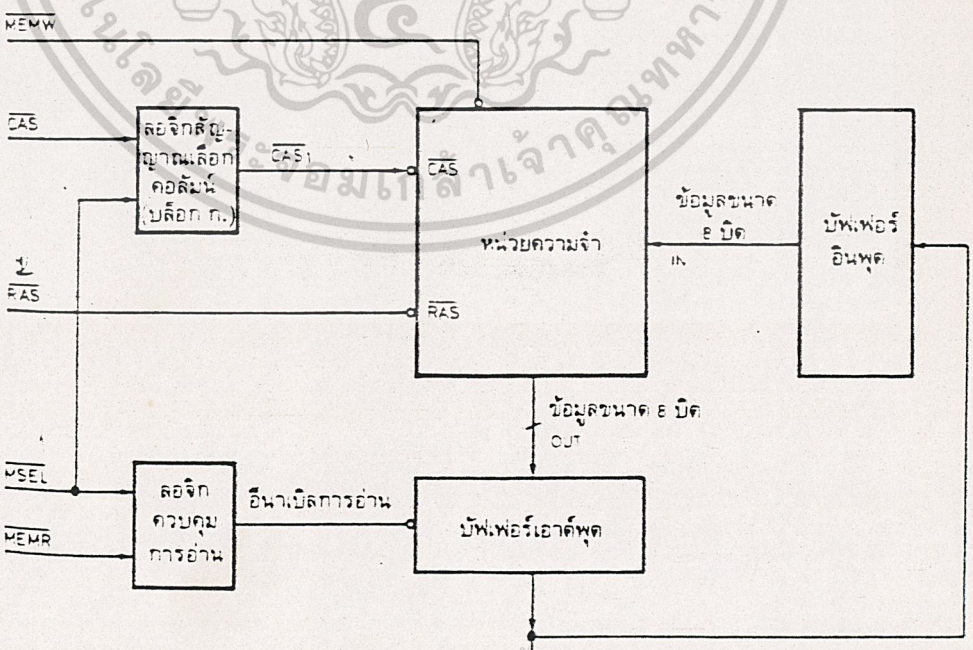


รูปที่ 4.1.4 แสดงถึงผังเวลาสำหรับการกำหนดแอดเดรสในหน่วยความจำ

74LS157 ก็จะเลือกเอาข้อมูลทางแนวนอนที่แอดเดรส A0 16 เข้าสู่ DRAM ก่อนและสัญญาณ RAS ก็จะแอดตีฟ "0" เป็นการแลทช์เอาแอดเดรสทางแนวนอนไว้ในตัว DRAM ก่อน จากนั้น MUX ก็จะเป็น "0" เป็นการเลือกให้แอดเดรสทางแนวตั้ง A7 A13 เข้าสู่แอดเดรสขาเดิมของ DRAM และสัญญาณ CAS ก็จะแอดตีฟ "0" เป็นการแลทช์เอาข้อมูลทางแนวตั้งเข้าไปไว้ในตัว DRAM (ต่อไป DRAM ก็จะถอดระหัสแอดเดรสนั้นต่อไป)



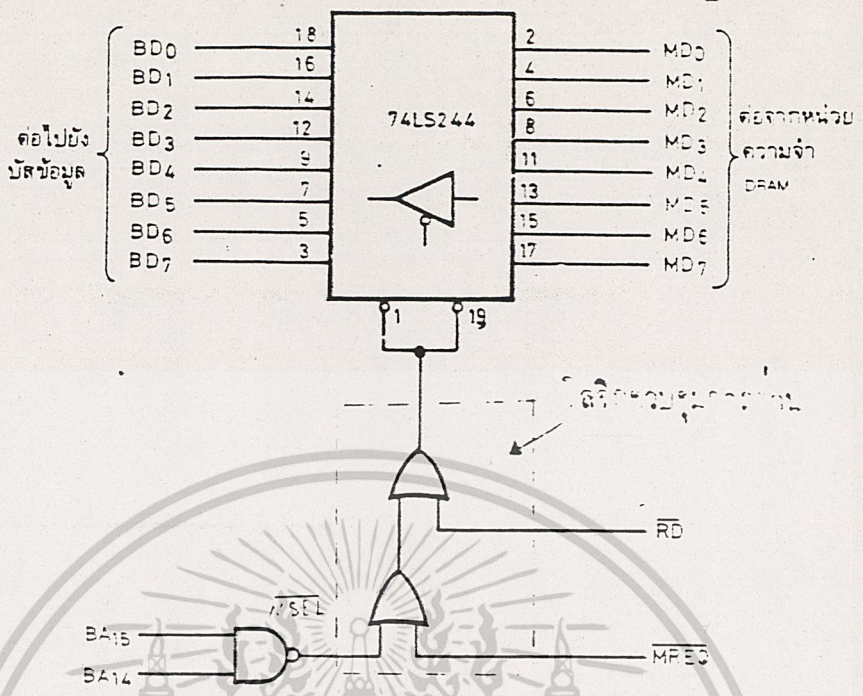
รูปที่ 4.1.5 แสดงถึงวงจรใช้งานและลำดับสัญญาณ RAS, MUX, CAS การอ้างแอดเดรส



รูปที่ 4.1.6 แสดงถึงบล็อกไดอะแกรมของการต่อ DRAM โดยทั่วไปกับระบบไมโครฯ

ซึ่งสัญญาณต่างๆ ที่กล่าวมา RAS,CAS,MUX เราจะต้องสร้างเองซึ่งจะได้กล่าวต่อไป ตอนเรามาดูถึงบล็อกไดอะแกรมของการต่อ DRAM เข้าสู่ระบบไมโครโปรเซสเซอร์ซึ่งโดยทั่วไปจะเป็นดังรูปที่ 4.1.6

จากรูปพอจะกล่าวคร่าวก่อนดังนี้ (เมื่อได้เห็นการออกแบบวงจรควบคุม DRAM ก็จะสามารถเข้าใจได้ดียิ่งขึ้น) เราจะต้องสร้างสัญญาณ MEMW,MEMR,MSEL ซึ่งสัญญาณเหล่านี้เราเคยสร้างมาแล้วจากหัวข้อของการต่อ SRAM ส่วนสัญญาณที่เพิ่มเข้ามาคือ RAS, CAS โดยสัญญาณของ RAS จะถูกต่อเข้าสู่ DRAM โดยตรง แต่สัญญาณ CAS นั้นจะต้องมารวมกับสัญญาณ MSEL ก่อนจึงได้สัญญาณ CAS1 (ซึ่งก็คือสัญญาณ CAS เดิม) เข้าสู่ตัว DRAM เหตุเป็นเช่นนี้เพราะว่าสัญญาณ MSEL คือสัญญาณที่เลือกตำแหน่งการวางพื้นที่หน่วยความจำเราจะนำมากำหนดให้สัญญาณ CAS เข้าสู่ DRAM ได้ก็ต่อเมื่อมีการอ้างแอดเดรสของ DRAM นี้ในแอดเดรสที่ถูกต้องที่เราทำการดีโค้ด (decode) ไว้เท่านั้นหากการอ้างแอดเดรสไม่ตรงกับค่าที่เราดีโค้ดไว้เราก็จะไม่ให้มีสัญญาณ CAS1 เข้าสู่ DRAM ตัว DRAM ก็จะไม่สามารถทำงานได้นั่นเองและจากรูปเนื่องจากว่าขาข้อมูลอินพุทและเอ้าพุทของ DRAM แยกกันคนละชุด เราจึงต้องมีการบัฟเฟอร์เพื่อจะได้แยกเป็นอินพุทส่วนหนึ่งและเอ้าพุทส่วนหนึ่งให้กับ DRAM โดยได้รับสัญญาณการอินาเบิ้ลอย่างถูกต้องตามสภาวะการของการเขียนอ่านข้อมูลจากส่วน <u>บล็อกควบคุมโลจิกการอ่าน</u> ในส่วนของบัฟเฟอร์นี้หากเป็นบัฟเฟอร์ด้านอินพุทข้อมูลเข้าสู่ DRAM เราสามารถใช้ TTL(74LS244) ต่ออินาเบิ้ลบัฟเฟอร์นี้ได้เนื่องจากว่า DRAM จะสนใจข้อมูลอินพุทกรณีที่เขียนข้อมูลเท่านั้น นอกนั้นข้อมูลทางอินพุทจะไม่มีผลใดๆต่อ DRAM ในด้านบัฟเฟอร์เอ้าพุทข้อมูลออกจาก DRAM เราจะต้องมีสัญญาณการอินาเบิ้ลที่มาจาก <u>บล็อกโลจิกควบคุมการอ่าน</u> (ในรูปที่ 4.1.6) เพราะข้อมูลที่ออกจาก DRAM ต้องถูกเปิดให้ผ่านบัฟเฟอร์ออกมาในเวลาที่ถูกต้องเพราะอาจไปกวนต่ออุปกรณ์ตัวอื่นที่ใช้ระบบบัสอันเดียวกันได้ ในรูปที่ 4.1.7 ซึ่งในแต่ละบล็อกจะได้อธิบายต่อไป

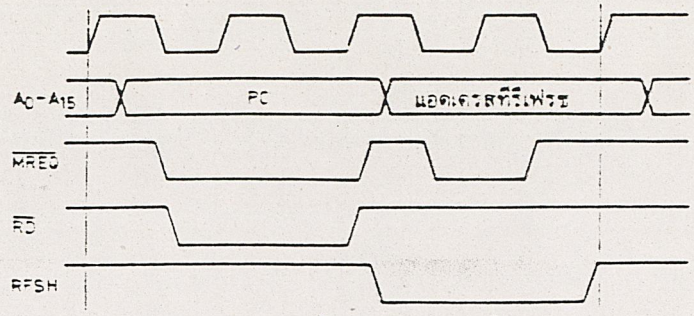


รูปที่ 4.1.7 แสดงถึงบัฟเฟอร์ที่ใช้ในระบบบัส เอ้าพุทของ DRAM

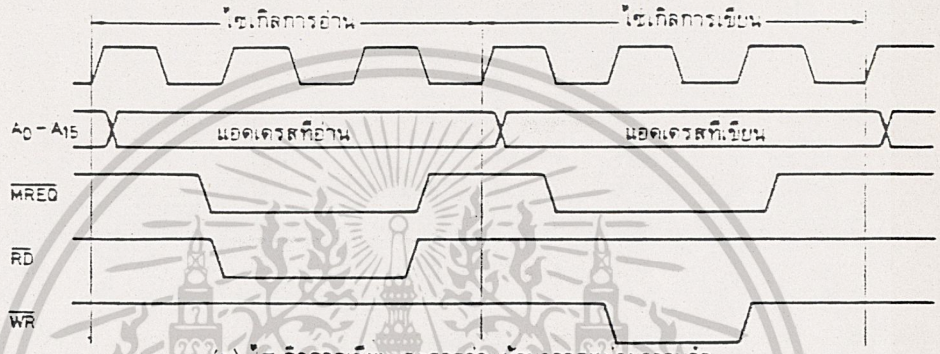
4.2 การแสดงสัญญาณควบคุมการอ่านเขียนของ DRAM

ในการสร้างสัญญาณ RAS,MUX,CAS นั้นเราก็นำมาจากความรู้จากเรื่องไดอะแกรมเวลาของการทำงานของ Z-80 ในช่วงแมทชีนไซเกิ้ลต่างๆ ซึ่งมีลำดับสัญญาณที่แน่นอนอยู่แล้ว โดยการใช้งานของ DRAM นั้นจะเกี่ยวข้องอยู่สามแมทชีนไซเกิ้ล โดยจะยกเอาสัญญาณที่ต้องใช้งานมาแสดงดังรูปที่ 4.2.1

ในการสร้างสัญญาณ RAS,MUX,CAS นั้นจากไดอะแกรมเวลาของลำดับสัญญาณในรูปที่ 4.1.5 นั้นสัญญาณ RAS จะแอดตีฟก่อนจากนั้นก็เป็น MUX (เพื่อมัลติเพล็กแอดเดรส) และต่อมาเป็นสัญญาณ CAS ซึ่งโดยปกติสัญญาณ MUX จะดีเลย์จากสัญญาณ RAS ประมาณ 50 nS และสัญญาณ CAS จะดีเลย์จาก MUX ไปประมาณ 150 nS ดังนั้นเราจึงต้องสร้างวงจรส่วนหน่วงเวลาดังรูปที่ 4.2.2

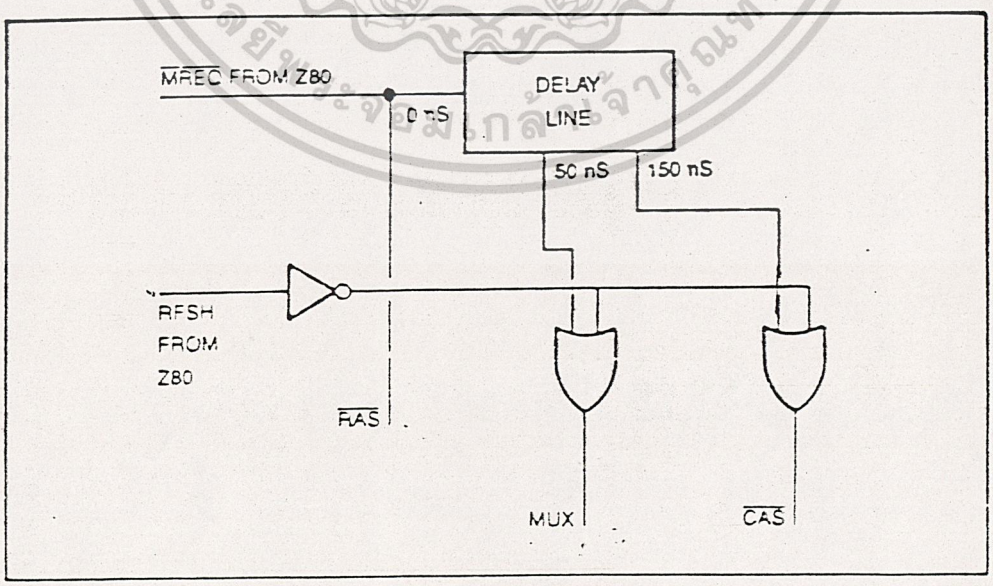


(ก) แผนผังเวลาของไซเกิล M1 ขณะเฟตช์ข้อมูล



(ข) ไซเกิลการเขียนและการอ่านข้อมูลจากหน่วยความจำ

รูปที่ 4.2.1 แสดงถึงแมทชีนไซเกิ้ลที่ใช้การทำงานของ DRAM ในรูป (ก) เป็นผังเวลาของ M1 ขณะที่เฟทช์ข้อมูลซึ่งส่วนของสัญญาณที่ใช้กับ DRAM คือช่วงของเวลาของการ รีเฟรช (T3,T4) ซึ่งจะกล่าวอีกครั้งถึงการรีเฟรชหน่วยความจำไดนามิกแรม

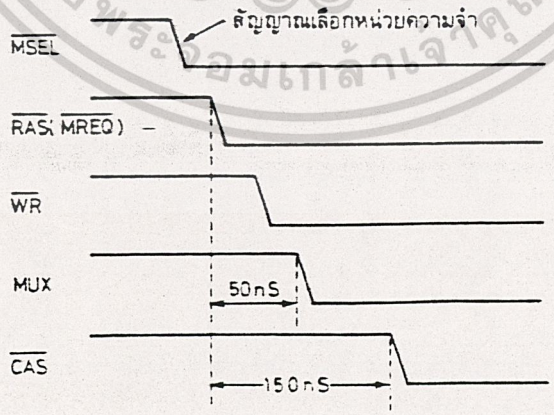


รูปที่ 4.2.2 แสดงถึงลำดับสร้างสัญญาณควบคม

จากรูปสัญญาณ MREQ ถูกนำมาสร้างเป็นสัญญาณ RAS เพราะหลังจาก PC ของ CPU ได้ให้ค่าแอดเดรสออกมา สัญญาณ MREQ ก็จะออกตามมาซึ่งเราก็นำมาใช้เป็น RAS เลยเพื่อให้ DRAM แลทช์แอดเดรสแนวนอนไว้ก่อนจากนั้นเราก็หน่วงสัญญาณ MREQ หรือ RAS ไปอีก 50 nS เพื่อให้เป็นสัญญาณ MUX ไปมัลติเพล็กตัว 74LS157 เลือกแอดเดรสแนวตั้งเข้าสู่ DRAM จากนั้นสัญญาณ MUX ก็จะถูกหน่วงไป 100 nS เพื่อให้เกิดสัญญาณ CAS สู่ DRAM ให้แลทช์แอดเดรสทางแนวตั้งนั้นไว้ ส่วนสัญญาณ RFSH นั้นจะนำมาผ่านอินเวอร์เตอร์ก่อนเข้าสู่ OR GATE ซึ่งหมายความว่า สัญญาณ MUX,CAS จะไม่แอดตีฟผ่าน OR GATE ออกมาได้หากอยู่ในช่วงมีสัญญาณการ รีเฟรช (T3,T4 ของ M1) เพราะในการรีเฟรชหน่วยความจำไดนามิกแรมจะใช้เฉพาะสัญญาณ RAS เท่านั้น

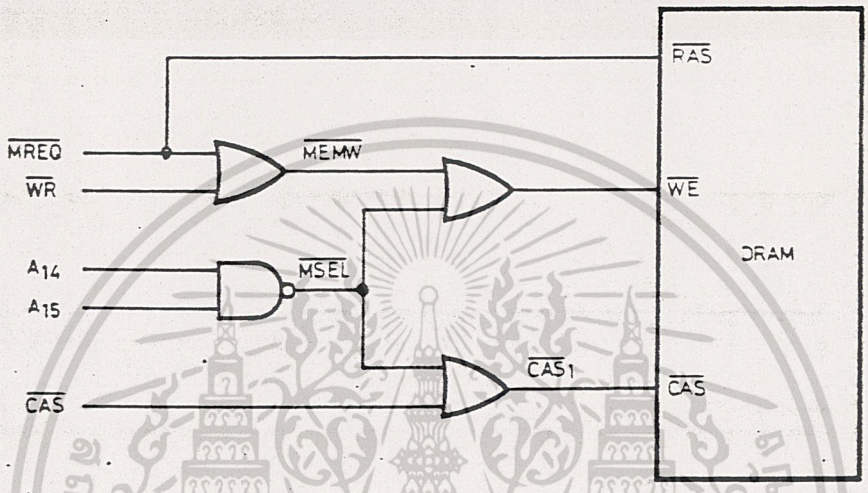
* การเขียนข้อมูลเข้าสู่ DRAM

ผังเวลาของการเขียนข้อมูลเข้าสู่หน่วยความจำไดนามิกแรมนั้นปกติไดอะแกรมเวลาจะเป็นไปตามรูปที่ 4.2.3 แต่เมื่อนักศึกษาจะใช้งานกับ DRAM จริงๆ ต้องศึกษาถึงไดอะแกรมของการอ่านเขียนตัว DRAM ที่จะใช้งานนั้นได้จากคู่มือซึ่งทั่วไปจะมีลักษณะคล้ายกันทุกเบอร์ และในตอนท้ายของหัวข้อ DRAM นี้ก็จะมีตัวอย่างของการออกแบบการใช้งาน DRAM



รูปที่ 4.2.3 แสดงถึงไดอะแกรมเวลาทั่วไปในการเขียนข้อมูลเข้าสู่ DRAM

ตามบล็อกที่ผ่านมารูปที่ 4.1.6 เราจะต้องสร้างบล็อกของ โลจิกสัญญาณเลือกคอลัมน์ ให้ได้สัญญาณเข้าสู่ DRAM ตามไดอะแกรมเวลาดังรูปข้างบนเมื่อเขียนข้อมูลเข้า DRAM โดยการประกอบวงจรง่ายๆ ด้วย OR GATE ดังแสดงในรูปที่ 4.2.4



รูปที่ 4.2.4 แสดงถึงวงจรง่ายๆ ของบล็อก โลจิกสัญญาณเลือกคอลัมน์

สมมุติให้ DRAM 4116 ซึ่งเป็นแรมขนาด 16 K ให้อยู่ในตำแหน่งที่ C000H-FFFFH เราก็จะสามารถสร้าง MSEL ได้ง่ายโดยการใช้ A14,A15 มาผ่าน NAND GATE นั่นหมายความว่าเมื่อมีการอ้างแอดเดรสที่ C000H-FFFFH (A14=1,A15=1) เท่านั้นจึงจะได้เอ้าพุทแอดตีฟ "0" (MSEL) สู่ชุด OR GATE ตัวต่อมา โดยขั้นตอนการเขียนสามารถอธิบายตามไดอะแกรมรูปที่ 4.2.3 ดังนี้

1.สัญญาณ MSEL จะแอดตีฟก่อนเนื่องจากว่าสร้างมาจากการดีโค้ด PC (Program counter)

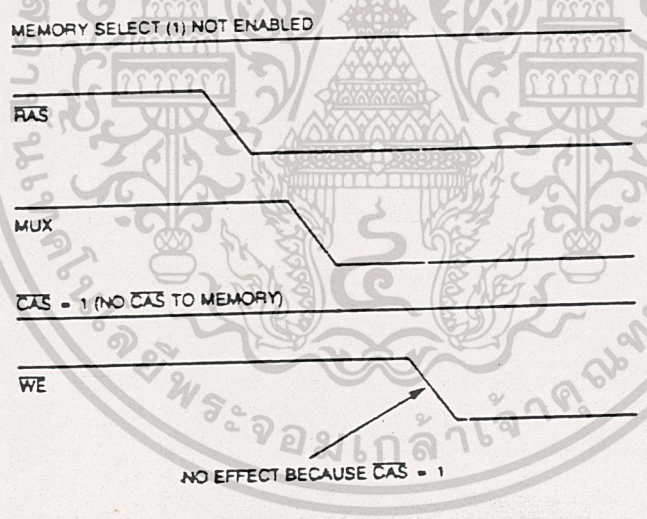
2.สัญญาณการเขียน RAS ซึ่งสร้างมาจาก MREQ ก็จะแอดตีฟตามมาเป็นการบอกให้ DRAM แลทช์แอดเดรสทางแนวนอนไว้

3.สัญญาณการเขียน WR จาก CPU จะแอดตีฟเป็นการกำหนดให้ DRAM เขียนข้อมูลแต่ยังไม่เขียน จนกว่าจะได้รับสัญญาณ CAS เรียบร้อยก่อน

4.สัญญาณ MUX ที่ดีเลย์มาจาก RAS จะแอดตีฟ "0" เป็นการทำให้ส่วนมัลติเพล็ก (74LS157) เลือกเอาสัญญาณ A7 A13 เข้าสู่ตัว DRAM

5.สัญญาณ CAS จะแอดตีฟเป็นการให้ DRAM แลทช์แอดเดรสแนวตั้งไว้ ต่อจากนั้นก็จะทำการเขียนข้อมูลเข้าสู่ DRAM ได้

หากตัวไมโครโปรเซสเซอร์ทำการเขียนเข้าสู่ตำแหน่งหน่วยความจำที่ไม่ใช่ของไดนามิกแรมจะทำให้สัญญาณ MSEL ไม่มีเป็นผลทำให้สัญญาณ CAS ไม่มีเข้าสู่ DRAM ฉะนั้นถึงแม้สัญญาณ RAS และ WE จะมีเกิดขึ้นเข้าสู่ตัว DRAM ก็จะไม่มีการทำงานอ่าน/เขียนหน่วยความจำ

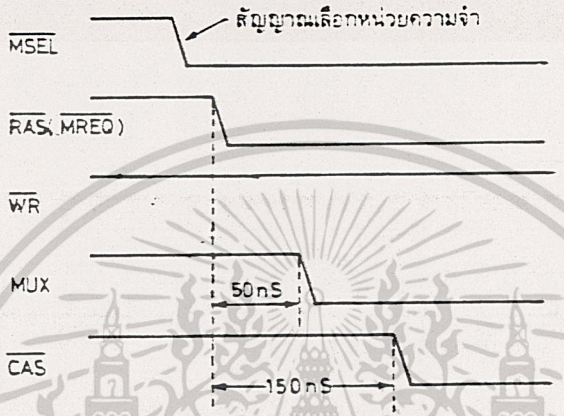


รูปที่ 4.2.5 แสดงถึงสัญญาณที่เกิดขึ้นที่ตัว DRAM กรณีทำการอ่านเขียนไม่ตรงแอดเดรสของ DRAM (C000H-FFFFH)

* การอ่านข้อมูลจากหน่วยความจำ DRAM

ในการข้อมูลจากตัว DRAM นี้จะต้องมีสัญญาณอีนาเบิ้ลให้บัฟเฟอร์ที่เอ้าพุทบัสข้อมูลของ DRAM ให้ปล่อยข้อมูลออกมา ซึ่งได้แสดงวงจรส่วนนี้ไปแล้วในรูปที่ 4.1.7

ส่วนสัญญาณควบคุมที่เข้าสู่ DRAM ก็จะมีลักษณะที่คล้ายกับการเขียนข้อมูลเข้าสู่ DRAM นั่นเอง คือสัญญาณ RAS,MUX,CAS จะเหมือนกับกรณีของการเขียนข้อมูล หากแต่สัญญาณ WE จะไม่แอดติฟเป็นผลทำให้ข้อมูลออกจากตัว DRAM ผ่านบัฟเฟอร์ ออกสู่บัสข้อมูลเพื่อให้ CPU อ่านเข้าสู่รีจิสเตอร์ภายในตัวมันต่อไป



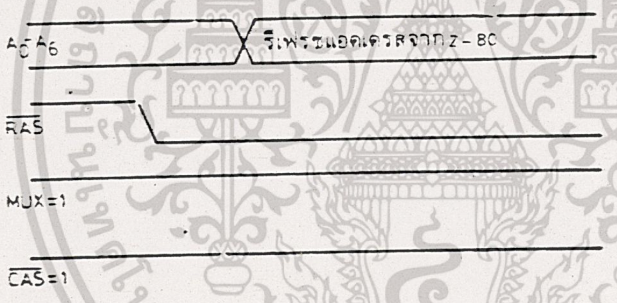
รูปที่ 4.2.6 แสดงไดอะแกรมเมื่อทำการอ่านข้อมูลจาก DRAM

4.3 การรีเฟรชหน่วยความจำ

ตัวของไดนามิกแรมแม้จะมีข้อดีอยู่หลายอย่างคือความจุสูงและราคาที่ต่ำมาก แต่ก็มีข้อเสียที่สำคัญมากคือ มันไม่สามารถที่จะเก็บข้อมูลอยู่ได้นานแม้จะมีไฟเลี้ยงระบบอยู่ก็ตามหากไม่ได้รับการรีเฟรชข้อมูลให้คงอยู่ เนื่องจากไดนามิกแรมนี้มีโครงสร้างเป็นลักษณะของตัวเก็บประจุ ดังนั้นเพื่อจะไม่ให้ประจุในตัวของไดนามิกนั้นรั่วไหลออกไปจนทำให้สถานะโลจิกเปลี่ยนไป จึงต้องมีการซ้ำข้อมูลเป็นระยะๆ และขบวนของการซ้ำข้อมูลก็คือการรีเฟรชนั่นเอง และในแต่ละ CELL ประจุจะต้องถูกรีเฟรชซ้ำทุกๆ 2-3 mS (แล้วแต่ข้อมูลของ DRAM แต่ละตัว) เพื่อไม่ให้ข้อมูลเสียไป

วิธีการรีเฟรชมีอยู่หลายวิธี ในตัวอย่างที่เราใช้ DRAM เบอร์ 4116 นี้ในแผ่นข้อมูลบอกว่าระยะเวลารีเฟรช 2 mS และในการรีเฟรชนั้นเราสามารถรีเฟรชแต่ละครั้งเป็นแถวทั้งแถวได้เลย ฉะนั้นเบอร์ 4116 มี 128 แถว (สังเกตุได้จาก แอดเดรสทางแนวนอนมี A0 A6) เพราะฉะนั้นในแต่ละแถวเราก็ต้องใช้เวลารีเฟรช 2/128 mS จะ

ได้เวลาประมาณ 16 uS และการให้แอดเดรสรีเฟรชทำได้ง่ายเพราะ Z-80 ออกแบบให้ทุก M1 จะมีช่วงการให้แอดเดรสรีเฟรชของหน่วยความจำ DRAM ออกมาเสมอและจะเพิ่มค่าเองโดยอัตโนมัติเพื่อจะได้ค่าแอดเดรสในช่วงของ M1 ครั้งต่อไป Z-80 จะให้ค่าแอด-เดรสออกที่ A0 A6 ซึ่งก็จะให้ค่าแอดเดรสได้ขนาด 128 ครั้งพอดี และค่าแอดเดรสนี้ก็จะออกมาในช่วงของ T3,T4 ซึ่งช่วงนี้สัญญาณ RFSH จะแอดติฟโลจิก "0" ด้วย และในการรีเฟรชนั้น ตัว DRAM จะต้องการเพียงสัญญาณ RAS ก็เพียงพอแล้ว โดยจะต้องไม่ให้สัญญาณ MUX ,CAS เกิดขึ้นเพราะจะเป็นการอ่านข้อมูลทำให้ข้อมูลออกมาสู่ระบบบัสได้ ดังนั้นเราจึงใช้สัญญาณ RFSH จาก Z-80 มาควบคุมสัญญาณ MUX,CAS ไม่ให้เข้าสู่ DRAM ในช่วงรีเฟรช



รูปที่ 4.3.1 แสดงถึงสัญญาณที่เข้าสู่ DRAM ในช่วงของการรีเฟรช

เราทราบว่าสัญญาณการ รีเฟรช DRAM จะมีออกมาในช่วงสุดท้ายของแมทชีนไซเกิ้ลของการเฟทช์ข้อมูลหรือทุก M1 ของแต่ละคำสั่ง เราทราบว่าจะต้องมีการรีเฟรช DRAM 4116 ทุกๆ 16 uS ฉะนั้นในแต่ละคำสั่ง จึงต้องใช้เวลาที่ห่างกัน ไม่เกิน 16 uS จึงเป็นผลให้เราต้องคำนึงถึงความเร็วของระบบฐานเวลา (CLOCK) ที่จะใช้ป้อน ด้วย เช่น หากระบบเราใช้ฐานเวลาคือ 4 MHs และดูจากคำสั่ง นั้นมีคำสั่งที่ใช้จำนวนของ T state มากสุดคือ 23T เราก็ทราบว่าช่วงเวลาที่คำสั่งที่จำนวน T มากที่สุดนี้ใช้ประมาณ 250 nsx23=5.7us นั่นแสดงว่าระบบที่ใช้ CLOCK = 4 MHz สามารถใช้ DRAM 4116 ได้ ในการใช้งานต้องคำนึงถึงช่วงเวลาของการ WAIT,DMA ด้วยเพราะช่วงเวลาดังกล่าวจะไม่มีการรีเฟรชหน่วยความจำ

4.4 ตัวอย่างของการออกแบบใช้งาน DRAM

ในตัวอย่างต่อไปนี้เป็นการอธิบายถึงการออกแบบเป็นขั้นตอนโดยได้นำวงจรที่สามารถใช้งานจริงมาพิจารณาออกแบบ และนักศึกษาสามารถนำไปประยุกต์พัฒนาใช้งานกับ DRAM เบอร์ต่างๆ ได้ตามต้องการ **โจทย์หรือสิ่งที่ต้องการ** ต้องการออกแบบใช้หน่วยความจำกับระบบไมโครโปรเซสเซอร์ ที่ใช้ COLCK ของระบบขนาด 4 MHz โดยต้องการใช้หน่วยความจำขนาด 32 KB ในตำแหน่ง 7000H-EFFFH โดยต้องการหน่วยความจำตำแหน่งดังกล่าวนี้จำนวน 2 เพจ ซ้อนกัน ?

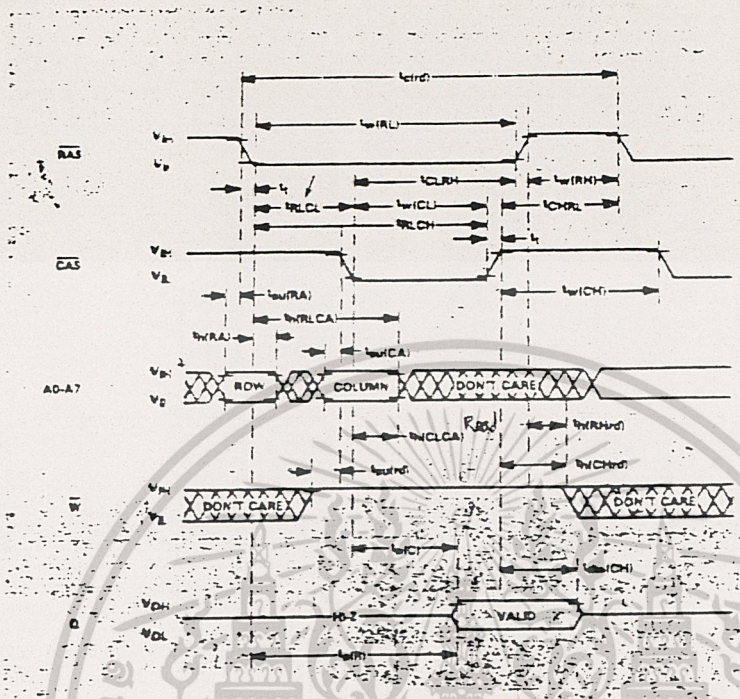
วิธีทำ

1.พิจารณาเบอร์ของ DRAM ที่จะนำมาใช้งาน

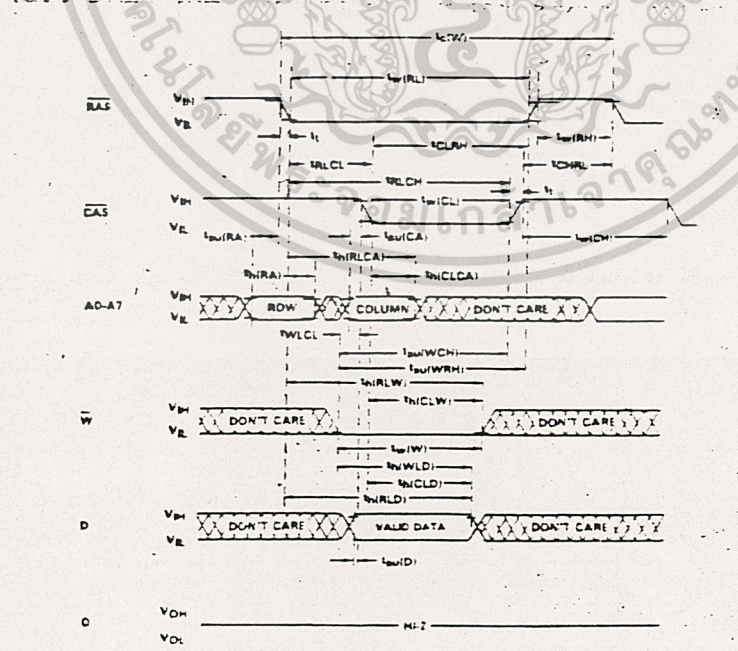
เราเลือกใช้แรมขนาด 64K เบอร์ 4164-12 เพราะแบ่งใช้ได้เพจละ 32K พอดี และค่า access time ประมาณ 120 ns ซึ่งมีค่าเพียงพอกับการอ่านเขียนในระบบที่ใช้ CLOCK 4MHz ซึ่งช่วงอ่านเขียนนั้น ใช้จำนวน 3T= 3 (255ns) = 750ns (มากกว่าค่า access time)

2. พิจารณาช่วงเวลาสำคัญจาก DATA BOOK

PARAMETER		ALT. SYMBOL	TMS 4164-12		TMS 4164-15		TMS 4164-20		TMS 4164-25		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{c(P)}$	Page mode cycle time	t_{PC}	140		160		225		275		ns
$t_{c(rd)}$	Read cycle time	t_{RC}	230		260		330		410		ns
$t_{c(W)}$	Write cycle time	t_{WC}	230		260		330		410		ns
$t_{c(rdW)}$	Read-write/read-modify-write cycle time	t_{RWC}	260		285		345		455		ns
$t_{w(CH)}$	Pulse width, $\overline{CAS}$ high (precharge time)	t_{CP}	50		50		80		100		ns
$t_{w(CL)}$	Pulse width, $\overline{CAS}$ low†	t_{CAS}	75	10,000	100	10,000	135	10,000	165	10,000	ns
$t_{w(RH)}$	Pulse width, $\overline{RAS}$ high (precharge time)	t_{RP}	100		100		120		150		ns
$t_{w(RL)}$	Pulse width, $\overline{RAS}$ low‡	t_{RAS}	120	10,000	150	10,000	200	10,000	250	10,000	ns
$t_{w(W)}$	Write pulse width	t_{WP}	45		45		55		75		ns
t_t	Transition times (rise and fall) for $\overline{RAS}$ and $\overline{CAS}$	t_T	3	50	3	50	3	50	3	50	ns
$t_{su(CA)}$	Column address setup time	t_{ASC}	0		−5		−5		−5		ns
$t_{su(RA)}$	Row address setup time	t_{ASR}	0		0		0		0		ns
$t_{su(D)}$	Data setup time	t_{DS}	0		0		0		0		ns
$t_{su(rd)}$	Read command setup time	t_{RCS}	0		0		0		0		ns
$t_{su(WCH)}$	Write command setup time before $\overline{CAS}$ high	t_{CWL}	50		60		80		100		ns
$t_{su(WRH)}$	Write command setup time before $\overline{RAS}$ high	t_{RWL}	50		60		80		100		ns
$t_{h(CLCA)}$	Column address hold time after $\overline{CAS}$ low	t_{CAH}	45		45		55		75		ns
$t_{h(RA)}$	Row address hold time	t_{RAH}	15		20		25		35		ns
$t_{h(RLCA)}$	Column address hold time after $\overline{RAS}$ low	t_{AR}	90		95		140		190		ns
$t_{h(CLD)}$	Data hold time after $\overline{CAS}$ low	t_{DH}	50		60		80		110		ns
$t_{h(RLD)}$	Data hold time after $\overline{RAS}$ low	t_{DHR}	95		110		145		195		ns
$t_{h(WLD)}$	Data hold time after W low	t_{DH}	45		45		55		75		ns
$t_{h(CHrd)}$	Read command hold time after $\overline{CAS}$ high	t_{RCH}	0		0		0		0		ns
$t_{h(RHrd)}$	Read command hold time after $\overline{RAS}$ high	t_{RRH}	5		5		5		5		ns
$t_{h(CLW)}$	Write command hold time after $\overline{CAS}$ low	t_{WCH}	50		60		80		110		ns
$t_{h(RLW)}$	Write command hold time after $\overline{RAS}$ low	t_{WCR}	95		110		145		195		ns
t_{RLCH}	Delay time, $\overline{RAS}$ low to $\overline{CAS}$ high	t_{CSH}	120		150		200		250		ns
t_{CHRL}	Delay time, $\overline{CAS}$ high to $\overline{RAS}$ low	t_{CRP}	0		0		0		0		ns
t_{CLRH}	Delay time, $\overline{CAS}$ low to $\overline{RAS}$ high	t_{RSH}	80		100		135		185		ns
t_{CLWL}	Delay time, $\overline{CAS}$ low to W low (read, modify-write-cycle only)	t_{CWD}	50		60		85		105		ns
t_{RLCL}	Delay time, $\overline{RAS}$ low to $\overline{CAS}$ low (maximum value specified only to guarantee access time)	t_{RCD}	15	45	20	50	25	65	35	85	ns
t_{RLWL}	Delay time, $\overline{RAS}$ low to W low (read, modify-write-cycle only)	t_{RWD}	85		110		130		190		ns
t_{WLCL}	Delay time, W low to $\overline{CAS}$ low (early write cycle)	t_{WCS}	−5		−5		−5		−5		ns
t_{rf}	Refresh time interval	t_{REF}		4		4		4		4	ms



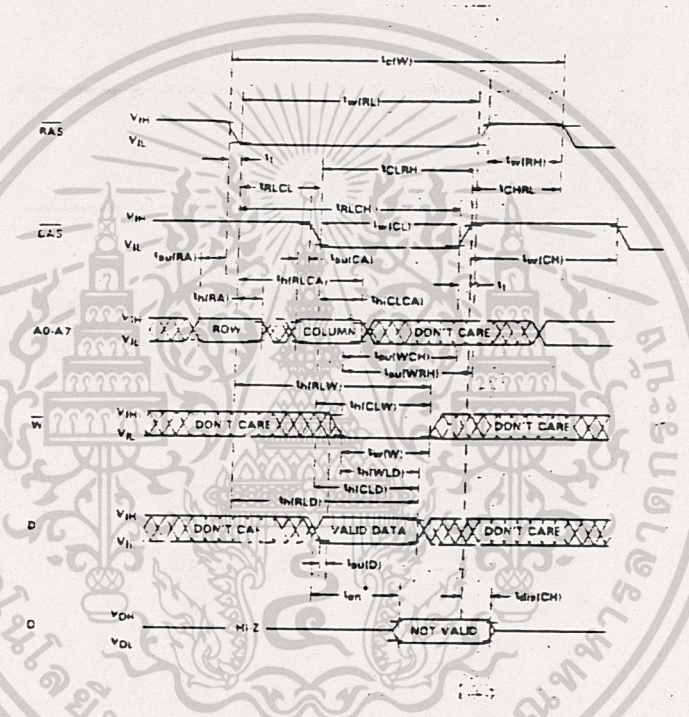
Read cycle timing



Early write cycle timing

ให้ดูที่ T_{RAS} คือความกว้างของพัลซ์ RAS ซึ่งก็คือเวลาของ ACCESS TIME นั่นเองซึ่งมีค่า 120 ns ตามต้องการ และดูที่ T_{RICL} (ที่ลูกศรชี้) คือระยะห่างระหว่างขอบขาลงของ RAS และ CAS จะเห็นว่ามีค่าอย่างมากสุดที่เชื่อได้ว่าทำงานได้ (MAX)= 45 ns เวลาออกแบบเราต้องใช้ช่วงเวลามากกว่านี้เพื่อความชัวร์ว่าทำงานแน่ๆ

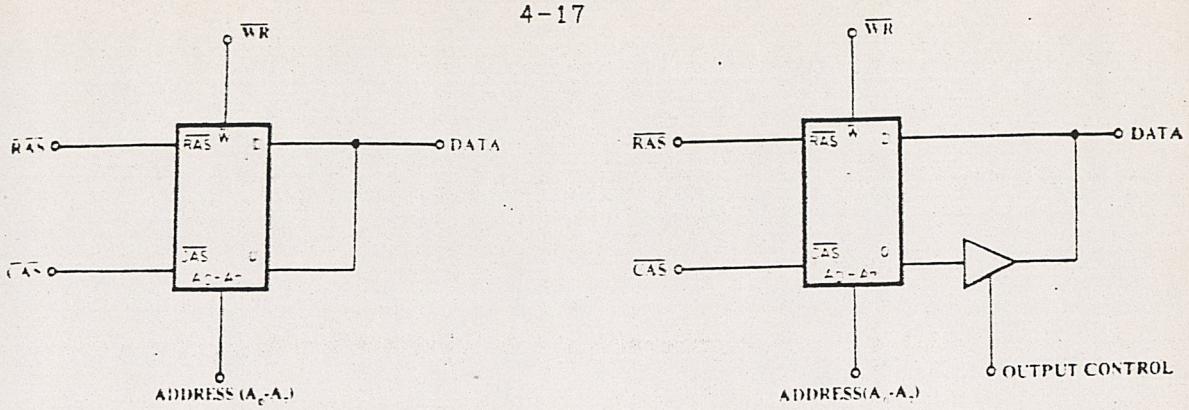
ช่วงเวลาการอ่านและเขียน จากไดอะแกรมของ 4161-12 แสดงดังรูปที่ 4.4.2 ข้างล่าง



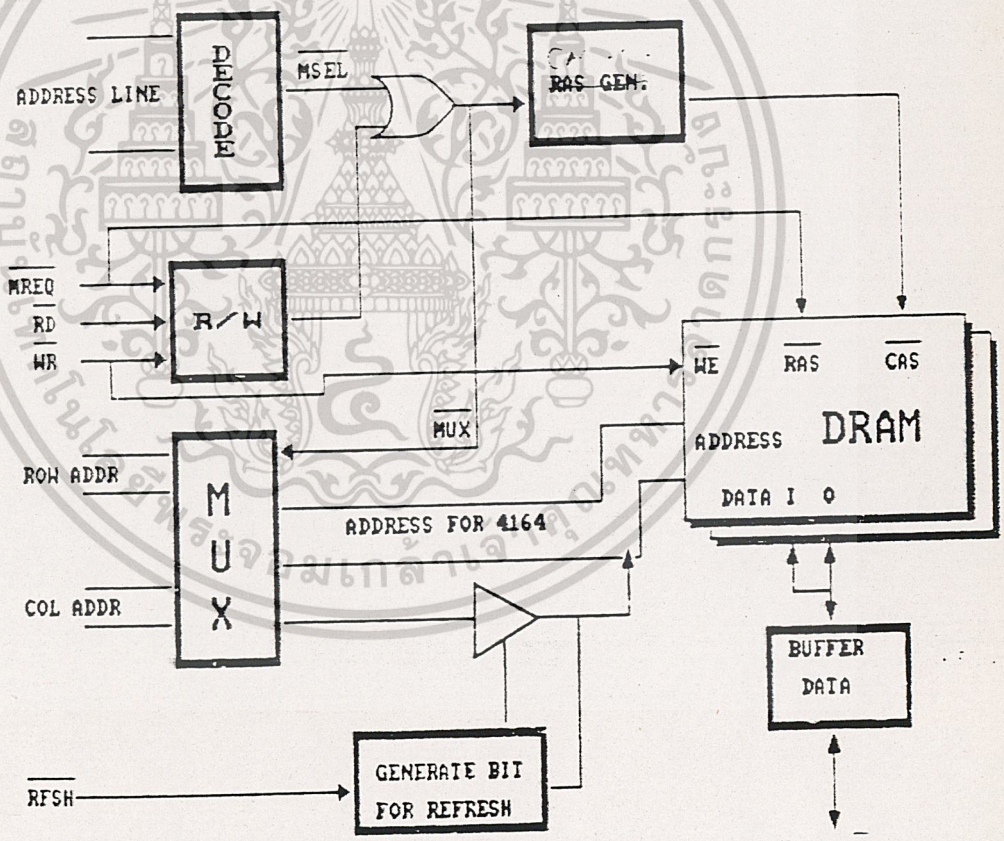
รูปที่ 4.4.2

Write cycle timing (Modify)

จากไดอะแกรมที่ผ่านมา ในช่วงของการอ่านนั้นไม่มีปัญหาใดๆ เพราะไซเกิ้ลก็ธรรมดาแต่ในช่วงของการเขียนนั้นจะเห็นว่าแรมสามารถทำงานได้สองแบบ **ในระบบแรกเป็นการทำงานเขียนแบบ Early write cycle timing** แบบนี้สังเกตุว่าพัลซ์ของการเขียน W จะแอดติฟ "0" ก่อนที่ CAS จะแอดติฟ ข้อมูลที่จะสามารถเขียนเข้าสู่แรมได้ในช่วง VALID DATA และที่เอ้าพุทของแรมเองจะไม่มีสัญญาณใด ๆ ออกมาคือเป็น Hi-Z (อิมพีแดนซ์สูง) ส่วนในแบบที่สองนั้นสัญญาณ w จะเข้ามาทีหลังสัญญาณ CAS ข้อมูลที่เข้าส่แรมก็สามารถที่จะถกเขียนเข้าไปได้ แต่สังเกตว่าที่เอ้าพทของแรมจะมีค่าของข้อ



รูปที่ 4.4.3 แสดงรูปด้านซ้ายที่ประหยัดบัฟเฟอร์



รูปที่ 4.4.4 แสดงบล็อกมาตฐานที่ใช้ออกแบบ

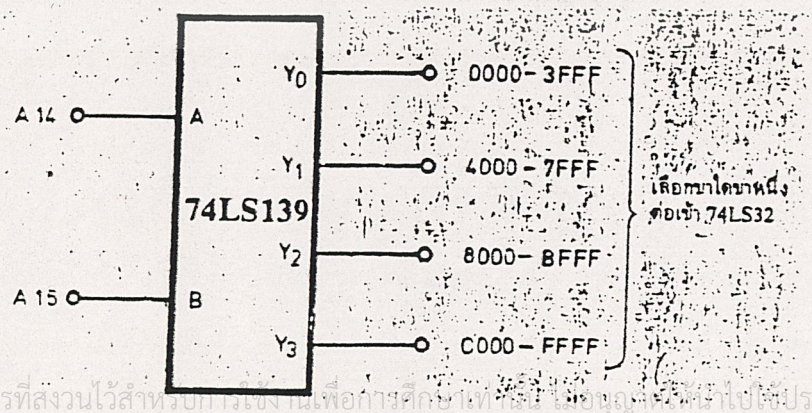
มูลออกมาซึ่งเป็นข้อมูลที่ไม่แน่นอน (NOT VALID) ซึ่งหากจะเปรียบเทียบการใช้งานแล้ว เราน่าจะใช้ลักษณะของการเขียนในแบบแรกมากกว่าเพราะว่าจะไม่มีข้อมูลเอ้าพุทออกมาช่วงที่เขียนทำให้ไม่มีการรบกวนข้อมูลเป็นผลทำให้ไม่ต้องใส่บัฟเฟอร์ที่ขาข้อมูลของแรมเป็นการประหยัด ส่วนในแบบที่สองเราต้องใส่บัฟเฟอร์เพราะไม่ให้ข้อมูลเกิดกวนกัน

3.) พิจารณาลักษณะของบล็อกมาตฐานที่ใช้งานจริง

ในรูปต่อไปนี้เป็นลักษณะของบล็อกมาตฐานที่ใช้งานจริงแสดงดังรูปที่ 4.4.4 ความจริงนักศึกษาอาจออกแบบเป็นลักษณะอื่นก็ได้ แต่ที่เลือกเอาแบบนี้ก็เพราะเป็นแบบที่ง่ายและหาอุปกรณ์ได้ง่าย สามารถประยุกต์ใช้งานกับเบอร์อื่นได้ทันทีโดยแก้ไขเพียงเล็กน้อย และในบล็อกเหล่านั้นเราจะใส่วงจรเข้าไปเลย และในตอนสุดท้ายของหัวข้อก็จะเป็นรูปวงจรที่สำเร็จใช้งานตามความต้องการ

DECODE หากเราต้องการตำแหน่งขาของแอดเดรสที่ตายตัวเราก็สามารถทำได้ง่ายโดยใช้อุปกรณ์ TTL ดิโค็ดดังที่เคยกล่าวมาในเรื่องของ SRAM เช่นเบอร์ 74LS139 ซึ่งเป็นการแบ่งขนาด 16 KB เป็นจำนวน 4 เอ้าพุทโดยใช้ A14,A15 มาทำการดิโค็ด ยิ่งหากต้องการขนาด 32 KB ก็สามารถทำได้ง่าย โดยใช้ inverter ตัวเดียวดังรูปที่ 4.4.5

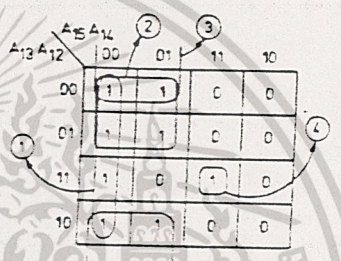
แต่หากเราต้องการแอดเดรสที่ไม่ตายตัวเช่น 7000H-EFFFH (จากโจทย์) เราก็สามารถทำได้โดยพิจารณาว่าด้วยหน่วยความจำขนาด 32K ขาแอดเดรสบิทที่จะนำมาดิโค็ดนั้นจะอยู่ใน 4 บิทสุดท้าย คือ A12,A13,A14,A15 แล้วนำมาทำตารางเพื่อ MAP เอ้าพุทที่ต้องการดังนี้



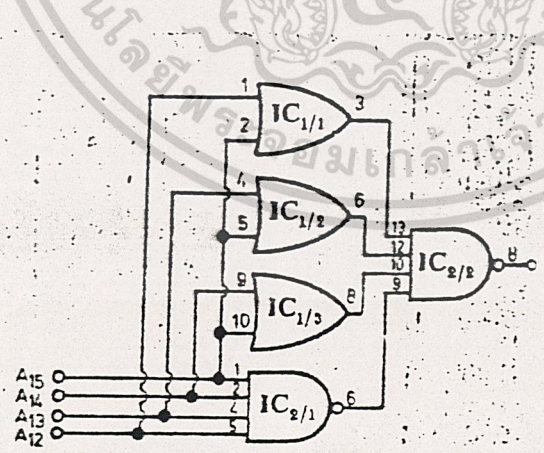
รูปที่ 4.4.5 แสดงดิโค็ดแอดเดรสใช้ 74LS139

A15	A14	A13	A12	OUTPUT
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

$$
\begin{aligned}
OUTPUT &= \overline{A}_{15}\overline{A}_{14} + \overline{A}_{15}\overline{A}_{13} + \overline{A}_{15}\overline{A}_{12} \\
&\quad + A_{15}A_{14}A_{13}A_{12} \\
&= \overline{\overline{\overline{A}_{15}\overline{A}_{14} + \overline{A}_{15}\overline{A}_{13} + \overline{A}_{15}\overline{A}_{12}}} \\
&\quad \overline{\overline{+ A_{15}A_{14}A_{13}A_{12}}} \\
&= \overline{(A_{15}+A_{14}) \cdot (A_{15}+A_{13})} \\
&\quad \overline{.(A_{15}+A_{12}).\overline{(A_{15}A_{14}A_{13}A_{12})}}
\end{aligned}
$$

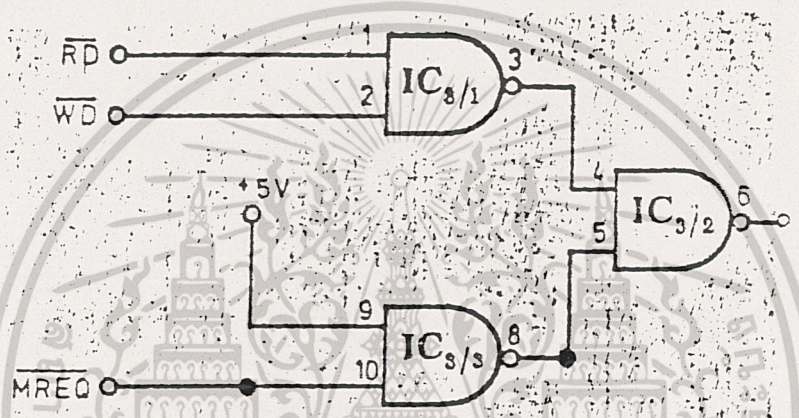


รูปที่ 4.4.6 แสดงภาพการดีโค๊ดแอดเดรสพื้นที่ๆ ต้องการวงจรที่ได้จะเป็นดังรูปที่ 4.4.7



รูปที่ 4.4.7 แสดงวงจรดีโค๊ดที่ได้ (7000H-EFFFH)

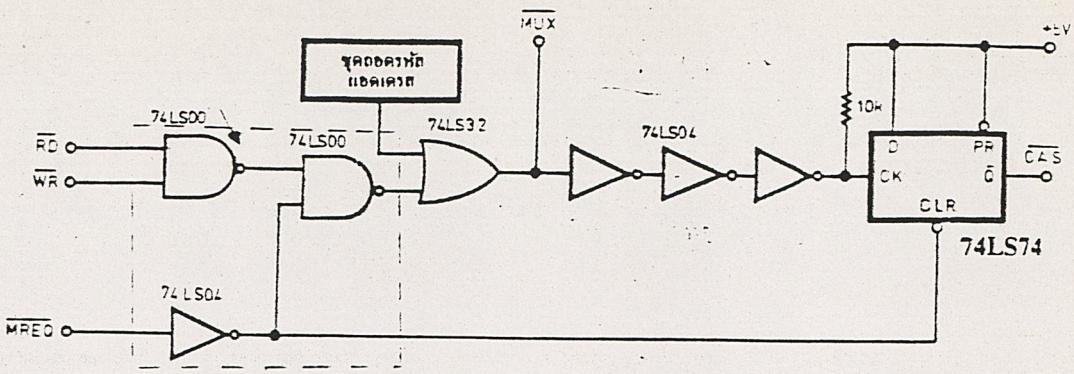
R/W เป็นวงจรที่รวมสัญญาณ RD,WR,MREQ เพื่อใช้ในการกำหนดช่วงเวลาการสร้าง CAS ร่วมกับสัญญาณจากส่วน DECODE เนื่องจากว่าเราต้องการให้สัญญาณ MUX,CAS แอดติฟหลังสัญญาณ WE (เพื่อให้เป็นหมวดการเขียนแบบ Early write cycle) ซึ่งวงจรเป็นแบบง่ายๆ ดังรูปที่ 4.4.8



รูปที่ 4.4.8 แสดงวงจรง่ายๆ ของ R/W

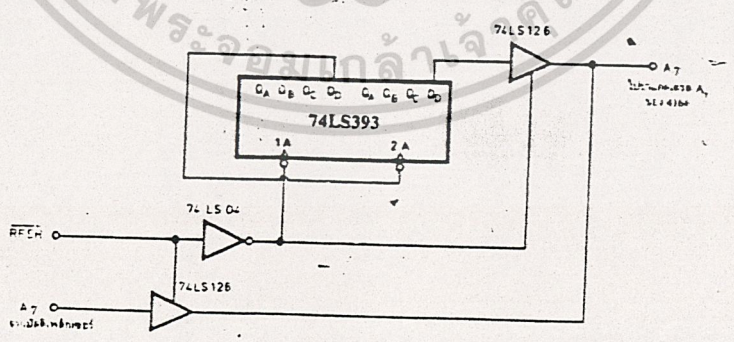
RAS GEN เป็นส่วนที่ผลิตสัญญาณ MUX,CAS ให้ส่วน มัลติเพล็กและ DRAM โดยสัญญาณของการมัลติเพล็กคือ MUX สามารถนำมาจากการ ORGATE ที่อินพุทได้เลยเพราะสัญญาณได้รับการดีเลย์ให้เกิดหลังสัญญาณ WR อยู่แล้ว ส่วนสัญญาณ RAS เราต้องดีเลย์ออกไปอีกให้พ้นเงื่อนไขที่เราดูจาก DATA 4164-12 คือช่วงของ RAS และ CAS ต่างห่างกัน 45 nS (t_{rscl}) โดยวงจรแสดงดังรูปที่ 4.4.9 สัญญาณ MUX จะถูกดีเลย์ไปด้วย 74LS04 ซึ่งจากแผ่นข้อมูลของ 74LS04 มีค่าเวลาดีเลย์ประมาณ 15 nS และตัว 74LS74 ซึ่งเป็น ฟลิบ-ฟลอบ นั้นมีค่าดีเลย์ 40 nS เหตุที่ใช้ฟลิบ-ฟลอบ เป็นตัวสุดท้ายในการให้สัญญาณ CAS ออกมานั้นเพราะ จะได้ขอบสัญญาณที่เหมือนสัญญาณ MUX ที่ดีเลย์มามากที่สุด ฉะนั้นรวมเวลาที่ดีเลย์จาก MUX มาก็เป็น 70 nS ซึ่งจะเห็นว่ามากกว่า t_{rscl}

สังเกตุได้ว่าสัญญาณของ MUX,CAS จะไม่สามารถเกิดขึ้นได้เลยหากไม่ได้รับการอ้าง แอดเดรสที่เป็นพื้นที่เรากำหนดไว้เพราะไม่มีสัญญาณเข้าสู่ส่วนของการสร้าง CAS



รูปที่ 4.4.9 แสดงวงจรส่วนของการสร้าง CAS

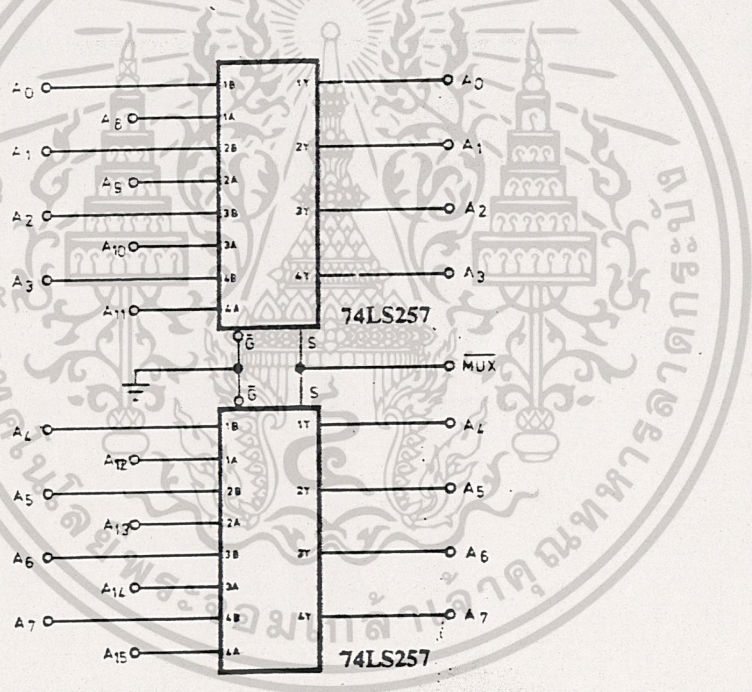
GENERATE BIT FOR REFRESH ใช้สร้างบิทแอดเดรสของการรีเฟรชเพิ่มขึ้นมา ถ้าหากนักศึกษาใช้ DRAM ที่มีขนาดของ ROW ADDR จำนวน 7 บิท ก็ไม่จำเป็นต้องมีส่วนนี้เพราะ CPU ได้มีการรีเฟรช แอดเดรสจำนวน 7 บิทอยู่แล้ว แต่หากเราต้องใช้ DRAM ที่ความจุมากๆ เช่นในกรณีนี้คือ 64 KB 4164 ซึ่งจะมี ROW ADDRESS ขนาด 8 บิทเราจึงต้องสร้างบิทแอดเดรสเพื่อการรีเฟรชเพิ่ม 1 บิท โดยสามารถสร้างได้ง่ายมากคือนำสัญญาณของ RFSH มาทำการนับด้วย TTL ขนาด 8 บิท เช่นเบอร์ตัวอย่างคือ 74LS393 โดยจะทำการนับ การรีเฟรชของ CPU หากครบ 128 ครั้ง ในครั้งต่อมาก็จะสร้างแอดเดรสบิทที่ 7 ขึ้นมาให้



รูปที่ 4.4.10 แสดงวงจรที่ใช้สร้างแอดเดรสรีเฟรชเพิ่ม 1 บิท

จากวงจรในรูปที่ 4.4.10 นั้น ในช่วงที่ไม่ใช่ของการรีเฟรช A7 ก็จะเป็นของแอดเดรสหน่วยความจำธรรมดา(ในกรณีที่ใช้หน่วยความจำที่มีความจุมากขึ้นไปอีกซึ่งต้องการบิทของการรีเฟรชเพิ่มขึ้นนักศึกษาก็สามารถทำได้โดยการเพิ่ม วงจรนับเข้าไป)

MUX เป็นวงจรส่วนของการมัลติเพล็กสัญญาณแอดเดรสหน่วยความจำจาก CPU ที่เข้าสู่ DRAM สามารถใช้วงจรง่ายๆ ดังที่ได้กล่าวมาแล้วข้างต้นโดยใช้ TTL ลำเร็จรูป เช่นเบอร์ 74LS257



รูปที่ 4.4.11 แสดงถึงวงจรส่วนมัลติเพล็กที่ใช้ TTL

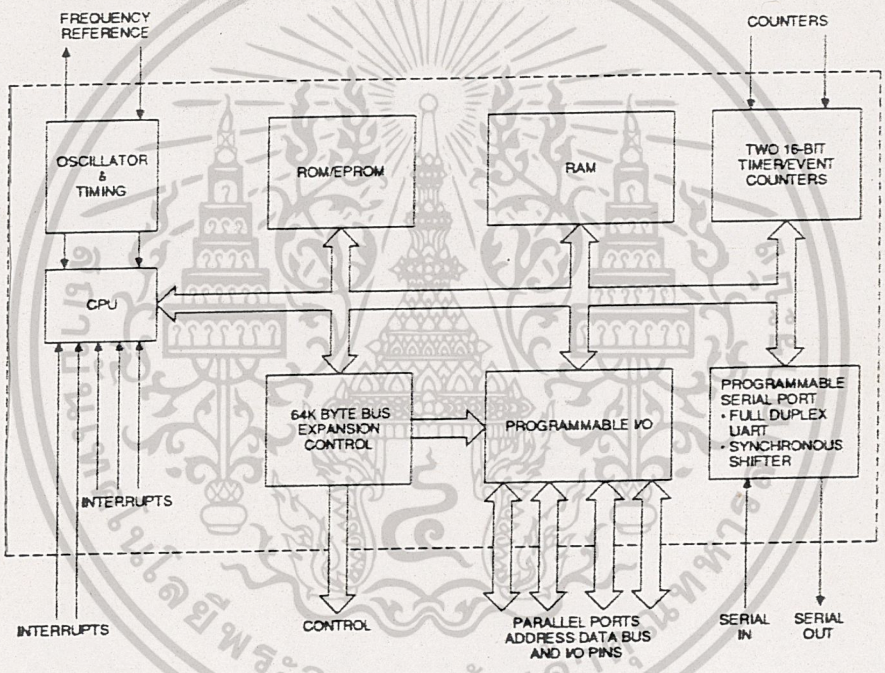
ให้สังเกตุที่ A15 ซึ่งเดิมเราจะต่อกับ A15 ของ CPU แต่ว่าเนื่องจากโจทย์เราต้องการใช้หน่วยความจำเพียง 32 KB จำนวน 2 เพทที่แอดเดรสเดียวกันคือ 7000H-EFFFH ฉะนั้นบิทที่เราจะต้องนำเอามาจากการเอ้าพุทพอร์ท I/O ซึ่งเมื่อเอ้าพุทค่า "0" มาที่บิทนี้ก็ทำการอ่านเขียนข้อมูลตำแหน่ง 7000H-EFFFH ได้โดยจะไม่มีการรบกวนข้อมูลกันระหว่างเพจ

BUFFER ปกติเราไม่ต้องต่อก็ได้เพราะสามารถที่จะเอาขา DATA OUT,DATA IN ของ DRAM ต่อกันได้เลย เพราะเราจัดให้แรมทำงานใน โหมดของ Early write cycle แต่ที่เอ้าพุทที่เชื่อมต่อกับระบบบัสก็ควรจะต่อบัฟเฟอร์ไว้เพราะนอกจากจะแก้ปัญหาของ LOAD EFFECT แล้วยังลดการรบกวนในสายด้วย

บทที่ 5

8052 EMBEDDED PROCESSORS

8052 เป็นไมโครคอลโทรลเลอร์ในตระกูล 8051 เป็นไมโครคอลโทรลเลอร์แบบ SIGLE SHIP มีลักษณะที่เด่นหลายอย่าง ซึ่งจะได้กล่าวต่อไปรูปที่ 5.1 แสดงโครงสร้างภายในของ 8052



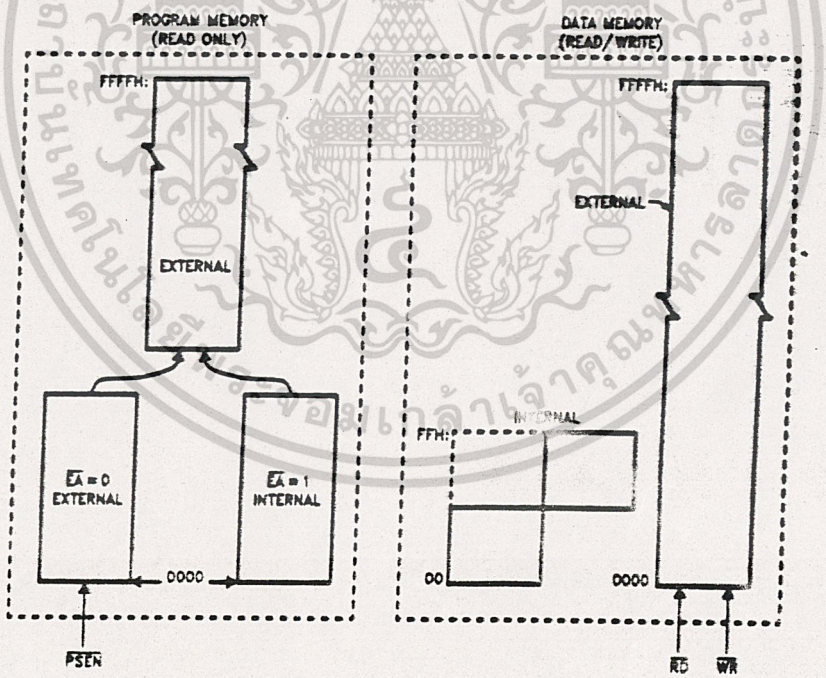
รูปที่ 5.1 Architectural strocturc of 8052

8052 เป็น ไอซี แบบ CMOS คุณสมบัติดังนี้

- 8-bit CPU สะดวกสำหรับการประยุกต์ใช้งาน
- มี ROM 8k-bytes ภายในตัว
- มี RAM 256 bytes ภายใน
- 16-bit TIMER/COUNTERS
- BOOLEAN PROCESSOR

- มี OSCILLATOR ภายใน
- 64 k PROGRAM MEMORY
- 64 k DATA MEMORY
- SOFWARE RESET
- รับ INTERRUPT ได้ 5 PIN

8052 สามารถต่อ หน่วยความจำภายนอกได้ถึง 64 k ซึ่งทำให้สะดวกในการนำไปประยุกต์ใช้งาน การต่อหน่วยความจำภายนอก สามารถต่อได้ 2 กรณีคือ PROGRAM MEMORY กับ DATA MEMORY ดังแสดงรูป 2 PROGRAM MEMORYหมายถึงการติดต่อกับหน่วยความจำประเภท PROM (PROGRAMMABLE READ ONLY MEMORY) คือสามารถอ่านข้อมูลได้อย่างเดียวในขณะที่ DATA MEMORY หมายถึงการติดต่อกับหน่วยความจำประเภท RAM (RANDOM ACCESS MEMORY) คือสามารถเขียนอ่านข้อมูลได้แต่ข้อมูลจะสูญหายไปเมื่อ หยุดจ่ายไฟให้

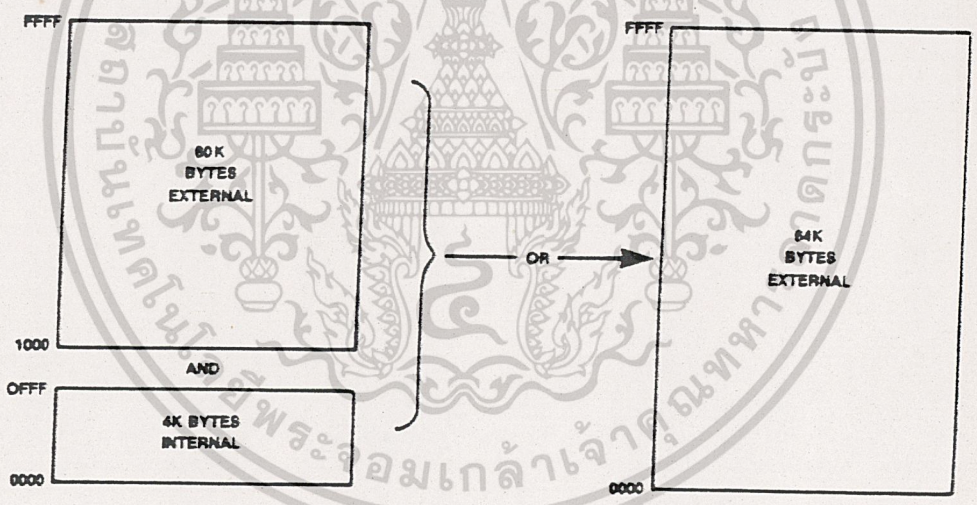


รูปที่ 5.2 โครงสร้างการต่อ 8052 กับหน่วยความจำ

PROGRAM MEMORY

รูปที่ 5.3 แสดง ตำแหน่งหน่วยความจำ ทุกครั้งที่เกิดการรีเซ็ท CPU จะเริ่มทำงานที่ตำแหน่ง 0000H จากรูปที่ 3 เมื่อเกิดการอินเตอร์รัพท์ CPU จะกระโดดไปทำงานที่แอดเดรส ที่ถูกกำหนดไว้แล้ว เช่น เมื่อเกิดอินเตอร์รัพท์ 0 CPU จะกระโดดไปที่แอดเดรส 0003H

ผู้ใช้สามารถเลือกใช้หรือไม่ใช้ ROM 8 k ภายใน 8052 ได้โดยการต่อขา EA (EXTERNAL ACCESS) เข้ากับ Vcc หรือ Vss ถ้าต่อ EA = Vcc CPU จะติดต่อกับ ROM ภายในเป็นแอดเดรส 0000 H ถึง 1FFFH และติดต่อกับหน่วยความจำภายนอกเป็นแอดเดรส 2000 H ถึง FFFFH

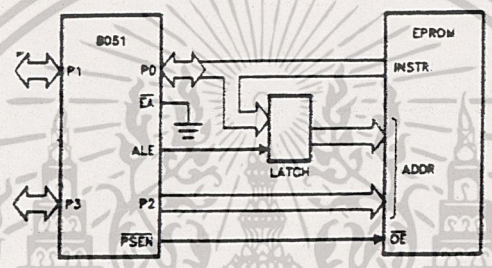


รูปที่ 5.3 แสดงโครงสร้าง PROGRAM MEMORY

แต่ถ้าต่อ EA = Vss CPU จะติดต่อกับหน่วยความจำภายนอกตั้งแต่แอดเดรส 0000H ถึง FFFFH โดย ROM ภายในจะไม่ถูกใช้งาน

นอกจากนี้แล้วขา PSEN (PROGRAM STOBE ENABLE) ของ CPU จะต้องต่อเข้ากับขา OE (OOTPUT ENABLE) ของหน่วยความจำ

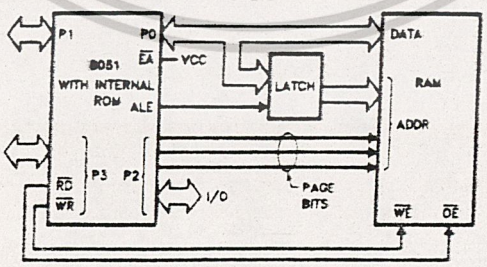
การต่อ CPU กับหน่วยความจำภายนอก แสดงไว้ในรูป 4 โดยที่พอร์ท 0 ของ CPU จะเป็นทั้ง แอดเดรสบัส และ ดาต้าบัส ในกรณีที่ พอร์ท 0 เป็น แอดเดรสบัส ขา ALE (ADDRESS LATCH ENABLE) จะเป็นโลจิก "0" เป็นการบอกว่าขณะนี้พอร์ท 0 ทำงานเป็นแอดเดรสบัสหลังจากนี้ขา ALE จะกลับเป็นโลจิก "1" แสดงว่าขณะนี้พอร์ท 0 เป็นดาต้าเบส และที่พอร์ท 2 จะเป็น แอดเดรส 8 บิทบน



รูปที่ 5.4 การต่อ CPU แบบ PROGRAM MEMORY

DATA MEMORY

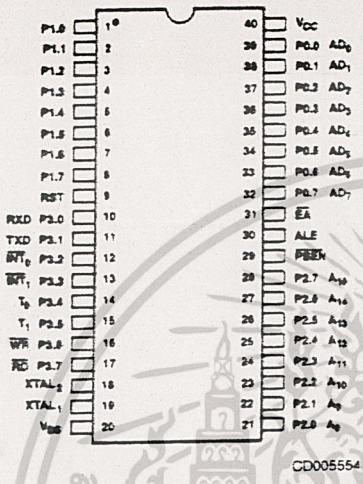
รูปที่ 5.5 เป็นการต่อ CPU เข้ากับ RAM คือเป็น DATA MEMORY CPU จะสร้างสัญญาณ RD และ WR ซึ่งจำเป็นในการอ่านข้อมูลจากหน่วยความจำ RAM



รูปที่ 5.5 การต่อ CPU แบบ DATA MEMORY

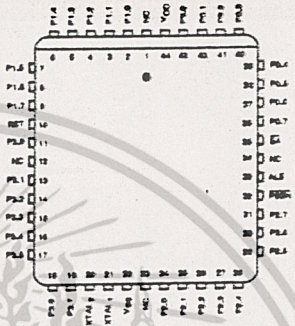
CONNECTION DIAGRAMS
Top View

DIP
80C51BH/80C31BH
80C52T2/80C32T2



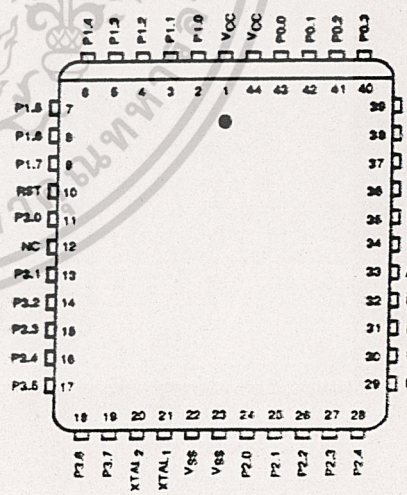
CD005554

PLCC
80C51BH/80C31BH



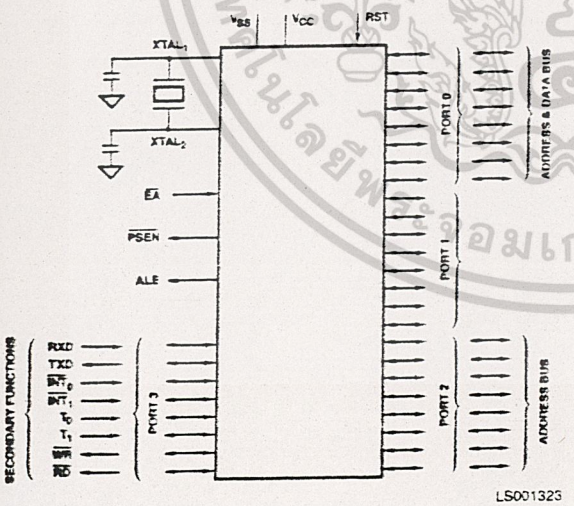
CD009443

PLCC
80C52T2/80C32T2



CD009444

LOGIC SYMBOL



LS001323

Note: Pin 1 is marked for orientation.

รูปที่ 5.6 แสดง CONNECTION DIAGRAM ของ 8052.

รายละเอียดทั่วไป

CPU 8052 จะมีตัวถัง 2 ลักษณะ คือแบบ DIP ซึ่งจะมี 40 ขา และแบบ PLCC ซึ่งมี 44 ขา ดังรูปที่ 5.6

รายละเอียดของขา

PORTO

PORTO เป็นแบบ OPEN-DRAIN เป็น PORT 2 ทิศทาง ทำหน้าที่ทั้งแอดเดรสบิส และดาต้าบิส เมื่อเป็นแอดเดรสบิสจะเป็นแอดเดรส 8 บิทล่าง

PORT 1

PORT 1 เป็น PORT ขนาด 8 บิท 2 ทิศทางมีรีซิสเตอร์ PULL UP อยู่ภายในแล้ว สามารถต่อกับเกทชนิด TTL ได้ 4 ตัว ปกติจะเป็น HIGH เนื่องจาก PULL UP เป็นแอดเดรสบิส 8 บิทล่าง ขณะทำการตรวจสอบข้อมูล (PROGRAM VERIFICATION)

PORT 2

PORT 2 เป็น PORT ขนาด 8 บิท 2 ทิศทาง มีรีซิสเตอร์ PULL UP อยู่ภายใน - เช่นเดียวกับ PORT 1 PORT 2 จะทำหน้าที่เป็นแอดเดรสบิส 8 บิทบน ขณะที่ติดต่อกับหน่วยความจำภายนอก

PORT 3

PORT 3 เป็น PORT ขนาด 8 บิท 2 ทิศทาง มีรีซิสเตอร์ PULL UP อยู่ภายใน เช่นเดียวกับ PORT 1 สามารถต่อ เกทชนิด TTL ได้ 4 ตัว PORT 3 นอกจากจะทำหน้าที่เป็นอินพุท เอ้าพุท พอร์ทแล้ว ยังมีหน้าที่พิเศษ ดังนี้

PORT	ALTERWATE FUNCTION
P 3.0	RxD (SERIAL INPUT PORT)
P 3.1	TxD (SERIAL OUTPUT PORT)
P 3.2	INT0 (EXTERNAL INTERRUPT 0)
P 3.3	INT1 (EXTERNAL INTERRUPT 1)
P 3.4	T0 (TIMER 0 EXTERNAL INPUT)
P 3.5	T1 (TIMER 1 EXTERNAL INPUT)
P 3.6	WR (EXTERNAL DATA MEMORY WRITE STROBE)
P 3.7	RD (EXTERNAL DATA MEMORY READ STROBE)

RST (INPUT,ACTIVE HIGH)

ถ้าขานี้เป็น HIGH (ประมาณ 2 MACHINE CYCLES) CPU จะถูกรีเซ็ท

ALE ADDRESS LATCH ENABLE (OUTPUT,ACTIVE HIGH)

ALE เป็นขา OUTPUT จะให้กำเนิดพัลซ์ สำหรับ LATCH แอดเดรสบีส 8บิท ล่างขณะที่ติดต่อกับ หน่วยความจำภายนอก

PSEN PROGRAM STROBE ENABLE (OUTPUT,ACTIVE LOW)

PSEN เป็นขาซึ่งทำหน้าที่เป็น READ STROBE ให้กับหน่วยความจำภายนอกชนิด PROM และจะไม่ทำงานในกรณีที่ CPU เฟทช์คำสั่งจาก หน่วยความจำภายในตัว CPU เอง

EA EXTERNAL ACESS ENABLE (INPUT,ACTIVE LOW)

ถ้าต่อ EA = V_{SS} จะทำให้ CPU ทำการเฟทช์คำสั่งจาก PROM ภายนอกตั้งแต่แอดเดรส 0000H ถึง OFFFH

ถ้าต่อ EA = V_{CC} จะทำให้ CPU เฟทช์คำสั่งจาก PROM ภายใน ตั้งแต่แอดเดรส 0000H ถึง OFFFH

X TAL_1 CRYSTAL (INPUT)

เป็นขาอินพุทต่อเข้ากับ ออสซิลเลเตอร์ภายใน

X TAL_2 CRYSTAL (OUTPUT)

เป็นขาเอ้าพุทจากการขยายสัญญาณภายใน CPU

V_{CC} POWER SUPPLY

ต่อกับแหล่งจ่ายไป

V_{SS} เป็น GROUND ของวงจร

การทำงานเมื่อเกิดการรีเซ็ท

การรีเซ็ท 8052 จะเกิดขึ้นได้ 4 กรณีคือ

1. POWER-ON RESET

   ที่ขา RESET ของ CPU จะถูกต่อด้วย คาปาซิเตอร์กับ V_{cc} ดังนั้น ทุกครั้งที่เราจ่ายไปให้กับ CPU จะถูกรีเซ็ท

2. HARDWARE RESET

   เกิดขณะที่ CPU กำลังทำงานอยู่ แล้วขา RESET ถูกต่อเข้ากับ V_{cc} จะเป็นการรีเซ็ท CPU เช่นกัน

3. WATEHDOG RESET

   เกิดขึ้นเมื่อ WATEHDAS TIMER เพิ่มค่าเกินค่าที่กำหนดไว้ในโปรแกรมจะทำให้ CPU ถูกรีเซ็ท 2 MACHINC CYCLE

4. SOFTWARE RESET

   เกิดขึ้นเมื่อ SOFTWARE เขียนค่าที่ตรงกับค่าใน WATCH DOS TIMER

SPECIAL FUNCTION REGISTER MAP

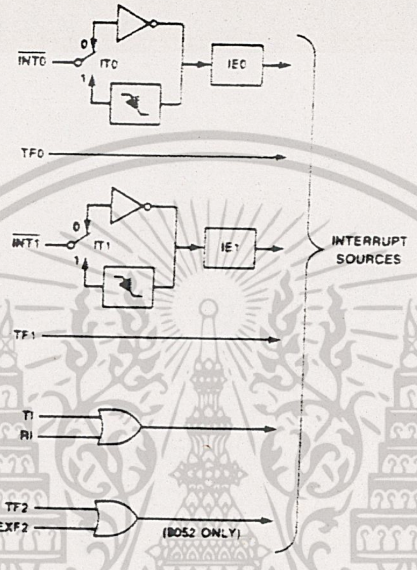
Addr (HEX)	Symbol	Name	Default After Power-On Reset
* 80	P0	Port 0	11111111
81	SP	Stack Pointer	00000111
82	DPL	Data Pointer Low	00000000
83	DPH	Data Pointer High	00000000
+ 84	DPL1	Data Pointer Low 1	00000000
+ 85	DPH1	Data Pointer High 1	00000000
+ 86	DPS	Data Pointer Selection	00000000
87	PCON	Power Control	0XXX0000
* 88	TCON	Timer/Counter Control	00000000
89	TMOD	Timer/Counter Mode Control	00000000
8A	TL0	Timer/Counter 0 Low Byte	00000000
8B	TL1	Timer/Counter 1 Low Byte	00000000
8C	TH0	Timer/Counter 0 High Byte	00000000
8D	TH1	Timer/Counter 1 High Byte	00000000
* 90	P1	Port 1	11111111
* 98	SCON	Serial Control	00000000
99	SBUF	Serial Data Buffer	Indeterminate
* A0	P2	Port 2	11111111
* A8	IE	Interrupt Enable Control	0XX00000
+ A9	WDS	Watchdog Selection	00000000
+ AA	WDK	Watchdog Key	00000000
* B0	P3	Port 3	11111111
* B8	IP	Interrupt Priority Control	XXX00000
* D0	PSW	Program Status Word	00000000
* E0	ACC	Accumulator	00000000
* F0	B	B Register	00000000

* Bit Addressable
+ New SFRs defined on the 80C521/80C321

ตารางที่ 1 เป็นตารางแสดงค่าของพอร์ทและรีจิสเตอร์เมื่อเกิดการรีเซ็ทขึ้น

อินเตอร์รัพท์

8052 สามารถรับรู้การอินเตอร์รัพท์ได้ถึง 5 ตำแหน่ง ดังรูป 5.7



รูปที่ 5.7 แสดงตำแหน่งการอินเตอร์รัพท์

อินเตอร์รัพท์ INT_0 และ INT_1 ของ 8052 สามารถถูกอินเตอร์รัพท์ได้ทั้งแบบ LEVEL หรือแบบขอบขา (TRANSITION) ซึ่งขึ้นอยู่กับการกำหนดค่าในบิท IT_0 และ IT_1 ของรีจิสเตอร์ TCON

รูปที่ 5.8 แสดงอินเตอร์รัพท์รีจิสเตอร์ เป็นรีจิสเตอร์ที่ใช้ควบคุมว่า CPU จะยอมรับหรือไม่ยอมรับการอินเตอร์รัพท์ ซึ่งสามารถสั่งได้ด้วยโปรแกรม เช่น EA=0 CPU จะไม่ยอมรับการอินเตอร์รัพท์ใด ๆ เลยแต่ถ้า EA=1 CPU ก็จะยอมรับการอินเตอร์รัพท์โดยขึ้นกับบิทอื่นใน IE (INTERRUPT ENABLE REGISTOR) ซึ่งจะมีอีก 6 บิท ดังรูปที่ 8 ยกตัวอย่าง เช่น ถ้า EX0=0 CPU ก็จะไม่ตอบสนองการอินเตอร์รัพท์จากขา IE.0 ในทางตรงกันข้าม หาก EX0 ถูกเซ็ทเป็น "1" CPU จะรับรู้การอินเตอร์รัพท์จากขา IE.0 และบิทอื่นของรีจิสเตอร์ IE ก็จะทำงานในลักษณะเดียวกับบิท EX0

(MSB)							(LSB)
EA	X	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved
ET2	IE.5	enables or disables the Timer 2 overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Timer 1 interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow Interrupt. If ET0 = 0, the Timer 0 Interrupt is disabled.
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

รูปที่ 5.8 แสดง INTERRUPT ENABLE REGISTER

เมื่อ CPU ได้รับสัญญาณอินเตอร์รัพท์จากภายนอก CPU จะสร้างคำสั่ง CALL เพื่อกระโดดไปทำงาน ตามตำแหน่ง ของแต่ละอินเตอร์รัพท์เวคเตอร์ ดังนี้

SOURCC	VECTOR ADDRESS
IE0	0003H
TF0	0003H
IE1	0013H
TF1	0013BH
RI+TI	0023H
TF2+EXF2	002BH

ยกตัวอย่างเช่น เมื่อมีการอินเตอร์รัพท์จากขา IE0 CPU จะสร้างคำสั่ง LCALL ไปยังตำแหน่งแอดเดรส 0003H และเก็บค่า โปรแกรมเคาเตอร์ไว้ใน STACK และจะทำงานตามคำสั่งใน โปรแกรมอินเตอร์รัพท์ จนกระทั่งพบคำสั่ง REIT CPU จะคืนค่าโปรแกรมเคาเตอร์จาก STACK และทำงานตามเมนโปรแกรมต่อไปการอินเตอร์รัพท์จะไม่ถูกทำหากเกิดสภาวะต่อไปนี้

1. CPU ทำโปรแกรมอินเตอร์รัพท์ ที่มีความสำคัญเท่าหรือสูงกว่า
2. การอินเตอร์รัพท์ต้องเกิดขึ้น MACHINE CYCLE สุดท้ายของคำสั่ง
3. CPU กำลังทำคำสั่ง RET1 หรือเกี่ยวข้องกับรีจิสเตอร์ IE หรือ IP

การอินเตอร์รัพท์จากภายนอกอาจเป็นแบบ LEVEL หรือแบบขอบขาลงขึ้นอยู่กับเซทบิทรีจิสเตอร์ TCON ถ้า IT_x=Ø การอินเตอร์รัพท์จะต้องเป็นศูนย์ แต่ถ้าหาก IT_x1=1 การอินเตอร์รัพท์จะเป็นขอบขาลงและจะทำให้ IE_x ใน TCON รีจิสเตอร์ถูกเซท

เมื่อมีการอินเตอร์รัพท์เข้ามาในแต่ละครั้งควรจะต้องเป็นโลวิกศูนย์อย่างน้อย 12 CLOCK เพื่อให้มั่นใจ และถ้าเป็นแบบขอบขาจะต้องเป็น HIGH อย่างน้อย 1 CLOCK และเปลี่ยนไปเป็น LOW อีกอย่างน้อย 1 CLOCK

ลำดับความสำคัญการอินเตอร์

ขาอินเตอร์รัพท์แต่ละขาจะมีความสำคัญมากน้อยไม่เท่ากัน เป็นอิสระต่อกันและสามารถกำหนดความสำคัญด้วยการเซ็ทค่าในรีจิสเตอร์ IP (INTERRUPT PRIORITY REGISTER)

ถ้าเกิดการร้องขอการอินเตอร์รัพท์ด้วยความสำคัญต่างกันพร้อมกัน CPU จะทำตามการร้องขอของขาอินเตอร์รัพท์ที่มีความสำคัญสูงกว่า แต่ถ้าหากความสำคัญเท่ากันและเกิดการร้องขอพร้อมกัน CPU จะทำตามการร้องขอการอินเตอร์รัพท์ตามลำดับดังนี้

	SOURCE	PRIORITY WITHIN LEVEL
1.	IEØ	(HIGHEST)
2.	TFØ	
3.	IE1	
4.	TF1	
5.	RI+TI	
6.	TF2+EXF2	

รูปที่ 5.9 แสดง INTERRUPT PRIORITY REGISTER เป็นรีจิสเตอร์ที่ใช้สำหรับกำหนดความสำคัญให้กับการอินเตอร์รัพท์แต่ละขา

(MSB)							(LSB)
X	X	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	defines the Timer 1 interrupt priority level. PT = 1 programs it to the higher priority level.
PX1	IP.2	defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

รูปที่ 5.9 แสดง IP: INTERRUPT PRIORITY REGISTER

INTERNAL TIMING

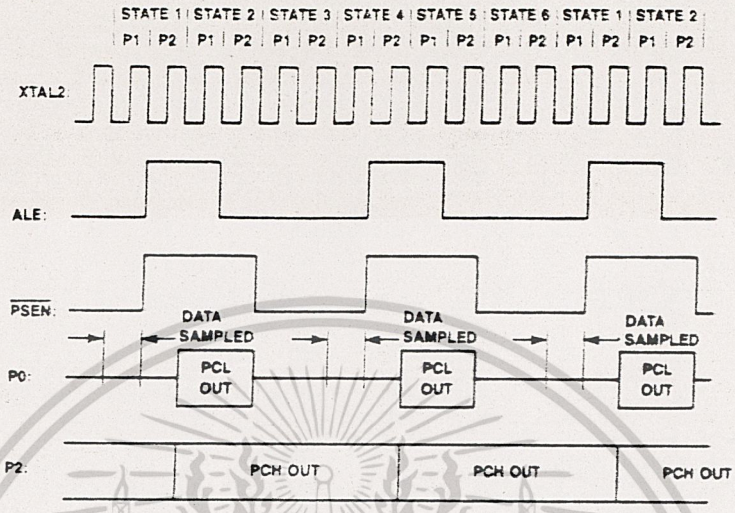
รูปที่ 5.10 ถึงรูปที่ 5.18 แสดงไทม์มิ่ง แสดง การกำหนดสัญญาณภายในจากสัญญาณ CLOCK จากรูปนั้นไม่ได้แสดงขอบขาขึ้นหรือขอบขาลงของสัญญาณ แต่ต้องการแสดงถึงสัดส่วนของสัญญาณเหล่านั้น เมื่อเทียบกับสัญญาณ X TAL 2

ขอบขาขึ้นหรือขอบขาลง จะขึ้นอยู่กับอุปกรณ์ที่มาติดต่อกับ CPU ซึ่งเวลาที่เกิดขึ้นจากการเปลี่ยนระดับจาก 0.8 V ไปเป็น 2 V ประมาณ 10 ns

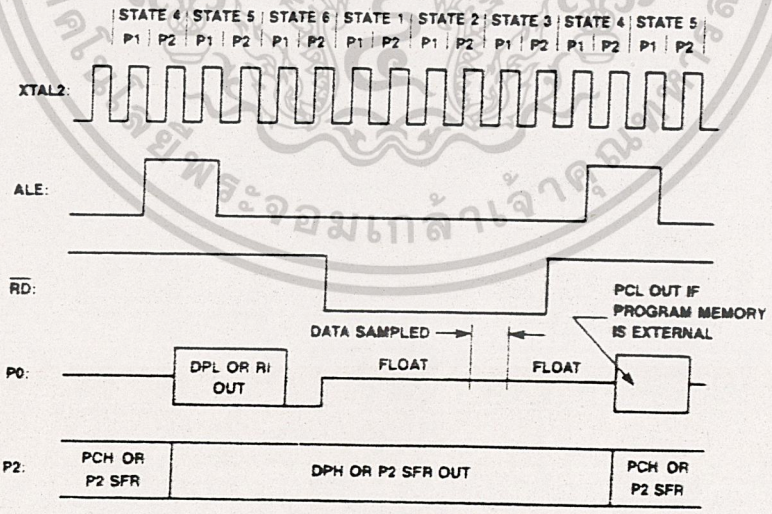
(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

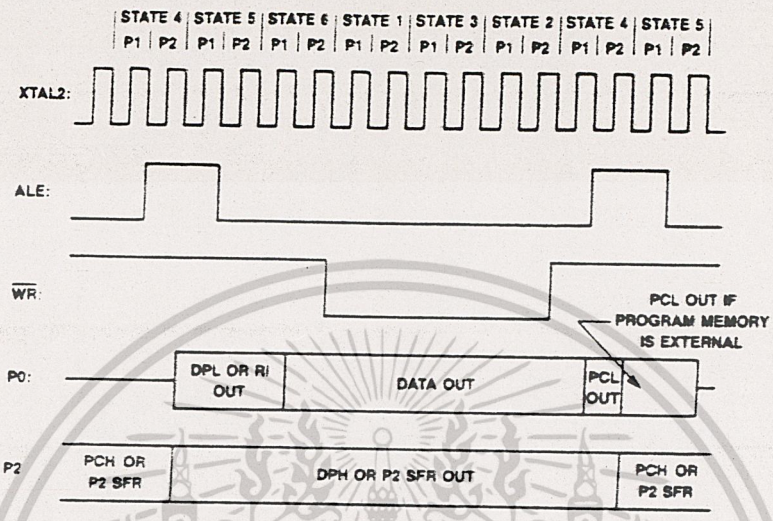
รูปที่ 5.14 TCON : TIMER/COUNTER CONTROL REGISTER



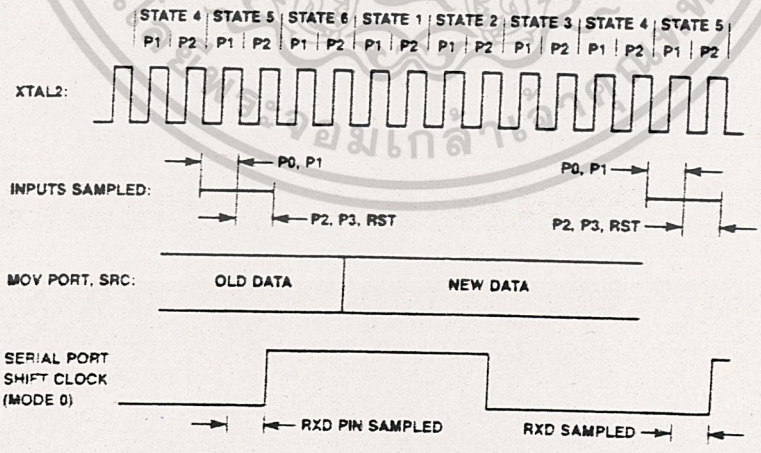
รูปที่ 5.10 EXTERNAL PROGRAM MEMORY FETCHES



รูปที่ 5.11 EXTERNAL DATA MEMORY READ CYCLE



รูปที่ 5.12 EXTERNAL DATA MEMORY WRITE CYCLE



รูปที่ 5.13 PORT OTERATION

TCON REGISTER

รีจิสเตอร์ TCON (TIME/COUNTER CONTROL REGISTER) เป็นรีจิสเตอร์-ใช้สำหรับกำหนดการชนิดของการอินเตอร์รัพท์ ว่าจะเลือกชนิด LEVEL หรือชนิดขอบขา อีกทั้งยังมีแฟลกเพื่อสภาวะของการอินเตอร์รัพท์ รูปที่ 5.14 แสดงรีจิสเตอร์ TCON

SPECIAL FUNCTION REGISTER

รูปที่ 5.15 แสดง SFR (SPECIAL FUNCTION REGISTER) เป็นรีจิสเตอร์พิเศษที่ใช้เฉพาะงาน ยกตัวอย่าง เช่น แอดดิวมูเลเตอร์ $PORT_0$,$PORT_1$ ดังนั้นในการใช้งาน REGISTER ในรูปที่ 15 จึงต้องอ้างไปที่ แอดเดรสนั้น

Table 3-1

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*-T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
-TH2	Timer/Counter 2 High Byte	0CDH
-TL2	Timer/Counter 2 Low Byte	0CCH
-RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
-RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
- = 8052 only

รูปที่ 5.15 SPECIAL FUNCTION REGISTER

รูปที่ 5.16 แสดง MEMORY MAP ของ SPECIAL FUNCTION REGISTER

SFR Memory Map

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

↑ Bit Addressable

Figure 3-5. Memory Map

รูปที่ 5.16 MEMORY MAP

PSW : PROGRAM STATUS WORD.

PSW : (PROGRAM STATUS WORD) เป็นรีจิสเตอร์ที่เก็บสภาวะที่เกิดขึ้นจากแอดคิวมูเลเตอร์ ดังแสดงรูปที่ 5.17

CPU TIMING

แต่ละ MACHINE CYCLE จะประกอบขึ้นด้วย 6 สเตท (ใช้ CLOCK 12 ลูก)และแต่ละสเตทจะมี 2 เฟส โดย 1 สัญญาณนาฬิกาเป็น 1 เฟส การกระทำทางคณิตศาสตร์หรือโลจิก จะกระทำต่อเฟสแรก ส่วนเฟสสองจะเป็นการส่งข้อมูลระหว่างรีจิสเตอร์

รูปที่ 5.18 แสดงการเฟสและเอ็กซิคิว เปรียบเทียบกับสัญญาณนาฬิกา สัญญาณ ALE (ADDRESS LATCH ENABLE) จะเกิดขึ้นระหว่าง S_1P_2 และระหว่าง S_2P_2 และ S_3P_1

PSW: Program Status Word. Bit Addressable.

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
—	PSW.1	Not implemented, reserved for future use.*
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

*User software should not write 1s to reserved bits. These bits may be used in future 8051 Family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

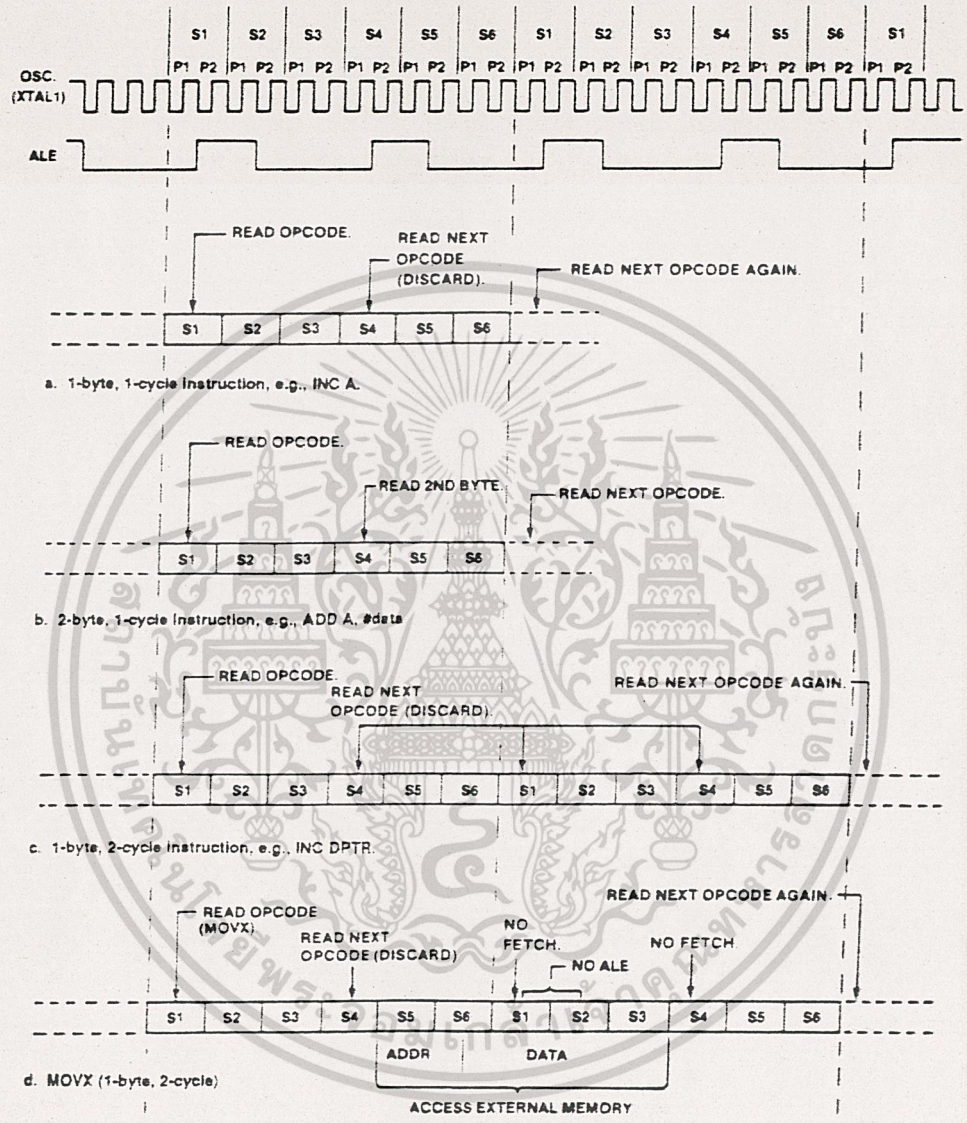
รูปที่ 5.17 PSW : PROGRAM STATUS WORD REGISTER

การเอ็กซีคิว คำสั่ง 1-CYCLE จะเริ่ม S1P2 อ๊อฟโค้ดจะถูกแลทช์เข้าไว้ในรี-จิสเตอร์ถ้าเป็นคำสั่ง 2 ไบท์ คำสั่งไบท์ที่ 2 จะถูกอ่านระหว่าง S4 แต่ถ้าเป็นคำสั่ง 1 ไบท์ CPU จะเฟทช์คำสั่งที่ S4 แต่จะไม่มีการอ่านไบท์ที่ 2 และจะไม่เพิ่มค่าโปรแกรมเคาร์เตอร์

การใช้งานและวงจร 8052

ขา EA (EXTERNAL ACESS) จะถูกต่อลงกราวด์ ดังนั้น CPU จะเฟทช์อ๊อฟโค้ดจาก PROM โดยเริ่มตั้งแต่แอดเดรส 0000H ไปจนถึงแอดเดรสสุดท้ายตามขนาดของ PROM

ขา PSEN (PROGRAM STOBE ENABLE) จะถูกต่อเข้ากับ ขา OE (OUTPUT ENABLE) ของ PROM เพื่อเป็นการอ่านข้อมูลจาก PROM



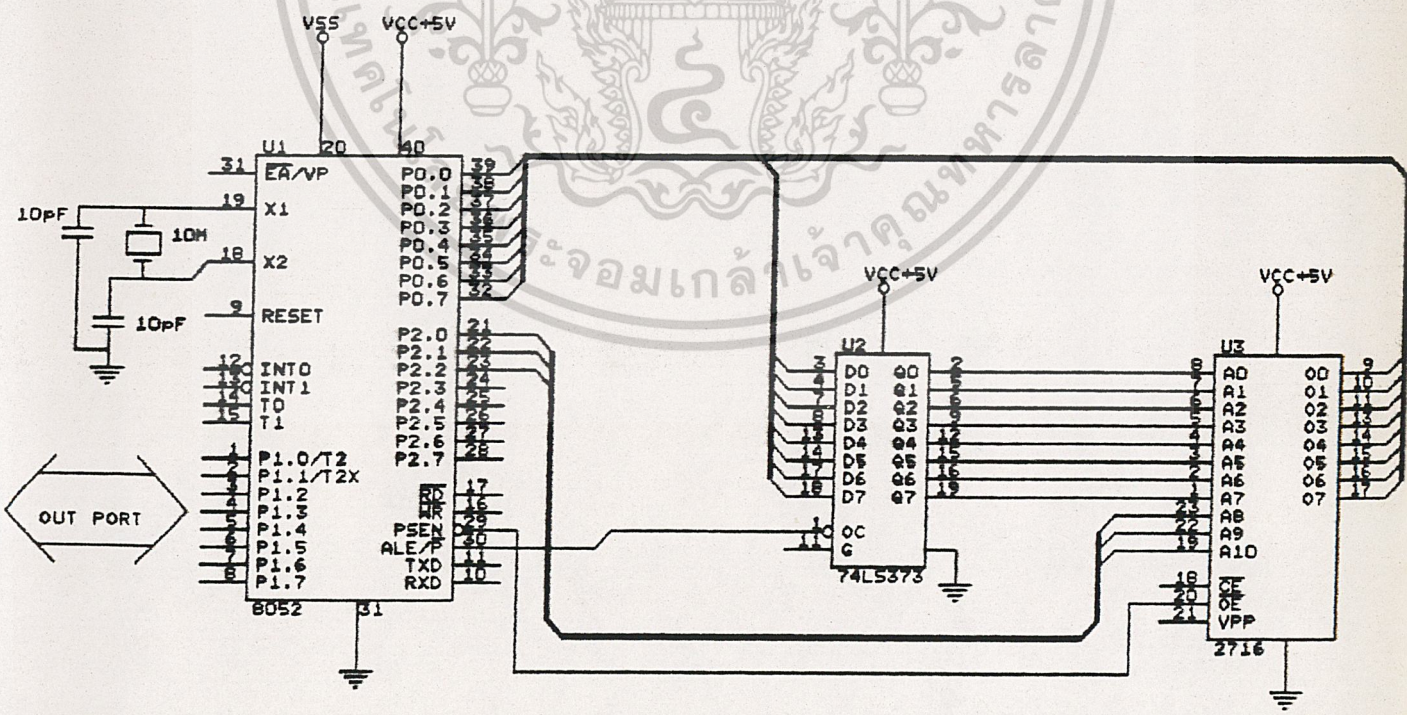
รูปที่ 5.18 8052 FETCH/EXECUTE SEQUENCES

พอร์ท 0 ของ CPU จะถูกใช้เป็นบิสข้อมูลและแอดเดรส 8 บิทล่างโดยทุกครั้งที่พอร์ท 0 ทำหน้าที่ส่งแอดเดรส ขา ALE (ADDRESS LATCH ENABLE) จะแอดทีฟไม่เป็น CLOCK ให้แก่ไอซี 74LS373 ซึ่งเป็นตัวแลทช์แอดแดรสให้กับ PROM และขณะเดียวกันพอร์ท 0 จะส่งข้อมูลให้กับ PROM ด้วยโดยขณะที่ พอร์ท 0 ทำหน้าที่ ส่งข้อมูล ขา ALE จะไม่แอดทีฟ

พอร์ท 2 ของ CPU จะทำหน้าที่เป็นแอดเดรสบิส 8 บิทบนต่อเข้ากับขาแอดเดรสของ PROM

พอร์ท 1 จะเป็น อินพุท,เอ้าพุท พอร์ท เพื่อส่งข้อมูลไปควบคุมการทำงานของชุดบันทึกเสียงพูด และพอร์ท 3 จะใช้เป็นตัวรับการอันเตอร์รัพท์จากสัญญาณภายนอกโดยจะตรวจสอบสัญญาณเรียกเข้าเมื่อมีผู้โทรศัพท์เข้า

CPU 8052 อาศัยการแลทช์ข้อมูลแบบ LEVEL ดังนั้นจึงเลือกใช้ 74LS373 เป็นไอซีแลทช์แบบ LEVEL ที่ขา OE (OUTPUT ENABLE) ของ 74LS373 จะถูกต้องลงกราวด์ได้ เอ้าพุทของ 74LS373 จะต่อเป็นแอดเดรส 8 บิทล่างของ PROM



รูปที่ 5.19 แสดงวงจร 8052

บทสรุป

จากที่กล่าวไว้ใน ปริญญานิพนธ์ ฉบับนี้ แนวทางเบื้องต้นในการบันทึกเสียงพูดโดยอาศัย IC # T 6668 เป็นตัวเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล และสัญญาณดิจิตอลเป็นอนาลอก IC # T 6668 มีข้อดีคือมีความสะดวกที่จะต่อร่วมกับ CPU ซึ่งใช้เบอร์ 8052 IC # T 6668 สามารถเลือกบิทเรทในการบันทึกได้ ซึ่งบิทเรทยิ่งสูงเสียงพูดที่ได้จะชัดเจนมีเสียงแหลมมาก แต่ระยะเวลาในการเก็บข้อมูลจะลดลง บิทเรทมีค่าต่าง ๆ คือ 8 , 11 , 16 , 32 KBit/sec

CPU ที่ใช้คือเบอร์ # 8052 มีความสะดวกมากเพราะภายในตัวมี RAM , ROM , 16 Bit Timmer และ Interruupt 5 Pin สามารถต่อออก Port ได้โดยตรง

ในส่วนของการถอดรหัสสัญญาณโทรศัพท์ชนิดกดปุ่ม ใช้ IC # MT 8870 จะถอดรหัสเป็น BCD 4 Bit จากการทดลอง วงจรถอดรหัสสามารถถอดรหัสได้ตรงตามที่กล่าวมาแล้วในบทที่ 2

การทดลองระยะเวลาการบันทึกของชุดบันทึกเสียง เมื่อใช้บิทเรทต่าง ๆ คือ

บิทเรท K/sec	ระยะเวลา
8	127
11	92
16	63
32	31

การทำงานของชุดตรวจสอบสัญญาณกระดิ่ง เวลาที่หน่วงสัญญาณกระดิ่ง 2 ครั้งใช้เวลา 10 วินาที ระยะเวลาในการอ่าน และบันทึกเสียงพูดใช้เวลา 52 วินาที

เมื่อรวมวงจรต่าง ๆ เข้าด้วยกัน แล้วทำการทดลองก็เป็นไปตามที่ต้องการ มีปัญหาพอสมควรแต่สามารถแก้ไขได้ หน้าที่การใช้งานของคีย์ต่าง ๆ ของโทรศัพท์ทำได้ตรงตามต้องการทุกประการ

กิตติกรรมประกาศ

ในการทำโครงงาน ปริญญานิพนธ์ ชุดบันทึกเสียงพูดและตรวจสอบข้อมูลทางโทรศัพท์ ที่สำเร็จลุล่วงมาด้วยดีต้องขอขอบพระคุณ อ.นิกร สุขุตมตันติ และเพื่อน ๆ ที่คอยให้คำปรึกษา หวังอย่างยิ่งว่า ปริญญานิพนธ์ ฉบับนี้ จะเป็นแนวทางในการค้นคว้าและพัฒนาต่อไป

หนังสืออ้างอิง

- Eight - Bit 80C51 Embedded Processors 1990 Data Book

-คู่มือการใช้งาน IC ของบริษัท MOTOLOLA

-คู่มือการใช้งาน IC ของบริษัท TOSHIBA

-หนังสืออิเล็คทรอนิคส์เวิลด์

-หนังสืออิเล็คทรอนิคส์เซมิคอนดักเตอร์

80C51BH/80C31BH/80C52T2/80C32T2

CMOS Single-Chip Microcontrollers

DISTINCTIVE CHARACTERISTICS

- Industry Standard CMOS Microcontrollers
- Low Power Modes—Idle & Power-Down
- 32 Programmable I/O Lines
- Two 16-bit Counter/Timers
- Programmable Serial Channel
  - Five-source, two-level Interrupt Structure
  - Boolean Processor
- 64K bytes Program Memory Space
- 64K bytes Data Memory Space

	RAM (bytes)	ROM (bytes)
80C31BH	128	—
80C51BH	128	4K
80C32T2	256	—
80C52T2	256	8K

80C51BH = 80C31BH + 4K bytes ROM
80C52T2 = 80C32T2 + 8K bytes ROM

GENERAL DESCRIPTION

The 80C51BH and 80C31BH are CMOS versions of the industry-standard 8051 architecture. The 80C52T2 and 80C32T2 are identical products except they contain double the on-chip memory.

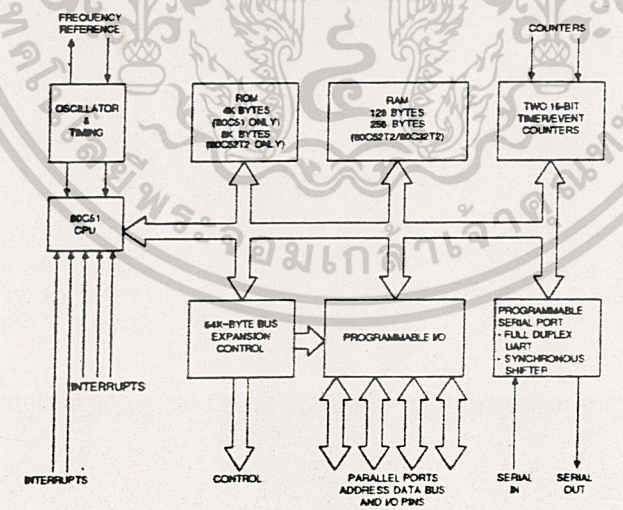
Both the 80C51BH and 80C31BH include 128 bytes of RAM, while the 80C52T2 and 80C32T2 include 256 bytes of RAM. The 80C51BH also includes 4K bytes of custom ROM program memory, while the 80C52T2 includes 8K bytes of ROM. The 80C52T2 and 80C32T2 are CMOS equivalents to the 8052AH and 8032AH except they contain three timers (T2) instead of two.

These CMOS products retain all of the features of the NMOS counterparts: 32 I/O lines; two 16-bit counter/timers; a full-duplex serial port; a five-source, two-level interrupt structure; and an on-chip oscillator and clock circuits.

In addition, all CMOS 80C51-based products have two software-selectable modes of reduced activity for further power conservation—idle and Power-Down. In the Idle mode, the CPU is frozen while the RAM, timers, serial port, and interrupt system continue to function. In the Power-Down mode, the RAM is saved and all other functions are inoperative.

The 80C51BH and 80C32T2 in PLCC packages offer improved noise tolerance by utilizing previously unused pins for additional V_{CC} and V_{SS} connections.

BLOCK DIAGRAM

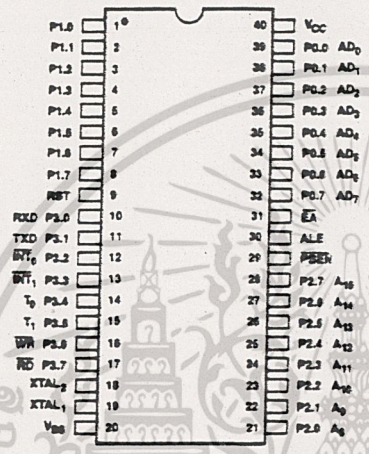


BD007232

Publication # 04815 Rev. D Amendment /0
Issue Date: October 1989

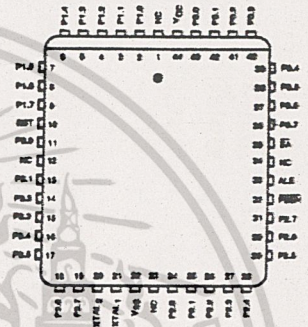
CONNECTION DIAGRAMS
Top View

DIP
80C51BH/80C31BH
80C52T2/80C32T2



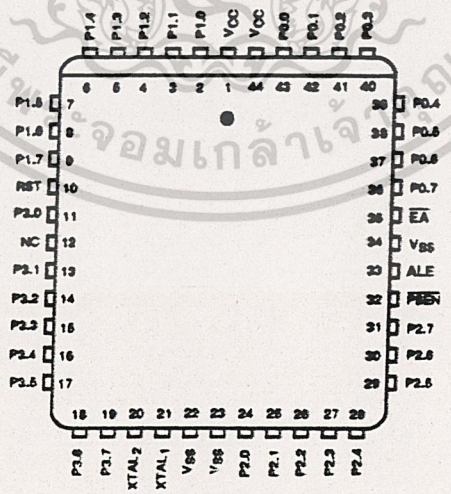
CD005554

PLCC
80C51BH/80C31BH



CD009443

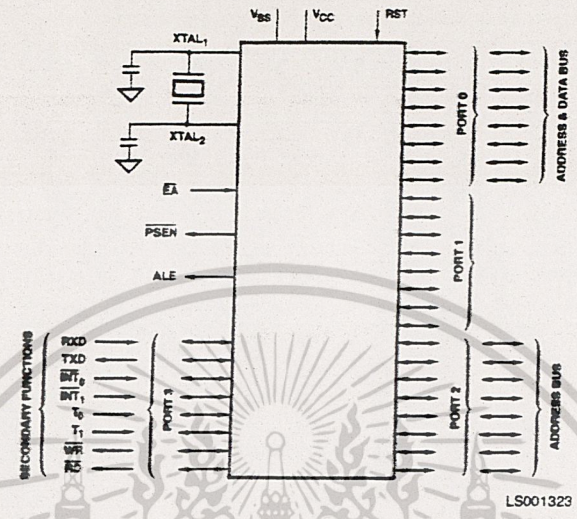
PLCC
80C52T2/80C32T2



CD009444

Note: Pin 1 is marked for orientation.

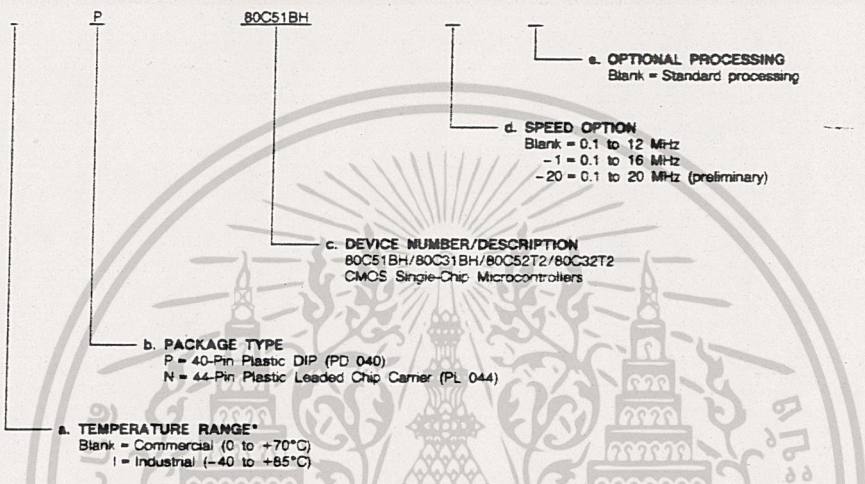
LOGIC SYMBOL



ORDERING INFORMATION

Commodity Products

AMD commodity products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of: **a. Temperature Range**
b. Package Type
c. Device Number
d. Speed Option
e. Optional Processing



Valid Combinations	
P, N IP, IN	80C51BH
	80C51BH-1
	80C31BH
	80C31BH-1
P	80C31BH-20
P, N IP, IN	80C52T2-1
	80C32T2-1

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

*This device will also be available in Military temperature range.

PIN DESCRIPTION

Port 0 (Bidirectional, Open Drain)
Port 0 is an open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float, and in that state can allow them to 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 1s. Port 0 also outputs the code bytes during program verification in the 80C51BH. External pullups are required during program verification.

Port 1 (Bidirectional)
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four LS TTL inputs. Port 1 pins that have 1s written to them are pulled High by the internal pullups and can be used as inputs while in this state. As inputs, Port 1 pins that are externally being pulled Low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the Low-order address bytes during program verification.

Port 2 (Bidirectional)
Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four LS TTL inputs. Port 2 pins having 1s written to them are pulled High by the internal pullups and can be used as inputs while in this state. As inputs, Port 2 pins externally being pulled Low will source current (I_{IL}) 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 1s. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function register.

Port 2 also receives the High-order address bits during ROM verification.

Port 3 (Bidirectional)
Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four LS TTL inputs. Port 3 pins that have 1s written to them are pulled High by the internal pullups and can be used as inputs while in this state. As inputs, Port 3 pins externally being pulled Low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features as listed below:

Port Pin	Alternate Function
$P_{3.0}$	RxD (serial input port)
$P_{3.1}$	TxD (serial output port)
$P_{3.2}$	$\overline{INT_0}$ (external interrupt 0)
$P_{3.3}$	$\overline{INT_1}$ (external interrupt 1)
$P_{3.4}$	T_0 (Timer 0 external input)
$P_{3.5}$	T_1 (Timer 1 external input)
$P_{3.6}$	$\overline{WR}$ (external Data Memory write strobe)
$P_{3.7}$	$\overline{RD}$ (external Data Memory read strobe)

RST Reset (Input, Active High)
A High on this pin (for two machine cycles while the oscillator is running) resets the device. An internal diffused resistor to V_{SS} permits power-on reset, using only an external capacitor to V_{CC}.

ALE Address Latch Enable (Output, Active High)
Address Latch Enable is the output pulse for latching the Low byte of the address during accesses to external memory.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, allowing use for external-timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$ Program Store Enable (Output, Active Low)
$\overline{PSEN}$ is the read strobe to external Program Memory. When the 80C51BH is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle—except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory. $\overline{PSEN}$ is not activated during fetches from internal Program Memory.

$\overline{EA}$ External Access Enable (Input, Active Low)
$\overline{EA}$ must be externally held Low to enable the device to fetch code from external Program Memory locations 0000H to 0FFFH. If $\overline{EA}$ is held High, the device executes from internal Program Memory unless the program counter contains an address greater than 0FFFH.

$XTAL_1$ Crystal (Input)
Input to the inverting-oscillator amplifier, and input to the internal clock-generator circuits.

$XTAL_2$ Crystal (Output)
Output from the inverting-oscillator amplifier.

V_{CC} Power Supply
Supply voltage during normal, idle, and power-down operations.

V_{SS} Circuit Ground

FUNCTIONAL DESCRIPTION

Oscillator Characteristics

$XTAL_1$ and $XTAL_2$ are the input and output, respectively, of an inverting amplifier which is configured for use as an on-chip oscillator (see Figure 1). Either a quartz crystal or ceramic resonator may be used.

To drive the device from an external clock source, $XTAL_1$ should be driven while $XTAL_2$ is left unconnected (see Figure 2). 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.

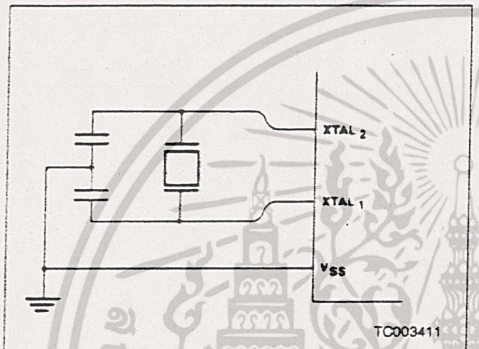


Figure 1. Crystal Oscillator

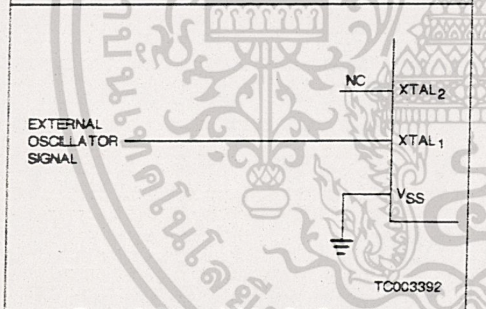


Figure 2. External Drive Configuration

Note: Different from NMOS configuration.

Idle and Power-Down Operation

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

These special modes are activated by software via the Special Function Register, PCON (Table 1). Its hardware address is 87H; PCON is not bit-addressable.

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

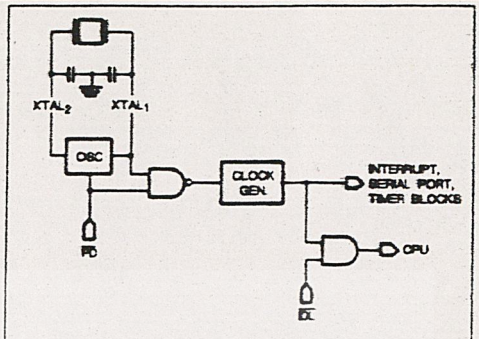


Figure 3. Idle and Power-Down Hardware

TABLE 1. PCON (Power Control Register)

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

Symbol	Position	Name and Description
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.

Idle Mode

The instruction that sets PCON.0 is the last instruction executed in the normal operating mode before 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 registers maintain their data during Idle. Table 2 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 a 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 mode 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 going into Power-Down. Once in Power-Down, the oscillator is stopped. Only the contents of the on-chip RAM are preserved. The Special Function Registers are not saved. A hardware reset is the only way of exiting the Power-Down mode.

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 2 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 P_2 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, P_1, shown in Figure 4.

80C51BH I/O Ports

The I/O port drive of the 80C51BH 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, P_1, turns on for two oscillator periods, pulling the output High very rapidly. As the output line is drawn High, pFET P_3 turns on through the inverter to supply the I_{OH} source current. This inverter and P_3 form a latch which holds the 1 and is supported by P_2.

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 its strong pullup turned on for the entire duration of the external memory access.

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 I_{TL} under the D.C Specifications. When the input goes below approximately 2 V, P_3 turns off to save I_{CC} current. Note, when returning to a logical 1, P_2 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.

TABLE 2. STATUS OF THE EXTERNAL PINS DURING IDLE AND POWER-DOWN MODES

Mode	Program Memory	ALE	$\overline{PSEN}$	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

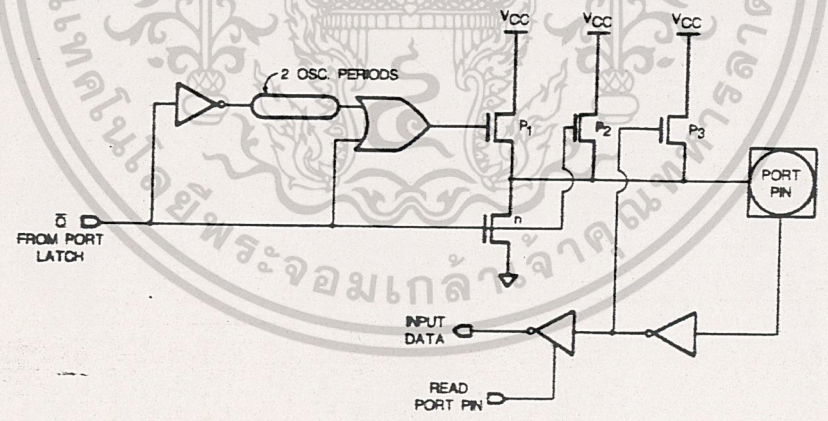


Figure 4. I/O Buffers in the 80C51BH (Ports 1, 2, 3)

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −65°C to +150°C
Voltage on Any
Pin to V_{SS} ..... −0.5 V to V_{CC} +0.5 V
Voltage on V_{CC} to V_{SS} ..... −0.5 V to 6.5 V
Power Dissipation ..... 200 mW

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
Temperature (T_A) ..... 0 to +70°C
80C51BH/80C31BH
Supply Voltage (V_{CC}) ..... +4 V to +6 V
80C52T2/80C32T2
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V
Ground (V_{SS}) ..... 0 V

Industrial (I) Devices
Temperature (T_A) ..... −40 to +85°C
80C51BH/80C31BH
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V
80C52T2/80C32T2
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V
Ground (V_{SS}) ..... 0 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating ranges unless otherwise specified

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
V_{IL}	Input Low Voltage (Except $\overline{EA}$)		−0.5	.2 V_{CC} − 0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		−0.5	.2 V_{CC} − 0.3	V
V_{IH}	Input High Voltage (Except $XTAL_1$, RST)		0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage ($XTAL_1$, RST)		0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage (Ports 1, 2, 3)	I_{OL} = 1.6 mA (Note 1)		0.45	V
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA (Note 1)		0.45	V
V_{OH}	Output High Voltage (Ports 1, 2, 3)	I_{OH} = −60 µA, V_{CC} = 5 V±10%	2.4		V
		I_{OH} = −25 µA	0.75 V_{CC}		V
		I_{OH} = −10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode, ALE, $\overline{PSEN}$)	I_{OH} = −400 µA, V_{CC} = 5 V±10%	2.4		V
		I_{OH} = −150 µA	0.75 V_{CC}		V
		I_{OH} = −40 µA (Note 2)	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)	V_{IN} = 0.45 V		−50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3)	V_{IN} = 2 V		−650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	150	kΩ
CIO	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{PD}	Power Down Current	V_{CC} = 2 to 6 V (Note 3)		50	µA

80C51BH/80C31BH MAXIMUM I_{CC} (mA)

	Operating (Note 4)			Idle (Note 5)		
Freq. / V_{CC}	4 V	5 V	6 V	4 V	5 V	6 V
0.1 MHz	1.2	1.5	2.5	0.5	0.7	1.1
3.5 MHz	4.3	5.7	7.5	1.1	1.6	2.2
8.0 MHz	8.3	11	14	1.8	2.7	3.7
12 MHz	12	16	20	2.5	3.7	5
16 MHz	16	20.5	25	3.5	5	6.5

80C52T2/80C32T2 MAXIMUM I_{CC} (mA)

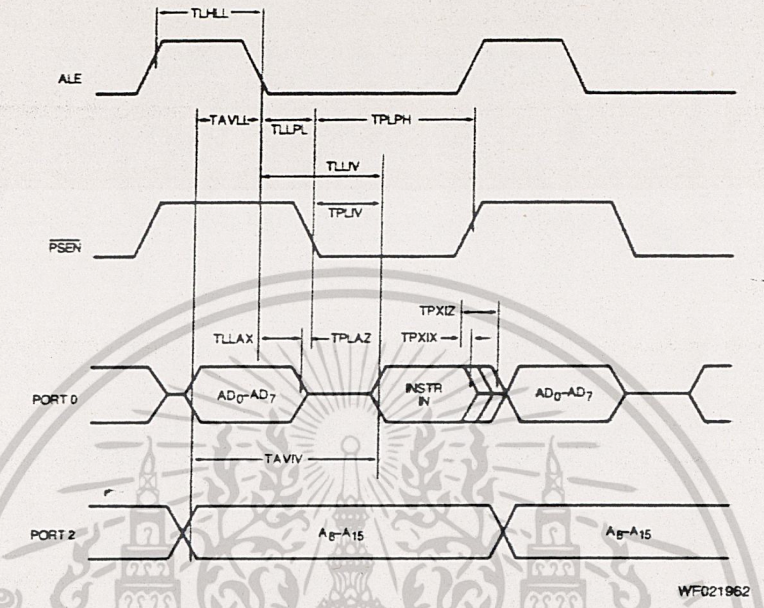
	Operating (Note 4)			Idle (Note 5)		
Freq. / V_{CC}	4.5 V	5.0 V	5.5 V	4.5 V	5.0 V	5.5 V
0.1 MHz	2.2	3.1	3.8	0.7	0.9	1.4
3.5 MHz	6	8	10	1.5	2	3
8.0 MHz	11	14	18	2.5	3.5	5
12 MHz	15	20	25	3.5	5	6
16 MHz	19	25	32	4.5	6.5	8.5

Notes:
1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}S 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 0.8 V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt-Trigger STROBE input.
2. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall before the .9 V_{CC} specification when the address bits are stabilizing.
3. Power-Down I_{CC} is measured with all outputs pins disconnected; $\overline{EA}$ = Port 0 = V_{CC}; $XTAL_2$ N.C.; RST = V_{SS}.
4. I_{CC} is measured with all output pins disconnected; $XTAL_1$ driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} − 0.5 V; $XTAL_2$ N.C. $\overline{EA}$ = RST = Port 0 = V_{CC}.
   I_{CC} would be slightly higher if a crystal oscillator is used.
5. Idle I_{CC} is measured with all output pins disconnected; $XTAL_1$ driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} − 0.5 V; $XTAL_2$ N.C. Port 0 = V_{CC}; $\overline{EA}$ = RST = V_{SS}.

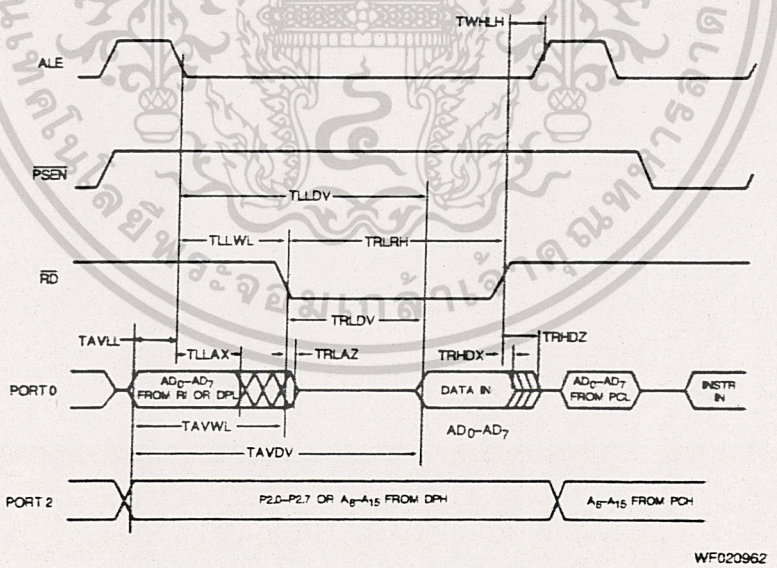
SWITCHING CHARACTERISTICS over operating ranges unless otherwise specified
(C_L for Port 0, ALE and $\overline{PSEN}$ Outputs = 100 pF; C_L for All Other Outputs = 80 pF)

Parameter Symbol	Parameter Description	20 MHz		16-MHz Osc.		12-MHz Osc.		Variable Oscillator		Units
		Min	Max	Min	Max	Min	Max	Min	Max	
External Program and Data Memory Characteristics										
1/TCLCL	Oscillator Frequency	0.1	20	0.1	16	0.1	12	0.1	16	MHz
TLHLL	ALE Pulse Width	60		85		127		2TCLCL – 40		ns
TAVLL	Address Valid to ALE Low	20		7		28		TCLCL – 55		ns
TLLAX	Address Hold After ALE Low	15		27		48		TCLCL – 35		ns
TLLIV	ALE Low to Valid Instr. In		120		150		234		4TCLCL – 100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	25		22		43		TCLCL – 40		ns
TPLPH	$\overline{PSEN}$ Pulse Width	115		142		205		3TCLCL – 45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr. In		75		83		145		3TCLCL – 105	ns
TPXIX	Input Instr. Hold After $\overline{PSEN}$	0		0		0		0		ns
TPXIZ	Input Instr. Float After $\overline{PSEN}$		35		38		59		TCLCL – 25	ns
TAVIV	Address to Valid Instr. In		165		208		312		5TCLCL – 105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		0		10		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	200		275		400		6TCLCL – 100		ns
TWLWH	$\overline{WR}$ Pulse Width	200		275		400		6TCLCL – 100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		145		148		252		5TCLCL – 165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		0		0		ns
TRHDZ	Data Float After $\overline{RD}$		60		55		97		2TCLCL – 70	ns
TLLDV	ALE Low to Valid Data In		310		350		517		8TCLCL – 150	ns
TAVDV	Address to Valid Data In		350		398		585		9TCLCL – 165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	100	200	137	238	200	300	3TCLCL – 50	3TCLCL + 50	ns
TAVWL	Address Valid to Read or Write Low	110		120		203		4TCLCL – 130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	95		2		23		TCLCL – 60		ns
TQVWH	Data Valid to Write High	200		287		433		7TCLCL – 150		ns
TWHQX	Data Hold After $\overline{WR}$	25		12		33		TCLCL – 50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	20	70	22	103	43	123	TCLCL – 40	TCLCL + 40	ns

SWITCHING WAVEFORMS

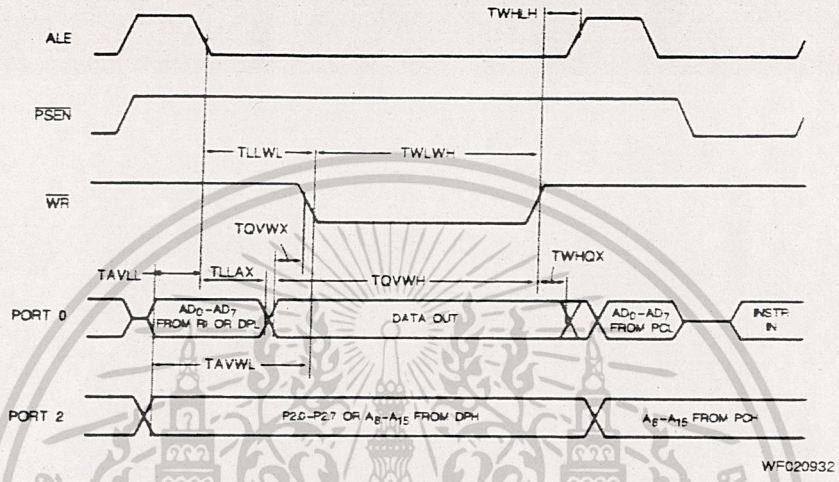


External Program Memory Read Cycle

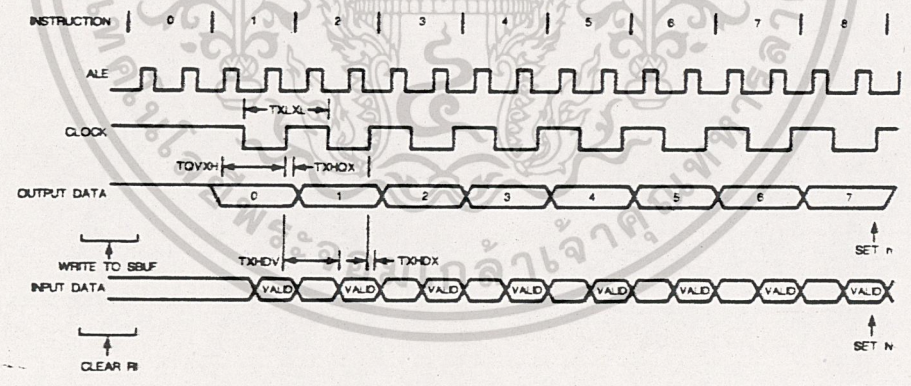


External Data Memory Read Cycle

SWITCHING WAVEFORMS (continued)



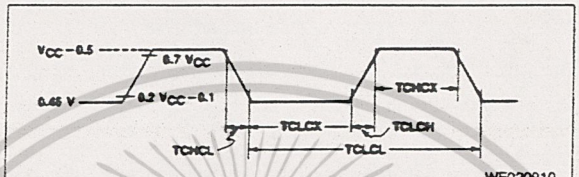
External Data Memory Write Cycle



Shift Register Timing Waveforms

EXTERNAL CLOCK DRIVE

Parameter Symbol	Parameter Description	Min.	Max.	Units
1/TCLCL	Oscillator Frequency	0.1	20	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

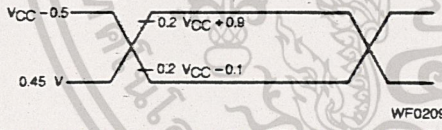


External Clock Drive Waveform

SERIAL PORT TIMING — SHIFT REGISTER MODE

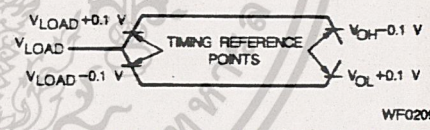
Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5 V ±20%; V_{SS} = 0 V; Load Capacitance = 80 pF

Parameter Symbol	Parameter Description	16 MHz Osc.		Variable Oscillator		Units
		Min.	Max.	Min.	Max.	
TXLXL	Serial Port Clock Cycle Time	750		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	492		10TCLCL – 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	8		2TCLCL – 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		492		10TCLCL – 133	ns



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 V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

AC Testing Input/Output Waveforms



For timing purposes a 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 V_{OH}/V_{OL} level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

Float Waveform

Instruction Set

INTRODUCTION

All members of the 8051 Family execute the same instruction set, optimized for 8-bit control applications. The instruction set provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. It provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that requrie Boolean processing. An overview of the instruction set is presented below, with a brief description of how certain instructions might be used.

PROGRAM STATUS WORD

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 4-1, resides in SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 1-7 . A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four banks is being referred to is made on the basis of the bits RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even.

Two bits in the PSW are uncommitted and may be used as general purpose status flags.

ADDRESSING MODES

The addressing modes in the 8051 Family instruction set are as follows:

Direct Addressing

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

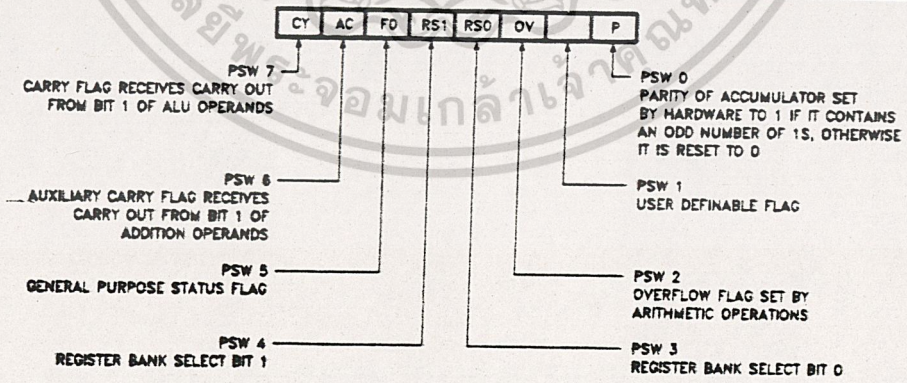


Figure 4-1. PSW (Program Status Word) Register in 8051 Family Devices

Register Instructions

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

Register-Specific Instructions

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumlator as A assemble as accumulator-specific opcodes.

Immediate Constants

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV A, #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

Indexed Addressing

Only Program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number. The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

ARITHMETIC INSTRUCTIONS

The menu of arithmetic instructions is listed in Table 4-1. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A, <byte> instruction can be written as:

```
ADD   A,7FH     (direct addressing)
ADD   A,@R0     (indirect addressing)
ADD   A,R7      (register addressing)
ADD   A,#127    (immediate constant)
```

Table 4-1. A List of the 8051 Family Arithmetic Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

The execution times listed in Table 4-1 assume a 12MHz clock frequency. All of the arithmetic instructions execute in 1 μs except the INC DPTR instruction, which takes 2 μs, and the Multiply and Divide instructions, which take 4 μs.

Note that any byte in the internal Data Memory space can be incremented or decremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2^n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4 μs and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

LOGICAL INSTRUCTIONS

Table 4-2 shows the list of 8051 Family logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and <byte> contains 01010011B, then

```
ANL     A,<byte>
```

will leave the Accumulator holding 00010001B.

Table 4-2. A List of the 8051 Family Logical Instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> .AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

The addressing modes that can be used to access the <byte> operand are listed in Table 3. Thus, the ANL A,<byte> instruction may take any of the forms

```
ANL   A,7FH     (direct addressing)
ANL   A,@R1     (indirect addressing)
ANL   A,R6      (register addressing)
ANL   A,#53H    (immediate constant)
```

All of the logical instructions that are Accumulator-specific execute in 1μs (using a 12 MHz clock). The others take 2 μs.

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

```
XRL   P1,#0FFH
```

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to stack it in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

```
MOV   B,#10
DIV   AB
SWAP  A
ADD   A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

DATA TRANSFERS

Internal RAM

Table 4-3 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. With a 12 MHz clock, all of these instructions execute in either 1 or 2 μs.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember the Upper 128 byes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in all 8051 Family devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128, if they are implemented, but not into SFR space.

Table 4-3. 8051 Family Data Transfer Instructions that Access Internal Data Memory Space

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μs

Figure 4-2. Shifting a BCD Number Two Digits to the Right

The Upper 128 are not implemented in 8051 Family devices with 128 bytes of RAM. With these devices, if the SP points to the Upper 128, PUSHed bytes are lost, and POPed bytes are indeterminate.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A,@Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting an 8-digit BCD number two digits to the right. Figure 4-2 shows how this can be done using direct MOVs, and for comparison how it can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes and 9 μs of execution time (assuming a 12 MHz clock). The same operation with XCHs uses less code and executes almost twice as fast.

		2A	2B	2C	2D	2E	ACC
	MOV R1,#2EH	00	12	34	56	78	XX
	MOV R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:						
LOOP:	MOV A,@R1	00	12	34	56	78	78
	XCHD A,@R0	00	12	34	58	78	76
	SWAP A	00	12	34	58	78	67
	MOV @R1,A	00	12	34	58	67	67
	DEC R1	00	12	34	58	67	67
	DEC R0	00	12	34	58	67	67
	CJNE R1,#2AH,LOOP						
	loop for R1 = 2DH:	00	12	38	45	67	45
	loop for R1 = 2CH:	00	18	23	45	67	23
	loop for R1 = 2BH:	08	01	23	45	67	01
	CLR A	08	01	23	45	67	00
	XCH A,2AH	00	01	23	45	67	08

Figure 4-3. Shifting a BCD Number One Digit to the Right

To right-shift by an odd number of digits, a one-digit shift must be executed. Figure 4-3 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the Accumulator are shown alongside each instruction.

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not Equal) is a loop control that will be described later.

The loop is executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

External RAM

Table 4-4 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or

R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few K bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few K bytes of RAM, as shown in Figure 1-5, without having to sacrifice all of Port 2.

All of these instructions execute in 2 μs, with a 12 MHz clock.

Table 4-4. 8051 Family Data Transfer Instructions that Access External Data Memory Space

Address Width	Mnemonic	Operation	Execution Time (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines. More about that later.

Lookup Tables

Table 4-5 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated. The mnemonic is MOVC for "move constant".

If the table access is to external Program Memory, then the read strobe is $\overline{PSEN}$.

Table 4-5. The 8051 Family Lookup Table Read Instructions

Mnemonic	Operation	Execution Time (μs)
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

The first MOVC instruction in Table 4-5 can accommodate a table of up to 256 entries, numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to beginning of the table. Then

```
MOVC    A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
        MOV     A,ENTRY_NUMBER
        CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:  MOVC    A,@A+PC
        RET
```

The table itself immediately follows the RET (return) instruction in Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 can not be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

BOOLEAN INSTRUCTIONS

8051 Family devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 other addressable bits. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

Table 4-6. A List of the 8051 Family
Boolean Instructions

Mnemonic		Operation	Execution Time (μs)
ANL	C,bit	C = C .AND. bit	2
ANL	C,/bit	C = C .AND. .NOT. bit	2
ORL	C,bit	C = C .OR. bit	2
ORL	C,/bit	C = C .OR. .NOT. bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT. C	1
CPL	bit	bit = .NOT. bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

The instruction set for the Boolean processor is shown in Table 4-6. All bit accesses are by direct addressing. Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV     C,FLAG
MOV     P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

```
C = bit1 .XRL. bit2
```

The software to do that could be as follows:

```
        MOV     C,bit1
        JNB     bit2,OVER
        CPL     C
OVER:   (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1 C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0 the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation.

All the PSW bits are directly addressable, so the Parity bit, or the general purpose flags, for example, are also available to the bit-test instructions.

Relative Offset

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program Memory. However, the destination address assembles to a relative offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore − 128 to + 127 Program Memory bytes relative to the first byte following the instruction.

JUMP INSTRUCTIONS

Table 4-7 shows the list of unconditional jumps.

Table 4-7. Unconditional Jumps In 8051 Family Devices

Mnemonic	Operation	Execution Time (μs)
JMP addr	Jump to addr	2
JMP @A + DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

The Table lists a single "JMP addr" instruction, but in fact there are three—SJMP, LJMP and AJMP—which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of −128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table, and the Accumulator is given an index to the table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV     DPTR,#JUMP_TABLE
MOV     A,INDEX_NUMBER
RL      A
JMP     @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP_TABLE:
        AJMP    CASE_0
        AJMP    CASE_1
        AJMP    CASE_2
        AJMP    CASE_3
        AJMP    CASE_4
```

Table 4-7 shows a single "CALL addr" instruction, but there are two of them—LCALL and ACALL—which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 4-8. Conditional Jumps in 8051 Family Devices

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A = <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

Table 4-8 shows the list of conditional jumps available to the 8051 Family user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of −128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
        MOV     COUNTER,#10
LOOP:   (begin loop)
          .
          .
          .
        (end loop)
        DJNZ    COUNTER,LOOP
        (continue)
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 4-3. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure 4-3, the two bytes were the data in R1 and the constant 2AH. The initial data in R1 was 2EH. Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

Table 4-9. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Chapter 2, page 2-24

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
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) addressed 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. A branch 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 conditional 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.

* — New operation not provided by 8048AH/8049AH.

Mnemonic		Description	Byte	Oscillator 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
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#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 4-9. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator 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

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
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

Table 4-9. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
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
MOV	@Ri,direct	Move direct byte 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
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@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 onto 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 low-order Digit indirect RAM with Acc	1	12
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 diret 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

Table 4-9. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING				
ACALL	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
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect 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
CJNE	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 addr11

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 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "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

$(PC) \leftarrow (PC) + 2$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{7\text{-}0})$
$(SP) \leftarrow (SP) + 1$
$((SP)) \leftarrow (PC_{15\text{-}8})$
$(PC_{10\text{-}0}) \leftarrow$ 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 if 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

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	1 r r r

Operation: ADD
(A) ← (A) + (Rn)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 1	direct address

Operation: ADD
(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 1 1 i

Operation: ADD
$(A) \leftarrow (A) + ((R_i))$

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 0	immediate data

Operation: ADD
$(A) \leftarrow (A) + \#data$

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 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 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) 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

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	1 r r r

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

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 1	direct address

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

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 1 1 i

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

ADDC A,#data

Bytes: 2

Cycles: 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 2K 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 at location 0345H and will load the PC with 0123H.

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

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	0 1 1 i

Operation: ANL
(A) ← (A) ∧ ((Ri))

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 0	immediate data

Operation: ANL
(A) ← (A) ∧ #data

```
0001   0000
0002   0000          B      .EQU    0F0H    ;B REGISTER
0003   0000          ACC    .EQU    0E0H    ;ACCUMULATOR
0004   0000          PSW    .EQU    0D0H    ;PROGRAM STATUS WORD
0005   0000          IPC    .EQU    0B8H    ;INTERRUPT PRIORITY
0006   0000          P3     .EQU    0B0H    ;PORT 3
0007   0000          IEC    .EQU    0A8H    ;INTERRUPT ENABLE
0008   0000          P2     .EQU    0A0H    ;PORT 2
0009   0000          SBUF   .EQU    99H     ;SEND BUFFER
0010   0000          SCON   .EQU    98H     ;SERIAL CONTROL
0011   0000          P1     .EQU    90H     ;PORT 1
0012   0000          TH1    .EQU    8DH     ;TIMER 1 HIGH
0013   0000          TH0    .EQU    8CH     ;TIMER 0 HIGH
0014   0000          TL1    .EQU    8BH     ;TIMER 1 LOW
0015   0000          TL0    .EQU    8AH     ;TIMER 0 LOW
0016   0000          TMOD   .EQU    89H     ;TIMER MODE
0017   0000          TCON   .EQU    88H     ;TIMER CONTROL
0018   0000          PCON   .EQU    87H     ;POWER CONTROL REGISTER
0019   0000          DPH    .EQU    83H     ;DATA POINTER HIGH
0020   0000          DPL    .EQU    82H     ;DATA POINTER LOW
0021   0000          SP     .EQU    81H     ;STACK POINTER
0022   0000          P0     .EQU    80H     ;PORT 0
0023   0000          ;
0024   0000          ;MCS-51 INTERNAL BIT ADDRESSES
0025   0000          ;
0026   0000          CY:    .EQU    0D7H    ;CARRY FLAG
0027   0000          AC:    .EQU    0D6H    ;AUXILIARY-CARRY FLAG
0028   0000          F0:    .EQU    0D5H    ;USER FLAG 0
0029   0000          RS1:   .EQU    0D4H    ;REGISTER SELECT MSB
0030   0000          RS0:   .EQU    0D3H    ;REGISTER SELECT LSB
0031   0000          OV:    .EQU    0D2H    ;OVERFLOW FLAG
```

```
0032  0000          P:     .EQU    0D0H   ;PARITY FLAG
0033  0000          PS:    .EQU    0BCH   ;PRIORITY SERIAL PORT
0034  0000          PT1:   .EQU    0BBH   ;PRIORITY TIMER 1
0035  0000          PX1:   .EQU    0BAH   ;PRIORITY EXTERNAL 1
0036  0000          PT0:   .EQU    0B9H   ;PRIORITY TIMER 0
0037  0000          PX0:   .EQU    0B8H   ;PRIORITY EXTERNAL 0
0038  0000          EA:    .EQU    0AFH   ;ENABLE ALL INTERRUPT
0039  0000          ES:    .EQU    0ACH   ;ENABLE SERIAL INTERRUPT
0040  0000          ET1:   .EQU    0ABH   ;ENABLE TIMER 1 INTERRUPT
0041  0000          EX1:   .EQU    0AAH   ;ENABLE EXTERNAL 1 INTERR
0042  0000          ET0:   .EQU    0A9H   ;ENABLE TIMER 0 INTERRUPT
0043  0000          EX0:   .EQU    0A8H   ;ENABLE EXTERNAL 0 INTERR
0044  0000          SM0:   .EQU    09FH   ;SERIAL MODE 0
0045  0000          SM1:   .EQU    09EH   ;SERIAL MODE 1
0046  0000          SM2:   .EQU    09DH   ;SERIAL MODE 2
0047  0000          REN:   .EQU    09CH   ;SERIAL RECEPTION ENABLE
0048  0000          TB8:   .EQU    09BH   ;TRANSMITT BIT 8
0049  0000          RB8:   .EQU    09AH   ;RECEIVE BIT 8
0050  0000          TI:    .EQU    099H   ;TRANSMIT INTERRUPT FLAG
0051  0000          RI:    .EQU    098H   ;RECEIVE INTERRUPT FLAG
0052  0000          TF1:   .EQU    08FH   ;TIMER 1 OVERFLOW FLAG
0053  0000          TR1:   .EQU    08EH   ;TIMER 1 RUN CONTROL BIT
0054  0000          TF0:   .EQU    08DH   ;TIMER 0 OVERFLOW FLAG
0055  0000          TR0:   .EQU    08CH   ;TIMER 0 RUN CONTROL BIT
0056  0000          IE1:   .EQU    08BH   ;EXT INTERR. 1 EDGE FLAG
0057  0000          IT1:   .EQU    08AH   ;EXT INTERR. 1 TYPE FLAG
0058  0000          IE0:   .EQU    089H   ;EXT INTERR. 0 EDGE FLAG
0059  0000          IT0:   .EQU    088H   ;EXT INTERR. 0 TYPE FLAG
0060  0000
0061  0000          ;*****************************************************
0062  0000          ;
```

```
0063   0000
0064   0000                .ORG    0000H
0065   0000 02 01 00        LJMP   START
0066   0003
0067   0003
0068   0003                .ORG    003H
0069   0003 02 01 50        LJMP   REC0     ;INT0 TO RECORD CH 0
0070   0006
071   0013                 .ORG    013H
0072   0013 02 00 80        LJMP    CALL   ;INT1 TO SELECTED CALL IN
0073   0016
0074   0080                .ORG    080H
0075   0080 BF 00 03    CALL:   CJNE   R7,#00H,CL
0076   0083 02 05 00        LJMP   REC1    ;IF  R7 =00  JUMP TO REC1
0077   0086 02 01 10    CL: LJMP   CHECK
0078   0089                ;*********  MAIN  PROGRAM  ***************
0079   0089
0080   0089
0081   0100                .ORG    0100H
0082   0100 D2 AF     START:SET    EA       ;ENABLE ALL INTTERRUPT
0083   0102 C2 AA           CLR    EX1      ;DISABLE INT1
0084   0104 C2 8A           CLR    IT1      ;SET INT1 FOR TRANSITION TRIG
0085   0106 D2 A8           SETB   EX0      ; ENABLE INT0
0086   0108 C2 88           CLR    IT0      ;SET INT0 TYPE LOW LEVEL TRIG
0087   010A 7F 00           MOV    R7,#00H ;CLEAR  COUNTER
0088   010C 74 00           MOV    A,#00H
0089   010E F5 90           MOV    P1,A     ;RESET PORT 0
0090   0110 74 80     CHECK:MOV    A,#080H
0091   0112 F5 90           MOV    P1,A     ;SET PORT 0 FOR READY STATE
0092   0114 30 B7 69        JNB    0B7H,PLAY ;IF P3.7  IS ZERO GOTO PLAY
0093   0117 02 01 10        LJMP   CHECK    ;GOTO PLAY
```

```
0094   011A
0095   011A
0096   011A
0097   011A
0098   011A              ;******  MODULE TO RECORD CHANNEL 0 ******
0099   011A
0100   0150              REC0:  ORG     0150H
0101   0150 74 00               MOV     A,#00H
0102   0152 F5 90               MOV     P1,A
0103   0154 12 02 80            LCALL   DELAY2    ; RESET
0104   0157 74 80               MOV     A,#80H
0105   0159 F5 90               MOV     P1,A
0106   015B 12 02 80            LCALL   DELAY2    ; NORMAL
0107   015E 74 D0               MOV     A,#0D0H
0108   0160 F5 90               MOV     P1,A
0109   0162 12 02 50            LCALL   DELAY1    ;RECORD CHANNEL 0
0110   0165 74 80               MOV     A,#80H
0111   0167 F5 90               MOV     P1,A
0112   0169 12 02 80            LCALL   DELAY2
0113   016C 74 64               MOV     A,#0A0
0114   016E F5 90               MOV     P1,A
0115   0170 12 02 80            LCALL   DELAY2     ;STOP THE RECORD CHANNEL 0
0116   0173 D2 AA               SETB    EX1       ;ENABLE  INT1
0117   0175 D2 8A               SETB    IT1       ;SET INT1 FOR TRANSITION
0118   0177 7F 00               MOV     R7,#00H
0119   0179 32                  RETI
0120   017A
0121   017A
0122   017A
0123   017A              ;******* MODULE  PLAY CHANNEL 1,2,3 *******
0124   017A
```

```
Ø125   Ø17A
Ø126   Ø180              PLAY:  .ORG    Ø180H
Ø127   Ø180 74 Ø1               MOV     A,#Ø1H
Ø128   Ø182 F5 9Ø               MOV     P1,A
Ø129   Ø184 12 Ø2 8Ø            LCALL   DELAY2
Ø130   Ø187 74 81               MOV     A,#Ø81H
Ø131   Ø189 F5 9Ø               MOV     P1,A
Ø132   Ø18B 12 Ø2 8Ø            LCALL   DELAY2
Ø133   Ø18E 74 C1               MOV     A,#ØC1H
Ø134   Ø19Ø F5 9Ø               MOV     P1,A
Ø135   Ø192 12 Ø2 8Ø            LCALL   DELAY2    ;PLAY CH 1 & DELAY 3Ø
Ø136   Ø195 Ø2 Ø1 1Ø            LJMP    CHECK
Ø137   Ø198
Ø138   Ø198
Ø139   Ø198              ;******  REM DELAY1 = 25 SEC  **********
Ø140   Ø198
Ø141   Ø198
Ø142   Ø25Ø              DELAY1 .ORG    Ø25ØH
Ø143   Ø25Ø 79 3Ø               MOV     R1,#Ø3ØH
Ø144   Ø252 ØØ           D11:   NOP
Ø145   Ø253 ØØ                  NOP
Ø146   Ø254 ØØ                  NOP
Ø147   Ø255 7A FF               MOV     R2,#ØFFH
Ø148   Ø257 ØØ           D12:   NOP
Ø149   Ø258 ØØ                  NOP
Ø15Ø   Ø259 ØØ                  NOP
Ø151   Ø25A 7B FF               MOV     R3,#ØFFH
Ø152   Ø25C ØØ           D13:   NOP
Ø153   Ø25D ØØ                  NOP
Ø154   Ø25E ØØ                  NOP
Ø155   Ø25F DB FB               DJNZ    R3,D13
```

```
0156   0261 DA F4              DJNZ      R2,D12
0157   0263 D9 ED              DJNZ      R1,D11
0158   0265 22                 RET
0159   0266
0160   0266
0161   0266
0162   0266            ;****** REM DELAY2 = 1.5 SEC ***********
0163   0266
0164   0266
0165   0280                    .ORG      0280H
0166   0280 79 02      DELAY2  MOV       R1,#02H
0167   0282 00         D21:    NOP
0168   0283 00                 NOP
0169   0284 00                 NOP
0170   0285 7A FF              MOV       R2,#0FFH
0171   0287 00         D22:    NOP
0172   0288 00                 NOP
0173   0289 00                 NOP
0174   028A 7B FF              MOV       R3,#0FFH
0175   028C 00         D23:    NOP
0176   028D 00                 NOP
0177   028E 00                 NOP
0178   028F DB FB              DJNZ      R3,D23
0179   0291 DA F4              DJNZ      R2,D22
0180   0293 D9 ED              DJNZ      R1,D21
0181   0295 22                 RET
0182   0296
0183   0296
0184   0296
0185   0296            ;******* MODULE PLAY CHANNEL 0 **********
0186   0296
```

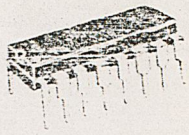
```
0187   0420                    .ORG          0420H
0188   0420
0189   0420              PLAY0:
0190   0420 74 80              MOV       A,#80H
0191   0422 F5 90              MOV       P1,A
0192   0424 12 02 80           LCALL     DELAY2
0193   0427 74 C0              MOV       A,#0C0H
0194   0429 F5 90              MOV       P1,A
0195   042B 12 02 50           LCALL     DELAY1  ;PLAY CH 0 & DELAY  30 SEC
0196   042E 22                 RET
0197   042F
0198   042F
0199   042F
0200   042F
0201   0500                    .ORG      0500H
0202   0500              REC1:
0203   0500
0204   0500 12 04 20           LCALL     PLAY0
0205   0503 74 01              MOV       A,#01H
0206   0505 F5 90              MOV       P1,A
0207   0507 12 02 80           LCALL     DELAY2
0208   050A 74 81              MOV       A,#081H
0209   050C F5 90              MOV       P1,A
0210   050E 12 02 80           LCALL     DELAY2
0211   0511 74 D1              MOV       A,#0D1H
0212   0513 F5 90              MOV       P1,A
0213   0515 12 02 50           LCALL     DELAY1
0214   0518 74 81              MOV       A,#081H
0215   051A F5 90              MOV       P1,A
0216   051C 12 02 80           LCALL     DELAY2
0217   051F 74 A1              MOV       A,#0A1H
```

```
0218    0521 F5 90              MOV         P1,A
0219    0523 12 02 80           LCALL       DELAY2
0220    0526 7F 00              MOV         R7,#00H
0221    0528 32                 RETI
0222    0529                    .END
```

MCM6256B

Advance Information

256K-Bit Dynamic RAM



P PACKAGE
PLASTIC
CASE 648

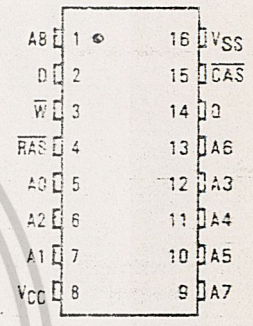
The MCM6256B is a 262,144 bit, high-speed, dynamic random access memory. Organized as 262,144 one-bit words and fabricated using N-channel silicon-gate MOS technology, this single +5 volt supply dynamic RAM combines high performance with low cost and improved reliability. All inputs and outputs are fully TTL compatible.

By multiplexing row and column address inputs, the MCM6256B requires only nine address lines and permits packaging in standard 16-pin 300 mil wide dual-in-line packages. Complete address decoding is done on-chip with address latches incorporated. Data out (Q) is controlled by $\overline{CAS}$ allowing greater system flexibility.

The MCM6256B features "page mode" which allows random column accesses of the 512 bits within the selected row.

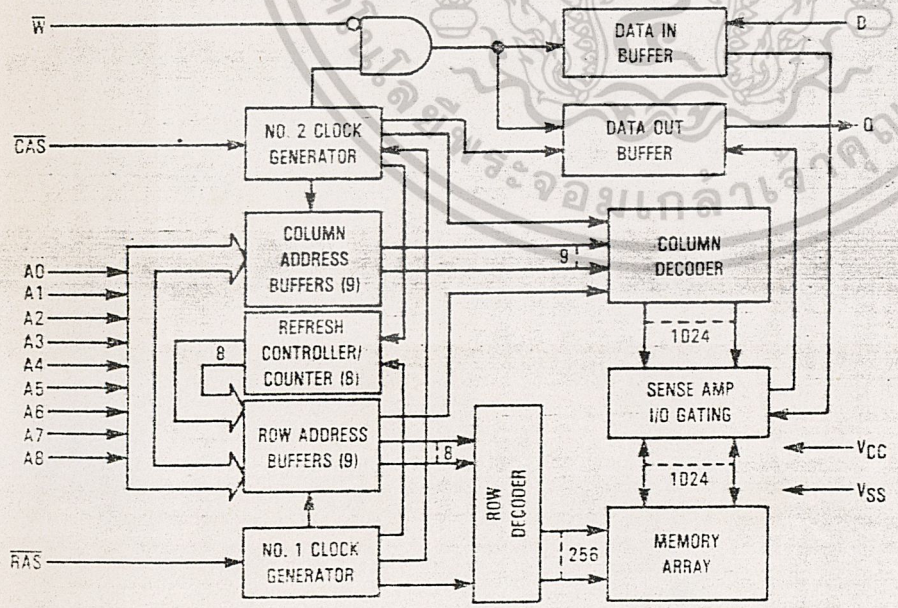
- Organized as 262,144 Words of 1 Bit
- Single +5 Volt Operation (±10%)
- Maximum Access Time: MCM6256B-10 = 100 ns
  MCM6256B-12 = 120 ns
  MCM6256B-15 = 150 ns
- Low Power Dissipation: MCM6256B-10 = 440 mW Maximum (Active)
  MCM6256B-12 = 396 mW Maximum (Active)
  MCM6256B-15 = 358 mW Maximum (Active)
  28 mW Maximum (Standby)
- Three-State Data Output
- Early-Write Common I/O Capability
- 256 Cycle, 4 ms Refresh
- $\overline{RAS}$-Only Refresh Mode
- $\overline{CAS}$ Before $\overline{RAS}$ Refresh
- Hidden Refresh
- Page Mode Capability

PIN ASSIGNMENT



PIN NAMES	
A0-A8	Address Input
D	Data In
Q	Data Out
$\overline{W}$	Read/Write Input
$\overline{RAS}$	Row Address Strobe
$\overline{CAS}$	Column Address Strobe
V_{CC}	Power (+5 V)
V_{SS}	Ground

BLOCK DIAGRAM



This document contains information on a new product. Specifications and information herein are subject to change without notice.

ABSOLUTE MAXIMUM RATINGS (See Note)

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	−1 to +7	V
Voltage Relative to V_{SS} for Any Pin Except V_{CC}	V_{in}, V_{out}	−1 to +7	V
Data Out Current	I_{out}	50	mA
Power Dissipation	P_D	1	W
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

NOTE: Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to RECOMMENDED OPERATING CONDITIONS. Exposure to higher than recommended voltages for extended periods of time could affect device reliability.

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

DC OPERATING CONDITIONS AND CHARACTERISTICS

(V_{CC} = 5.0 V ±10%, T_A = 0 to 70°C, Unless Otherwise Noted)

RECOMMENDED OPERATING CONDITIONS

Parameter	Symbol	Min	Typ	Max	Unit	Notes
Supply Voltage (Operating Voltage Range)	V_{CC}	4.5	5.0	5.5	V	1
	V_{SS}	0	0	0	V	1
Logic 1 Voltage, All Inputs	V_{IH}	2.4	—	6.5	V	1
Logic 0 Voltage, All Inputs	V_{IL}	−1.0	—	0.8	V	1

DC CHARACTERISTICS

Characteristic	Symbol	Min	Max	Unit	Notes
V_{CC} Power Supply Current	I_{CC1}			mA	2
MCM6256B-10, t_{RC} = 190 ns		—	80		
MCM6256B-12, t_{RC} = 220 ns		—	72		
MCM6256B-15, t_{RC} = 260 ns		—	65		
V_{CC} Power Supply Current (Standby) ($\overline{RAS}$ = $\overline{CAS}$ = V_{IH})	I_{CC2}	—	5.0	mA	
V_{CC} Power Supply Current During $\overline{RAS}$ only Refresh Cycles ($\overline{CAS}$ = V_{IH})	I_{CC3}			mA	2
MCM6256B-10, t_{RC} = 190 ns		—	70		
MCM6256B-12, t_{RC} = 220 ns		—	62		
MCM6256B-15, t_{RC} = 260 ns		—	55		
V_{CC} Power Supply Current During Page Mode Cycle ($\overline{RAS}$ = V_{IL})	I_{CC4}			mA	2
MCM6256B-10, t_{PC} = 100 ns		—	60		
MCM6256B-12, t_{PC} = 120 ns		—	55		
MCM6256B-15, t_{PC} = 145 ns		—	50		
V_{CC} Power Supply Current During $\overline{CAS}$ Before $\overline{RAS}$ Refresh	I_{CC5}			mA	2
MCM6256B-10, t_{RC} = 190 ns		—	70		
MCM6256B-12, t_{RC} = 220 ns		—	62		
MCM6256B-15, t_{RC} = 260 ns		—	55		
Input Leakage Current (V_{SS} < V_{in} < V_{CC})	$I_{lkg(I)}$	−10	10	μA	
Output Leakage Current ($\overline{CAS}$ at Logic 1, V_{SS} < V_{out} < V_{CC})	$I_{lkg(O)}$	−10	10	μA	
Output Logic 1 Voltage (I_{out} = −5 mA)	V_{OH}	2.4	—	V	
Output Logic 0 Voltage (I_{out} = 4.2 mA)	V_{OL}	—	0.4	V	

CAPACITANCE (f = 1.0 MHz, T_A = 25°C, V_{CC} = 5 V, Periodically Sampled Rather Than 100% Tested)

Parameter		Symbol	Typ	Max	Unit	Notes
Input Capacitance	A0-A8, D	C_{in}	—	5	pF	3
	$\overline{RAS}$, $\overline{CAS}$, $\overline{W}$		—	7	pF	3
Output Capacitance ($\overline{CAS}$ = V_{IH} to Disable Output)	Q	C_{out}	—	7	pF	3

NOTES:

1. All voltages referenced to V_{SS}.
2. Current is a function of cycle rate and output loading; maximum current is measured at the fastest cycle rate with the output open.
3. Capacitance measured with a Boonton Meter or effective capacitance calculated from the equation: $C = I\Delta t/\Delta V$.

AC OPERATING CONDITIONS AND CHARACTERISTICS

(V_{CC} = 5.0 V ±10%, T_A = 0 to 70°C, Unless Otherwise Noted)

READ, WRITE, AND READ-MODIFY-WRITE CYCLES (See Notes 1, 2, 3, and 5)

Parameter	Symbol		MCM6256B-10		MCM6256B-12		MCM6256B-15		Unit	Notes
	Standard	Alternate	Min	Max	Min	Max	Min	Max		
Random Read or Write Cycle Time	t_{RELREL}	t_{RC}	190	—	220	—	260	—	ns	4, 5
Read-Write Cycle Time	t_{RELREL}	t_{RWC}	200	—	240	—	265	—	ns	4, 5
Read-Modify-Write Cycle Time	t_{RELREL}	t_{RMW}	220	—	260	—	310	—	ns	4, 5
Access Time from $\overline{RAS}$	t_{RELQV}	t_{RAC}	—	100	—	120	—	150	ns	6, 7
Access Time from $\overline{CAS}$	t_{CELQV}	t_{CAC}	—	50	—	60	—	75	ns	7, 8
Output Buffer and Turn-Off Delay	t_{CEHQZ}	t_{OFF}	5	25	5	30	5	35	ns	9
$\overline{RAS}$ Precharge Time	t_{REHREL}	t_{RP}	80	—	90	—	100	—	ns	—
$\overline{RAS}$ Pulse Width	t_{RELREH}	t_{RAS}	100	10,000	120	10,000	150	10,000	ns	—
$\overline{CAS}$ Pulse Width	t_{CELCEH}	t_{CAS}	50	10,000	60	10,000	75	10,000	ns	—
$\overline{RAS}$ to $\overline{CAS}$ Delay Time	t_{RELCEL}	t_{RCD}	25	50	25	60	25	75	ns	10
Row Address Setup Time	t_{AVREL}	t_{ASR}	0	—	0	—	0	—	ns	—
Row Address Hold Time	t_{RELAX}	t_{RAH}	15	—	15	—	15	—	ns	—
Column Address Setup Time	t_{AVCEL}	t_{ASC}	0	—	0	—	0	—	ns	—
Column Address Hold Time	t_{CELAX}	t_{CAH}	20	—	25	—	30	—	ns	—
Column Address Hold Time Referenced to $\overline{RAS}$	t_{RELAX}	t_{AR}	70	—	85	—	105	—	ns	—
Transition Time (Rise and Fall)	t_T	t_T	3	50	3	50	3	50	ns	—
Read Command Setup Time	t_{WHCEL}	t_{RCS}	0	—	0	—	0	—	ns	—
Read Command Hold Time Referenced to $\overline{CAS}$	t_{CEHWX}	t_{RCH}	0	—	0	—	0	—	ns	11
Read Command Hold Time Referenced to $\overline{RAS}$	t_{REHWX}	t_{RRH}	10	—	15	—	20	—	ns	11
Write Command Hold Time	t_{CELWH}	t_{WCH}	20	—	25	—	30	—	ns	—
Write Command Hold Time Referenced to $\overline{RAS}$	t_{RELWH}	t_{WCR}	70	—	85	—	105	—	ns	—
Write Command Pulse Width	t_{WLWH}	t_{WP}	20	—	25	—	30	—	ns	—
Write Command to $\overline{RAS}$ Lead Time	t_{WLREH}	t_{RWL}	25	—	35	—	45	—	ns	—
Write Command to $\overline{CAS}$ Lead Time	t_{WLCEH}	t_{CWL}	25	—	35	—	45	—	ns	—
Data in Setup Time	t_{DVCEL}	t_{DS}	0	—	0	—	0	—	ns	12
Data in Hold Time	t_{CELDX}	t_{DH}	20	—	25	—	30	—	ns	12
Data in Hold Time Referenced to $\overline{RAS}$	t_{RELDX}	t_{DHR}	70	—	85	—	105	—	ns	—
$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	t_{CEHREL}	t_{CRP}	10	—	10	—	10	—	ns	—
$\overline{RAS}$ Hold Time	t_{CELREH}	t_{RSH}	50	—	60	—	75	—	ns	—
Refresh Period	t_{RVRV}	t_{RFSH}	—	4	—	4	—	4	ms	—

(continued)

NOTES:

1. V_{IH} min and V_{IL} max are reference levels for measuring timing of input signals. Transition times are measured between V_{IH} and V_{IL}.
2. An initial pause of 200 μs is required after power-up followed by 8 $\overline{RAS}$ cycles before proper device operation is guaranteed.
3. The transition time specification applies for all input signals. In addition to meeting the transition rate specification, all input signals must transmit between V_{IH} and V_{IL} (or between V_{IL} and V_{IH}) in a monotonic manner.
4. The specifications for t_{RC} (min) and t_{RMW} (min) are used only to indicate cycle time at which proper operation over the full temperature range (0°C ≤ T_A ≤ 70°C) is assured.
5. AC measurements t_T = 5.0 ns.
6. Assumes that $t_{RCD} \leq t_{RCD}$ (max).
7. Measured with a current load equivalent to 2 TTL (−200) μA, +4 mA) loads and 100 pF with the data output trip points set at V_{OH} = 2.0 V and V_{OL} = 0.8 V.
8. Assumes that $t_{RCD} \geq t_{RCD}$ (max).
9. t_{OFF} (max) defines the time at which the output achieves the open circuit condition and is not referenced to output voltage levels.
10. Operation within the t_{RCD} (max) limit ensures that t_{RAC} (max) can be met. t_{RCD} (max) is specified as a reference point only; if t_{RCD} is greater than the specified t_{RCD} (max) limit, then access time is controlled exclusively by t_{CAC}.
11. Either t_{RRH} or t_{RCH} must be satisfied for a read cycle.
12. These parameters are referenced to $\overline{CAS}$ leading edge in random write cycles and to $\overline{WRITE}$ leading edge in delayed write or read-modify-write cycles.

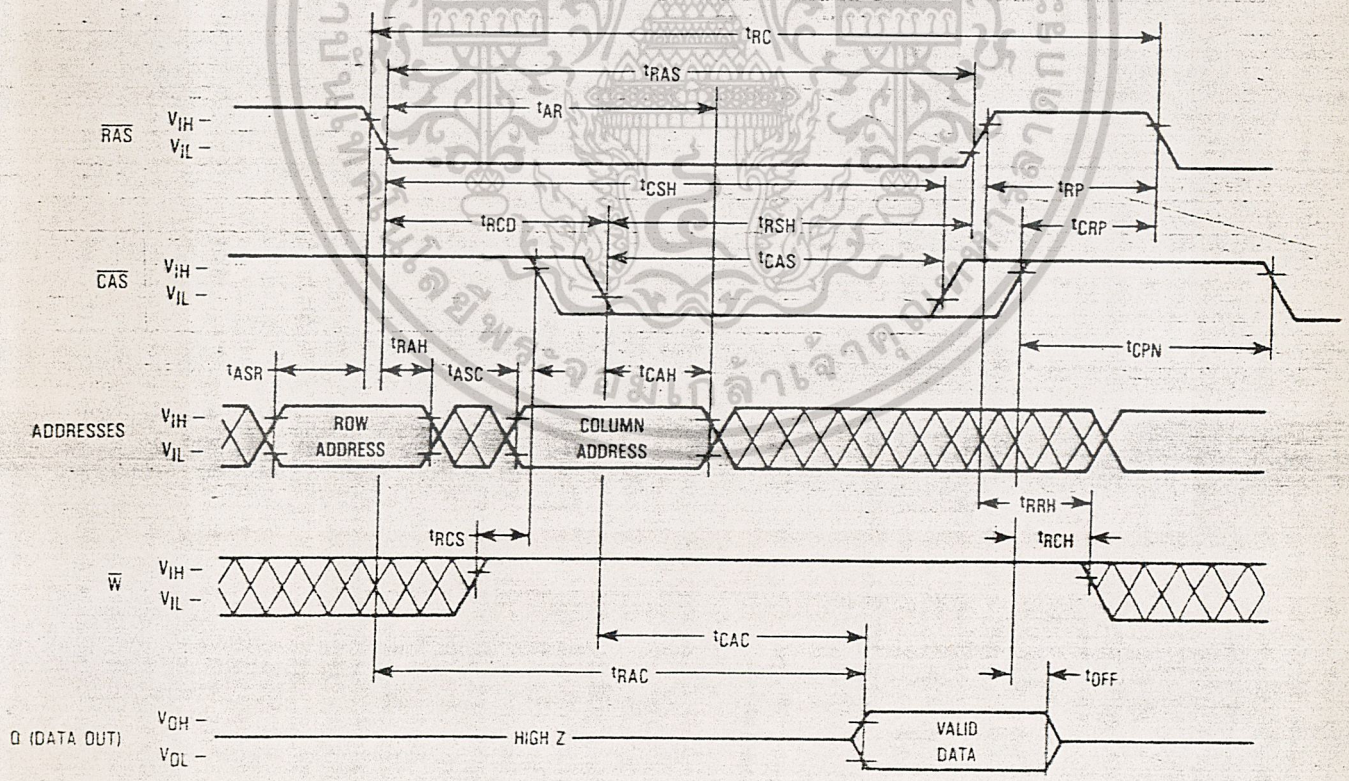
READ, WRITE, AND READ-MODIFY-WRITE CYCLES (Continued)

Parameter	Symbol		MCM6256B-10		MCM6256B-12		MCM6256B-15		Unit	Notes
	Standard	Alternate	Min	Max	Min	Max	Min	Max		
Write Command Setup Time	t_{WLCEL}	t_{WCS}	0	–	0	–	0	–	ns	13
$\overline{CAS}$ to Write Delay	t_{CELWL}	t_{CWD}	30	–	40	–	50	–	ns	13
$\overline{RAS}$ to Write Delay	t_{RELWL}	t_{RWD}	80	–	100	–	125	–	ns	13
$\overline{CAS}$ Hold Time	t_{RELCEH}	t_{CSH}	100	–	120	–	150	–	ns	–
$\overline{CAS}$ Precharge Time	t_{CEHCEL}	t_{CPN}	15	–	20	–	25	–	ns	–
$\overline{CAS}$ Precharge Time (Page Mode Cycle Only)	t_{CEHCEL}	t_{CP}	40	–	50	–	60	–	ns	–
Page Mode Cycle Time	t_{CELCEL}	t_{PC}	100	–	120	–	145	–	ns	–
Page Mode Read-Write Cycle Time	t_{CELCEL}	t_{PRWC}	110	–	140	–	170	–	ns	–
Page Mode Read-Modify-Write Cycle Time	t_{CELCEL}	t_{PRMW}	130	–	160	–	195	–	ns	–
$\overline{CAS}$ Hold Time for $\overline{CAS}$ Before $\overline{RAS}$ Refresh	t_{RELCEH}	t_{CHR}	30	–	30	–	30	–	ns	–
$\overline{CAS}$ Setup Time for $\overline{CAS}$ Before $\overline{RAS}$ Refresh	t_{RELCEL}	t_{CSR}	10	–	10	–	10	–	ns	–
$\overline{CAS}$ Precharge to $\overline{CAS}$ Active Time	t_{REHCEL}	t_{RPC}	0	–	0	–	0	–	ns	–
$\overline{CAS}$ Precharge Time for $\overline{CAS}$ Before $\overline{RAS}$ Counter Test	t_{CEHCEL}	t_{CPT}	40	–	50	–	60	–	ns	–

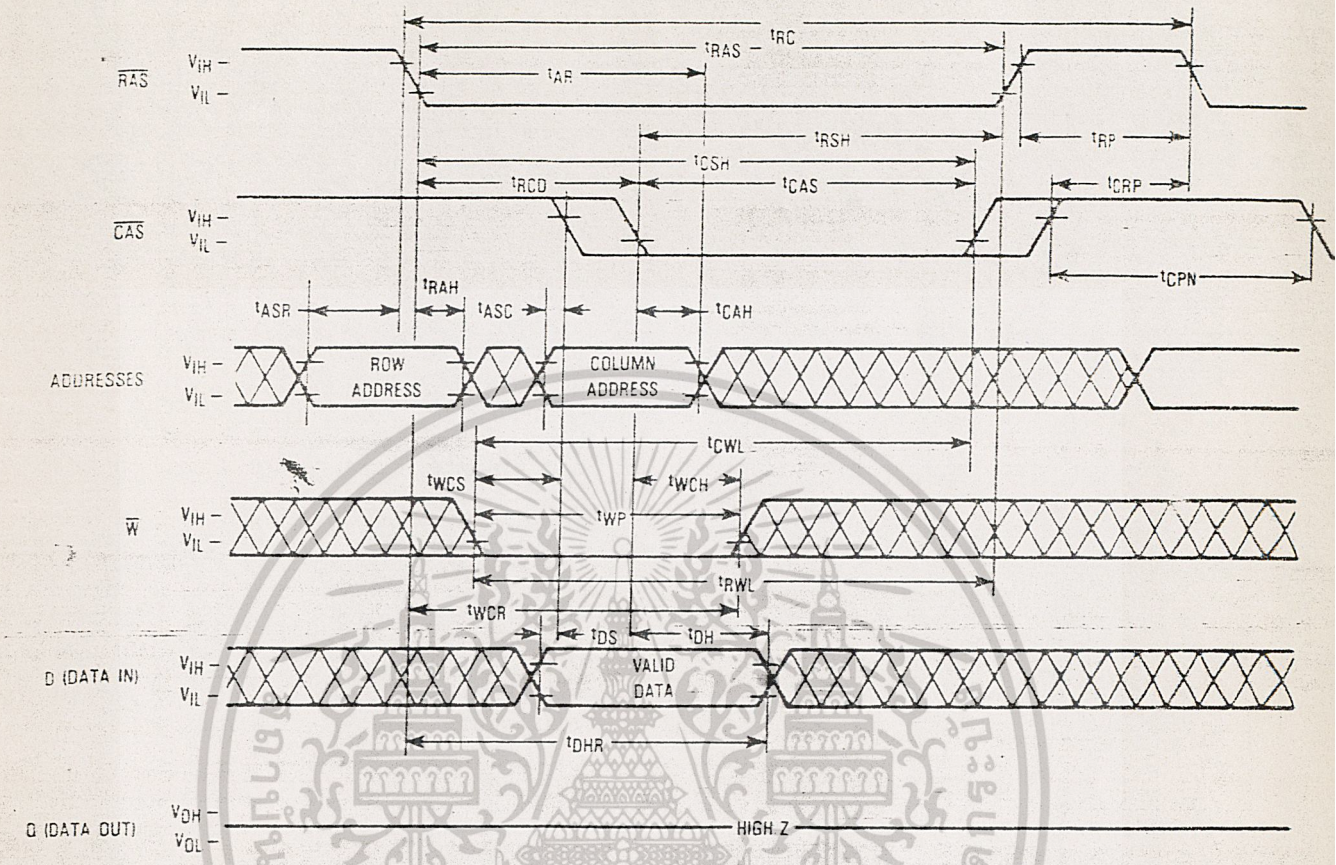
NOTES:

13. t_{WCS}, t_{CWD}, and t_{RWD} are not restrictive operating parameters. They are included in the data sheet as electrical characteristics only; if $t_{WCS} \geq t_{WCS}$ (min), the cycle is an early write cycle and the data out pin will remain open circuit (high impedance) throughout the entire cycle; if $t_{CWD} \geq t_{CWD}$ (min) and $t_{RWD} \geq t_{RWD}$ (min), the cycle is read-write cycle and the data out will contain data read from the selected cell; if neither of the above sets of conditions is satisfied, the condition of the data out (at access time) is indeterminate.

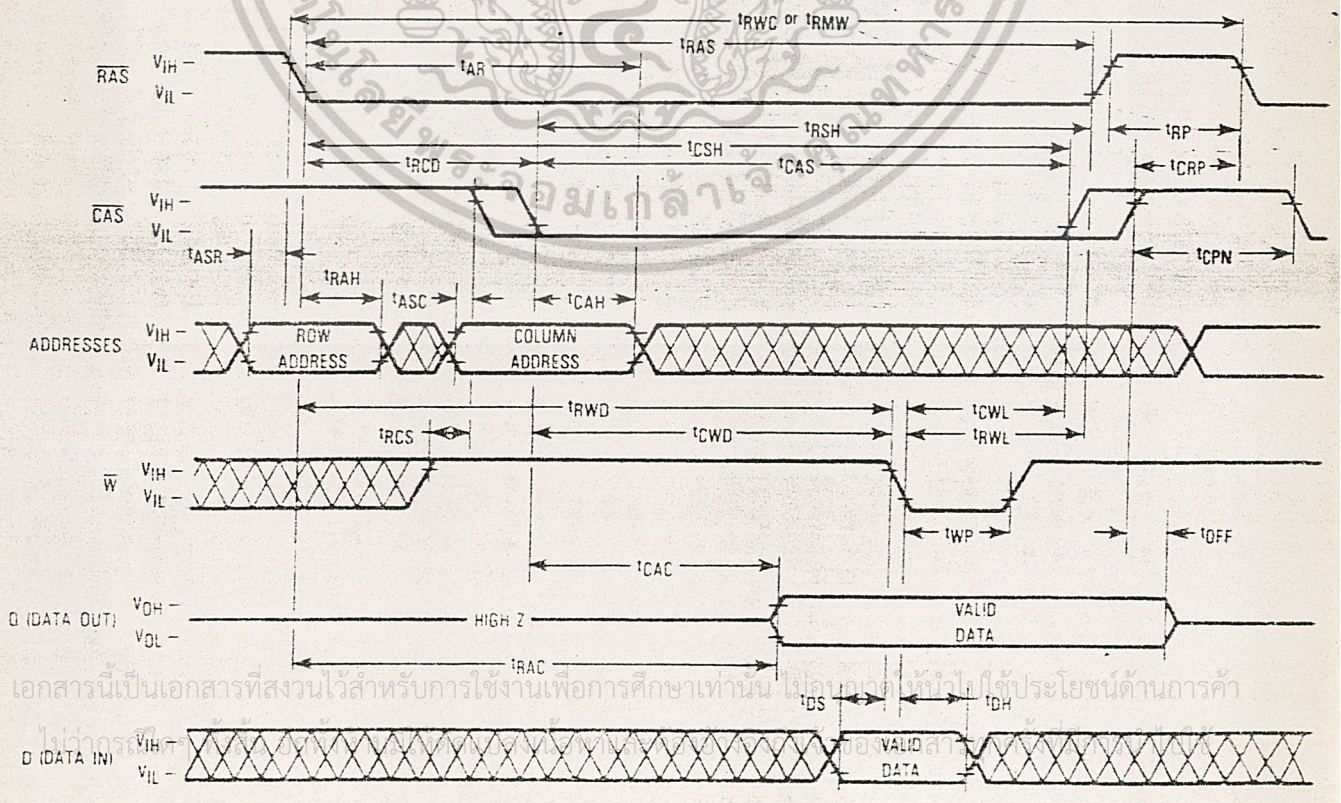
READ CYCLE TIMING



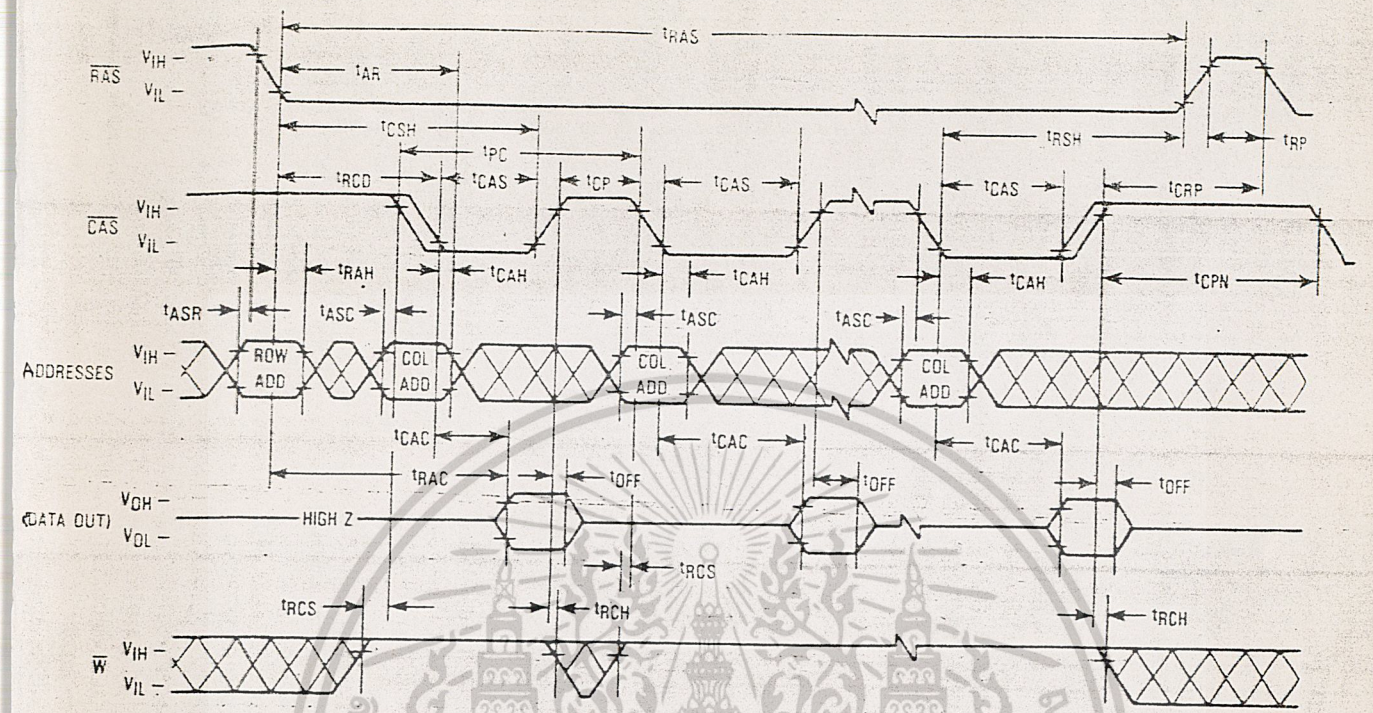
WRITE CYCLE TIMING



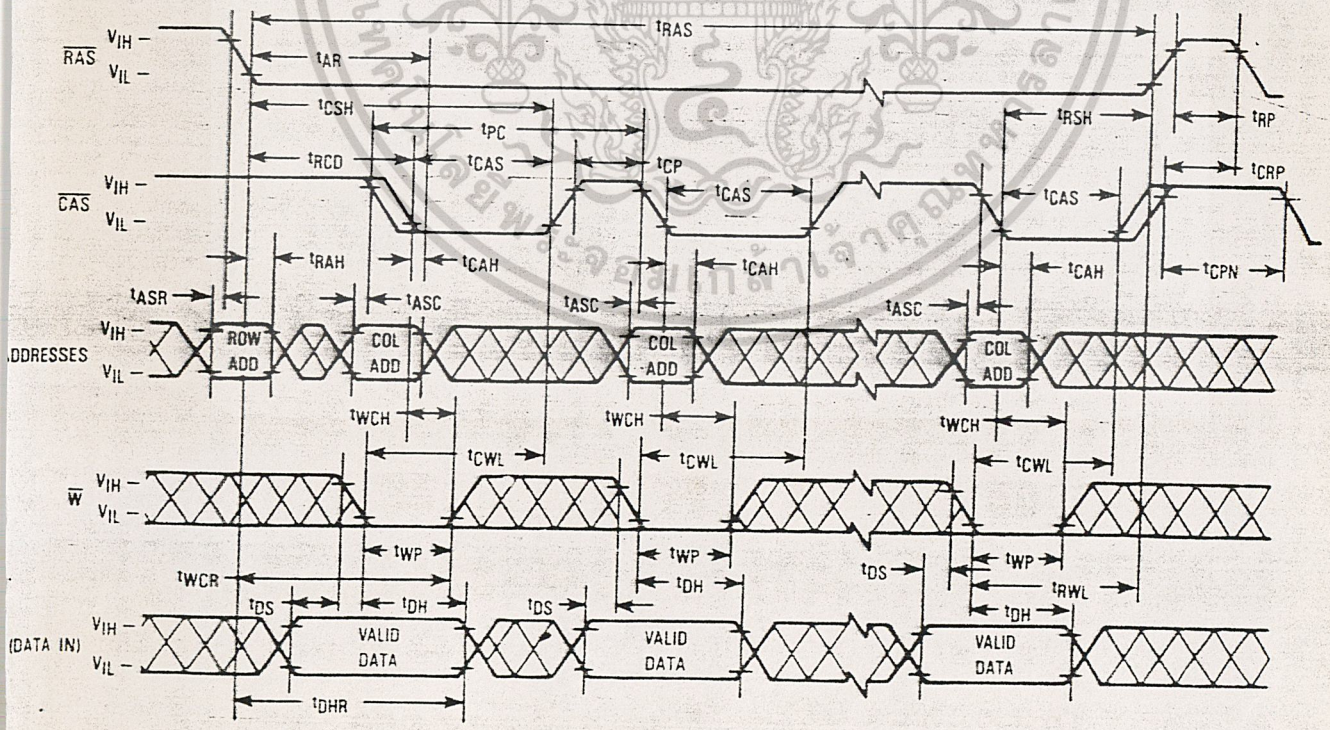
READ-WRITE/READ-MODIFY-WRITE CYCLE



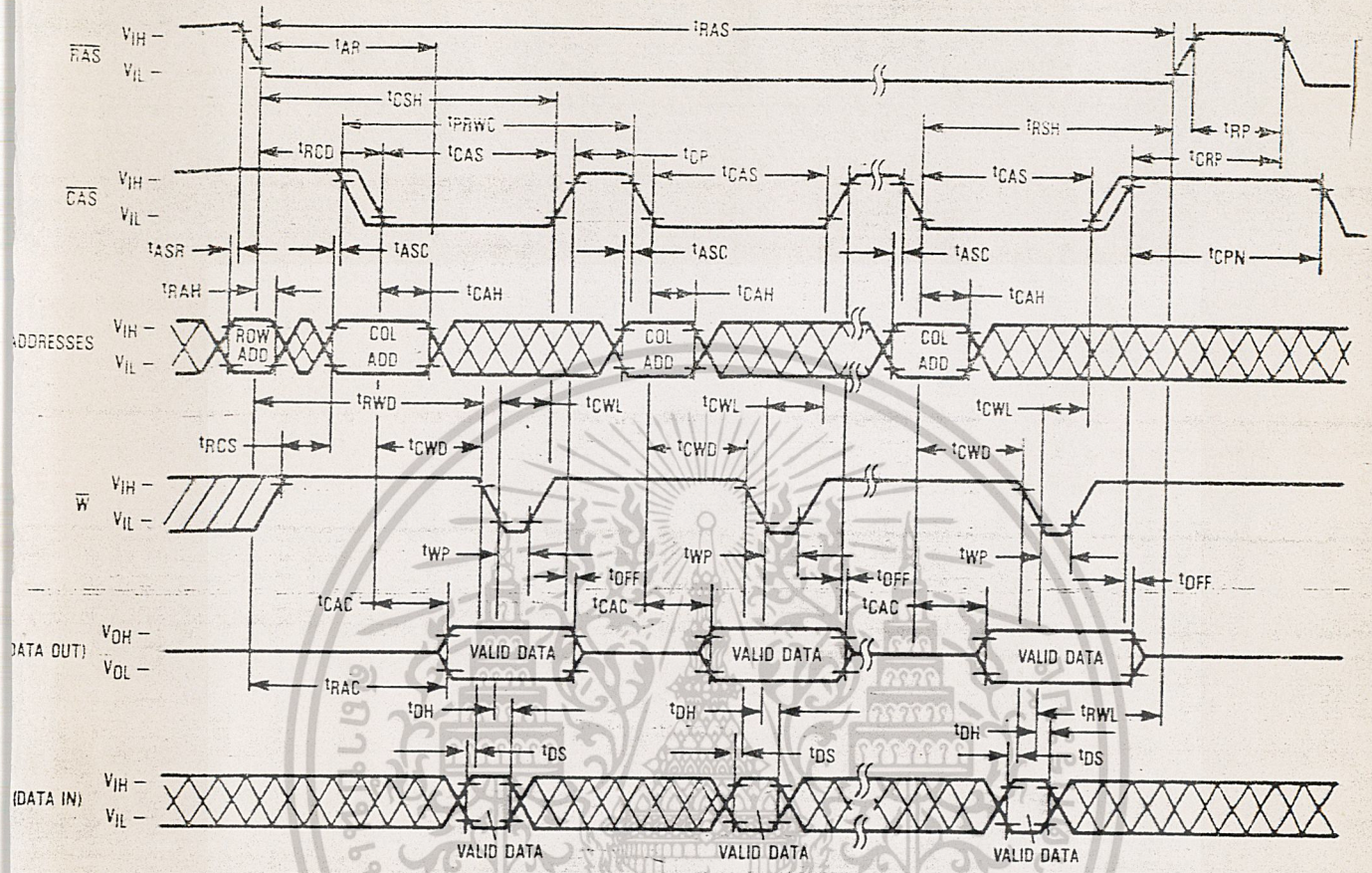
PAGE MODE READ CYCLE



PAGE MODE WRITE CYCLE

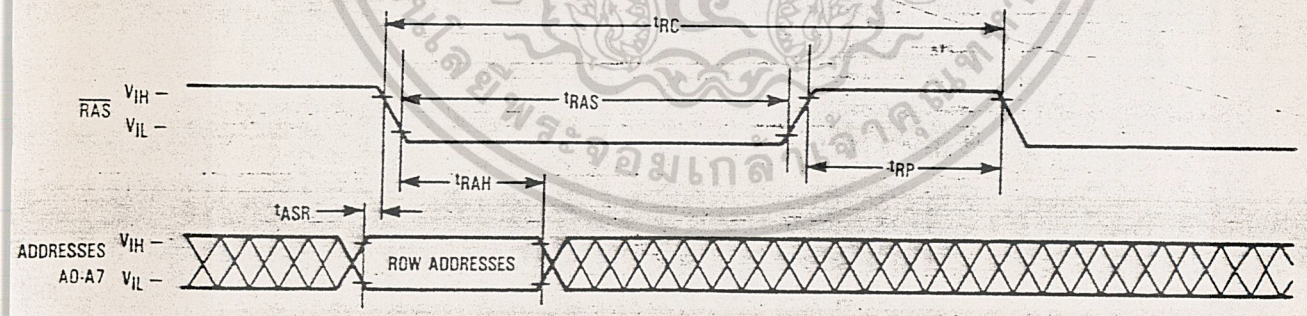


PAGE MODE READ-WRITE/READ-MODIFY-WRITE CYCLE



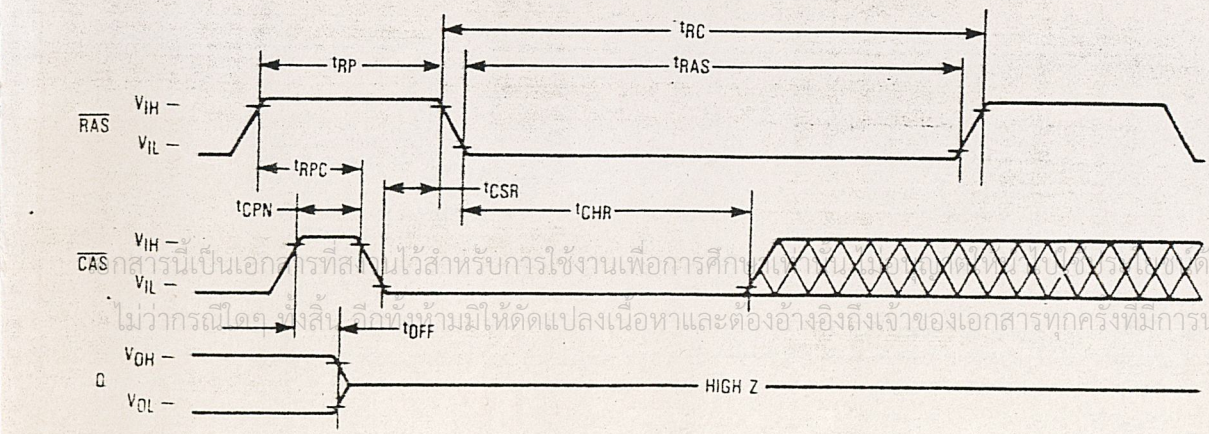
RAS-ONLY REFRESH CYCLE

(D, W, and A8 are Don't Care, CAS is High)

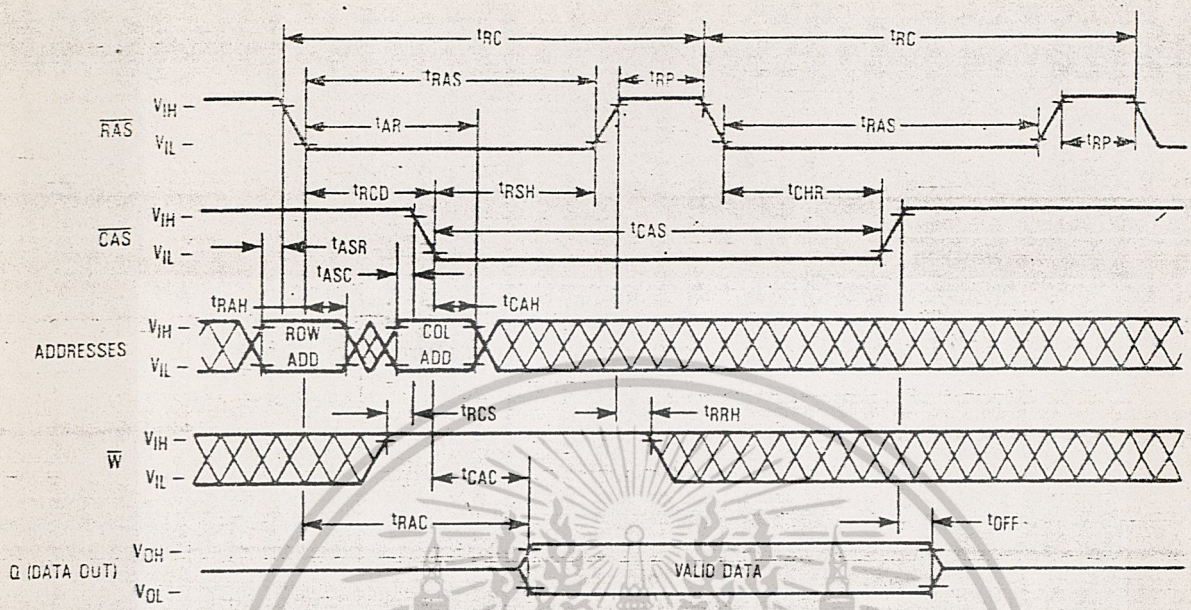


CAS-BEFORE RAS REFRESH CYCLE

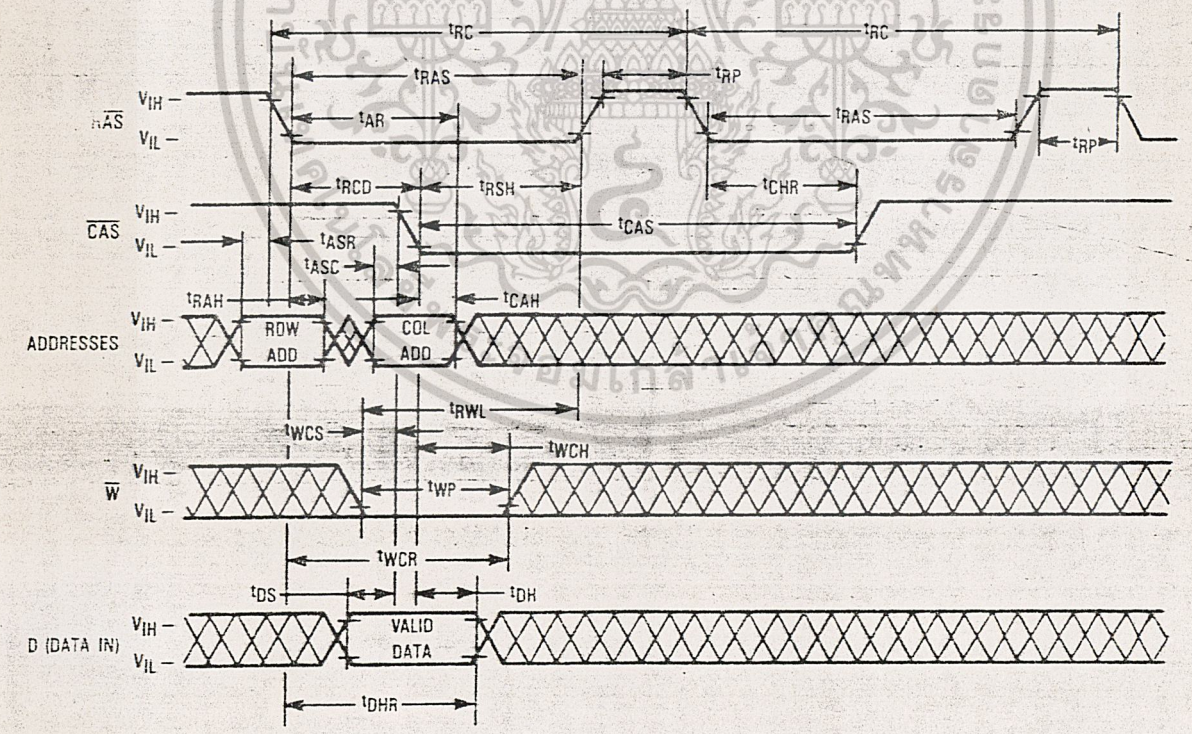
(W, D, and A0-A8 are Don't Care)



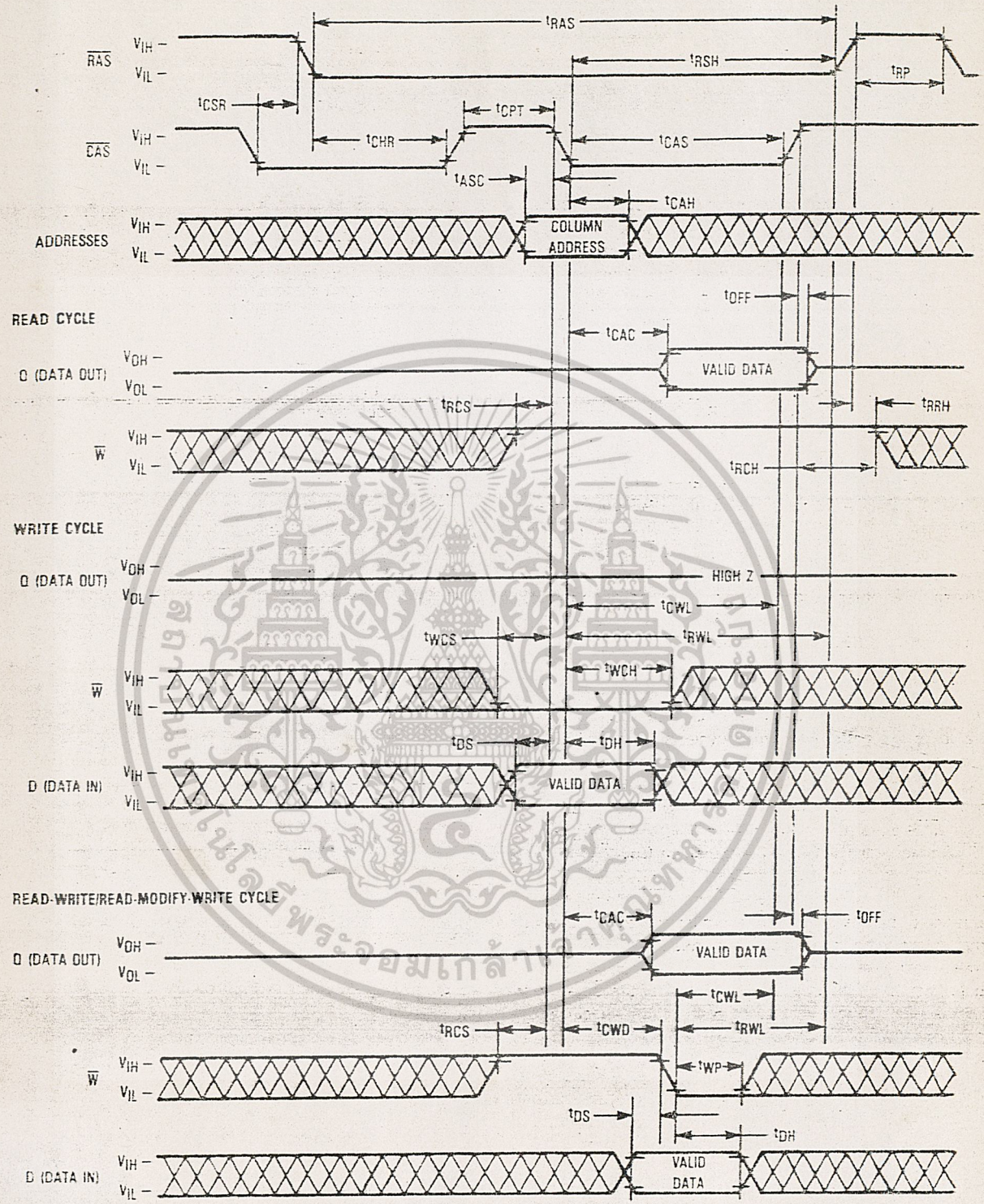
HIDDEN REFRESH CYCLE (READ)



HIDDEN REFRESH CYCLE (WRITE)



$\overline{CAS}$ BEFORE $\overline{RAS}$ REFRESH COUNTER TEST CYCLE



DEVICE INITIALIZATION

On power-up an initial pause of 200 microseconds is required for the internal substrate generator pump to establish the correct bias voltage. This is to be followed by a minimum of eight active cycles of the row address strobe (clock) to initialize the various dynamic nodes internal to the device. During an extended inactive state of the device (greater than 4 milliseconds with device powered up), the wake up sequence (8 active cycles) will be necessary to assure proper device operation.

ADDRESSING THE RAM

The nine address pins on the device are time multiplexed with two separate 9-bit address fields that are strobed at the beginning of the memory cycle by two clocks (active negative) called the row address strobe ($\overline{RAS}$) and the column address strobe ($\overline{CAS}$). A total of eighteen address bits will decode one of the 262,144 cell locations in the device. The column address strobe follows the row address strobe by a specified minimum and maximum time called "t_{RCD}," which is the row to column strobe delay. This time interval is also referred to as the multiplex window which gives flexibility to a system designer to set up his external addresses into the RAM. These conditions have to be met for normal read or write cycles. This initial portion of the cycle accomplishes the normal addressing of the device. There are, however, two other variations in addressing the 256K RAM, one is called the $\overline{RAS}$ only refresh cycle (described later) where an 8-bit row address field is presented on the input pins and latched by the $\overline{RAS}$ clock. The most significant bit on Row Address A8 (pin 1) is not required for refresh. The other variation, which is called page mode, allows the user to column access the 512 bits within a selected row. (See PAGE-MODE CYCLES section.)

READ CYCLE

A read cycle is referred to as a normal read cycle to differentiate it from a page mode read cycle, a read-while-write cycle, and read-modify-write cycle which are covered in a later section.

The memory read cycle begins with the row addresses valid and the $\overline{RAS}$ clock transitioning from V_{IH} to the V_{IL} level. The $\overline{CAS}$ clock must also make a transition from V_{IH} to the V_{IL} level at the specified t_{RCD} timing limits when the column addresses are latched. Both the $\overline{RAS}$ and $\overline{CAS}$ clocks trigger a sequence of events which are controlled by several delayed internal clocks. Also, these clocks are linked in such a manner that the access time of the device is independent of the address multiplex window. The only stipulation is that the $\overline{CAS}$ clock must be active before or at the t_{RCD} maximum specification for an access (data valid) from the $\overline{RAS}$ clock edge to be guaranteed (t_{RAC}). If the t_{RCD} maximum condition is not met, the access (t_{CAC}) from the $\overline{CAS}$ clock active transition will determine read access time. The external $\overline{CAS}$ signal is ignored until an internal $\overline{RAS}$ signal is available. This gating feature on the $\overline{CAS}$ clock will allow the external $\overline{CAS}$ signal to become active as soon as the row address hold time (t_{RAH}) specification has been met and defines the t_{RCD} minimum specification. The time difference between t_{RCD} minimum and t_{RCD} maximum can be used to absorb skew delays in switching the address bus from row to column addresses and in generating the $\overline{CAS}$ clock.

Once the clocks have become active, they must stay active for the minimum (t_{RAS}) period for the $\overline{RAS}$ clock and the minimum (t_{CAS}) period for the $\overline{CAS}$ clock. The $\overline{RAS}$ clock must stay inactive for the minimum (t_{RP}) time. The former for the completion of the cycle in progress, and the latter for the device internal circuitry to be precharged for the next active cycle.

Data out is not latched and is valid as long as the $\overline{CAS}$ clock is active; the output will switch to the three-state mode when the $\overline{CAS}$ clock goes inactive. To perform a read cycle, the write ($\overline{W}$) input must be held at the V_{IH} level from the time the $\overline{CAS}$ clock makes its active transition (t_{RCS}) to the time when it transitions into the inactive (t_{RCH}) mode.

WRITE CYCLE

A write cycle is similar to a read cycle except that the Write ($\overline{W}$) clock must go active (V_{IL} level) at or before the $\overline{CAS}$ clock goes active at a minimum t_{WCS} time. If the above condition is met, then the cycle in progress is referred to as an early write cycle. In an early write cycle, the write clock and the data in are referenced to the active transition of the $\overline{CAS}$ clock edge. There are two important parameters with respect to the write cycle: the column strobe to write lead time (t_{CWL}) and the row strobe to write lead time (t_{RWL}). These define the minimum time that $\overline{RAS}$ and $\overline{CAS}$ clocks need to be active after the write operation has started ($\overline{W}$ clock at V_{IL} level).

It is also possible to perform a late write cycle. For this cycle the write clock is activated after the $\overline{CAS}$ goes low which is beyond t_{WCS} minimum time. Thus the parameters t_{CWL} and t_{RWL} must be satisfied before terminating this cycle. The difference between an early write cycle and a late write cycle is that in a late write cycle the write ($\overline{W}$) clock can occur much later in time with respect to the active transition of the $\overline{CAS}$ clock. This time could be as long as 10 microseconds — $[t_{RWL} + t_{RP} + 2t_T]$.

At the start of an early write cycle, the data out is in a high impedance condition and remains inactive throughout the cycle. The data out remains three-state because the active transition of the write ($\overline{W}$) clock prevents the $\overline{CAS}$ clock from enabling the data-out buffers. The three-state condition (high impedance) of the data out pin during a write cycle can be effectively utilized in systems that have a common input/output bus. The only stipulation is that the system use only early write mode operations for all write cycles to avoid bus contention.

READ-MODIFY-WRITE AND READ-WHILE-WRITE CYCLES

As the name implies, both a read and a write cycle are accomplished at a selected bit during a single access. The read-modify-write cycle is similar to the late write cycle discussed above.

For the read-modify-write cycle a normal read cycle is initiated with the write ($\overline{W}$) clock at the V_{IH} level until the read data occurs at the device access time (t_{RAC}). At this time the write ($\overline{W}$) clock is asserted. The data in is setup and held with respect to the active edge of the write clock. The cycle described assumes a zero modify time between read and write.

Another variation of the read-modify-write cycle is the read-while-write cycle. For this cycle, t_{CWD} plays an important role. A read-while-write cycle starts as a normal read cycle with the write ($\overline{W}$) clock being asserted at minimum t_{CWD} time, depending upon the application. This results in starting a write operation to the selected cell even before data out

occurs. The minimum specification on t_{CWD} assures that data out does occur. In this case, the data in is set up with respect to write ($\overline{W}$) clock active edge.

PAGE-MODE CYCLES

Page mode operation allows fast successive data operations at the 512 column locations. Page access (t_{CAC}) is typically half the regular $\overline{RAS}$ clock access (t_{RAC}) on the Motorola 256K dynamic RAM. Page mode operation consists of holding the $\overline{RAS}$ clock active while cycling the $\overline{CAS}$ clock to access the column locations determined by the 9-bit column address field.

The page cycle is always initiated with a row address being provided and latched by the $\overline{RAS}$ clock, followed by the column address and $\overline{CAS}$ clock. From the timing illustrated, the initial cycle is a normal read or write cycle, that has been previously described, followed by the short $\overline{CAS}$ cycles (t_{PC}). The $\overline{CAS}$ cycle time (t_{PC}) consists of the $\overline{CAS}$ clock active time (t_{CAS}), and $\overline{CAS}$ clock precharge time (t_{CP}) and two transitions. In addition to read and write cycles, a read-modify-write cycle can also be performed in a page mode operation. For a read-modify-write or read-while-write type cycle, the conditions normal to that mode of operation will apply in the page mode also. In practice, any combination of read, write and read-modify-write cycles can be performed to suit a particular application.

REFRESH CYCLES

The dynamic RAM design is based on capacitor charge storage for each bit in the array. This charge will tend to degrade with time and temperature. Therefore, to retain the correct information, the bits need to be refreshed at least once every 4 milliseconds. This is accomplished by sequentially cycling through the 256 row address locations every 4 milliseconds, (i.e., at least one row every 15.6 microseconds like the 64K dynamic RAM). A normal read or write operation to the RAM will serve to refresh all the bits (1024) associated with the particular rows decoded.

$\overline{RAS}$-Only Refresh

In this refresh method, the system must perform a $\overline{RAS}$-only cycle on 256 row addresses every 4 milliseconds. The row addresses are latched in with the $\overline{RAS}$ clock, and the associated internal row locations are refreshed. As the heading implies, the $\overline{CAS}$ clock is not required and must be in or at a V_{IH} level.

$\overline{CAS}$ Before $\overline{RAS}$ Refresh

This refresh cycle is initiated when $\overline{RAS}$ falls after $\overline{CAS}$ has been low (by t_{CSR}). This activates the internal refresh counter which generates the address to be refreshed. Externally applied addresses are ignored during the automatic refresh cycle. If the output buffer was off before the automatic refresh cycle, the output will stay in the high impedance state. If the output was enabled by $\overline{CAS}$ in the previous cycle, the data out will be maintained during the automatic refresh cycle as long as $\overline{CAS}$ is held active (hidden refresh).

Hidden Refresh

The hidden refresh method allows refresh cycles to be performed while maintaining valid data at the output pin. Hidden refresh is performed by holding $\overline{CAS}$ at V_{IL} and taking $\overline{RAS}$ high and after a specified precharge period (t_{RP}), executing a $\overline{CAS}$ before $\overline{RAS}$ refresh cycle. (See Figure 1.)

$\overline{CAS}$ BEFORE $\overline{RAS}$ REFRESH COUNTER TEST

The internal refresh operation of MCM6256B can be tested by $\overline{CAS}$ before $\overline{RAS}$ refresh counter test. This cycle performs read/write operation taking the internal counter address as row address and the input address as column address.

The test is performed after a minimum of 8 $\overline{CAS}$ before $\overline{RAS}$ cycles as initialization cycles. The test procedure is as follows.

1. Write a "0" into all memory cells.
2. Select any column address and read the "0"s written in step 1. Write a "1" into each cell of the selected column by performing $\overline{CAS}$ before $\overline{RAS}$ Refresh Counter Test Read-Write Cycle (see timing diagram). Repeat 256 times.
3. Read the "1"s (use a normal read mode) written in step 2.
4. Select the same column address as step 2, read the "1"s and write a "0" into each cell by performing $\overline{CAS}$ before $\overline{RAS}$ Refresh Counter Test Read-Write Cycle (see timing diagram). Repeat 256 times.
5. Read the "0"s (use a normal read mode) written in step 4.
6. Repeat steps 1 through 5 using complement data.

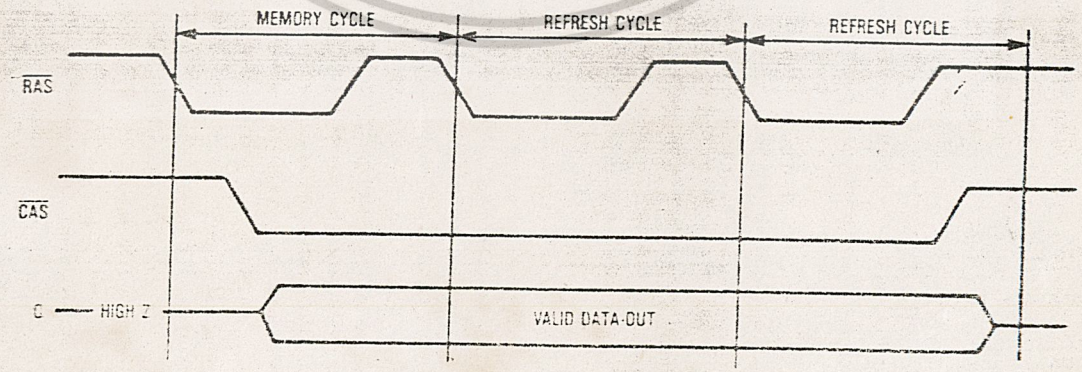


Figure 1. Hidden Refresh Cycle