เครื่องบันทึก และตอบรับโทรศัพท์อัตโนมัติ

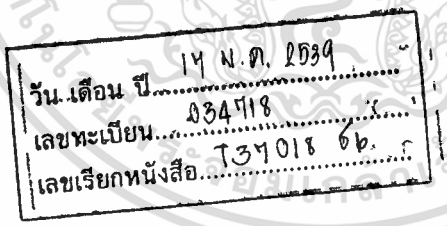
VOICE MAIL EXCHANGE

โดย

นาย ชาตรี อภินันท์กูล

นาย สุทธิชัย ทองผา

นาย สุเทพ เตชะบุญเกียรติ



ปริญญานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเลคทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ ปีการศึกษา 2537

ภาควิชา อิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องบันทึกและตอบรับโทรศัพท์อัตโนมัติ

VOICE MAIL EXCHANGE

ผู้จัดทำ

1. นาย ชาตรี อภินันท์กูล 34102094
2. นาย สุทธิชัย ทองผา 34108431
3. นาย สุเทพ เตชะบุญเกียรติ 34108434

(ดร. กิติพล ชิตสกุล)

อาจารย์ที่ปรึกษา

สารบัญ

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

เครื่องบันทึกและตอบรับโทรศัพท์อัตโนมัติ

VOICE MAIL EXCHANGE

โดย นาย ชาตรี อภินันท์กูล

นาย สุทธิชัย ทองผา

นาย สุเทพ เตชะบุญเกียรติ

อาจารย์ที่ปรึกษา ดร. กิตติพล ชิตสกุล

บทคัดย่อ

โครงงานเครื่องบันทึกและตอบรับโทรศัพท์อัตโนมัตินี้ มาจากแนวความคิดที่จะพัฒนาเครื่องสำหรับนำไปใช้ในการบันทึกข้อความ หรือฝากข้อความ หรือคำพูดของผู้ใช้โทรศัพท์ขณะที่ฝ่ายหนึ่งไม่ว่าง ทำให้ผู้สนทนาโทรศัพท์ไม่จำเป็นต้องว่างพร้อมกัน

โครงงานนี้ ใช้ไมโครโปรเซสเซอร์ Z280 เป็นส่วนควบคุมหลัก โดยจะมีการต่อกับหน่วยความจำไดนามิกแรม และต่อเชื่อมกับฮาร์ดดิสก์แบบ IDE เป็นหน่วยเก็บข้อมูล ทำให้ไม่ต้องพึ่งพาการใช้งานเครื่องคอมพิวเตอร์ IBM PC แต่อย่างใด

เนื่องจากฮาร์ดดิสก์ที่ใช้งานมีขนาด 540 เมกกะไบท์ และการสุ่มสัญญาณเสียงใช้ความถี่ 8 กิโลเฮิรตซ์ ดังนั้นจะเก็บข้อมูลได้ทั้งหมดประมาณ 18 ชั่วโมง และจะใช้งานกับโทรศัพท์ในขณะใดขณะหนึ่งได้ 16 คู่สาย

ABSTRACT

Voice mail exchange is developed to record voice or data for the person who has many job in many place so that he cannot talk with other in that time. If he uses voice mail exchange, he can listen the voice record when he is free from the job. So he can communicate with the other in the other time.

This project "voice mail exchange" uses Z280 microprocessor to control the system and interface with dynamic ram and IDE harddisk to store the data. So this project doesn't need IBM PC to control anything.

Because of 540 Mbyte harddisk's capacity, this project can store voice data about 18 hours and use 16 telephone lines in the same time.

บทนำ

ความเป็นมา

ปัจจุบันความก้าวหน้าของเทคโนโลยีเป็นไปอย่างรวดเร็ว ทั้งเทคโนโลยีการสื่อสาร และคอมพิวเตอร์ ราคาของอุปกรณ์สื่อสารหรือคอมพิวเตอร์ก็ลดลงมาก ทำให้คนมีความสามารถหาซื้ออุปกรณ์เหล่านี้ได้มากขึ้น ทำให้มีการใช้เทคโนโลยีเหล่านี้เพิ่มขึ้นอย่างมาก และรูปแบบหลากหลายชนิดมากขึ้น

โทรศัพท์ นับเป็นอุปกรณ์สื่อสารพื้นฐานอย่างหนึ่ง ที่ปัจจุบันมีการใช้งานอย่างแพร่หลาย โทรศัพท์ให้ความสะดวกสบายในการติดต่อสื่อสารระหว่างบุคคลโดยไม่ต้องเดินทางไปหากัน แต่โทรศัพท์ก็มีจุดอ่อนที่เป็นการติดต่อที่คู่สนทนาทั้งสองฝ่ายต้องพูดโต้ตอบกันในเวลาเดียวกัน ดังนั้นถ้ามีคู่สนทนาฝ่ายหนึ่งไม่ว่าง การติดต่อสื่อสารก็ไม่อาจเกิดขึ้นได้

ดังนั้นโครงงานนี้ คือเครื่องตอบรับและบันทึกโทรศัพท์อัตโนมัติ ซึ่งเป็นโครงงานที่บันทึกเสียงจากโทรศัพท์ และแปลงสัญญาณเสียงเป็นดิจิตอลเพื่อจะนำไปจัดเก็บ เมื่อต้องการจะฟังเสียงก็จะนำข้อมูลที่จัดเก็บไว้มาใช้จะสร้างขึ้นมาเพื่อแก้ปัญหานี้ โดยจะใช้ไมโครโปรเซสเซอร์ Z280 เป็นตัวควบคุมการทำงาน

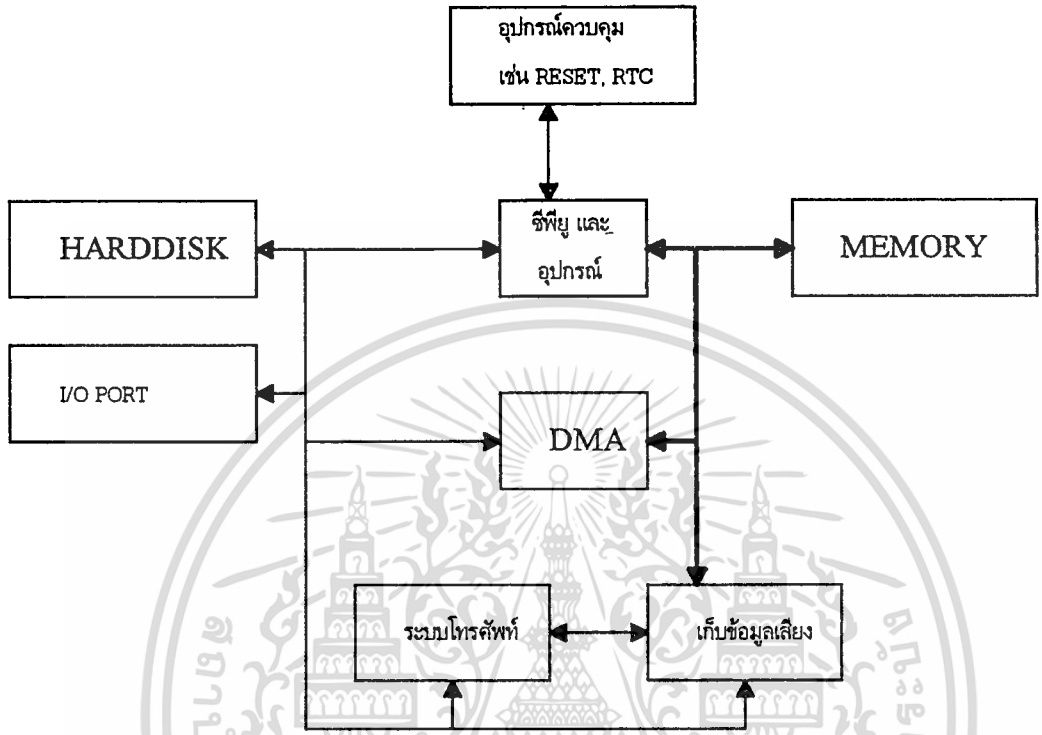
วัตถุประสงค์

- ☐ ศึกษาการใช้งานไมโครโปรเซสเซอร์ Z280
- ☐ ศึกษาการต่อเชื่อมและใช้งานฮาร์ดดิสก์แบบ IDE
- ☐ ศึกษาการต่อเชื่อมอุปกรณ์โทรศัพท์
- ☐ ศึกษาการเก็บข้อมูลเสียงเพื่อนำมาใช้ในภายหลัง

รายละเอียดโดยย่อ

1. ข้อกำหนดเบื้องต้นสำหรับคุณสมบัติทั้งหมดของโครงงานนี้ ได้แก่
2. สามารถทำการรับสัญญาณกริ่งโทรศัพท์ได้โดยอัตโนมัติ
3. สามารถรับข้อมูลจากโทรศัพท์โดยการกดจากแป้นตัวเลขบนเครื่องโทรศัพท์ แล้วนำข้อมูลที่รับมาได้ มาทำการค้นหาข้อมูลที่ต้องการได้อย่างอัตโนมัติ
4. สามารถนำข้อมูลที่ค้นหาได้แล้ว มาเปลี่ยนเป็นสัญญาณเสียงเพื่อส่งออกผ่านทางคู่สายโทรศัพท์ไปยังผู้ที่ต้องการทราบข้อมูลได้
5. สามารถทำการบันทึกสัญญาณเสียงและทำการโหลดสัญญาณเสียงออกมาได้
6. สามารถทำงานได้ตั้งแต่ 1 คู่สาย ถึงสูงสุด 16 คู่สาย โดยสามารถทำงานพร้อมกันได้
7. บันทึกและอ่านข้อมูลเสียงจากฮาร์ดดิสก์ได้โดยตรง

โครงงานตอบรับและบันทึกโทรศัพท์อัตโนมัติ จะบันทึกเสียงจากโทรศัพท์ และแปลงสัญญาณเสียงเป็นดิจิตอลเพื่อจะนำไปจัดเก็บ เมื่อต้องการจะฟังเสียงก็จะนำข้อมูลที่จัดเก็บไว้มาใช้ ซึ่งจะมีโครงสร้างการทำงานดังนี้



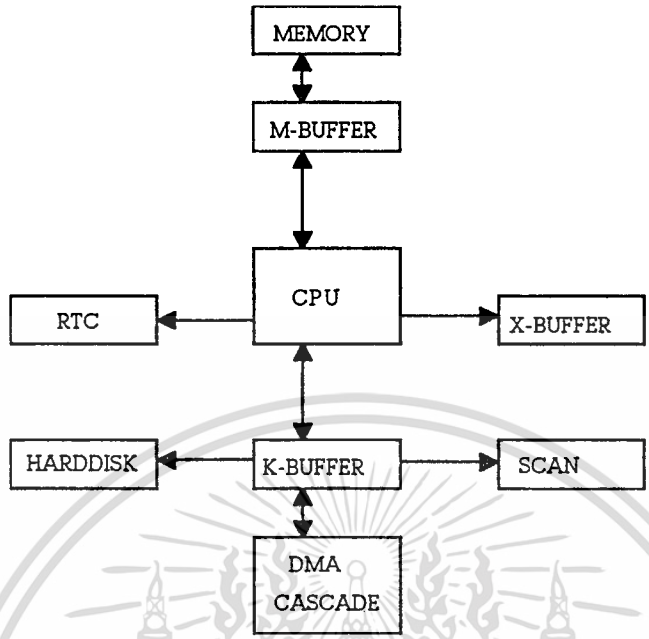
โครงสร้างการทำงาน

แต่เนื่องจากอุปกรณ์ที่ใช้จะมีเป็นจำนวนมากเลยต้องแยกการต่อออกเป็นหลายๆ ส่วน คือ

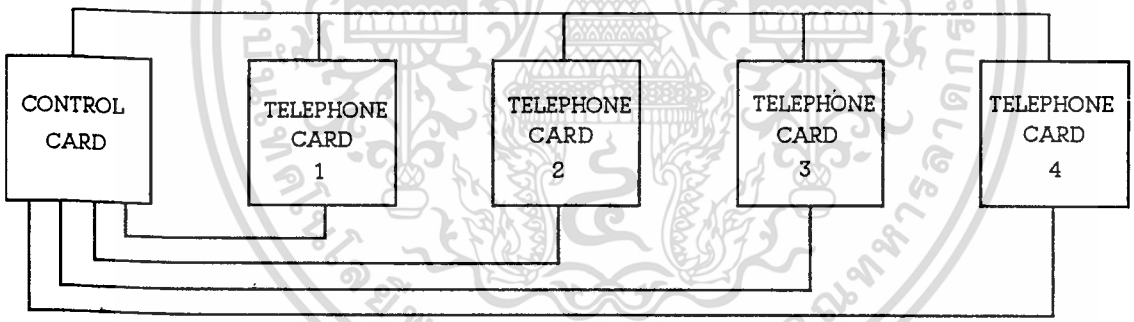
1. ส่วน Control Card ประกอบด้วย ซีพียู RAM บัฟเฟอร์ต่างๆ อุปกรณ์สนับสนุนการต่อ DMA แบบ CASCADE และอุปกรณ์ควบคุมต่างๆ

2. ส่วน Telephone Card ประกอบด้วยส่วน ระบบโทรศัพท์ ส่วนเก็บข้อมูลเสียง

ส่วน Control Card จะแสดงได้ดังรูป



จากข้อกำหนดข้างต้น จะทำให้เกิดการเชื่อมต่อกัน ได้ดังนี้



รูปที่ 1 Block Diagram ของโครงงานเครื่องตอบรับโทรศัพท์

วงจรทั้งหมดจะทำงานโดยใช้ไมโครโปรเซสเซอร์ Z280 ซึ่งเป็นไมโครโปรเซสเซอร์ 16 bit เบอร์ใหม่ของ ZILOG ทำให้ไม่ต้องพึ่งเครื่อง IBM PC แต่อย่างใด ซึ่งข้อกำหนดเบื้องต้นสำหรับวงจรทั้งหมด สามารถอธิบายอย่างคร่าวๆ ได้ดังนี้

1. ไมโครโปรเซสเซอร์ Z280 เป็นอุปกรณ์หลักที่ใช้ติดต่อกับอุปกรณ์อินพุท/เอาท์พุต หน่วยความจำ วงจรส่วนอินเตอร์เฟสกับโทรศัพท์ต่างๆ

2. วงจรเรียลไทม์คล็อก (RTC) เป็นฐานเวลาของระบบ

3. ระบบหน่วยความจำ เนื่องจากโครงงานนี้ไม่ใช้ IBM PC จึงมีหน่วยความจำที่เป็น Dynamic RAM เอง

4. ฮาร์ดดิสต์ อินเตอร์เฟส เป็นอินเตอร์เฟสกับฮาร์ดดิสต์แบบ IDE ซึ่งมีความนิยมและราคาถูกที่สุด

5. วงจรในการ SCAN ซึ่งเป็นการควบคุมและติดต่อกับวงจรส่วนอินเตอร์เฟสกับโทรศัพท์

6. วงจรสำหรับการติดต่อกับโทรศัพท์ ทำหน้าที่รับสัญญาณกริ่งโทรศัพท์ รับรหัสตัวเลขจากโทรศัพท์

7. วงจร DMA Controller ใช้ในการรับส่งข้อมูลระหว่างวงจรโทรศัพท์กับหน่วยความจำโดยตรง

บทที่ 1

Z280 Microprocessor Unit

คุณสมบัติ

- ☐ ออกแบบโดยใช้เทคโนโลยี CMOS ทำให้ใช้พลังงานต่ำ
- ☐ ปรับปรุงกลุ่มคำสั่งของ Z80 โดยยังคงรักษาความเข้ากันได้กับ object-code ของไมโครโปรเซสเซอร์ Z80
- ☐ เป็น CPU 16 bit มีระบบบัสแบบ pipe line 3 ขั้น และแบ่งแยก user และ system mode
- ☐ สามารถต่อโปรเซสเซอร์ร่วม และต่อโปรเซสเซอร์หลายๆ ตัวได้
- ☐ มีส่วนจัดการหน่วยความจำ (MMU) อยู่ภายในทำให้สามารถอ้างแอดเดรสได้ถึง 16M byte
- ☐ มีหน่วยความจำแคชที่สามารถเก็บคำสั่งและข้อมูลได้ 256 ไบท์ทำให้สามารถโหลดด้วยความเร็วสูงได้ (burst load)
- ☐ มี bus ที่มีประสิทธิภาพสูงคือ Z-BUS ซึ่งมีขนาด 16 บิท หรือ บัสขนาด 8 bit ที่เหมือนกับ Z80
- ☐ มี counter/timer ขนาด 16 บิทอยู่ 3 ช่อง
- ☐ มี DMA อยู่ 4 ช่อง
- ☐ มี full-duplex UART
- ☐ มี Refresh controller สำหรับ dynamic RAM
- ☐ มีวงจร Oscillator อยู่ภายใน
- ☐ มีความถี่สัญญาณนาฬิกา 20-25 Mhz (แต่ทำงานที่ความถี่หารสอง)

1.1 ลักษณะทั่วไป

Z280 เป็นไมโครโปรเซสเซอร์ขนาด 16 บิท ตัวใหม่ของบริษัท ZILOG ซึ่งได้รับการออกแบบให้มีความสามารถ และประสิทธิภาพสูง มีการส่งผ่านข้อมูลได้เร็วและมีประสิทธิภาพ อ้างอิงหน่วยความจำได้มากขึ้น ในขณะที่ยังเข้ากันได้กับชุดคำสั่ง Z80 เดิม Z280 เหมาะแก่การพัฒนาสำหรับผลิตภัณฑ์ที่ใช้ Z80 ในปัจจุบัน และ เหมาะสมที่จะใช้เป็นไมโครโปรเซสเซอร์ที่ใช้งานกับผลิตภัณฑ์ที่ต้องการความสามารถมากๆ ต่อไปในอนาคต

Z280 มีการพัฒนามาจาก CPU Z80 และเพื่อความสมบูรณ์ของระบบ ไมโครโปรเซสเซอร์ Z280 จึงสามารถทำงานได้ทั้งโหมด user และ system ซึ่งทำให้สามารถป้องกันการแทรกแซงระบบจากโปรแกรมของผู้ใช้ได้ system mode ได้รับการสนับสนุนการทำงานจากรีจีสเตอร์ system stack pointer ส่วนรีจีสเตอร์ IX และ IY ถูกปรับปรุงเพิ่มเติมจากเดิมที่ใช้เป็นเพียง index

register ให้ใช้เป็นรีจีสเตอร์ ที่ใช้งานแบบ 16 บิท หรือแบบ 8 บิท 2 ตัวก็ได้ และรีจีสเตอร์ R ที่เดิมใช้เพียงรีเฟรชหน่วยความจำก็สามารถนำมาใช้เก็บข้อมูลได้ในไมโครโปรเซสเซอร์ Z280

ชุดคำสั่งของ CPU Z80 ยังคงรวมอยู่ในไมโครโปรเซสเซอร์ Z280 ทำให้ Z280 ยังคงเข้ากันได้กับรหัสไบนารีของ Z80 Z280 ได้มีการเพิ่มเติมการอ้างแอดเดรสแบบ index mode with full 16 bit displacement, program counter relative with 16 bit displacement, stack pointer relative with 16 bit displacement และ base index mode เพิ่มเติมจาก Z80 เดิม ซึ่งการอ้างแอดเดรสแบบใหม่นี้ได้ผนวกเข้าไปในคำสั่งเดิมของ Z80 ทำให้มีความสามารถและความยืดหยุ่นมากขึ้น คำสั่งที่มีเพิ่มเติมได้แก่การสั่งคูณและหารเลข 8 และ 16 บิทแบบมีเครื่องหมายและไม่มีเครื่องหมาย คำสั่งเกี่ยวกับเครื่องหมายทั้ง 8 บิทและ 16 บิท และคำสั่งทดสอบและกำหนดค่า ซึ่งรองรับการใช้ multiprocessor คำสั่ง 16 บิทที่ถูกเพิ่มขึ้นมายังมี คำสั่ง เปรียบเทียบ เพิ่มหน่วยความจำ ลดหน่วยความจำ คำสั่ง บวกและลบ แบบ 16 bit

ในปัจจุบันระบบไมโครโปรเซสเซอร์ 8 บิท จำนวนมากต้องมีการเพิ่มเติมส่วนที่ทำให้อ้างอิงหน่วยความจำได้มากกว่า 64 K byte แต่ Z280 มี MEMORY MANAGEMENT UNIT (MMU) อยู่ภายในตัวซึ่งสามารถอ้างอิงหน่วยความจำได้ถึง 16 M byte

มีการเพิ่ม I/O Page register ซึ่งมีขนาด 8 บิท ไว้ใช้เลือก page ของ I/O address โดยค่าใน I/O Page Register จะกำหนดขา AD_{16} - AD_{23} และคำสั่ง I/O ยังสามารถควบคุม address ของขา AD_0 - AD_{15} ได้ ดังนั้นจึงสามารถควบคุม I/O address ได้ถึง 24 บิท

Z280 มีหน่วยความจำภายใน 256 ไบท์ หน่วยความจำนี้สามารถใช้เป็นแคชความเร็วสูงหรือ local memory ที่กำหนดตำแหน่ง address คงที่ได้ เมื่อกำหนดเป็นแคชยังสามารถโปรแกรมให้เลือกเก็บเฉพาะคำสั่ง เฉพาะข้อมูล หรือทั้งข้อมูลและคำสั่ง หน่วยความจำแคชทำให้โปรแกรมทำงานได้เร็วขึ้นโดยลดการติดต่อกับระบบบัสภายนอกลง การทำงานและการปรับปรุงค่าของหน่วยความจำแคชเป็นไปอย่างอัตโนมัติและง่ายดายสำหรับผู้ใช้ และเมื่อเราใช้เป็น local memory เราก็สามารถโปรแกรม address นั้นได้

คุณสมบัติหลายอย่างซึ่งแต่เดิมเป็นอุปกรณ์สนับสนุนภายนอก ถูกรวมอยู่ภายใน Z280 ทำให้ลดจำนวนอุปกรณ์และการเชื่อมต่อกับบัสภายนอกลง Z280 บรรจุวงจรกำเนิดสัญญาณ (oscillator) และวงจร refresh controller ไว้สำหรับ รีเฟรชไดนามิกแรม ถึง 10 bit และได้เตรียมอุปกรณ์สนับสนุนภายในเพื่อช่วยให้การออกแบบระบบยืดหยุ่นขึ้นด้วย โดยได้เพิ่มส่วน Direct Memory Access (DMA) ถึง 4 ช่องเพื่อรองรับการส่งข้อมูลความเร็วสูง DMA แต่ละช่องทำงานโดยใช้ address ต้นทางและปลายทาง 24 บิท และตัวนับแบบ 16 บิท์ แต่ละช่องของ DMA สามารถโปรแกรมให้ทำงานใน single transaction, burst หรือ continuous mode ระบบที่ต้องการวงจรนับและฐานเวลาก็สามารถใช้ 16 bit counter/timer ซึ่งมีอยู่ใน Z280 ถึง 3 ตัว การทำงานของ counter/timer เราสามารถควบคุมจากภายนอกโดยใช้ gate และ trigger input และเราสามารถโปรแกรมให้เป็นแบบ ทริกซ้ำ หรือ ไม่ทริกซ้ำ ก็ได้ UART แบบ full duplex สามารถรับส่งข้อมูลและรูปแบบได้หลากหลาย ซึ่งคุณสมบัติที่ปรากฏทำให้ง่ายในการสื่อสารข้อมูลอนุกรมแบบไม่ประสานจังหวะ (Asynchronous serial communication)

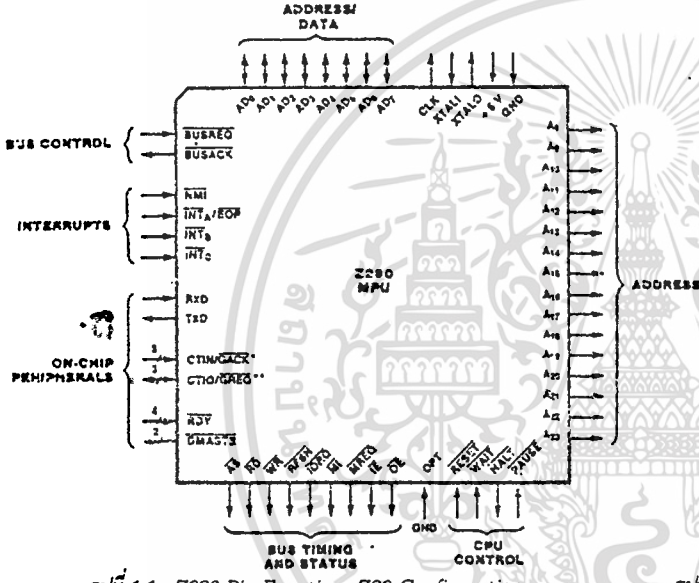
Z280 มีคุณสมบัติในการควบคุมฐานเวลาของบัส โดยอนุญาตให้ผู้ใช้กำหนดฐานเวลาของบัสที่เหมาะสมในแต่ละระบบ เมื่อรีเซตไมโครโปรเซสเซอร์ ผู้ใช้สามารถโปรแกรมให้ฐานเวลาของระบบเป็น หนึ่งในสี่ ครึ่งหนึ่งหรือเท่ากับความเร็วภายใน CPU โดยค่าที่กำหนดจะเป็นความเร็วครึ่งหนึ่ง นอกเหนือจากการควบคุมฐานเวลาของบัสแล้ว ผู้ใช้สามารถที่จะแทรก wait state ในการใช้บัสชนิดต่างๆได้ โดยปราศจากการใช้ฮาร์ดแวร์ภายนอก wait state หนึ่งถึงสามลูกสามารถที่จะแทรกในระหว่างการใช้หน่วยความจำ, I/O และการตอบรับสัญญาณอินเตอร์รัพท์ นอกจากนั้นการแยกหน่วยความจำเป็นแบงค์ wait state สามารถที่จะแทรกได้สำหรับ upper หรือ lower memory ซึ่งจะทำให้ง่ายในการใช้ความเร็วที่แตกต่างของ ROM และ RAM ในระบบเดียวกัน

ลักษณะที่เพิ่มเติมของการอินเตอร์เฟสแบบ 16 บิท คือความสามารถในการสนับสนุนการใช้ dynamic RAM ใน "nibble-mode" (หรือ burst mode) ในโหมดนี้ความเร็วในการอ่านหน่วยความจำจะเป็นสองเท่า burst mode จะทำให้การใช้หน่วยความจำแคชมีประสิทธิภาพมากขึ้น โดยใช้หลักการที่ว่าจะมีความเป็นไปได้อย่างมากที่ข้อมูลที่จะอ่านมีอยู่ในหน่วยความจำแคชอยู่แล้ว

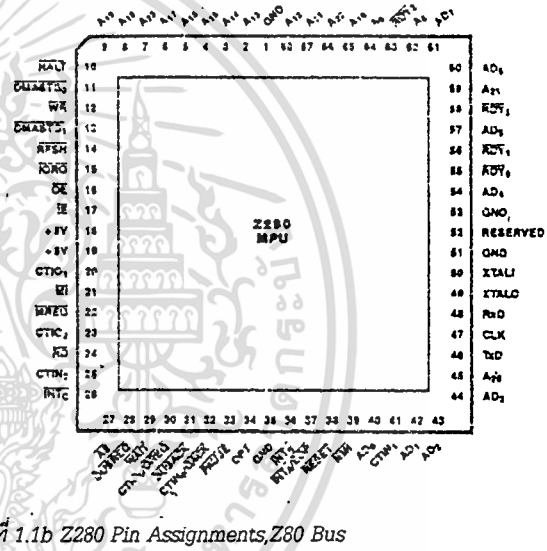
Z280 สนับสนุนการใช้ Zilog's Extended Processor Architecture (EPA) ในหลายวิธี โดย Z280 สามารถที่จะดักจับคำสั่งของ Extended Processor Unit (EPU) เพื่อที่จะใช้ซอฟท์แวร์จำลองการทำงาน EPU ในการอินเตอร์เฟสแบบ 16 บิท Z280 สามารถต่อได้โดยตรงกับ EPU

ระบบ Multiprocessor เป็นสถาปัตยกรรมที่ถูกออกแบบให้ใช้ในระบบของ Z280 ได้ เมื่อทำงานใน multiprocessor mode Local Address register จะถูกใช้ในการแยกระหว่าง local และ global memory การเข้าถึงข้อมูลแบบ global จะถูกควบคุมผ่านทางโปรโตคอล global request และ global acknowledge

ลักษณะการจัดขาของไมโครโปรเซสเซอร์ Z280 แสดงได้ในรูปที่ 1.1 และ1.2 และมี block diagram แสดงในรูปที่ 1.3

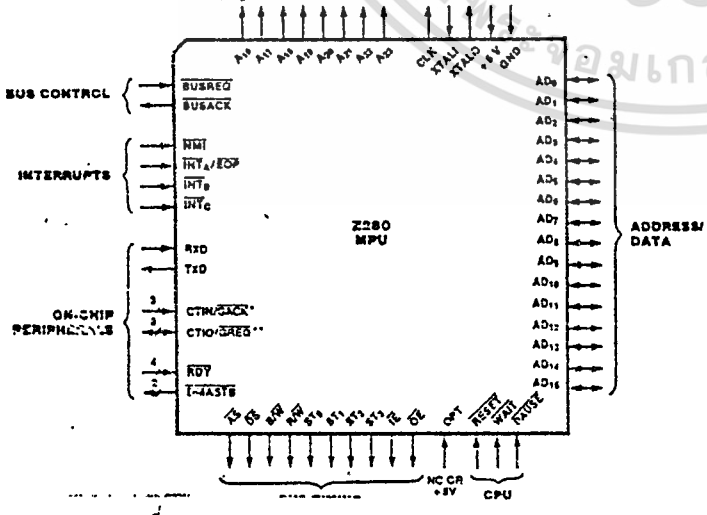


รูปที่ 1.1a Z280 Pin Functions,Z80 Configuration



รูปที่ 1.1b Z280 Pin Assignments,Z80 Bus

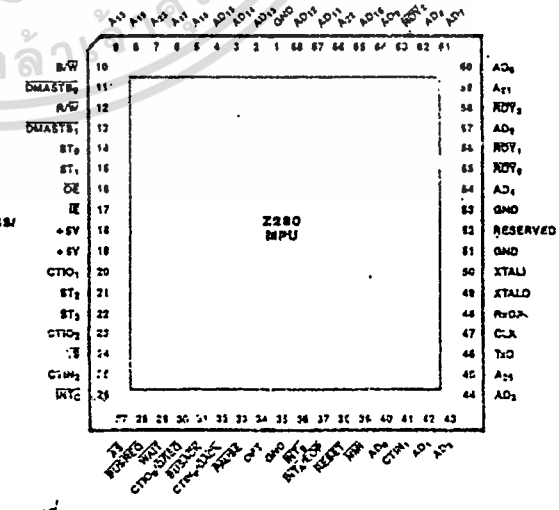
(Input OPT tied to GND)



รูปที่ 1.2a Z280 Pin Functions,Z-BUS Configuration

(input OPT tied to +5V or not connected)

(Input OPT tied GND)



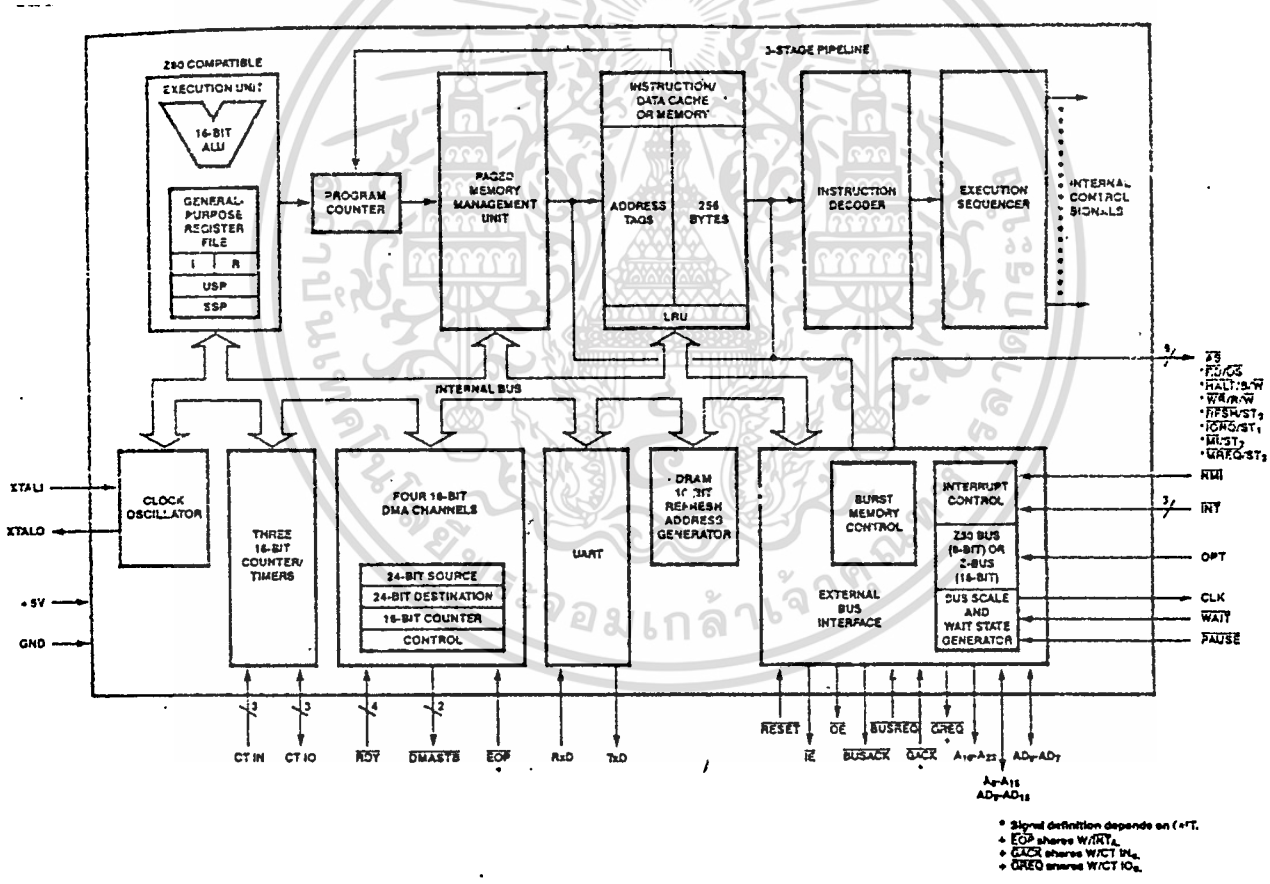
รูปที่ 1.2b Z280 Pin Assignments,Z-BUS

(input OPT tied to +5V or not connected)

1.2 การทำงานใน User และ System mode

Z280 สามารถทำงานได้ทั้งใน user และ system mode ในโหมด user บางคำสั่งไม่สามารถใช้ได้และรีจีสเตอร์บางตัวไม่สามารถควบคุมได้ โดยทั่วไปการทำงานในโหมดนี้ใช้สำหรับโปรแกรมแอพพลิเคชั่น ส่วนในโหมด system ทุกคำสั่งสามารถทำงานได้และรีจีสเตอร์ภายใน CPU ทุกตัวสามารถควบคุมได้ โหมดนี้ใช้กับโปรแกรมที่ทำงานในระบบปฏิบัติการ การแบ่งแยกโหมดเช่นนี้ทำให้ระบบมีความปลอดภัยมากขึ้น

เพื่อที่จะรองรับการทำงานทั้ง 2 โหมด จึงต้องมีตัวชี้สแตก 2 ตัว โดยตัวหนึ่งสำหรับ user stack อีกตัวสำหรับ system stack stack pointer ทั้งสองตัวนี้ทำให้ง่ายในการสับเปลี่ยนการทำงานเมื่อมีอินเตอร์รัพท์หรือ trap เกิดขึ้น เพื่อที่จะแน่ใจได้ว่า user stack ไม่มีข้อมูลของระบบอยู่ ข้อมูลของระบบขณะที่เกิดอินเตอร์รัพท์หรือ trap จะถูกเก็บลง system stack ก่อนที่สถานะใหม่ของโปรแกรมจะถูกโหลดขึ้นมา



รูปที่ 1.3 Z280 MPU Block Diagram

1.3 การอ้างแอดเดรส

สถาปัตยกรรมของ Z280 ออกแบบให้อ้างแอดเดรสได้แตกต่างกัน 4 ชนิด คือ

- CPU register space
- CPU control and status register space
- Memory address space
- I/O address space

CPU register Space ประกอบด้วยพื้นที่รีจีสเตอร์ทั้งหมดที่แสดงในรูปที่ 1.4 รีจีสเตอร์ประเภทนี้ถูกใช้ในการโยกย้าย data และ address การเข้าถึงรีจีสเตอร์เหล่านี้จะระบุในชุดคำสั่ง โดยที่รีจีสเตอร์เหล่านี้มีชื่อ A,F,B,C,D,E,H,L,A',F',B',C',D',E',H',L',IX, IY,SSP,USP,PC,I และ R

CPU Control and Status Register Space ประกอบด้วยพื้นที่รีจีสเตอร์ทั้งหมดที่แสดงในรูปที่ 1.5 รีจีสเตอร์เหล่านี้ควบคุมการทำงานของ CPU และสามารถกำหนดค่ารีจีสเตอร์เหล่านี้ได้ด้วย คำสั่ง Load Control ชนิดพิเศษเท่านั้น รีจีสเตอร์เหล่านี้ประกอบไปด้วย Bus Timing and Control register,Bus Timing and Initialization register,Local Address register,Cache Control register,Master status register,Interrupt Status register,Interrupt/Trap Vector Table Pointer ,I/O Page register,Trap Control register และ System Stack Limit register

Memory Address Space มีการอ้างแอดเดรส 2 ชนิดใน Z280 ชนิดหนึ่งสำหรับการทำงานใน user mode อีกชนิดสำหรับการทำงานใน system mode ซึ่งจะเลือกได้โดยกำหนด User/System Mode (U/S) bit ใน Master Status register ซึ่งไปควบคุมการเลือกของ Page Descriptor register ในระหว่างการแปลงหาแอดเดรสที่แท้จริง

แต่ละพื้นที่การแอดเดรสสามารถถูกมองเป็นพื้นที่ขนาด 64 K byte เรียงต่อๆกันไป การอ้างแอดเดรสแบบ 8 bit (byte) เป็นการอ้างแอดเดรสพื้นฐานที่สุดในการอ้างหน่วยความจำแบบนี้ อย่างไรก็ตามยังมีการอ้างแอดเดรสแบบอื่นๆอีก เช่น bit,word (2 byte),byte string และ multiple-byte ในการทำงานกับ EPU

การอ้างแอดเดรสแบบ multiple-byte คือการแอดเดรสแบบ byte กับส่วนแอดเดรสล่างสุด การอ้างแอดเดรสแบบ multiple-byte สามารถที่จะเก็บค่าแอดเดรสเริ่มต้นของหน่วยความจำทั้งแอดเดรสคู่และแอดเดรสคี่

I/O Address Space การอ้างแอดเดรสแบบนี้จะเกิดขึ้นเมื่อมีการใช้คำสั่งเกี่ยวกับ I/O เท่านั้น (เช่น คำสั่ง IN,OUT,I/O block move) โดยที่แอดเดรสของ I/O แบบ logical ซึ่งเป็นข้อมูล 8 bit บน Address A0-A7 จะรวมกับ รีจีสเตอร์ A ในการอ้างแอดเดรสโหมด Direct Address หรือ รวมกับรีจีสเตอร์ B ในการอ้างแอดเดรสโหมด Indirect Register หรือ รวมกับรีจีสเตอร์ B เช่นเดียวกัน เมื่อมีการใช้คำสั่งเกี่ยวกับ I/O block บนสายแอดเดรส A8-A15 ทำให้ได้แอดเดรสขนาด 16 บิท ซึ่งจะไปรวมกับ page register ที่กำหนดการใช้เพจของ I/O ซึ่งมีขนาด 8 บิท ดังนั้นการอ้างแอดเดรสที่สมบูรณ์ในการระบุแอดเดรสของ I/Oพอร์ทประกอบด้วย I/O page อยู่บนสายแอดเดรส A23-A16 ,ค่าของรีจีสเตอร์ A หรือ B อยู่บนสายแอดเดรส A8-A15 และ I/O address อยู่บนสายแอดเดรส A7-A0

ข้อแตกต่าง การอ้างอิงหน่วยความจำแบบ 16 bit หรือเฟทช์คำสั่งจะเกิดการอ้างอิงหน่วยความจำสองแบบ แต่การอ้างหรือเฟทช์คำสั่ง I/O แบบ 16 bit จะเกิดการจัดการ I/O bus แบบเดียว โดยไม่สนใจขนาดของบัสหรือการอ้าง I/O พอร์ท อย่างไรก็ตามอุปกรณ์สนับสนุนภายในที่มีรีจีสเตอร์ขนาด 16 bit จะต้องใช้คำสั่ง word I/O เท่านั้น โดยไม่ขึ้นกับขนาดของบัสภายนอก(ตารางที่ 1.1)

1.4 ชนิดของข้อมูล

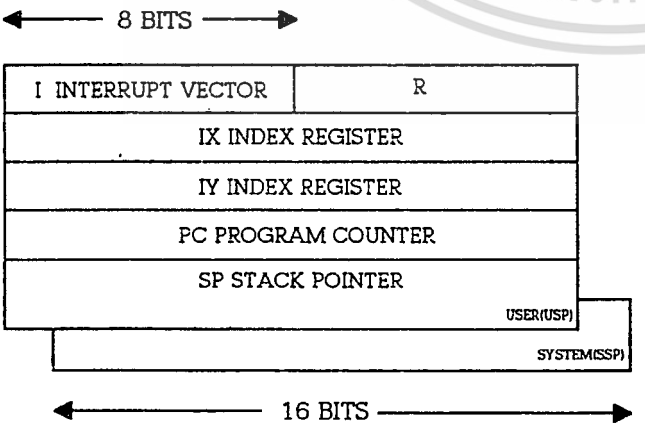
CPU สามารถจัดการกับ bit ,binary-coded decimal (BCD) (4 bit), byte (8 bit), word (16 bit), byte string และ word string แต่ละบิทในรีจีสเตอร์หรือหน่วยความจำสามารถถูก set,clear และ test ค่าได้ รหัส BCD 2 ตัวสามารถรวมให้เป็น 1 byte ได้โดยใช้คำสั่ง Decimal Adjust Accumulator (พร้อมกับการบวกและลบเลข binary) และคำสั่ง Rotate Digit ข้อมูลระดับ byte ทำงานในคำสั่ง load ข้อมูลระดับ 8 บิท, คาสั่งทางคณิตศาสตร์,คำสั่งทางตรรกะ และคำสั่งเลื่อนและหมุนข้อมูล ข้อมูลระดับ word ทำงานลักษณะคล้ายกันในคำสั่ง load 16 bit และคำสั่งทางคณิตศาสตร์ 16 bit ในการทำคำสั่ง block move และ search สามารถที่จะย้ายข้อมูลระดับ byte string ได้ถึง 64K byte คำสั่ง block I/O word สามารถย้ายข้อมูลระดับ word string ได้ถึง 32K word ในการทำงานที่สนับสนุน EPU ข้อมูลระดับ byte string สามารถที่จะถูกถ่ายเทไปมาโดย CPU ได้ถึง 16 byte

1.5 รีจีสเตอร์ของ Z280

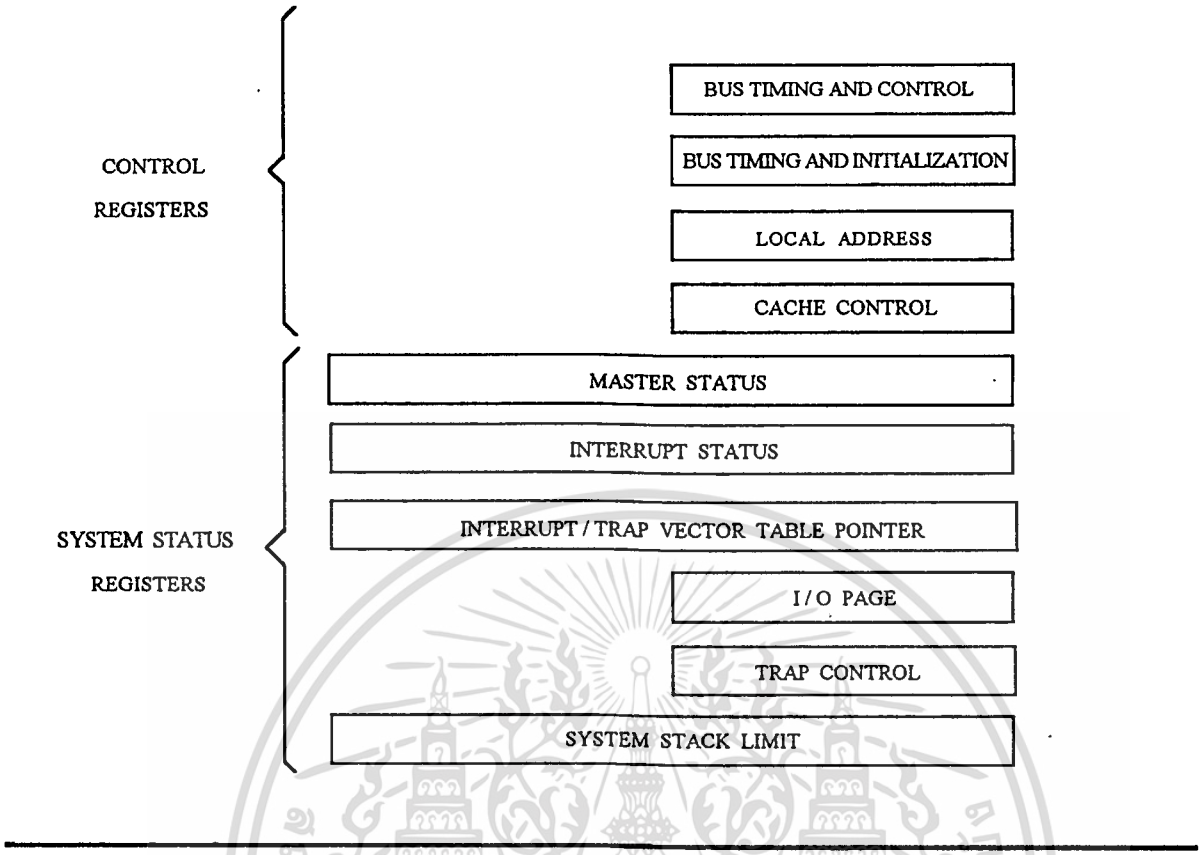
ไมโครโปรเซสเซอร์ Z280 มีรีจีสเตอร์ที่สามารถถูกโปรแกรมได้ถึง 23 ตัว (รูปที่ 1.4) ซึ่งรีจีสเตอร์เหล่านี้อยู่ใน CPU register space

รีจีสเตอร์หลักในการทำงาน รีจีสเตอร์ใช้งานถูกแบ่งเป็นรีจีสเตอร์ขนาด 8 bit 2 กลุ่ม คือ รีจีสเตอร์หลักและรีจีสเตอร์สำรอง (มีสัญลักษณ์ ') แต่ละกลุ่มประกอบด้วย แอคคิวมูเลเตอร์ (A), รีจีสเตอร์แฟลก (F) และรีจีสเตอร์ใช้งานทั่วไป 6 ตัว(B,C,D,E,H และ L) ในเวลาหนึ่งจะมีอยู่รีจีสเตอร์กลุ่มเดียวเท่านั้นที่ active เมื่อมีการรีเซตกลุ่มรีจีสเตอร์หลักจะ active คำสั่ง Exchange จะยอมให้โปรแกรมเมอร์สลับการทำงานในกลุ่มที่ active และกลุ่มที่ไม่ active

PRIMARY FILE		AUXILIARY FILE	
A ACCUMULATOR	F FLAG รีจีสเตอร์	A' ACCUMULATOR	F' FLAG รีจีสเตอร์
B GENERAL PURPOSE	C GENERAL PURPOSE	B' GENERAL PURPOSE	C' GENERAL PURPOSE
D GENERAL PURPOSE	E GENERAL PURPOSE	D' GENERAL PURPOSE	E' GENERAL PURPOSE
H GENERAL PURPOSE	L GENERAL PURPOSE	H' GENERAL PURPOSE	L' GENERAL PURPOSE



รูปที่ 1.4 CPU Register Configuration



รูปที่ 1.5 CPU Control and Status Register

แอคคิวมูเลเตอร์เป็นรีจีสเตอร์ปลายทางในการทำงานทางคณิตศาสตร์และตรรกะ 8 bit รีจีสเตอร์ใช้งานทั่วไป 6 ตัวสามารถที่จะจับกันเป็นคู่ได้ (BC,DE และ HL) เพื่อใช้เป็นรีจีสเตอร์ใช้งานทั่วไปขนาด 16 bit ส่วนรีจีสเตอร์ HL จะสงวนไว้สำหรับเป็นแอคคิวมูเลเตอร์ขนาด 16 bit สำหรับการทำงานทางคณิตศาสตร์ 16 bit

CPU Flag Register รีจีสเตอร์แฟลกจะ set หรือ reset ตามการทำงานของ CPU อยู่ 6 bit ดังรูปที่ 1.6

S	Z	0	H	0	P / V	N	C

รูปที่ 1.6 CPU Flag Register

แต่ละ bit ในรีจีสเตอร์นี้คือ

Carry (C) บิทนี้จะถูก set เมื่อคำสั่งบวกเกิดการทดหรือคำสั่งลบเกิดการขอยืม คำสั่งทางตรรกะและคำสั่งเลื่อนและหมุนข้อมูลที่เหมาะสมก็จะมีผลกับบิท

Add/Subtract (N) บิทนี้ถูกใช้โดยคำสั่ง Decimal Adjust Accumulator เพื่อแยกระหว่างการทำคำสั่งบวกและลบ บิทนี้จะถูก set สำหรับคำสั่งลบและ clear สำหรับคำสั่งบวก

Parity/Overflow (P/V) ระหว่างทำคำสั่งทางคณิตศาสตร์บิทนี้จะถูก set เมื่อเกิดสภาวะโอเวอร์โฟลว์ ระหว่างทำคำสั่งตรรกะและคำสั่งหมุนบิทนี้จะถูก set เมื่อคำตอบมีพาริตี้คู่หรือ clear เมื่อเป็นพาริตี้คี่

Half Carry (H) บิทนี้จะถูก set ถ้าการกระทำทางคณิตศาสตร์ขนาด 8 bit เกิดการทดหรือการยืมระหว่างบิทที่ 3 และ 4 หรือถ้าเป็นการกระทำขนาด 16 bit เกิดการทดหรือยืมระหว่างบิทที่ 11 และ 12 บิทนี้ถูกใช้ตรวจสอบความถูกต้องของคำตอบในการบวกหรือลบเลข BCD

Zero (Z) บิทนี้จะถูก set ถ้าคำตอบในการกระทำทางคณิตศาสตร์หรือตรรกะเป็นศูนย์

Sign(S) บิทนี้ใช้เก็บค่าบิทที่มีนัยสำคัญสูงสุดของแอคคิวมูเลเตอร์ Sign flag นี้ถูกใช้เป็นตัวระบุคำตอบของคำสั่ง test และ set ด้วย

Peripheral	Address (Hexadecimal)			
Refresh Rate Register	FFxxE8			
UART				
Configuration	FExx10			
Transmitter Control/Status	FExx12			
Receiver Control/Status	FExx14			
Receiver Data	FExx16			
Transmitter Data	FExx18			
MMU				
Master Control	FFxxF0			
Page Descriptor Register Pointer	FFxxF1			
Descriptor Select Port	FFxxF5			
Block Move Port	FFxxF4			
Invalidation I/O Port	FFxxF2			
Page Descriptor Registers*				
User PDR 0	00			
User PDR 1	01			
. .	. .			
User PDR 14	0E			
User PDR 15	0F			
System PDR 0	10			
System PDR 1	11			
. .	. .			
System PDR 14	1E			
System PDR 15	1F			
DMA				
Master Control	FFxx1F			
	DMA0	**DMA1**	**DMA2**	**DMA3**
Destination Address (bits 0-11)	FFxx00	FFxx08	FFxx10	FFxx18
Destination Address (bits 12-23)	FFxx01	FFxx09	FFxx11	FFxx19
Source Address (bits 0-11)	FFxx02	FFxx0A	FFxx12	FFxx1A
Source Address (bits 12-23)	FFxx03	FFxx0B	FFxx13	FFxx1B
Count	FFxx04	FFxx0C	FFxx14	FFxx1C
Transaction Descriptor	FFxx05	FFxx0D	FFxx15	FFxx1D
Counter/Timer	**C/T0**	**C/T1**		**C/T2**
Configuration	FExxE0	FExxE8		FExxF8
Command/Status	FExxE1	FExxE9		FExxF9
Time Constant	FExxE2	FExxEA		FExxFA
Count-Time	FExxE3	FExxEB		FExxFB

*The Page Descriptor register address must be loaded into the Page Descriptor Register Pointer in order to access that Page Descriptor register.

ตารางที่ 1.1 On-Chip Peripheral I/O Addresses

1.6 รีจีสเตอร์ที่ใช้งานเฉพาะด้าน

Index Registers มี 2 ตัว คือ IX และ IY โดยแต่ละตัวเก็บค่าแอดเดรสขนาด 16 บิท ซึ่งถูกใช้เป็นดัชนีในการอ้างแอดเดรส Index register สามารถที่จะทำหน้าที่เป็นรีจีสเตอร์ใช้งานทั่วไปโดยแบ่งเป็น ไบท์บนและไบท์ล่างซึ่งสามารถเข้าถึงแต่ละตัวได้โดยอิสระ ไบท์สูงและไบท์ต่ำของรีจีสเตอร์ IX เรียกว่า IXH และ IXL ส่วนไบท์สูงและไบท์ต่ำของรีจีสเตอร์ IY เรียกว่า IYH และ IYL

Interrupt Register (I) ถูกใช้ในอินเตอร์รัพท์โหมด 2 ซึ่งจะสร้างค่าแอดเดรสทางโลจิกโดยอ้อมขนาด 16 bit เพื่อชี้ interrupt service routine Interrupt register จะส่งแอดเดรสโดยอ้อม 8 บิทบนรวมกับ 8 บิทล่างของอุปกรณ์ที่ร้องขออินเตอร์รัพท์

Program Counter (PC) เป็นรีจีสเตอร์ขนาด 16 บิทที่เป็นตัวกำหนดตำแหน่งสัมพันธ์ (relative) ของคำสั่งในโปรแกรมที่กำลังกระทำอยู่ เพื่อที่จะนำไปกำหนดตำแหน่งที่แท้จริงต่อไป ในสภาวะกระทำการเฟทช์ โปรแกรมเคาน์เตอร์จะเก็บแอดเดรสทางโลจิกขนาด 16 บิท ซึ่งคำสั่งของโปรแกรมในขณะนั้นอ่านจากหน่วยความจำ

R Register รีจีสเตอร์ R ถูกใช้เหมือนกับรีจีสเตอร์ใช้งานทั่วไปขนาด 8 บิทที่ทั้งเขียนและอ่านได้ รีจีสเตอร์ R ไม่มีความสัมพันธ์กับแอดเดรสที่ถูกรีเฟรชและค่าของมันจะเปลี่ยนโดยผู้ใช้เท่านั้น

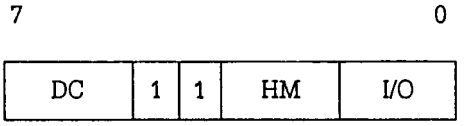
Stack Pointers ทั้ง 2 ตัว คือ User Stack Pointer (USR) และ System Stack Pointer (SSR) สนับสนุนการทำงาน 2 โหมดของไมโครโปรเซสเซอร์ SSP ถูกใช้บันทึกข้อมูลเมื่อเกิดอินเตอร์รัพท์หรือ trap เกิดขึ้นและสนับสนุนการ call และ return โปรแกรมย่อย ใน system mode USP ถูกใช้สนับสนุนการ call และ return โปรแกรมย่อย ใน user mode

Status and Control Registers ไมโครโปรเซสเซอร์ Z280 มีรีจีสเตอร์บอกสภาวะและควบคุมที่โปรแกรมได้อยู่ 10 ตัว

ตารางที่ 1.2 แสดงตำแหน่งของรีจีสเตอร์เหล่านี้ในพื้นที่แอดเดรสของรีจีสเตอร์บอกสภาวะและควบคุม

CONTROL รีจีสเตอร์ NAME	Address (Hexadecimal)
Bus Timing and Control รีจีสเตอร์	Control 02
Bus Timing and Initialization รีจีสเตอร์	Control FF
Cache Control[1]	Control 12
Interrupt Status	Control 16
Interrupt/Trap Vector Table	Control 06
I/O Page รีจีสเตอร์	Control 08
Local Address รีจีสเตอร์[2]	Control 14
Master Status (MSR)	Control 00
Stack Limit	Control 04
Trap Control	Control 10
1 ดูรายละเอียดจาก On-chip memory	
2 ดูรายละเอียดจาก multiprocessing mode of operation	

Bus Timing and Control Register เป็นรีจีสเตอร์ขนาด 8 bit (รูปที่ 1.7) ที่ควบคุมฐานเวลาในการจัดการแอดเดรสส่วน high memory และ ฐานเวลาลูกโซ่สำหรับการร้องขออินเตอร์รัพท์ได้ดีเทียบเท่าการร้องขออินเตอร์รัพท์จากขาของไมโครโปรเซสเซอร์โดยตรง เมื่อรีเซตซีพียู รีจีสเตอร์นี้จะมีค่า 30 H



รูปที่ 1.7 Bus Timing and Control Register

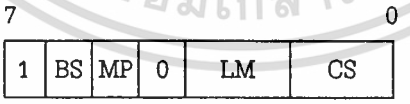
แต่ละบิทในรีจีสเตอร์นี้ประกอบด้วย

I/O Wait Insertion(I/O) 2 บิทนี้บอกถึงจำนวน wait state ที่จะเพิ่มเข้าไป (ซึ่งในการติดต่อกับ I/O จะแทรก 1 ลูกโดยอัตโนมัติอยู่แล้ว) จะถูกแทรกโดย CPU ทั้งในขณะการจัดการ I/O และ vector response timing (00=ไม่มีการแทรก,01=หนึ่ง,10=สอง,11=สาม)

High Memory Wait Insertion(HM) 2 บิทนี้บอกจำนวนของ wait state (00=ไม่มีแทรก,01=หนึ่ง,10=สอง,11=สาม) ที่จะถูกแทรกอย่างอัตโนมัติโดย CPU ในการจัดการหน่วยความจำเมื่อ MMU ทำงาน และบิทที่ 15 ของ Page Descriptor register จะต้องเป็น 1

Daisy Chain Timing(DC) 2 บิทนี้ใช้บอกจำนวนของ wait state ที่จะเพิ่มเข้ามาอย่างอัตโนมัติสำหรับ CPU ในขณะสัญญาณการตอบรับอินเตอร์รัพท์ถูกยืนยัน (00=ไม่มีแทรก,01=หนึ่ง,10=สอง,11=สาม) ถ้า DC มีค่าเป็น 01 ระบุ wait state จำนวน 1 ไซเคิลที่ถูกเพิ่มเข้าไปในการผ่านสัญญาณปกติ 4 ไซเคิลระหว่างสัญญาณตอบรับอินเตอร์รัพท์ $\overline{AS}$,$\overline{DS}$ (หรือ IORQ) ถูกยืนยัน

Bus Timing and Initialization Register เป็นรีจีสเตอร์ขนาด 8 บิทที่ถูกใช้ระบุสัญญาณควบคุมสำหรับการอินเตอร์เฟสบัสภายนอกเมื่อ MMU ทำงาน หรือเมื่อ MMU ไม่ทำงานพร้อมกับบิทที่15 ของ Page Description register เป็น 0 รีจีสเตอร์ตัวนี้ถูกใช้ควบคุมความสัมพันธ์ระหว่างอัตราความถี่สัญญาณภายในโปรเซสเซอร์กับการความถี่ที่บัสด้วย รีจีสเตอร์ตัวนี้สามารถโปรแกรมได้โดย reset โดย hardware ภายนอก



รูปที่ 1.8 Bus Timing and Initialization Register

เมื่อมีการ reset รีจีสเตอร์ตัวนี้จะมีค่าเริ่มต้นได้ 2 แบบ ขึ้นอยู่กับมีสัญญาณ WAIT เข้ามาขณะที่เป็นขอบขาขึ้นของการรีเซตหรือไม่ ถ้าไม่มีสัญญาณ WAIT รีจีสเตอร์จะมีค่า 80H ถ้ามีสัญญาณ WAIT ระหว่างรีเซตแล้ว รีจีสเตอร์นี้จะมีค่าตามข้อมูลบนแอดเดรสบัส

แต่ละบิทในรีจีสเตอร์นี้ประกอบด้วย

Clock Scaling (CS) 2 บิทนี้ ใช้บอกสเกลของสัญญาณนาฬิกาภายใน CPU สำหรับการติดต่อกันบนบัสทั้งหมด (00 = 1 bus clock cycle เท่ากับ 2 clock cycle ภายใน processor ,01 = bus clock cycle เท่ากับ clock cycle ของ processor ภายใน, 10 = 1 bus clock cycle เท่ากับ 4 clock cycle ภายใน processor ,11 = สงวนไว้) ทั้งสองบิทนี้ไม่สามารถแก้ไขได้ด้วย software

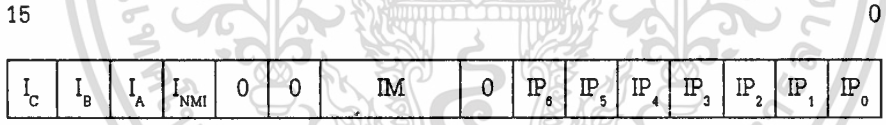
Low Memory Wait Insertion (LM) 2 บิทนี้ใช้ระบุจำนวนของ wait state (00 = ไม่ได้แทรก ,01 = หนึ่ง ,10 = สอง, 11 = สาม) สำหรับ CPU ซึ่งจะแทรกอย่างอัตโนมัติเพื่อให้ CPU ติดต่อกับหน่วยความจำเมื่อ MMU ไม่ทำงานหรือเมื่อ MMU ทำงาน พร้อมกับบิทที่ 15 ของ Page Descriptor register เป็น 0

Multiprocessor Configuration Enable (MP) ในบิทนี้บอกการทำงานใน mode multiprocessor (0 = ไม่ทำงาน,1 = ทำงาน) (ดูในส่วน Multiprocessor Mode)

Bootstrap Mode Enable (BS) ในบิทนี้บอกการทำงานของ bootstrap mode (0 = ไม่ทำงาน,1 = ทำงาน) (ดูในส่วน UART สำหรับรายละเอียดเกี่ยวกับ bootstrap mode) เมื่ออ่านค่ากลับจะมีค่าเป็นหนึ่ง

ระหว่าง Initialization AD4 จะเป็น 0

Interrupt Status Register เป็นรีจีสเตอร์ขนาด 16 บิท (รูปที่ 1.9) ที่ใช้ระบุโหมดของอินเทอรัพท์ที่จะใช้ และให้มีสัญญาณร้องขออินเทอรัพท์จาก sources ค้างอยู่ รีจีสเตอร์นี้ยังประกอบไปด้วยบิทที่ระบุว่าสัญญาณอินเทอรัพท์ถูกชี้หรือไม่ มีเพียงบิท interrupt vector enable เท่านั้นที่เขียนข้อมูลได้ บิทอื่น ๆ ทั้งหมดจะอ่านข้อมูลได้อย่างเดียว



รูปที่ 1.9 Interrupt Status Register

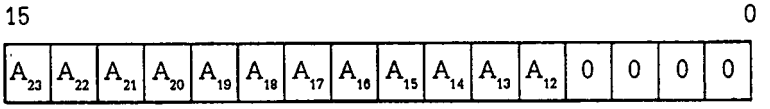
แต่ละบิทใน register นี้ประกอบไปด้วย

Interrupt Request Pending (IP) เมื่อบิท IP_n เป็น 1 การร้องขออินเทอรัพท์จาก sources ที่ระดับ n จะค้างอยู่ (ดูในส่วน Interrupt และ Trap Structure)

Interrupt Mode (IM) ค่าของ n ใน 2 บิทนี้ใช้ระบุโหมดการทำงานของอินเทอรัพท์ 2 บิทนี้สามารถเปลี่ยนได้โดยคำสั่ง IM

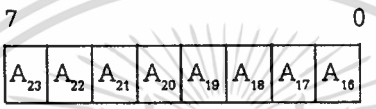
Interrupt Vector Enable (I) 4 บิทนี้ใช้ระบุว่าขา interrupt input ทั้ง 4 ขา แต่ละขาถูกชี้อยู่ เมื่อ I_n เป็น 1 การ interrupt บนขา interrupt n จะถูกชี้เมื่อ CPU อยู่ในอินเทอรัพท์โหมด 3 เมื่อ I_n ถูกเคลียร์เป็น 0 การอินเทอรัพท์ทั้งหมดจะใช้ตาราง Interrupt/Trap Vector เดียวกันทั้งหมด บิทนี้จะไม่ถูกสนใจเลย ถ้าไม่ได้ทำงานใน interrupt mode 3

Interrupt/Trap Vector Table Pointer เป็น register ขนาด 16 บิท (รูปที่ 1.10) ที่เก็บค่าแอดเดรสแท้จริง (physical address) ที่มีนัยสำคัญสูงสุด 12 บิท ของตาราง Interrupt/Trap Vector ส่วน 12 บิทล่างของ physical address จะเป็นศูนย์ ส่วน 4 บิทที่มีนัยสำคัญต่ำสุดของรีจีสเตอร์นี้ จะต้องโปรแกรมให้เป็นศูนย์



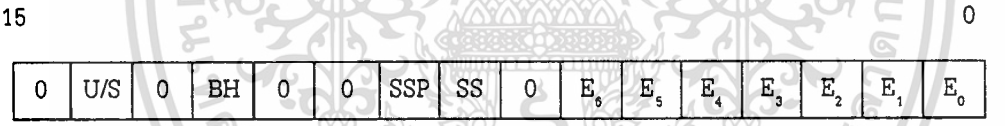
รูปที่ 1.10 Interrupt/Trap Vector Table Pointer

I/O Page Register เป็น register ขนาด 8 บิท (รูปที่ 1.11) ที่ใช้เพิ่มเติมสัญญาณ output 16 บิทซึ่งระบุ I/O address ในระหว่างการจัดการ I/O



รูปที่ 1.11 I/O Page Register

Master Status Register เป็นรีจีสเตอร์ ขนาด 16 บิท (รูปที่ 1.12) ที่บันทึกข้อมูลสภาวะเกี่ยวกับการปฏิบัติการโปรแกรมในปัจจุบันไว้ รีจีสเตอร์นี้จะถูกเคลียร์เป็นศูนย์เมื่อมีการ reset



รูปที่ 1.12 Master Status Register

แต่ละบิทใน register นี้ประกอบด้วย

Interrupt Request Enable (E_n) มี interrupt enable อยู่ 7 บิท แต่ละบิทมีไว้สำหรับ maskable interrupt source ในแต่ละอย่าง (ทั้งภายนอกและภายใน) เมื่อบิท E_n เป็นหนึ่ง การร้องขอ interrupt จาก sources ที่ระดับ n จะได้รับการตอบรับจาก CPU เมื่อบิทนี้เป็นศูนย์ การร้องขอ interrupt ที่ระดับ n จะไม่ได้รับการตอบรับ

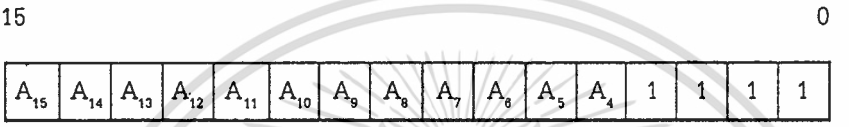
Single-Step (SS) ในขณะที่บิทนี้เป็นหนึ่ง CPU จะอยู่ใน single-stepping mode ถ้าบิทนี้เคลียร์เป็นศูนย์ การทำงานแบบ single-step โดยอัตโนมัติจะไม่ทำงาน บิทนี้จะถูกเคลียร์โดยอัตโนมัติเมื่อเกิด trap หรือ interrupt

Single-Step Pending (SSP) ขณะที่บิทนี้เป็นหนึ่ง CPU จะสร้าง trap ก่อนที่จะ execute คำสั่ง บิท SS จะ copy ตัวเองไปที่ SSP อย่างอัตโนมัติ เมื่อสิ้นสุดการทำงานในแต่ละคำสั่งโดยสมบูรณ์ เมื่อเกิด Single-Step,Page Fault,Privileged Instruction,Breakpoint-on-Halt หรือ Division trap บิทนี้จะถูกเคลียร์เป็นศูนย์โดยอัตโนมัติ ดังนั้นบิท SSP ที่ถูกบันทึกใน Master Status register จะถูกเคลียร์เป็นศูนย์

Breakpoint-on-Halt Enable (BH) ในขณะที่บิทนี้เป็นหนึ่ง เมื่อ CPU พบคำสั่ง HALT จะสร้าง Breakpoint trap เพื่อหยุดคำสั่งนี้ เมื่อบิทนี้ถูกเคลียร์เป็นศูนย์ คำสั่ง HALT จะถูกปฏิบัติโดยปกติ

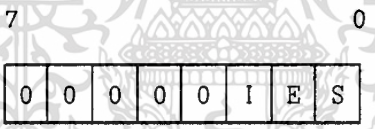
User/System Mode (U/S) ในขณะที่บิทนี้เคลียร์เป็นศูนย์ CPU จะทำงานใน system mode ในขณะที่บิทนี้เซตเป็นหนึ่ง CPU จะทำงานใน user mode

System Stack Limit Register เป็นรีจีสเตอร์ขนาด 16 บิท (รูปที่ 1.13) ที่ใช้ชี้ว่าเมื่อไรจะต้องสร้าง System Stack Overflow Warning trap ถ้าให้รีจีสเตอร์ตัวนี้มีผลโดยการเซต control bit ใน Trap Control register การ push ค่าลงไปบน system stack จะทำให้ 12 บิทที่มีนัยสำคัญสูงสุดในรีจีสเตอร์ตัวนี้ ถูกเปรียบเทียบกับ 12 บิทบนของ System Stack Pointer และ trap จะเกิดขึ้นถ้าทั้งสองส่วนเหมือนกัน 4 บิทต่ำสุดของรีจีสเตอร์นี้จะต้องโปรแกรมให้เป็นศูนย์ แต่จะถูกอ่านกลับเป็นหนึ่ง



รูปที่ 1.13 System Stack Limit register

Trap Control register เป็นรีจีสเตอร์ ขนาด 8 บิท (รูปที่ 1.14) ที่ทำให้เกิด maskable trap ได้ เมื่อรีเซต รีจีสเตอร์นี้จะเป็นศูนย์ทั้งหมด



รูปที่ 1.14 Trap Control register

แต่ละบิทในรีจีสเตอร์ประกอบด้วย

System Stack Overflow Warning (S) ในขณะที่บิทนี้เซตเป็นหนึ่ง CPU จะเกิด Stack Overflow Warning trap เมื่อเกิด system stack ในพื้นที่หน่วยความจำที่เฉพาะเจาะจง

EPU Enable (E) ในขณะที่บิทนี้เคลียร์เป็น 0 CPU จะเกิด trap เมื่อใดก็ตามที่พบคำสั่ง EPA

Inhibit User I/O (I) ในขณะที่บิทนี้เซตเป็นหนึ่ง CPU จะสร้าง Privileged Instruction trap เมื่อพบคำสั่ง I/O ใน user mode

Cache Control and Local Address register ดูในส่วนของ On-Chip Memory สำหรับข้อมูลเกี่ยวกับ Cache Control รีจีสเตอร์ และ ส่วนของ Multiprocessor Mode สำหรับข้อมูลของ Local Address register

1.7 Interrupt and Trap Structure

ไมโครโปรเซสเซอร์ Z280 ได้เตรียมโครงสร้างของการอินเตอร์รัพท์และ trap ที่ยืดหยุ่นและมีประสิทธิภาพมาก อินเตอร์รัพท์คือการร้องขอบริการจาก CPU แบบไม่ประสานจังหวะ (asynchronous) และโดยทั่วไปถูกทริกโดยอุปกรณ์ที่ต้องการขอบริการ trap คือผลลัพธ์ที่ได้จากการปฏิบัติตามคำสั่งที่แน่นอนแบบประสานจังหวะ (synchronous)

Interrupts ไมโครโปรเซสเซอร์ Z280 มีการอินเตอร์รัพท์อยู่ 2 ชนิดคือ nonmaskable และ maskable การอินเตอร์รัพท์แบบ nonmaskable (NMI) ไม่สามารถหยุดยั้งโดยซอฟท์แวร์และโดยทั่วไปสงวนไว้สำหรับการร้องขอบริการจากอุปกรณ์ภายนอกที่มีความต้องการบริการสูงสุด ส่วนการอินเตอร์รัพท์แบบ maskable สามารถเลือกให้อินเตอร์รัพท์หรือไม่โดยกำหนดที่ซอฟท์แวร์ แต่ทั้งการอินเตอร์รัพท์แบบ nonmaskable และ maskable สามารถที่จะโปรแกรมให้ชี้ตำแหน่ง address หรือไม่ชี้ก็ได้ การยอมรับอินเตอร์รัพท์จะเกิดระหว่างการปฏิบัติคำสั่งต่างๆและการตอบรับจะตามมาหลังจากปฏิบัติตามคำสั่งก่อนหน้านี้โดยสมบูรณ์แล้ว เมื่อเกิดอินเตอร์รัพท์ขณะใช้ คำสั่งเคลื่อนย้ายบล็อก,ค้นหา และ I/O ซึ่งเป็นคำสั่งที่ทำงานวนรอบคำสั่งเดิมไปเรื่อยๆ จนกว่างานจะเสร็จ จะมีการตอบสนองอินเตอร์รัพท์หลังจากทำคำสั่งปัจจุบันเสร็จ และเมื่อทำตาม interrupt routine เสร็จสิ้นแล้วก็จะกลับมาทำคำสั่งเดิม (เคลื่อนย้ายบล็อก,ค้นหา หรือ I/O) ต่อไป

Interrupt Sources ไมโครโปรเซสเซอร์ Z280 ยอมรับการอินเตอร์รัพท์แบบ nonmaskable จากขา NMI เท่านั้น แต่ไมโครโปรเซสเซอร์ Z280 ยอมรับการอินเตอร์รัพท์แบบ maskable จากขา INT และจากอุปกรณ์สนับสนุนภายในคือ counter/timer,DMA channel และตัวรับและตัวส่งของ UART

ขาอินเตอร์รัพท์ A,B และ C สามารถถูกโปรแกรมให้เลือกการอินเตอร์รัพท์แบบมีตัวชี้โดยกำหนดค่าของบิทที่เหมาะสมใน Interrupt Status register ในโหมด 3 การอินเตอร์รัพท์ภายนอกสามารถถูกโปรแกรมให้เป็นแบบมีตัวชี้ตำแหน่งหรือไม่มีตัวชี้ก็ได้

การทำงานของอินเตอร์รัพท์ในโหมดต่างๆ ตัว CPU มีการอินเตอร์รัพท์อยู่ 4 โหมด สามโหมดแรกเป็นโหมดการอินเตอร์รัพท์ที่รับมาจากไมโครโปรเซสเซอร์ Z80 เพื่อให้สัญญาณอินเตอร์รัพท์สามารถเข้ากันได้ โหมดที่สี่เตรียมขึ้นเพื่อให้เกิดความยืดหยุ่นขึ้นในการจัดการอินเตอร์รัพท์ อุปกรณ์สนับสนุนภายในใช้โหมดที่สี่นี้โดยไม่คำนึงถึงการร้องขออินเตอร์รัพท์ภายนอกว่าเลือกโหมดไหน โหมดในการอินเตอร์รัพท์ถูกเลือกได้โดยใช้คำสั่งพิเศษ IM0,IM1,IM2 หรือ IM3 ในการรีเซต ไมโครโปรเซสเซอร์ Z280 จะเซตค่าตัวเองให้อยู่ในอินเตอร์รัพท์โหมดศูนย์โดยอัตโนมัติ โหมดของอินเตอร์รัพท์ปัจจุบันสามารถอ่านได้จาก Interrupt Status Register

Mode 0: โหมดนี้เหมือนกับการอินเตอร์รัพท์ในไมโครโปรเซสเซอร์ 8080 ในโหมดนี้อุปกรณ์ที่ร้องขออินเตอร์รัพท์ที่ขา maskable interrupt ใดๆ สามารถส่งคำสั่ง call หรือ restart ลงบน data bus และ CPU จะปฏิบัติตาม ผลก็คืออุปกรณ์ที่ร้องขออินเตอร์รัพท์จะส่งคำสั่งใหม่ แทนที่คำสั่งเดิมในหน่วยความจำที่ควรจะปฏิบัติต่อไป

Mode 1: เมื่อโหมดนี้ถูกเลือก CPU จะตอบรับการอินเตอร์รัพท์ภายนอกแบบ maskable โดยทำคำสั่ง restart ไปที่ logical address 0038H ในพื้นที่ส่วนที่เป็น system program

Mode 2: โหมดนี้เป็นโหมดที่ตอบรับการอินเตอร์รัพท์แบบมีตัวชี้ตำแหน่ง เมื่อมีการอินเตอร์รัพท์ตัวชี้ address ขนาด 16 บิทถูกสร้างขึ้นโดย 8 บิทบนของตัวชี้นี้ถูกสร้างจากข้อมูลภายในรีจีสเตอร์ I ส่วน 8 บิทล่างของตัวชี้จะได้มาจากอุปกรณ์ที่ร้องขออิน

เตอร์รัพท์ ตัวชี้ขนาด 16 bit ที่ถูกสร้างนี้ทำงานใน logical address ในพื้นที่ของข้อมูลระบบ ซึ่งสามารถถูกแปลงโดย MMU ไปเป็น physical address ต่อไป

Mode 3: โหมดนี้เป็นโหมดจำเพาะสำหรับระบบปฏิบัติการ ที่ใช้ประโยชน์ส่วนที่พัฒนาต่อจากไมโครโปรเซสเซอร์ Z80 (เช่น single-step และ user/system mode) เพราะว่าสถานะปัจจุบันใน Master Status register จะถูกบันทึกโดยอัตโนมัติ และจะโหลดสถานะใหม่จากการอินเตอร์รัพท์ ตาราง vector สามารถใช้สำหรับ interrupt source ภายนอกเพื่อเพิ่ม interrupt vector สำหรับ Z8000 family,Z80 family และ Z8500 Universal Peripheral เมื่อมีการร้องขออินเตอร์รัพท์ (ทั้ง maskable หรือ nonmaskable) Master Status register,แอดเดรสของคำสั่งต่อไปที่จะถูกปฏิบัติ และ "reason code" ขนาด 16 bit จะถูก push ลงใน system stack ค่าของ Master Status register และ Program Counter ค่าใหม่จะถูกอ่านมาจาก Interrupt/Trap Vector Table

"reason code" ซึ่งเกิดจากอินเตอร์รัพท์ภายนอก คือข้อมูลของ bus ระหว่างการตอบรับอินเตอร์รัพท์ ซึ่งก็คือข้อมูล 8 บิทบนบัสข้อมูล ส่วนไบท์ที่มีนัยสำคัญสูงสุดของ reason code จะเป็นศูนย์ทั้งหมด สำหรับ reason code ที่เกิดจากการอินเตอร์รัพท์ของอุปกรณ์สนับสนุนภายใน จะเหมือนกับอุปกรณ์สนับสนุนสร้างอินเตอร์รัพท์ และเหมือนกับ vector address ใน Interrupt/Trap Vector Table ซึ่ง Interrupt/Trap Vector Table Pointer ถูกใช้ในการอ้างอิงตารางนี้

Traps ไมโครโปรเซสเซอร์ Z280 รองรับการเกิด trap ภายในถึง 8 trap trap ต่อไปนี้สามารถถูกยับยั้งไม่ให้มีผลได้ มีดังนี้ EPA trap (ซึ่งอนุญาตให้ซอฟท์แวร์จำลองการทำงานของ EPU) ; Stack Warning trap (ซึ่งเกิดเมื่อสิ้นสุดคำสั่งที่ทำให้เกิด trap) ; Breakpoint-on-Halt trap (ซึ่งเกิดขึ้นเมื่อพบคำสั่ง HALT) และ Single-Step trap ซึ่งเกิดในการใช้แต่ละคำสั่ง รวมทั้ง คำสั่ง I/O สามารถถูกระบุให้เป็นคำสั่งพิเศษ หรือ Privileged instruction trap จะทำให้คำสั่งจบการทำงานโดยปราศจากการแก้ไขค่ารีจีสเตอร์ต่างๆ ใน CPU (ยกเว้นสำหรับ System Stack Pointer ซึ่งถูกปรับปรุงเมื่อสถานะของโปรแกรมถูก push ลงบน system stack)

การบันทึกสถานะของโปรแกรมใน system stack และการเฟทช์สถานะของโปรแกรมค่าใหม่จาก Interrupt/Trap Vector Table มีลักษณะเหมือนกับการทำงานใน interrupt mode ต่างๆ

Trap สามารถเกิดขึ้นได้ถ้าลักษณะที่ทำให้เกิด trap ของ Z280 CPU เกิดขึ้นอย่างชัดเจนเท่านั้น (เช่น การเกิด System Stack Overflow warning) trap ไม่สามารถเกิดขึ้นได้จากกลุ่มคำสั่งของ Z80 ยกเว้นที่เกิดจากระบบปฏิบัติการซึ่งใช้ส่วนที่เพิ่มเติมของ Z280

Extended Instruction การ trap นี้จะเกิดขึ้นเมื่อ CPU พบคำสั่ง extend ในขณะที่บิทของ Extended Processing Architecture (EPA) ใน Trap Control register เป็นศูนย์ Trap vector 4 แบบถูกใช้โดย EPA trap ซึ่ง trap vector แต่ละแบบก็ใช้ตามคำสั่ง EPA แต่ละประเภท สิ่งนี้ทำให้ง่ายต่อการจัดการ trap ที่ใช้คำสั่ง I/O เพื่อเข้าถึง EPU หรือใช้ซอฟท์แวร์เลียนแบบการทำงานของ EPU

Privileged Instruction การ trap นี้จะเกิดขึ้นเมื่อใดก็ตามที่พยายามปฏิบัติตามคำสั่งชนิดพิเศษนี้ ในขณะที่ CPU อยู่ในโหมด user (User/System Mode control bit ใน Master Status register เป็นหนึ่ง)

System Call การ trap นี้จะเกิดขึ้นเมื่อใดก็ตามที่ปฏิบัติตามคำสั่ง System Call (SC)

Access Violation การ trap นี้จะเกิดขึ้นเมื่อใดก็ตามที่โหมดการแปลงของ MMU ทำงานและการอ้างแอดเดรสที่ถูกแปลงไม่ถูกต้อง หรือ(สำหรับการบันทึก)เกิด write-protected

System Stack Overflow Warning การ trap นี้จะเกิดขึ้นเมื่อบิท Stack Overflow Warning ใน Trap Control register ถูกเซตเป็นหนึ่งเท่านั้น สำหรับการพุช system stack ในแต่ละครั้ง บิทที่มีนัยสำคัญสูงสุดใน Stack Pointer register จะถูกเปรียบเทียบกับข้อมูลของ Stack Limit register และจะเกิด trap เมื่อค่าทั้งสองเหมือนกัน Stack Overflow Warning bit จะเคลียร์ตัวเองเป็นศูนย์โดยอัตโนมัติเพื่อป้องกันการเกิด trap ซ้ำ

Division Exception การ trap นี้จะเกิดขึ้นเมื่อหารเป็นศูนย์ (กรณีที่หารด้วยศูนย์) หรือค่าเศษส่วนจริงไม่สามารถแสดงความละเอียดได้เพียงพอ (overflow) รีจีสเตอร์แฟลกใน CPU จะเซตค่าเพื่อแบ่งแยกในสองกรณีนี้

Single-Step การ trap นี้จะเกิดขึ้นก่อนปฏิบัติคำสั่งใดๆ ถ้า Single-Step Pending control bit ใน Master Status register ถูกเซตเป็น 1 control bit 2 บิทใน Master Status register ถูกใช้สำหรับ Single-Step trap ซึ่ง Single-Step bit (bit 8) ที่ถูกเคลียร์มาก่อนหน้านี้แล้วกำลังจะถูกเซต ก่อให้เกิด trap ขึ้นหลังจากปฏิบัติคำสั่งต่อมา ในขณะที่บิทนี้ถูกเซตเป็น 1 ถ้าการปฏิบัติตามคำสั่งก่อให้เกิด trap Single-step trap จะเกิดขึ้นหลังจากการปฏิบัติ trap-handling routine Single-Step Pending bit (bit 9) ถูกใช้โดยโปรเซสเซอร์เพื่อทำให้แน่ใจว่าเกิด Single-Step trap เพียง 1 ครั้งเท่านั้น สำหรับแต่ละคำสั่งที่ถูกปฏิบัติขณะที่ Single-Step bit ถูกเซตเป็น 1

Breakpoint-on-Halt การ trap นี้เกิดขึ้นเมื่อใดก็ตามที่ Breakpoint-on-Halt bit ใน Master Status register เป็น 1 และมีคำสั่ง Halt

การห้ามใช้ Interrupt และ Trap การอินเตอร์รัพท์แบบ maskable สามารถให้ทำงานหรือไม่ให้ทำงานเป็นอิสระกันได้โดยผ่านทางซอฟท์แวร์โดยเซตหรือเคลียร์ค่าที่ control bit ที่เหมาะสมใน Master status register

มี 7 bit บน Master Status register ที่ใช้ระบุว่าจะยอมรับอินเตอร์รัพท์หรือไม่ การร้องขออินเตอร์รัพท์ซึ่งแบ่งเป็นกลุ่ม แต่ละกลุ่มถูกควบคุมให้แยกกันโดย Interrupt Enable control bit โดยแต่ละตัวถูกแสดงเรียงตามลำดับจากที่มีลำดับความสำคัญ (priority) น้อยที่สุดมาหาลำดับความสำคัญมากที่สุด

- ☐ Maskable Interrupt A line (bit 0)
- ☐ Counter/Timer 0, DMA0 (bit 1)
- ☐ Maskable Interrupt B line (bit 2)
- ☐ Counter/Timer1,UART receiver, DMA1 (bit 3)
- ☐ Maskable Interrupt Cline (bit 4)
- ☐ UART Transmitter, DMA2 (bit 5)
- ☐ Counter/Timer2, DMA3 (bit 6)

เมื่อ source ที่ขออินเตอร์รัพท์ถูกดิสเอเบิล CPU จะไม่สนใจการร้องขออินเตอร์รัพท์จาก source นี้

System Stack Overflow Warning trap,Privileged Instruction trap (คำสั่ง I/O ใน user mode),หรือ Extended Instruction trap สามารถถูกทำให้ใช้ได้โดยเซต control bit ใน Trap Control register ส่วน Single-Step และ Breakpoint-on-Halt trap สามารถที่ถูกทำให้ใช้ได้โดยเซต control bit ใน Master Status register trap เหล่านี้เท่านั้นที่สามารถทำให้ใช้ไม่ได้

Interrupt/Trap Vector Table รูปแบบของ Interrupt/Trap Vector Table ประกอบด้วยคู่ของ Master Status register และ Program Counter แต่ละคู่ใช้สำหรับแยกแหล่งเกิดอินเตอร์รัพท์และ trap สำหรับการอินเตอร์รัพท์ภายนอกจะมีการแยก Master Status register word และ Program Counter word ออกจากกัน (สำหรับใช้งาน ถ้าสัญญาณอินพุทไม่ถูกชี้) ถ้าการอินเตอร์รัพท์ภายนอกถูกชี้ตำแหน่ง vector table จะประกอบด้วยค่าใน Program Counter ที่ชี้ตำแหน่งที่เป็นไปได้ 128 ตำแหน่ง โดยใช้ค่าที่ได้มาจากอินพุท ดังนั้นสำหรับ vector interrupt จะมีเพียง Master Status register ตัวเดียวเท่านั้นสำหรับแต่ละชนิดของการอินเตอร์รัพท์

รูปแบบของ Interrupt/Trap Vector Table แสดงได้ดังตารางที่ 1.3

Address (Hexadecimal)	Content
00	Reserved
04	NMI Vector
08	Interrupt Line A Vector (End of Process)
0C	Interrupt Line B Vector
10	Interrupt Line C Vector
14	C-T0
18	C-T1
1C	Reserved
20	C-T2
24	DMA0 Vector
28	DMA1 Vector
2C	DMA2 Vector
30	DMA3 Vector
34	UART Receiver Vector
38	UART Transmitter Vector
3C	Single-Step Trap Vector
40	Breakpoint-on-Halt Trap Vector
44	Division Exception Trap Vector
48	Stack Overflow Warning Trap Vector
4C	Page Fault Trap Vector
50	System Call Trap Vector
54	Privileged Instruction Trap Vector
58	EPU <— Memory Trap Vector
5C	Memory <— EPU Trap Vector
60	A <— EPU Trap Vector
64	EPU Internal Operation Trap Vector
68 - 6C	Reserved
70 - 16E	128 Program Counters for NMI and Interrupt line A Vectors (MSR from 04 and 08 , respectively)
170 - 26E	128 Program Counters for Interrupt Line B Vectors (MSR from 0C)
270 - 36E	128 Program Counters for Interrupt Line C Vectors (MSR from 10)

ตารางที่ 1.3 Interrupt/Trap Vector Table

1.8 โหมดการอ้างแอดเดรส

โหมดการอ้างแอดเดรสถูกใช้โดย CPU เพื่อที่คำนวณหาตำแหน่งประสิทธิผลของโอเปอร์แรนด์ที่ต้องการสำหรับการปฏิบัติตามคำสั่ง มีการอ้างแอดเดรส 9 โหมดที่ใช้งานใน Z280 โดยมีโหมดการอ้างแอดเดรสเพิ่มขึ้นจาก Z80 อยู่ 4 โหมด (Indexed with 16 bit displacement, Stack Pointer Relative,Program Counter Relative และ Base Index) และยังคงรักษาโหมดการอ้างอิงแอดเดรสของ Z80 ทั้ง 5 โหมดอยู่

Register Mode ตัวโอเปอร์แรนด์คือรีจีสเตอร์ 8 bit (A,B,C,D,E,H,L,IXH,IXL,IYH หรือ IYL) หรือรีจีสเตอร์ 16 bit (BC,DE,HL,IX,IY หรือ SP) หรือรีจีสเตอร์พิเศษ (I หรือ R) ตัวใดตัวหนึ่งก็ได้

Immediate Mode ตัวโอเปอร์แรนด์อยู่ในตัวคำสั่งเองและไม่มีการอ้างแอดเดรสประสิทธิผล

Indirect register Mode ข้อมูลของตัวรีจีสเตอร์จะใช้บอกแอดเดรสประสิทธิผลของตัวโอเปอร์แรนด์ รีจีสเตอร์ HL ใช้เป็นรีจีสเตอร์ที่ใช้เข้าถึงหน่วยความจำ (สำหรับคำสั่ง Load To หรือ Load From Accumulator, BC และ DE สามารถที่จะถูกใช้อ้างถึงหน่วยตำแหน่งโดยอ้อมได้ด้วย สำหรับคำสั่ง JP นั้นสามารถใช้ IX และ IY ในการอ้างตำแหน่งโดยอ้อมนี้ได้เช่นกัน) รีจีสเตอร์ C ถูกใช้ในการเข้าถึง I/O และกำหนด control register

Direct address Mode แอดเดรสที่ใช้เป็นโอเปอร์แรนด์นี้คือแอดเดรสที่อยู่ในตัวคำสั่ง ตัวโอเปอร์แรนด์ที่ระบุเป็นได้ทั้งแอดเดรสของ I/O หรือของหน่วยความจำขึ้นอยู่กับคำสั่ง

Indexed Mode แอดเดรสประสิทธิผลของโอเปอร์แรนด์นี้ คือ ตำแหน่งที่ระบุโดยการบวกแอดเดรส 16 bit ที่บรรจุอยู่ในคำสั่งกับค่า index แบบ 2'complement ที่เก็บอยู่ในรีจีสเตอร์ HL,IX หรือ IY

Short Index Mode แอดเดรสประสิทธิผลของตัวโอเปอร์แรนด์ คือ ตำแหน่งที่ถูกคำนวณโดยการบวกเลข 8 bit แบบ 2'complement ที่คิดเครื่องหมาย ซึ่งบรรจุอยู่ในคำสั่ง กับข้อมูลในรีจีสเตอร์ IX หรือ IY การอ้างแอดเดรสโหมดนี้สมมูลกับ indexed mode ใน CPU Z80

Program Counter Relative Mode ค่า displacement ขนาด 8 หรือ 16 bit ที่บรรจุในคำสั่งจะถูกบวกเข้าไปใน Program Counter เพื่อสร้างแอดเดรสประสิทธิผลของตัวโอเปอร์แรนด์

Stack Pointer Relative แอดเดรสประสิทธิผลของโอเปอร์แรนด์ คือ ตำแหน่งที่คำนวณโดยการบวกค่า displacement ขนาด 16 บิท ที่เก็บแบบ 2'complement ซึ่งบรรจุอยู่ในคำสั่ง กับข้อมูลใน Stack Pointer

Base Index Mode แอดเดรสประสิทธิผลของตัวโอเปอร์แรนด์ คือ ตำแหน่งแอดเดรสที่คำนวณโดยการบวกข้อมูลของ HL,IX หรือ IY ตัวใดตัวหนึ่ง กับข้อมูลของรีจีสเตอร์ อีกตัวของรีจีสเตอร์ 3 ตัว นี้

1.9 EXTENDED PROCESSING ARCHITECTURE

คุณสมบัติ

Zilog Extended Processing Architecture (EPA) ทำให้เกิดความยืดหยุ่นสูงและสามารถขยายทั้งฮาร์ดแวร์และซอฟท์แวร์เพื่อขยายความสามารถของ Z280 ลักษณะของ EPA มีดังนี้

- ยอมให้เพิ่มเติมกลุ่มคำสั่งของ Z280 โดยอุปกรณ์ภายนอก
- เพิ่มความสามารถระบบโดยการใช้โปรเซสเซอร์พิเศษต่อภายนอกทำงานขนานกันไปได้ถึง 4 ตัว
- เป็นส่วนที่ออกแบบเพิ่มขยายระบบโดยใช้ฐานจาก Z280 CPU
- จัดการ multiple microprocessor ได้ง่าย โดยผ่านทางการสื่อสารแบบ "single instruction stream"
- สามารถต่อระหว่าง EPU และ Z280 ได้โดยตรง โดยไม่ต้องเพิ่มวงจรตรรกะภายนอก
- สามารถเพิ่มเติม EPUs เพื่อให้ระบบโตขึ้น และ เพิ่มเติม EPUs ที่พัฒนาขึ้นมาทำงานแบบพิเศษ

ลักษณะทั่วไป

ในการประมวลผลของไมโครโปรเซสเซอร์ Zilog Z-BUS Z280 สามารถที่จะเสริมความสามารถโดยปกติได้โดยใช้ส่วน Extended Processing Architecture (EPA) ซึ่ง EPA จะยอมให้ Z280 CPU ต่อระบบเพิ่มกับ Extended Processing Unit (EPU) เพื่อความเหมาะสมได้ถึง 4 ตัวซึ่งแต่ละตัวจะทำงานพิเศษขนานไปพร้อมกับการทำงานของ CPU หลัก

EPUs สามารถต่อโดยตรงกับ Z-BUS และ EPUs จะคอยตรวจดูคำสั่งที่ออกจาก CPU เมื่อพบคำสั่งที่มุ่งทำงานกับ EPU ซึ่งเรียกว่า template แล้ว EPUs จะตอบรับอย่างเหมาะสม โดยจะรับหรือปล่อยข้อมูลหรือสถานะลงบน Z-BUS โดยการใช้สัญญาณควบคุมที่เกิดจาก Z280 และจะกระทำคำสั่งของมันโดยตรง

ตัว CPU จะตอบสนองต่อคำสั่งที่มีต่อ EPU และปล่อยโอเปอร์แรนด์และข้อมูลไปที่ EPU ตัว EPU จะรู้จัก template ที่มันได้รับและจะปฏิบัติตามโดยการใช้ข้อมูลที่ได้รับจาก template และ/หรือ ข้อมูลภายในรีจีสเตอร์ของมันเอง มีการใช้คำสั่งเกี่ยวกับ EPU อยู่ 3 class

- การรับส่งข้อมูลระหว่างหน่วยความจำ กับ EPU register
- การรับส่งข้อมูลระหว่าง MPU register และ EPU status register
- การปฏิบัติการภายในตัว EPU

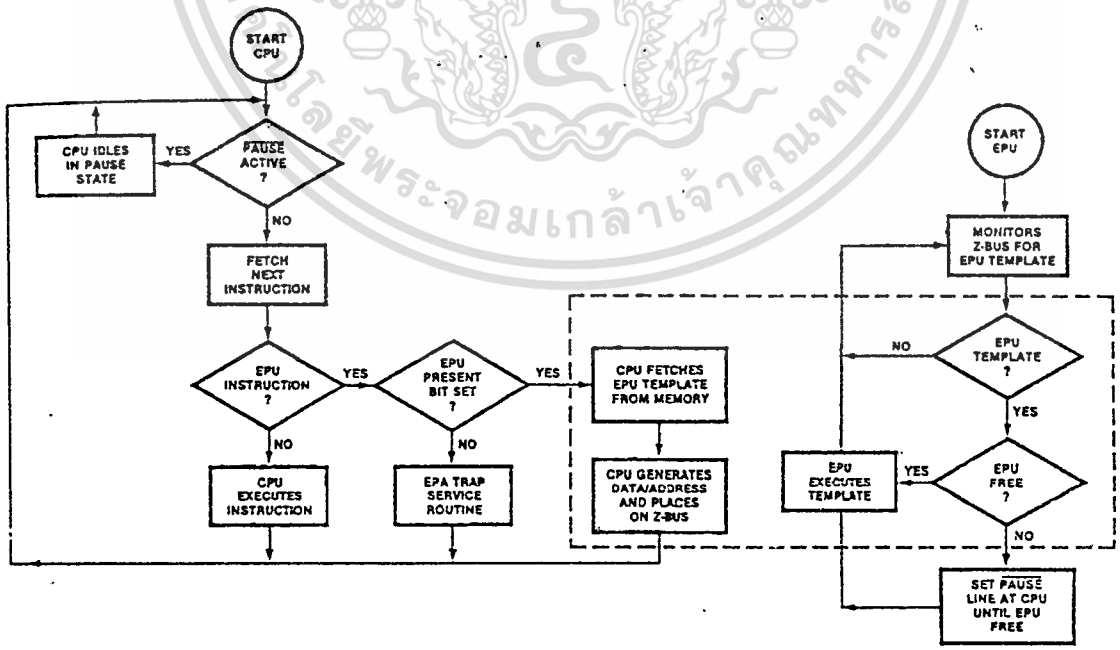
มีการอ้างแอดเดรสอยู่ 6 โหมดที่สามารถใช้กับการรับส่งข้อมูลระหว่าง EPU register , MPU และ หน่วยความจำหลัก

- Indirect Register
- Direct Address
- Indexed

- Program Counter Relative
- Stack Pointer Relative
- Base Index

ความสามารถส่วนที่เพิ่มขึ้นทางฮาร์ดแวร์ของ EPA คือมีกลไกการเกิด extended instruction trap ที่ยอมให้ซอฟท์แวร์เลียนแบบการทำงานของ EPU ได้ EPU present bit ใน Trap Control Register ของ Z280 ระบุว่ามี EPU ต่ออยู่จริงหรือไม่ ถ้าไม่ได้ต่ออยู่ เมื่อ CPU พบคำสั่ง extended แล้วเกิด trap จะทำให้ซอฟท์แวร์ "trap handler" สามารถเลียนแบบการทำงานของ EPU ได้ ดังนั้น EPA software trap routine จะถูกใช้ในระบบที่ไม่มี EPU ต่ออยู่

การปฏิบัติคำสั่งของ CPU และ EPA แสดงได้ดังรูปที่ 1.16 ตัว CPU จะเริ่มทำงานโดยเฟทช์คำสั่งและพิจารณาว่าเป็นคำสั่งของ EPA หรือไม่ ถ้าเป็นคำสั่งของ EPU จะตรวจสอบ EPU Enable bit ใน Trap Control Register ถ้า EPU Enable bit ถูกรีเซต(E=0) CPU จะสร้าง trap และคำสั่งของ EPU สามารถที่จะถูกจำลองการทำงานโดย EPU instruction trap software routine อย่างไรก็ตามถ้า EPU Enable bit ถูกเซต (E=1) แสดงว่ามี EPU ต่ออยู่ในระบบแล้ว EPU template ขนาด 4 byte จะถูกเฟทช์จากหน่วยความจำ การเฟทช์ของ EPU template จะถูกระบุโดยขา status ST_0-ST_3 EPU แต่ละตัวจะตรวจสอบ Z-BUS และที่ขาสถานะอย่างต่อเนื่องเพื่อหา template ของมัน หลังจากมีการเฟทช์ EPU template แล้ว CPU จะมีการส่งข้อมูลที่เหมาะสม(ถ้าจำเป็น)ระหว่าง EPU และหน่วยความจำ หรือระหว่าง CPU และ EPU การติดต่อเหล่านี้ดูได้จากรหัสข้อมูลที่ขา status ถ้า EPU เป็นอิสระเมื่อ template และข้อมูล เกิดขึ้นมา template ของ EPU จะถูกกระทำ แต่ถ้า EPU ยังคงทำการประมวลผลคำสั่งก่อนหน้านี้อยู่ สัญญาณ PAUSE จะเกิดขึ้น เพื่อหยุดการทำงานของคำสั่งต่อไปของ CPU ไว้ก่อนจนกว่า EPU จะปฏิบัติงานเสร็จ หลังจากการปฏิบัติ template สมบูรณ์แล้ว EPU จะไปยกเลิกสัญญาณ PAUSE และคำสั่งของ CPU จะถูกปฏิบัติการต่อไป



รูปที่ 1.16 EPA and Z280 MPU Instruction Execution

1.10 การจัดการหน่วยความจำ

คุณสมบัติ

- การแปลงแอดเดรสแบบไดนามิคอยู่ภายใน chip
- ยอมให้อ้างแอดเดรสของ physical memory ได้ถึง 16M byte
- แบ่งแยกการแปลงแอดเดรสระหว่าง user และ system mode
- ยอมให้ข้อมูลและคำสั่งอยู่ในหน่วยความจำที่แยกพื้นที่กันได้
- ป้องกันการเขียนทับสำหรับแต่ละเพจในหน่วยความจำ
- สามารถใช้หน่วยความจำเสมือนได้

ลักษณะทั่วไป

ไมโครโปรเซสเซอร์ Z280 ได้รวมเอา Memory Manager Unit (MMU) ไว้ภายในซึ่งสามารถแปลง logical address ไปเป็น physical address ได้ โดยอนุญาตให้เข้าถึง physical memory ได้มากกว่า 64K bytes และเตรียมคุณสมบัติป้องกันหน่วยความจำซึ่งจะพบได้ในระบบใหญ่ ถ้าใช้ MMU ตัว CPU จะสามารถอ้าง physical memory ได้ถึง 16M bytes MMU ยังให้กลไกการตรวจจับ ซึ่งจะสร้าง page fault เมื่อเกิดเงื่อนไขที่ผิดพลาด คำสั่งที่ถูกยกเลิกโดย page fault สามารถที่จะถูกเริ่มต้นใหม่ในลักษณะเดียวกันกับระบบที่มีความต้องการหน่วยความจำเสมือน เมื่อเกิดการรีเซตคุณสมบัติต่าง ๆ ของ MMU จะไม่ทำงาน ดังนั้น logical address จะผ่านไปเป็น physical address โดยไม่ถูกแปลงค่าใดๆ

พื้นที่ physical address ถูกขยายโดยการแบ่งเป็นพื้นที่ logical address ขนาด 64 K byte (แต่ละพื้นที่ถูกใช้โดยโปรแกรม) เป็นเพจ แต่ละเพจจะถูกแมป(แปลง)ไปเป็นพื้นที่ physical address ที่ใหญ่กว่าของไมโครโปรเซสเซอร์ Z280 กระบวนการแมปปิ้งทำให้แอดเดรสของซอฟท์แวร์ผู้ใช้ไม่เกี่ยวข้องกับ physical memory ดังนั้นผู้ใช้จึงไม่ต้องคำนึงว่าต้องเก็บข้อมูลไว้ใน physical memory ไหน ขนาดจริงของแต่ละเพจขึ้นอยู่กับ โหมดการแบ่งแยก โปรแกรม/ข้อมูล ทำงานหรือไม่ ถ้าทำงาน แต่ละเพจจะยาว 8K byte และถ้าไม่ทำงานความยาวของเพจจะยาว 4K byte โดยใช้เทคนิคการแมปปิ้งเพจ logical address ขนาด 16 bit สามารถแปลงเป็น physical address ขนาดยาว 24 bit ได้ การแปลงแอดเดรสสามารถเกิดได้ทั้งโหมด system และ user ซึ่งทั้งสองโหมดก็สามารถแบ่งแยกระหว่างโปรแกรมกับข้อมูลได้โดยง่าย MMU ยอมให้การอ้างอิงคำสั่ง แบ่งแยกกับ การอ้างอิงข้อมูล ซึ่งทำให้โปรแกรมที่มีขนาดถึง 64K byte สามารถจัดการกับข้อมูลที่มีขนาดได้ถึง 64 K byte ได้โดยไม่ต้องให้ระบบปฏิบัติการเข้ามาช่วยเหลือ

สถาปัตยกรรมของ MMU

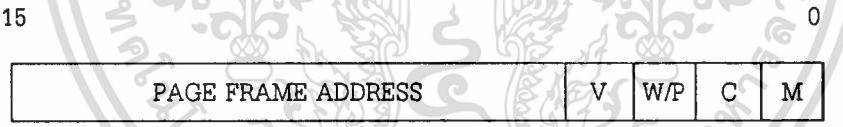
Z280 ประกอบด้วยรีจีสเตอร์ต่างๆ ดังต่อไปนี้ กลุ่มของ Page Description Register ซึ่งใช้แปลงแอดเดรสอยู่ 2 กลุ่ม โดยที่แต่ละกลุ่มมีรีจีสเตอร์ดังรูป1.17 ถึง 16 ตัว , รีจีสเตอร์ขนาด 16 bit ที่ควบคุมการแปลง , Page Description Register Pointer , พอร์ท I/O ที่เขียนได้อย่างเดียวซึ่งสามารถทำให้ไม่ใช้ค่าในกลุ่ม page description ในบางกลุ่มได้ และ พอร์ท I/O 2 พอร์ท ซึ่งใช้

กำหนดค่าของ Page Description Register Page Description Register กลุ่มหนึ่งใช้กับการทำงานใน system mode และอีกกลุ่มทำงานใน user mode

ในขณะที่แอดเดรสกำลังถูกแปลง attribute จะมีความสัมพันธ์กับ logical page ซึ่งตำแหน่งนั้นถูกตรวจสอบ logical page ที่ถูกต้องจะพิจารณาโดย CPU mode (user หรือ system) ,address space (โปรแกรม/ข้อมูล) และ 4 bit ที่มีนัยสำคัญสูงสุดของ logical address แต่ละเพจสามารถป้องกันการเขียน เพื่อป้องกันตัวเองจากการถูกแก้ไขโดยงานที่กำลังปฏิบัติอยู่ และสามารถที่จะกำหนดเป็น non-cacheable เพื่อป้องกันไม่ให้เขียนข้อมูลลง cache เพื่อใช้ในภายหลัง ความสามารถต่อมาคือการใช้ประโยชน์ระบบ multiprocessor เพื่อที่จะแน่ใจว่า processor สามารถเข้าถึงข้อมูลล่าสุดที่เกิดจากการใช้อุปกรณ์ร่วมกัน ในแต่ละเพจ ของ MMU จะมีบิทอยู่บิทหนึ่งที่ระบุว่าเพจนั้นได้ถูกแก้ไขหรือไม่

Page Description Register แต่ละตัวจะมี Valid bit เพื่อใช้บอกว่า Page Descriptor Register เก็บข้อมูลที่ต้องหรือไม่ ความพยายามใดๆ ของ MMU ที่จะแปลงแอดเดรสในส่วน Page Descriptor Register ที่ไม่ได้เก็บข้อมูลถูกต้อง จะเกิด page fault ขึ้น Valid bit สำหรับกลุ่มของ Page Descriptor Register สามารถที่จะรีเซตโดยการเขียนไปที่ MMU control port

ในแต่ละโหมดการทำงานของ CPU MMU สามารถที่จะแยกการเฟทช์คำสั่ง จากการเฟทช์ข้อมูล และแยกพื้นที่ program address จาก data address เมื่อโหมดการแยกระหว่าง โปรแกรม/ข้อมูล ในโหมดการทำงานปัจจุบันของ CPU (user หรือ system) Page Descriptor Register ทั้ง 16 ตัวจะถูกแบ่งเป็น 2 กลุ่ม กลุ่มหนึ่งสำหรับการเฟทช์คำสั่งและอีกกลุ่มสำหรับการเฟทช์ข้อมูล การเฟทช์คำสั่งหรือการเข้าถึงข้อมูลทำได้ได้โดยการใช้ การอ้างแอดเดรสในโหมด Program Counter Relative ซึ่งจะถูกแปลงโดยรีจีสเตอร์ MMU ซึ่งจะเกี่ยวข้องกับพื้นที่ในส่วน program address ส่วนข้อมูลจะเข้าถึงโดยใช้การอ้างแอดเดรสโหมดอื่น และเข้าถึง Interrupt Vector Table ในอินเตอร์รัพท์โหมด 2 โดยใช้รีจีสเตอร์ MMU ที่เกี่ยวข้องกับพื้นที่ในส่วน data address การทำงานของ MMU ในโหมดนี้ขนาดของเพจคือ 8192 byte ใน MMU Master Control Register จะมี control bit อยู่ 2 bit เป็นอิสระกัน ไม่ว่าการทำงานของ MPU ใน user และ system mode มีการแบ่งแยกพื้นที่โปรแกรมและข้อมูลหรือไม่ก็ตาม



รูปที่ 1.17 Page Descriptor Register

Page Descriptor Register แต่ละตัวประกอบด้วย attribute 4 bit และ page frame address 12 bit attribute จะอยู่ที่ 4 bit ที่มีนัยสำคัญต่ำสุด และมีบิตควบคุมและสถานะดังนี้

Modified (M) บิทนี้จะถูกเซตโดยอัตโนมัติ เมื่อ logical address ในเพจนี้ถูกเขียนสำเร็จ บิทนี้สามารถเคลียร์เป็นศูนย์ โดยการใช้ software routine โหลดค่า Page Descriptor Register ถ้า Valid bit เป็น 0 บิทนี้ก็จะไม่มีความหมาย

Cacheable (C) ขณะที่บิทนี้เป็น 1 ข้อมูลที่อ่านจากเพจนี้สามารถเก็บลงในแคชได้ แต่ถ้าบิทนี้เป็น 0 กลไกการควบคุมแคชจะยับยั้งไม่ให้มีการเก็บข้อมูลลงแคช

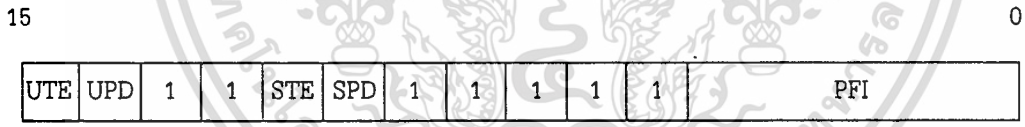
Write-Protect (WP) ขณะที่บิทนี้ถูกเซตเป็น 1 ถ้า CPU จะเขียนข้อมูลลงใน logical address ที่อยู่ในเพจนี้ จะเกิด page fault ขึ้น ทำให้ป้องกันการเขียนเกิดขึ้น ถ้าบิทนี้เคลียร์ตัวเองเป็น 0 การเข้าถึงที่ถูกต้องทั้งหมดจะถูกยอมรับ

Valid (V) ถ้าบิทนี้เซตเป็น 1 Page Descriptor Register จะบรรจุข้อมูลที่ถูกต้อง ถ้าบิทนี้ถูกเคลียร์เป็น 0 การเข้าถึง logical address ของ CPU ทั้งหมดที่อยู่ในเพจนี้จะเกิด page fault

MMU Control Register และ I/O Port

การทำงานของ MMU ถูกควบคุมโดย control Register ตัวหนึ่งและ I/O พอร์ทเฉพาะ 4 ตัว MMU Master Control Register (รูปที่ 1.18) จะกำหนดการแบ่งแยกพื้นที่ address ระหว่าง โปรแกรม/ข้อมูล ทั้งใน user และ system mode และไม่ว่า logical address จะเกิดในโหมด user หรือ system ก็จะถูกแปลงโดย MMU การเข้าถึง Page Descriptor Register จะเข้าผ่านทาง Register address ที่อยู่ใน Page Descriptor Register Pointer ส่วน Descriptor Select Port ถูกใช้เข้าถึง Page Descriptor Register ที่ถูกชี้โดย Page Descriptor Register Pointer หลังจากเข้าถึงแล้ว Page Descriptor Register Pointer จะยังไม่เปลี่ยนแปลงค่า คำสั่ง Block Move I/O Port ถูกใช้ย้ายบล็อกของ word ระหว่าง Page Descriptor Register กับหน่วยความจำ เพื่อเขียนหรืออ่าน I/O พอร์ทซึ่งเข้าถึงข้อมูลที่ถูกชี้โดย Page Descriptor Register Pointer เสร็จแล้วเพิ่มค่าของตัวชี้ไปอีกหนึ่ง Invalidation I/O Port ใช้ยกเลิกการใช้ block ของ Page Descriptor Register ดังนั้นการเขียนออกพอร์ทนี้ จะทำให้ Valid bit ใน block ที่ถูกเลือกของ Page Descriptor Register ถูกเคลียร์เป็น 0 ซึ่งจะแสดงว่า Page descriptor Register นี้ ไม่มีข้อมูลที่ถูกต้องต่อไปแล้ว

MMU Master Control Register (I/O address ตำแหน่ง FFxxF0) ควบคุมการทำงานของ MMU รีจีสเตอร์นี้ประกอบด้วย control bit 4 bit ส่วนบิทอื่นๆ ในรีจีสเตอร์นี้ต้องถูกเคลียร์เป็น 0 control bit ทั้ง 4 ของ MMU Master Control Register มีดังนี้



รูปที่ 1.18 MMU Master Control Register

Page Fault Identifier (PFI) ข้อมูลทั้ง 5 บิทที่ถูกเก็บไว้นี้จะแสดงว่า Page Descriptor Register ไหนกำลังถูกใช้เมื่อมีการเข้าถึงที่ละเมิดการป้องกัน

System Mode Program/Data Separation Enable (SDP) ในขณะที่บิทนี้เซตเป็น 1 การเฟทช์คำสั่งและเข้าถึงข้อมูลที่ใช้การอ้างแอดเดรสแบบ PC Relative จะใช้ system mode Page Descriptor 8-15, และการอ้างอิงข้อมูลที่ไม่ได้ใช้การอ้างแอดเดรสแบบ PC Relative จะใช้ system mode Page Descriptor Register 0-7 ในขณะที่บิทนี้เคลียร์เป็น 0 system mode Page Descriptor Register 0-15 ถูกใช้แปลงคำสั่งและอ้างอิงข้อมูล

System Mode Translate Enable (STE) ในขณะที่บิทนี้เซตเป็น 1 logical address ที่เกิดใน system mode จะถูกแปลงแอดเดรส ถ้าบิทนี้เคลียร์เป็น 0 แอดเดรสที่ผ่าน MMU จะขยายบิทที่เหลือที่มีนัยสำคัญสูงกว่าเป็น 0 และไม่มีการตรวจสอบ attribute หรือแก้ไขบิทที่ถูกกระทำ

User Mode Program/Data Space Separation Enable (UPD) ในขณะที่บิทนี้เซตเป็น 1 การเฟทช์คำสั่งและเข้าถึงข้อมูลที่ใช้การอ้างแอดเดรสแบบ PC Relative จะใช้ user mode Page Descriptor Register 8-15 และการอ้างอิงข้อมูลที่ไม่ได้ใช้การอ้างแอดเดรสแบบ PC Relative จะใช้ user mode Page Descriptor register 0-7 ในขณะที่บิทนี้เคลียร์เป็น 0 user mode Page Descriptor Register 0-15 ถูกใช้แปลงคำสั่งและการอ้างอิงข้อมูล

User Mode Translated Enable (UTE) ในขณะที่บิทนี้เซตเป็น 1 logical address ที่เกิดใน user mode จะถูกแปลงแอดเดรส ในขณะที่บิทนี้ถูกเคลียร์เป็น 0 แอดเดรสที่ผ่าน MMU จะขยายบิทที่เหลือที่มีนัยสำคัญสูงกว่าเป็น 0 และไม่มีการตรวจสอบ attribute หรือแก้ไขบิทที่ถูกกระทำ

Page Descriptor Register Pointer การเคลื่อนย้ายของข้อมูลเข้าหรือออกจาก MMU Page Descriptor Register จะใช้ Page Descriptor Register Pointer ซึ่งมีตำแหน่ง I/O address FFxxF1 โดยรีจีสเตอร์ขนาด 8 บิทตัวนี้จะเก็บค่าแอดเดรสตัวใดตัวหนึ่งของ Page Descriptor Register ไว้ เมื่อใช้คำสั่ง Word I/O เพื่อเข้าถึง I/O address FFxxF5 (Descriptor Select Port) รีจีสเตอร์ตัวนี้ถูกใช้เข้าถึง Page Descriptor Register เมื่อใช้คำสั่ง word I/O เข้าถึง I/O address FFxxF4 (Block Move I/O Port) รีจีสเตอร์ตัวนี้ถูกใช้เข้าถึง Page Descriptor Register ด้วย แต่หลังจากการเข้าถึงแล้ว รีจีสเตอร์ตัวนี้จะเพิ่มค่าตัวเองขึ้นหนึ่งค่าโดยอัตโนมัติ

Descriptor Select Port การเคลื่อนย้ายข้อมูลหนึ่ง word เข้าหรือออกจาก Page Descriptor Register ทำได้โดยเขียนหรืออ่าน word จาก I/O พอร์ทตำแหน่ง FFxxF5 คำสั่ง word I/O ใด ๆสามารถที่จะเข้าถึง Page Descriptor Register โดยผ่านทางพอร์ทนี้ โดยที่ในขณะนั้นได้เตรียมค่าใน Page Descriptor Register Pointer ที่เหมาะสมแล้ว

Block Move I/O Port การเคลื่อนย้ายข้อมูลเป็น block เข้าหรือออกจาก Page Descriptor Register ถูกทำได้โดยการเขียนหรืออ่าน word ไปที่ I/O พอร์ทเฉพาะที่ตำแหน่ง FFxxF4 คำสั่ง word I/O ใดๆ สามารถเข้าถึง Page Descriptor register ผ่านทางพอร์ทนี้ โดยที่ในขณะนั้นได้เตรียมค่าใน Page Descriptor Register Pointer ที่เหมาะสมแล้ว

Invalidation I/O Port Valid bit สามารถถูกเคลียร์ได้ (เช่น ค่าใน Page Descriptor Register ถูกยกเลิก) โดยการเขียนไปที่พอร์ทเฉพาะที่มีขนาด 8 บิทนี้ (ตารางที่ 1.4) โดยมีตำแหน่ง I/O address FFxxF2 Valid bit แต่ละตัวสามารถถูกเซตค่าในภายหลังโดยการเขียนจากซอฟท์แวร์ไปที่ Page Descriptor Register การอ่านค่าจาก I/O พอร์ทนี้จะได้ค่าที่ไม่สามารถคาดเดาได้

Encoding	Registers Invalid
01_H	System Page Descriptor Register 0-7
02_H	System Page Descriptor Register 8-15
03_H	System Page Descriptor Register 0-15
04_H	User Page Descriptor Register 0-7
08_H	User Page Descriptor Register 8-15
$0C_H$	User Page Descriptor Register 0-15

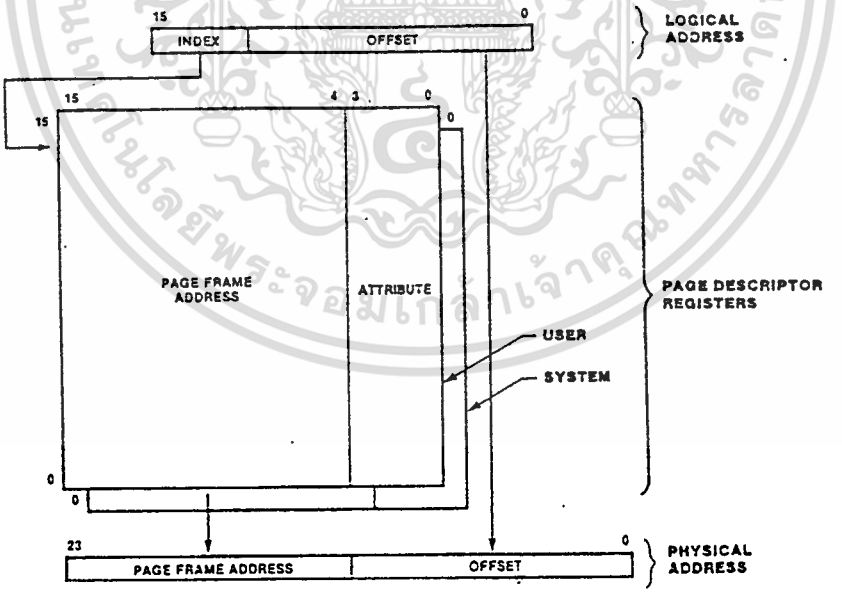
ตารางที่ 1.4 Invalidation Port Table

1.11 กลไกการแปลงแอดเดรส

Address Translation Without Program/Data Separation

การแปลงแอดเดรสแสดงให้เห็นได้ในรูปที่ 1.19A ขณะที่ Program/Data Space Separation bit ถูกเคลียร์เป็น 0 logical address ขนาด 16 bitจะถูกแบ่งเป็น 2 กลุ่ม โดยกลุ่มแรกซึ่งมี 4 บิทใช้เป็นดัชนี (index) เลือก Page Descriptor Register หนึ่งเพจจากทั้งหมด 16 page และอีกกลุ่มซึ่งมี 12 บิท ใช้เป็น offset เพื่อกำหนด 12 bit ต่ำสุดของ physical address ดังนั้น physical address จะประกอบด้วย page frame address (bit 4-15) ที่ได้มาจาก Page Descriptor register รวมกับ 12 bit offset ที่ได้มาจากค่า offset ของ logical address

ขนาดเพจที่ได้จะมีขนาด 4 K bytes ดังนั้น Page descriptor register 0 จะมี logical address ตั้งแต่ 0000_H ถึง $0FFF_H$ ส่วน Page descriptor register 1 จะมี logical address ตั้งแต่ 1000_H ถึง $1FFF_H$ และเป็นอย่างนี้ไปเรื่อยๆ จน Page descriptor register 15 จะมี logical address ตั้งแต่ $EFFF_H$ ถึง $FFFF_H$



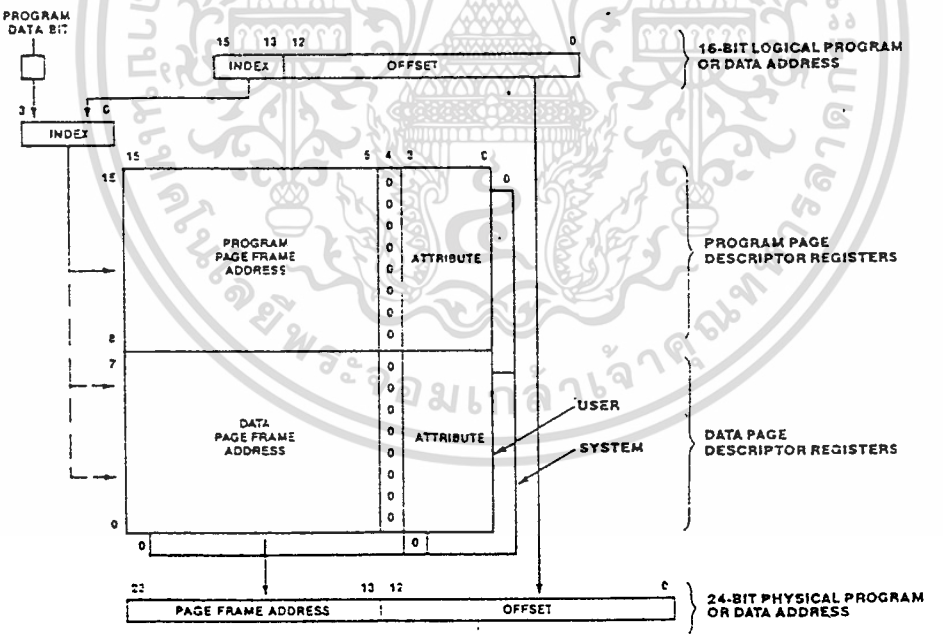
รูปที่ 1.19A Address Translation without Program/Data Separation

Address Translation Without Program/Data Separation

การแปลงตำแหน่งแอดเดรส สามารถแสดงให้เห็นดังรูปที่ 1.19B ในขณะที่ Progam/Data Space bit ถูกเซตเป็น 1 logical address จะถูกแบ่งเป็นกลุ่มดัชนี (index) ขนาด 3 บิท กับ กลุ่ม offset ที่มีขนาด 13 บิท โดยที่ Page Descriptor Register จะประกอบด้วยค่า Page Frame Address field ขนาด 11 บิท (bit 5-15 โดยที่ bit 4 = 0) physical address คือ ผลที่เกิดจากการรวมกันของ page frame address และ logical offset ส่วน Page Descriptor Register ถูกเลือกโดยใช้ค่าดัชนี (index) ขนาด 4 บิท ที่ประกอบด้วย Program/Data Address bit 1 บิทจากซีพียู และ Index bit 3 บิทจาก logical address

ขนาดเพจที่ได้จะมีขนาด 8 K bytes ดังนั้น Page descriptor register 0 จะมี logical address ตั้งแต่ 0000_H ถึง $1FFF_H$ ในส่วน data address Page descriptor register 1 จะมี logical address ตั้งแต่ 2000_H ถึง $3FFF_H$ ในส่วน data address เช่นกันและเป็นอย่างนี้ไปเรื่อยๆ จน Page descriptor register 7

Page descriptor registor 8 จะมี logical address ตั้งแต่ 0000_H ถึง $1FFF_H$ ในส่วน program address Page descriptor register 9 จะมี logical address ตั้งแต่ 2000_H ถึง $3FFF_H$ ในส่วน program address เช่นกันและเป็นอย่างนี้ไปเรื่อยๆ จน Page descriptor register 15



รูปที่ 1.19A Address Translation with Program/Data Separation

1.12 ON-CHIP MEMORY

คุณสมบัติ

- 256 byte local memory
- สามารถกำหนดให้เป็นแคชความเร็วสูง
- สามารถโปรแกรมให้เป็นแคช สำหรับคำสั่ง,สำหรับข้อมูล หรือทั้งสำหรับคำสั่งและข้อมูล
- มีการทำงานที่เร็วกว่าโดยมีการใช้บัสภายนอกให้น้อยที่สุด
- ผู้ใช้ใช้ได้ง่าย
- สามารถกำหนดให้เป็นหน่วยความจำภายในที่ผู้ใช้กำหนดแอดเดรสได้เอง

ไมโครโปรเซสเซอร์ Z280 มีหน่วยความจำอยู่ภายใน 256 byte ซึ่งสามารถใช้เป็นหน่วยความจำตำแหน่งใดๆ ที่ระบบกำหนดได้ หรือใช้เป็นแคชสำหรับคำสั่งหรือข้อมูล โหมดการทำงานสามารถโปรแกรมได้ (ให้เป็นหน่วยความจำ หรือ แคช) เมื่อรีเซต มันจะทำงานเป็นแคชสำหรับคำสั่งอย่างเดียวเท่านั้นโดยอัตโนมัติ

สถาปัตยกรรมของหน่วยความจำภายใน

หน่วยความจำภายในมีโครงสร้างเป็น 16 แถวๆ ละ 16 byte แต่ละแถวสามารถที่จะ copy ค่าทั้ง 16 byte ที่เรียงกันมาของ physical address ในตำแหน่งที่มี 20 บิทที่มีนัยสำคัญสูงสุดเหมือนกัน แต่ละ byte ในแคชจะมี Valid bit ที่แสดงว่าแคชได้เก็บข้อมูลที่เหมือนกับในหน่วยความจำที่ตำแหน่ง physical memory หรือไม่ รูปที่ 1.20 แสดงถึงโครงสร้างของ cache

	20 BITS	16 BITS	16 x 8 BITS
LINE 0	TAG 0	VALID BITS	CACHE DATA
LINE1	TAG 1	VALID BITS	CACHE DATA
LINE2	TAG 2	VALID BITS	CACHE DATA
•	•	•	•
•	•	•	•
•	•	•	•
LINE15	TAG 15	VALID BITS	CACHE DATA

Tag n = แอดเดรส 20 บิทที่สัมพันธ์กับแถว n

Valid bits = 16 บิทที่แสดงว่าค่าในแต่ละไบท์ของแคชยังเก็บค่าที่ถูกต้อง

Cache data = 16 ไบท์

รูปที่ 1.20 Cache Organization

หน่วยความจำภายในมีการทำงานอยู่ 2 โหมด ถ้า Memory/Cache bit ใน Cache Control Register ถูกเซตเป็น 1 แล้ว หน่วยความจำภายในทั้ง 256 byte จะถูกปฏิบัติเช่นเดียวกับตำแหน่งใน physical memory หน่วยความจำจะเข้าถึงแอดเดรสที่ครอบคลุมหน่วยความจำภายในโดยไม่ต้องติดต่อกับบัสภายนอก ดังนั้นการเข้าถึงหน่วยความจำส่วนนี้จะเร็วกว่า ในโหมดนี้จะไม่สนใจ Valid bit

ถ้า Memory/Cache bit ถูกเคลียร์เป็น 0 แล้วหน่วยความจำภายในทั้ง 256 byte จะถูกทำเป็น cache memory แต่ละแถวจะใช้อัลกอริทึม least-recently used (LRU) โดยเมื่อแคชอ่านข้อมูลไม่พบ (และ MMU ไม่ขัดขวางการใช้แคช) แคชในแถวที่มีการเข้าถึงข้อมูลน้อยที่สุดจะถูกเลือกให้เก็บข้อมูลใหม่ ทุกไบท์ในแถวที่ถูกเลือกจะถูกทำเครื่องหมายยกเลิกไว้ ยกเว้นในไบท์ที่เก็บข้อมูลใหม่ เมื่อแคชอ่านข้อมูลไม่พบ 1 หรือ 2 byte (ขึ้นอยู่กับขนาดของbus) ข้อมูลจะถูกเฟทช์จากหน่วยความจำหลัก ยกเว้นสำหรับการเฟทช์คำสั่งใน burst mode ซึ่งแคชจะไม่ pre-fetch หรืออ่านข้อมูลเกินไปกว่าแอดเดรสที่ต้องการในปัจจุบัน การที่ปัจจุบันแคชไม่มีตำแหน่งของข้อมูลที่จะเขียนนั้น จะไม่ทำให้มีการติดต่อกับหน่วยความจำในตำแหน่งนั้นแต่อย่างใด

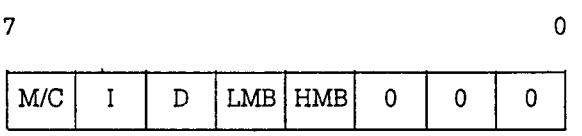
แคชสามารถเก็บได้ทั้งคำสั่งและข้อมูล Control bit 2 บิทใน Cache Control Register สามารถเซตให้แคชเก็บคำสั่งและเก็บข้อมูลได้ โดยไม่ขึ้นต่อกัน ถ้าแคชเก็บข้อมูล การเขียนข้อมูลในตำแหน่งที่บรรจุอยู่ในแคช จะทำให้มีการติดต่อบัสภายนอกเพื่อแก้ไขข้อมูลของหน่วยความจำในตำแหน่งนั้นด้วย

ทั้ง CPU และ DMAs ภายในเข้าถึงแคชได้ สำหรับ CPU ถ้า MMU ทำงาน การเข้าถึงแคชเป็นได้ทั้ง cacheable หรือ non-cacheable ขึ้นอยู่กับค่าของ Cacheable bit ใน Page Descriptor Register ที่ถูกใช้แปลง logical address ถ้า MMU ไม่ทำงาน การติดต่อหน่วยความจำทั้งหมดจะถือว่าเป็นแบบ cacheable ค่า control bit 2 บิทใน Cache Control Register คือ Cache Instruction Disable bit และ Cache Data Disable bit จะใช้บ่งบอกการทำงานของแคชในเงื่อนไขต่างๆ แต่ละบิทเหล่านี้เป็นตัวกำหนดการทำงานของแคชสำหรับคำสั่งหรือสำหรับข้อมูล

เมื่อหน่วยความจำภายในถูกใช้เป็นตำแหน่งในหน่วยความจำที่คงที่ ทั้ง Cache Instruction Disable หรือ Cache Data Disable bit จะไม่ถูกใช้และไม่มีการแบ่งแยกระหว่างการเข้าถึงข้อมูลหรือคำสั่ง

โดยทั่วไปเมื่ออุปกรณ์ต่างๆ เช่น on-chip DMAs เคลื่อนย้ายข้อมูลไปที่หน่วยความจำ ข้อมูลบนแคชจะถูกแก้ไข ถ้าเขียนสู่ตำแหน่งที่ถูกต้องในแคช แต่กลไก LRU จะไม่มีผล สำหรับการเคลื่อนย้ายข้อมูลจาก EPU ไปหน่วยความจำ ถ้าแคชเก็บตำแหน่งที่ถูกใช้โดย EPU แล้ว on-chip cache จะถูกแก้ไขด้วย

Cache Control register การทำงานของหน่วยความจำภายในถูกควบคุมโดย Cache Control Register (รูปที่ 1.21) ซึ่งเป็นรีจิสเตอร์ขนาด 8 bit ที่เข้าถึงได้โดยการใช้คำสั่ง load control รีจิสเตอร์นี้มี control bit อยู่ 5 bit ส่วนบิทอื่นๆต้องถูกเคลียร์เป็น 0



รูปที่ 1.21 Cache Control register

แต่ละ bit ในรีจิสเตอร์นี้คือ

High Memory Burst Capability (HMB) เป็นบิทที่ระบุว่ามี memory burst transaction เกิดขึ้นหรือไม่เมื่อ MMU ทำงาน และบิทที่ 15 ของ Page Descriptor register ที่ถูกเลือกเป็น 1 (0 = ไม่รองรับ burst mode , 1 = รองรับ burst mode)

Low Memory Burst Capability (LMB) เป็นบิทที่ระบุว่ามี memory burst transaction เกิดขึ้นหรือไม่ เมื่อ MMU ไม่ทำงานหรือเมื่อ MMU ทำงานแต่บิทที่ 15 ของ Page Descriptor register ที่ถูกเลือกเป็น 0 (0 = ไม่รองรับ burst mode , 1 = รองรับ burst mode)

Cache Data Disable (D) ในขณะที่บิทนี้ถูกเคลียร์เป็น 0 และ M/C bit เป็น 0 (cache mode) การเฟทช์ข้อมูลจะถูก copy ลงไปในแคช แต่ถ้า M/C bit เป็น 1 ค่าของบิทนี้จะไม่ถูกสนใจ

Cache Instructions Disable (I) ในขณะที่บิทนี้ถูกเคลียร์เป็นศูนย์ และ M/C bit เป็น 0 (cache mode) การเฟทช์คำสั่งจะถูก copy ลงไปในแคช แต่ถ้า M/C bit เป็น 1 ค่าของบิทนี้จะไม่ถูกสนใจ

Memory/Cache (M/C) ในขณะที่บิทนี้ถูกเซตเป็น 1 หน่วยความจำภายในนี้จะถูกใช้เป็น physical memory ในขณะที่บิทนี้ถูกเคลียร์เป็น 0 หน่วยความจำส่วนนี้จะถูกใช้เป็นแคช

ถ้าหน่วยความจำภายในนี้ถูกใช้เป็นตำแหน่งในหน่วยความจำคงที่ ผู้ใช้สามารถโปรแกรมเลือกย่านแอดเดรสของหน่วยความจำ สำหรับหน่วยความจำภายในนี้

1.13 CLOCK OSCILLATOR

ไมโครโปรเซสเซอร์ Z280 มีวงจร clock oscillator/generator อยู่ภายใน ซึ่งสามารถต่อขนานกับคริสตอล เป็นวงจรเรโซแนนท์ หรือจะต่อกับแหล่งกำเนิดสัญญาณนาฬิกาที่เหมาะสมใดๆ ก็ได้ สัญญาณนาฬิกาที่สร้างจากวงจร oscillator ภายในนี้ ใช้กับระบบที่เหลือทั้งหมด ความถี่ของสัญญาณนาฬิกาของโปรเซสเซอร์จะเท่ากับ ครึ่งหนึ่งของความถี่พื้นฐานของคริสตอลหรือสัญญาณนาฬิกาภายนอก

1.14 REFRESH

ไมโครโปรเซสเซอร์ Z280 มีกลไกภายในสำหรับการรีเฟรชหน่วยความจำไดนามิก ซึ่งสามารถทำให้กลไกนี้ทำงานได้โดยการเซต Refresh Enable bit ใน Refresh Rate register ให้เป็น 1 อัตราการรีเฟรชหน่วยความจำสามารถกำหนดโดย Refresh Rate register การจัดการรีเฟรชมีลักษณะเดียวกับการจัดการหน่วยความจำ ยกเว้นสัญญาณ status ที่แตกต่างกัน และไม่มีการเคลื่อนย้ายข้อมูล การรีเฟรชสามารถแทรกเกิดได้ทันทีหลังจากสัญญาณนาฬิกาลูกสุดท้ายของการจัดการ bus ใดๆ รวมถึงการปฏิบัติภายใน

การรีเฟรสจะเกิดขึ้นทันทีที่เป็นไปได้ หลังจากคาบเวลาการรีเฟรสผ่านไปแล้ว (โดยทั่วไปหลังจากสัญญาณนาฬิกาลูกสุดท้ายของการจัดการ bus ปัจจุบัน) ถ้าไมโครโปรเซสเซอร์รับสัญญาณการร้องขออินเตอร์รัพท์ การรีเฟรสจะถูกกระทำเป็นอันดับแรก เมื่อไมโครโปรเซสเซอร์ Z280 ไม่ได้ควบคุมบัส หรืออยู่ในภาวะ Wait state วงจรภายในจะบันทึกจำนวนคาบเวลาของการรีเฟรช แต่ไม่สามารถรีเฟรชได้ เมื่อไมโครโปรเซสเซอร์ควบคุมบัสอีกครั้ง หรือสัญญาณ WAIT ไม่แอคทีฟ และการจัดการบัสสมบูรณ์ กลไกการรีเฟรชจะส่งจำนวนการรีเฟรชที่ได้ข้ามไปออกมาทันทีทันใด วงจรภายในสามารถบันทึกการข้ามรีเฟรชไปได้ถึง 256 ค่า

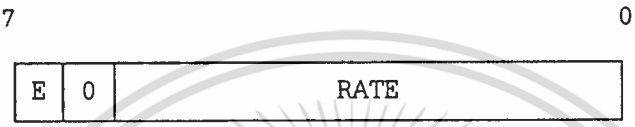
แอดเดรสที่ถูกรีเฟรช 10 บิทจะเกิดขึ้นมาสำหรับการปฏิบัติการรีเฟรชในแต่ละครั้ง โดยจะเพิ่มแอดเดรสที่รีเฟรชขึ้น 2 ค่า สำหรับ data bus ขนาด 16 bit และเพิ่มแอดเดรสที่รีเฟรชขึ้น 1 ค่าสำหรับ data bus ขนาด 8 bit

เมื่อมีการรีเซต Refresh Rate Register จะมีค่า 88_H การรีเฟรชจะทำงาน ด้วยอัตรา 32 ไซเกิลของสัญญาณนาฬิกาของโปรเซสเซอร์และแอดเดรสที่รีเฟรชจะไม่มีผล

กลไกการรีเฟรชถูกควบคุมโดยรีจีสเตอร์ควบคุม 8 bit ดังต่อไปนี้

Refresh Rate register

รีจีสเตอร์ขนาด 8 บิท ตัวนี้ (รูปที่ 1.22) ทำให้กลไกการรีเฟรชทำงานและระบุความถี่ของการรีเฟรช แต่ละบิทในรีจีสเตอร์นี้ประกอบด้วย



รูปที่ 1.22 Refresh Rate register

Refresh (Rate) :กลุ่มนี้ใช้ระบุอัตราการรีเฟรชเป็นไซเกิลสัญญาณนาฬิกาของโปรเซสเซอร์ ค่า n ในกลุ่มนี้ระบุคาบเวลาของการรีเฟรชเป็น 4n โดยที่ Rate = 0 ระบุว่าเป็น 256 clock cycle

Refresh Enable (E) เมื่อบิทนี้ถูกเซตเป็น 1 กลไกการรีเฟรชจะทำงาน

1.15 UART

UART ของ Z280 ส่งและรับข้อมูลอนุกรมโดยใช้โปรโตคอลการสื่อสารข้อมูลแบบไม่ประสาน (asynchronous data communication protocol) แบบพื้นฐาน

การส่งและการรับข้อมูลสามารถทำได้ทั้งแบบ ห้า,หก,เจ็ดหรือแปด บิทต่อตัวอักษร รวมกับพาริตี้คู่หรือคี่ ตัวส่งสามารถใส่ stop bit หนึ่งหรือสองบิท และสามารถให้ break output ที่เวลาใดๆ การรับสัญญาณถูกป้องกันการรบกวนโดยกลไก "transient spike-rejection" ซึ่งจะตรวจสอบสัญญาณหลังจากตรวจพบสัญญาณ Low level ต่อไปอีกครึ่งบิท ถ้าสัญญาณ Low level ไม่คงอยู่ (ในกรณีของการเกิด transient) กระบวนการสร้างตัวอักษร จะไม่ถูกเริ่มต้น framing error และ overrun ถูกตรวจพบและจะเก็บ character บางส่วนซึ่งเกิดขึ้น ยิ่งกว่านั้น กระบวนการตรวจสอบภายในหลีกเลี่ยงที่จะแปล framing error เป็น start bit ใหม่ framing error เป็นผลลัพธ์ในการรวมหนึ่งส่วนสองต่อเวลาแต่ละบิทไปที่จุดซึ่งการค้นหาสำหรับ start bit ต่อไปถูกเริ่มต้น

UART ใช้ความถี่สัญญาณนาฬิกาเดียวกันสำหรับทั้งตัวส่งและตัวรับ สัญญาณอินพุทสำหรับวงจรนาฬิกาแบบ UART ได้รับมาจาก counter/timer 1 ไม่สัญญาณนั้นจะมาจากสัญญาณนาฬิกาภายนอกหรือจากเอาท์พุทของ counter/timer ที่เกิดจากสัญญาณนาฬิกาภายในโปรเซสเซอร์ สัญญาณนาฬิกาที่เข้า UART จะถูกหารความถี่ลง 1,16,32,หรือ64 เท่าทั้งตัวส่งและตัวรับ

DMA 2 ช่องสามารถใช้เป็นอิสระในการเคลื่อนย้าย character ระหว่างหน่วยความจำและตัวส่งหรือตัวรับ โดยปราศจากการแทรกแซงของซีพียู ทั้งตัวส่งและตัวรับสามารถขัดจังหวะการทำงานของ CPU ได้

UART ใช้ขาภายนอก 2 ขา คือ Transmit และ Receive ข้อมูลที่จะส่งจะถูกปล่อยออกแบบอนุกรมในขา Transmit และข้อมูลที่รับเข้าจะถูกอ่านจากขา Receive

การส่งแบบ Asynchronous

Transmitter Data Output line เป็น High (ทำเครื่องหมาย) เมื่อตัวส่งไม่มีข้อมูลที่จะส่ง ภายใต้การควบคุมของโปรแกรมคำสั่ง Send Break สามารถที่จะเก็บ Data Output line เป็น Low(ว่าง) จนกระทั่งคำสั่งถูกเคลียร์

UART จะเพิ่ม start bit โดยอัตโนมัติ เมื่อพาริตี้บิทที่ถูกโปรแกรม (คี่,คู่หรือไม่มีพาริตี้) และจำนวนของ stop bit ที่ถูกโปรแกรมเป็นข้อมูลที่จะส่งออก เมื่อตัวอักษรมีขนาด ห้า,หก หรือเจ็ด บิท บิทที่มีนัยสำคัญสูงสุดที่ไม่ได้ใช้ใน Transmitter Data register จะไม่ถูกสนใจโดย UART

ข้อมูลอนุกรมที่ถูกเลื่อนจากตัวส่งที่อัตราความเร็วเท่ากับ 1,1/16,1/32 หรือ 1/64 ของอัตราความเร็วสัญญาณนาฬิกาถูกส่งไปที่สัญญาณนาฬิกาตัวส่ง ข้อมูลอนุกรมถูกเลื่อนออกบนขอบขาลงของสัญญาณนาฬิกา

การรับแบบ Asynchronous

การปฏิบัติการรับแบบ asynchronous เริ่มเมื่อ Receive Enable bit ใน Receive Control/Status register ถูกเซตเป็นหนึ่ง logic Low บนขา Receive input ระบุถึงบิทเริ่มต้น (start bit) ถ้า logic Low ยังคงอยู่ในช่วงเวลาครึ่งหนึ่งของคาบเวลาในแต่ละบิท แสดงว่าบิทเริ่มต้นมีค่าแล้ว และข้อมูลอินพุทจะถูกสุ่มที่ครึ่งเวลาของการส่ง จนกระทั่ง character ถูกรวมเข้าด้วยกันหมด

วิธีการตรวจหาบิทเริ่มต้นนี้ ปรับปรุงการลดข้อผิดพลาด ที่เกิดเมื่อมีสัญญาณรบกวนบนสายสัญญาณอื่นๆ ถ้าเลือกโหมด clock x1 การชิงโครไนซ์บิทต้องกระทำภายนอก ข้อมูลที่รับเข้ามาจะถูกสุ่มตัวอย่างที่ขอบขาขึ้นของสัญญาณนาฬิกา

ตัวอักษรที่รับได้ถูกอ่านจาก Receive Data register ถ้าพาริตี้สามารถทำงาน พาริตี้บิทถูกรวมเป็นส่วนหนึ่งของตัวอักษร และไม่มีการกำจัดพาริตี้บิทออกจากตัวอักษรที่มีความยาวมากกว่า 8 บิท ถ้าขนาดตัวอักษรยังคงน้อยกว่า 8 บิท ค่า 1 จะถูกต่อเข้ากับตำแหน่งไม่ได้ใช้ในลำดับสูงของตัวอักษร

ขณะที่ตัวรับใช้รีจีสเตอร์ที่เลื่อนข้อมูลได้ขนาด 8 บิทเป็นบัฟเฟอร์ในการรับข้อมูลอยู่นั้น ซีพียูจะมีเวลาพอเพียงที่จะบริการอินเตอร์รัพท์และรับตัวอักษรที่รวบรวมโดย UART ตัวรับจะมีบัฟเฟอร์ที่เก็บ error flag สำหรับแต่ละ character ด้วย ข้อมูลใน receive buffer error flag นี้ถูกโหลดเวลาเดียวกับ character ข้อมูล

หลังจาก character ถูกได้รับแล้วจะมีการตรวจสอบสำหรับเงื่อนไข error ต่อไปนี้

- ☐ Parity Error เมื่อพาริตี้บิทของตัวอักษรไม่ตรงกับพาริตี้ที่ถูกโปรแกรมไว้
- ☐ Framing Error ถ้าตัวอักษรที่ถูกรวมไม่มี stop bit (ตัวอย่าง Low level ถูกยืนยันเป็น stop bit)
- ☐ Receiver Overrun Error ถ้าซีพียูไม่สามารถอ่านตัวอักษรเพราะได้รับตัวอักษรมากกว่า 1 ตัว

เมื่อแฟลก Parity Error และ Receive Overrun Error flag ถูกแลทช์ไว้ สถานะผิดพลาดนี้จะมีผลต่อข้อผิดพลาดของตัวอักษรปัจจุบันที่อยู่ในรีจีสเตอร์รับข้อมูล บวกกับข้อผิดพลาดทาง Parity หรือ Overrun ใดๆที่ตรวจพบขณะที่เขียนสู่ Receiver Control/Status register ครั้งล่าสุด ดังนั้นเพื่อให้มีการตอบสนองระหว่างสถานะผิดพลาดในบัฟเฟอร์ กับข้อมูลที่รับได้ในบัฟเฟอร์ ต้องอ่านข้อมูลของ Receiver Control/Status register ก่อนจะอ่านข้อมูลอื่น

Polled Operation

ในสิ่งแวดล้อมแบบ poll Receive Character Available bit ใน Receiver Control/Status register จำเป็นต้องถูกตรวจสอบเพื่อให้ซีพียูรู้ว่าเมื่อไรจะอ่าน character บิทนี้จะเคลียร์อย่างอัตโนมัติเมื่อ Receiver Data register ถูกอ่าน การป้องกันการเขียนข้อมูลทับในการปฏิบัติแบบ poll poll transmitter buffer status จำเป็นต้องตรวจสอบก่อนการเขียนออกสู่ตัวส่ง Transmit Buffer Empty bit ใน Transmitter Control/Status register จะถูกเซตเป็นหนึ่ง เมื่อใดก็ตามที่ transmit buffer ว่างเปล่า

UART Control and Status register

การทำงานของ UART ถูกควบคุมโดย control and status register 3 ตัว UART Configuration register ระบุขนาดของ character,พาริตี้,clock source,การกำหนดสเกลและ loop-back enable ทั้งตัวส่งและตัวรับมี control/status register เป็นของตัวเอง

UART Configuration Register เป็นรีจีสเตอร์ขนาด 8 bit (รูปที่ 1.23) บรรจุข้อมูลควบคุมสำหรับทั้งตัวส่งและตัวรับ

7					0
B/C	P	E/O	CS	CR	LB

รูปที่ 1.23 UART Configuration รีจีสเตอร์

แต่ละบิทประกอบด้วย

Loopback Enable (LB) UART สามารถเกิด local loopback ในโหมดนี้ขาส่งข้อมูลภายในจะเชื่อมต่อกับขารับข้อมูลภายใน ดังนั้นจึงไม่ต้องสนใจสัญญาณภายนอก ถ้าบิทนี้เซตเป็นหนึ่ง loopback mode จะทำงาน

Clock Rate (CR) 2 บิทนี้ใช้ระบุตัวคูณระหว่างสัญญาณนาฬิกาและอัตราความเร็วของข้อมูล(00=data rate x 1, 01=data rate x 16, 10=data rate x 32, 11=data rate x 64) อัตราความเร็วเดียวกันนี้ถูกใช้สำหรับทั้งตัวรับและตัวส่ง ถ้า x 1 clock rate ถูกเลือก การซิงโครไนซ์บิทต้องกระทำภายนอก

Clock Select (CS) บิทนี้ใช้ระบุสัญญาณนาฬิกาอินพุทสำหรับ UART ถ้าบิทนี้ถูกเซตเป็นหนึ่งพัลส์เอาท์พุทของ counter/timer 1 ถูกใช้สำหรับ bit-rate generation ถ้าบิทนี้เคลียร์เป็นศูนย์ขาอินพุทของ counter/timer 1 จะเตรียมสัญญาณนาฬิกาจากภายนอก

Parity Even/Odd(E/O) ถ้าพาริตี้ทำงาน บิทนี้จะใช้ตรวจสอบว่าเป็นพารีตี้คู่หรือคี่ (1=คู่)

Parity (P) ถ้าบิทนี้เซตเป็นหนึ่ง ตำแหน่งบิทที่เพิ่มขึ้น(ส่วนที่เพิ่มขึ้นจะระบุอยู่ใน bits/character control field) ถูกเพิ่มในข้อมูลที่ส่งและรับที่ด้านรับ ในด้านรับพาริตี้บิทที่รับได้รับจะถูกส่งไปซีพียูเหมือนเป็นตัวอักษร ถ้าไม่ได้เลือก 8 bit/character

Bit/Character (B/C) 2 บิทนี้ใช้พิจารณาจำนวนของบิทต่อ 1 ตัวอักษร ถ้าบิทนี้ถูกเปลี่ยนแปลงระหว่างเวลาที่รวมเป็นตัวอักษร ผลลัพธ์จะคาดเดาไม่ได้ (00=5 bit/character, 01=6 bit/character, 10=7 bit/character, 11= 8 bit/character)

Transmitter Control/Status Register เป็นรีจีสเตอร์ขนาด 8 bit (รูปที่ 1.24) ระบุการทำงานของตัวส่ง

7							0
EN	IE	0	SB	BRK	FRC	VAL	BE

รูปที่ 1.24 Transmitter Control/Status รีจีสเตอร์

control bit ในรีจีสเตอร์นี้คือ

Transmitter Buffer Empty (BE) บิทนี้จะเซตเป็นหนึ่งโดยอัตโนมัติเมื่อใดก็ตามที่ transmitter buffer ว่างเปล่าและเคลียร์เป็นศูนย์ เมื่อ character ถูกโหลดไปที่ transmitter buffer บิทนี้จะอยู่ในภาวะถูกเซตหลังจากเกิดการรีเซตระบบ บิทนี้ถูกควบคุมโดยวงจรควบคุมของ UART สามารถถูกอ่านโดย I/O read แต่ไม่สามารถเซตเป็นหนึ่งหรือเคลียร์เป็นศูนย์โดย I/O write

Value (VAL) บิทนี้ใช้พิจารณาค่าของ bit ที่ส่งในขณะที่ FRC bit เป็นหนึ่งและ dummy character ถูกโหลดไปที่ transmitter buffer เมื่อบิทนี้เป็นหนึ่ง mark character (เป็น 1 ทั้งหมด) ถูกส่งออก เมื่อบิทนี้เป็นศูนย์ break character(เป็น 0 ทั้งหมด)จะถูกส่งออก

Force Character (FRC) เมื่อบิทนี้ถูกเซตเป็นหนึ่ง character ใดๆถูกโหลดไปที่ transmitter buffer ทำให้ transmitter output ยังคงเป็นสัญญาณ High หรือ Low(ซึ่งระบุโดย VAL bit) ในช่วงระยะเวลาที่ต้องการส่ง character การเซตบิทนี้เป็นหนึ่งและเซต VAL bit เป็นหนึ่งหรือศูนย์ และโหลดจำนวนของ dummy character ที่เหมาะสมไปที่ transmitter buffer ทำให้โปรแกรมกำเนิดสัญญาณ marking หรือเบรกช่วงของ multiple-character

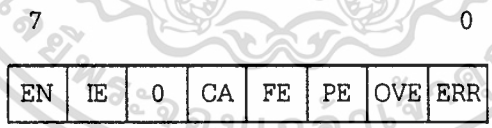
Send Break (BRK) เมื่อเซตเป็นหนึ่งบิทนี้จะบังคับ transmitter output ทันทีทันใดให้ส่งเงื่อนไขว่างเปล่าโดยไม่สนใจข้อมูลที่กำลังส่ง เมื่อบิทนี้ถูกเคลียร์เป็นศูนย์ตัวส่งจะกลับสู่ค่า marking

Stop Bit (SB) บิทนี้ใช้พิจารณาจำนวนของ stop bit ที่เพิ่มเข้าไปในการส่ง asynchronous character ตัวรับจะตรวจสอบ stop bit เสมอ ถ้าบิทนี้ถูกเซตเป็นหนึ่ง stop bit 2 บิทจะถูกเพิ่มอย่างอัตโนมัติในการส่ง character ถ้าบิทนี้เคลียร์เป็นศูนย์จะมี stop bit เพียงบิทเดียวเท่านั้นที่ถูกเพิ่ม

Transmitter Interrupt Enable (IE) เมื่อบิทนี้ถูกเซตเป็นหนึ่งการร้องขออินเตอร์รัพท์จะเกิดเมื่อใดก็ตามที่ transmitter buffer ว่างเปล่า เมื่อบิทนี้ถูกเคลียร์เป็นศูนย์จะไม่มีการร้องขอ

Transmitter Enable (EN) ในขณะที่บิทนี้ถูกเคลียร์เป็นศูนย์ ข้อมูลจะไม่ถูกส่งและ transmitter output จะถูก mark ถ้าบิทนี้ถูกเคลียร์เป็นศูนย์ หลังจากการส่งเริ่มต้นไปแล้ว ข้อมูลในกระบวนการส่งจะถูกส่งอย่างสมบูรณ์

Receiver Control/Status Register เป็นรีจีสเตอร์ขนาด 8 bit (รูปที่ 26)ใช้ระบุการทำงานของตัวรับ control bit อธิบายได้ดังนี้



รูปที่ 1.25 Receiver Control/Status รีจีสเตอร์

Receiver Error (ERR) บิทนี้คือการ OR กับทางตรรกของ bit PE,OVE และ FE

Framing Error (FE) บิทนี้จะเซตตัวเองเป็นหนึ่งโดยอัตโนมัติ เมื่อตัวอักษรที่รับมามี framing error

Parity Error (PE) เมื่อพาริตี้สามารถทำงาน บิทนี้ถูกเซตเป็นหนึ่งโดยอัตโนมัติสำหรับตัวอักษรที่มีพาริตี้ไม่ตรงกับพาริตี้ที่โปรแกรม(คู่/คี่) บิทนี้ถูกแลทช์ ถ้ามีความผิดพลาดเกิดขึ้น และบิทนี้ยังคงเซตเป็นหนึ่งอยู่จนกระทั่งถูกเคลียร์โดยซอฟท์แวร์

Receive Overrun Error (OVE) บิทนี้จะเซตเป็นหนึ่งโดยอัตโนมัติเมื่อได้รับตัวอักษรมากกว่า 2 ตัว ไม่ว่าจะอ่านได้จากซีพียู (หรือ DMA) ตัวอักษรที่รับมาล่าสุดเท่านั้นที่จะถูกตรวจข้อผิดพลาด แต่เมื่อตัวอักษรนี้ถูกอ่าน เงื่อนไขความผิดพลาดจะถูกแลทช์จนกระทั่งถูกเคลียร์โดยซอฟท์แวร์

Receiver Character available (CA) บิทนี้เซตตัวเองเป็นหนึ่งโดยอัตโนมัติ เมื่อมีตัวอักษรล่าสุดใน receive buffer และจะเคลียร์ตัวเองเป็นศูนย์โดยอัตโนมัติ เมื่อ Receiver Data register ถูกอ่าน บิทนี้ถูกควบคุมโดยวงจรควบคุม UART บิทนี้สามารถอ่านโดย I/O read แต่ไม่สามารถเซตหรือเคลียร์โดย I/O write

Receiver Interrupt Enable (IE) ในขณะที่บิทนี้เป็นหนึ่ง การร้องขออินเตอร์รัพท์จะเกิดเมื่อตัวรับได้รับ error หรือตัวรับสามารถหา character ได้

Receiver Enable (EN) เมื่อบิทนี้เซตเป็นหนึ่ง การทำงานของตัวรับจะเริ่มต้นขึ้น บิทนี้จะเซตหลังจาก parameter ใน UART Configuration register ถูกเซตเท่านั้น

UART Bootstrapping Option

Z280 สนับสนุนการเริ่มต้นหน่วยความจำโดยอัตโนมัติ หลังจากมีการรีเซตโดยผ่านทาง UART ความสามารถของ Bootstrap นี้จะรองรับระบบที่ไม่มี ROM หลังจากรีเซตหน่วยความจำสามารถเริ่มต้นโดยการเชื่อมต่อแบบอนุกรมก่อนที่ Z280 จะเฟทช์ข้อมูลจากหน่วยความจำ

ที่ขอบขาขึ้นของ Reset ขาสัญญาณ AD จะรับรู้ถ้ามีสัญญาณ WAIT ถ้า AD_0 กำลังเป็น High Z280 จะเข้าสู่สภาวะ Halt โดยอัตโนมัติ UART ถูก initial โดยอัตโนมัติให้รับข้อมูล character 8 bit โดยใช้ odd parity ที่อัตราการส่ง x 16 clock rate แหล่งกำเนิดสัญญาณนาฬิกามาจากภายนอก การส่งข้อมูลสามารถเริ่มต้นได้เมื่อเวลาผ่านไปอย่างน้อย 15 clock cycle

ระหว่างการทำ bootstrap DMA channel 0 ถูกใช้เคลื่อนย้าย character ที่ได้รับไปที่หน่วยความจำ channel นี้ถูก initial ตามนี้

Transfer Descriptor Register - IE, EPS, และ TC cleared, ST-byte transfer, BRP-continuous, TYPE-flowthrough, DAD-Auto-increment memory address

DMA Master Control Register - DOR and EOP set

Count Register - 0100 (256 byte ถูกเคลื่อนย้าย)

Destination Address Register - 000000 (แอดเดรสหน่วยความจำเริ่มแรก = 0)

Source Address Register - undefined (ไม่ใช้เมื่อ DMA 0 ถูกเชื่อมต่อกับ UART)

character ที่ได้รับถูกวางในหน่วยความจำของ physical memory ตำแหน่งที่ศูนย์ ถ้าเกิดความผิดพลาดขึ้น Z280 จะทำให้ขาสัญญาณ Transmitter Output เป็น Low วงจรภายนอกตรวจสอบขาสัญญาณนี้สามารถใช้สัญญาณนี้เริ่มขบวนการนี้ใหม่อีกครั้ง โดยเริ่มต้นด้วยการรีเซต และให้สัญญาณ AD_0 ที่ขอบขาขึ้นของการ Reset

หลังจากข้อมูลถูกส่งไป 256 byte แล้ว Z280 จะเริ่มต้นการทำงานโดยอัตโนมัติโดยเฟทช์คำสั่งแรก จากหน่วยความจำตำแหน่งที่ศูนย์

1.16 DMA CHANNELS

ไมโครโปรเซสเซอร์ Z280 มี Direct Memory Access (DMA) อยู่ในตัว 4 channel เพื่อให้รับส่งข้อมูลความเร็วสูง มี DMA อยู่ 2 ชนิดๆละ 2 channel โดยที่มี 2 channels สนับสนุนการทำงานใน flyby transaction และอีก 2 channel ไม่สนับสนุน

แต่ DMA ทั้ง 2 ชนิดก็มีประสิทธิภาพเหมือนกัน ถึงแม้ว่าทั้ง 2 ชนิดมีลำดับความสำคัญในการร้องขออินเตอร์รัพท์และการร้องขอ BUS ต่างกัน

DMA แต่ละ channel เป็นอุปกรณ์ที่มีประสิทธิภาพและมีความสามารถหลากหลายในการควบคุมและประมวลผลการรับส่งของข้อมูล ฟังก์ชั่นพื้นฐานของการจัดการการรับส่งข้อมูลที่เป็นอิสระจากซีพียูระหว่าง 2 พอร์ทที่ต้องการคุณสมบัติบางอย่างนั้น ต้องการวงจรภายนอกเพิ่มเติมเพียงเล็กน้อยหรืออาจไม่ต้องการเลย ทั้งสำหรับดาต้าบัสขนาด 8 บิทและ 16 บิท

การรับส่งข้อมูลสามารถกระทำระหว่าง 2 พอร์ทใดๆ (ต้นทางและปลายทาง) รวมทั้ง หน่วยความจำกับ I/O, I/O กับหน่วยความจำ,หน่วยความจำกับหน่วยความจำ, I/O กับ I/O ยกเว้นในโหมด flyby แอดเดรสของ 2 พอร์ทถูกสร้างอย่างอัตโนมัติสำหรับการติดต่อแต่ละอย่าง และแอดเดรสของพอร์ทก็สามารถมีค่าคงที่หรือมีค่า เพิ่มขึ้น/ลดลง ได้

ระหว่างการรับส่ง DMA จะควบคุมแอดเดรสของระบบและดาต้าบัส ข้อมูลถูกอ่านจากพอร์ทที่ถูกกำหนดแอดเดรสพอร์ทหนึ่งและถูกเขียนที่พอร์ทที่ถูกกำหนดแอดเดรสอีกพอร์ทหนึ่งแบบ byte ต่อ byte หรือ word ต่อ word พอร์ทที่ใช้สามารถโปรแกรมให้เป็นได้ทั้งหน่วยความจำหลักของระบบหรืออุปกรณ์ I/O สนับสนุนอื่นๆ

ในการเคลื่อนย้ายข้อมูลโดย DMA ทั้งแบบ flyby และ flowthrough ถ้าตำแหน่งปลายทางตรงกับหน่วยความจำแคชภายใน (หรือหน่วยความจำคงที่) หน่วยความจำแคชภายในตำแหน่งนั้นจะได้รับการแก้ไขค่าให้ถูกต้อง

ยกเว้นใน flyby mode ตำแหน่งแอดเดรสขนาด 24 บิททั้ง 2 ค่าถูกสร้างมาจาก DMA สำหรับการรับส่งข้อมูลทุกๆอย่าง แอดเดรสค่าหนึ่งสำหรับพอร์ทต้นทางและอีกค่าสำหรับพอร์ทปลายทาง counter นับแอดเดรสที่อ่านได้ทั้ง 2 ตัว(แต่ละตัวมี 3 byte)ใช้เก็บแอดเดรสปัจจุบันของแต่ละพอร์ท

อุปกรณ์ DMA ติดต่อหน่วยความจำและ I/O เช่นเดียวกับ CPU จัดการบัส ซึ่งถูกระบุโดย bus timing Register ที่เหมาะสม

Mode of Transfer Operation

DMA แต่ละ channel สามารถถูกโปรแกรมให้ทำงานในหนึ่งในสามโหมดต่อไปนี้

- ☐ Single Transaction ในการทำงานกับข้อมูลแต่ละครั้งจะถูกทำเป็น byte หรือ word
- ☐ Burst การทำงานกับข้อมูลจะทำต่อเนื่องจนกระทั่ง สัญญาณ port's Ready line ที่ส่งไป DMA ไม่แอคทีฟ
- ☐ Continuous การทำงานกับข้อมูลจะทำต่อเนื่องจนกระทั่ง ข้อมูลสุดท้ายของกลุ่มที่ถูกโปรแกรมไว้มาถึง หรือสัญญาณสิ้นสุดการทำงาน (EOP) ถูกตรวจพบก่อนที่ system bus จะถูกปลดปล่อยจาก DMA

ในทุกโหมด เมื่อข้อมูลเป็น byte หรือ word ถูกอ่านโดย DMA channel การทำงานจะสมบูรณ์เป็นลำดับโดยไม่สนใจสภาวะของสัญญาณอื่น(รวมทั้งสัญญาณ port's Ready line)

Pin Description

แต่ละ channel ของ DMA มีสัญญาณ Ready input line รวมกับ DMA 2 channel ที่มี flyby output line ที่สนับสนุนการรับส่งข้อมูลความเร็วสูงระหว่างอุปกรณ์ I/O และหน่วยความจำ

flyby output ถูกยืนยันโดย DMA channel ซึ่งส่งสัญญาณไปอุปกรณ์รอบข้างที่เกี่ยวข้องกับ DMA channel ให้อุปกรณ์เหล่านี้มีส่วนเกี่ยวข้องกับการรับส่งข้อมูลระหว่าง flyby bus transaction ปัจจุบัน

สัญญาณ Ready line ถูกสุ่มที่ขอบขาขึ้นของแต่ละ processor clock cycle ถ้าสัญญาณ Ready แอคทีฟ DMA channel ร้องขอการควบคุม system bus ภายนอกเพื่อที่จะจัดการ DMA เมื่อ system bus ภายนอกอนุญาตให้ใช้การรับส่งของ DMA DMA channel ที่ยังมีการร้องขอค้างอยู่และมีลำดับความสำคัญสูงสุดจะถือว่าเป็น bus mastership ลำดับความสำคัญของ DMA channel เรียงจากสูงสุดไปต่ำสุดคือ DMA0,DMA1,DMA2 และ DMA3 DMA channel ใน burst mode จะปล่อย bus mastership ไปที่ DMA channel ที่มีลำดับความสำคัญสูงสุด เมื่อสัญญาณ Ready line ไม่ถูกยืนยัน (หรือมีสัญญาณ EOP หรือการนับสิ้นสุด) เท่านั้น DMA channel ใน continuous mode จะปล่อย bus mastership เมื่อมีสัญญาณ EOP หรือการนับสิ้นสุดเท่านั้น

ลำดับความสำคัญของ On-Chip DMA Channel และการร้องขอบัสภายนอก

On-chip DMA channel ถูกจัดลำดับความสำคัญเท่ากับการร้องขอบัสภายนอก ซึ่งจะมีลำดับความสำคัญต่ำ DMAs ภายในเหล่านี้จะทำตัวราวกับว่าเป็นอุปกรณ์ภายนอกที่ร้องขอการใช้บัส ซึ่งจะทำให้เกิดการยึดครองบัส , การปลดปล่อยบัส และการเรียงลำดับความสำคัญในการเข้าถึงบัส

End-of-Process

ถ้ากำหนดให้สามารถเกิด End-of-Process ได้ โดยการโปรแกรมค่า control bit ใน DMA Master Control Register จะทำให้การรับส่งข้อมูลโดย DMA channel สามารถสิ้นสุดก่อนกำหนดโดยการให้ค่า Low บนขา Interrupt A ระหว่างการรับส่งข้อมูล EOP สามารถเกิดขึ้นโดยไม่ต้องเซตค่าของ Interrupt A Enable bit ใน Master Status Register แต่อย่างใด เมื่อเกิดสัญญาณ EOP บิท EOP Signaled (EPS) ใน Transaction Descriptor Register ของ DMA ตัวที่แอคทีฟถูกเซตเป็น 1 และ Enable bit ถูกเคลียร์เป็น 0 ถ้าสามารถร้องขออินเตอร์รัพท์ได้ (IE=1 ใน Transaction Descriptor Register) การร้องขออินเตอร์รัพท์จะถูกสร้างโดย DMA channel ที่แอคทีฟ เมื่อเกิดสัญญาณ EOP หลังจากเกิดสัญญาณ EOP แล้ว DMA จะปล่อยการใช้บัสภายใน 16 ไซเคิลของ DMA bus transaction ท้ายสุด

ถ้าสัญญาณ End-of-Process บนขา Interrupt A ยังคงอยู่ เมื่อซีพียูเป็นตัวจัดการบัสแล้ว สัญญาณนี้จะถูกแปลเป็นการร้องขออินเตอร์รัพท์ ดังนั้นทั้ง DMA channel และอุปกรณ์ที่สร้างสัญญาณ EOP ภายนอกสามารถร้องขออินเตอร์รัพท์ในเวลาเดียวกัน การแยกบิทที่ทำเครื่องหมายใน Master Status Register ทำให้ CPU ยอมรับการอินเตอร์รัพท์จากสองแหล่งนี้

ใน flowthrough transaction ถ้าเกิดสัญญาณ EOP ในขณะที่ข้อมูลกำลังถูกอ่านไปที่ไมโครโปรเซสเซอร์ Z280 การรับส่งข้อมูลจะถูกยกเลิกและข้อมูลไม่ได้ถูกเขียนออกจากไมโครโปรเซสเซอร์ Z280

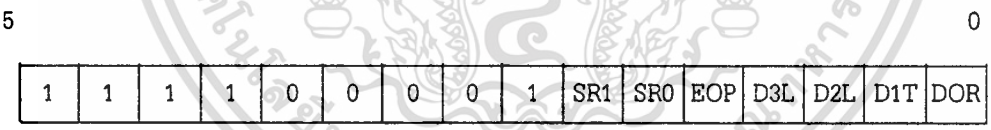
DMA Linking

อุปกรณ์ DMA สามารถที่จะเชื่อมโยงเข้าด้วยกันเพื่อให้การเคลื่อนย้ายข้อมูลของ DMA ไปตำแหน่งในหน่วยความจำที่ไม่ติดกัน (linked operation) ประสบความสำเร็จ บิทใน DMA Master Control Register ยอมให้ DMA3 เชื่อมโยงกับ DMA1 และ DMA2 เชื่อมโยงกับ DMA0 ถ้าบิทที่เหมาะสมใน DMA Master Control Register ถูกเซตเป็น 1 master DMA (0 หรือ 1) จะส่งสัญญาณเชื่อมโยง DMA ในแต่ละครั้งที่การรับส่งสมบูรณ์(count=0) การปฏิบัตินี้ราวกับว่าสัญญาณแสดงความพร้อมภายในเข้าไปเชื่อมโยงกับ DMA ที่ได้โหลด master DMA control Register ขึ้นมาใหม่

ข้อมูลเป็น word ถูกโหลดไปที่ master DMA control Register เป็นลำดับต่อไปนี้ Destination Address Register (2 word), Source Address Register (1 word), Count (1 word), Transfer Descriptor Register (1 word) หลังจากข้อมูลทั้ง 6 word ถูกถ่ายโอนไปแล้ว master DMA ไม่ยืนยันสัญญาณเตรียมพร้อมภายในและเริ่มต้นการเคลื่อนย้าย ข้อมูล block ต่อไป DMA ที่ถูกเชื่อมโยงสามารถถูกโปรแกรมให้อินเตอร์รัพท์ CPU เมื่อนับเท่ากับศูนย์ (แจ้งแก่ซอฟท์แวร์ว่า block สุดท้ายกำลังถูกเคลื่อนย้าย) หรือ master DMA สามารถที่จะถูกโปรแกรมให้อินเตอร์รัพท์ CPU ที่การนับเท่ากับศูนย์ เมื่อการเคลื่อนย้าย block สุดท้ายถูกโปรแกรมไปที่ master DMA (แจ้งแก่ซอฟท์แวร์ว่าลำดับที่เข้ามาทั้งหมดของการเคลื่อนย้ายถูกทำให้สมบูรณ์แล้ว) เมื่อการเชื่อมโยงทำงาน สัญญาณ Ready line ภายนอกไม่ถูกยืนยันโดย master DMA เมื่อการนับเท่ากับศูนย์แล้ว ทั้ง master และ linked DMA จะร้องขออินเตอร์รัพท์เมื่อใดก็ตามที่เงื่อนไขที่ถูกโปรแกรมไว้เกิดขึ้น

เมื่อมีการโปรแกรมเชื่อมโยง DMA ข้อมูลเป็น word สุดท้ายที่ถูกโปรแกรมจะต้องเป็น master DMA's Transaction Descriptor Register ตัว DMA ที่ถูก link จำเป็นต้องถูกโปรแกรมก่อนที่ master DMA's status Register จะถูกโปรแกรม

DMA Master Control Register เป็นรีจีสเตอร์ขนาด 16 bit (รูปที่ 1.26) ใช้ระบุโครงสร้างทั่วไปของ on-chip DMA channel ทั้ง 4 channel เช่น การเชื่อมโยงของ DMA channel, software ready enable และ EOP enable



รูปที่ 1.26 DMA Master Control Register

แต่ละบิทในรีจีสเตอร์นี้ประกอบด้วย

DMA0 to Receiver Link (D0R) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 0 ถูกเชื่อมโยงกับตัวรับของ UART

DMA1 to Transmitter Link (D1T) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 1 ถูกเชื่อมโยงเข้ากับตัวส่งของ UART

DMA2 Link (D2L) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 2 ถูกเชื่อมโยงเข้ากับ DMA channel 0

DMA3 Link (D3L) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 3 ถูกเชื่อมโยงกับ DMA channel 1

End-of-Process (EOP) เมื่อบิทนี้ถูกเซตเป็น 1 ขา INT_A ถูกใช้เป็นสัญญาณ End-of-Process สำหรับ DMA channel ที่แอคทีฟ

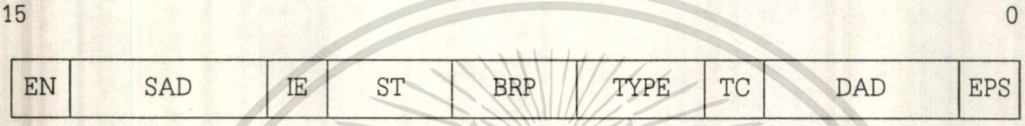
Software Ready for DMA0(SR0) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 0 จะร้องขอบริการ ถ้าสามารถทำได้

Software Ready for DMA1(SR1) เมื่อบิทนี้ถูกเซตเป็น 1 DMA channel 1 จะร้องขอบริการ ถ้าสามารถทำได้

บิท D11-D8 ต้องโปรแกรมให้เป็น 0

DMA Channel Control Register

Transaction Descriptor Register เป็นรีจีสเตอร์ขนาด 16 bit 4 ตัวสำหรับแต่ละ channel (รูปที่ 1.27) ซึ่งบอกถึงชนิดของการเคลื่อนย้ายข้อมูลของ DMA ที่ถูกกระทำและบรรจุข้อมูลการควบคุมและสภาวะไว้ แต่ละบิทในรีจีสเตอร์นี้ประกอบด้วย



รูปที่ 1.27 Transaction Descriptor Register

End-of-Process Signaled (EPS) บิทนี้ถูกเซตเป็น 1 โดยอัตโนมัติเมื่อ channel แอคทีฟและมีสัญญาณ End-of-Process บนขา Interrupt A ดังนั้นเป็นการสิ้นสุดการทำงานก่อนกำหนดของการเคลื่อนย้ายข้อมูล

Destination Address Descriptor (DAD) การเซตค่าของ 3 bit นี้ใช้ระบุชนิดของตำแหน่ง (หน่วยความจำหรือ I/O)และแอดเดรสถูกถ่ายเทอย่างไร (เพิ่มขึ้น,ลดลง หรือไม่เปลี่ยนแปลง) แสดงในตารางที่ 1.5

Encoding	Address modification operation
000	ตำแหน่งหน่วยความจำเพิ่มขึ้นโดยอัตโนมัติ
001	ตำแหน่งหน่วยความจำลดลงโดยอัตโนมัติ
010	ตำแหน่งของหน่วยความจำไม่เปลี่ยนแปลง
011	ไม่ได้ใช้
100	ตำแหน่ง I/O เพิ่มขึ้นหนึ่งโดยอัตโนมัติ
101	ตำแหน่ง I/O ลดลงหนึ่งโดยอัตโนมัติ
110	ตำแหน่ง I/O ไม่เปลี่ยนแปลง
111	ไม่ได้ใช้

ตารางที่ 1.5 SAD and DAD Encodings

Transfer Complete (TC) บิทนี้จะเซตตัวเองเป็น 1 โดยอัตโนมัติ เมื่อรีจีสเตอร์นับ นับถึงศูนย์

Transaction Type (Type) 2 บิทนี้ใช้ระบุชนิดของการปฏิบัติการเป็น flyby หรือ flowthrough (00 = flowthrough, 01 = reserved, 10 = flyby write, 11 = flyby read) การทำงานในโหมด flowthrough เกิดการจัดการบัส 2 ครั้งสำหรับการทำงานของ DMA คือ การอ่านจากต้นทาง ตามด้วยการเขียนที่ปลายทาง ในขณะที่การปฏิบัติการ flyby เกิดการจัดการบัสเพียงครั้งเดียวเท่านั้น สำหรับการปฏิบัติการ DMA ในโหมด flyby การเขียนสู่หน่วยความจำ ขา flyby output จะส่งสัญญาณมาแทนที่การจัดการ I/O ที่กำลังทำงานอยู่ และข้อมูลภายใน Destination Address Register ถูกส่งออกมาระบุตำแหน่งในหน่วยความจำ ในโหมด flyby การอ่านจากหน่วยความจำ ขา flyby output จะส่งสัญญาณมาแทนที่การจัดการ I/O ที่กำลังทำงานอยู่และข้อมูลใน Source Address Register ถูกส่งออกมาระบุตำแหน่งในหน่วยความจำ มีเพียง DMA 2 channel เท่านั้นที่สามารถใช้ flyby ได้

Bus Request Protocol (BRP) การเซตค่าของ 2 bit นี้ใช้ระบุโหมดในการทำงานของ DMA (ตารางที่ 1.6)

Encoding	DMA
0 0	Single Transaction
0 1	Burst
1 0	Continuous
1 1	Reserved

ตารางที่ 1.6 Bus Request Protocol(BRP)

Size of Transfer (ST) 2 บิทนี้ใช้ระบุขนาดของข้อมูลที่ถูกเคลื่อนย้ายโดย DMA channel (ตารางที่ 1.7) สำหรับการเคลื่อนย้ายข้อมูลที่เป็น word เข้าหรือออกจากหน่วยความจำ แอดเดรสของหน่วยความจำจะต้องเป็นคู่ (บิทที่มีนัยสำคัญต่ำสุดเป็นศูนย์) การเคลื่อนย้ายข้อมูลระดับ long word (32 bit) จะทำงานในโหมด flyby เท่านั้น และแคชจะไม่ทำงาน

Encoding ST1 ST0	Size of Transfer	Number to increment/ Decrement By
0 0	Byte	1
0 1	16-bit word	2
1 0	32-bit longword	4
1 1	Reserved	

ตารางที่ 1.7 Size of Transaction(ST)

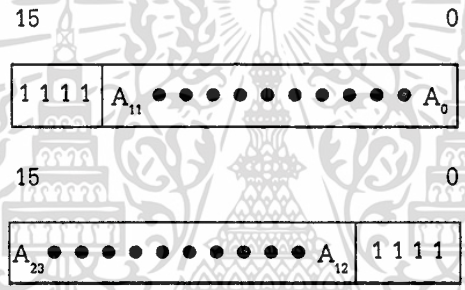
Interrupt Enable(IE) เมื่อบิทนี้ถูกเซตเป็น 1 DMA จะร้องขออินเตอร์รัพท์ที่จุดสิ้นสุดของการนับหรือจุดสิ้นสุดของกระบวนการ เมื่อบิทนี้เป็น 0 จะไม่มีสัญญาณร้องขออินเตอร์เกิดขึ้น

Source Address Descriptor (SAD) การเซตค่าของ 3 บิทนี้ใช้ระบุชนิดของตำแหน่ง (หน่วยความจำหรือ I/O) และถูกเคลื่อนย้ายอย่างไร (เพิ่มขึ้น,ลดลง หรือไม่เปลี่ยนแปลง) แสดงในตารางที่ 1.5

DMA Enable (EN) ในขณะที่บิทนี้เป็น 1 การเคลื่อนย้ายข้อมูลโดย DMA สามารถทำงานได้

Count Register เป็นรีจีสเตอร์ขนาด 16 บิทที่ถูกโปรแกรมให้เก็บจำนวนของการเคลื่อนย้ายข้อมูลของ DMA ที่ถูกกระทำ เมื่อข้อมูลที่มีอยู่ใน count Register เป็นศูนย์ สัญญาณร้องขอต่างๆบนขา RDY input จะไม่ถูกสนใจต่อไป DMA channel สามารถถูกโปรแกรมให้เกิดการอินเตอร์รัพท์เมื่อ count Register นับถึงศูนย์

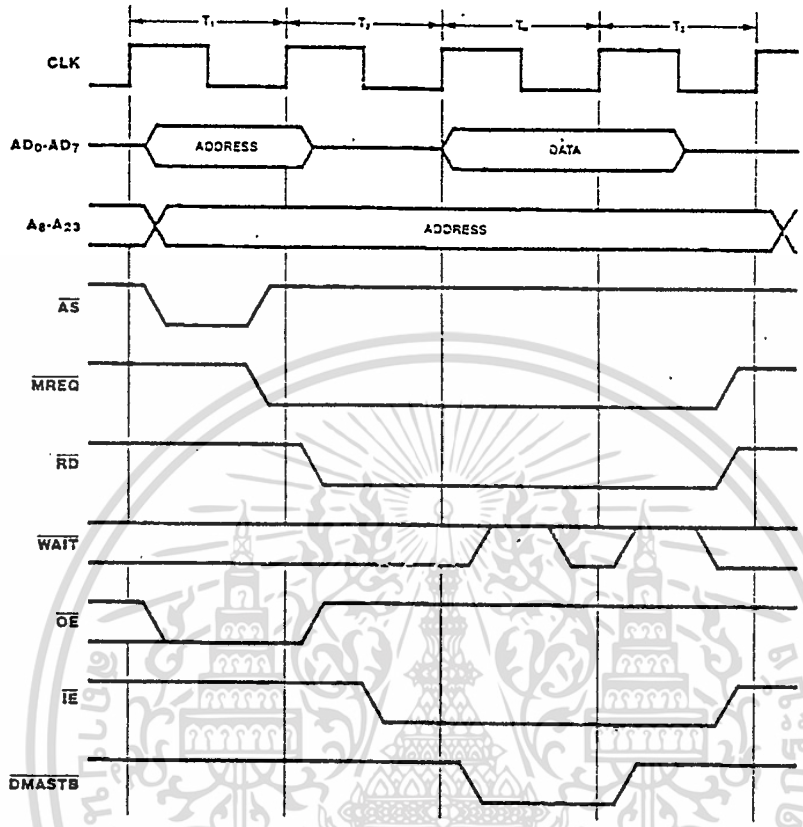
Source Address Register and Destination Address Register เป็นรีจีสเตอร์ขนาด 24 บิทที่เก็บ physical address ขนาด 24 บิทที่ถูกใช้ระหว่างการจัดการ DMA ค่าในรีจีสเตอร์ 2 ตัวนี้จะไม่ถูกแปลงโดย MMU ในโหมด flyby จะมีรีจีสเตอร์ตัวเดียวเท่านั้นที่กำหนดแอดเดรสสำหรับการจัดการบัส เช่นเดียวกับที่ระบุในบิทที่บอกโหมดใน Transfer Descriptor Register รูปแบบของรีจีสเตอร์เหล่านี้แสดงในรูปที่ 1.28



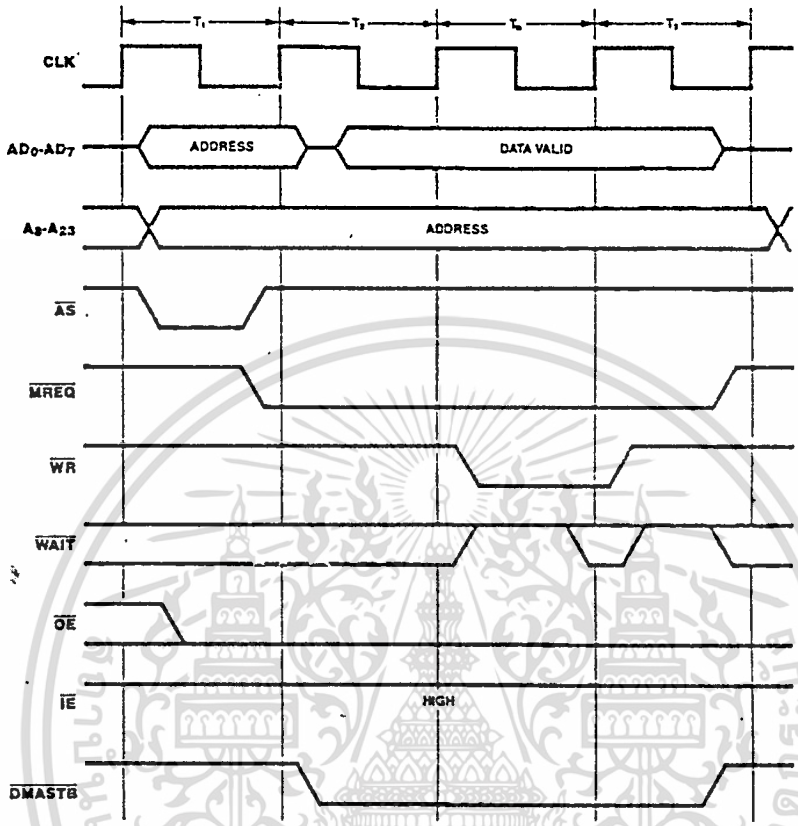
รูปที่ 1.28 Source and Destination Address Register Format

Flyby Transaction Timing

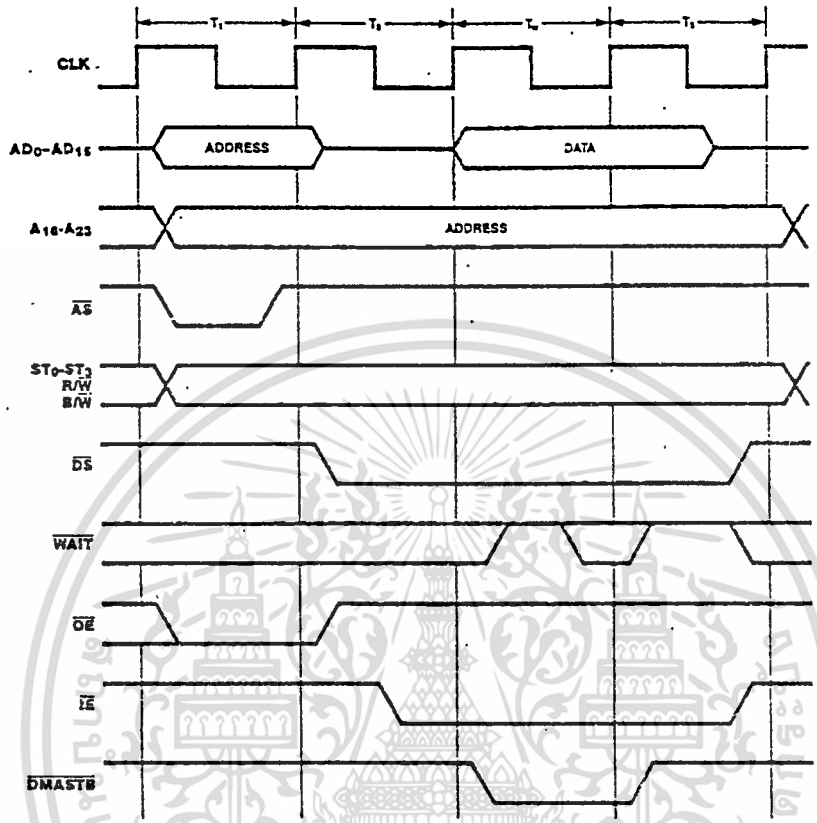
ค่า Transaction Type ใน Transaction Descriptor Register ใช้ระบุว่าการจัดการบัสเป็นการอ่านหรือการเขียน สำหรับการอ่านในโหมด flyby, Source Address Descriptor ใช้ระบุว่าการจัดการคือการอ่านจากหน่วยความจำ สำหรับการเขียนในโหมด flyby Destination Address Descriptor ใช้ระบุว่าการจัดการเป็นการเขียนไปที่หน่วยความจำ รวมกับ wait state ที่สามารถถูกแทรกได้อย่างอัตโนมัติถ้ากำหนดค่าใน timing Register อย่างเหมาะสม ดูรูปที่ 1.29 และ 1.30 สำหรับ timing diagram



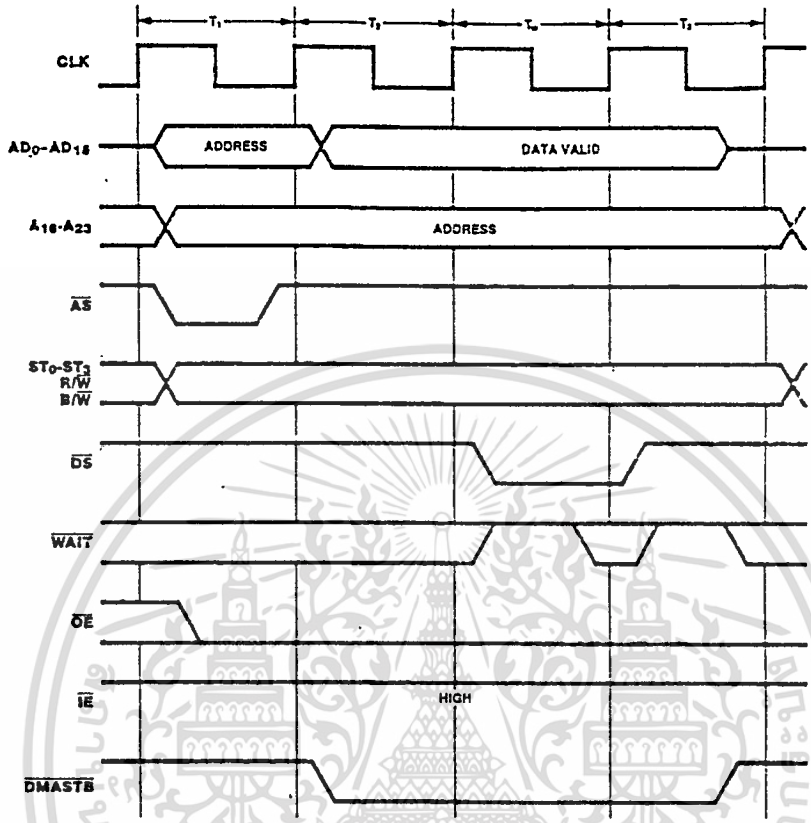
รูปที่ 1.29a On-Chip DMA Channel Flyby Memory Read Transaction Z80 Bus



รูปที่ 1.29b On-Chip Channel Flyby Memory Write Transaction Z80 Bus



รูปที่ 1.30a On-Chip DMA Channel Flyby Memory Read Transaction Z-BUS



รูปที่ 1.30b On-Chip DMA Channel Flyby Memory Write Transaction Z-BUS

1.17 COUNTER/TIMERS

ไมโครโปรเซสเซอร์ Z280 มี counter/timing อยู่ภายใน 3 ตัวซึ่งสามารถโปรแกรมโดย system software สำหรับการประยุกต์ด้านการนับและฐานเวลาได้อย่างมาก counter/timer ทั้ง 3 channel สามารถถูกโปรแกรมได้โดยอิสระเพื่อตอบสนองความต้องการพื้นฐานทั่วไปของระบบไมโครคอมพิวเตอร์ สำหรับการนับเหตุการณ์,การอินเตอร์รัพท์และฐานเวลาเป็นช่วง และตัวกำเนิดสัญญาณนาฬิกาทั่วไป

การโปรแกรม counter/timer ทำอย่างตรงไปตรงมา แต่ละ channel ถูกโปรแกรมด้วยคำสั่ง 4 byte การใช้งานในครั้งแรก แต่ละ channel จะนับลงและมีทางเลือกที่จะโหลดค่าเวลาคงที่นั้นอีกครั้งอย่างอัตโนมัติและเริ่มนับใหม่ ดังนั้นการเขียนโปรแกรมวนรอบเพื่อรอเวลาไม่จำเป็นต้องใช้งานต่อไป กระบวนการอินเตอร์รัพท์ทำได้ง่ายเพราะแต่ละ channel ใช้เวคเตอร์จาก Interrupt/Trap Vector Table

แต่ละ channel ถูกโปรแกรมเป็นอิสระโดยใช้รีจีสเตอร์ 3 ตัวคือ configuration Reg., control Reg. และ time-constant Reg. configuration Reg. ใช้เลือกโหมดในการทำงาน (counter หรือ timer) เลือกให้ทำงานหรือไม่ทำงาน channel interrupt และเลือกค่าพารามิเตอร์ในการทำงานอื่นๆ ในโหมดฐานเวลาสัญญาณนาฬิกาภายใน CPU ถูกหารด้วยสี่เพื่อเป็นอินพุทของ counter/timer ค่าของเวลาคงที่นี้จะมีค่าจาก 0 ถึง 65,535

ระหว่างการทำงาน counter channel แต่ละตัวจะลดค่าในรีจีสเตอร์ time-constant ลง ในการทำงานในโหมด counter ตัว counter จะนับลงเมื่อมีสัญญาณพัลส์เข้ามาจนกระทั่ง counter/timer พบเงื่อนไขที่ทำให้หยุด การนับลงแต่ละครั้งจะสัมพันธ์กับสัญญาณ clock ภายในโปรเซสเซอร์ สำหรับการนับที่มีค่ามากกว่า 65,536 สามารถใช้ counter 2 ตัวต่อ cascade กันได้ เมื่อ counter/timer พบเงื่อนไขที่ทำให้หยุด การนับลงของ counter จะรีเซตโดยอัตโนมัติกลับไปมีค่า time-constant ตามที่ได้ตั้งไว้

ในโหมด timer จะพิจารณาช่วงเวลาโดยไม่ต้องคิดเวลาที่ใช้ในโลจิกส่วนที่เพิ่มมาหรือซอฟท์แวร์ ที่ทำงานวนรอบ ช่วงของเวลาจะได้จากการหาร processor clock ภายในด้วยสี่ และลดค่าของ downcounter ที่ตั้งไว้ก่อน ดังนั้นช่วงเวลาคือผลรวมของคาบเวลาของ processor clock ที่ลดค่าไปสี่เท่า หารกับช่วงเวลาคงที่ที่กำหนดค่าไว้ก่อนใน downcounter ตัว timer ถูกทริกโดยการเซตค่า software trigger control bit ใน Control/Status Register หรือโดยอินพุทภายนอก

counter/timer ทั้ง 3 channel สามารถเกิดสัญญาณเอาท์พุทภายนอกเมื่อพบเงื่อนไข count/time output สัญญาณเอาท์พุทจะเป็น high เมื่อ downcounter ภายในที่กำหนดค่าได้เป็นศูนย์ทั้งหมด

แต่ละ channel สามารถโปรแกรมให้สร้างสัญญาณ Interrupt Request ซึ่งเกิดได้เพียงกรณีที่ Interrupt Enable control bit ใน channel นั้นถูกโปรแกรมซอฟท์แวร์เซตค่าเป็น 1 เมื่อ CPU Z280 ยอมรับการร้องขออินเตอร์รัพท์ก็จะชี้ไปที่ตาราง Interrupt Vector โดยอัตโนมัติ

Z280 ทั้ง 3 channel ถูกเรียงลำดับความสำคัญและความเหมาะสมเป็น slot ที่แตกต่างกัน 3 ช่องในโครงสร้างความสำคัญของการอินเตอร์รัพท์ของอุปกรณ์สนับสนุนภายใน โดยที่ channel 0 มีลำดับความสำคัญสูงสุดและ channel 2 มีลำดับความสำคัญต่ำสุด แต่ละ channel มีการแยกการทำงานของการอินเตอร์รัพท์และ CPU's Master Status Register มี control bit เฉพาะที่ใช้ในการเลือกการอินเตอร์รัพท์สำหรับแต่ละ channel

โหมดการทำงาน

counter/timer channel มีการทำงานพื้นฐานอยู่ 2 โหมดคือ ตัวนับ(counter) หรือ ฐานเวลา(timer) ในโหมดตัวนับจะตรวจสอบสัญญาณอินพุทภายนอกและบันทึกสภาวะการเปลี่ยนแปลงจาก Low เป็น High ของสัญญาณ ในโหมดฐานเวลา สัญญาณ processor clock ที่ถูกหารด้วยสี่จะถูกใช้แทนที่สัญญาณอินพุทจากภายนอก ในระหว่างการนับหรือฐานเวลาสามารถให้ทำงานต่อเนื่องโดยใช้การทริก หรือทำงานในช่วงเวลาที่ระบุโดยใช้ gate และ gate/trigger operation การนับสามารถเริ่มต้นใหม่อัตโนมัติโดยการโปรแกรม Retrigger Enable control bit ใน counter/timer's Configuration Register

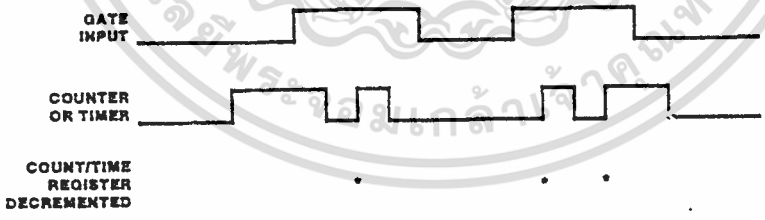
counter/timer แต่ละ channel มี software gate และ trigger ที่ง่ายต่อการขยายความสามารถทางฮาร์ดแวร์ของ counter/timer

Counting Operation ในขณะที่พบเงื่อนไขการทำงานที่เหมาะสม counter/timer จะตรวจสอบสัญญาณอินพุทที่เข้ามาจากสภาวะ Low เป็น High เมื่อสภาวะนี้เกิดขึ้น Count/Time Register ถูกลดค่าลงหนึ่ง

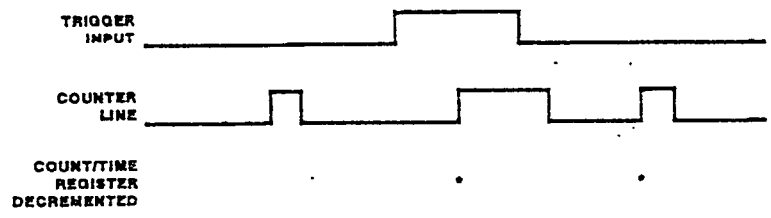
Timing Operation ในขณะที่พบเงื่อนไขที่เหมาะสมถูก counter/timer จะตรวจสอบ processor clock ภายในที่ถูกหารด้วยสี่จากสภาวะ Low เป็น High เมื่อสภาวะนี้เกิดขึ้น Count/Time Register ถูกลดค่าลงหนึ่ง

Gate Operation counter/timer สามารถโปรแกรมให้นับหรือเป็นฐานเวลาเมื่อพบ gating condition เท่านั้น ในขณะที่ counter/timer ทำงานและเลือกให้ gate ภายนอกสามารถทำงาน สัญญาณอินพุทภายนอกจะถูกตรวจสอบในขณะที่สัญญาณนี้เป็น High เท่านั้นที่มีการทำงานในการนับและฐานเวลา software gate ช่วยให้ลดสถานะของสัญญาณอินพุท ในขณะที่ software gate bit ใน Command and Status Register ถูกเคลียร์เป็น 0 สถานะของ gate จะไม่ถูกพบโดยไม่ต้องคำนึงถึงสัญญาณบน gating line การใช้ gate facility อธิบายในรูปที่ 1.31

Trigger Operation counter/timer สามารถโปรแกรมให้นับหรือเป็นฐานเวลาหลังจากพบ triggering condition เท่านั้น ในขณะที่ counter/timer ทำงานและความสามารถทริกจากภายนอกถูกโปรแกรม สัญญาณอินพุทภายนอกถูกตรวจสอบ หลังจากสัญญาณนี้มีสถานะจาก Low เป็น High เท่านั้นที่ทำให้การนับและฐานเวลาทำงาน software trigger facility ทำให้สภาวะทริกถูกพบและจะไม่สนใจการแอคทีฟของสัญญาณนี้อีกต่อไป การทริกมีรูปประกอบในรูปที่ 1.32



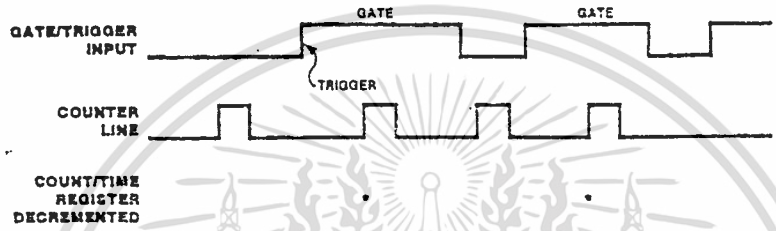
รูปที่ 1.31 Gate Facility



รูปที่ 1.32 Trigger Operation

Gate/Trigger Operation สัญญาณอินพุทหนึ่งเส้นสามารถใช้ได้ทั้งหน้าที่ gating และ triggering สถานะจาก Low เป็น High ของสัญญาณเส้นนี้ทำตัวเหมือนเป็น trigger และต่อมาเมื่อมีสถานะ High บนสัญญาณเส้นนี้จะคล้ายมีสัญญาณ gate ถ้าโหมดทริกซ้ำไม่ได้ถูกโปรแกรม สถานะจาก Low ไปสู่ High ในภายหลังไม่ทำให้เกิด trigger การปฏิบัติการ Gate/Trigger แสดงได้ในรูปที่ 1.33

กลไกของ gate และ trigger ทางซอฟท์แวร์สามารถใช้ทำงานในโหมดนี้ได้ การใช้ gate ทางซอฟท์แวร์ก่อนการ trigger (ฮาร์ดแวร์หรือซอฟท์แวร์)ไม่มีผลต่อ counter/timer หลังจากมีการ trigger ทาง hardware หรือ software แล้ว software gate จำเป็นต้องถูกเซตเป็น 1 เพื่อให้ Count/Time Register ถูกลดค่าลงหนึ่ง software trigger หลังจากมีการ trigger ทาง hardware หรือ software แล้วจะไม่มีผลยกเว้น Retrigger Enable control bit ถูกเซตเป็น 1

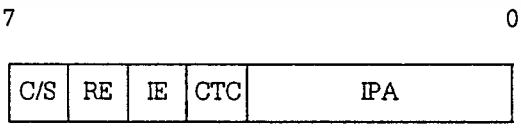


รูปที่ 1.33 Gate/Trigger Operation

Counter/Timer Control and Status Registers

Counter/timer แต่ละตัวมี control Register 8 บิท 2 ตัวและ count Register 16 บิท 2 ตัว Configuration Register และ Command/Status Register กำหนดการทำงานของ counter/timer โดยที่ Counter/Timer Command/Status Register ให้ข้อมูลเกี่ยวกับการทำงานในปัจจุบัน Time Constant Register มีค่าเริ่มต้นสำหรับ counter/timer และ Count/Time Register มีค่าของการนับในปัจจุบัน

Counter/Timer Configuration Register เป็นรีจีสเตอร์ขนาด 8 บิท (รูปที่ 1.34) ที่ระบุโหมดการทำงานของ counter/timer โครงสร้างการจัดขา ถ้าการร้องขออินเตอร์รัพท์เกิดขึ้นและลำดับการนับลงจะถูกเริ่มต้นใหม่โดยอัตโนมัติ เมื่อการนับนับถึงศูนย์หรือเมื่อเกิดการ trigger



รูปที่ 1.34 Counter/Timer Configuration Register

แต่ละ field ในรีจีสเตอร์นี้ประกอบด้วย

Input Pin Assignments (IPA) 4 บิทนี้ใช้ระบุการทำงานของสัญญาณอินพุทที่ใช้เกี่ยวข้องกับ counter/timer ว่าจะให้ counter/timer ตรวจสอบสัญญาณอินพุทภายนอก (counting operation) หรือใช้ processor clock ภายในที่มีการหารความถี่ (timing operation) 4 บิทนี้สามารถเกี่ยวข้องกับการกำเนิดสัญญาณเอาท์พุท (EO), การเลือกสัญญาณจากภายนอกหรือสัญญาณนาฬิกาภายใน(C/T), สามารถใช้ gating facility (G) และสามารถใช้ triggering facility (T) ตัวเลือกที่ได้เลือกจะไปกำหนดฟังก์ชันที่มีส่วนสัมพันธ์กับสัญญาณอินพุทแต่ละอันที่เกี่ยวข้องกับ counter/timer ดังแสดงในตารางที่ 1.8

IPA Field				Pin Functionality		
EO	C/T	G	T	Counter/Timer I/O	Counter/Timer Input	Notes
0	0	0	0	Unused	Unused	Timer
0	0	0	1	Unused	Trigger	Timer
0	0	1	0	Gate	Unused	Timer
0	0	1	1	Gate	Trigger	Timer
0	1	0	0	Unused	Input	Counter
0	1	0	1	Trigger	Input	Counter
0	1	1	0	Gate	Input	Counter
0	1	1	1	Gate/trigger	Input	Counter
1	0	0	0	Output	Unused	Timer
1	0	0	1	Output	Trigger	Timer
1	0	1	0	Output	Gate	Timer
1	0	1	1	Output	Gate/trigger	Timer
1	1	0	0	Output	Input	Counter
1	1	0	1	Unused	Unused	Reserved
1	1	1	0	Unused	Unused	Reserved
1	1	1	1	Unused	Unused	Reserved

ตาราง 1.8 Input Pin Functionality

Counter/Timer Cascade (CTC) เมื่อบิทนี้ถูกเซตเป็น 1 counter/timer 0 และ 1 จะสร้าง counter ขนาด 32 บิท เมื่อใช้เป็น 32 bit counter/timer ค่าใน Configuration Register และ Command/Status Register สำหรับ Counter/Timer 0 จะไม่ถูกสนใจยกเว้นค่า IE,CTC,EO,CIP,CC และ COR บิท CTC ใน Counter/Timer Configuration Register ของ counter/timer 1 และ 2 ไม่เคยถูกใช้เลย

Interrupt Enable (IE) ในขณะที่บิทนี้ถูกเซตเป็น 1 counter/timer จะร้องขออินเตอร์รัพท์เมื่อเกิดเงื่อนไข count/time ในขณะที่บิทนี้เป็น 0 จะไม่มีการร้องขออินเตอร์รัพท์เกิดขึ้น

Retrigger Enable (RE) ในขณะที่บิทนี้ถูกเซตเป็น 1 ค่า time constant จะถูกโหลดอย่างอัตโนมัติไปที่ Count/Time Register เมื่อได้รับ trigger input ในขณะที่ counter/timer กำลังทำการนับลง ในขณะที่บิทนี้เป็น 0 จะไม่มีการโหลดค่าใหม่เกิดขึ้น

Continuous/Single Cycle (C/S) ในขณะที่บิทนี้ถูกเซตเป็น 1 ลำดับในการนับลงจะถูกเริ่มต้นใหม่อย่างอัตโนมัติเมื่อมีการนับถึงศูนย์โดยการโหลดค่า time constant ไปที่ Count/Time Register ในขณะที่บิทนี้เป็น 0 จะไม่มีการโหลดค่าใดๆ

Counter/Timer Command/Status Register เป็นรีจีสเตอร์ขนาด 8 บิท (รูปที่ 1.35) ที่ให้ซอฟท์แวร์ใช้ควบคุมการทำงานของ counter/timer และสะท้อนสถานะปัจจุบันของการทำงานของ counter/timer control bit ในรีจีสเตอร์นี้ทำให้ counter/timer ทำงานและทำให้ gate และ trigger ทางซอฟท์แวร์สามารถทำได้ status bit ระบุว่าในขณะการนับดำเนินอยู่นั้น มีเงื่อนไขการเกิด count/time output หรือเงื่อนไขการเกิดในครั้งที่สอง

รูปที่ 1.35 Counter/Timer Command/Status Register

แต่ละ field ในรีจีสเตอร์นี้ประกอบด้วย

Count Overrun (COR) เมื่อบิทนี้ถูกเซตเป็น 1 จะเกิดเงื่อนไข count/time output และบิท CC ถูกเซตเป็น 1 เพื่อที่จะระบุเงื่อนไข count overrun ในขณะที่บิทนี้ถูกเซตเป็น 0 ไม่เกิดเงื่อนไข count/time output และบิท CC ไม่ถูกเซตหลังจากที่บิท CC ถูกเคลียร์โดยซอฟท์แวร์ บิทนี้สามารถที่จะอ่านหรือเขียน (เซตหรือเคลียร์) โดยคำสั่ง I/O

Count/Time Output Condition has been Met (CC) เมื่อบิทนี้ถูกเซตเป็น 1 Count/Time Register จะถูกลดค่าลงไปสู่ศูนย์โดยวงจรควบคุม counter/timer ในโหมด single cycle เมื่อบิทนี้ถูกเคลียร์เป็น 0 การนับจะไม่พบเงื่อนไข count/time output ซึ่งบิทนี้ได้ถูกเคลียร์โดยซอฟท์แวร์ไปแล้ว บิทนี้สามารถถูกอ่านหรือถูกเขียน (เซตหรือเคลียร์) โดยคำสั่ง I/O

Count in Progress (CIP) ในขณะที่บิทนี้ถูกเซตเป็น 1 counter/timer กำลังทำงานและ Count/Time Register ไม่ใช่ค่า 0 ในขณะที่บิทนี้ถูกเคลียร์เป็น 0 counter/timer จะไม่ทำงาน บิทนี้ถูกควบคุมโดยวงจรควบคุม counter/timer บิทนี้สามารถอ่านโดยการอ่าน I/O แต่ไม่สามารถถูกเซตหรือเคลียร์โดยคำสั่งเขียน I/O

Software Trigger (TG) เมื่อบิทนี้ถูกเซตเป็น 1 (และ trigger operation ของ counter/timer ทำงาน) และ Enable bit ถูกเซตเป็น 1 ด้วย trigger operation จะทำงานที่ขอบขาขึ้นของคาบเวลาแรกของ processor clock หลังจากที่ค่าเดิมถูกเคลียร์มาก่อนหน้านี้แล้ว ดังนั้นถ้าไม่เกิด hardware trigger ข้อมูลของ Time Constant Register จะถูกโหลดเข้าสู่ Count/Time Register และเริ่มต้นลำดับการนับลง ถ้าเกิด hardware trigger และ Retrigger Enable ถูกเซตเป็น 1 counter/timer จะถูกทริกซ้ำ การเขียนค่า

1 ในบิทนี้เมื่อค่าเดิมคือ 1 จะไม่มีผลในการทำงานของ counter/timer เมื่อบิทนี้ถูกเคลียร์เป็น 0 จะทำให้บิทนี้ไม่มีผลกับการทำงานของ counter/timer

Software Gate (GT) เมื่อบิทนี้ถูกเซตค่าเป็น 1 (และ gate operation ของ counter/timer ทำงาน) และ Enable bit ถูกเซตค่าเป็น 1 ด้วย การทำงานจะเริ่มต้นที่ขอบขาขึ้นของคาบเวลาแรกของ processor clock หลังจากที่ค่าเดิมในบิทนี้ถูกเคลียร์เป็น 0 แล้ว การเขียนค่า 1 ในบิทนี้เมื่อค่าเดิมเป็น 1 ไม่มีผลต่อการปฏิบัติการของ counter/timer เมื่อบิทนี้ถูกเคลียร์เป็น 0 การนับลงจะหยุดชงัก

Enable (E) ในขณะที่บิทนี้ถูกเซตค่าเป็น 1 counter/timer จะทำงาน การทำงานจะเริ่มขึ้นที่ขอบขาขึ้นของคาบเวลาแรกของ processor clock โดยค่าก่อนหน้านี้ต้องถูกเคลียร์เป็น 0 แล้ว การรีเซตจะเคลียร์ค่าบิทนี้ ในขณะที่บิทนี้ถูกเคลียร์เป็น 0 ค่าใน Time Constant Register ถูกถ่ายเทไปที่ Count/Time Register ถ้าค่าใน Time Constant Register เป็นศูนย์ทั้งหมด output ของ counter/timer จะเป็น 1 ดังนั้นเมื่อ counter/timer ไม่ทำงาน counter/timer output พร้อมกับ Time Constant Register สามารถถูกใช้เป็น I/O พอร์ทได้ การเขียนค่า 1 ในบิทนี้เมื่อค่าก่อนหน้านี้เป็นหนึ่งที่ไม่มีผลกับการทำงานของ counter/timer ในขณะที่บิทนี้มีค่าเป็น 0 counter/timer จะไม่มีการปฏิบัติการในคาบเวลาถัดไปของ processor clock

Time Constant Register เป็นรีจีสเตอร์ขนาด 16 บิทที่เก็บค่าซึ่งถูกโหลดไปที่ Count/Time Register โดยอัตโนมัติ เมื่อ counter/timer ทำงานหรืออยู่ในโหมด continuous หรือ retrigger เมื่อนับถึงศูนย์หรือการ trigger ถูกยืนยันตามลำดับ รีจีสเตอร์นี้สามารถที่จะถูกอ่านหรือถูกเขียนโดยคำสั่ง I/O

Count/Time Register เป็นรีจีสเตอร์ขนาด 16 บิทที่เก็บค่าการนับหรือฐานเวลาที่ดำเนินอยู่ในปัจจุบัน ค่านี้จะถูกโหลดอย่างอัตโนมัติจาก Time Constant Register และสามารถถูกอ่านโดยซอฟท์แวร์โดยใช้คำสั่งการอ่าน I/O

Pin Description

counter/timer มีสัญญาณอินพุทภายนอกที่เกี่ยวข้องสองสัญญาณ สัญญาณ I/O รับส่งข้อมูลระหว่าง counter/timer และ อุปกรณ์ภายนอกอื่นๆ ขาอินพุทรับสัญญาณจากอุปกรณ์ภายนอกสำหรับ counter/timer การตีความของสัญญาณในขานี้พิจารณาจากส่วน Input Assignment ใน Configuration Register

1.18 การเชื่อมต่อภายนอกแบบ Z-Bus

คุณสมบัติ

- data bus ขนาด 16 bit
- มีการ multiplex สัญญาณ address และ data
- รองรับการเคลื่อนย้ายข้อมูลความเร็วสูงในโหมด burst
- สามารถต่อกับระบบ EPA ได้

การจัดระบบขา

A_{16}-A_{23} Address (เป็นขา 3-state output แอคทีฟที่ high) ใช้ระบุแอดเดรสของ I/O และหน่วยความจำ ขณะติดต่อกับบัส

AD_0-AD_{15} Address/Data (เป็น 3-state แบบ 2 ทิศทางแอคทีฟที่ high) มีการ multiplex สัญญาณระหว่าง address กับ data เพื่อที่จะระบุแอดเดรสของ I/O แอดเดรสของหน่วยความจำและข้อมูล

$\overline{AS}$ Address Strobe (เป็นขา 3-state output แอคทีฟที่ low) ที่ขอบขาขึ้นของสัญญาณ Address Strobe จะเป็นการเริ่มต้นของการติดต่อกับบัส และแสดงว่ามีสัญญาณ address, status, R/W และ B/W เรียบร้อยแล้ว

$\overline{BUSACK}$ Bus Acknowledge (เป็นขา output แอคทีฟที่ low) เมื่อมีสถานะเป็น low แสดงว่าซีพียูเลิกควบคุมบัส เพื่อตอบสนองการขอใช้บัสของอุปกรณ์

$\overline{BUSREQ}$ Bus Request (เป็นขา input แอคทีฟที่ low) เมื่อขานี้เป็น low แสดงว่ามีการขอใช้บัสจากอุปกรณ์ภายนอก

B/W Byte/Word (เป็นขา 3-state output) ถ้าสัญญาณที่ขานี้มีสถานะเป็น low จะส่งข้อมูลเป็นแบบ word (16 บิท) ถ้าเป็น high จะส่งข้อมูลเป็นแบบ byte (8 บิท)

CLK Clock Output (เป็นขา output) ความถี่การทำงานภายในโปรเซสเซอร์ คือความถี่ได้จากออสซิลเลเตอร์ภายนอกหรือคริสตอลหารด้วยสอง หลังจากนั้นความถี่การทำงานภายในโปรเซสเซอร์จะถูกหารด้วย 1,2 หรือ 4 ซึ่งแล้วแต่การโปรแกรม ได้เป็นความถี่ที่ออกจากขา CLK

CTIN Counter/Timer input (เป็นขา input ทำงานที่สภาวะเป็น high) ขานี้จะรับสัญญาณภายนอกสำหรับใช้ใน counter/timer

CTIO Counter/timer I/O (เป็นขา 3-state แบบสองทิศทาง ทำงานที่สภาวะเป็น high) เป็นขาที่ถ่ายทอดสัญญาณ I/O ระหว่าง counter/timer และอุปกรณ์ภายนอก

$\overline{DMASTB}$ DMA Flyby Strobe (เป็นขา output ทำงานที่สภาวะเป็น Low) เป็นขาที่ใช้เลือกอุปกรณ์เสริมภายนอกสำหรับการถ่ายเทข้อมูลของ DMA ในโหมด flyby

$\overline{DS}$ Data Strobe (เป็นขา 3-state output แอคทีฟที่ low) เป็นสัญญาณที่กำหนดการเคลื่อนย้ายข้อมูล เข้า-ออกจาก bus master

$\overline{EOP}$ End of Process (เป็นขา input ทำงานที่สภาวะเป็น Low) อุปกรณ์ภายนอกสามารถหยุดการทำงานของ DMA โดยให้ขานี้เป็น Low EOP จะใช้กับ DMA channel ที่ทำงานเท่านั้น ถ้าไม่มี channel ไหนทำงาน สัญญาณ EOP จะไม่มีผล

$\overline{GACK}$ Global Acknowledge (เป็นขา input ทำงานที่สภาวะเป็น Low) เมื่อขานี้เป็น Low แสดงว่า CPU ยอมให้ global bus มาควบคุม

$\overline{GREQ}$ Global Request (เป็นขา 3-state output ทำงานที่สภาวะเป็น Low) เมื่อขานี้เป็น Low แสดงว่า CPU ได้ทำการควบคุมหรือพยายามเข้าควบคุม global bus

$\overline{IE}$ Input Enable (เป็นขา 3-state output ทำงานที่สภาวะเป็น Low) เมื่อขานี้เป็น Low แสดงว่า Address หรือข้อมูลถูกเคลื่อนย้ายไปที่ซีพียู

$\overline{INT}$ Maskable Interrupt (เป็นขา input ทำงานที่สภาวะเป็น Low) เมื่อขานี้เป็น Low แสดงว่ามีการร้องขอการ Interrupt

$\overline{NMI}$ Nonmaskable Interrupt (เป็นขา input ทำงานที่ขอบขาลงของ clock) เมื่อมีการเปลี่ยนแปลงสัญญาณจาก high ไปเป็น low แสดงว่าเกิด nonmaskable interrupt

$\overline{OE}$ Output Enable (เป็นขา 3-state output ทำงานที่สภาวะ Low) เมื่อขานี้เป็น Low แสดงว่า Address/Data ถูกเคลื่อนย้ายออกจาก CPU

OPT Bus Option (เป็น input) ขานี้จะกำหนดชนิดของบัส เมื่อมีการรีเซท โดยมีสถานะดังนี้

OPT	Bus Interface
0	Z80 Bus,8 บิท
1	Z-Bus,16 บิท

$\overline{PAUSE}$ CPU PAUSE (เป็นขา input ทำงานที่สภาวะ Low) เมื่อขานี้เป็น Low ซีพียูจะงดการเคลื่อนย้ายข้อมูลเข้าหรือออกจาก EPU หรืองดการเริ่มทำคำสั่งใหม่

$\overline{RDY}$ DMA Ready (เป็นขา input ทำงานที่สภาวะ Low) ขานี้ถูกตรวจสอบโดย channel ใน DMA ว่าเมื่อไรอุปกรณ์ภายนอกที่ต่อกับ DMA พร้อมที่จะอ่านหรือเขียน เมื่อ DMA channel พร้อมจะทำงาน สัญญาณ ready จะไม่ได้ควบคุมกิจกรรม DMA โดยตรง แต่จะขึ้นกับโหมดการทำงาน

$\overline{RESET}$ Reset (เป็นขา input ทำงานที่สภาวะ Low) เมื่อขานี้เป็น Low จะทำการรีเซท CPU และอุปกรณ์บนชิป

R/$\overline{W}$ Read/Write (เป็นขา 3-state output) เป็นสัญญาณที่บอกถึงทิศทางของการเคลื่อนย้ายข้อมูลของหน่วยความจำ, I/O หรือ EPU ถ้ามีสถานะเป็น low จะเป็นการเขียน ถ้าเป็น high จะเป็นการอ่าน

RxD UART Receive (เป็นขา input ทำงานที่สภาวะ high) ขานี้จะรับข้อมูลอนุกรมในระดับแรงดัน TTL มาตรฐานเข้าในซีพียู

ST_0-ST_3 Status (เป็นขา 3-state output ทำงานที่สภาวะ high) สายสัญญาณทั้งสี่เส้นนี้บอกชนิดของการจัดการที่เกิดขึ้นบนบัสและให้ข้อมูลเพิ่มเติมเกี่ยวกับการจัดการบัส

TxD UART Transmit (เป็นขา output ทำงานที่สภาวะ high) ขานี้จะส่งข้อมูลอนุกรมออกจากซีพียูในระดับแรงดัน TTL มาตรฐาน

$\overline{WAIT}$ Wait (เป็นขา input ทำงานที่สภาวะ Low) ขณะที่ขานี้เป็น Low แสดงว่ามีอุปกรณ์ที่ต้องการใช้เวลาเพิ่มขึ้นในการรับส่งข้อมูล เนื่องจากซีพียูมีความเร็วสูงกว่าการทำงานของตัวอุปกรณ์

XTALI Clock/Crystal Input (time-base input) สามารถนำเอาคริสตอลมาต่อขนานหรือแหล่งกำเนิดความถี่ภายนอกป้อนเข้าเพื่อเป็นความถี่ในการทำงานของ CPU

XTALO Clock/Crytal Output ใช้ต่อขนานกับ Crystal เพื่อใช้กับออสซิลเลเตอร์ภายใน

+5V Power Supply Voltage (ปกติจะมีค่า +5V)

GND Ground กราวนด์อ้างอิง

Bus Operation

ในระบบบัส จะมีการทำงานได้ 2 อย่าง คือ การจัดการ (Transaction) และ การร้องขอ (Request) ในขณะใดขณะหนึ่งจะมีอุปกรณ์เพียงตัวเดียวที่ควบคุมบัส ซึ่งอาจจะเป็น CPU หรืออุปกรณ์อื่นก็ได้ ซึ่งการเกิด transaction มี 8 ชนิดคือ

1. Burst Memory จะเป็นการเคลื่อนย้ายคำสั่งจำนวน 4 word จากหน่วยความจำไปยัง CPU
2. DMA Flyby เป็นการเคลื่อนย้ายข้อมูลของอุปกรณ์ DMA ระหว่างหน่วยความจำและอุปกรณ์ภายนอก
3. EPU Transfer เป็นการเคลื่อนย้ายข้อมูลระหว่าง CPU กับ EPU
4. Halt ทำให้ซีพียูอยู่ในสภาวะหยุดการทำงาน
5. Interrupt Acknowledge CPU จะรับรู้ถึงการอินเตอร์รัพท์และตอบสนองการอินเตอรัพท์ต่ออุปกรณ์ที่ร้องขออินเตอร์รัพท์นั้น
6. I/O เป็นการเคลื่อนย้ายข้อมูลเข้าหรือออกจากอุปกรณ์ภายนอก
7. Memory เป็นการเคลื่อนย้ายข้อมูลเข้าหรือออกจากหน่วยความจำ
8. Refresh เป็นการรีเฟรชหน่วยความจำแบบไดนามิค

การร้องขอ (request) มี 2 ชนิด

1. Bus เป็นการขอใช้บัสเพื่อเริ่มการ transaction
2. Interrupt เป็นการร้องขอบริการจาก CPU

เมื่อเกิดอินเตอร์รัพท์หรือการขอใช้บัสเกิดขึ้น ผลจะเกิดตามชนิดที่ขอ ถ้าเป็นการขออินเตอร์รัพท์จากภายนอก การตอบรับอินเตอร์รัพท์จะเริ่มต้นค่าโดยซีพียู แต่ถ้ามีการขอใช้บัส ไมโครโปรเซสเซอร์จะตัดบัสออกจากระบบและส่งสัญญาณตอบรับออกไป

Transaction

การ transaction จะเริ่มขึ้นเมื่อสัญญาณ $\overline{AS}$ มีค่าเป็น Low และกลับเป็น High อีกครั้ง ในช่วงขอบขาขึ้นนี้สัญญาณ ST_0-ST_3, R/W และ B/W ที่ส่งออกมาสามารถนำค่านี้ไปใช้ได้แล้ว ชนิดของการ transaction จะแสดงโดยใช้ขา ST_0-ST_3 ตามตารางที่ 1.9

Status Line S3-S0	Type of Transaction
0000	Reserved
0001	Refresh
0010	I/O transaction
0011	Halt
0100	Interrupt acknowledge line A
0101	NMI acknowledge
0110	Interrupt acknowledge line B
0111	Interrupt acknowledge line C
1000	Transfer between CPU and memory,cacheable
1001	Transfer between CPU and memory,non-cacheable
1010	Data transfer between EPU and memory
1011	Reserved
1100	EPU Instruction fetch,template,subsequent words
1101	EPU Instruction fetch,template,first word
1110	Data transfer between EPU and CPU
1111	Test and Set (data transfers)

ตารางที่ 1.9 Status Code Table

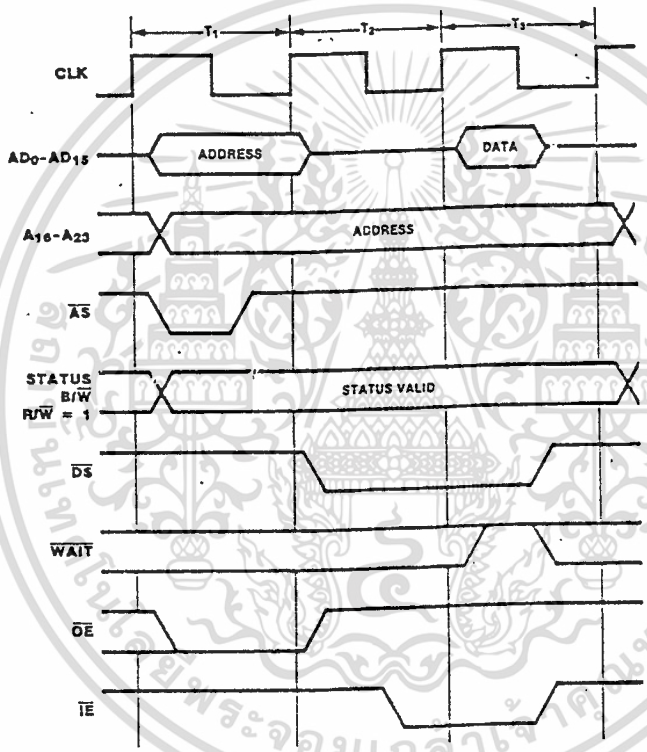
Wait Cycle เมื่อซีพียูอ่านข้อมูลเข้ามาจะมีการตรวจสอบสัญญาณ WAIT ที่ขอบขาลงของ clock ก่อนที่สัญญาณ $\overline{DS}$ จะมีค่าเป็น high ถ้าสัญญาณ WAIT เป็น Low จะมีสัญญาณ clock แทรกเพิ่มขึ้นมาทำให้เวลาของการ transaction ช้าลง เพื่อรอให้หน่วยความจำหรือ I/O มีความพร้อมที่จะทำการเคลื่อนย้ายข้อมูล การเพิ่มสถานะ WAIT ทำได้โดยการโปรแกรมที่รีจีสเตอร์ Bus Timing and Control และ Bus Timing and Initialization

Memory Transaction เป็นการเคลื่อนย้ายข้อมูลเข้าหรือออกจากหน่วยความจำ เมื่ออุปกรณ์ที่ควบคุมบัสที่เรียกว่า Bus Master มีการเข้าถึงหน่วยความจำ ถ้าไม่มีสัญญาณ WAIT การเคลื่อนย้ายข้อมูลจะใช้เวลา 3 บัสไซเกิล

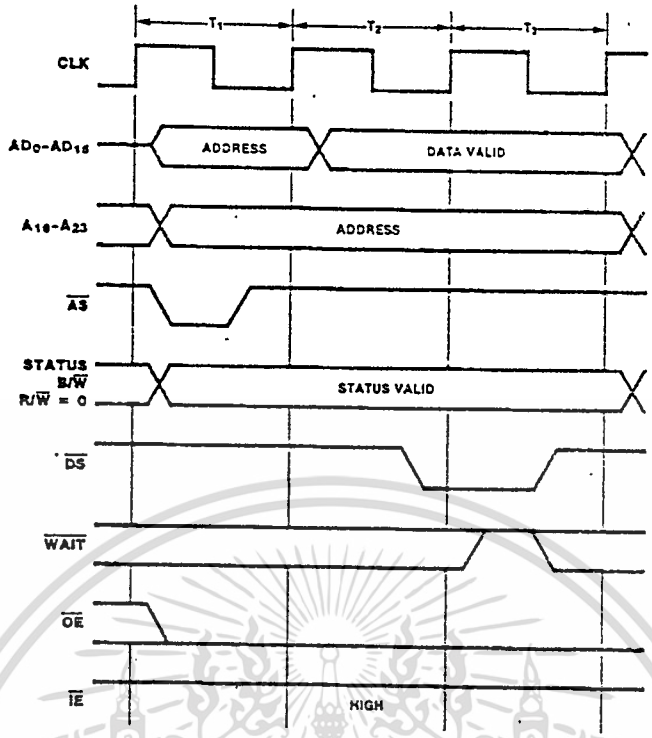
การเคลื่อนย้ายข้อมูลแบบ byte จากแอดเดรสคี่ของหน่วยความจำ ($AD_0 = 1$) จะทำการเคลื่อนย้ายข้อมูลผ่านทาง AD_0-AD_7(AD_0 เป็นบิทที่ศูนย์) การเคลื่อนย้ายข้อมูลแบบ byte จากแอดเดรสคู่ของหน่วยความจำ(AD_0 เป็นศูนย์) จะทำการเคลื่อนย้ายข้อมูลผ่านทาง AD_8-AD_{15} (AD_8 เป็นบิทที่ศูนย์) การอ่านแบบ byte ค่าสัญญาณ B/$\overline{W}$ และ R/$\overline{W}$ จะมีค่าเป็น high ส่วนการเขียนแบบ byte ค่าสัญญาณ B/$\overline{W}$ มีค่าเป็น high แต่สัญญาณ R/$\overline{W}$ มีค่าเป็น low ระหว่างที่ซีพียูทำการเขียนข้อมูลแบบ byte จะปรากฏข้อมูลที่เหมือนกัน 2 ชุดที่ A_0-A_{15}

การเคลื่อนย้ายข้อมูลแบบ word ค่า B/$\overline{W}$ เป็น Low โดยจะมีการเคลื่อนย้ายข้อมูลพร้อมกัน 16 บิท โดย AD_7-AD_0 เป็น byte สูง AD_{15}-AD_8 เป็น byte ต่ำ ค่าแอดเดรสต้องเป็นแอดเดรสคู่เท่านั้น

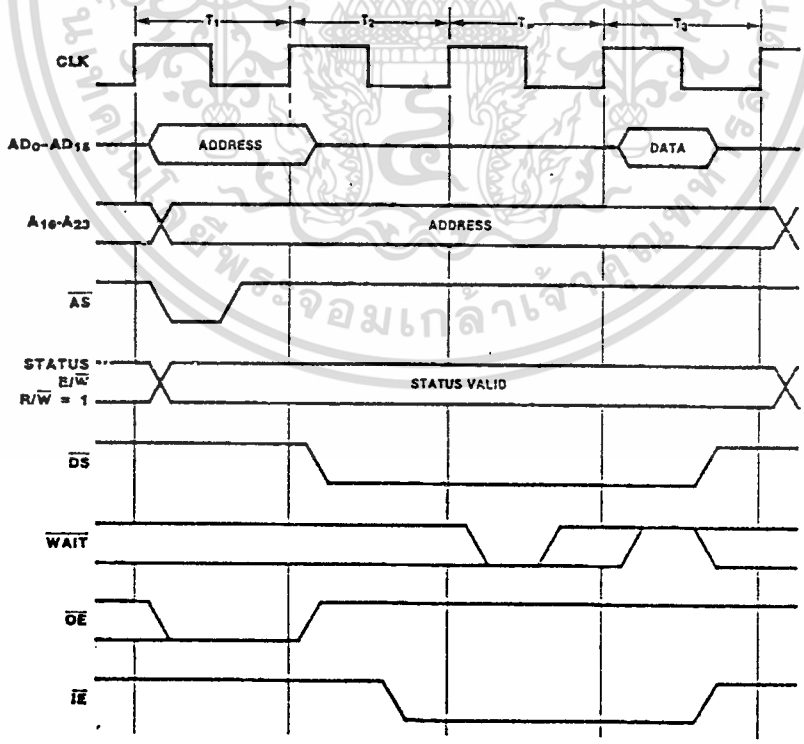
การจัดการหน่วยความจำสามารถแสดงได้ตาม timing ในรูป 1.46-1.50



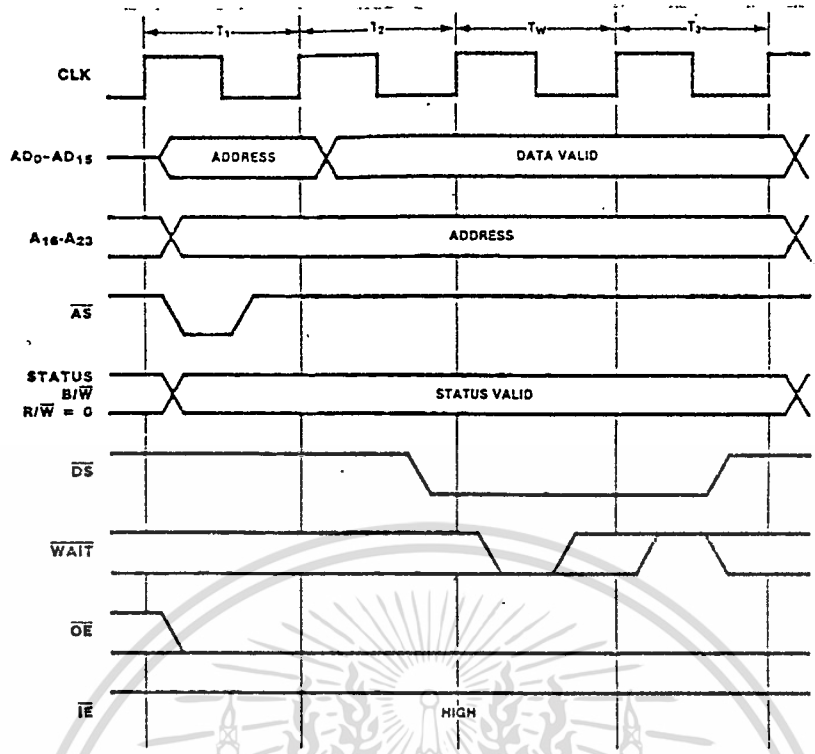
รูปที่ 1.46 Memory Read Timing



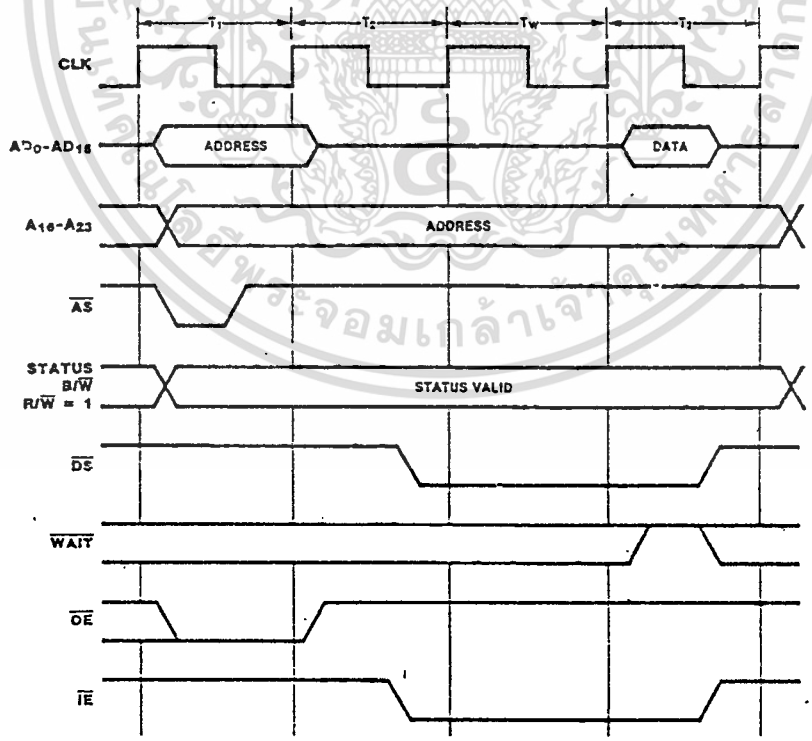
รูปที่ 1.47 Memory Write Timing



รูปที่ 1.48 Memory Read Timing with External Wait Cycle



รูปที่ 1.49 Memory Write Timing with External Wait Cycle



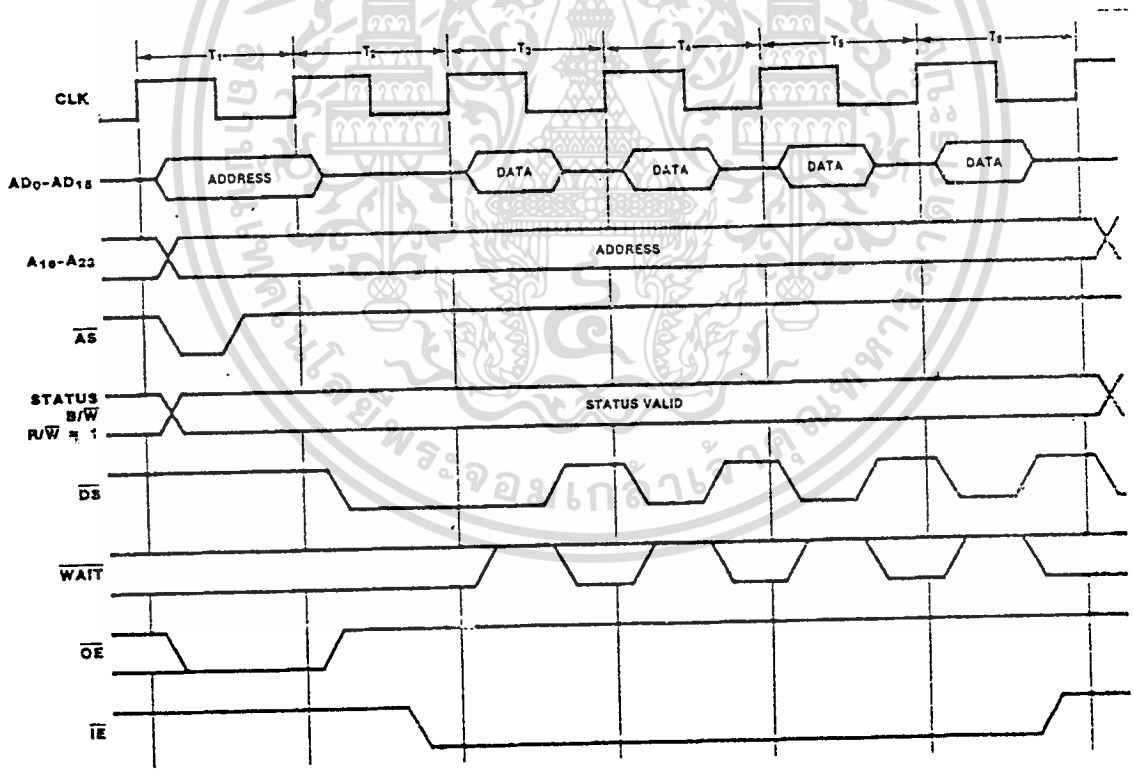
รูปที่ 1.50 Memory Read Timing with Internal Cycle

Burst Memory Transaction[1] เป็นการเคลื่อนย้ายข้อมูลหลายๆครั้งโดยมีการใช้สัญญาณ $\overline{AS}$ เพียงครั้งเดียว โดยเป็นการอ่านข้อมูลแบบ word จำนวน 4 ครั้ง แอดเดรสของเวิร์ดตัวแรกจะมีค่า 3 บิทที่มีนัยสำคัญต่ำสุดเป็นศูนย์ บิทควบคุมของ Cache Control register จะกำหนดว่าระบบหน่วยความจำจะรองรับ burst transaction หรือไม่

CPU จะใช้ burst transaction ในการเฟทช์คำสั่งเท่านั้น ถ้าคำสั่งที่จะเฟทช์อยู่ในหน่วยความจำที่สามารถทำ burst transaction ได้ ซีพียูจะอ่านข้อมูล 8 ไบท์ที่บรรจุคำสั่งไบท์แรกไว้ (การเฟทช์ EPA template จะไม่ใช้ burst transaction)

การโยกย้ายข้อมูลไบท์แรกของ burst transaction จะเหมือนกับการอ่านข้อมูลธรรมดา รวมทั้งในการทำงานของ wait state ด้วย ในการถ่ายเทข้อมูลแรก ถ้ามีสัญญาณ WAIT เกิดขึ้น ซีพียูก็จะไม่ถ่ายเทข้อมูลจนกว่าสัญญาณ WAIT จะหมดไป

Halt Transaction[2] จะไม่มีการถ่ายเทข้อมูล การทำงานจะคล้ายการติดต่อหน่วยความจำแต่จะไม่มีสัญญาณ $\overline{DS}$ เปลี่ยนเป็น High Halt Transaction เกิดขึ้นเมื่อ CPU ทำคำสั่ง HALT หรือเมื่อเกิดการ trap หรือการผิดพลาดของบัส แอดเดรสที่ปรากฏจะเป็นแอดเดรสของคำสั่ง HALT หรือคำสั่งที่ทำให้เกิด trap หรือการผิดพลาดของบัส ST_0-ST_3 จะมีค่าเป็น 0011

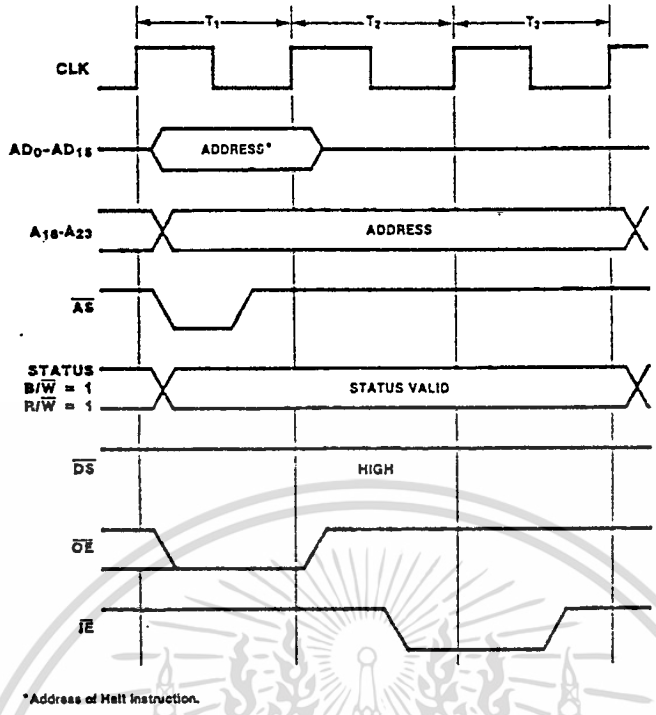


รูปที่ 1.51 Burst Memory Read Timing

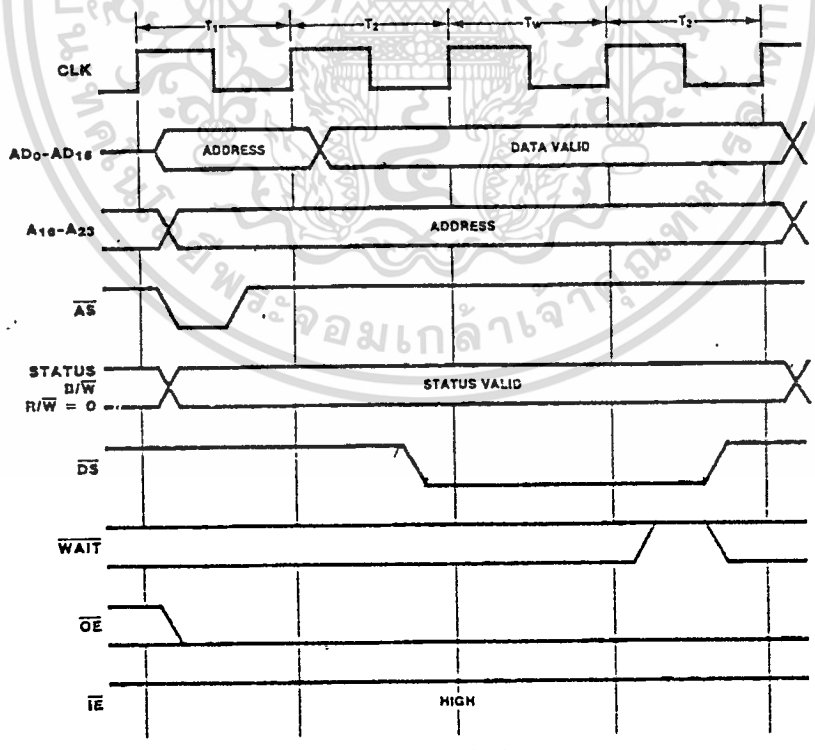
---

[1] Burst Transaction สามารถเกิดเฉพาะในโหมด Z-BUS เท่านั้น

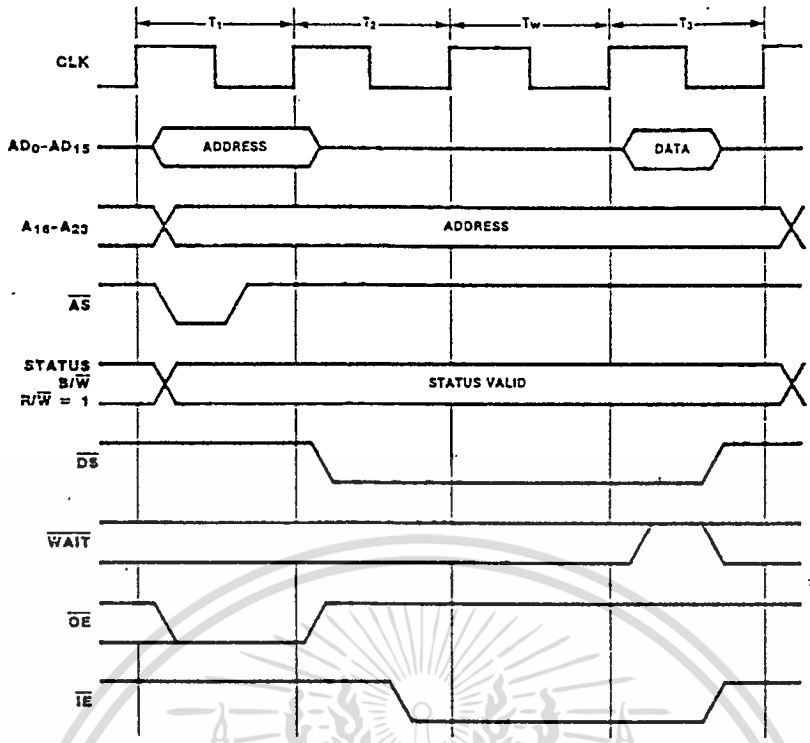
[2] สัญญาณ WAIT จะไม่มีผลระหว่างการเกิด Halt Transaction



รูปที่ 1.52 Halt Timing



รูปที่ 1.53 I/O Write Timing



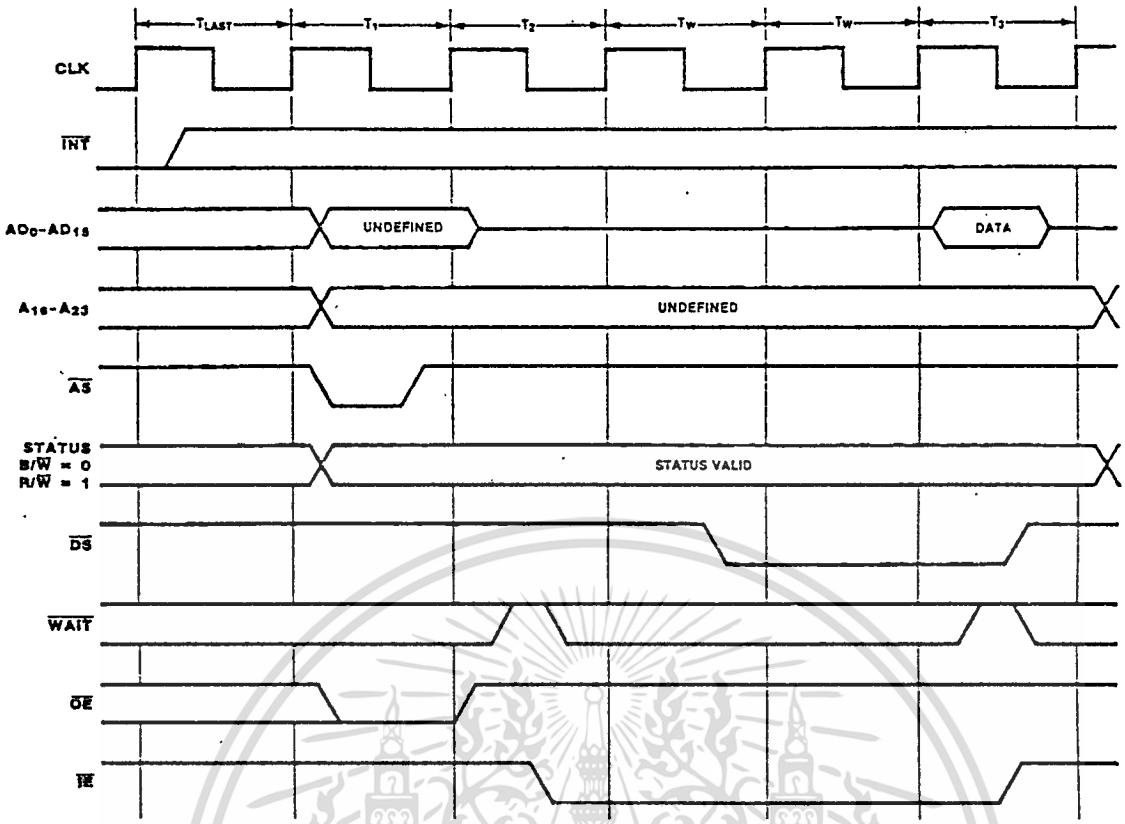
รูปที่ 1.54 I/O Read Timing

I/O Transaction จะเคลื่อนย้ายข้อมูลเข้าหรือออกจากอุปกรณ์สนับสนุนเกิดขึ้นเมื่อมีการทำคำสั่งเกี่ยวกับ I/O แต่การเคลื่อนย้ายข้อมูลกับอุปกรณ์บนซีพียู (I/O page FE_H และ FF_H) จะไม่ทำให้เกิดการติดต่อกับบัสภายนอก การจัดการ I/O ต้องใช้เวลาอย่างน้อย 4 บัสไซเคิลและอาจจะใช้เวลาเพิ่มถ้ามี wait state สัญญาณบนขาแสดงสถานะเมื่อมีการติดต่อกับ I/O จะเป็น 0010

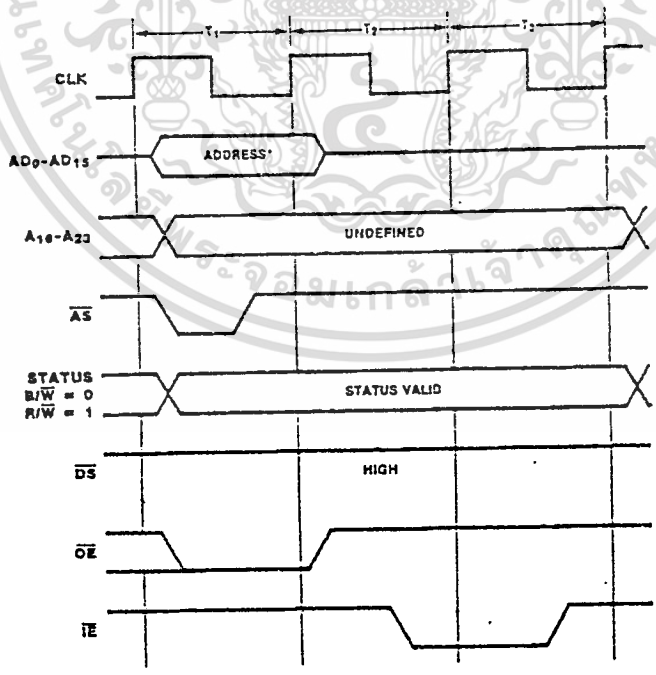
แอดเดรสของ I/O จะปรากฏที่ AD_0-AD_{15} และ A_{16}-A_{23} ในการส่งข้อมูลแบบ byte (B/$\overline{W}$ เป็น high) จะส่งผ่านทาง AD_0-AD_7 และจะให้อุปกรณ์ภายนอกติดต่อเพียง 8 บิทเท่านั้น ในการส่งแบบ word (B/$\overline{W}$ เป็น low) จะส่งไบท์สูงทาง AD_0-AD_7 และไบท์ต่ำทาง AD_8-AD_{15} I/O Transaction แสดงได้ดังรูปที่ 1.53-1.54

Interrupt Acknowledge Transaction (รูปที่ 1.55) เกิดขึ้นเมื่อมีการขออินเตอร์รัพท์จากอุปกรณ์ภายนอก การตอบรับอินเตอร์รัพท์ต้องใช้เวลาอย่างน้อย 5 cycle โดยเป็นช่วง WAIT 2 cycle ซึ่ง cycle แรกจะเป็นการจัดลำดับความสำคัญของการอินเตอร์รัพท์ cycle ที่สองจะเป็นการหน่วงเวลาเพื่อให้มีการอ่านข้อมูลเข้ามา โดยที่สถานะ wait อาจเพิ่มได้อีกโดยการโปรแกรมที่ Bus Timing and Control register ส่วนการถ่ายเทข้อมูลจะส่งออกก่อนที่ $\overline{DS}$ จะเปลี่ยนเป็น High

Refresh Transaction (รูปที่ 1.56) เกิดขึ้นเมื่อมีการรีเฟรชไดนามิคแรม แอดเดรสของการรีเฟรชซึ่งมีขนาด 10 บิทจะออกทางบัส 10 บิทล่างในไซเกิลแรกของการรีเฟรช โดยขณะนี้ ST_0-ST_3 มีค่า 0001 การรีเฟรชสามารถเกิดขึ้นได้แม้ว่าในขณะนั้นซีพียูจะอยู่ในสถานะ Halt หรือเกิดข้อผิดพลาดก็ตาม



รูปที่ 1.55 Interrupt Acknowledge Timing



*10 least-significant bits are Refresh address.

รูปที่ 1.56 Memory Refresh Timing

CPU-Extended Processing Unit Interaction

ไมโครโปรเซสเซอร์ Z280 ที่ใช้ Z-BUS สัญญาณ $\overline{\text{PAUSE}}$ และ Extended Processing Unit (EPU) หนึ่งตัวหรือมากกว่าจะทำงานร่วมกันได้เหมือนเป็นส่วนหนึ่งของซีพียู โดยที่ซีพียูได้เตรียม address, status และ timing signal ส่วน EPU จะให้และตรวจจับข้อมูล EPU จะตรวจสอบ status และ timing signal ที่ออกจากซีพียูเพื่อที่จะได้รู้ว่าจะต้องจัดการหน่วยความจำเมื่อใด สำหรับการเคลื่อนย้ายข้อมูลจาก EPU ไปหน่วยความจำ ซีพียูจะตัดตัวเองออกจากระบบโดยทำให้ขาข้อมูลเป็น 3 state ในขณะที่สัญญาณ $\overline{\text{DS}}$ เป็น Low ทำให้ EPU สามารถใช้บัสได้

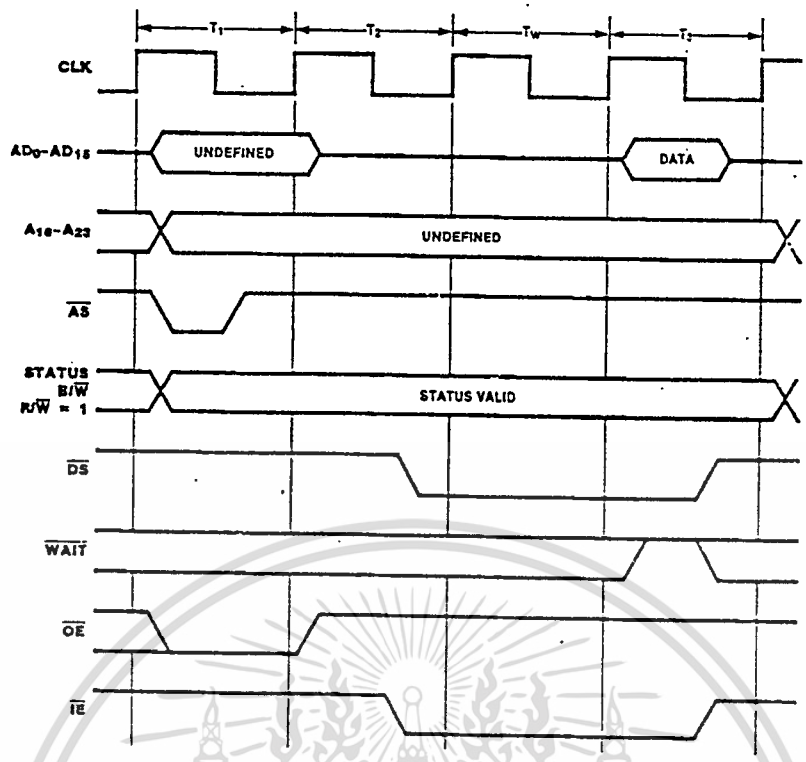
ในการจัดการที่ EPU มีส่วนร่วมในการจัดการบัสจะทำตามลำดับดังนี้

- ☐ เมื่อซีพียูเฟทช์ EPA instruction template word แรกจากหน่วยความจำ (ST_3-ST_0=1101) EPU จะตรวจจับคำสั่งที่หน่วยความจำคืนให้ โดยภายใน template ID field จะระบุว่า EPU จะปฏิบัติตามคำสั่งหรือไม่
- ☐ ถ้าไม่มีการรีเฟรชหน่วยความจำ ซีพียูจะเฟทช์คำสั่งที่สอง word ต่อไป (ST_3-ST_0=1100) EPU จำเป็นต้องตรวจจับ word นี้ ถ้า template ไม่ถูกต้อง จะมีการเฟทช์ครั้งที่สาม (ST_3-ST_0=1100)
- ☐ ถ้าคำสั่งทำให้เกิดการอ่านและการเขียนกับหน่วยความจำและขณะนั้นไม่มีการรีเฟรชหน่วยความจำแล้ว ซีพียูจะมีการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำและ EPU (ST_3-ST_0=1010) ตัว EPU จะต้องให้ข้อมูล (การเขียน:R/$\overline{\text{W}}$=Low) หรือตรวจจับข้อมูล (การอ่าน:R/$\overline{\text{W}}$=high) แต่ไม่ว่าจะเป็นแบบใด ซีพียูจะให้ขาข้อมูลเป็น 3-state ในขณะที่ข้อมูลถูกเคลื่อนย้าย ($\overline{\text{DS}}$ Low)
- ☐ ถ้าคำสั่งทำให้เกิดการเคลื่อนย้ายข้อมูลจาก EPU ไปที่ Z280 แล้ว ถ้าไม่มีการรีเฟรชหน่วยความจำจะมีการเคลื่อนย้ายข้อมูลระหว่าง EPU และซีพียู (ST_3-ST_0=1110)

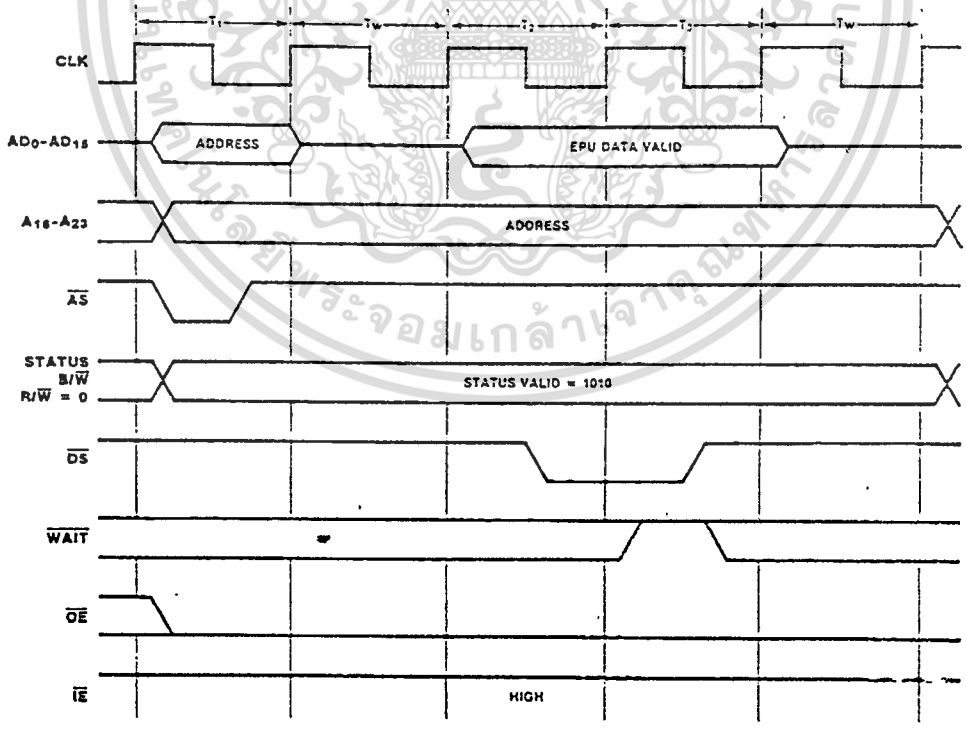
ในการทำตามลำดับนี้ EPU จะตรวจสอบขาสถานะเพื่อที่จะตรวจสอบว่าการจัดการบัสทำโดยซีพียู ในระบบที่มี EPU หลายตัวจะไม่มีการระบุบนบัสว่า EPU ตัวไหนทำงานกับซีพียูในเวลาที่ถูกให้มา ในกรณีนี้ EPUs จำเป็นต้องพิจารณาจาก template ที่รับมา

เมื่อ EPU เริ่มต้นทำคำสั่งเพิ่มเติม ซีพียูยังคงสามารถที่จะเฟทช์และเอ็กซิคิวคำสั่งต่อไปได้ ถ้า EPU ต้องการหยุดการทำคำสั่งของซีพียูหรือการจัดการบัสอื่นๆ EPU จะให้สัญญาณ $\overline{\text{PAUSE}}$ เพื่อหยุดซีพียูจนกระทั่ง EPU พร้อมต่อการทำงานของไมโครโปรเซสเซอร์ในภายหลัง กระบวนการนี้ถูกใช้ประสานจังหวะการทำงานของ MPU-EPU

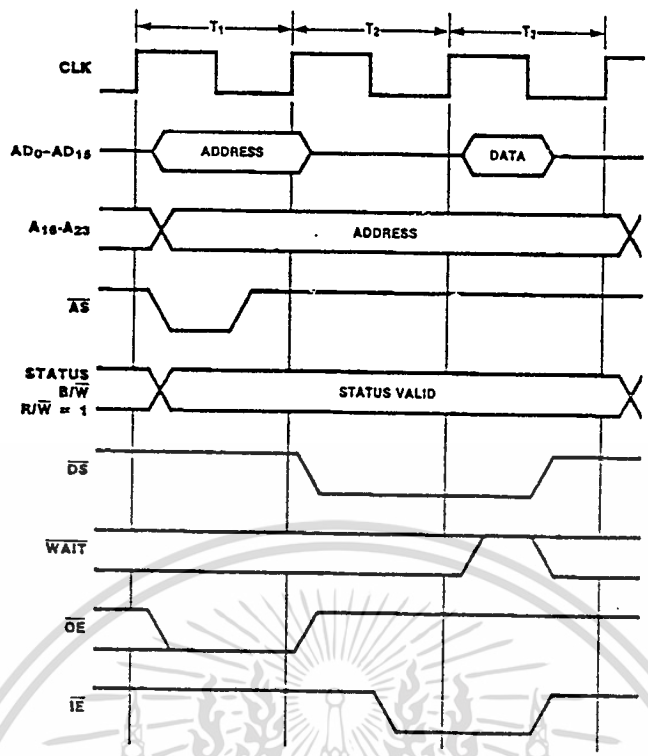
EPU Transfer Transaction (รูป 1.57-1.59) ทำให้ซีพียูสามารถเคลื่อนย้ายข้อมูลเข้าหรือออกจาก EPU หรือ อ่านหรือเขียนไปที่รีจีสเตอร์สถานะของ EPU โดยใช้คำสั่งของ EPU



รูปที่ 1.57 EPU to CPU Timing



รูปที่ 1.58 EPU Write to Memory



รูปที่ 1.59 Memory to EPU Timing

Request

Z280 รองรับสัญญาณร้องขอ 3 ชนิดคือ

- ☐ Interrupt request อุปกรณ์ตัวอื่นขอมา ซีพียูรับรู้และตอบรับ
- ☐ Bus request อุปกรณ์ภายนอกที่มีความสามารถจัดการบัสขอมา ซีพียูรับรู้และตอบรับ
- ☐ Global request ซีพียูหรือ DMA ภายในขอใช้ global bus

Interrupt Requests Z280 รองรับการอินเตอร์รัพท์ 2 ชนิดคือแบบ maskable และ nonmaskable ถ้าสัญญาณที่เข้าขา NMI เปลี่ยนจาก High เป็น Low และคงค่านี้ไว้แสดงว่าเกิดการขอการอินเตอร์รัพท์ ซีพียูจะเริ่มทำการอินเตอร์รัพท์ที่ไซเกิลสุดท้ายของคำสั่ง

ถ้าเป็น maskable interrupt รีจิสเตอร์ Master Status จะเป็นตัวบอกว่ามีการขออินเตอร์รัพท์มาจากที่ใดบ้าง และจะทำการจัดลำดับความสำคัญของการอินเตอร์รัพท์ โดยอ่านทางขาข้อมูล แต่การทำงานกับข้อมูลนี้จะขึ้นกับโหมดของการอินเตอร์รัพท์

Bus Request เกิดเมื่ออุปกรณ์ภายนอกที่มีความสามารถจัดการบัสร้องขอการใช้บัส (เช่น DMA Controller) โดยการเปลี่ยนสัญญาณที่ขา BUEREQ เป็น Low ซึ่งถ้าเกิดที่จุดเริ่มต้นสัญญาณนาฬิกาจะทำให้เกิดสัญญาณ BUSACK หลังจากที่การจัดการบัสเสร็จสิ้น ซีพียูจะเลิกควบคุมระบบบัส ขาทุกขายกเว้นขา BUSACK จะอยู่สภาวะ 3-state การขอใช้บัสของ DMA channels ภายในจะมีลำดับสำคัญกว่าการขอใช้บัสของอุปกรณ์ภายนอก

1.19 RESET

การรีเซตจะเริ่มต้นที่ clock สุดท้ายของการทำงานเมื่อสัญญาณที่ขา RESET มีค่าเป็น low ถ้าในขณะนั้นมีการจัดการบัสอยู่ ซีพียูจะรอให้ทำเสร็จก่อนจึงทำการรีเซต การรีเซตจะต้องมีช่วงเวลาอย่างต่ำ 128 clock cycle เมื่ออยู่ในสภาวะรีเซตขา AD ทั้งหมดอยู่ในภาวะ 3-state, ขาควบคุมจะเป็น high, ความถี่ที่ขา CLK จะเป็นความถี่ของโปรเซสเซอร์ที่ถูกหารด้วยสี่ เมื่อรีเซตแล้วค่ารีจีสเตอร์ต่างๆเป็นดังตารางที่ 1.10 และ 1.11 แต่การรีเซตไม่มีผลต่อ

- CPU Register file รวมถึง Stack Pointer
- Page Description registers
- Interrupt/Trap Vector Table Pointer Register

เมื่อเลิกการรีเซท ถ้ามีการขอใช้บัส Z280 จะอนุญาตให้ใช้บัสก่อนที่จะอ่านคำสั่งแรกที่ตำแหน่ง 0

Register	Value Loaded on Reset (Hexadecimal)	Comments
Program Counter	0000	
System Stack Pointer	0000	
I	00	
R	00	
Master Status	0000	System mode, Single-Step disabled, Breakpoint-on-Halt disabled All maskable interrupts disabled
Bus Timing and Control	30	No automatic wait states for I/O, upper 8M bytes of memory, or interrupt acknowledges
Bus Timing and Initialization	80	CLK output 2x processor clock period, no automatic wait states for lower 8 Mbytes of memory, bootstrap mode disabled.
I/O Page	00	I/O Page 0 in use
Cache Control	20	Cache enabled for instructions All valid bits cleared to 0 Burst mode disabled
Trap Control	00	EPA trap disabled, I/O not privileged
System Stack Limit	0000	System Stack Overflow Warning trap disabled
Local Address	00	All memory transactions are made to local bus
Interrupt Status	00xx	Interrupt mode 0, nonvectored interrupts, current state of interrupt requests (indicated by xx)
Interrupt/Trap Vector Table Pointer		Unaffected
CPU Registers AF, BC, DE, HL, IX, IY, AF', BC', DE', DE', HL		Unaffected
User Stack Pointer		Unaffected
MMU Master Control	0000	MMU disabled
MMU Page Descriptor Register, Page Descriptor Register Pointer		Unaffected

ตารางที่ 1.10 Effect of a Reset on Z280 CPU and รีจีสเตอร์

Register	Value Loaded on Reset (Hexadecimal)	Comments
Refresh	88	Refresh enabled, rate = 32
Counter/Timers:		
Configuration	00	Timer mode, single-cycle mode
Command/Status	00	Timer disabled
DMA Channels:		
Master Control	0000*	No DMA linking, EOP disabled, Software Ready disabled
DMA0 Transaction Descriptor	0100*	DMA0 disabled, continuous mode
DMA1/2/3 Transaction Descriptor	—	EN, IE, TC, and EPS fields cleared, other fields unaffected
DMA0 Destination Address	000000	
DMA0 Count	0100	
UART:		
Configuration	00*	5 bits/character, parity disabled, external clock, x1 clock rate, loop back disabled
Transmitter Control/Status	01	Transmitter disabled, transmit buffer empty
Receiver Control/Status	00*	Receiver disabled

*Unless bootstrap mode is selected.

ตารางที่ 1.11 Effect of a Reset on Z280 On-Chip Peripheral รีจีสเตอร์

บทที่ 2

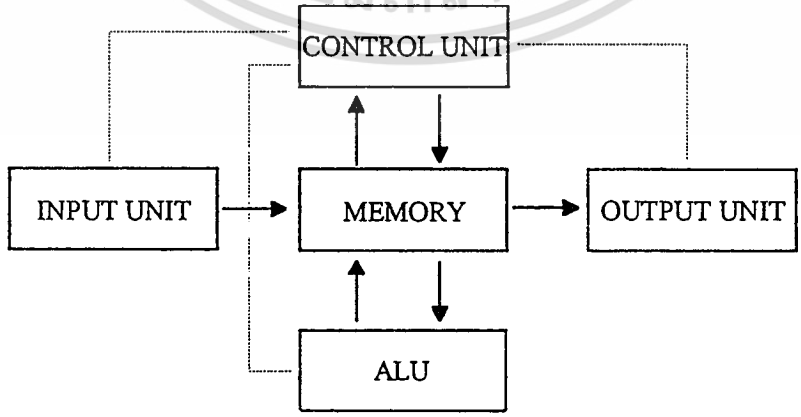
การทดลองใช้งาน Z280

เครื่องคอมพิวเตอร์เป็นสมองกลอิเลคทรอนิกส์ที่ทำงานอย่างหนึ่งได้รวดเร็วและดีกว่าคนมาก มันสามารถรับข้อมูลเป็นจำนวนมากและแยกแยะได้เร็วกว่าคน ด้วยความเร็วนี้เองงานชิ้นเดียวกันอาจจะต้องใช้คนทำถึงเป็นอาทิตย์หรือเป็นเดือน แต่คอมพิวเตอร์อาจใช้เวลาเพียงหน่วยวินาทีเท่านั้น ด้วยเหตุนี้เราจึงใช้ความสามารถของคอมพิวเตอร์มาประมวลข้อมูลจำนวนมากๆในเวลาสั้นๆ สำหรับงานที่ต้องการความรวดเร็ว การใช้คอมพิวเตอร์อาจจะเป็นทางออกทางเดียวก็ได้ และก็ด้วยความเร็วและความสามารถสูงของคอมพิวเตอร์นี้เราจึงนำมาใช้งานในด้านธุรกิจอุตสาหกรรมและห้องทดลอง เพื่อลดรายจ่ายของงาน

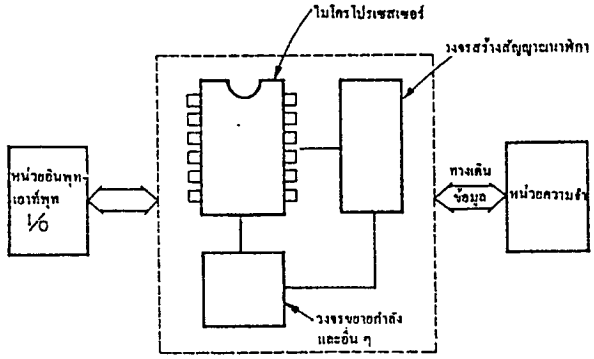
ไมโครคอมพิวเตอร์คือคอมพิวเตอร์ที่มีไมโครโปรเซสเซอร์เป็นตัวประมวลผลกลางหรือซีพียู(Central Processing Unit) และทำงานร่วมกับหน่วยอื่นอีก 2 หน่วยคือ หน่วยความจำ (memory unit) หน่วยรับส่งสัญญาณเข้าออก (input/output unit) เขียนเป็นบล็อกไดอะแกรมได้ดังรูป

รูปที่ 2.1 บล็อกไดอะแกรมของไมโครคอมพิวเตอร์

หน้าที่ของ CPU นอกจากจะคำนวณทางคณิตศาสตร์แล้วยังต้องส่งสัญญาณควบคุม หรือให้จังหวะแก่อุปกรณ์อื่นๆทำงานไปพร้อมๆกันด้วย ภายในตัวซีพียูจึงแบ่งเป็น 2 ส่วนใหญ่ๆ คือ วงจรควบคุม และวงจรคำนวณ เราสามารถแทนบล็อกไดอะแกรม CPU ด้วยรูปตัวจริงของไมโครโปรเซสเซอร์ได้ ดังรูปที่ 2.2 เฉพาะตัวไมโครโปรเซสเซอร์ใดๆก็ยังทำอะไรไม่ได้ ตัวมันเองทำงานเป็นลำดับขั้นในวงจรก็จริง แต่ก็ยังต้องอาศัยวงจรส่วนประกอบอื่นๆที่เพิ่มเติมเข้ามาเพื่อช่วยให้ CPU ทำงานได้ และติดต่อควบคุมอุปกรณ์อื่นๆได้ ส่วนที่เพิ่มเติมเข้ามาคือ วงจรสัญญาณนาฬิกา วงจรถอดรหัส วงจรขยายกำลัง ตามรูปที่ 2.3



รูปที่ 2.2 บล็อกไดอะแกรมซีพียู



รูปที่ 2.3 บล็อกไดอะแกรมการเชื่อมต่อไมโครโปรเซสเซอร์

วงจรส่วนต่างๆในระบบ ไม่ได้มีเพียงไมโครโปรเซสเซอร์เท่านั้น วงจรประกอบอื่นๆยังมีอีกมาก แต่ละวงจรก็มีหน้าที่สำคัญต่อระบบเช่นเดียวกันจึงจะทำให้ไมโครโปรเซสเซอร์ติดต่อควบคุมหน่วยต่างๆได้ วงจรที่สำคัญที่ช่วยต่อเชื่อม ซีพียูกับอุปกรณ์อื่นๆ เรียกว่า วงจรอินเตอร์เฟส (Interface circuits) วงจรอินเตอร์เฟสมีตั้งแต่ วงจรถอดรหัส วงจรขับกำลัง วงจรควบคุมการส่งผ่านสัญญาณ ความซับซ้อนของวงจรอินเตอร์เฟสจะมีมากเพียงใดขึ้นอยู่กับชนิดของอุปกรณ์ที่ซีพียูต้องติดต่อด้วย ระบบไมโครคอมพิวเตอร์ที่มีในท้องตลาดมีให้เลือกหลายระดับตามการใช้งานโดยแตกต่างกันไปตาม รูปร่าง ขนาด และส่วนประกอบ ไมโครคอมพิวเตอร์แบบแผ่นวงจรเดี่ยว เป็นคอมพิวเตอร์ระดับหนึ่ง ไมโครคอมพิวเตอร์แบบนี้มีข้อจำกัดในการใช้งานอยู่บ้าง เพราะอุปกรณ์ I/O และหน่วยความจำขนาดใหญ่ถูกตัดออกไป มีเหลือแต่ส่วนที่จำเป็นสำหรับใช้งานเท่านั้นได้แก่ CPU วงจรอินเตอร์เฟสอยู่จำนวนหนึ่ง หน่วยความจำขนาดพอประมาณไม่ใหญ่มาก และมีส่วนแสดงข้อมูลเป็นตัวเลขแบบง่ายๆ พร้อมกับคีย์บอร์ดขนาดหนึ่ง

ไมโครคอมพิวเตอร์แบบแผ่นวงจรเดี่ยว เหมาะสำหรับการศึกษาและการใช้อุปกรณ์สนับสนุนการออกแบบระบบที่ใหญ่กว่าหรือออกแบบวงจรอิเลคทรอนิกส์ใช้งานอื่นๆ โดยอาศัยไมโครโปรเซสเซอร์เป็นตัวหลักในอุปกรณ์นั้นๆ เนื่องจากอุปกรณ์ I/O มีจำกัดนี้เอง ภาษาที่ใช้ก็ถูกจำกัดให้อยู่ในระดับต่ำ เช่น ภาษาเครื่อง (Assembly)

ในการเลือกไมโครโปรเซสเซอร์ที่เหมาะสำหรับงานต่างๆ จะต้องพิจารณาข้อมูลให้ละเอียดจึงจะได้ระบบที่ดีเหมาะกับงานมากที่สุด ข้อนี้ขึ้นอยู่กับข้อจำกัดในการเลือกใช้เอง นอกจากเทคโนโลยีต่างๆแล้ว โครงสร้างภายในไมโครโปรเซสเซอร์แต่ละเบอร์ก็ต่างกันออกไป โครงสร้างภายในหมายถึง ขนาดของข้อมูล จำนวนคำสั่ง การตอบรับสัญญาณจากภายนอก ซึ่งต้องศึกษาจากรายละเอียดของผู้ผลิต ในส่วนของโครงงานนี้เลือกใช้ไมโครโปรเซสเซอร์ Z280 ซึ่งเป็นไมโครโปรเซสเซอร์ขนาด 16 bit ตัวใหม่ของ ZILOG มาประกอบกันเป็นไมโครคอมพิวเตอร์แบบแผ่นวงจรเดี่ยว ซึ่งมีวงจรประกอบต่างๆคือ

- วงจรแหล่งจ่ายไฟตรง
- วงจรถอดรหัสสัญญาณควบคุม
- วงจรหน่วยความจำ
- วงจรรับส่งสัญญาณเข้าออก
- โปรแกรมมอนิเตอร์

2.1 คุณสมบัติทั่วไป

Z280 SINGLE BOARD เป็นไมโครคอมพิวเตอร์แผ่นวงจรเดี่ยวที่ใช้ไมโครโปรเซสเซอร์เบอร์ Z280 ซึ่งเป็นไมโครโปรเซสเซอร์ขนาด 16 บิทตัวใหม่ของบริษัท ZILOG ทำงานที่ความถี่ 10 Mhz มีหน่วยความจำเป็น EPROM ขนาด 64 Kbyte, RAM ขนาด 64 Kbyte และยังมีหน่วยความจำ 256 byte (on-chip memory) ภายในตัวไมโครโปรเซสเซอร์ มีวงจรแหล่งจ่ายไฟกระแสตรงอยู่บนบอร์ดสามารถจ่ายกระแสให้แก่ระบบได้ถึง 3 แอมป์

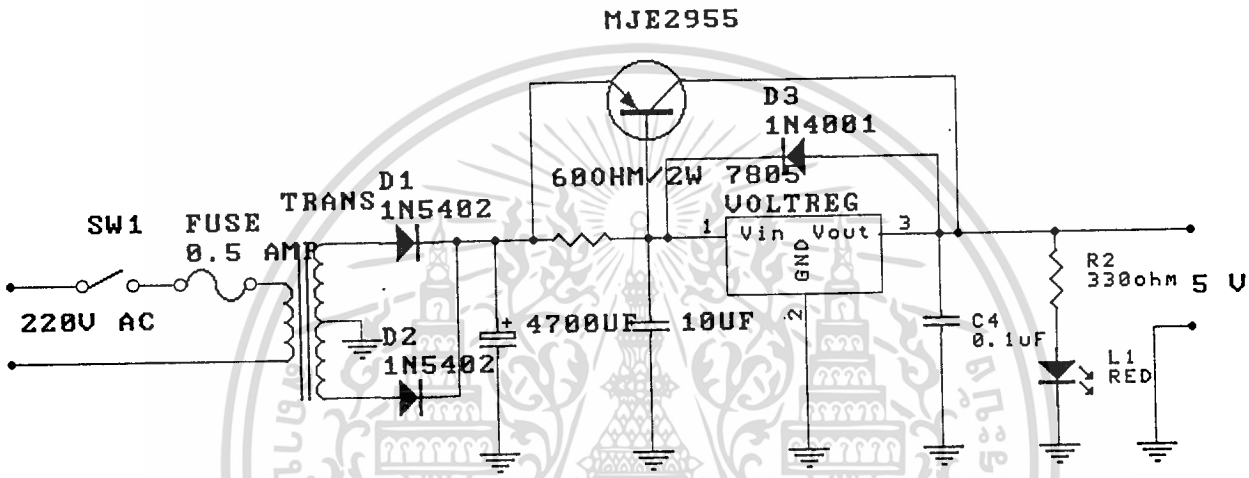
สำหรับการติดต่อกับผู้ใช้งาน Z280 SINGLE BOARD มีสวิตช์คีย์บอร์ดให้จำนวน 24 คีย์ มีการแสดงผลเป็น 7-SEGMENT รวมกับ LED ตัวเดี่ยวๆที่ทำหน้าที่เป็นแฟลกอีกรวมเป็น 9 หลัก ซึ่งจะให้ความสะดวกในด้านการป้อนและตรวจสอบข้อมูลในหน่วยความจำ หรือ ในรีจีสเตอร์ การใช้ฮาร์ดแวร์อินเตอร์รัพท์ การรันโปรแกรมทีละคำสั่งแบบซิงเกิลสเตป ตลอดจนการรันโปรแกรมตามแอดเดรสที่ต้องการ และมีวงจร digital to analog converter เพื่อแปลงข้อมูลจากดิจิตอล เป็น อนาลอก นำข้อมูลเข้าสู่ Amplifier เพื่อออกสู่ลำโพง ในขณะเดียวกันภายในไมโครโปรเซสเซอร์ยังมีส่วน DMA, Timer/Counter, Interrupt ,Trap ที่บอร์ดนี้สามารถรองรับการทำงานได้

ไมโครคอมพิวเตอร์ทุกเครื่องจะมีลักษณะการทำงานที่คล้ายกัน ซึ่งแบ่งออกเป็น 2 ส่วน คือ ฮาร์ดแวร์และซอฟท์แวร์ ส่วนที่เป็นฮาร์ดแวร์ได้แก่อุปกรณ์ทั้งหมดที่สร้างขึ้นเป็นวงจรคอมพิวเตอร์ ประกอบด้วย ซีพียู หน่วยความจำ พอร์ทอินพุท/เอาท์พุท สวิตช์คีย์บอร์ด ตัวแสดงผล ตลอดจนอุปกรณ์อื่นๆ สำหรับซอฟท์แวร์เป็นส่วนที่เกิดมาจากการนำคำสั่ง(instuctor) ของซีพียูมาเขียนเรียงกันเป็นโปรแกรมเพื่อให้เกิดการทำงานอย่างใดอย่างหนึ่ง ซอฟท์แวร์ในเครื่องไมโครคอมพิวเตอร์แผ่นพิมพ์เดี่ยวนี้จะมี 2 ส่วน ส่วนหนึ่งถูกเก็บอยู่ใน EPROM ส่วนนี้จะอยู่ถาวร ไม่ว่าจะมีไฟเลี้ยงให้กับวงจรหรือไม่ก็ตาม ส่วนนี้จะเก็บโปรแกรม หรือ Subroutine ที่จำเป็น ซึ่งอาจจะมีส่วนโปรแกรมมอนิเตอร์ด้วย โดยโปรแกรมมอนิเตอร์เป็นโปรแกรมจัดการระบบต่างๆทางด้านฮาร์ดแวร์ของเครื่อง ซึ่งจะมีมากน้อยแล้วแต่ความต้องการของผู้ออกแบบสร้าง ซอฟท์แวร์อีกส่วนหนึ่งจะอยู่ใน RAM ซึ่งเป็นพื้นที่สำหรับผู้ใช้งานป้อนโปรแกรมลงไปทางคีย์บอร์ด หรือเป็นค่าของตัวแปรในการทำงาน

ดังนั้น ในการที่จะนำไมโครคอมพิวเตอร์แผ่นพิมพ์เดี่ยวไปใช้ในงานระบบควบคุมหรือทำงานประมวลผลใดๆ จำเป็นจะต้องอาศัยการทำงานร่วมกันอย่างถูกต้องทั้งฮาร์ดแวร์และซอฟท์แวร์

2.2 วงจรแหล่งจ่ายไฟกระแสตรง

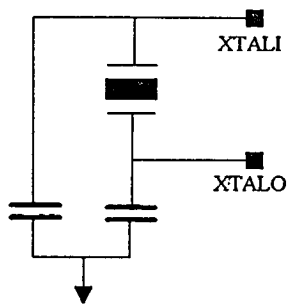
เนื่องจากในปัจจุบันมีไอซีเรคกูเลเตอร์ ออกมาเป็นจำนวนมาก ทำให้การออกแบบแหล่งจ่ายไฟกระแสตรงง่ายขึ้นไม่ยุ่งยากซับซ้อน ในส่วนของโครงงานนี้ใช้ไอซีเรคกูเลเตอร์เบอร์ 7805 ซึ่งเป็นเรคกูเลเตอร์ที่ให้ output voltage คงที่บวก.5 โวลท์ แต่เนื่องจาก 7805 จ่ายกระแสให้แก่โหลดได้ไม่เกิน 1 แอมป์ จึงจำเป็นต้องใช้ ทรานซิสเตอร์เบอร์ MJE2955 ทำหน้าที่เป็นตัวจ่ายกระแสให้เพิ่มขึ้น ใช้หม้อแปลงขนาด 9-0-9 โวลท์ ขนาด 3 แอมป์ ใช้ไดโอดเบอร์ 1N5402 สองตัวต่อกันเป็น วงจร full wave rectifier คาปาซิเตอร์ 4700 F/16 V เป็นตัวกรองกระแส และด้าน output ของไอซี 7805 มี LED ที่ต่ออนุกรมกับตัวต้านทาน เพื่อแสดงการทำงานของวงจรส่วนนี้ ตามรูปที่ 2.4



รูปที่ 2.4 วงจรแหล่งจ่ายไฟกระแสตรง

2.3 วงจรออสซิเลเตอร์

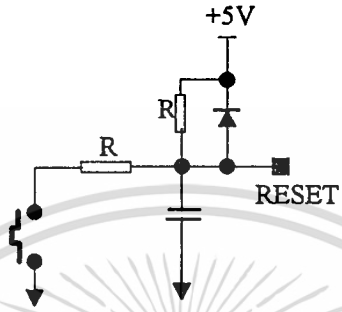
ไมโครโปรเซสเซอร์ Z280 มีวงจร oscillator อยู่ภายในโดยสามารถต่อ X-TAL ได้โดยตรงหรือใช้สัญญาณ clock จากภายนอกก็ได้ แต่ทั้งสองกรณีความถี่ของระบบจะเป็นครึ่งหนึ่งของสัญญาณ clock ที่ป้อนเข้ามา ในโครงงานนี้ใช้ X-TAL ขนาด 20 Mhz ต่อเข้าโดยตรงกับตัวไมโครโปรเซสเซอร์โดยมีคาปาซิเตอร์ค่าน้อยๆ ต่ออยู่เพื่อทำหน้าที่เป็น load capacitive จะมี clock ของระบบเท่ากับ 10 Mhz ดังรูปที่ 2.5 ถ้าใช้สัญญาณ clock ภายนอกสัญญาณจะถูกต่อเข้ากับขา XTALI ในขณะที่ขา XTALO จะปล่อยลอยไว้



รูปที่ 2.5 วงจร oscillator

2.4 สวิตช์รีเซตและสวิตช์เบรก

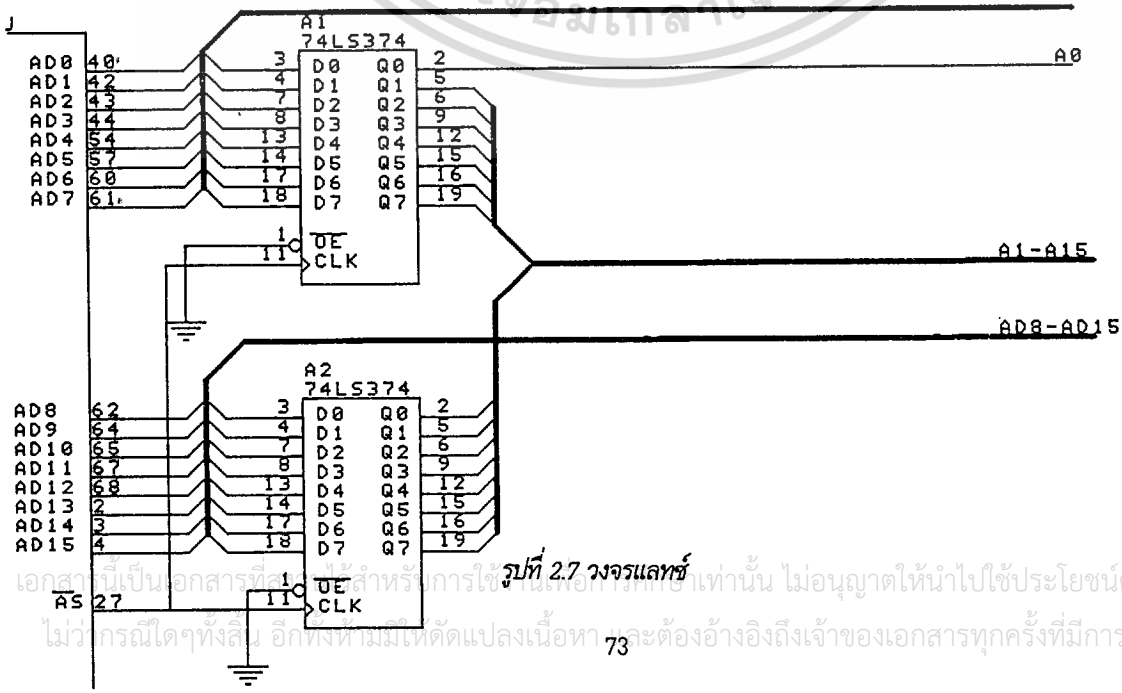
ขา RESET จะ active เมื่อ logic low ขานี้ปกติจะมีตัวต้านทานต่อ pull-up ไว้ และต่อโดยตรงกับสวิตช์เพื่อใช้เป็นสวิตช์รีเซต โดยไดโอดและคาปาซิเตอร์ที่ต่อคร่อมขานี้ไว้จะทำหน้าที่กันสัญญาณรบกวนจากภายนอกที่อาจจะเกิดขึ้นได้ ดังรูปที่ 2.6 ส่วนขา $\overline{INTA}$ ซึ่งเป็นขา maskable interrupt ขาหนึ่งของตัวไมโครโปรเซสเซอร์ ปกติจะ active ที่ logic low เช่นเดียวกัน โดยต่อตัวต้านทาน pull-up ไว้และต่อโดยตรงกับสวิตช์เพื่อทำหน้าที่เป็นสวิตช์เบรก



รูปที่ 2.6 สวิตช์ รีเซต

2.5 address bus และ data bus

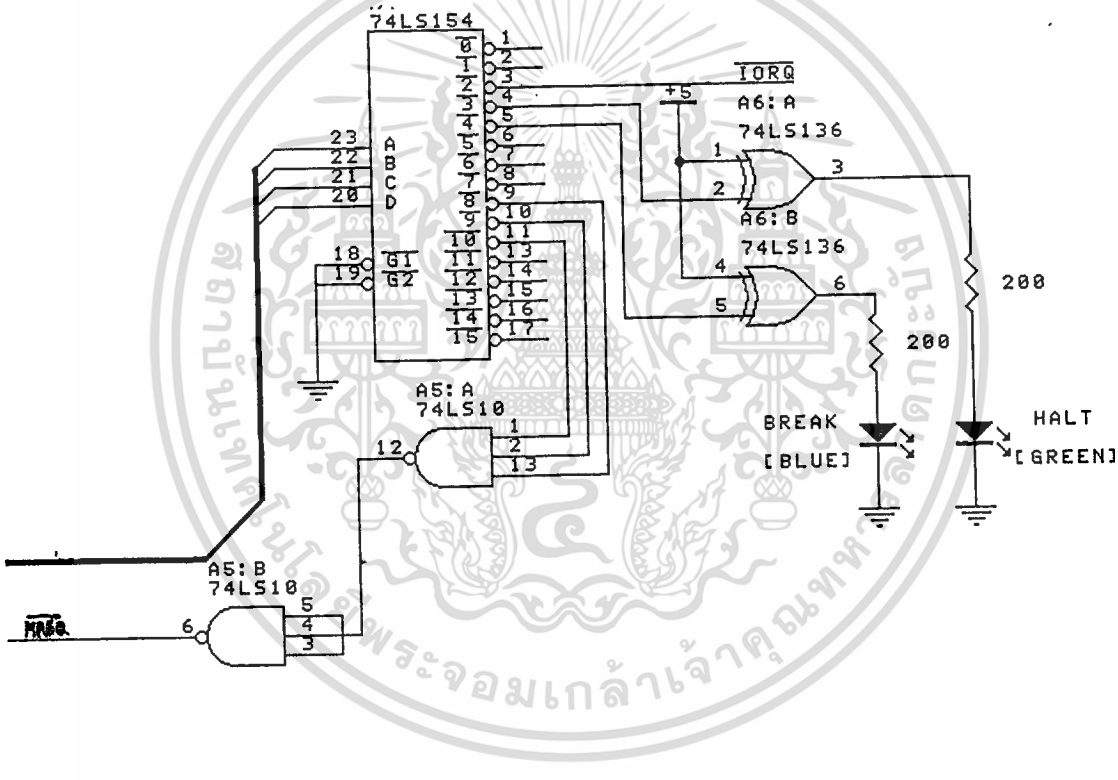
เนื่องจากไมโครโปรเซสเซอร์ Z280 มีการอินเตอร์เฟสกับภายนอก 2 แบบคือ Z80 bus ซึ่งเป็นบัสขนาด 8 บิทที่เหมือนกับ Z80 และ Z bus ซึ่งเป็นบัสขนาด 16 บิท โดยการจัดขาและสัญญาณของทั้งสองแบบจะไม่เหมือนกัน ในโครงงานนี้เลือกการอินเตอร์เฟสแบบ Z bus โดยป้อน logic high ให้กับขา OPT ของตัวไมโครโปรเซสเซอร์ เนื่องจาก Z bus มีขาข้อมูลถึง 16 บิท ดังนั้นเพื่อให้ประหยัดขาไอซี จึงใช้ขา address และ data ร่วมกัน คือขา AD_0 - AD_{15} จึงต้องใช้วิธี multiplex ระหว่างขา address และ data โดยใช้ 74LS374 ซึ่งเป็น D-flipflop เป็นตัวแลทช์ข้อมูล และใช้ขา $\overline{AS}$ (address strobe) จากไมโครโปรเซสเซอร์ต่อตรงกับขา CLK ของ 74LS374 เพื่อแยกสัญญาณระหว่าง address bus และ data bus ดังรูปที่ 2.7 และ Z280 ยังมีขา address A16-A23 อีก 8 เส้น รวมกันแล้วมี address bus อยู่ 24 เส้น ทำให้สามารถอ้าง address ได้ถึง 16 Mbyte และมี data bus อยู่ 16 เส้น



รูปที่ 2.7 วงจรแลทช์

2.6 วงจรควบคุม

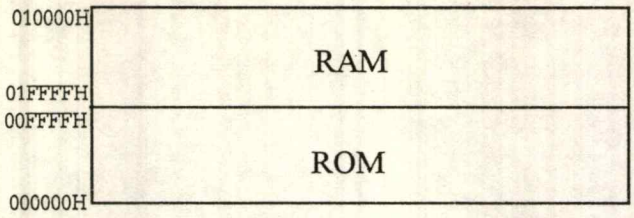
เนื่องจาก Z Bus ของไมโครโปรเซสเซอร์ Z280 ไม่มีขาควบคุมต่างๆโดยตรงเป็น control bus แต่มีขา status ST0-ST3 จึงจำเป็นต้องนำขา ST ทั้ง 4 ขามาถอดรหัสเพื่อสร้างสัญญาณควบคุมจากบทที่ 1 ในตารางที่ 1.9 เราต้องการขา status ST3-ST0 เป็น 0010 เพื่อสร้างสัญญาณ $\overline{IORQ}$ 0011 สร้างสัญญาณ HALT 0100 สร้างสัญญาณตอบรับอินเตอร์รัพท์ A1000, 1001 และ 1010 ใช้สร้างสัญญาณ $\overline{MREQ}$ ในที่นี้ใช้ 74LS154 ซึ่งเป็น 4 to 16 decoder ถอดรหัสสัญญาณจากขา status โดยขา 2 ของ 74LS154 ใช้เป็นสัญญาณ $\overline{IORQ}$ ในการติดต่อกับอุปกรณ์อินพุท/เอาท์พุท ขาที่ 4 และ 5 ต่อกับขาอินพุทข้างหนึ่งของ XOR ซึ่งต่อเป็น NOT GATE โดยต่ออนุกรมกับตัวต้านทาน และ LED เพื่อบอกสภาวะกระทำคำสั่ง HALT และสภาวะ BREAK และขา 8,9,10 ของ 74LS154 ต่อเข้ากับขาอินพุทของ NAND GATE 3 อินพุท 2 ตัว เพื่อทำเป็น AND GATE ใช้สร้างสัญญาณ $\overline{MREQ}$ ในการติดต่อกับหน่วยความจำ ดังรูปที่ 2.8



รูปที่ 2.8 การสร้างสัญญาณควบคุม

2.7 การจัดหน่วยความจำ

หน่วยความจำเป็นส่วนหลักที่สำคัญมากในระบบคอมพิวเตอร์ การจัดตำแหน่งแบ่งสรรพื้นที่ใช้งานเป็นสิ่งที่สำคัญเพื่อให้ซีพียูมองเห็นและรู้ว่าจะทำงานอย่างไร หน่วยความจำใน Z280 SINGLE BOARD สามารถเขียนเป็นแผนผังได้ดังรูปที่ 2.9 จะเห็นว่าพื้นที่ส่วนของ EPROM อยู่ที่แอดเดรส 000000H-00FFFF ส่วน RAM สำหรับผู้ใช้งานอยู่ที่แอดเดรส010000H-01FFFFH



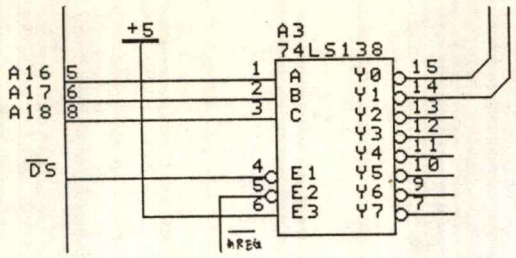
รูปที่ 2.9 แสดงการจัดแบ่งพื้นที่ของหน่วยความจำ

Z280 มีขาแอดเดรส 24 เส้น จึงสามารถอ้างแอดเดรสได้ 2^{24} = 16777216 ตำแหน่ง หรือ 16 เมกกะแอดเดรส โดยถูกออกแบบให้แต่ละแอดเดรสสามารถติดต่อข้อมูลในหน่วยความจำได้ 8 บิทหรือ 1 ไบท์จึงทำให้ติดต่อข้อมูลได้ทั้งหมด 16 เมกกะไบต์ โดยข้อมูลที่ตัวซีพียูอ่านได้แต่ละครั้ง จะเป็นได้ทั้งแบบครั้งละ byte(8 bit) หรือครั้งละ word(16 bit) โดยแต่ละส่วนก็จะมีสัญญาณมาควบคุมให้ทำงานแยกจากกันโดยใช้สัญญาณ A0 และ B/$\overline{W}$ ซึ่งสัญญาณใดจะทำงานขึ้นอยู่กับขนาดของข้อมูลที่ต้องติดต่อ ดังสรุปได้ตามตาราง

B/$\overline{W}$	A0	Y0	$\overline{CE}$ A	$\overline{CE}$ B	$\overline{CE}$ C	$\overline{CE}$ D	การทำงาน
0	0	0	0	0	1	1	ติดต่อกับ ROM เป็น WORD
0	0	1	1	1	0	0	ติดต่อกับ RAM เป็น WORD
0	1	0	1	1	1	1	ไม่ได้ใช้
0	1	1	1	1	1	1	ไม่ได้ใช้
1	0	0	1	0	1	1	ติดต่อข้อมูลเป็น BYTE จาก ROM ทาง D8-D15
1	0	1	1	1	1	0	ติดต่อข้อมูลเป็น BYTE จาก RAM ทาง D8-D15
1	1	0	0	1	1	1	ติดต่อข้อมูลเป็น BYTE จาก ROM ทาง D0-D7
1	1	1	1	1	0	1	ติดต่อข้อมูลเป็น BYTE จาก RAM ทาง D0-D7&

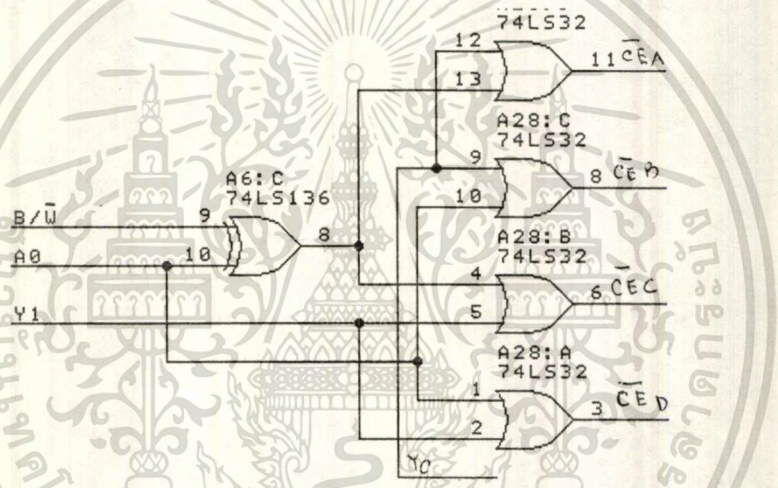
ตารางที่ 2.1 แสดงการทำงานของ A0 และ B/$\overline{W}$ เมื่อติดต่อกับหน่วยความจำ

ในโครงงานนี้ใช้ EPROM เบอร์ 27256 และ RAM เบอร์ 62256 ซึ่งเป็นหน่วยความจำขนาดตัวละ 32 Kbyte อย่างละ 2 ตัว จึงใช้ขาแอดเดรส A_{16}-A_{18} ของไมโครโปรเซสเซอร์ มาถอดรหัสโดยใช้ 74LS138 decoder เพื่อเลือกช่วงหน่วยความจำ ช่วงละ 64 Kbyte โดยใช้สัญญาณ Y0 จาก 74LS138 เพื่อเลือก ROM และสัญญาณ Y1 เพื่อเลือก RAM ส่วนสัญญาณอื่นๆไว้ใช้ขยายระบบในอนาคต ซึ่ง 74LS138 มีขา $\overline{G2A}$,$\overline{G2B}$ นำมาต่อเข้ากับสัญญาณ $\overline{DS}$ จากไมโครโปรเซสเซอร์ และ $\overline{MERQ}$ ในการถอดรหัสจากขา status ดังรูปที่ 2.10



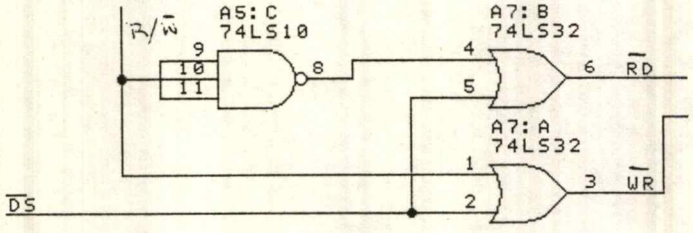
รูปที่ 2.10 การถอดรหัสเพื่อเลือกช่วงหน่วยความจำ

การเลือกหน่วยความจำว่าเป็นแอดเดรสคู่หรือแอดเดรสคี่จะใช้ A0 โดยถ้า A0 เป็น 0 ซีพียูจะติดต่อกับหน่วยความจำทาง D8-D15 และถ้า A0 เป็น 1 ซีพียูจะติดต่อกับหน่วยความจำทาง D0-D7 ทั้งนี้สัญญาณ B/$\overline{W}$ จะต้องเป็น 1 ด้วยเพื่อบ่งบอกถึงการติดต่อกับข้อมูลเป็น byte ถ้าสัญญาณ B/$\overline{W}$ เป็น 0 ซีพียูจะติดต่อกับข้อมูลเป็น word ซึ่งจะใช้ขา data bus D0-D15 และจะต้องนำมาประกอบกับสัญญาณ Y0 และ Y1 จาก 74LS138



รูปที่ 2.11 วงจรเข้ารหัสหน่วยความจำทั้งหมด

ในหน่วยความจำส่วนที่เป็น RAM เนื่องจากตัวไมโครโปรเซสเซอร์มีขา R/$\overline{W}$ เพียงขาเดียว และในวงจรต้องการสัญญาณทั้ง $\overline{RD}$ และ $\overline{WR}$ จึงนำสัญญาณ $\overline{DS}$ ซึ่งเป็นสัญญาณที่มีช่วงแคบที่สุดใน Timing Diagram มาประกอบกับสัญญาณ R/$\overline{W}$ เพื่อสร้างสัญญาณ $\overline{RD}$ และ $\overline{WR}$ เพื่อต่อกับขา $\overline{WR}$ และ $\overline{OE}$ ของ chip 62256 ดังรูป



รูปที่ 2.12 การสร้างสัญญาณ $\overline{RD}$ และ $\overline{WR}$

2.8 การจัดอินพุท/เอาท์พุทพอร์ท

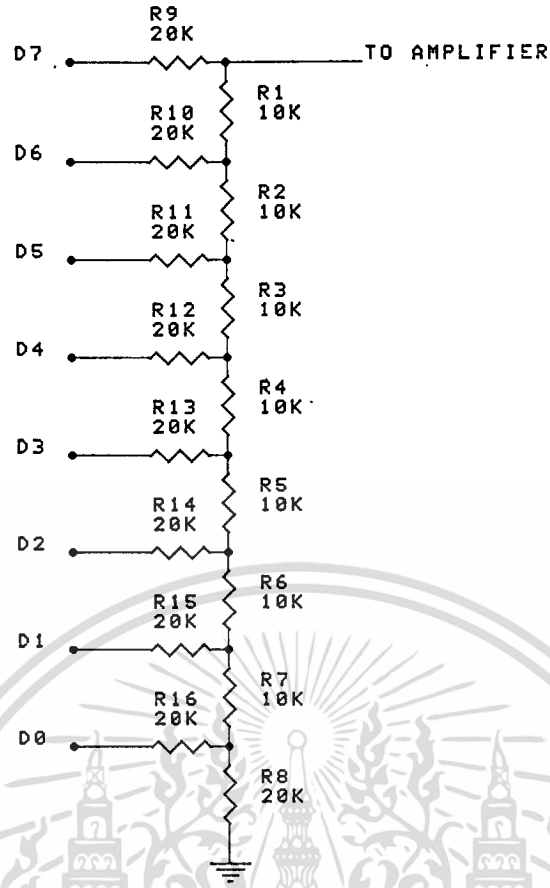
เนื่องจากการจัดอินพุท/เอาท์พุทพอร์ทของ Z280 SINGLE BOARD ใช้การจัดแบบ ไอโอแมปไอโอ (I/O mapped I/O) คือ พื้นที่การจัดอินพุท/เอาท์พุท จะแยกออกจากพื้นที่หน่วยความจำอย่างสิ้นเชิง ซึ่งข้อดีของระบบนี้ทำให้พื้นที่ส่วนที่เป็นหน่วยความจำสามารถใช้งานได้เต็มที่ โดยไม่เกี่ยวข้องกับส่วนที่เป็นอินพุท/เอาท์พุทพอร์ท เหมาะสำหรับระบบงานที่ต้องการขยายให้มีขนาดใหญ่

ไมโครโปรเซสเซอร์ Z280 สามารถอ้างพอร์ทได้ 24 บิท (A0-A23) การจัดอินพุท/เอาท์พุทพอร์ทแสดงในตารางที่ 2. จะเห็นว่ามีการใช้พอร์ทอยู่ 4 พอร์ทในวงจรนี้คือ พอร์ทหมายเลข 00,01,02 และ 03 โดยใช้สายแอดเดรส A0-A2 และสัญญาณ $\overline{DS}$ กับ $\overline{IORQ}$ ที่ถอดรหัสจากขา status มาใช้ในการถอดรหัสเพื่อกำหนดหมายเลขพอร์ทโดยใช้ไอซี 74LS138 มาใช้ในการถอดรหัสสัญญาณเลือกพอร์ท

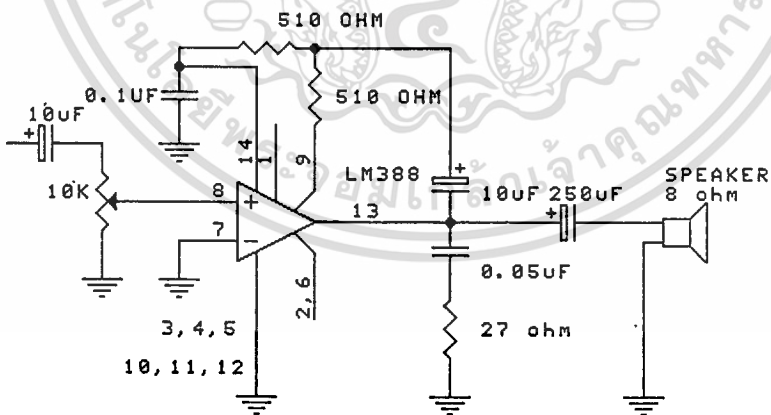
หมายเลขพอร์ท	ชื่อพอร์ท
00	DAC และ Amplifier
01	Display
02	Keyboard(Input)
03	เลือกหลัก (Output)
04-07	อินพุท/เอาท์พุทพอร์ทที่ขยายเพิ่มเติมได้

ตารางที่ 2.2 แสดงรายละเอียดการจัดอินพุท/เอาท์พุทพอร์ท

เนื่องจากไมโครโปรเซสเซอร์ทำงานได้เร็วกว่าอินพุท/เอาท์พุทพอร์ทมาก ดังนั้นในส่วนของเอาท์พุทพอร์ทจะใช้ 74LS374 ซึ่งเป็น D-flip flop มาทำหน้าที่แลทช์ก่อนที่จะนำข้อมูลออก และอินพุทพอร์ทจะใช้ 74LS244 ซึ่งเป็น tri-state buffer มาเป็นตัวรับข้อมูล พอร์ทหมายเลข 00 ประกอบไปด้วยวงจร DAC และวงจร Amplifier วงจรดีเอซี (DAC=Digital to Analog Converter) คือวงจรใช้สำหรับเชื่อมต่อระหว่างวงจรดิจิตอลกับวงจรอนาลอก โดยจะแปลงสัญญาณดิจิตอลที่ได้จากไมโครโปรเซสเซอร์เป็นค่าศักดาไฟฟ้า ในวงจรนี้ใช้ DAC แบบ R-2R leader ซึ่งเป็นการต่อ DAC ที่หาอุปกรณ์ง่ายที่สุดและนิยมที่สุด ศักดาไฟฟ้าที่ได้จากวงจร DAC จะถูกต่อเข้ากับออปแอมป์เพื่อขยายต่อไป โดยในที่นี้ใช้ LM388 หลังจากนั้นจึงส่งข้อมูลอนาลอกออกสู่ลำโพงต่อไป

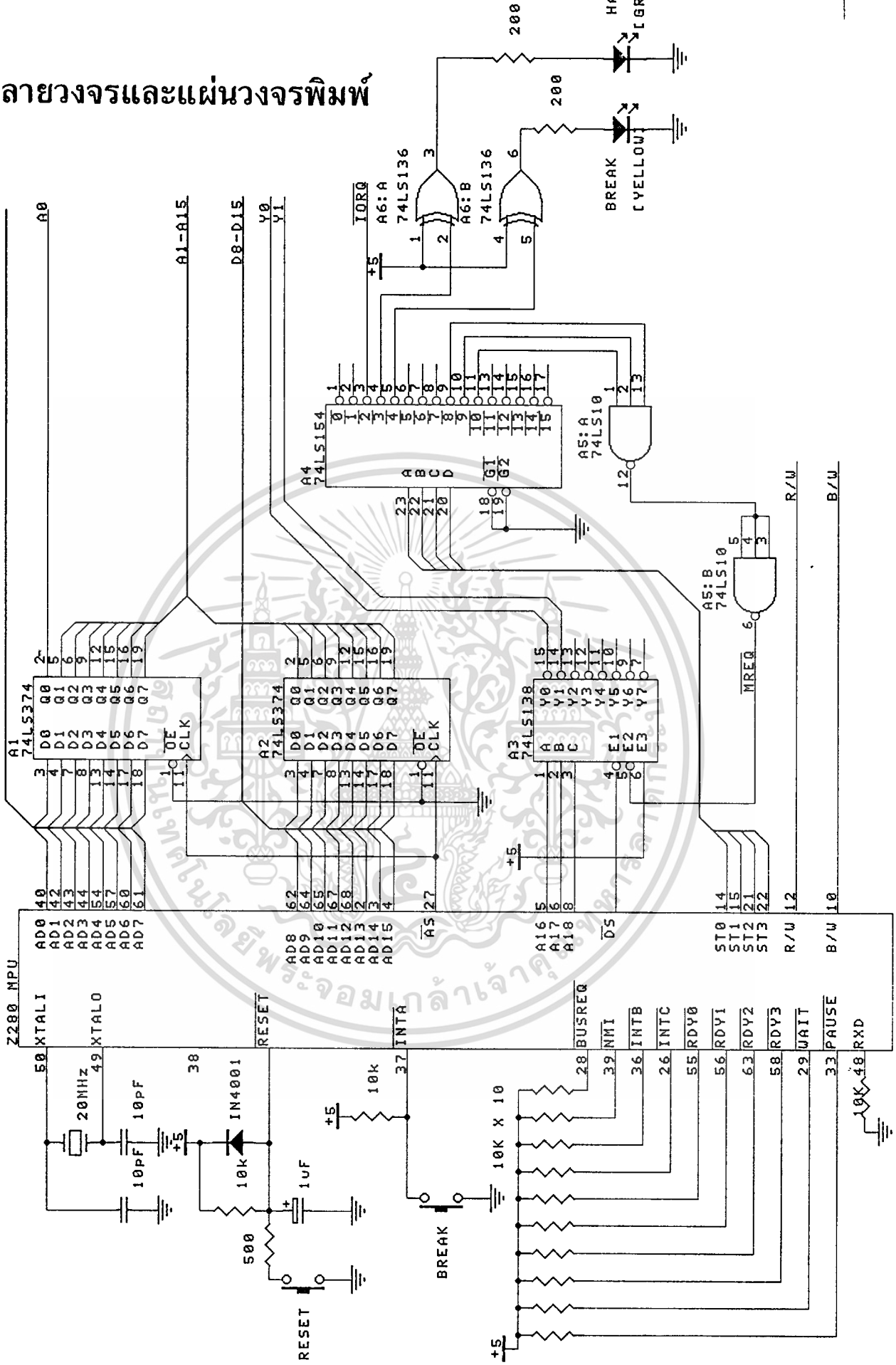


รูปที่ 2.13 วงจร DAC

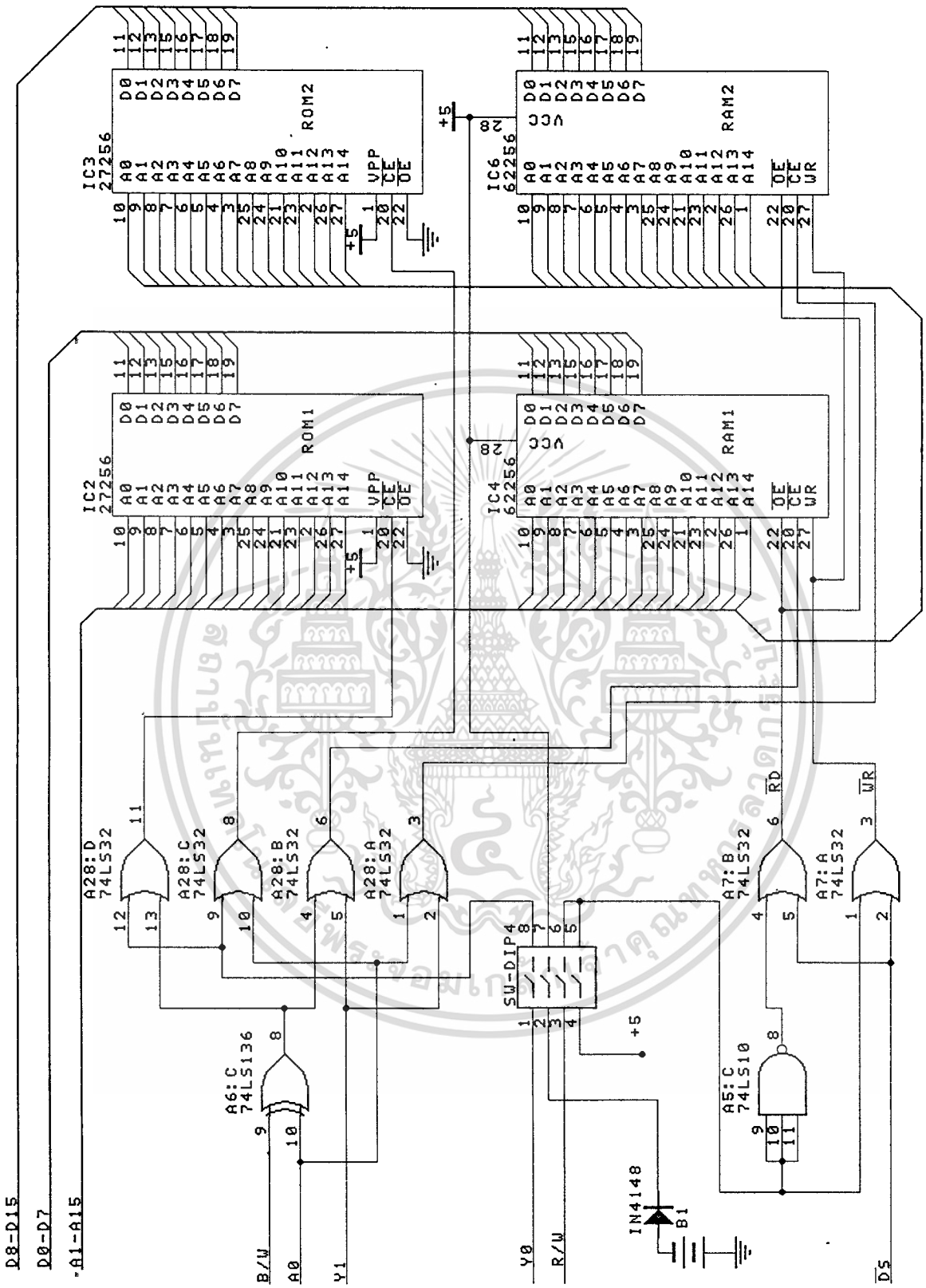


รูปที่ 2.14 วงจร Amplifier

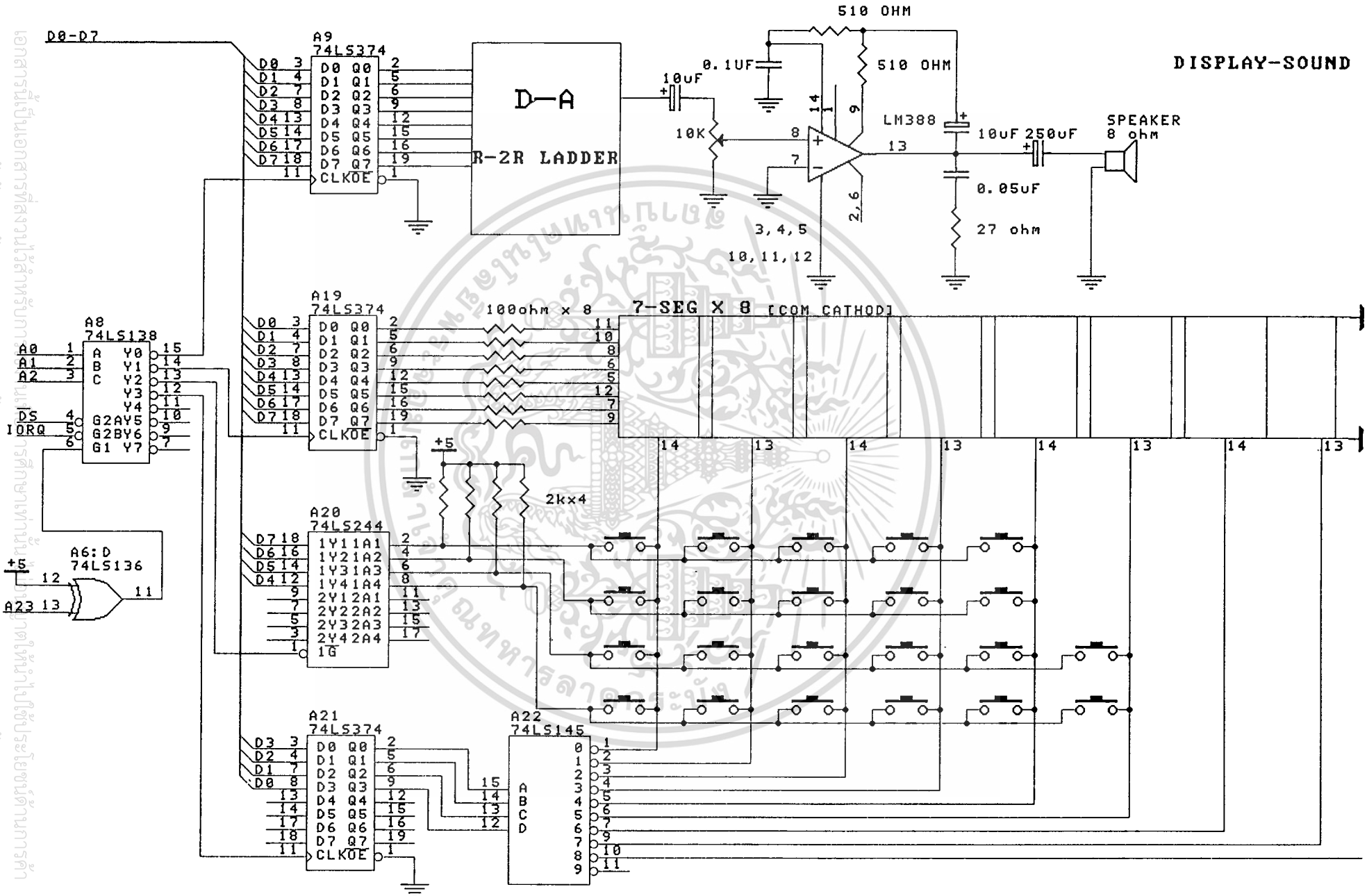
2.9 ลายวงจรและแผ่นวงจรพิมพ์



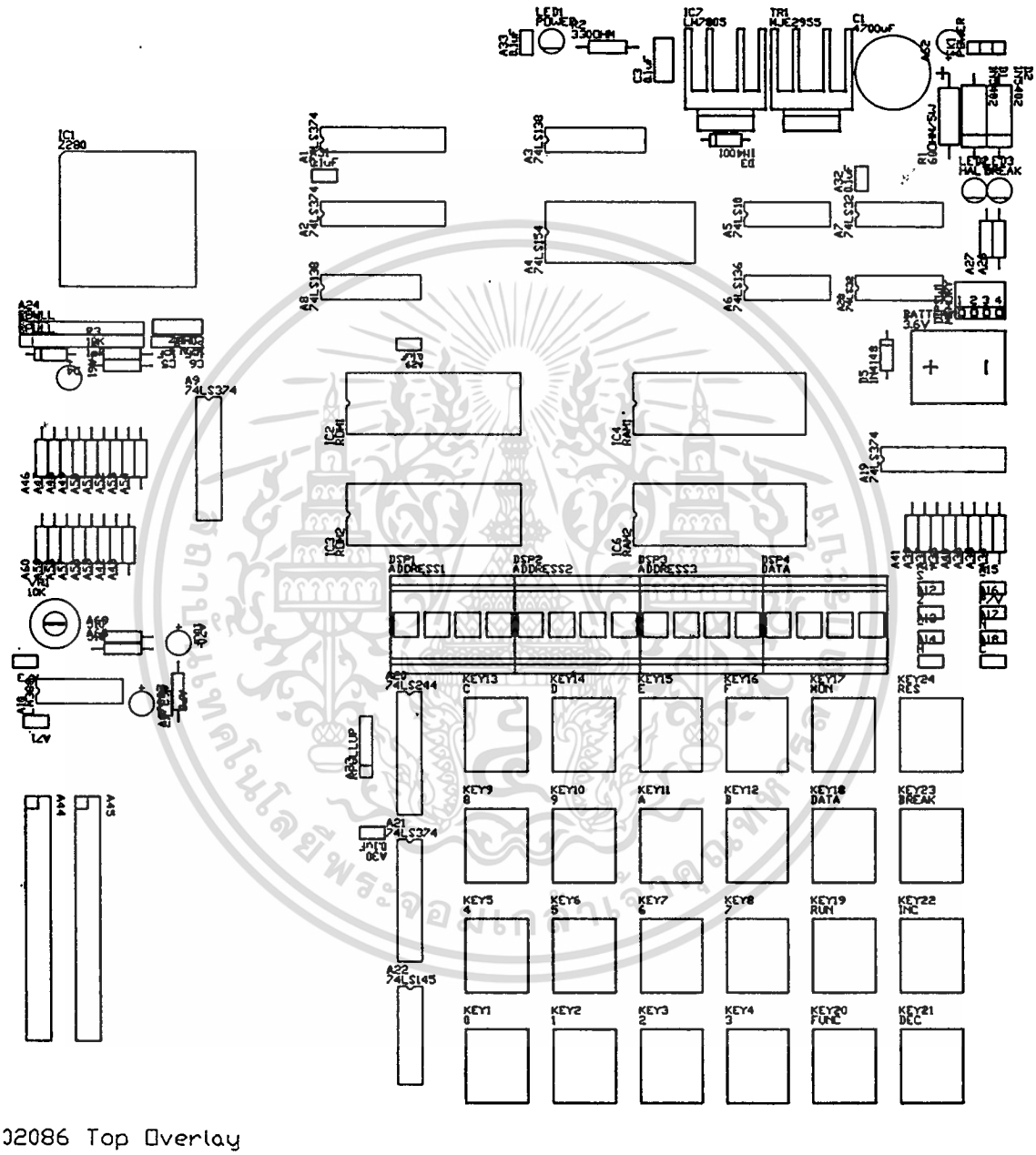
รูปที่ 2.15 ส่วนควบคุมซิงเกิลบอร์ด โดย Z280



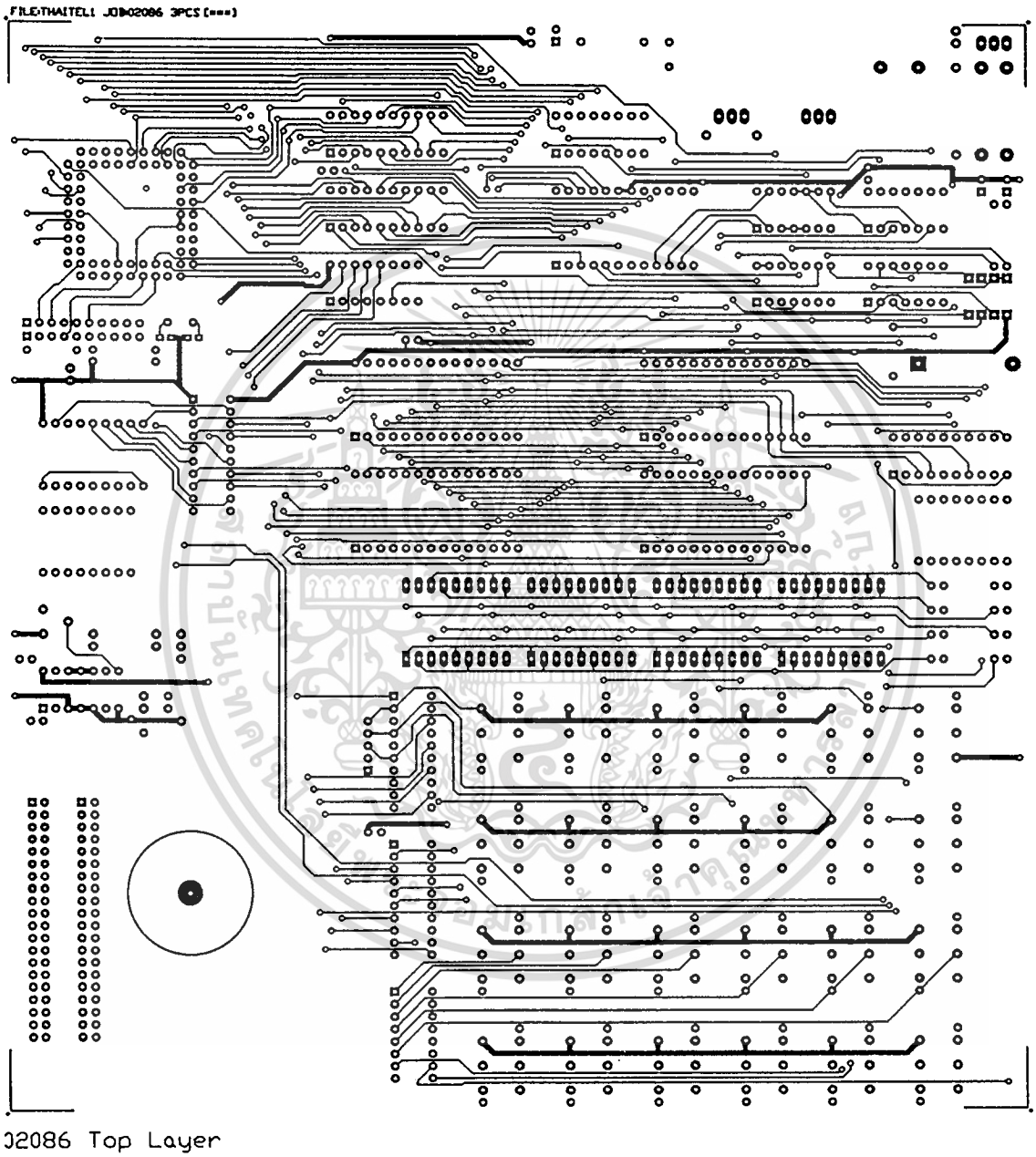
รูปที่ 2.16 การจัดหน่วยความจำของชิงเกิลบอร์ด



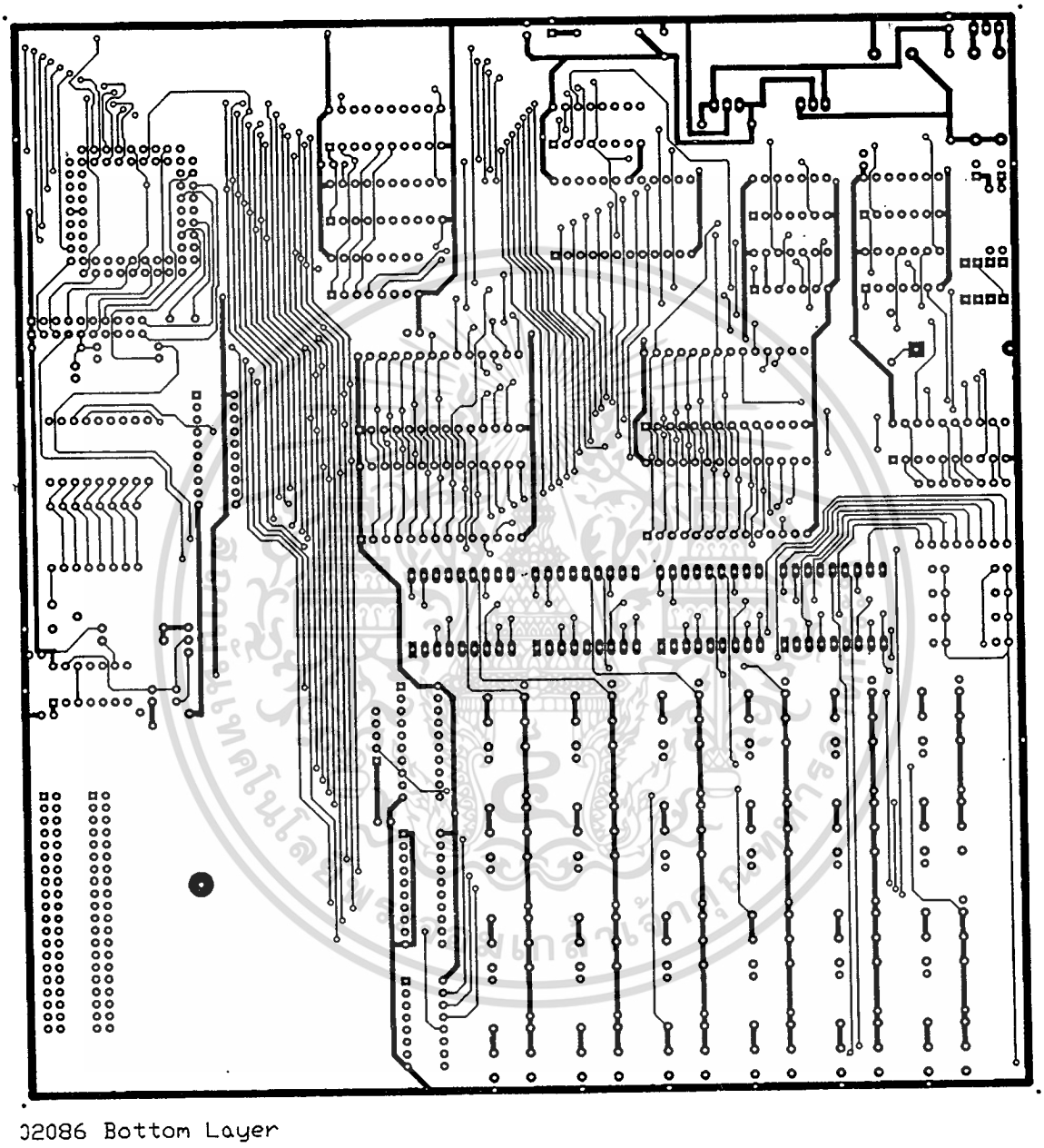
รูปที่ 2.17 ส่วนแสดงผลโดยใช้ 7 เซกเมนต์ และลำโพง



รูปที่ 2.18 แสดงการลงอุปกรณ์แผ่นวงจรชิงเกิลบอร์ด



รูปที่ 2.19 ลายวงจรพิมพ์ด้านบนของชิงเกิลบอร์ด



รูปที่ 2.20 ลายวงจรพิมพ์ด้านล่างของซิงเกิลบอร์ด

บทที่ 3

การใช้งาน Z280 และต่อเชื่อมกับอุปกรณ์ต่าง ๆ

3.1 วงจรกำเนิดความถี่ (OSCILLATOR)

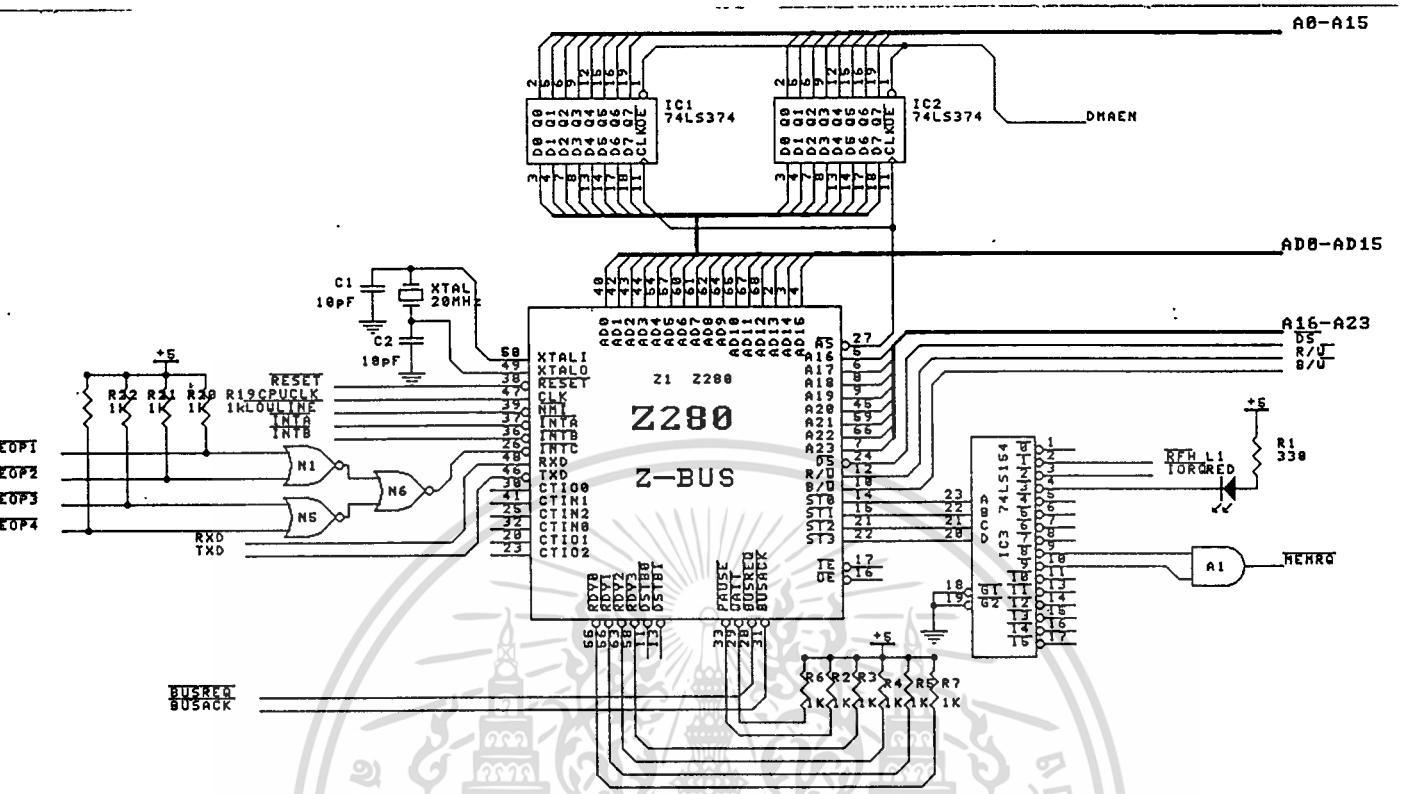
ไมโครโปรเซสเซอร์ Z280 มีวงจร Oscillator อยู่ภายในโดยสามารถต่อ X-TAL ได้โดยตรงหรือใช้สัญญาณ clock จากภายนอกก็ได้ แต่ทั้งสองกรณีความถี่ของระบบจะเป็นครึ่งหนึ่งของสัญญาณ clock ที่ป้อนเข้ามา ในโครงงานนี้ใช้ X-TAL ขนาด 20 MHz ต่อเข้าโดยตรงกับตัวไมโครโปรเซสเซอร์โดยคาปาซิเตอร์ค่าน้อยๆต่ออยู่เพื่อทำหน้าที่เป็น load capacitive จะมี clock ของระบบเท่ากับ 10 MHz

3.2 อินเตอร์รัพท์(Interrupt)

การอินเตอร์รัพท์ใน CPU เบอร์ Z280 แบ่งออกเป็น 2 ชนิด คือ NMI (Non-Maskable Interrupt) และ INT (Maskable Interrupt) แต่สำหรับในโครงงานนี้ NMI จะถูกใช้ในการตรวจสอบระบบไฟเลี้ยง การอินเตอร์รัพท์แบบ Maskable Interrupt ในระบบมีอยู่ 3 ขา INT A ใช้ในระบบเรียลไทม์คล็อก INT B ใช้ในวงจร Harddisk Interface ส่วน INT C ใช้ในการรับสัญญาณ EOP จาก DMA Controller

3.3 Address Bus และ Data Bus

เนื่องจากไมโครโปรเซสเซอร์ Z280 มีการอินเตอร์เฟสกับภายนอก 2 แบบ คือ Z80 BUS ซึ่งเป็น BUS ขนาด 8 bit ที่เหมือนกับ Z80 และ Z-BUS ซึ่งเป็น BUS ขนาด 16 bit โดยการจัดขาและสัญญาณทั้งสองแบบจะไม่เหมือนกัน ในโครงงานนี้เลือกการอินเตอร์เฟสแบบ Z-BUS โดยป้อน logic high ให้กับขา OPT ของตัวไมโครโปรเซสเซอร์ เนื่องจาก Z-BUS มีขาข้อมูลถึง 16 bit ดังนั้นเพื่อให้ประหยัดขาไอซี จึงใช้ขา address และ data ร่วมกันคือขา AD_0-AD_{15} จึงต้องใช้วิธี Multiplex ระหว่างขา address และ data โดยใช้ 74LS374 ซึ่งเป็น D-flipflop เป็นตัวแลทช์ข้อมูล และใช้ขา $\overline{AS}$ (Address Strobe) จากไมโครโปรเซสเซอร์ต่อตรงกับขา CLK ของ 74LS374 เพื่อแยกสัญญาณระหว่าง address bus และ data bus และ Z280 ยังมีขา address A_{16}-A_{23} อีก 8 เส้น รวมกันแล้วมี address bus 24 เส้น ทำให้สามารถอ้าง address หน่วยความจำได้ถึง 16 Mbyte แต่ในโครงงานนี้มีหน่วยความจำชนิด Dynamic Ram อยู่ 8 Mbyte และส่วนที่ เป็น Rom ของระบบอยู่ 64 kByte

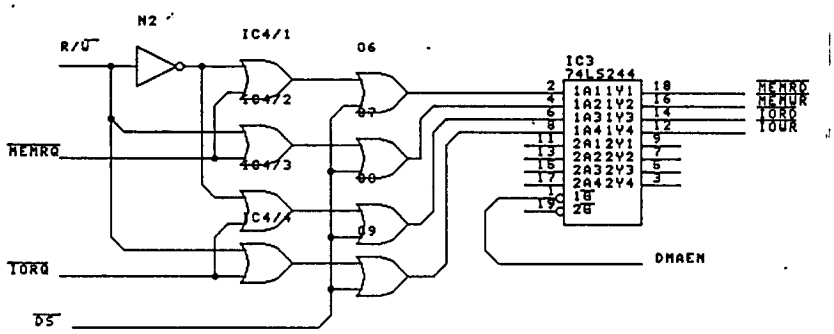


รูป 3.1 แสดง address bus data bus และ control bus ของระบบ

3.4 วงจรควบคุม

เนื่องจาก BUS แบบ Z-BUS ของไมโครโปรเซสเซอร์ Z280 ไม่มีขาควบคุมต่างๆโดยตรงเป็น control bus แต่มีขา status ST0-ST3 จึงจำเป็นต้องนำขา ST ทั้ง 4 ขามาถอดรหัสเพื่อสร้างสัญญาณควบคุมจากบทที่ 1 ในตารางที่ 1.9 เราต้องการขา status ST3-ST0 เป็น 0011 เพื่อสร้างสัญญาณ RFH ใช้ในการรีเฟรชหน่วยความจำ Dynamic Ram 0010 สร้างสัญญาณ $\overline{IORQ}$ ใช้ในการติดต่อกับอุปกรณ์อินพุท เอาพุทต่างๆ 0011 จะแอคทีฟเมื่อกระทำคำสั่ง HALT 1000 และ 1001 ใช้สร้างสัญญาณ $\overline{MREQ}$ ในที่นี้ใช้ 74LS154 ซึ่งเป็น 4 to 16 decoder ถอดรหัสสัญญาณจากขา status

และเนื่องจาก Z280 มีขาสัญญาณ R/$\overline{W}$ เส้นเดียวที่ใช้ระบุการอ่านหรือเขียน ทั้งหน่วยความจำและอุปกรณ์ I/O เราจึงจำเป็นต้องนำสัญญาณ R/$\overline{W}$, $\overline{MREQ}$, $\overline{IORQ}$ และ $\overline{DS}$ ซึ่งเป็นสัญญาณที่มีส่วนแคบที่สุดใน Timing Diagram มาสร้างสัญญาณ $\overline{MEMRD}$, $\overline{MEMWR}$, $\overline{IORD}$,$\overline{IOWR}$ ดังรูป 3.2 ซึ่งสัญญาณทั้ง 4 เส้น จะต้องผ่าน tri-state เนื่องจากเมื่อเกิดขบวนการ DMA สัญญาณควบคุมทั้ง 4 นี้จะมาจากตัว DMA Controller



รูปที่ 3.2 การสร้างสัญญาณ MEMRD ,MEMWR, IORD, IOWR

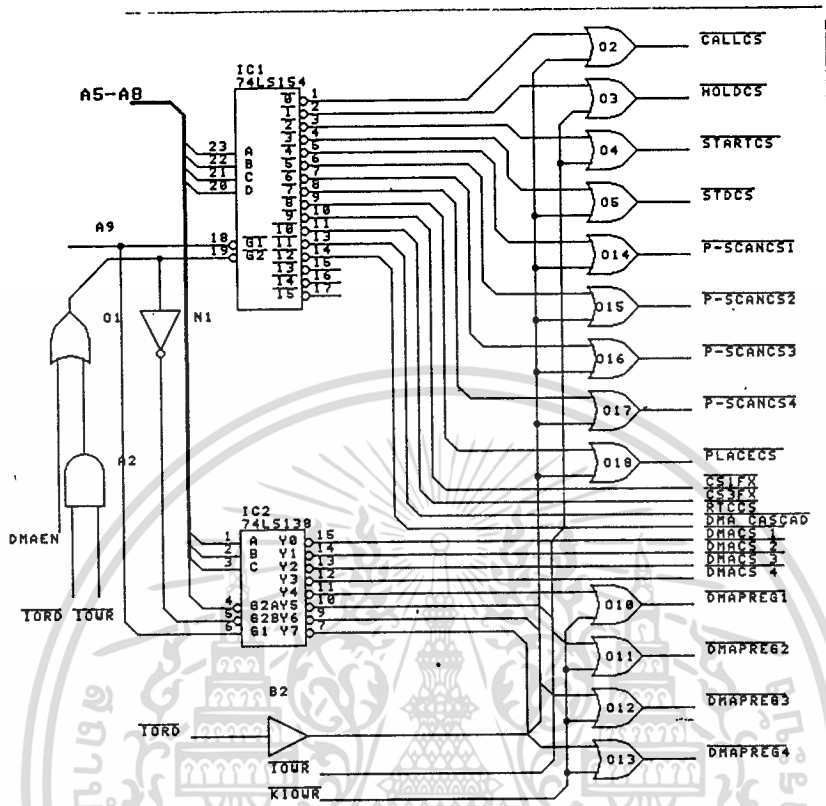
3.5 การจัดอินพุท/เอาท์พุทพอร์ท

เนื่องจากการจัดอินพุท เอาท์พุทพอร์ทของโครงงานเครื่องตอบรับโทรศัพท์นี้ ใช้การจัดแบบ ไอโอแมปไอโอ (I/O mapped I/O) คือพื้นที่การจัดอินพุท เอาท์พุท จะแยกออกจากพื้นที่หน่วยความจำอย่างสิ้นเชิง ซึ่งข้อดีของระบบนี้ทำให้พื้นที่ส่วนที่เป็นหน่วยความจำสามารถใช้งานได้เต็มที่โดยไม่เกี่ยวข้องกับส่วนที่เป็นอินพุท เอาพุทพอร์ท เหมาะสำหรับระบบงานที่ต้องการขยายให้มีขนาดใหญ่

ไมโครโปรเซสเซอร์ Z280 สามารถอ้างพอร์ทได้ 24 บิท (A_0-A_{23}) การจัดอินพุท เอาท์พุทพอร์ทแสดงในตาราง จะเห็นว่ามีพอร์ทใช้งานในวงจรนี้อยู่หลายพอร์ท โดยใช้สาย Address A_5-A_9 สัญญาณ IORD, IOWR และ DMAEN มาใช้ในการถอดรหัสเพื่อกำหนดหมายเลขพอร์ทโดยใช้ไอซี 74LS154 และ 74LS138 มาใช้ในการถอดรหัสสัญญาณเลือกพอร์ท

000H-00FH	CALLCS-
010H-01FH	HOLDCS-
020H-02FH	STARTCS-
030H-03FH	STDCS-
040H-04FH	P-SCANCS1-
050H-05FH	P-SCANCS2-
060H-06FH	P-SCANCS3-
070H-07FH	P-SCANCS4-
080H-08FH	PLACECS-
090H-09FH	CS1FX-
0A0H-0AFH	CS3FX-
0B0H-0BFH	RTCCS-
0C0H-0CFH	DMA CASCADE-
100H-10FH	DMACS1-
110H-11FH	DMACS2-
120H-12FH	DMACS3-
130H-13FH	DMACS4-
140H-14FH	DMAPREG1-
150H-15FH	DMAPREG2-
160H-16FH	DMAPREG3-
170H-17FH	DMAPREG4-

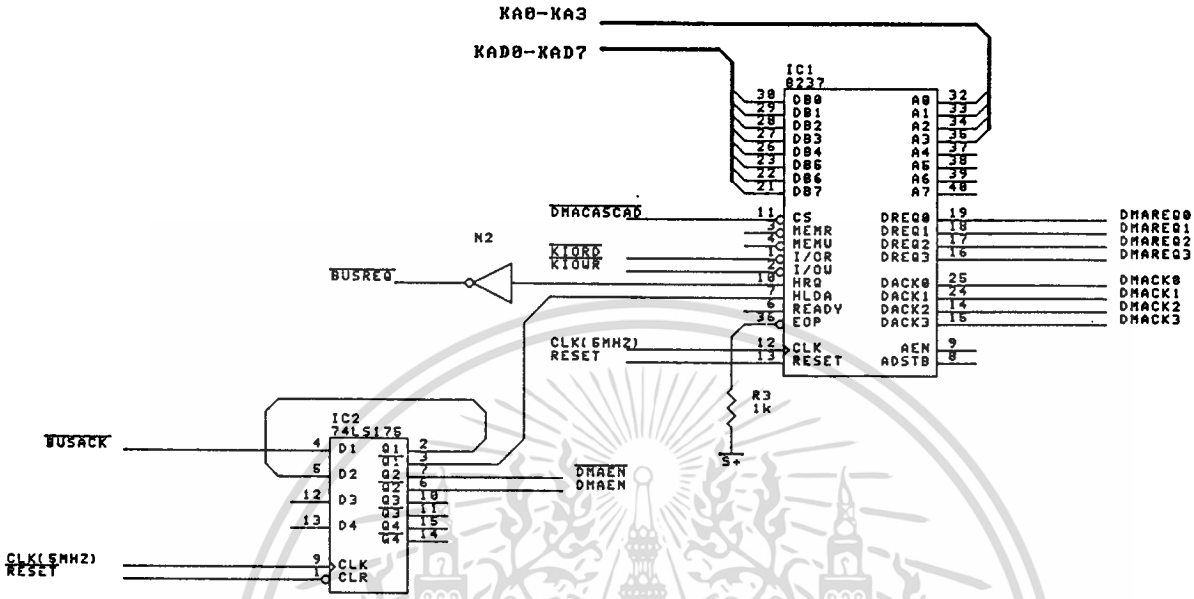
ตารางที่ 3.1 การจัดอินพุทเอาพุทพอร์ทของระบบ



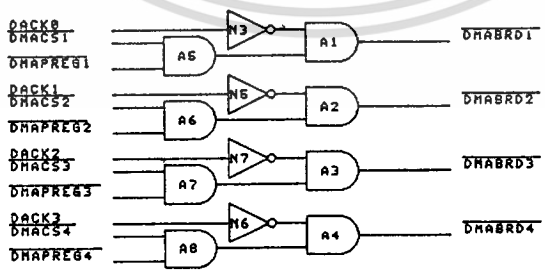
รูปที่ 3.3 วงจรถอดรหัสเลือกพอร์ทในระบบ

3.6 วงจร DMA Controller

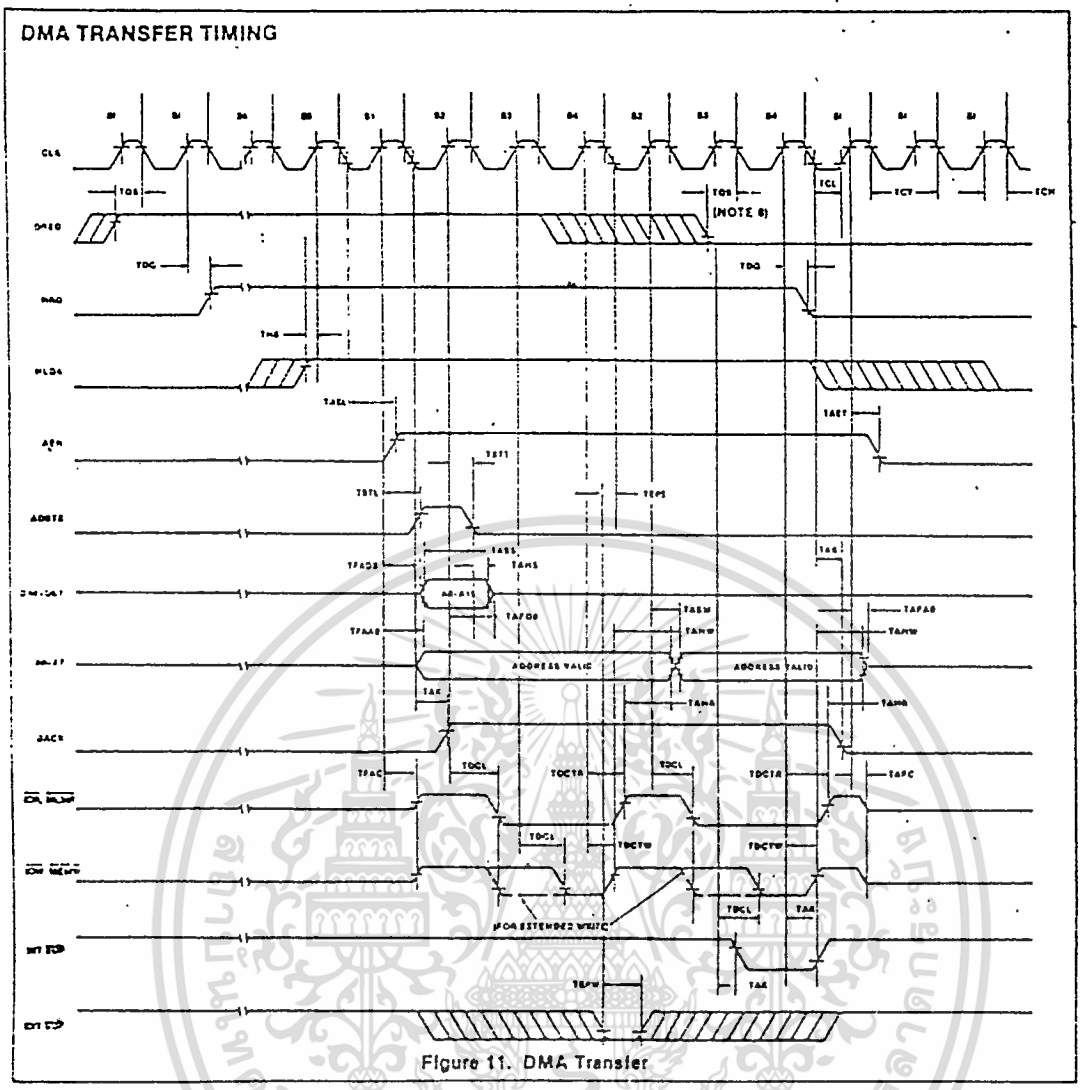
เนื่องจากการรับส่งข้อมูลระหว่างวงจรโทรศัพท์กับระบบใช้กระบวนการ DMA ซึ่งข้อมูลที่รับส่งจะไม่ผ่านตัว CPU แต่จะเกิดการติดต่อกันระหว่าง วงจรโทรศัพท์กับหน่วยความจำโดยตรง และวงจรส่วนนี้เป็นวงจร DMA Controller stage แรก ซึ่งถูกโปรแกรมให้มีการทำงานในโหมด cascade ทำหน้าที่เป็นทางผ่านของการของการร้องขอทำกระบวนการ DMA จาก DMA stage ที่สองเท่านั้น ส่วน DMA stage ที่สองจะอยู่บนการ์ดโทรศัพท์ ถูกโปรแกรมให้ทำงานในโหมด single transfer ซึ่งสามารถขยายจำนวน channel โทรศัพท์ได้ เมื่อเกิดขบวนการ DMA จะมีการ enable หรือ disable วงจร buffer ชุดต่างๆในระบบ เพื่อที่ตัว DMA Controller จะเป็นตัวควบคุมการทำงานแทน (รายละเอียด DMA อ่านเพิ่มเติมส่วน การใช้ DMA เก็บข้อมูลโดยไอซีเบอร์ 8237A-5) โดย Timing Digram แสดงได้ดังรูป



รูปที่ 3.4 วงจร DMA CONTROLLER STAGE ที่ 1



รูปที่ 3.5 วงจร enable/disable buffer



รูปที่ 3.6 DMA TRANSFER TIMING

3.7 วงจร SCAN

เนื่องจากในการติดต่อและควบคุมวงจรโทรศัพท์หลายๆ Channel จะมีสัญญาณควบคุมจำนวนมาก ในระบบจะมีการ scan ตรวจสัญญาณต่างๆ และส่งสัญญาณไปควบคุมวงจรโทรศัพท์โดยใช้พอร์ทต่างๆ ที่ถอดรหัสออกมา ซึ่งมีทั้งอินพุท เอาพุทพอร์ท โดยใช้ไอซี 74LS244 ทำหน้าที่เป็นพอร์ทอินพุท และ 74LS374 ทำหน้าที่เป็นพอร์ทเอาต์พุท ในระบบเมื่อใช้งานสูงสุด 16 channel จะใช้พอร์ทขนาด 16 บิทอยู่ 9 พอร์ท

3.8 วงจร Buffer

ในโครงงานนี้มีอุปกรณ์ I/O และหน่วยความจำอยู่หลายชุดจึงจำเป็นต้องมี วงจร Buffer อยู่หลายชุด โดย Buffer ชุด M ใช้ในการติดต่อกับหน่วยความจำ buffer ชุด X ใช้ต่อกับ connector buffer ชุด K ใช้ติดต่อกับวงจร scan ,DMA Controller stage 1 และ Harddisk Interface buffer ชุด T ใช้ในการติดต่อกับวงจรโทรศัพท์

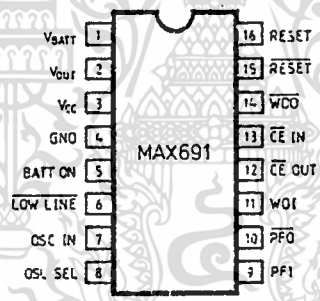
3.9 MAX691 ไอซีช่วยงาน CPU

MAX 691 เป็นชิพไอซีในตระกูล MAX 690 จากบริษัท MAXIM ทำหน้าที่เป็นผู้ช่วยงานที่จำเป็นหลายๆอย่างในระบบไมโครคอมพิวเตอร์ เพื่อลดอุปกรณ์ต่างๆลงและทำให้ระบบมีประสิทธิภาพและความเชื่อถือได้ในการทำงาน MAX691 สามารถทำงานได้โดยต่อกับอุปกรณ์ภายนอกเพียงเล็กน้อยเท่านั้น หน้าที่และการใช้งานต่างๆมีดังนี้

- รีเซต CPU เมื่อเริ่มจ่ายไฟให้กับระบบ(power up) และเมื่อแรงดันไฟเลี้ยงตกลงกว่าที่กำหนด
- ตัดต่อไฟเลี้ยงกับแบตเตอรี่ เพื่อจ่ายให้กับ ซีมอสแรม วงจรนาฬิกา หรืออุปกรณ์ลอจิกกำลังต่ำ เพื่อให้เก็บรักษาข้อมูลไว้ได้
- ให้สัญญาณ low batt เมื่อแรงดันจากแบตเตอรี่ตกลงกว่าที่กำหนด
- ป้องกันการเขียนข้อมูลใน RAM หรือ EEPROM ในสภาวะไฟเลี้ยงไม่พร้อม

การจัดขาและหน้าที่

MAX 691 เป็นไอซีแบบ DIP มีจำนวนขา 16 ขา ด้วยกันโดยหน้าที่และการใช้งานของแต่ละขามีดังนี้



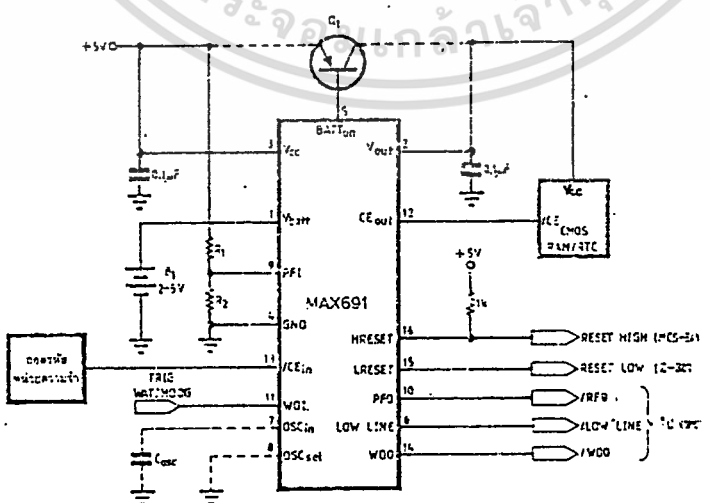
รูปที่ 3.7 แสดงการจัดขาของ MAX 691

- V_{BATT} ต่อกับขั้วบวกของแบตเตอรี่ เมื่อต้องการแบคอัพ CMOS RAM หรือ RTC ถ้าไม่ใช้ให้ต่อลงกราวด์
- V_{OUT} ให้ไฟบวกที่สูงกว่าจากไฟเลี้ยงหรือจากแบตเตอรี่ และต่อไปยังขาไฟเลี้ยงของอุปกรณ์ที่ต้องการแบคอัพ ถ้าไม่ใช้ให้ต่อให้ต่อขานี้เข้ากับไฟบวกของระบบ
- V_{CC} ไฟเลี้ยงขนาด 5 โวลท์
- GND กราวด์ของระบบ
- BATT ON เป็น high เมื่อขา V_{OUT} ได้แรงดันจากแบตเตอรี่ ขานี้จ่ายกระแสซิงค์ได้ 25 mA เพื่อขับทรานซิสเตอร์ pnp ภายนอก เพื่อเพิ่มกระแสที่ขา V_{OUT} ให้สูงขึ้น
- LOW LINE เป็น low เมื่อไฟเลี้ยงตกลงต่ำกว่า 4.65 โวลท์

- OSC_{in} เมื่อกำหนดให้ขา OSC_{sel} เป็น low ขานี้จะรับสัญญาณนาฬิกาภายนอก เพื่อกำหนดช่วงเวลาในการรีเซต และกำหนดช่วงเวลาสูงสุดของวงจรวอตช์ด็อก
- OSC_{sel} เลือกสัญญาณนาฬิกาจากภายนอกหรือภายในตัว MAX 691
- PFI ขาอินพุทของวงจรเปรียบเทียบตก เมื่อแรงดันที่ขานี้ตกลงต่ำกว่า 1.3 โวลท์ ทำให้มีสัญญาณ low ที่ขา PFO เพื่อใช้ในการอินเตอร์รัพท์ CPU ได้
- PFO เป็น low เมื่อแรงดันที่ขา PFI ต่ำกว่า 1.3 โวลท์
- WDI อินพุทสำหรับการกระตุ้นวงจรวอตท์ด็อก โดยการเปลี่ยนระดับของสัญญาณจากเอาพุทพอร์ท หรือสัญญาณ port select ก็ได้ แต่จะต้องกระตุ้นภายในช่วงเวลาที่กำหนด
- CE_{OUT} เป็น low ตามขา CE_{in} และในสภาวะที่ไฟเลี้ยงปกติ คือสูงกว่า 4.65 โวลท์เท่านั้น
- CE_{in} เป็นสัญญาณ chip enable ของ CMOS RAM หรือ RTC ที่ต้องการแบคอัพโดยขานี้จะต่อมาจากส่วนที่ถอดรหัสหน่วยความจำ
- WDO เป็น low ถ้า CPU ไม่กระตุ้นวงจรวอตช์ ภายในเวลาที่กำหนดและในสภาวะที่ไฟเลี้ยงต่ำกว่า 4.65 โวลท์ จะไม่มีการกระตุ้นวงจรวอตช์ด็อกทำให้ขานี้เป็น high ด้วย
- RESET LOW เป็นขาที่ให้สัญญาณรีเซต low เพื่อรีเซต CPU เช่น Z80
- RESET HIGH เป็นขาที่ให้สัญญาณรีเซต high เพื่อรีเซต CPU เช่น MCS-51

การใช้งาน

จากวงจรรูปที่ 3.8 ขาไฟเลี้ยงของซีมอสแรมหรือ RTC จะต่อกับขา V_{OUT} ของ MAX691 โดยขานี้จะได้แรงดันจากไฟเลี้ยงของระบบ แต่เมื่อไม่มีไฟเลี้ยงหรือไฟเลี้ยงตกลงต่ำกว่า 4.65 โวลท์ ขา V_{OUT} จะได้รับแรงดันจากแบตเตอรี่จะเห็นว่าแรงดันที่ขา V_{OUT} นี้จะสูงพอ ที่จะเลี้ยงวงจรส่วนนี้ได้ โดยขา V_{OUT} จ่ายกระแสได้ 50 mA แต่ถ้าต้องการกระแสมากขึ้น ก็สามารถต่อทรานซิสเตอร์จากภายนอกเข้าไปได้ รายละเอียดการใช้งานอื่นๆ มีดังต่อไปนี้



รูปที่ 3.8 แสดงวงจรการใช้งานจริงของ MAX 691

สัญญาณรีเซต

วงจรเปรียบเทียบแรงดันจะให้สัญญาณรีเซต เมื่อไฟเลี้ยงตกลงต่ำกว่า 4.65 โวลท์หรือเมื่อ CPU ไม่กระตุ้นวงจรวอทช์ด็อกภายในเวลาที่กำหนด สัญญาณรีเซตจะมีช่วงเวลา 50 มิลลิวินาทีหรือกำหนดจากสัญญาณที่ป้อนให้กับขา OSC_{in} จะเห็นว่าไม่จำเป็นต้องต่อตัวเก็บประจุที่ขารีเซตของ CPU เพราะภายใน MAX 691 ได้กำหนดช่วงเวลาในการรีเซตให้แล้ว สำหรับการรีเซต high จะต้องต่อตัวต้านทานพูลอัพเพื่อเพิ่มกระแสให้สามารถรีเซตได้ด้วย

การตรวจจับแรงดันตก

ไฟเลี้ยงของระบบหรือแรงดันจากแบตเตอรี่ จะถูกแบ่งด้วยตัวต้านทานภายนอก แล้วป้อนให้กับขา PFI (power fail input) เพื่อเป็นระดับในการเปรียบเทียบ การตกของไฟเลี้ยงหรือแบตเตอรี่ เช่น ถ้าต้องการตรวจจับไฟเลี้ยงที่ต่ำกว่า 4.65 โวลท์ จะเกิดสัญญาณ low ที่ขา PFO และให้เป็นการนอนมาสเคเบิลอินเตอร์รัพท์ให้แก่ CPU เพื่อทำการเก็บข้อมูลที่สำคัญไว้ในที่ปลอดภัยก่อน

การป้องกันการเขียนข้อมูลใน RAM

สัญญาณ CE_{OUT} ซึ่งใช้เป็นชิพอีนาเบิลให้กับชีมอสแรมที่ต้องการแบคอัพ จะเกิดตามสัญญาณที่ขา CE_{in} และในสภาวะที่ไฟเลี้ยงปกติคือสูงกว่า 4.65 โวลท์เท่านั้น เมื่อไฟเลี้ยงตกลงต่ำกว่านี้จะทำให้ขา CE_{OUT} เป็น high เป็นการป้องกันการเขียนข้อมูลได้ และทำให้ขา LOW LINE เป็น low เพื่อให้ CPU ตรวจสอบสถานะของระบบได้ด้วย

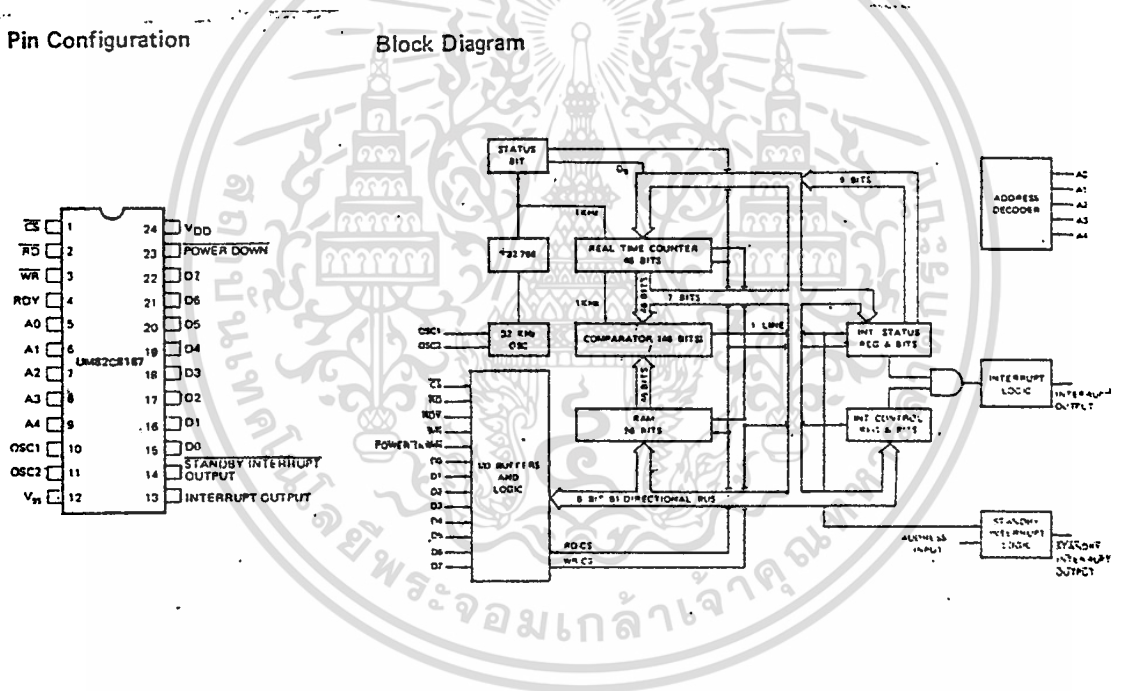
การตัดต่อแรงดัน V_{OUT}

แรงดันที่ขา V_{OUT} จะได้จากแรงดันที่สูงกว่า ระหว่างไฟเลี้ยงกับแบตเตอรี่และจ่ายกระแสได้ 50 มิลลิแอมป์ ซึ่งก็เพียงพอกับ CMOS RAM 1-2 ตัว แต่ถ้าต้องการกระแสที่สูงขึ้นก็ต่อทรานซิสเตอร์ pnp ขนานกับ V_{OUT} จากภายนอกได้โดยใช้สัญญาณจากขา BATT ON กระแส 25 มิลลิแอมป์ ควบคุม ขาเบส

3.10 RTC (REAL TIME CLOCK)

ในโครงงานนี้จำเป็นต้องใช้เวลาที่ถูกต้อง ดังนั้นจึงต้องมีส่วนที่สร้างเวลามาตรฐาน แม้ว่า Z280 จะมี COUNTER ในตัวซึ่งอาจจะใช้สร้างเวลามาตรฐานได้ แต่จะมีปัญหาเมื่อปิดเครื่อง และเปิดเครื่องใหม่ ข้อมูลเวลาที่มีอยู่ก็จะหายไปเสียแล้ว ดังนั้นจำเป็นต้องมีอุปกรณ์ตัวอื่นซึ่งสามารถให้เวลาที่ถูกต้องได้ แม้ว่าจะได้ปิดเครื่องและเปิดเครื่องใหม่แล้วก็ตาม ดังนั้นในโครงงานนี้จึงเลือกใช้ MM58167 ซึ่งเป็น RTC ซึ่งข้อมูลเวลาไม่เสียเมื่อปิดเครื่อง (ทั้งนี้ RTC ต้องมีแบตเตอรีสำรองด้วย)

MM58167 มีส่วนประกอบที่สำคัญคือ ตัวกำเนิดสัญญาณนาฬิกา (oscillator) ตัวนับเวลา (real time counter) ตัวเปรียบเทียบ (comparator) และ RAM การใช้งานเพียงแต่เขียนข้อมูลแบบ BCD ไปที่ตัวนับเวลา ตามแอดเดรสของเคาน์เตอร์แต่ละตัว เช่น วัน ชั่วโมง หรือ นาที ตัวนับเวลาก็จะเริ่มเดินตามเวลาที่ตั้งให้ และในทางตรงข้ามเราก็สามารถอ่านค่าของเวลาจากตัวนับเวลาได้เช่นกัน นอกจากนั้นผู้ใช้ยังสามารถตั้งเวลาในการให้สัญญาณเตือนได้ โดยสามารถตั้งให้ RTC ให้สัญญาณอินเตอร์เอาต์พุต ในวันเวลาใดก็ได้ โดยที่ตั้งเวลาไปที่ RAM เมื่อเวลาของตัวนั้นเดินมาตรงกับเวลาใน RAM จะทำให้เกิดการเปรียบเทียบขึ้นและให้สัญญาณอินเตอร์รัพท์เอาต์พุต ซึ่งส่วนประกอบที่สำคัญของ MM58167 แสดงได้ดังรูป 3.9



รูปที่ 3.9 แสดงการจัดวางขาต่างๆ และบล็อกไดอะแกรมของ MM58167

ตัวนับเวลา

ตัวนับเวลาเป็นตัวนับและจัดการเกี่ยวกับเวลา ถูกแบ่งเป็นดิจิตละ 4 บิท ซึ่งการเข้าถึงตัวนับเวลาจะกระทำครั้งละ 2 ดิจิต (ในขณะที่ read และ write) ซึ่งแต่ละดิจิตจะให้ค่า BCD ดังแสดงในตาราง บิทที่ไม่ใช้จะถูกโฮลด์ด้วย ลอจิก "0" ซึ่งเราไม่ต้องสนใจในขณะที่ทำการเขียนข้อมูลลงบนบัสข้อมูล เหตุที่บางบิทไม่ใช้ก็เนื่องจากว่า ไม่จำเป็นต้องใช้ในการให้ข้อมูลแบบ BCD ของบางหลัก ตัวอย่างเช่น ในหลักสิบของชั่วโมงจะไม่เกินเลข 2 ฉะนั้นเราจะใช้เพียง 2 บิทเท่านั้น ไม่ต้องใช้ในบิทที่ 6 และ 7

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

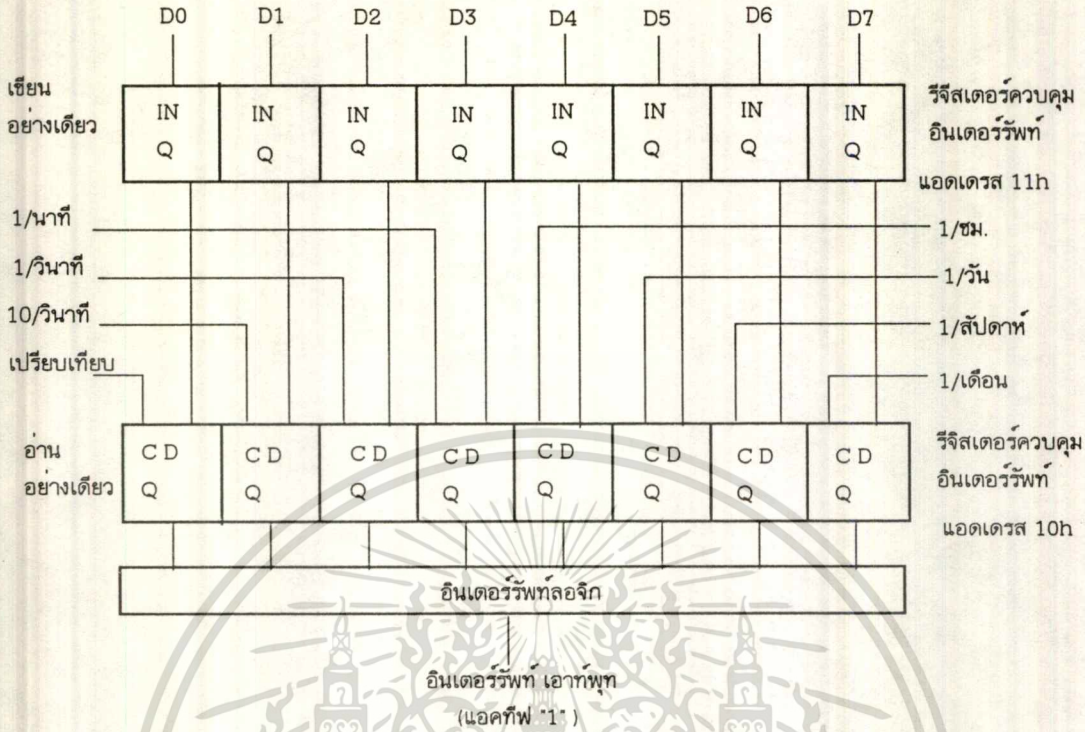
ตารางที่ 3.2 แสดงรายละเอียดของข้อมูลเป็นดิจิตในแต่ละเคาน์เตอร์แอดเดรส

RAM

MM58167 มี RAM ขนาด 56 บิท ใช้ในการเก็บข้อมูลเมื่อไฟดับหรือใช้เก็บข้อมูลการตั้งปลุกเพื่อที่จะเปรียบเทียบกับตัวนับเวลา ข้อมูลใน RAM จะสามารถเปรียบเทียบกับตัวนับเวลา และมีดิจิตที่ไม่ใช้คือ หลักหน่วยของ 1/1000 ของวินาที และหลักสิบของวันในสัปดาห์ (เพราะไม่ใช้ในตัวนับเวลา) RAM จะถูกกำหนดให้มีรูปแบบเหมือนกับตัวนับเวลา อย่างไรก็ตามยังมีบิทที่ยังไม่ใช้อยู่ ซึ่งบิทที่ไม่ใช้ในตัวนับเวลานี้จะเปรียบเทียบกับ "0" ใน RAM

อินเตอร์รัพท์และตัวเปรียบเทียบ

มีสัญญาณอินเตอร์รัพท์อยู่ 2 อย่าง อย่างแรกคือ อินเตอร์รัพท์เอาต์พุท (แอคทีฟ "1") เอาต์พุทนี้สามารถจะโปรแกรมให้เกิดสัญญาณทางออกได้ถึง 8 อย่าง คือ 10Hz, 1Hz, 1 นาที/ครั้ง, 1 ชั่วโมง/ครั้ง, 1 วัน/ครั้ง, 1 สัปดาห์/ครั้ง, 1 เดือน/ครั้ง วิธีการที่จะอีนาเบิลสัญญาณอินเตอร์รัพท์คือ ให้ลอจิก "1" แก่รีจิสเตอร์ควบคุมการอินเตอร์รัพท์ (interrupt control register) ในบิทที่ตรงกับความถี่ที่เราต้องการจะให้เกิดสัญญาณอินเตอร์รัพท์ (ดูรูปประกอบ) เช่น ต้องการให้สัญญาณอินเตอร์รัพท์ทุกๆ 1 นาทีก็ให้ D2 เป็น "1" เขียนไปที่รีจิสเตอร์ควบคุมการอินเตอร์รัพท์



รูปที่ 3.10 แสดงการอีนาเบิ้ลสัญญาณอินเตอร์รัพท์ต่างๆ

เราสามารถเซตรีจิสเตอร์ควบคุมอินเตอร์รัพท์ครั้งละ 1 บิท หรือมากกว่าก็ได้ ตัวอย่างเช่น ต้องการให้มีสัญญาณอินเตอร์รัพท์ทุกวินาที และทุกๆชั่วโมง ก็เซตบิทที่ 3 (D3) กับบิทที่ 2 (D2) ในที่นี้คือ 0Ch โดยเขียนไปที่แอดเดรสของรีจิสเตอร์ควบคุมอินเตอร์รัพท์

เมื่อเวลานับมาถึงค่าสูงสุดของแต่ละภาคจะทำให้เกิดคล็อก ให้กับรีจิสเตอร์ควบคุมอินเตอร์รัพท์ ซึ่งจะทำให้อินเตอร์รัพท์เอาต์พุทเป็น "1" (บิทใดบิทหนึ่งต้องถูกอีนาเบิ้ลด้วย) การอ่านรีจิสเตอร์ควบคุมอินเตอร์รัพท์ ทำให้เราทราบว่าสัญญาณอินเตอร์รัพท์เป็นสัญญาณของบิทใด อีกทั้งยังเป็นการรีเซลรีจิสเตอร์ควบคุมอินเตอร์รัพท์อีกด้วย

การอ่านรีจิสเตอร์ควบคุมอินเตอร์รัพท์นี้จะได้ข้อมูลบนบัสข้อมูล ซึ่งประกอบด้วยบิทที่ทำให้เกิดการอินเตอร์รัพท์โดยจะให้ค่าเป็น "1" ที่บิทนั้น หลังจากรอบของการอ่านจะทำให้รีจิสเตอร์ควบคุมอินเตอร์รัพต์ถูกรีเซต

อินเตอร์รัพท์อีกอย่างคือ สแตนด์บาย อินเตอร์รัพท์ (เอาท์พุทเป็นแบบโอเพ่นเดรน, แอคทีฟ "0") อินเตอร์รัพท์ตัวนี้จะเกิดขึ้นเมื่อเราได้ทำการอีนาเบิลไว้ และเกิดการเปรียบเทียบใน RAM กับตัวนับเวลา การอีนาเบิลทำได้โดยเขียน 01h ไปที่แอดเดรส 16h และในทางตรงกันข้ามถ้าให้ 00h ที่แอดเดรส 16h จะเป็นการดิสเอเบิล

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

ตาราง 3.3 แสดงแอดเดรสของฟังก์ชันต่างๆ

เพาเวอร์ดาวน์โหมด

ขาเพาเวอร์ดาวน์ (power down) เป็นตัวเลือกชิพที่สำคัญที่สอง มันจะดิสเอเบิลสัญญาณออกทั้งหมด ยกเว้น สัญญาณสแตนด์บายด์อินเตอร์รัพท์ (standby interrupt) เมื่อขานี้ได้รับลอจิก "0" MM58167 จะไม่ตอบสนองแก่สัญญาณจากภายนอกแต่นาฬิกาจะยังคงเดินตามปกติ และจะยังให้สัญญาณสแตนด์บายอินเตอร์รัพท์ (ขา 14) ถ้าได้มีการโปรแกรมให้ขานี้ทำงานไว้ก่อนแล้ว เมื่อต้องการเปลี่ยนจากโหมดการทำงานปกติมาเป็นสแตนด์บายโหมด (standby mode) ควรจะให้ขาเพาเวอร์ดาวน์เป็นลอจิก "0" อย่างน้อยที่สุด 1 μs ก่อนที่จะทำการลดระดับลงมาเป็นสแตนด์บายโหมด

เมื่อต้องการเปลี่ยนกลับมาสู่การทำงานปกติผู้ใช้ต้องมั่นใจว่าขาอินพุตอื่นๆ ต้องเป็นสัญญาณที่ถูกต้องก่อนที่จะกลับมาสู่โหมดการทำงานปกติ ทั้งนี้เพื่อป้องกันข้อมูลของนาฬิกาเสียไป จะทำให้นาฬิกาเดินผิด ตัวอย่างนี้ได้แก่ การที่ขา $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ของ MM58167 มีสัญญาณเปลี่ยนแปลงในขณะที่กลับสู่โหมดปกติ จะทำให้มีการเขียนข้อมูลไปที่ตัวนับเวลาหรือใน RAM

ตัวนับเวลาและ RAM รีเซต คำสั่ง GO

ตัวนับเวลา (counter) และ RAM สามารถรีเซตได้เขียน FFh ที่แอดเดรส 12h, 13h ตามลำดับ การให้พัลส์ของการเขียนไปที่แอดเดรส 15h (คำสั่ง GO) จะรีเซตตัวนับของวินาที ขณะทำการเขียนไปที่แอดเดรส 15h นี้ MM58167 จะไม่สนใจข้อมูลบนบัสข้อมูล แต่ผลของคำสั่ง GO มีดังนี้

ถ้าตัวนับของวินาทีนับได้มากกว่า 39 เมื่อเราใช้คำสั่ง GO (แอดเดรส 15h) จะทำให้หลักของนาทีเพิ่มขึ้น ในกรณีอื่นๆ จะไม่มีผลต่อหลักนาที

บิทสถานะ (status bit)

บิทสถานะจะบอกผู้ใช้ว่าขณะที่ทำการอ่านตัวนับนั้น ตัวนับกำลังอยู่ในช่วงของการอัพเดตเวลา ข้อมูลที่อ่านได้อาจมีการผิดพลาดเกิดขึ้น บิทสถานะนี้จะอ่านได้จากแอดเดรส 14h ของ RTC โดยจะให้ลอจิก "1" ที่บิท 0 ของบัสข้อมูล ในขณะที่บิทอื่นๆ เป็น "0" หมด ถ้าสัญญาณนี้ปรากฏขึ้นภายหลังการอ่าน ตัวนับควรมีการอ่านตัวนับใหม่ ที่ขอบขาลงของสัญญาณ read ที่แอดเดรส 14h จะรีเซตบิทสถานะด้วย

ตัวกำเนิดสัญญาณนาฬิกา

เป็นตัวกำเนิดสัญญาณนาฬิกาแบบเรโซแนนซ์ขนาน โดยใช้อุปกรณ์ภายนอกเพียงตัวเก็บประจุ 1 ตัว, ตัวต้านทาง 1 MΩ 1 ตัว, แร่กำเนิดความถี่ 1 ตัว โดยตัวต้านทานจะต่ออยู่ระหว่างขั้ว OSC in (ขา 10) และ OSC out (ขา 11) เพื่อที่จะไบอัสตัวอินเวอร์เตอร์ที่อยู่ภายในให้ทำงานอยู่ในช่วงที่เป็นเชิงเส้น สำหรับแร่แบบไมโครเพาเวอร์คริสตอลจะใช้ตัวต้านทานต่ออนุกรมกับขา OSC out โดยใช้ตัวต้านทางมีค่าโดยประมาณ 200 k ส่วนตัวเก็บประจุโดยปกติจะมีค่าอยู่ในช่วง 20 pF-25pF แร่ที่ใช้มีความถี่ 32768 Hz

คอนโทรลไลน์

สัญญาณ read, write, เลือกชิพเป็นสัญญาณอินพุททำงานที่ลอจิก "0" และสัญญาณ ready เป็นสัญญาณออก (โอเพ่น เดรนเอาต์พุต) ที่จุดเริ่มต้นของการอ่าน หรือการเขียน ขา ready จะให้สัญญาณเอาต์พุทเป็น "0" อยู่จนกระทั่งข้อมูลปรากฏบนบัสข้อมูลเรียบร้อย หรือข้อมูลได้ถูกแลตช์ไว้แล้วในขณะช่วงของการเขียน

3.11 การใช้ DMA ของระบบในการเก็บข้อมูลโดยไอซีเบอร์ 8237A-5

ขั้นตอนในขบวนการ DMA

1. ก่อนที่ 8237A-5 จะทำงานตามที่เราต้องการได้นั้น ต้องทำการโปรแกรมการทำงานที่ต้องการให้กับ 8237A-5 ก่อน คือ

   - กำหนดรูปแบบการส่งผ่านข้อมูลในขบวนการ DMA ว่าเป็นการเขียน (DMA write) หรืออ่าน (DMA read) ข้อมูล จากหน่วยความจำ
   - กำหนดลักษณะการส่งผ่านข้อมูลว่าเป็นการส่งผ่านข้อมูลเพียงไบท์เดียว หรือหลายไบท์ต่อการเกิดขบวนการ DMA 1 ครั้ง
   - จำนวนไบท์ของข้อมูลที่ต้องการจะรับหรือส่ง โดยใช้ขบวนการ DMA
   - กำหนดลำดับความสำคัญของแต่ละแชนแนลใน 8237A-5
   - ตำแหน่งแอดเดรสเริ่มต้น (ของหน่วยความจำ) ของการส่งผ่านข้อมูลในขบวนการ DMA
   - อีนาเบิลการขอ DMA ของแต่ละแชนแนลใน 8237A-5

   สำหรับการโปรแกรม 8237A-5 คือ out คำสั่งต่างๆ ที่ใช้ในการโปรแกรม 8237A-5 ให้กับแอดเดรสของพอร์ทที่เป็น 8237A-5

2. เมื่ออุปกรณ์หรือวงจรใดต้องการที่จะใช้ขบวนการ DMA อุปกรณ์นั้นก็จะส่งสัญญาณให้กับขา DREQ ขาใดขาหนึ่งของ 8237A-5 เพื่อขอใช้ขบวนการ DMA สำหรับขา DREQ จะมี 4 ขา หรือ 4 แชนแนล

3. แชนแนลทั้ง 4 ของ 8237A-5 สามารถเลือกใช้ได้ 2 แบบ คือ

   - Fixed Priority โหมดนี้แชนแนลที่ 0 จะมีลำดับความสำคัญสูงสุด แชนแนลที่ 3 มีลำดับความสำคัญต่ำสุด
   - Rotating Priority จะมีการทำงานคือ ในระหว่างที่ 8237A-5 ทำขบวนการ DMA ให้กับแชนแนลใดอยู่นั้น อุปกรณ์ภายนอกในแชนแนลอื่นๆ จะไม่สามารถขอใช้ขบวนการ DMA ซ้อนขึ้นได้ ไม่ว่าแชนแนลนั้นจะมีลำดับความสำคัญสูงกว่าหรือไม่ก็ตาม

4. เมื่อมีการขอ DMA 8237A-5 จะทำการตรวจสอบว่าเป็นของแชนแนลใด เมื่อทราบแล้วก็จะส่งสัญญาณ HRQ ออกมา ซึ่งจะมาเป็นสัญญาณ $\overline{\text{BUSREQ}}$ ให้กับ Z280 มีผลให้แอดเดรส บัสข้อมูลและบัสควบคุม มีสถานะเป็น Hi-Impedance และ Z280 จะส่งสัญญาณ $\overline{\text{BUSACK}}$ ออกมา ซึ่งจะกลับมาเป็นสัญญาณ HOLDA ให้กับ 8237A-5

5. เมื่อ 8237A-5 ได้รับสัญญาณ HOLDA แล้ว ก็จะส่งสัญญาณ DACK ไปให้กับอุปกรณ์ที่ขอ DMA เพื่อให้อุปกรณ์นั้นๆ ทราบว่า 8237A-5 พร้อมที่จะทำขบวนการ DMA ให้กับอุปกรณ์นั้นแล้ว

6. จากนั้น 8237A-5 จะเข้าควบคุมบัสต่างๆ ภายในระบบแทน Z280 โดยการส่งค่าแอดเดรสให้กับหน่วยความจำที่อุปกรณ์ขอ DMA นั้น ต้องการจะรับหรือส่งข้อมูลด้วย และจะส่งสัญญาณควบคุมต่างๆ ที่จำเป็นต้องใช้ในการส่งผ่านข้อมูล (สัญญาณ $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$) ออกมายังบัสควบคุมด้วย

7. หลังจากวงจรหรืออุปกรณ์ภายนอกนั้นได้รับสัญญาณ DACK แล้ว ก็จะยกเลิกสัญญาณ DRQ ที่ส่งให้กับ 8237A-5 จากนั้นเมื่อ 8237A-5 เสร็จจากการทำงานในขบวนการ DMA แล้ว ก็จะยกเลิกสัญญาณ HRQ จะทำให้ Z280 ยกเลิกสัญญาณ HOLDA ที่ส่งให้กับ 8237A-5 เป็นผลให้ Z280 กลับเข้าสู่การทำงานปกติต่อจากที่ได้ทำไว้

รีจิสเตอร์ Address และ Counter ใน 8237A-5

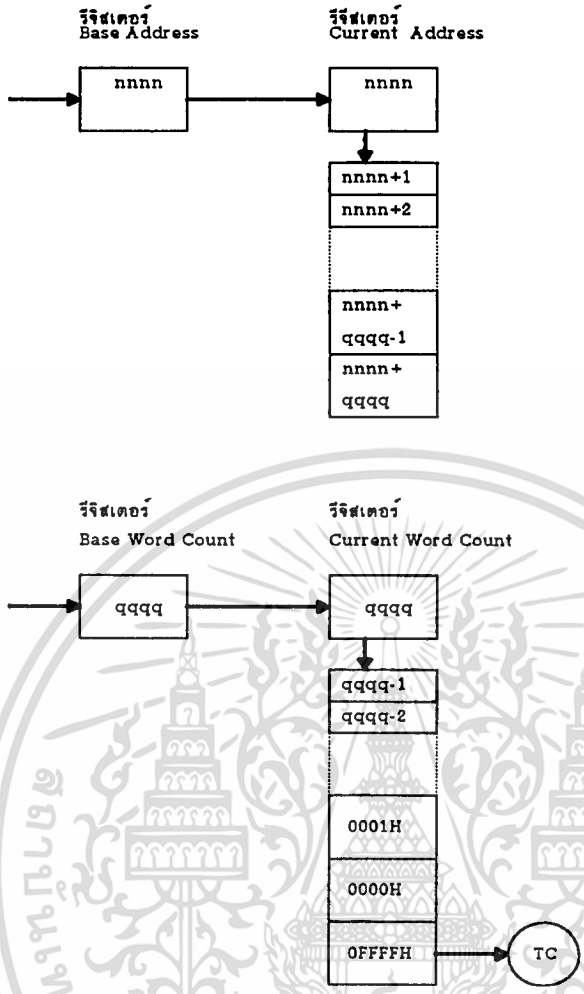
รีจิสเตอร์ในกลุ่มของ Address และ Counter เป็นรีจิสเตอร์ขนาด 16 บิท ซึ่งจะมีอยู่ทุกแชนแนลของ 8237A-5 โดยแต่ละแชนแนลจะมีรีจิสเตอร์ในกลุ่มนี้อยู่ 4 รีจิสเตอร์ คือ Base Address, Base Counter, Current Address และ Current Counter

1. รีจิสเตอร์ Base Address ใช้สำหรับกำหนดค่าแอดเดรสเริ่มต้นของหน่วยความจำที่ใช้ในการรับ/ส่งข้อมูลให้กับอุปกรณ์ภายนอกในระหว่างขบวนการ DMA

2. รีจิสเตอร์ Current Address ข้อมูลภายในรีจิสเตอร์จะเป็นข้อมูลที่แสดงถึงค่าแอดเดรสของหน่วยความจำที่ 8237A-5 จะกำหนดให้รับ/ส่ง ข้อมูลในไซเคิลต่อไปของขบวนการ DMA สำหรับข้อมูล (ค่าแอดเดรส) ภายในรีจิสเตอร์นี้จะถูกกำหนดโดย 8237A-5 (เราสามารถอ่านข้อมูลจากรีจิสเตอร์นี้ได้ แต่จะเขียนข้อมูลลงในรีจิสเตอร์นี้ไม่ได้) กล่าวคือ เมื่อเราทำการส่งค่าแอดเดรสให้กับรีจิสเตอร์ Base Address แล้ว 8237A-5 จะก็อปปี้ข้อมูลจากรีจิสเตอร์ Base Address มาให้กับรีจิสเตอร์ Current Address โดยอัตโนมัติ จากนั้นข้อมูลภายในรีจิสเตอร์นี้จะถูกเพิ่มขึ้นหรือลดลง 1 หลังจากที่ 8237A-5 เสร็จจากการทำงานในแต่ละไซเคิลของการส่งผ่านข้อมูลในขบวนการ DMA

3. รีจิสเตอร์ Base Word Count ข้อมูลในรีจิสเตอร์นี้จะใช้สำหรับกำหนดจำนวนไบท์ทั้งหมดของข้อมูลที่ต้องการจะส่งผ่านในขบวนการ DMA ของแต่ละแชนแนล เมื่อ 8237A-5 ทำขบวนการส่งผ่านข้อมูลในแชนแนลใดครบตามจำนวนไบท์ที่กำหนดในรีจิสเตอร์ Base Word Count ของแชนแนลนั้นแล้ว สัญญาณ TC (Terminal Count) จะแอคทีฟ

4. รีจิสเตอร์ Current Word Count ข้อมูลในรีจิสเตอร์นี้จะแสดงจำนวนไบท์ของข้อมูลที่เหลืออยู่ในการส่งผ่านข้อมูลของขบวนการ DMA ของแต่ละแชนแนล สำหรับข้อมูลในรีจิสเตอร์ Current Word Count จะถูกกำหนดโดย 8237A-5 คือ เมื่อเราทำการส่งข้อมูลให้กับรีจิสเตอร์ Base Word Count ของแชนแนลใด 8237A-5 ก็จะทำการก็อปปี้ข้อมูลนั้นไปให้กับรีจิสเตอร์ Current Word Count ของแชนแนลนั้นด้วย



รูปที่ 3.11 การอ้างแอดเดรสของไอซี 8237A-5

จากไดอะแกรมข้างต้น เมื่อรีจิสเตอร์ Base Address และรีจิสเตอร์ Base Word Count ได้รับข้อมูล (ที่ถูกส่งมาจากซีพียู) จากบัสข้อมูลแล้ว 8237A-5 จะก็อปปี้ข้อมูลนั้น (ในไดอะแกรม คือ nnnn และ qqqq) ส่งให้กับรีจิสเตอร์ Current Address และ Current Word Count ตามลำดับ จากนั้นเมื่อ 8237A-5 เสร็จจากการทำขบวนการส่งผ่านข้อมูลไบท์แรกในขบวนการ DMA ค่าในรีจิสเตอร์ Current Address จะเพิ่มขึ้นเป็น nnnn + 1 และค่าในรีจิสเตอร์ Current Word Count จะลดลงเป็น qqqq - 1 ซึ่งค่าของ Current Address และ Current Word Count จะเพิ่มขึ้นและลดลง 1 ในทุกครั้งที่ 8237A-5 เสร็จจากการทำขบวนการส่งผ่านข้อมูลในแต่ละไบท์ และเมื่อค่าของข้อมูลในรีจิสเตอร์ Current Word Count ถูกลดลงจนมีค่าเป็น 0FFFF H ก็จะทำให้สัญญาณ TC แอคทีฟ

โหมดการทำงานของ 8237A-5

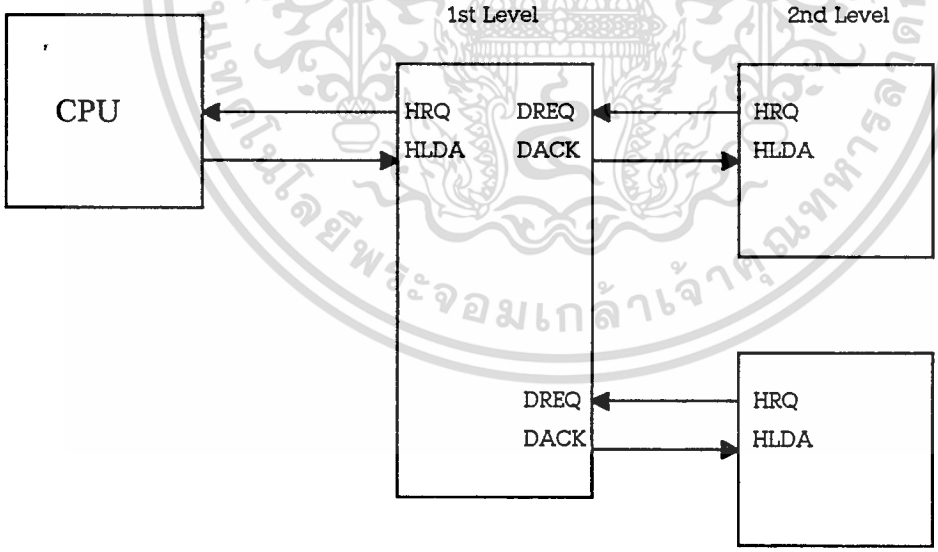
1. โหมด Single Transfer 8237A-5 จะทำการส่งผ่านข้อมูลระหว่างหน่วยความจำกับวงจร หรือ อุปกรณ์ภายนอกเพียง 1 ไบท์ และทำการลดค่าในรีจิสเตอร์ Current Word count ลง 1 รวมทั้งลดหรือเพิ่มค่าในรีจิสเตอร์ Current Address ลงหรือขึ้น 1 ตามที่

กำหนดโดยบิท 5 ของรีจิสเตอร์ Mode กรณีที่ค่าในรีจิสเตอร์ Current Word Count ถูกลดลงจนมีค่าเป็น 0FFFFH สัญญาณ TC จะถูกส่งออกมาทางขา EOP

2. โหมด Blocked Transfer เมื่อสัญญาณที่ขา DRQ ถูกทำให้แอคทีฟ 8237A-5 จะเริ่มต้นขบวนการ DMA เพื่อการส่งผ่านข้อมูลระหว่างหน่วยความจำกับอุปกรณ์ภายนอกโดยการส่งผ่านข้อมูลนี้จะดำเนินไปจนกว่าค่าในรีจิสเตอร์ Word Count จะถูกลดค่าลงจนเป็น 0FFFF H หรือจนกว่าวงจรภายนอกจะทำให้สัญญาณที่ขา EOP ของ 8237A-5 แอคทีฟ สำหรับสัญญาณที่ขา DRQ นั้นไม่จำเป็นต้องแอคทีฟอยู่จนเสร็จขบวนการส่งผ่านข้อมูล แต่จะต้องแอคทีฟอยู่จนกว่าสัญญาณที่ขา DACK ของ 8237A-5 จะแอคทีฟ เมื่อ 8237A-5 เข้าสู่การทำงานในโหมด Blocked Transfer แล้ว 8237A-5 จะไม่ยอมรับการขอ DMA ใดๆ ที่เกิดซ้อนขึ้น ไม่ว่าการขอ DMA นั้นจะเกิดขึ้นในแชนแนลที่มีลำดับความสำคัญสูงกว่าแชนแนลที่ 8237A-5 ตอบสนองอยู่หรือไม่ก็ตาม

3. โหมด Demand Transfer การส่งผ่านข้อมูลในโหมดนี้มีลักษณะการทำงานคล้ายกับในโหมด Block Transfer แต่จะมีข้อแตกต่างคือ ในโหมด Block Transfer นั้น หลังจากที่สัญญาณ DACK ของ 8237A-5 แอคทีฟแล้ว วงจรหรืออุปกรณ์ภายนอกที่ขอใช้ DMA สามารถยกเลิกสัญญาณ DRQ ได้หลังจากได้รับสัญญาณ DACK จาก 8237A-5 โดยที่ 8237A-5 ยังคงทำการส่งผ่านข้อมูลต่อไปได้จนข้อมูลในรีจิสเตอร์ Current Word Count ถูกลดลงจนเป็น 0FFFF H หรือสัญญาณที่ขา EOP ของ 8237A-5 ถูกทำให้แอคทีฟ แต่ในโหมด Demand Transfer ถ้าวงจรหรืออุปกรณ์ภายนอกยกเลิกสัญญาณ DRQ เมื่อใด 8237A-5 ก็จะหยุดขบวนการส่งผ่านข้อมูลด้วย

4. โหมด Cascade การทำงานในโหมดนี้จะช่วยให้เราสามารถใช้งาน 8237A-5 ในระบบร่วมกันได้มากกว่า 1 ตัว โดยการต่อกันแบบ Cascade ซึ่งจะทำให้เพิ่มจำนวนแชนแนลสำหรับการขอ DMA ได้มากกว่า 4 แชนแนล

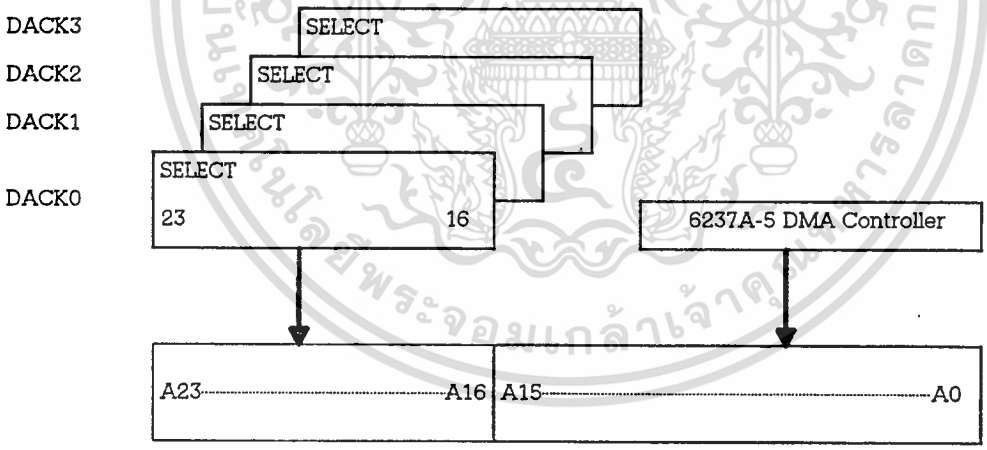


รูปที่ 3.12 การต่อไอซี 8237A-5 ในโหมด Cascade

การอ้างแอดเดรสของ 8237A-5

จากลักษณะการจัดเรียงขาของ 8237A-5 จำนวนขาที่เป็นบัสแอดเดรสของ 8237A-5 จะมีเพียง 8 ขา คือ A_0-A_7 โดยที่ขา A_0-A_3 เป็นแบบ Bi-Directional ซึ่งจะใช้เป็นอินพุทเมื่อซีพียูต้องการจะติดต่อกับรีจิสเตอร์ภายในของ 8237A-5 แต่ในระหว่างที่ 8237A-5 สร้างขบวนการ DMA เพื่อช่วยในการส่งผ่านข้อมูลระหว่างหน่วยความจำกับอุปกรณ์ภายนอก (หรือระหว่างหน่วยความจำ) ขา A_0-A_3 จะถูกใช้เป็นเอาท์พุทร่วมกับ A_4-A_7 ในการอ้างแอดเดรสของหน่วยความจำที่เกี่ยวข้อง ซึ่งจะเห็นได้ว่าไม่เพียงพอที่จะอ้างแอดเดรสได้ครบทั้ง 64 Kbyte ดังนั้น 8237A-5 จึงอาศัยวิธีการมัลติเพล็กซ์ค่าแอดเดรส 8 บิทบนที่เหลือ คือ A_8-A_{15} ออกมาบนขา DB0-DB7 ของ 8237A-5 โดย 8237A-5 จะใช้ช่วงเวลาของสัญญาณนาฬิกา S_1 ในแต่ละบัสไซเคิล ส่งค่าแอดเดรส 8 บิทบนออกมาบนขา DB0-DB7 เพื่อส่งให้กับอุปกรณ์ที่ทำหน้าที่แลทช์ค่าแอดเดรส 8 บิทบนให้ออกมาบนบัสแอดเดรสของระบบ โดยขอบขาลงของสัญญาณจากขา ADSTB จะถูกใช้สำหรับควบคุมให้อุปกรณ์นี้ทำการแลทช์ค่าแอดเดรส A_8-A_{15} ซึ่ง 8237A-5 ส่งออกมาบนขา DB0-DB7

ในการอ้างแอดเดรสของ Z280 จะสามารถอ้างได้ถึง 16 Mbyte ซึ่งมากเกินกว่าที่ 8237A-5 จะอ้างถึงได้ จึงต้องอาศัยวิธีการทางฮาร์ดแวร์เข้าช่วยโดยจะสร้างแอดเดรสอีก 8 บิทบน คือ A_{16}-A_{23} โดยเรียกว่า Page Register ซึ่งการเขียนหรืออ่านข้อมูล (แอดเดรส A_{16}-A_{23}) จาก Page register จะทำได้โดยการเขียนหรืออ่านข้อมูลผ่านทางพอร์ท I/O ของ Z280 ข้อมูลที่เก็บอยู่ใน Page Register จะถูกส่งออกมายังเส้นแอดเดรส A_{16}-A_{23} ของบัสแอดเดรสเมื่อ 8237A-5 ตอบสนองต่อการขอ DMA ที่เกิดขึ้นโดย ถ้า 8237A-5 ทำการตอบสนองให้กับการขอ DMA ในแชนแนลใด Page Register ก็จะส่งแอดเดรส 8 บิทบนของแชนแนลนั้นออกมาบนบัสแอดเดรส เมื่อรวมกับแอดเดรส 16 บิทที่ 8237A-5 ส่งออกมาแล้ว ก็จะครบ 24 บิทพอดี สำหรับวงจรในส่วนของ Page Register จะใช้ IC เบอร์ 74LS670 2 ตัว



รูปที่ 3.13 การอ้างแอดเดรสแบบเพจรีจิสเตอร์

บทที่ 4

การต่อเชื่อมกับหน่วยความจำ

4.1 การจัดหน่วยความจำ

หน่วยความจำเป็นส่วนหลักที่สำคัญมากในโครงงานเครื่องตอบรับโทรศัพท์ การจัดตำแหน่งแบ่งสรรพื้นที่ใช้งานเป็นสิ่งที่สำคัญ เพื่อให้ CPU และ DMA Controller รู้ว่าจะทำงานอย่างไร ซึ่งหน่วยความจำในโครงงานเครื่องตอบรับโทรศัพท์แบ่งเป็นหน่วยความจำ 64 Kbyte และ หน่วยความจำส่วนที่ เป็น Dynamic Ram 8 MByte

Z280 มีขาแอดเดรส 24 เส้นจึงสามารถอ้างแอดเดรสได้ 2^{24} = 16777216 ตำแหน่งหรือ 16 MByte โดยถูกออกแบบให้แต่ละแอดเดรสสามารถติดต่อข้อมูลในหน่วยความจำได้ 8 บิทหรือ 1 Byte จึงทำให้ติดต่อข้อมูลได้ทั้งหมด 16 MByte โดยข้อมูลที่ตัว CPU อ่านได้แต่ละครั้ง จะเป็นได้ทั้งแบบ ครั้งละ Byte (8 bit) หรือครั้งละ word (16 bit) โดยแต่ละส่วนก็จะมีสัญญาณมาควบคุมให้ทำงานแยกจากกันโดยใช้สัญญาณ A0 และ B/$\overline{W}$ ซึ่งสัญญาณใดจะทำงานขึ้นอยู่กับขนาดของข้อมูลที่ต้องติดต่อ

การเลือกหน่วยความจำว่าเป็นแอดเดรสคู่หรือคี่จะใช้ A_0 โดยถ้า A_0 เป็น 0 ซีพียูจะติดต่อกับหน่วยความจำทาง D8-D15 และถ้า A_0 เป็น 1 ซีพียูจะติดต่อกับหน่วยความจำทาง D0-D7 ทั้งนี้สัญญาณ B/$\overline{W}$ จะต้องเป็น 1 ด้วยเพื่อบ่งบอกถึงการติดต่อกับข้อมูลเป็น Byte ถ้าสัญญาณ B/$\overline{W}$ เป็น 0 ซีพียูจะติดต่อกับข้อมูลเป็น word ซึ่งจะใช้ขา data bus D0-D15 ส่วนเมื่อเกิดขบวนการ DMA การรับส่งข้อมูลระหว่างวงจรส่วนโทรศัพท์กับหน่วยความจำจะเป็นการติดต่อแบบ Byte

ในโครงงานนี้ใช้ EPROM เบอร์ 27256 และ SIMM RAM เบอร์ 21019 ซึ่งเป็น Dynamic RAM ขนาด 1 Mbyte จำนวน 8 Mbyte ช่วงละ 2 MByte

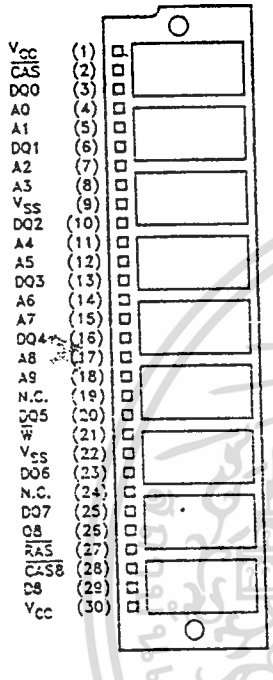
4.2 การใช้งาน Dynamic RAM

เนื่องจาก Dynamic Ram มีความจุต่อชิพสูงขนาดเล็ก กินกำลังไฟต่ำ และราคายังถูกกว่าแบบ static Ram เมื่อมีขนาดความจุเดียวกัน ถึงแม้ว่าการใช้งานจะยุ่งยากกว่าก็ยังคุ้มค่ากว่าการใช้ Static Ram ข้อยุ่งยากก็คือ Dynamic Ram จะต้องมีการรีเฟรช (refresh) ตลอดเวลา การรีเฟรชของ Dynamic Ram คือการทำขบวนการอ่านซ้ำเข้าไปในหน่วยความจำ ทั้งนี้เนื่องจากว่า กรรมวิธีการเก็บข้อมูลภายในเซลของหน่วยความจำแบบ Dynamic Ram จะมีลักษณะคล้ายกับการเก็บประจุของตัวเก็บประจุ (capacitor) ซึ่งจะต้องมีการเติมประจุอยู่เรื่อยๆ ตัวเก็บประจุจึงสามารถเก็บประจุนั้นอยู่ได้ การรีเฟรชก็เช่นกัน จะทำให้ Dynamic Ram สามารถเก็บข้อมูลอยู่ได้

4.3 ลักษณะการอ่าน,เขียน และรีเฟรชของ Dynamic RAM

เนื่องจากว่าการจัดขาแอดเดรสของ Dynamic Ram เป็นแบบมัลติเพล็กซ์แอดเดรส จึงทำให้จำนวนขาของ Dynamic Ram มีไม่มาก ดังในรูปที่ 4.1 เป็นการจัดขาต่างๆของ Dynamic Ram เบอร์ 21019 จากรูปที่ 4.1 จะเห็นได้ว่ามีขาแอดเดรสเพียง 10 ขา (A0-A9) เท่านั้น ซึ่งสัญญาณแอดเดรสนี้จะถูกมัลติเพล็กซ์เข้าไป โดยแบ่งเป็นโรวแอดเดรส (Row Address) และคอลัมน์แอดเดรส (Column Address) อย่างละ 10 บิท รวมเป็น 20 บิทแอดเดรส ในการอ่านหรือเขียนข้อมูลจะต้องส่งสัญญาณแอดเดรสไบท์ที่มีค่าเลข

นัยสำคัญต่ำ (A1-A10) เข้าไปที่ SIMM RAM ก่อน โดยผ่านทางขา A0-A9 ซึ่งชุดแอดเดรสส่วนนี้เรียกว่าโรวแอดเดรสโดยจะต้องสร้างสัญญาณโรวแอดเดรสสโตรบ (Row Address Strobe ,$\overline{RAS}$) ซึ่งเป็นสัญญาณแลทช์โรวแอดเดรสให้กับ SIMM RAM จากนั้นจึงส่งสัญญาณแอดเดรสไบท์ที่มีค่าเลขนัยสำคัญสูง (A11-A20) เข้าไปยัง SIMM RAM โดยผ่านทางขา A0-A9 เช่นเดียวกัน ซึ่งชุดแอดเดรสส่วนนี้เรียกว่า คอลัมน์แอดเดรส โดยจะต้องสร้างสัญญาณคอลัมน์สโตรบ (Column Address Strobe) ซึ่งเป็นสัญญาณแลทช์คอลัมน์แอดเดรสให้กับ SIMM RAM ด้วย



Pin Names

A0–A9	Address Inputs
DQ1–DQ7	Data Inputs/Outputs
D8	Data Input
Q8	Data Output
$\overline{RAS}$	Row Address Strobe
$\overline{CAS}$–$\overline{CAS8}$	Column Address Strobe
$\overline{W}$	R/W Input
V_{CC}	Power (+5V)
V_{SS}	Ground
N.C.	No Connection

รูปที่ 4.1 แสดงการจัดขาของ SIMM RAM

4.4 ช่วงเวลาที่สำคัญในการพิจารณาออกแบบ

4.4.1 ช่วงเวลาการอ่าน

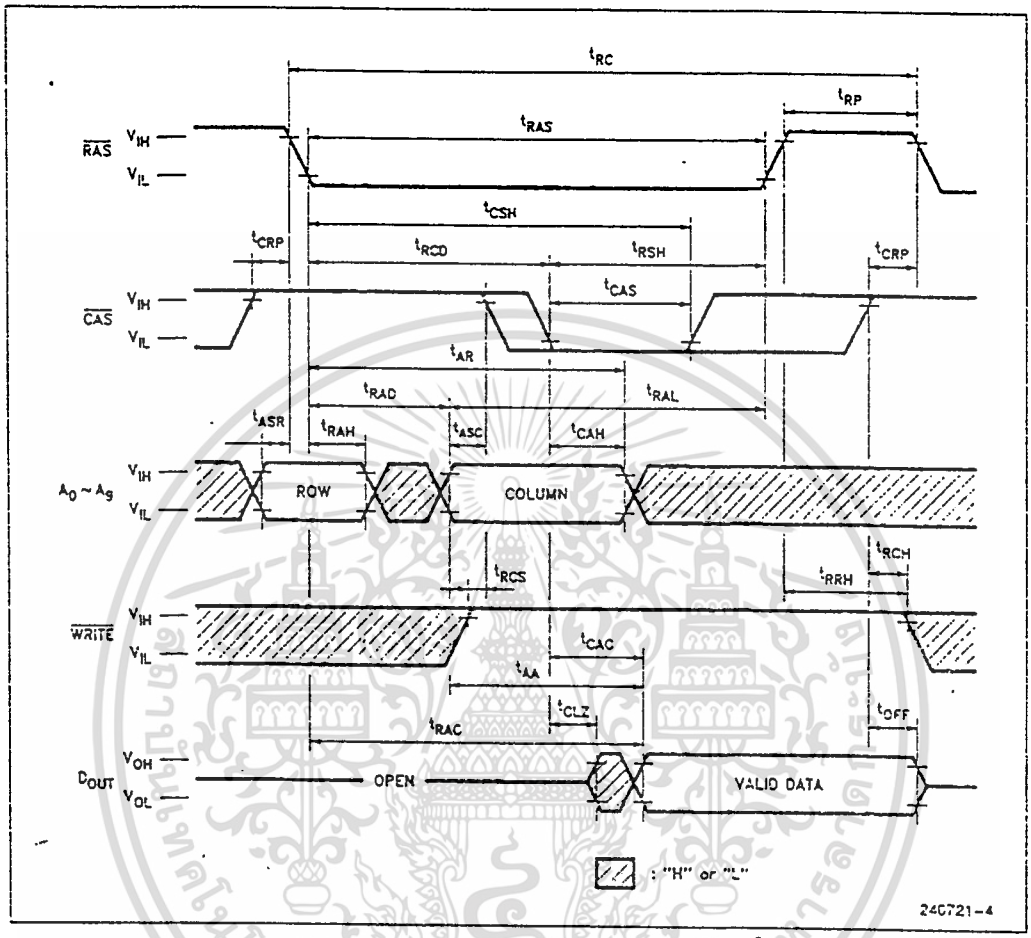
สำหรับช่วงเวลาการอ่านแสดงแผนผังเวลาในรูปที่ 4.2 มีช่วงเวลาที่สำคัญในการพิจารณาคือ ค่าของ t_{RCD} ซึ่งเป็นช่วงหน่วงเวลาระหว่างสัญญาณ RAS และ CAS จากตารางที่ 4 จะเห็นว่า t_{RCD} มีค่าต่ำสุดเป็น 20 ns (สำหรับ 21019-06 ซึ่งมี access time เป็น 60 ns) และ 25 ns (สำหรับ 21019-08 ซึ่งมี access time เป็น 80 ns) และมีค่าสูงสุดเป็น 40 ns (21019-06) และ 60 ns (21019-08) ค่าสูงสุดเป็นค่าที่เชื่อถือได้แน่นอนว่าทำงานได้ ดังนั้นในการออกแบบควรจะให้ t_{RCD} มีค่าไม่น้อยกว่า 40 ns (21019-06) หรือ 60 ns (21019-08)

A.C. CHARACTERISTICS[1, 2] (T_A = 0°C to 70°C, V_{CC} = 5V ±10%)

Symbol	Parameter	21019-06		21019-07		21019-08		21019-10		Units	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
t_{RAC}	Access Time from $\overline{RAS}$		60		70		80		100	ns	4, 7
t_{CAC}	Acces Time from $\overline{CAS}$		20		20		20		25	ns	5, 7
t_{CAA}	Access Time from Column Address		30		35		45		50	ns	6, 7
t_{CPA}	Access Time from $\overline{CAS}$ Precharge		40		45		45		55	ns	7, 14
t_{CLZ}	Output Low Impedance Time from $\overline{CAS}$ Low	0	20	0	20	5		5		ns	7
t_{OFF}	Output Disable Time after $\overline{CAS}$ High	0	20	0	20	0	20	0	30	ns	
t_{REF}	Refresh Cycle Time		8		8		8		8	ms	
t_T	Transition Time	3	50	3	50	3	50	3	50	ns	
t_{RP}	$\overline{RAS}$ High Pulse Width	55		60		70		80		ns	
t_{CRP}	$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	10		10		10		10		ns	
t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	20	40	20	50	25	60	25	75	ns	9, 10
t_{CPN}	$\overline{CAS}$ High Pulse Width	35		35		35		35		ns	
t_{RAD}	Column Address Delay Time from $\overline{RAS}$ Low	15	30	15	35	20	40	20	50	ns	11
t_{ASR}	Row Address Setup Time before $\overline{RAS}$ Low	0		0		0		0		ns	
t_{ASC}	Column Address Time before $\overline{CAS}$ Low	0	20	0	20	0	20	0	20	ns	
t_{RAH}	Row Address Hold Time after $\overline{RAS}$ LOW	15		15		15		15		ns	
t_{CAH}	Column Address Hold Time after $\overline{CAS}$ Low or $\overline{W}$ Low	20		20		20		20		ns	

ตาราง 4.1 ชื่อพารามิเตอร์และระยะเวลาของ Dynamic Ram 21019

READ CYCLE



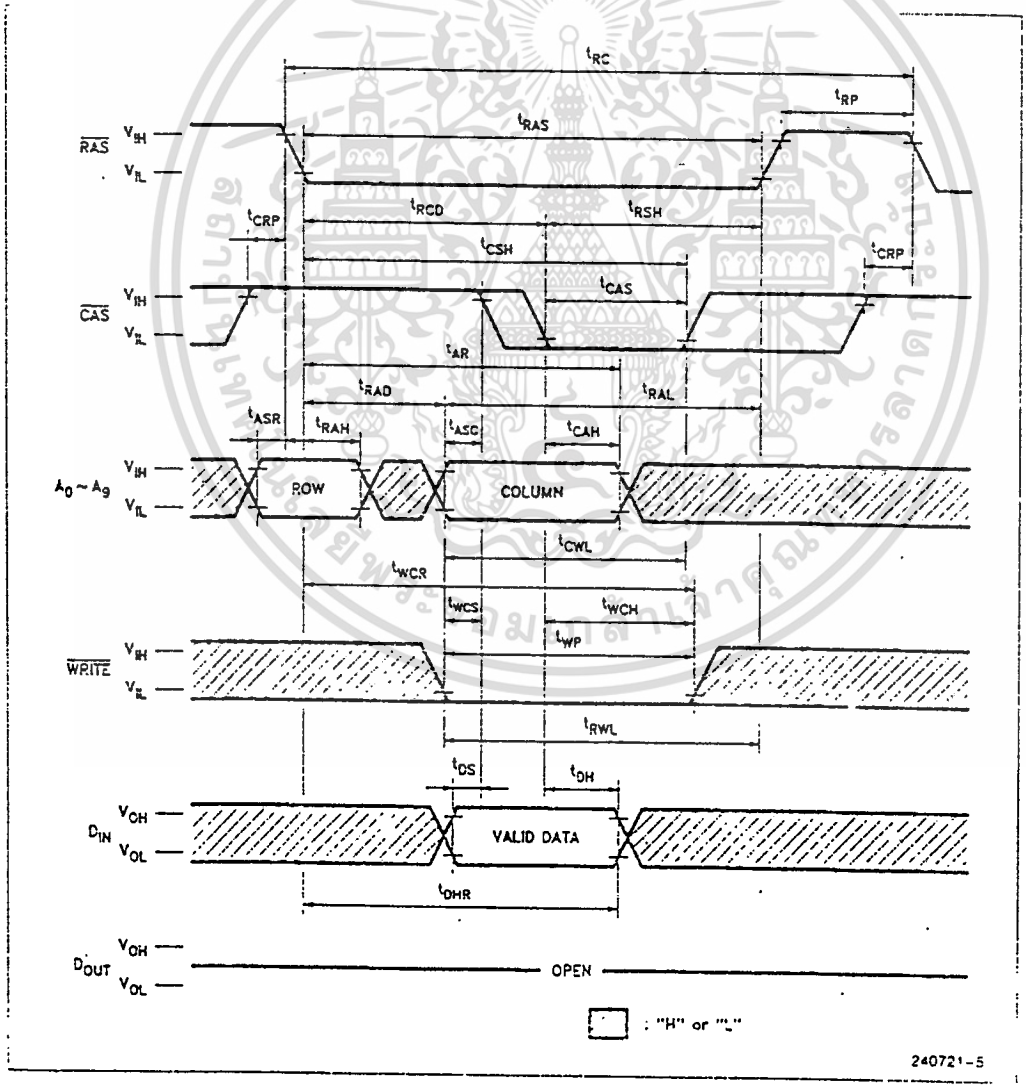
รูปที่ 4.2 READ CYCLE TIMING

4.4.2 ช่วงเวลาในการเขียน

ช่วงเวลาของการเขียนข้อมูลเข้าไปยัง dynamic Ram มีที่น่าสังเกตคือมี 2 ลักษณะ ดังนี้

1. early write cycle timing (รูปที่ 4.3)
2. write cycle timing

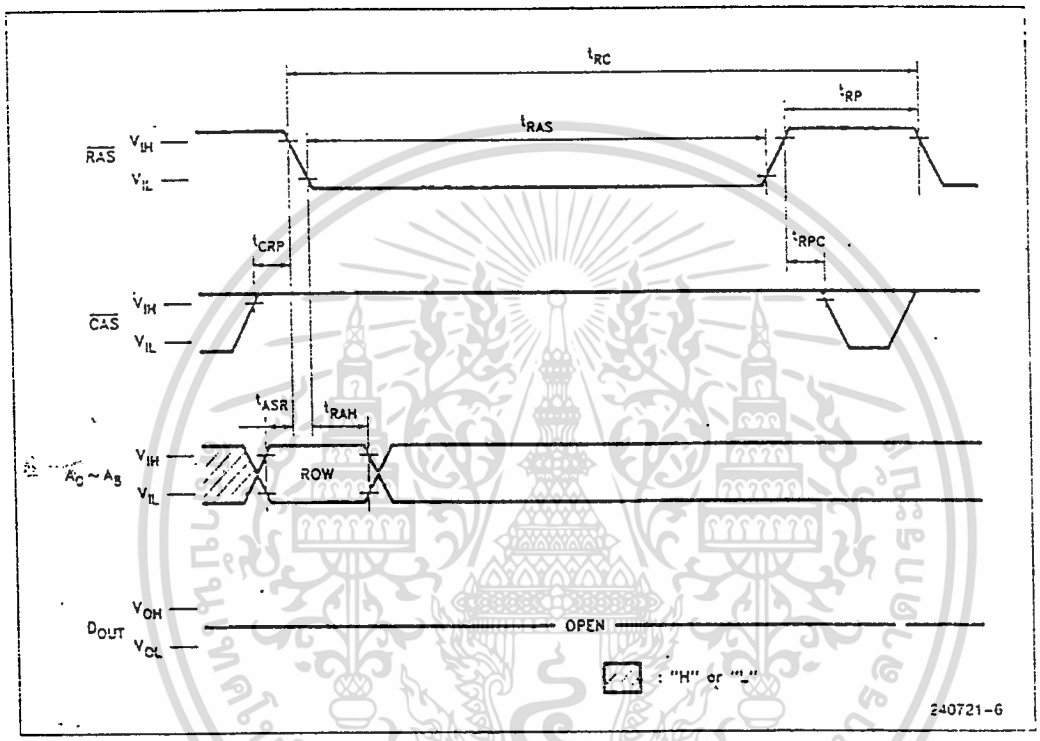
เมื่อพิจารณาในลักษณะที่ 1 จากรูปที่ 4.3 จะเห็นว่า สัญญาณเขียน ($\overline{W}$) จะมาก่อนสัญญาณ $\overline{CAS}$ เอาท์พุทของ Dynamic Ram (D_{OUT}) เป็น hi-impedance ส่วนในลักษณะที่ 2 สัญญาณเขียนมาทีหลังสัญญาณ $\overline{CAS}$ ในที่นี้จะกล่าวถึงการทำงานในลักษณะที่ 1 คือ early write cycle เท่านั้น เมื่อพิจารณาเปรียบเทียบกับช่วงเวลาการอ่านในรูปที่ 4.2 จะเห็นว่า มีส่วนเหมือนกันกับช่วงเวลาของการอ่านตรงที่ W เป็นลอจิก "1" เมื่อยังไม่มี CAS ดังนั้นค่าในหน่วยความจำจะออกมาที่ D_{OUT} ด้วย เป็น NOT VALID ในการออกแบบจึงควรหลีกเลี่ยงการเกิดข้อมูลในลักษณะนี้เกิดขึ้น ซึ่งบางครั้งจะเห็นว่า มีการนำเอา tri-state มากั้นที่ output Q เสียก่อน แต่การออกแบบในที่นี้จะใช้ตามลักษณะที่ 1 คือ early write cycle ทั้งนี้เพื่อเป็นการประหยัด tri-state ได้อีก 1 ตัว



รูปที่ 4.3 early write cycle timing

4.4.3 ช่วงเวลาของการรีเฟรช

ในรูปที่ 4.4 แสดงแผนผังเวลาของการรีเฟรช Dynamic Ram ซึ่งวิธีการรีเฟรชข้อมูลในหน่วยความจำแบบ Dynamic Ram ทำได้โดยการป้อนแอดเดรสที่จะทำการรีเฟรชเข้าไปยัง A0-A9 จากนั้นให้ $\overline{RAS}$ (โดยที่ไม่ต้องให้ $\overline{CAS}$) ซึ่งก็หมายความว่าการรีเฟรชข้อมูลของ Dynamic Ram จะรีเฟรชที่ละโรวมี 1024 คอลัมน์ จากนั้นจึงเปลี่ยนแอดเดรสเพื่อทำการรีเฟรชต่อไปจนครบทั้ง 1024 โรว ซึ่งเรียกว่าครบรอบของการรีเฟรช รอบของการรีเฟรช Dynamic Ram 21019 นี้จะต้องใช้เวลาไม่เกิน 8 ms

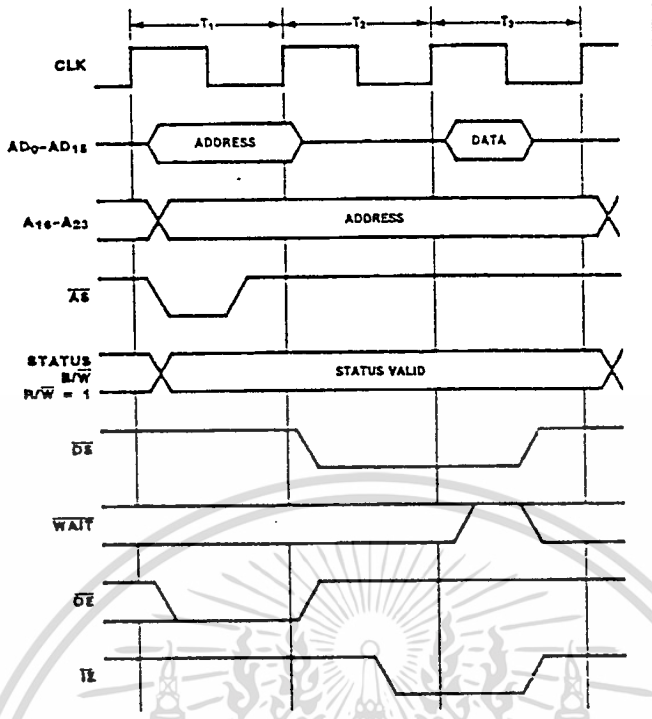


รูปที่ 4.4 ช่วงเวลาของการรีเฟรช (only refresh timing)

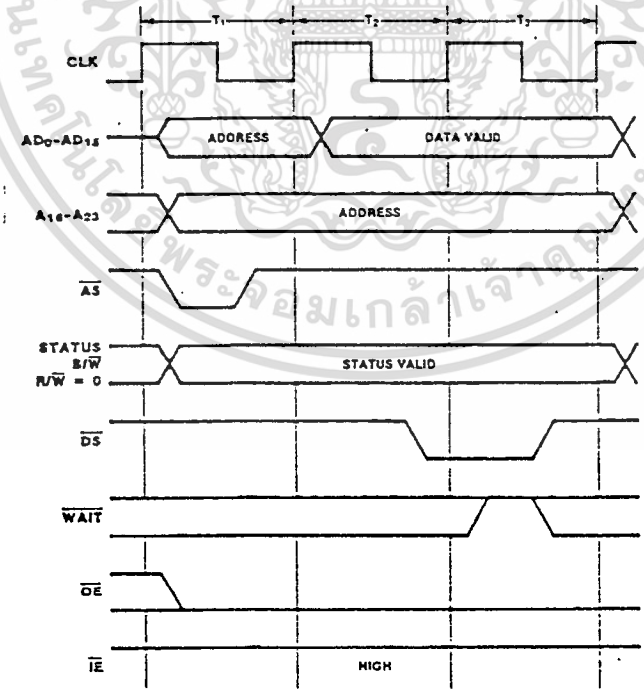
4.5 พิจารณาช่วงเวลาต่าง ๆของ Z280 CPU

4.5.1 ช่วงเวลาการอ่านและเขียนข้อมูลของ CPU

จากรูปที่ 4.5 และ 4.6 จะเห็นได้ว่าเมื่อ CPU อ่านหรือเขียนข้อมูลทั้งสัญญาณ status และ สัญญาณ R/W จะเกิดขึ้นพร้อมกันแต่ในระบบ เราสร้างสัญญาณ $\overline{MEMRD}$, $\overline{MEMWR}$ จากการประกอบกันของสัญญาณ $\overline{MREQ}$, R/W และ $\overline{DS}$ ซึ่งเมื่อเปรียบเทียบกับรูปที่ 4.3 (early write cycle) ก็หมายความว่าจะต้องหน่วงให้ $\overline{CAS}$ ที่จะเกิดขึ้น ต้องเกิดที่หลัง $\overline{MEMWR}$ จากระบบอีกด้วย



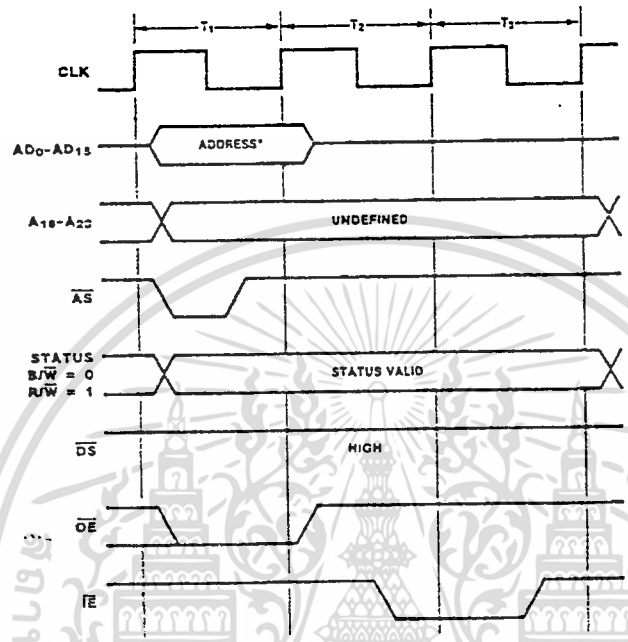
รูปที่ 4.5 Memory read timing



รูปที่ 4.6 Memory write timing

4.5.2 ช่วงเวลาการส่งสัญญาณรีเฟรชจาก CPU

เป็นที่น่าสังเกตว่าในระบบ Z280 CPU จะมีรีเฟรชแอดเดรสขนาด 10 บิท ออกมาที่ A0-A9 แต่การรีเฟรช Dynamic Ram 21019 จะต้องใช้รีเฟรชแอดเดรสขนาด 11 บิท ตัวนั้นจึงต้องสร้างรีเฟรชแอดเดรส A10 เพิ่มขึ้นมา จากที่ได้กล่าวในตอนต้นว่า รอบของการรีเฟรช 21019 จะต้องใช้เวลาไม่เกิน 8 ms ดังนั้นในการรีเฟรชใน 1 โรวจะต้องใช้เวลาไม่เกิน 8ms / 1024 = 7.81 μs



รูปที่ 4.7 Memory refresh timing

ข้อพิจารณาระหว่างช่วงเวลาต่างๆของ CPU และ Dynamic Ram

1. ทุกครั้งที่มีการเฟทช์คำสั่งจะมีสัญญาณรีเฟรชออกมาจาก CPU เพื่อทำการรีเฟรชหน่วยความจำ ในช่วงเวลารีเฟรชนี้ 21019 ต้องการเพียงสัญญาณ $\overline{RAS}$ และระบบจะส่งสัญญาณ RFH, $\overline{MEMRD}$ และ $\overline{MEMWR}$ ออกมา
2. ทุกครั้งที่มีการอ่านหรือเขียนข้อมูล 21019 ต้องการ W, $\overline{RAS}$ และ $\overline{CAS}$ ส่วนในระบบมีสัญญาณ $\overline{MEMRD}$, $\overline{MEMWR}$ ออกมา

จากข้อพิจารณา 2 ข้อข้างต้นเห็นว่าจะต้อง มัลติเพล็กซ์แอดเดรสระหว่า A1-A10 กับ A11-A20 ของ CPU สัญญาณมัลติเพล็กซ์นี้จะมีก็ต่อเมื่อเกิดสัญญาณ $\overline{RAS}$ แล้วเท่านั้น จากรูปสามารถเขียนเป็นวงจรโดยใช้ IC Multiplex เบอร์ 74LS257

เนื่องจากว่า Z280 CPU ส่งรีเฟรชแอดเดรสออกมาเพียง 10 บิท ดังนั้นจึงต้องสร้างรีเฟรชแอดเดรสต่ออีก 1 เส้นเป็นบิทที่ 11 (A10) วิธีการก็คือสร้าง A10 โดยการที่นับสัญญาณจาก RFH โดยการเพิ่ม counter และ tri-state

4.6 การสร้างสัญญาณ MUX, RAS และ CAS

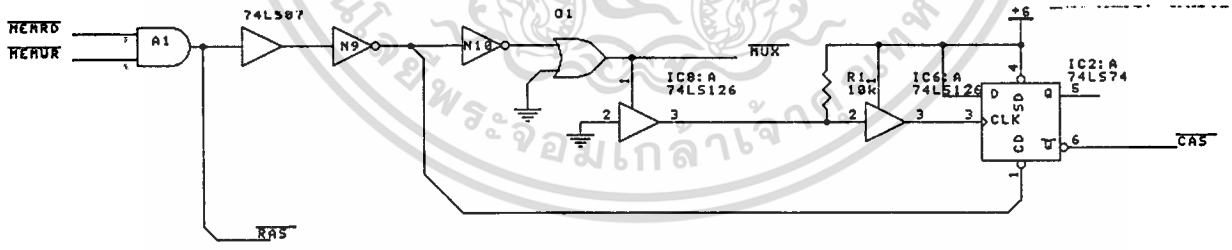
1. RAS

จะเห็นว่าทุกครั้งที่มีการติดต่อกับ Dynamic Ram จะต้องให้มีสัญญาณ $\overline{RAS}$ เกิดขึ้นเสมอ ไม่ว่าจะเป็นการอ่าน การเขียน หรือการรีเฟรช และเมื่อพิจารณาที่ระบบ ทุกครั้งที่มีการติดต่อกับ Dynamic Ram ก็จะมี $\overline{MEMRD}$, $\overline{MEMWR}$ ทุกครั้งเช่นกัน ดังนั้นจึงสามารถใช้ $\overline{MEMRD}$ และ $\overline{MEMWR}$ จากระบบมา AND กันเป็นสัญญาณ $\overline{RAS}$ ของ Dynamic Ram ได้โดยตรง

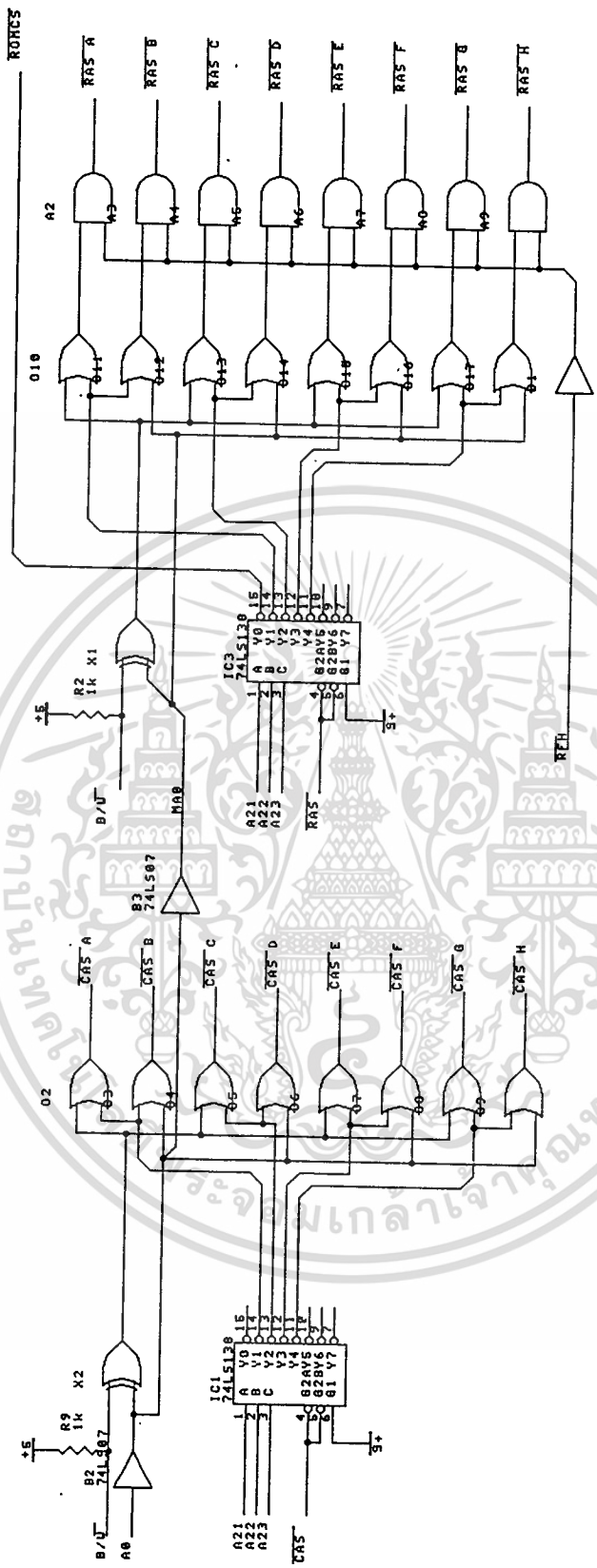
2. MUX

โดยปกติแล้วจะเป็นลอจิก "1" อยู่เสมอจนกว่าจะเกิดสัญญาณ $\overline{RAS}$ แล้วซึ่งหมายความว่า เมื่อต้องการคอลัมน์แอดเดรส (A11-A20) สัญญาณ $\overline{MUX}$ จะต้องเป็น "0" ทั้งนี้เพื่อให้ 74LS257 เลือกคอลัมน์แอดเดรส (A11-A20) เพื่อให้คอลัมน์แอดเดรสไปรอที่ขา A0-A9 ของ Dynamic Ram จากนั้นจึงให้ $\overline{CAS}$ ดังนั้น $\overline{MUX}$ และ $\overline{CAS}$ สามารถสร้างจากสัญญาณ $\overline{RAS}$ ที่หน่วงเวลาออกไปนั้นเอง ซึ่งจากวงจรในทางปฏิบัติจะได้ช่วงเวลาจาก $\overline{RAS}$ ถึง $\overline{MUX}$ ประมาณ 40 ns และเมื่อหน่วงสัญญาณ $\overline{MUX}$ เพื่อให้ได้ $\overline{CAS}$ โดยป้อนเข้าที่ 74LS74 (ทั้งนี้เพื่อต้องการเซตสัญญาณที่เข้าให้ได้สัญญาณ $\overline{CAS}$ ที่ลอจิกขาขึ้นใกล้เคียงกับสัญญาณ $\overline{RAS}$ มากที่สุด) ระยะเวลาในการหน่วงสัญญาณในช่วงนี้จะทำให้ได้ช่วงเวลาจาก $\overline{MUX}$ ถึง $\overline{CAS}$ ประมาณ 34 ns

สัญญาณรีเฟรชแอดเดรส A10 จะถูกควบคุมโดย $\overline{RFH}$ จาก CPU คือเมื่อมีการรีเฟรช 74LS126 จะทำการเปิดให้รีเฟรชแอดเดรส A10 ผ่านไปและ 74LS126 อีกตัวจะปิดสัญญาณแอดเดรสจากมัลติเพล็กซ์ และเมื่อไม่มีการรีเฟรช 74LS126 จะให้สัญญาณจากมัลติเพล็กซ์ผ่านไปดังเดิม



รูปที่ 4.8 วงจรสร้างสัญญาณ $\overline{RAS}$,$\overline{MUX}$ และ $\overline{CAS}$



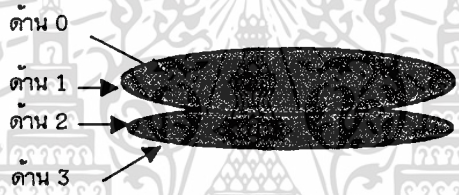
รูปที่ 4.9 วงจรถอดรหัสเลือกช่วงหน่วยความจำ

บทที่ 5

ฮาร์ดดิสต์อินเตอร์เฟส

5.1 ลักษณะของฮาร์ดดิสต์

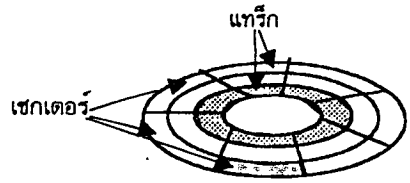
ฮาร์ดดิสต์ประกอบด้วยแผ่นจานโลหะแข็งที่เรียกว่า แพลตเตอร์ (platter) วางซ้อนกัน โดยมีช่องว่างระหว่างแผ่นและมีหัวอ่านและหัวบันทึกข้อมูลที่เป็นแม่เหล็กไฟฟ้าในแต่ละข้างของแต่ละแพลตเตอร์ เช่น ฮาร์ดดิสต์ 340 Mbyte จะมีหัวอ่าน 16 หัวอ่าน หัวอ่าน/บันทึก (disk head) ทำหน้าที่บันทึกและอ่านข้อมูลบนแผ่นดิสต์ด้วยระบบแม่เหล็กไฟฟ้า ซึ่งเป็นวิธีเดียวกันกับการบันทึกภาพเก็บไว้ในม้วนวิดีโอเทป ฮาร์ดดิสต์ของไมโครคอมพิวเตอร์บางครั้งเรียกว่า Fixed drive หรือ Winchester drive



รูปที่ 5.1 ลักษณะของฮาร์ดดิสต์

5.2 ไซลินเดอร์ หัวอ่าน/บันทึก แทร็ก และเซกเตอร์

แผ่นดิสต์แบ่งพื้นที่แยกออกจากศูนย์กลางเป็นส่วนๆ เรียกว่า เซกเตอร์ (sector) แต่ละเซกเตอร์มีความจุ 512 ไบท์ หลายๆ เซกเตอร์รวมกันเข้าเป็นแทร็ก (track) แต่เนื่องจากฮาร์ดดิสต์มีแพลตเตอร์วางซ้อนกันหลายแผ่น และแต่ละแพลตเตอร์ก็มี 2 ด้าน ดังนั้นจึงต้องใช้หัวอ่าน/บันทึก ในแต่ละด้าน ทำให้ต้องมีหัวอ่าน/บันทึก หลายหัว และเมื่อซ้อนแพลตเตอร์หลายๆแผ่นส่วนของแทร็กที่ตรงกันของแต่ละแพลตเตอร์ สามารถเรียกว่า ไซลินเดอร์ (Cylinder)

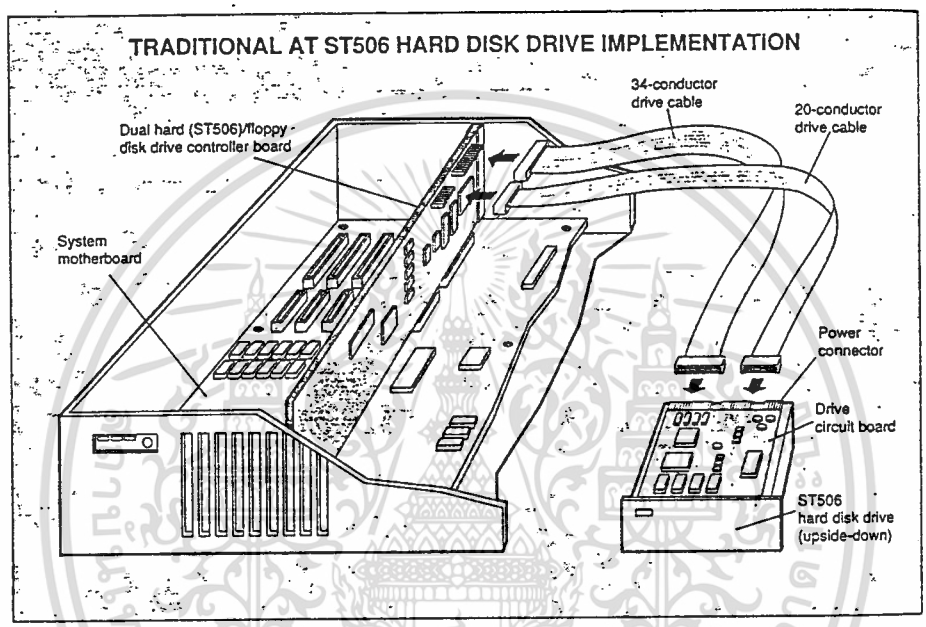


รูปที่ 5.2 การจัดฮาร์ดดิสต์เป็นแทร็กและเซกเตอร์

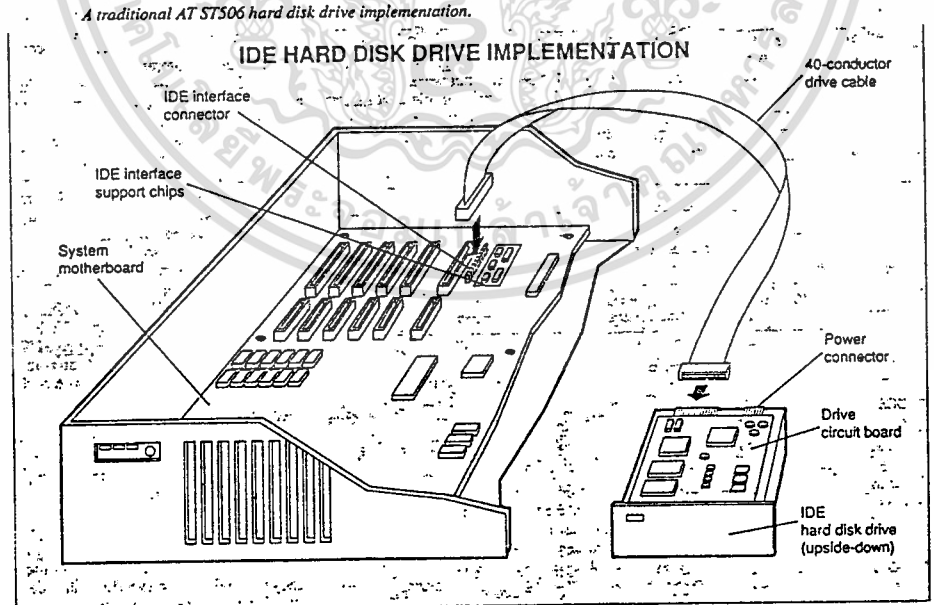
5.3 การเชื่อมต่อฮาร์ดดิสต์แบบ IDE

IDE ย่อมาจากคำว่า Integrated Drive Electronics ซึ่งหมายความว่า IDE Drive รวมส่วนควบคุมฮาร์ดดิสต์ทั้งหมดลงในวงจรของฮาร์ดดิสต์แล้ว ดังนั้นจึงไม่ต้องมีส่วนควบคุม (Hard disk drive controller) ภายนอกอีก การเชื่อมต่อฮาร์ดดิสต์แบบ IDE กับระบบภายนอกจึงต้องการอุปกรณ์เพิ่มเติมไม่มาก

IDE มีการพัฒนามาจากมาตรฐาน AT ST506 ซึ่งเป็นมาตรฐานของฮาร์ดดิสต์ในอดีต ซึ่งมาตรฐาน AT ST506 นั้นจำเป็นต้องมีส่วนควบคุมภายนอกจำนวนมาก จากรูปจะเห็นได้ว่าต้องมีบอร์ดที่ควบคุมฮาร์ดดิสต์แบบ AT ST506 เป็นพิเศษ แต่เมื่อเปรียบเทียบกับฮาร์ดดิสต์ที่ใช้มาตรฐาน IDE แล้วจะเห็นได้ว่าส่วนที่เชื่อมต่อกับฮาร์ดดิสต์จะมีขนาดเล็กลงมาก



A traditional AT ST506 hard disk drive implementation.



With IDE, the hard disk drive controller moves onto the hard disk, freeing an expansion slot. Simple, inexpensive circuitry on the motherboard is all that's needed to accommodate an IDE drive.

รูปที่ 5.3 การเชื่อมต่อฮาร์ดดิสต์แบบ ST506 กับ การเชื่อมต่อแบบ IDE

คอนเนคเตอร์เชื่อมต่อ IDE มี 40 ขาและใช้ร่วมกับสายสัญญาณแบบ 40 เส้น โดยคอนเนคเตอร์ขาที่ 20 จะถูกหักทิ้งเพื่อเป็นจุดสังเกต และสัญญาณจากฮาร์ดดิสก์ IDE จะมีขนาดเท่ากับมาตรฐาน TTL คอนเนคเตอร์แต่ละขาของฮาร์ดดิสก์มีดังต่อไปนี้

COMPARISON OF IDE AND AT I/O CHANNEL SIGNAL CONNECTIONS

Table 1: *With the exception of the chip selects and drive intercommunication signals, IDE signals connect directly to AT I/O channel signals (N/A = not applicable).*

IDE signal name	IDE connector pin assignment	AT I/O channel signal and pin assignment		Signal direction	Optional?
RESET –	1	RESET DRV (inv[1])	B2	To drive	No
Ground	2	Ground	N/A	N/A	No
DD7	3	SD7	A2	Bidirectional	No
DD8	4	SD8	C11	Bidirectional	No
DD6	5	SD6	A3	Bidirectional	No
DD9	6	SD9	C12	Bidirectional	No
DD5	7	SD5	A4	Bidirectional	No
DD10	8	SD10	C13	Bidirectional	No
DD4	9	SD4	A5	Bidirectional	No
DD11	10	SD11	C14	Bidirectional	No
DD3	11	SD3	A6	Bidirectional	No
DD12	12	SD12	C15	Bidirectional	No
DD2	13	SD2	A7	Bidirectional	No
DD13	14	SD13	C15	Bidirectional	No
DD1	15	SD1	A8	Bidirectional	No
DD14	16	SD14	C17	Bidirectional	No
DD0	17	SD0	A9	Bidirectional	No
DD15	18	SD15	C18	Bidirectional	No
Ground	19	Ground	N/A	N/A	No
(keypin)	20	N/A	N/A	N/A	No
DMARQ	21	DRQx	N/A	From drive	Yes
Ground	22	Ground	N/A	N/A	No
DIOW –	23	– IOW	B13	To drive	No
Ground	24	Ground	N/A	N/A	No
DIOR –	25	– IOR	B14	To drive	No
Ground	26	Ground	N/A	N/A	No
IORDY	27	IOCHRDY	A10	From drive	Yes
SPSYNC	28	N/A	N/A	Interdrive	No
DMACK –	29	– DACKx	N/A	To drive	Yes
Ground	30	Ground	N/A	N/A	No
INTRQ	31	IRQ14	D7	From drive	No
IOCS16 –	32	– I/OCS16	D2	From drive	No
DA1	33	SA1	A30	To drive	No
PDIAG –	34	N/A	N/A	Interdrive	No
DA0	35	SA0	A31	To drive	No
DA2	36	SA2	A29	To drive	No
CS1FX –	37	N/A	N/A	To drive	No
CS3FX –	38	N/A	N/A	To drive	No
DASP –	39	N/A	N/A	From drive[2]	No
Ground	40	Ground	N/A	N/A	No

Notes:
[1]The IDE Reset signal polarity is inverted from the AT bus signal.
[2]DASP – is also an interdrive signal.

ตารางที่ 5.1 แสดงการจัดเรียงขาของฮาร์ดดิสก์แบบ IDE

CS1FX- (Drive Chip Select 0)	ใช้สั่งว่าให้ติดต่อกับ Command Block Register
CS3FX- (Drive Chip Select 1)	ใช้สั่งว่าให้ติดต่อกับ Control Block Register
DA0-DA2 (Drive Address Bus)	เป็นแอดเดรสของฮาร์ดดิสต์ใช้ร่วมกับขาข้อมูลเพื่อกำหนดจุดหมายของคำสั่งหรือข้อมูล
DASP- (Drive active/drive 1 present)	ใช้แสดงว่าไดร์ฟแอคทีฟ หรือ บอกว่ามีไดร์ฟ 1 อยู่ด้วย
DD0 - DD15 (Drive Data Bus)	ใช้ส่งและรับข้อมูลและคำสั่งเข้าสู่ฮาร์ดดิสต์
DIOW- (Drive I/O Write)	ใช้สั่งการเขียนข้อมูลหรือคำสั่งสู่ฮาร์ดดิสต์
DIOR- (Drive I/O Read)	ใช้สั่งการอ่านข้อมูลหรือสถานะจากฮาร์ดดิสต์
DMACK- (DMA acknowledge)	ใช้ตอบรับการขอ DMA
DMARQ (DMA request)	ใช้ขอการใช้ DMA
INTRQ (Drive Interrupt)	ใช้อินเตอร์รัพท์ซีพียู
IOCS16- (Drive 16-bit I/O)	ใช้แสดงว่าข้อมูลเป็นแบบ 8 บิทหรือ 16 บิท
IORDY (I/O channel ready)	ใช้เพื่อขยายช่วงเวลาในการเคลื่อนย้ายข้อมูล
PDIAG- (Passed diagnostics)	ใช้เป็นช่องทางให้ไดร์ฟ 1 ติดต่อกับไดร์ฟ 0
RESET-	ใช้ Reset ฮาร์ดดิสต์ เพื่อเริ่มต้นใหม่
SPSYNC : CSEL (Spindle synchronization/cable select)	ใช้เพื่อควบคุมระหว่าง 2 ไดร์ฟ
GROUND	ใช้เป็นกราวนด์ของระบบ

การสั่งงานฮาร์ดดิสต์แบบ IDE ทำได้โดยส่งค่าพารามิเตอร์ต่างๆไปที่รีจิสเตอร์ของฮาร์ดดิสต์ และส่งคำสั่งไปที่ตัวฮาร์ดดิสต์ให้ทำงาน (สำหรับเครื่อง PC จะเป็นแอดเดรส 1F7h) ซึ่งค่าแอดเดรสของรีจีสเตอร์ต่างๆจะเป็นดังตารางต่อไปนี้

HARD DISK DRIVE REGISTER DEFINITIONS

Table 2: *For compatibility with the standard ST506 controller, chip selects are active in the 1F0-to-1F7 and 3F0-to-3F7 I/O-addressing ranges (N/A = not applicable).*

I/O address	Read register	Write register	Hard or floppy?
1F0	Data register	Data register	Hard
1F1	Error register	Write precomp	Hard
1F2	Sector count	Sector count	Hard
1F3	Sector number	Sector number	Hard
1F4	Cylinder low	Cylinder low	Hard
1F5	Cylinder high	Cylinder high	Hard
1F6	Drive/head	Drive/head	Hard
1F7	Status register	Command register	Hard
3F2	N/A	Digital output	Floppy
3F4	Main status	Main status	Floppy
3F5	Diskette data	Diskette data	Floppy
3F6	N/A	Fixed disk	Hard
3F7	Digital input	Diskette control	Hard/floppy*

Notes:
* The digital-input register includes 7 bits for the hard disk and one for the floppy disk.
All I/O addresses are in hexadecimal.

ตาราง 5.2 แสดงรีจีสเตอร์ของฮาร์ดดิสต์

ในส่วนคำสั่งของฮาร์ดดิสต์แบบ IDE นั้นจะแบ่งออกได้เป็น 2 กลุ่ม

1. กลุ่มคำสั่งบังคับ (mandatory) เป็นคำสั่งที่ผู้ผลิตฮาร์ดดิสต์ IDE ต้องทำรองรับคำสั่งเหล่านี้
2. กลุ่มคำสั่งเลือก (optional) เป็นคำสั่งที่อาจจะมีหรือไม่มีในฮาร์ดดิสต์ที่ผลิตก็ได้

คำสั่งของ IDE ทั้งกลุ่มคำสั่งบังคับและคำสั่งเลือก อาจแบ่งออกเป็นระดับในการตอบสนองคำสั่งได้ 3 ระดับคือ

- ระดับ 1 เมื่อไดร์ฟได้รับคำสั่งจะเซตบิต BSY (busy) ใน Status Register ภายใน 400 นาโนวินาที
- ระดับ 2 เมื่อไดร์ฟได้รับคำสั่งจะเซตบิต BSY เตรียมส่วนเซกเตอร์บัฟเฟอร์ให้พร้อมรับการเขียน และเซตบิต DRQ (data request) ใน Status Register ภายใน 700 ไมโครวินาที แล้วจึงเคลียร์บิท BSY
- ระดับ 3 จะเหมือนกับคำสั่งในระดับ 2 แต่จะใช้เวลา 20 มิลลิวินาทีในการเซตบิต DRQ

คำสั่งของฮาร์ดดิสต์ IDE สามารถแสดงได้ดังตารางแสดงคำสั่ง

5.4 ขีดจำกัดของฮาร์ดดิสต์แบบ IDE

จำนวนไซลินเดอร์	=	65536
จำนวนหัวอ่าน/บันทึก	=	16
จำนวนเซกเตอร์ต่อแทร็ก	=	256
จำนวนไบต์ต่อเซกเตอร์	=	512
จำนวนฮาร์ดดิสต์ที่ต่อร่วมกันได้	=	2 ตัว/controller interface

ซึ่งถ้าดูจากขนาดสูงสุดที่เป็นได้ของฮาร์ดดิสต์จะเท่ากับ 128 GByte ซึ่งมีขนาดที่ใหญ่มากและยังสามารถต่อร่วมกับ ฮาร์ดดิสต์ได้ถึง 2 ตัว ทำให้ไม่มีปัญหาในด้านขนาดความจุของฮาร์ดดิสต์

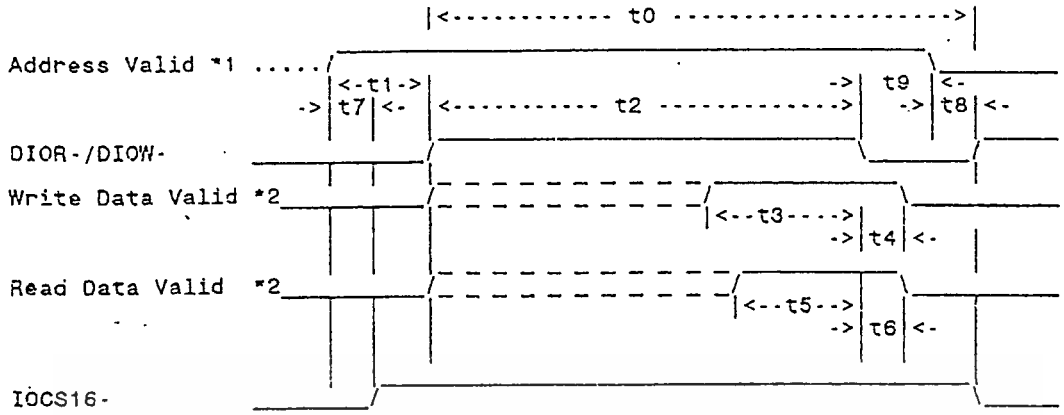
IDE COMMANDS

Table 3: *Mandatory commands are those supported by the original IBM AT ST506 controller. When BIOS support for optional commands, such as* `Read Multiple` *and* `Write Multiple`*, materializes, drive vendors will be able to support IDE's advanced capabilities. All command codes are in hexadecimal; N/A = not applicable.*

Command	Class	Command code	Optional?
Check Power Mode	1	98 E5	Yes
Execute Drive Diagnostic	1	90	No
Format Track	2	50	No
Identify Drive	1	EC	Yes
Idle	1	97 E3	Yes
Idle Immediate	1	95 E1	Yes
Initialize Drive Parameters	1	91	No
Recalibrate	1	1x	No
Read Buffer	1	E4	Yes
Read DMA (with retry)	1	C8	Yes
Read DMA (without retry)	1	C9	Yes
Read Multiple	1	C4	Yes
Read Sector(s) (with retry)	1	20	No
Read Sector(s) (without retry)	1	21	No
Read Long (with retry)	1	22	No
Read Long (without retry)	1	23	No
Read Verify Sector(s) (with retry)	1	40	No
Read Verify Sector(s) (without retry)	1	41	No
Seek	1	7x	No
Set Features	1	EF	Yes
Set Multiple Mode	1	C6	Yes
Set Sleep Mode	1	99 E6	Yes
Standby	1	96 E2	Yes
Standby Immediate	1	94 E0	Yes
Write Buffer	2	E8	Yes
Write DMA (with retry)	3	CA	Yes
Write DMA (without retry)	3	CB	Yes
Write Multiple	3	C5	Yes
Write Same	3	E9	Yes
Write Sector(s) (with retry)	2	30	No
Write Sector(s) (without retry)	2	31	No
Write Sector(s) (with retry)	2	32	No
Write Sector(s) (without retry)	2	33	No
Write Verify	3	3C	Yes
Vendor unique	N/A	9A	N/A
Vendor unique	N/A	C0-C3	N/A
Vendor unique	N/A	8x	N/A
Vendor unique	N/A	F5-FF	N/A
Reserved: all remaining codes			

ตารางที่ 5.3 แสดงคำสั่งที่ใช้ในการติดต่อกับฮาร์ดดิสก์

5.5 วงจรเชื่อมต่อกับฮาร์ดดิสต์แบบ IDE



*1 Drive Address consists of signals CS1FX-, CS3FX- and DA2-0
*2 Data consists of DD0-15 (16-bit) or DD0-7 (8-bit)

	PIO timing parameters		Mode 0 nsec	Mode 1 nsec	Mode 2 nsec
t0	Cycle time	(min)	600	383	240
t1	Address valid to DIOR-/DIOW- setup	(min)	70	50	30
t2	DIOR-/DIOW- 16-bit	(min)	165	125	100
	Pulse width 8-bit	(min)	290	290	290
t3	DIOW- data setup	(min)	60	45	30
t4	DIOW- data hold	(min)	30	20	15
t5	DIOR- data setup	(min)	50	35	20
t6	DIOR- data hold	(min)	5	5	5
t7	Addr valid to IOCS16- assertion	(max)	90	50	40
t8	Addr valid to IOCS16- negation	(max)	60	45	30
t9	DIOR-/DIOW- to address valid hold	(min)	20	15	10

รูปที่ 5.4 สัญญาณแสดงการนำข้อมูลเข้าและออกจากฮาร์ดดิสต์

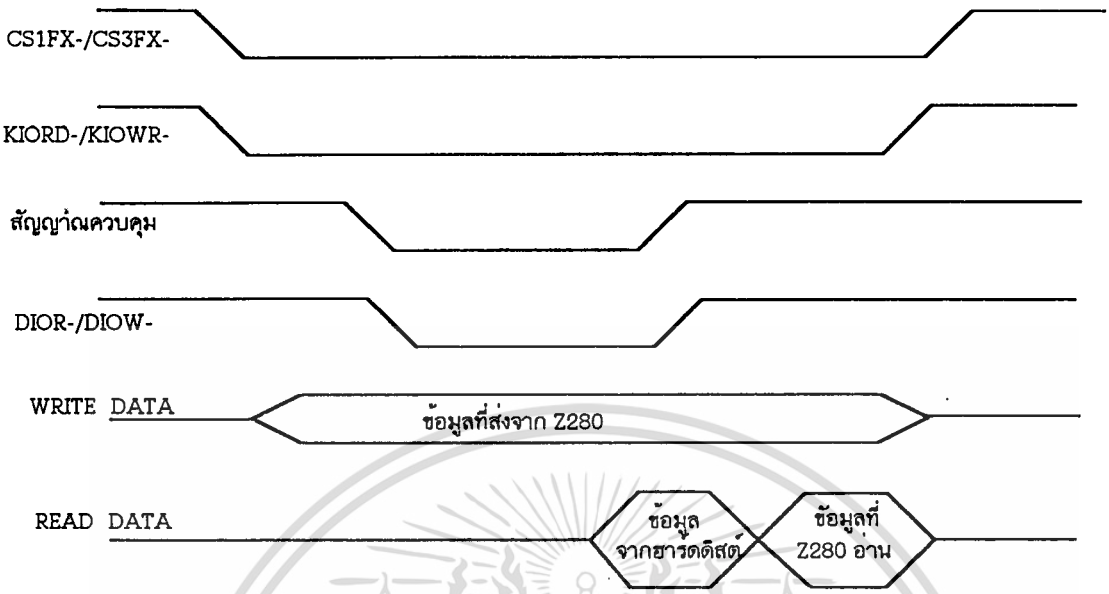
จาก Timing Diagram จะเห็นได้ว่าการติดต่อกับ ฮาร์ดดิสต์แบบ IDE จำเป็นต้องทำตามช่วงเวลาที่กำหนดเท่านั้น ดังนั้น ไมโครโปรเซสเซอร์ Z280 จึงต้องมีวงจรภายนอกมาช่วยทำตามช่วงเวลาของฮาร์ดดิสต์ IDE ได้

ถ้าดูจาก Timing Diagram จะเห็นว่า เราต้องส่ง ค่าแอดเดรสที่ถูกต้องและสัญญาณ CS1FX- (CX3FX-) ไปก่อน หลังจากนั้นเมื่อเวลาผ่านไปอย่างน้อย 70 nsec (คิดจาก mode 0) จึงจะส่งสัญญาณอ่านหรือเขียนตามไป และซีพียูจะอ่านหรือเขียนข้อมูลได้ หลังส่งสัญญาณสิ้นสุดการอ่านหรือการเขียน ซึ่งรูปแบบการส่งสัญญาณให้ฮาร์ดดิสต์แบบนี้จะแตกต่างจากไมโครโปรเซสเซอร์ Z280 ที่จะส่งสัญญาณ CS และ สัญญาณ RD/WR ไปพร้อมกัน และไมโครโปรเซสเซอร์ Z280 จะอ่านหรือเขียนข้อมูลก่อนที่จะสิ้นสุดสัญญาณ CS (ซึ่งควบคุมโดย $\overline{DS}$) ทำให้แตกต่างจากวิธีของฮาร์ดดิสต์แบบ IDE (ดูรูปสัญญาณของ Z280 ประกอบ)

ดังนั้นเพื่อเป็นการแก้ปัญหาจึงต้องออกแบบวงจรเพิ่มเติมให้สัญญาณควบคุมฮาร์ดดิสต์ทำงานในช่วง CS ของไมโครโปรเซสเซอร์ Z280 โดยจะส่งสัญญาณ CS1FX- และ CS3FX- ที่สร้างโดยไมโครโปรเซสเซอร์ Z280 ไปให้ฮาร์ดดิสต์ก่อน โดยผ่านบัฟเฟอร์ 74LS244 และนำสัญญาณ CS1FX- และ CS3FX- ไปผ่านวงจร AND เพื่อให้เลือกส่งสัญญาณ CLK 5 MHz ส่งไปยัง D - ฟลิปฟลอป เพื่อสร้างสัญญาณควบคุม การอ่านหรือการเขียน ในช่วงเวลาที่กำหนด ซึ่งจะเริ่มทำงานที่ขอบขาขึ้นของสัญญาณ CLK และสัญญาณ CS1FX หรือ CS3FX เป็น LOW แล้ว ทำให้ได้ช่วงเวลาช่วงหนึ่ง (คือช่วง t_1 ของ IDE) และสัญญาณนี้จะนำไปสร้างสัญญาณ DIOR- หรือ DIOW- ต่อไป

หลังจากที่สร้างสัญญาณควบคุม การอ่านหรือการเขียน เรียบร้อยแล้ว ก็ต้องให้สัญญาณนี้หยุดลงก่อนสัญญาณ CS1FX- หรือ CS3FX- จะสิ้นสุดลง เลยต้องให้ D-ฟลิปฟลอป นับไป 2 คล็อก แล้วส่งสัญญาณไปเคลียร์สัญญาณควบคุม การอ่านหรือการเขียน ไม่ให้

ทำงาน แต่สัญญาณที่มาเคลียร์นี้จะค้างตลอดจนกว่าสัญญาณ CS1FX หรือ CS3FX จะไม่แอคทีฟแล้วกลับเป็น High อีกครั้ง

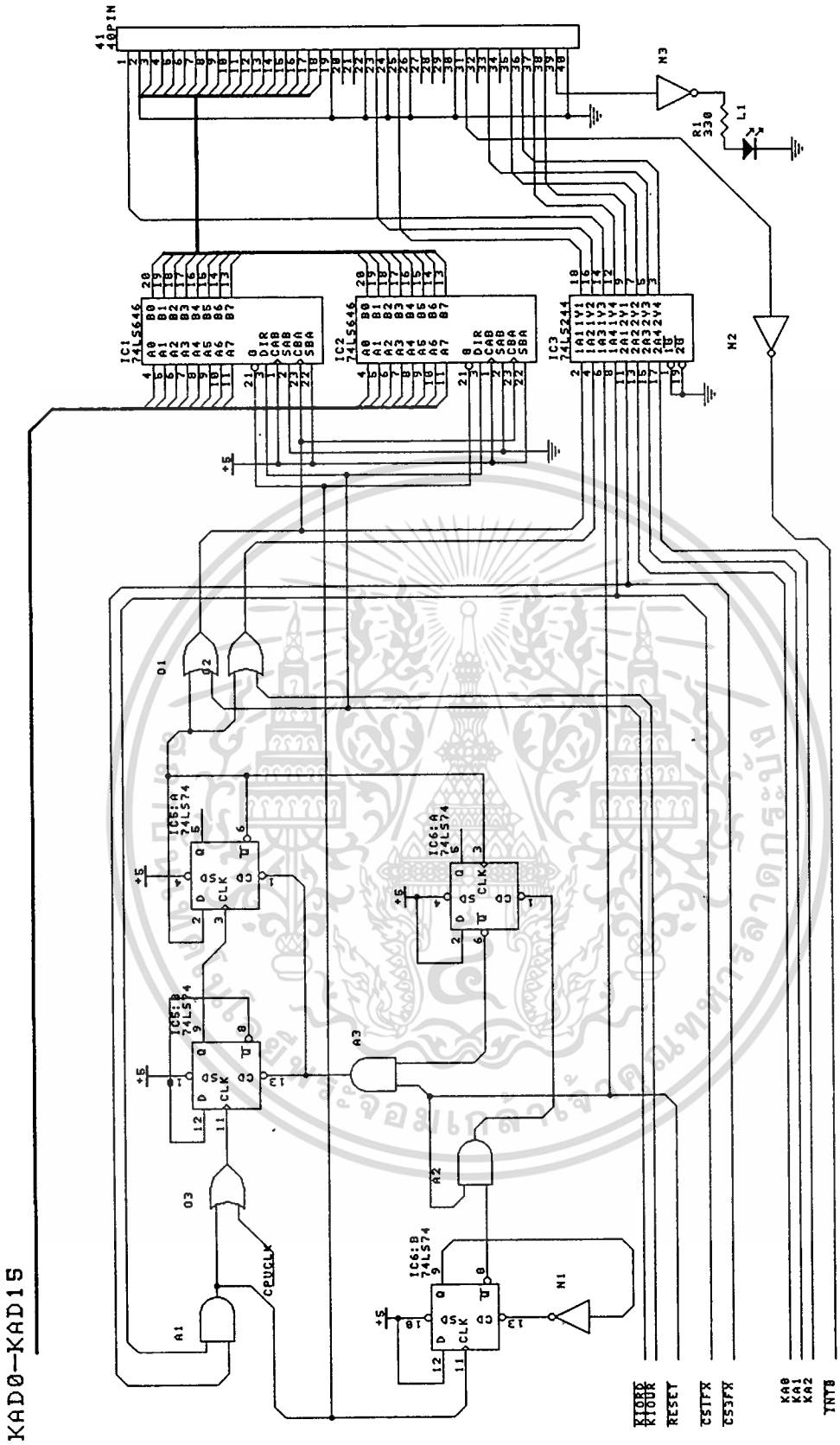


รูปที่ 5.5 สัญญาณที่วงจรสร้างให้ฮาร์ดดิสต์

แม้ว่าตอนนี้จะสร้างสัญญาณควบคุมต่างๆ ได้แล้วก็ตาม แต่ก็ยังมีปัญหาการอ่านหรือเขียนข้อมูลของฮาร์ดดิสต์ กับไมโครโปรเซสเซอร์ Z280 คนละช่วงเวลาอยู่ ดังนั้นจึงต้องใช้ IC 74LS646 ซึ่งเป็นบัฟเฟอร์ และสามารถใช้เก็บข้อมูลชั่วคราวได้ เช่น ในโครงงานนี้จะต่อให้ IC 74LS646 สามารถเก็บข้อมูลไว้ใน IC เมื่อสัญญาณ DIOR ที่ส่งให้ฮาร์ดดิสต์เลิกแอคทีฟ (เปลี่ยนจาก Low เป็น High) และหลังจากนั้นเมื่อไมโครโปรเซสเซอร์ Z280 ต้องการอ่านข้อมูลจึงมาอ่านจาก IC 74LS646 ในภายหลัง

แต่ข้อมูลที่เขียนจากไมโครโปรเซสเซอร์ Z280 ไปยังฮาร์ดดิสต์ไม่ต้องเก็บข้อมูลไว้ใน IC 74LS646 ก่อน เพราะว่าช่วงเวลาการส่งสัญญาณข้อมูลของไมโครโปรเซสเซอร์ Z280 กว้างกว่า ซึ่งทำให้เพียงพอต่อการใช้งานของฮาร์ดดิสต์แบบ IDE แล้ว

ในวงจรสัญญาณจากขา DASP- จะผ่าน NOT เกทเพื่อเป็นบัฟเฟอร์ไปขับ LED เมื่อมีการใช้งานฮาร์ดดิสต์



รูปที่ 5.6 วงจรเชื่อมต่อฮาร์ดดิสต์

บทที่ 6

ทฤษฎีการเชื่อมต่อโทรศัพท์ และ การแปลงสัญญาณเสียง

6.1 ความรู้เบื้องต้นเกี่ยวกับโทรศัพท์

เครื่องโทรศัพท์ (Telephone) ประกอบด้วยส่วนต่างๆ ที่สำคัญ คือ เครื่องส่ง (Transmitter), เครื่องรับ (Receiver), กระดิ่ง (Ringer), Hook Switch และหน้าปัดสำหรับหมุนหรือกดหมายเลข (Dial) สำหรับเครื่องส่งและเครื่องรับรวมเรียกว่า ปากพูดหูฟัง (Handset) ซึ่งเป็นอุปกรณ์ที่ใช้สำหรับเปลี่ยนพลังงานเสียงที่เกิดจากการพูดให้เป็นพลังงานไฟฟ้า และเปลี่ยนพลังงานไฟฟ้าที่ได้รับให้เป็นพลังงานเสียงอีกครั้งหนึ่ง โดยเราจะใช้ Transmitter เป็นตัวเปลี่ยนพลังงานเสียงให้เป็นพลังงานไฟฟ้า และใช้ Receiver เป็นตัวเปลี่ยนพลังงานไฟฟ้าที่ได้รับให้กลับเป็นพลังเสียงอีกครั้งหนึ่ง

หน้าที่หลักของโทรศัพท์

1. เครื่องโทรศัพท์จะรับรู้ว่ามีผู้ต้องการใช้โทรศัพท์เมื่อมีการยกหูโทรศัพท์ขึ้น
2. เครื่องโทรศัพท์จะส่งสัญญาณที่เรียกว่า สัญญาณหมุน (dial tone) บอกว่าพร้อมที่จะให้ทำการกดหรือหมุนหมายเลขที่จะติดต่อได้ ก็คือเสียงที่ได้ยินเมื่อเวลายกหู เป็นสัญญาณเสียงที่มีความถี่ 350 Hz กับ 440 Hz มอดูเลตรวมกัน
3. เครื่องโทรศัพท์จะทำหน้าที่ส่งรหัสหมายเลขที่ผู้เรียกต้องการติดต่อด้วยไปยังชุมสายที่ควบคุม
4. เครื่องโทรศัพท์จะส่งสัญญาณบอกผู้เรียกว่า หมายเลขที่ต้องการติดต่อว่างหรือไม่ ถ้าว่างจะส่งสัญญาณกลับ (Ring back) ความถี่ 440 Hz กับ 480 Hz มอดูเลทกันมาโดยจะดัง 2 วินาที แล้วเงียบ 4 วินาที แต่ถ้าไม่ว่างจะส่งสัญญาณความถี่ 480 Hz กับ 620 Hz มอดูเลทกันมา
5. เครื่องโทรศัพท์จะปรับระดับแรงดันอัตโนมัติ กรณีเกิดการเปลี่ยนแปลงแรงดัน
6. สามารถเปลี่ยนรูปพลังงานเสียงเป็นไฟฟ้า และพลังงานไฟฟ้าเป็นเสียง
7. เครื่องโทรศัพท์จะส่งสัญญาณไปยังชุมสายเพื่อแจ้งให้ทราบว่าสิ้นสุดการใช้งานแล้ว ให้ชุมสายเลิกทำการติดต่อกับอีกฝ่ายหนึ่ง

หน้าปัดของเครื่องโทรศัพท์มี 2 แบบ คือ

1. แบบหมุน (Rotating Type)
2. แบบกดปุ่ม (Push Button)

ในโครงงานนี้จะใช้กับโทรศัพท์แบบกดปุ่มได้เท่านั้น โดยโทรศัพท์แบบกดปุ่มจะใช้กรรมวิธีของ Dual Tone Multifrequency (DTMF) ในการเลือกฟังก์ชันในการติดต่อกับ voice mail box โดยทั่วไปจะมี 12 ปุ่มแบ่งเป็น 4 แถว และ 3 คอลัมน์ และเครื่องโทรศัพท์แบบมี 16 ปุ่มโดยเพิ่มอีกหนึ่งคอลัมน์ ดังตาราง

High Freq. group (Hz)	1,209	1,336	1,447	1,663	
Low Freq. group (Hz)					
697	1	2	3	A	R1
770	4	5	6	B	R2
852	7	8	9	C	R3
941	*	0	#	D	R4
	C1	C2	C3	C4	

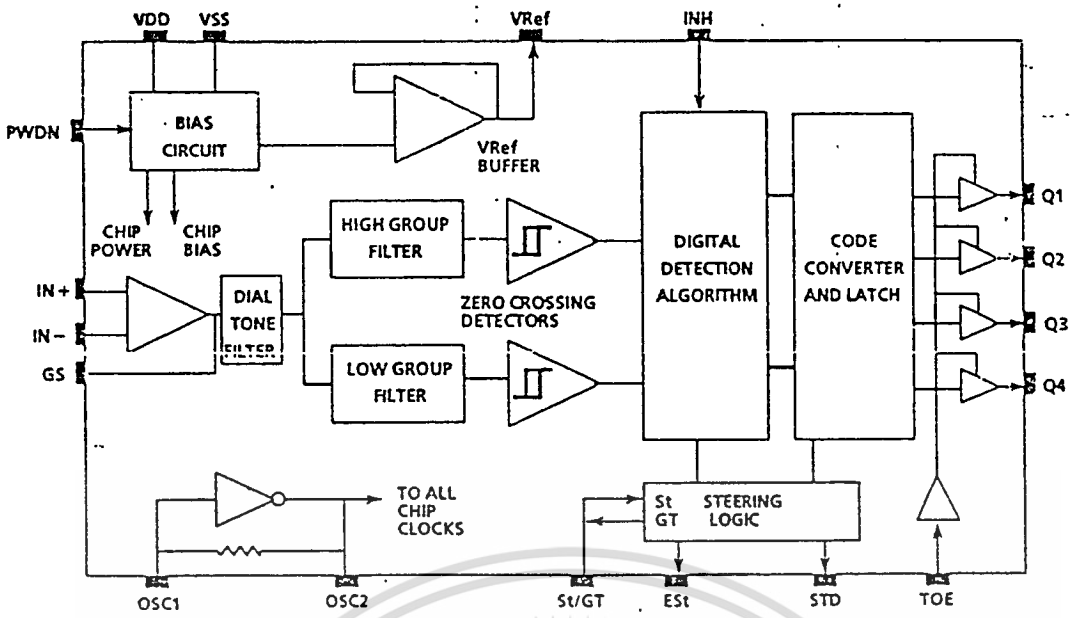
ตาราง 6.1 แสดงความถี่ของ DTMF

ความถี่ที่ใช้ในแต่ละแถวและคอลัมน์จะมีความถี่ต่างกัน ความถี่ของทั้ง 4 แถวเรียกว่าเป็นกลุ่มความถี่ต่ำ (Low Group Frequency) และความถี่ทั้ง 4 คอลัมน์เรียกว่าเป็นกลุ่มความถี่สูง (High Group Frequency) การกดที่หมายเลขใดๆ จะทำให้วงจรอิเลคทรอนิกส์ภายในเครื่องโทรศัพท์ผลิตความถี่ออกมา 2 ความถี่ เช่น กดหมายเลข 5 ความถี่ที่ผลิตออกมาคือ 770 Hz และ 1336 Hz เป็นต้น

ข้อกำหนดต่างๆ ที่จำเป็นเพื่อที่จะไม่ทำให้การถอดรหัสสัญญาณ DTMF เกิดความผิดพลาดขึ้น

1. วงจรจะยังคงสามารถถอดรหัสได้อย่างถูกต้อง ถึงแม้สัญญาณที่รับเข้ามาจะมีความถี่ที่เบี่ยงเบนไปจากค่าที่กำหนดไว้เป็นมาตรฐาน แต่ต้องไม่เกิน $\pm$ 2% และจะไม่ยอมให้สัญญาณที่มีค่าเบี่ยงเบนมากกว่า $\pm$ 3% จากค่ามาตรฐานผ่านวงจรกรองความถี่ไปได้
2. วงจรถอดรหัสจะสามารถถอดรหัสได้ก็ต่อเมื่อได้รับสัญญาณเข้ามามีระยะเวลาอย่างน้อย 40 มิลลิวินาที
3. วงจรถอดรหัสจะทำการถอดรหัสได้ถูกต้อง ก็ต่อเมื่อสัญญาณ DTMF ที่รับเข้ามาในวงจรจะต้องมีช่วงเวลาที่ห่างกับสัญญาณ DTMF ที่รับเข้ามาก่อนหน้านี้เป็นเวลาอย่างน้อย 35 มิลลิวินาที
4. วงจรถอดรหัสจะต้องสามารถถอดรหัสสัญญาณ DTMF ที่มีไดนามิกส์เรนจ์สูงกว่า 27.5 dB ได้ โดยไม่เกิดความผิดพลาด และยังสามารถทำงานได้ในกรณีที่สัญญาณทั้ง 2 ความถี่ที่ประกอบกันขึ้นเป็นสัญญาณ DTMF มีแอมปริจูดแตกต่างกันมากกว่า 6 dB
5. วงจรถอดรหัสยังคงทำงานได้ตลอดเวลา ไม่ว่าขณะนั้นจะปรากฏเสียงพูดหรือมีสัญญาณรบกวนจากภายนอกเข้ามายังวงจรถอดรหัสก็ไม่ทำให้การถอดรหัสผิดพลาด

ในโครงงานนี้จะใช้ไอซีเบอร์ 8870 เป็นตัวถอดรหัส DTMF โดย MT8870 จะทำการแปลงสัญญาณ DTMF เป็นสัญญาณดิจิตอล 4 บิท บล็อกไดอะแกรมของ MT8870 มีรายละเอียดดังนี้



รูปที่ 6.1 บล็อกไดอะแกรมของไอซีเบอร์ MT8870

โครงสร้างของ MT8870

โครงสร้างภายในของ MT8870 ประกอบด้วยวงจรกรองความถี่และวงจรถอดรหัสฟังก์ชันทางดิจิตอล เป็นไอซีสร้างโดยใช้เทคโนโลยี ISO^2-CMOS ในส่วนของวงจรกรองความถี่ใช้เทคนิคของสวิตช์คาปาซิเตอร์ฟิลเตอร์ สำหรับกรองความถี่สูงและต่ำ ส่วนวงจรถอดรหัสใช้เทคนิคการนับทางดิจิตอลเพื่อตรวจจับและถอดรหัสทั้ง 16 ความถี่ออกเป็นเลขฐานสองขนาด 4 บิท และเช็คช่วงเวลาที่สัญญาณเข้ามา ส่วนภาคอินพุทเป็นออปแอมป์ ซึ่งสามารถปรับอัตราขยายได้โดยต่ออุปกรณ์ภายนอกเอาท์พุทเป็นวงจรแลตช์ 3 สถานะ

ฟังก์ชันการทำงานภายใน MT8870 มีส่วนสำคัญ 5 ส่วน

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

ภาคกรองสัญญาณความถี่

ในส่วนนี้จะแยกสัญญาณ DTMF ที่เข้ามาออกเป็น 2 กลุ่มความถี่ คือ ช่วงความถี่สูงและช่วงความถี่ต่ำ โดยใช้วงจรกรองแถบความถี่อันดับ 6 ชนิดสวิตช์คาปาซิเตอร์ (Six order Switched Capacitor band pass filter)

ภาคถอดรหัส

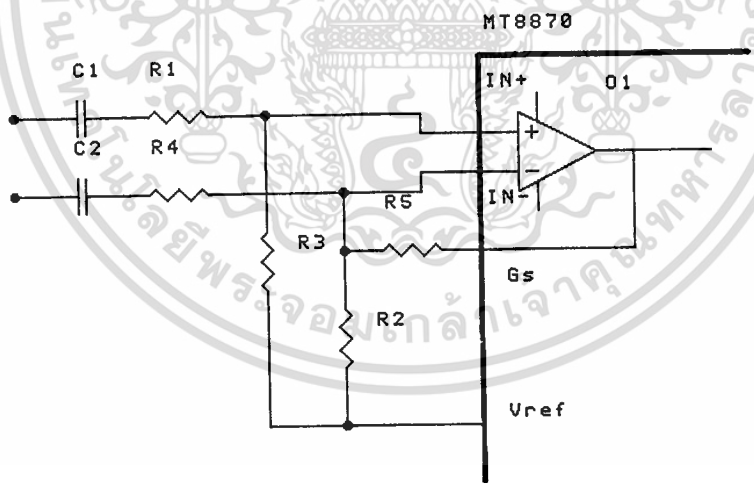
ความถี่ DTMF ที่ถูกกรองเรียบร้อยแล้วจะผ่านเข้าวงจรถอดรหัสความถี่ออกเป็นตัวเลข โดยใช้เทคนิคการนับแบบดิจิตอล และมีการตรวจสอบความถี่ที่เข้ามาว่าเป็นความถี่มาตรฐาน DTMF หรือไม่ เพื่อป้องกันความถี่อื่นเข้ามาผสม เมื่อตรวจสอบว่าความถี่นั้นถูกต้อง สัญญาณที่ขา Est (Early Steering) ก็จะแอคทีฟสำหรับค่าที่ถอดรหัสได้จากความถี่ต่างๆนั้น

ภาคตรวจสอบสัญญาณ

ก่อนที่จะมีการถอดรหัสความถี่ออกไปที่เอาท์พุท จะมีการตรวจสอบช่วงความถี่ที่เข้ามาว่ามีระยะเวลาตามที่กำหนดหรือไม่ โดยสังเกตจากระยะเวลาการกดปุ่มโทรศัพท์ซึ่งต้องกดปุ่มให้มีความถี่ออกมาเป็นช่วงเวลาพอสมควร มิฉะนั้นวงจรส่วนนี้จะไม่รับโดยถือว่าสัญญาณนั้นไม่ถูกต้อง ส่วนช่วงเวลายาวเท่าใดสามารถตั้งได้โดยใช้ RC ต่อภายนอก สัญญาณที่ขา Est จะเป็น high นานใกล้เคียงกับระยะเวลาที่มีความถี่ DTMF เข้ามานานเท่ากับหรือมากกว่าช่วงเวลาที่เราตั้งไว้ จึงจะได้รับการยอมรับว่าสัญญาณความถี่นั้นถูกต้อง หรือพูดได้ว่าเวลาที่เราตั้งไว้โดย RC ก็คือการ์ดไทม์นั่นเอง

ภาคขยายสัญญาณความแตกต่าง

วงจรส่วนอินพุทของ MT 8870 เป็นภาคขยายออปแอมป์ที่สามารถปรับอัตราขยายโดยต่อวงจรภายนอกเพิ่มเข้าไป ดังรูป



รูปที่ 6.2 โครงสร้างวงจรแบบดิฟเฟอเรนเชียล

ได้ $R_3 = \frac{R_2 \times R_5}{R_2 + R_5}$

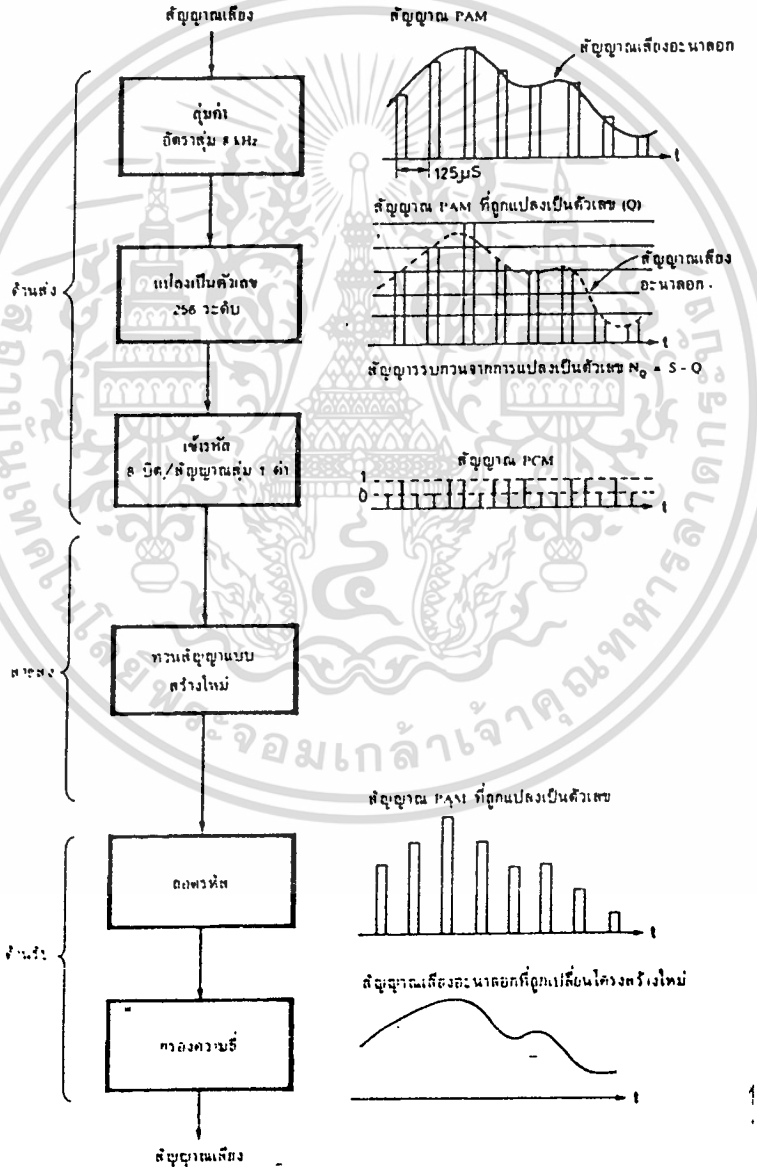
อัตราขยายแรงดัน $(A_{v\,diff}) = \frac{R_5}{R_1}$

อินพุทอิมพีแดนซ์ $Z_{(Indiff)} = 2\sqrt{R_1^2 + \left(\frac{1}{W_C}\right)^2}$

6.2 การสื่อสารระบบ PCM

PCM (Pulse Code Modulation) เป็นหนึ่งในหลายๆเทคนิคของการส่งสัญญาณต่อเนื่อง (อะนาลอก) ในรูปของสัญญาณไม่ต่อเนื่อง (ดิจิตอล) แนวคิดนี้เริ่มต้นมาตั้งแต่ พ.ศ. 2480 โดยชาวอเมริกันชื่อนาย A.H.Reeves ซึ่งสามารถแก้ปัญหาสัญญาณรบกวน (noise) และความผิดเพี้ยน (distortion) ที่ยังไม่มีผู้ใดสามารถเอาชนะได้ในระบบอะนาลอกอย่างไรก็ดีระบบ PCM ยังไม่ได้รับการยอมรับกันในสมัยนั้น เนื่องจากยังไม่มีการพัฒนาอุปกรณ์พัลส์ความเร็วสูง ในเวลาต่อมาเมื่อได้มีการประดิษฐ์ทรานซิสเตอร์ในปี 2488 จึงมีผู้คิดค้นวิธีการทางเทคนิคพัลส์ความเร็วสูงขึ้น ทำให้การพัฒนาระบบ PCM มีความเร็วสูงขึ้น

ในปี พ.ศ.2502 ห้องปฏิบัติการทางโทรศัพท์เบล ของ AT&T ก็ประสบผลสำเร็จในการพัฒนาชุมสายดิจิตอล (digital Exchange) ซึ่งใช้เทคนิค PCM ในช่องสัญญาณเสียง (channel speech path) แต่ก็ไม่มีการนำออกมาใช้เชิงพาณิชย์เพราะราคาระบบดิจิตอลในขณะนั้นสูงกว่าระบบอะนาลอกหลายเท่าตัว แต่ปัจจุบันด้วยเทคโนโลยี LSI และ VLSI ทำให้ราคาของระบบดิจิตอลลงมาต่ำกว่าระบบอนาลอก

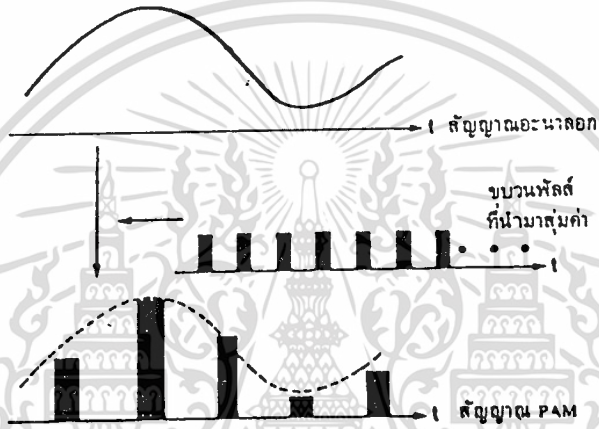


รูปที่ 6.3 ขั้นตอนการประมวลผลสัญญาณในระบบ PCM

หลักการเบื้องต้นของ PCM

จากรูปที่ 1 แสดงให้เห็นถึงขั้นตอนในการแปลงสัญญาณเสียงอะนาลอกไปเป็นสัญญาณดิจิตอล PCM และการแปลงกลับกัน สัญญาณเสียงอะนาลอกจะถูกทำการสุ่มค่า (sampling) แปลงเป็นตัวเลข (quantizing) และเข้ารหัส (coding) ซึ่งทั้งสามขั้นตอนรวมกันเรียกว่า การเปลี่ยนสัญญาณอะนาลอกไปเป็นสัญญาณดิจิตอล (A/D conversion)

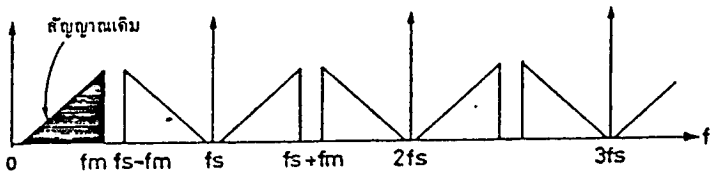
ปกติแล้วแอมปริจูดของสัญญาณเสียงอะนาลอกจะต่อเนื่องกันตลอดตามแกนของเวลา การสุ่มค่าก็คือ ขบวนการนำค่าแอมปลิจูดของสัญญาณอะนาลอกบางค่าในช่วงเวลาซึ่งห่างกันคงที่มาเรียงต่อกัน วิธีการนี้เปรียบเสมือนกับการมอดูเลตทางแอมปลิจูดโดยมีสัญญาณพาห์เป็นขบวนพัลส์ที่มีคาบเวลาคงที่ ซึ่งมามอดูเลตกับสัญญาณเสียงอะนาลอกนั่นเอง ผลลัพธ์ที่ได้จะเป็นสัญญาณที่ไม่ต่อเนื่องตามแกนเวลา ซึ่งเรียกว่า PAM (Pulse Amplitude Modulation) (ดูรูปที่ 6.4)



รูปที่ 6.4 การสุ่มค่าสัญญาณอะนาลอก

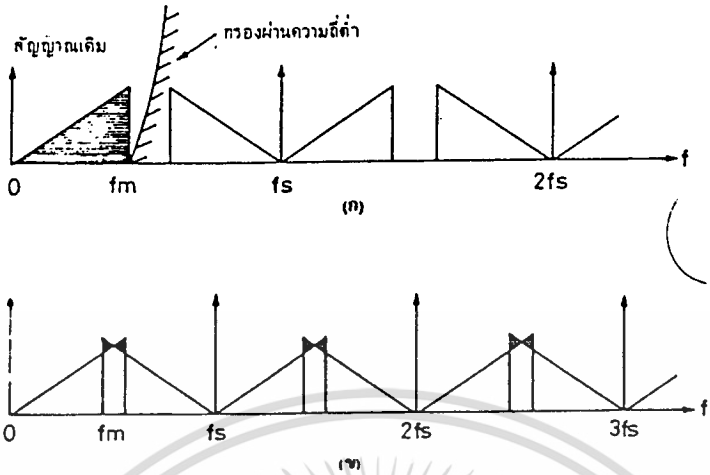
องค์ประกอบสำคัญที่ต้องคำนึงถึงในการสุ่มค่าสัญญาณก็คือ การใช้ความถี่ของพัลส์เท่าไรจึงจะเหมาะสม สิ่งนี้ได้ผ่านการพิสูจน์ทางคณิตศาสตร์และตั้งเป็นทฤษฎีเรียกว่า ทฤษฎีการสุ่มค่า (sampling theory) ซึ่งกล่าวว่า สัญญาณที่มีแอมปลิจูดเป็นฟังก์ชันของเวลา ถูกทำการสุ่มค่าทุกๆ ช่วงเวลาห่างที่คงที่ เมื่อทำการสุ่มค่าด้วยความถี่อย่างต่ำเป็น 2 เท่าของความถี่สูงสุดของสัญญาณนั้นแล้วก็จะสามารถเก็บข่าวสารหรือเนื้อหาของสัญญาณต้นแบบไว้ได้ทั้งหมด

ในรูปที่ 6.5 แสดงให้เห็นแถบความถี่ (frequency spectrum) ของสัญญาณ PAM ซึ่งได้ผ่านขั้นตอนการสุ่มค่าแล้วโดยมีความถี่สูงสุดของสัญญาณอะนาลอก (f_m) และความถี่ของการสุ่มค่า (f_s) เมื่อนำขบวนสัญญาณพัลส์มาวิเคราะห์ทางคณิตศาสตร์ด้วยอนุกรมฟูเรียร์ (fourier series) พบว่าประกอบด้วยฮาร์โมนิกของสัญญาณซายน์ (sine) ที่มีความถี่เป็นทวีคูณของความถี่ขบวนพัลส์รวมกันอยู่ ($0, f_s, 2f_s, 3f_s, \ldots$)



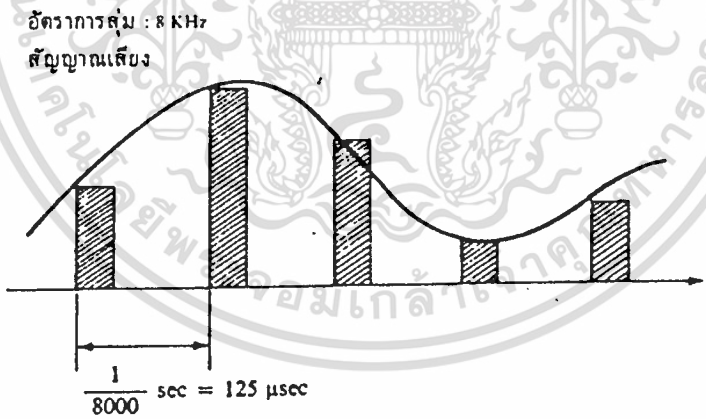
รูปที่ 6.5 แถบความถี่สัญญาณของสัญญาณ PAM ซึ่งผ่านการสุ่มค่าแล้ว

จากรูป6.6ข จะเห็นว่าถ้าความถี่ f_s มีค่ามากกว่า 2 เท่าของค่าความถี่ f_m ความถี่แถบข้าง (side band) จะไม่ซ้อนทับกัน ซึ่งกรณีนี้สามารถนำสัญญาณอะนาลอกกลับคืนมาได้โดยการผ่านวงจรกรองผ่านความถี่ต่ำ (low pass filter) ในทางตรงข้าม (ดูรูปที่ 6.6ข ประกอบ) ถ้าความถี่ f_s มีค่าน้อยกว่าสองเท่าของความถี่ f_m ความถี่แถบข้างจะซ้อนทับกันซึ่งไม่มีวิธีการใดๆที่จะนำสัญญาณอะนาลอกกลับคืนมาได้



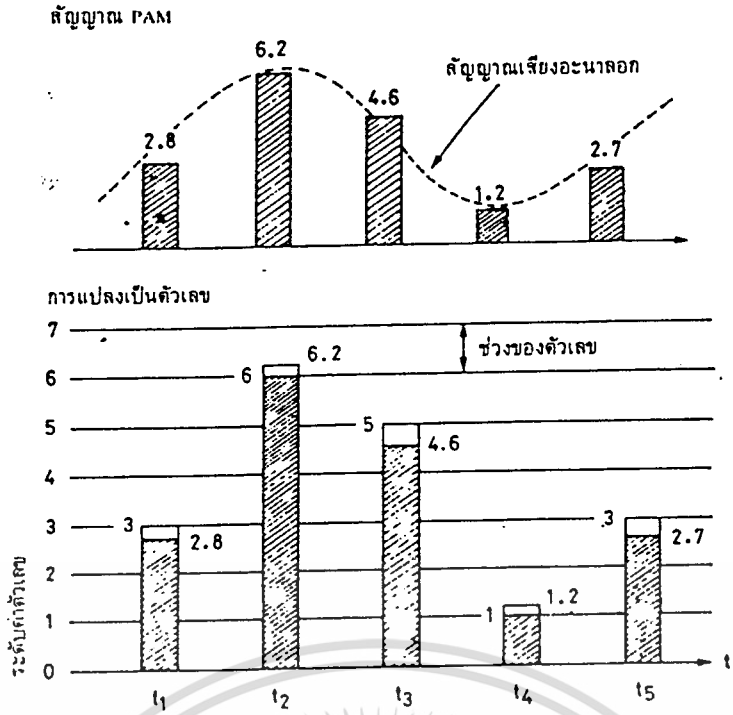
รูปที่ 6.6 แสดงถึงการใช้ค่าความถี่ในการสุ่มที่แตกต่างกัน

CCITT (International Telephone & Telegraph Consultative Committee) แนะนำให้ใช้ความถี่ของการสุ่มค่า 8 กิโลเฮิรตซ์สำหรับสัญญาณเสียงซึ่งโดยปกติแล้วแถบความถี่สัญญาณเสียงที่ใช้ในระบบโทรศัพท์มีค่าจำกัดระหว่าง 0.3 ถึง 3.4 kHz ดังนั้นค่าความถี่ f_s ตามทฤษฎีควรจะเป็น 2 เท่าของ 3.4 kHz หรือเท่ากับ 6.8 kHz อย่างไรก็ดีในทางปฏิบัติจะใช้ค่าความถี่เท่ากับ 8 kHz ด้วยเหตุผลทางด้านเทคนิคในการสร้างวงจรกรองสัญญาณจะได้ง่ายขึ้น ช่วงเวลาที่ใช้ในการสุ่มค่าแต่ละครั้งเท่ากับ 1/8000 วินาที หรือ 125 ไมโครวินาที (ดูรูปที่ 6.7)



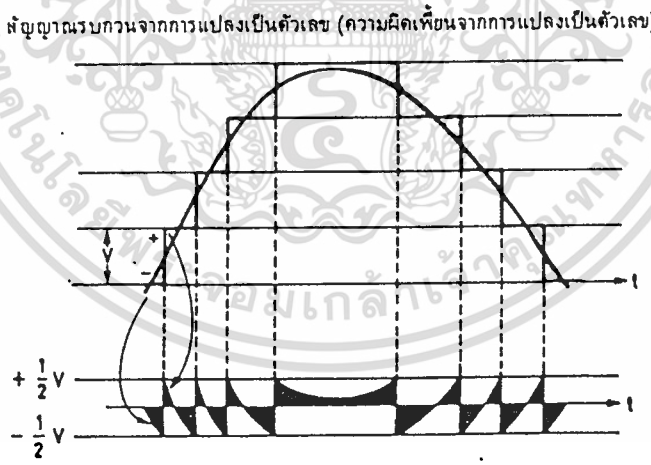
รูปที่ 6.7 ความถี่ของสัญญาณสุ่มค่า 8 KHz สำหรับสัญญาณเสียง

ในขั้นตอนต่อมาสัญญาณ PAM นั้นจะถูกส่งไปผ่านการแปลงเป็นตัวเลข โดยแบ่งขนาดของแอมปลิจูดออกเป็นช่วงๆ (ดูในรูปที่ .86) ค่าของแต่ละช่วงที่ถูกแบ่งเรียกว่าระดับค่าตัวเลข (quantizing level) และระยะระหว่างช่วงที่ถูกแบ่งเรียกว่าช่วงของตัวเลข (quantizing interval) ขนาดของแต่ละสัญญาณสุ่มในสัญญาณ PAM จะถูกแทนด้วยระดับค่าตัวเลข (quantizing level) ที่ใกล้เคียงกับขนาดของมัน ตัวอย่างเช่น สัญญาณที่ถูกสุ่มที่เวลา t_1 มีขนาด 2.8 จะถูกแทนด้วยระดับ 3.0 หรือที่เวลาสัญญาณ t_2 สัญญาณสุ่มมีขนาด 6.2 จะถูกแทนด้วยระดับ 6.0 (ดูรูปที่ 6) ซึ่งเป็นการทำระดับของสัญญาณให้มีช่วงห่างที่แน่นอน เพื่อให้สามารถนำไปแปลงเป็นรหัสฐานสองที่สอดคล้องกับแต่ละระดับ



รูปที่ 6.8 การประมาณระดับของสัญญาณอะนาลอกที่ถูกสุ่มแล้ว

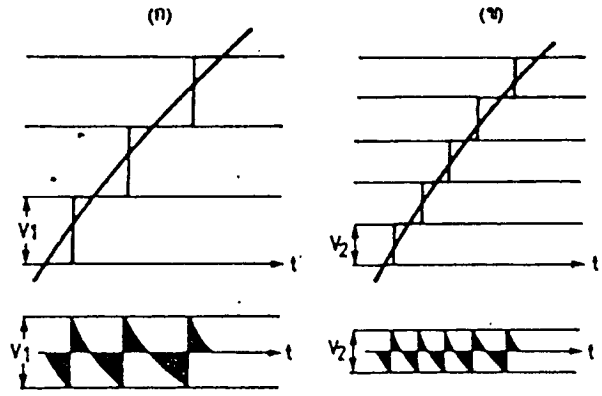
สัญญาณ PAM ที่ถูกแปลงเป็นตัวเลขแล้วจะเป็นเพียงค่าประมาณของสัญญาณอะนาลอก ดังนั้น จึงทำให้เกิดมีค่าผิดพลาดระหว่างสัญญาณทั้งสองทางขนาดของแอมปลิจูด ค่าผิดพลาดนี้เรียกว่า สัญญาณรบกวนจากการแปลงเป็นตัวเลข (quantizing noise) หรือการผิดเพี้ยนจากการแปลงเป็นตัวเลข จะกระจายสม่ำเสมอในระหว่างช่วงของตัวเลขและไม่ขึ้นกับแอมปลิจูดของสัญญาณอะนาลอก นั่นคือระดับกำลังงานของสัญญาณรบกวนจากการแปลงเป็นตัวเลขนั้นค่อนข้างจะคงที่และเป็นอิสระจากระดับกำลังงานของสัญญาณอะนาลอก จะเห็นได้ว่าสัญญาณรบกวนจากการแปลงเป็นตัวเลขนี้เป็นสิ่งที่ไม่สามารถหลีกเลี่ยงได้ แต่สามารถทำให้ลดลงเพื่อรักษาระดับคุณภาพของเสียง (เช่นความชัดเจน)



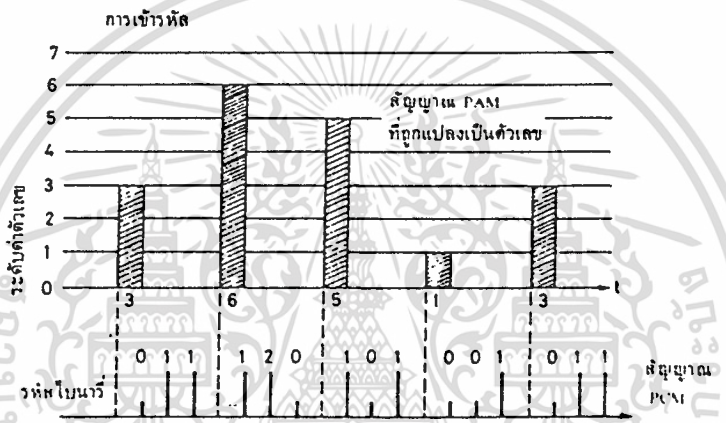
รูปที่ 6.9 สัญญาณรบกวนเนื่องจากการแปลงเป็นตัวเลข

ด้วยเหตุผลที่ว่าแอมปลิจูดของสัญญาณรบกวนจากการแปลงเป็นตัวเลขไม่มีทางเกินกว่าช่วงของตัวเลข ดังนั้นสัญญาณรบกวนจากการแปลงเป็นตัวเลขในรูปที่ 6.10ก. จึงน้อยกว่าในรูปที่ 6.10ข หากเรากำหนดให้ช่วงของตัวเลขมีช่วงเล็กพอเพียงแล้ว สัญญาณรบกวนจากการแปลงเป็นตัวเลขก็จะสามารถลดลงสู่ระดับที่เหมาะสมได้

หลังจากผ่านการแปลงเป็นตัวเลขแล้ว สัญญาณ PAM ที่ได้จะนำไปเข้ารหัสโดยเปลี่ยนเป็นรหัสฐานสอง จากรูปที่ 9 แต่ละค่าที่ผ่านการแปลงเป็นตัวเลขแล้วจะถูกแปลงเป็นเลขฐานสอง 3 บิท สัญญาณที่ได้นี้เรียกว่า สัญญาณ PCM (Pulse Code Modulation) ระดับค่าตัวเลขจะถูกกำหนดโดยจำนวนบิทของเลขฐาน 2 ของแต่ละค่า เช่น ถ้าใช้ n บิทในการเข้ารหัสต่อหนึ่งค่า จะได้จำนวนค่าตัวเลขเท่ากับ 2^n ระดับ ซึ่งทาง CCITT แนะนำให้ใช้การเข้ารหัส 8 บิทต่อหนึ่งค่า หรือให้มีระดับของค่าตัวเลขเท่ากับ $2^n = 256$ ระดับ

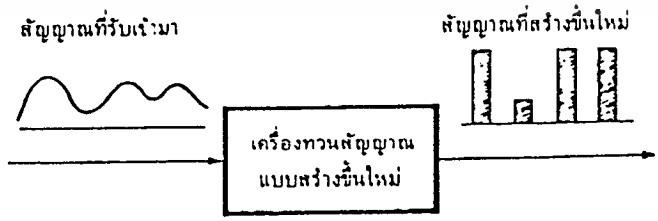


รูปที่ 6.10 ความสัมพันธ์ระหว่างช่วงตัวเลขกับสัญญาณรบกวนจากการแปลงเป็นตัวเลข



รูปที่ 6.11 การนำสัญญาณ PAM มาเข้ารหัส

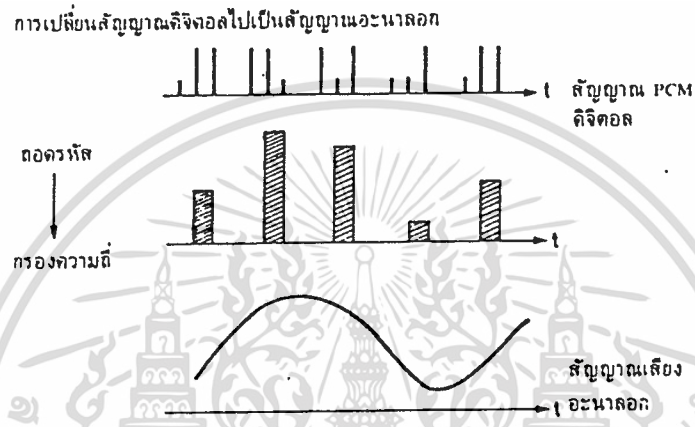
จากทฤษฎีสายส่งสัญญาณทำให้เราทราบว่าหากทำการส่งสัญญาณ PCM ไปตามสายส่งสัญญาณจะเกิดความผิดเพี้ยนของสัญญาณได้เนื่องจากสัญญาณรบกวน, การแทรกสอด (interference) ระหว่างทางและการตอบสนองทางความถี่ (frequency response) แต่ปัญหานี้แก้ไขได้โดยการสร้างสัญญาณ PCM ขึ้นใหม่ที่ด้านรับ ตราบใดที่เครื่องรับยังสามารถตัดสินใจได้อย่างถูกต้องว่าสัญญาณผิดเพี้ยนที่ได้รับนั้นเป็นบิทค่า 0 หรือ 1 ดังนั้นการส่งสัญญาณระบบดิจิตอลจะไม่มีการสะสมผลของสัญญาณรบกวนและสัญญาณแทรกสอด ดังเช่นระบบอะนาลอก



รูปที่ 6.12 การสร้างสัญญาณอะนาลอกขึ้นมาใหม่

ด้านรับเมื่อเครื่องรับได้สัญญาณดิจิตอล PCM ก็จะแปลงกลับไปเป็นสัญญาณอะนาลอก (ดูรูป 6.13) โดยผ่านขั้นตอนการถอดรหัส และการกรองสัญญาณซึ่งเรียกขั้นตอนทั้งสองรวมกันว่า การเปลี่ยนสัญญาณดิจิตอลไปเป็นสัญญาณอะนาลอก (D/A conversion)

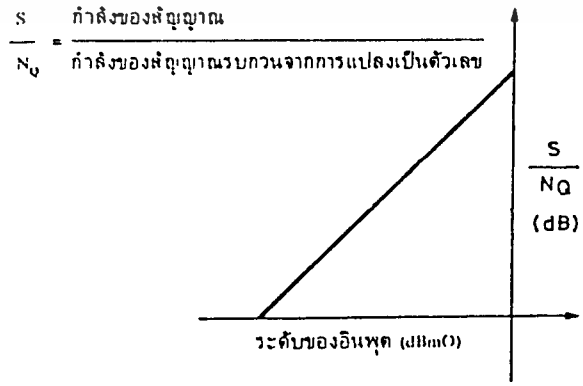
การถอดรหัสนั้นก็ตรงข้ามกับการเข้ารหัส (ดูรูป 6.13) โดยเริ่มต้นจากรหัสฐานสองที่มาจากเครื่องรับสัญญาณ PCM จะถูกนำมาคำนวณและสร้างเป็นระดับค่าตัวเลข และสัญญาณสุ่มค่าจะถูกสร้างขึ้นใหม่ ซึ่งสอดคล้องกับระดับที่คำนวณได้จากข้อมูลฐาน 2 ที่ได้รับนี้ สัญญาณ PCM ที่ถูกแปลงเป็นตัวเลขแล้วที่ด้านส่งก็จะถูกสร้างขึ้นใหม่ที่ด้านรับซึ่งสัญญาณที่ได้ก็ยังคงมีสัญญาณรบกวนจากการแปลงเป็นตัวเลขเช่นเดียวกับทางด้านส่ง สัญญาณ PCM ที่สร้างขึ้นใหม่ที่ด้านรับก็จะถูกส่งผ่านไปยังวงจรกรองผ่านทางความถี่ต่ำก็จะได้สัญญาณเสียงอะนาลอกต่อเนื่องตามแกนเวลา



รูปที่ 6.13 การแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอะนาลอก

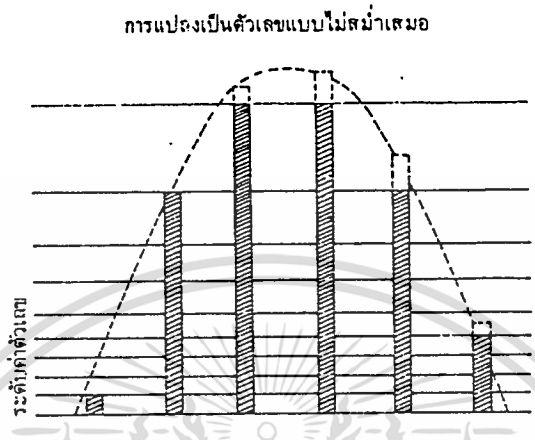
6.3 การอัดและการขยาย (Companding and Expanding)

ในการสื่อสารระบบ PCM สิ่งที่เราไม่สามารถหลีกเลี่ยงได้คือ สัญญาณรบกวนจากการแปลงเป็นตัวเลขเพื่อลดผลที่เกิดขึ้นนี้จึงแก้ไขได้โดยใช้ ขบวนการอัดและขยายสัญญาณ (ไม่ใช่การขยายสัญญาณแบบลิเนียร์) จากที่เคยกล่าวแล้วว่าระดับของสัญญาณรบกวนจากการแปลงเป็นตัวเลขค่อนข้างจะคงที่ไม่ขึ้นกับระดับกำลังงานของสัญญาณเสียง ดังนั้น อัตราส่วนของสัญญาณเสียงต่อสัญญาณรบกวนจากการแปลงเป็นตัวเลข (S/N_Q) จะดีเมื่อระดับความแรงสัญญาณเสียงสูงและจะเลวเมื่อระดับความแรงสัญญาณเสียงต่ำ (ดูรูปที่ 6.14)



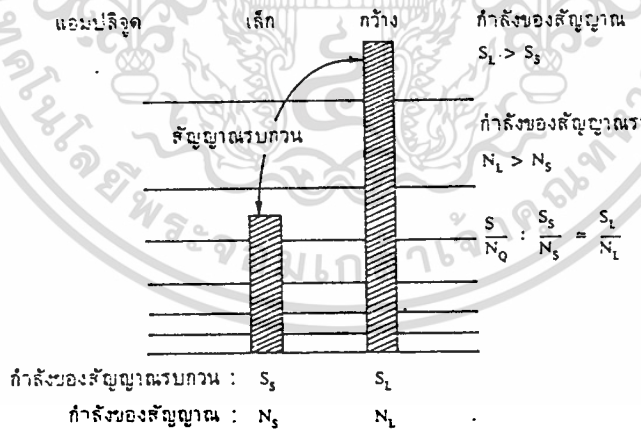
รูปที่ 6.14 ความสัมพันธ์ระหว่างระดับสัญญาณอินพุทและสัญญาณเสียงต่อสัญญาณรบกวนจากการแปลงเป็นตัวเลข

ในทางปฏิบัติหากต้องการให้คุณภาพเสียงดีแล้วค่า S/N_Q ควรจะมีค่าคงที่ในทุกๆ ระดับความแรงของสัญญาณและไม่ควรแก้ปัญหาด้วยการใช้จำนวนบิตมากเกินไป ด้วยจุดประสงค์นี้มีการลดขนาดช่วงของตัวเลขลงที่แอมปลิจูดสัญญาณต่ำๆ และขยายช่วงของตัวเลขขึ้นที่แอมปลิจูดสัญญาณสูงๆ การแปลงเป็นตัวเลขแบบนี้จึงมีช่วงของตัวเลขไม่เท่ากันแตกต่างกันไปตามระดับแอมปลิจูดของสัญญาณซึ่งเรียกว่า การแปลงเป็นตัวเลขแบบไม่สม่ำเสมอ (non-uniform quantizing) (ดูรูป 6.15) ด้วยเหตุผลดังนี้



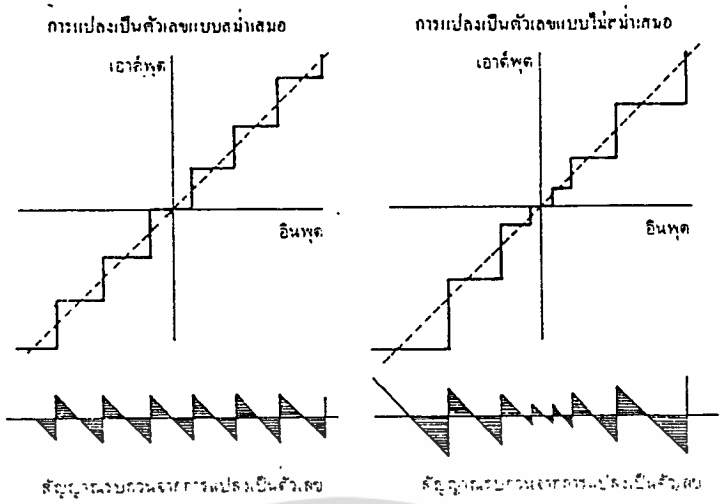
รูปที่ 6.15 การแปลงเป็นตัวเลขแบบไม่สม่ำเสมอ

การกระจายของแอมปลิจูดของสัญญาณเสียงนั้นไม่สม่ำเสมอ (มีการแกว่งขึ้นลงตลอดเวลา) แอมปลิจูดต่ำๆ มีโอกาสเกิดมากกว่าแอมปลิจูดสูงๆ ดังนั้น ค่า S/N_Q สามารถที่จะสังเคราะห์ขึ้นได้ดีกว่าถ้าสัญญาณรบกวนจากการแปลงเป็นตัวเลขถูกทำให้ลดลงสำหรับค่าแอมปลิจูดที่มีโอกาสเกิดมากกว่าและถูกทำให้เพิ่มขึ้นสำหรับค่าแอมปลิจูดที่มีโอกาสเกิดน้อยกว่า (ดูรูป 6.16)



รูปที่ 6.16 ระดับกำลังของค่า S/N

ในรูปที่ 6.16 เป็นการเปรียบเทียบให้เห็นสัญญาณรบกวนจากการแปลงเป็นตัวเลขแบบสม่ำเสมอและการแปลงเป็นตัวเลขแบบไม่สม่ำเสมอ จะเห็นได้ว่าการแปลงเป็นตัวเลขแบบไม่สม่ำเสมอสามารถที่จะลดสัญญาณรบกวนจากการแปลงเป็นตัวเลขได้ที่ค่าแอมปลิจูดสัญญาณต่ำๆ



รูปที่ 6.17 เปรียบเทียบสัญญาณรบกวนระหว่างการแปลงแบบสม่ำเสมอและแบบไม่สม่ำเสมอ

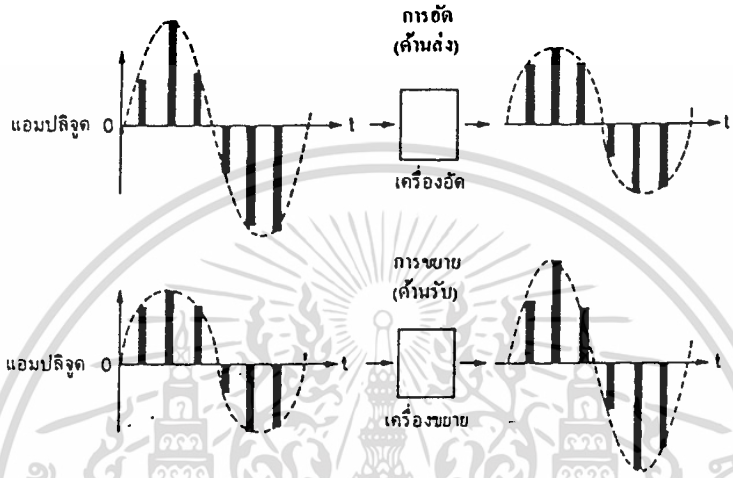
ในกรณีของการแปลงเป็นตัวเลขแบบสม่ำเสมอจะต้องใช้จำนวนระดับค่าตัวเลขประมาณ 2000 ระดับ เพื่อที่จะรักษาคุณภาพเสียงพูดให้อยู่ในเกณฑ์ดี แม้ที่ระดับสัญญาณแอมปลิจูดต่ำก็ตามซึ่งจะต้องใช้จำนวนบิทต่อสัญญาณถึง 11 บิท ซึ่งการใช้จำนวนบิตมากเช่นนี้ต้องใช้อุปกรณ์พัลส์ความเร็วสูงมากซึ่งทำให้ระบบมีราคาแพง ในขณะที่การแปลงเป็นตัวเลขแบบไม่สม่ำเสมอต้องการเพียง 128 ระดับค่าตัวเลขและ 7 บิทต่อสัญญาณสุ่ม 1 ค่าเท่านั้นก็เพียงพอที่จะทำให้ได้ค่า S/N_Q ระดับเดียวกับการแปลงเป็นตัวเลขแบบสม่ำเสมอที่ระดับสัญญาณแอมปลิจูดต่ำ อย่างไรก็ดี CCITT แนะนำการใช้งานการแปลงเป็นตัวเลขแบบไม่สม่ำเสมอด้วยรหัส 8 บิทต่อสัญญาณสุ่ม 1 ค่า และ 256 ระดับค่าตัวเลขเพื่อให้มั่นใจว่าจะได้คุณภาพเสียงที่ดีพอ (ดูรูป 6.17)

ระดับค่าตัวเลขของการแปลงเป็นตัวเลขแบบสม่ำเสมอและการแปลงเป็นตัวเลขแบบไม่สม่ำเสมอ

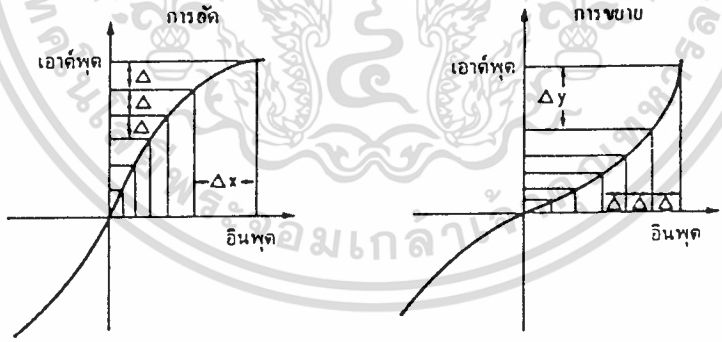
การแปลงเป็นตัวเลขแบบไม่สม่ำเสมอสร้างขึ้นโดยใช้หลักการจัดการกับสัญญาณที่ด้านส่งและด้านรับซึ่งเรียกว่า การอัดและการขยายตามลำดับ รูปที่ 6.18 แสดงให้เห็นหลักการนี้ ทางด้านส่งสัญญาณที่มีแอมปลิจูดสูงจะถูกอัด (compressed) โดยตัวอัด (compressor) แล้วนำไปแปลงเป็นตัวเลขแบบไม่สม่ำเสมอ ขบวนการนี้ทำให้เกิดผลลัพธ์แบบเดียวกับการแปรเปลี่ยนช่วงของตัวเลขโดยขึ้นกับขนาดของแอมปลิจูด ทางด้านรับสัญญาณ PAM จะถูกสร้างขึ้นมาใหม่ โดยการส่งสัญญาณไปยังเครื่องขยาย (expander) ซึ่งมีคุณสมบัติตรงข้ามกับเครื่องอัด (ดูรูป 6.19)

สัญญาณเสียงนั้นมีช่วงการแกว่งขึ้น-ลงของสัญญาณกว้าง (wide dynamic range) ซึ่งการที่จะได้คุณภาพเสียงที่ดีนั้นสัญญาณรบกวนจากการแปลงเป็นตัวเลขต้องมีอัตราส่วนคงที่เมื่อเทียบกับแอมปลิจูดของสัญญาณ ตลอดย่านความกว้างของการแกว่งขึ้นลงของแอมปลิจูดของสัญญาณ เพื่อที่จะบรรลุสิ่งนี้จึงแก้โดยการใช้คุณสมบัติของฟังก์ชันลอการิตมิก (Logarithmic) ในการอัดและขยายสัญญาณ ซึ่งมีผลให้สัญญาณรบกวนจากการแปลงเป็นตัวเลขที่ระดับแอมปลิจูดสัญญาณต่ำๆ สามารถลดลงอย่างน่าพอใจและเราสามารถที่จะรักษาค่า S/N_Q ไว้ให้คงที่ตลอดย่างกว้างการแกว่งขึ้น-ลงของแอมปลิจูดของสัญญาณ

CCITT แนะนำให้ใช้คุณสมบัติของลอการิตมิกในการอัดและขยายสัญญาณ 2 แบบดังนี้ แบบแรกเรียกกันว่า A-law นิยมใช้กันในยุโรป ส่วนอีกแบบเรียกว่า u-law นิยมใช้กันในแถบอเมริกาเหนือและญี่ปุ่น ในรูปที่ 6.18 และ 6.19 แสดงให้เห็นคุณสมบัติการอัดและการขยายของ A-law และ u-law ตามลำดับ เส้นโค้งทั้งสองแสดงให้เห็นลักษณะการอัดสำหรับแอมปลิจูดของสัญญาณชีกบวก (ชีกลบเป็นลักษณะคล้ายกันแต่ไม่ได้แสดงรูปไว้) CCITT แนะนำว่าการอัดและการขยายนั้นในทางปฏิบัติควรคำนึงการประมวลผลสัญญาณดิจิตอล ซึ่งจะเห็นได้ชัดจากรูปว่าคุณสมบัติในการอัดและการขยายแบบลอการิตมิกถูกนำมาใช้ในทางปฏิบัติโดยการแบ่งโค้งออกเป็นช่วงๆ แต่ละช่วงประมาณด้วยกราฟเส้นตรง โค้งของ A-law และ u-law นั้นถูกประมาณด้วยกราฟเส้นตรง 13 ช่วงและ 15 ช่วงตามลำดับซึ่งทั้ง A-law และ u-law ใช้ระดับค่าตัวเลข 256 ระดับและเข้ารหัสแต่ละสัญญาณสุ่ม 1 ค่าด้วย 8 บิท



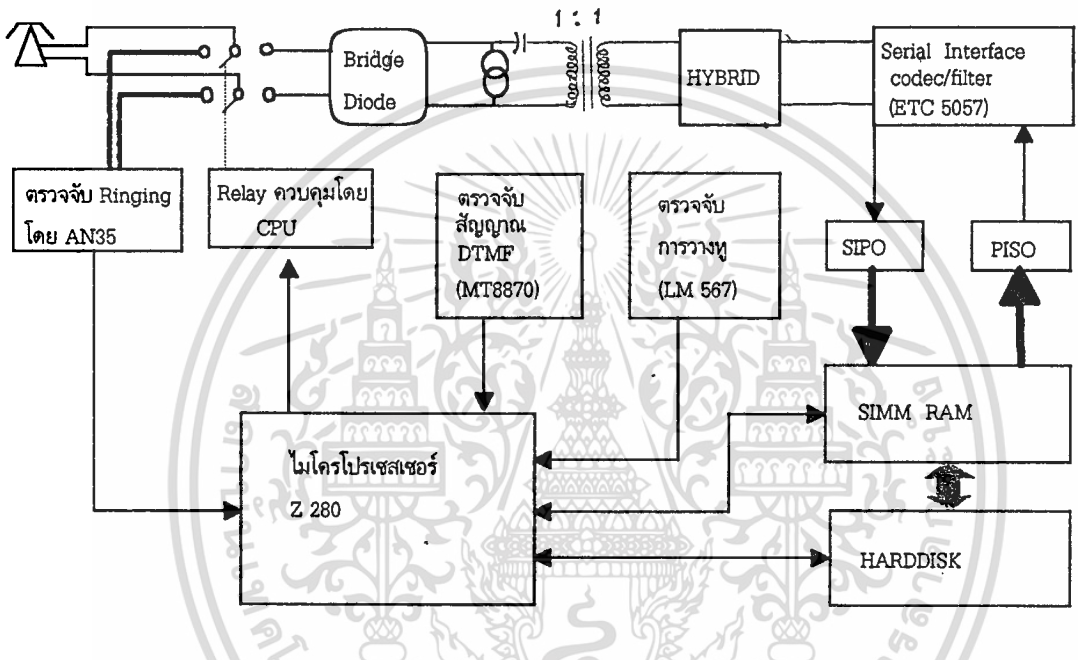
รูปที่ 6.18 การแปลงเป็นตัวเลขแบบไม่สม่ำเสมอโดยใช้หลักการ อัดและการขยาย



รูปที่ 6.19 คุณลักษณะของการอัดและการขยาย

บทที่ 7

วงจรส่วนติดต่อกับโทรศัพท์ และ ส่วนถ่ายทอดสัญญาณเสียง



รูปที่ 7.1 บล็อกไดอะแกรมของส่วนติดต่อกับโทรศัพท์และส่วนถ่ายทอดสัญญาณเสียง

1. ส่วนตรวจจับสัญญาณกระดิ่ง

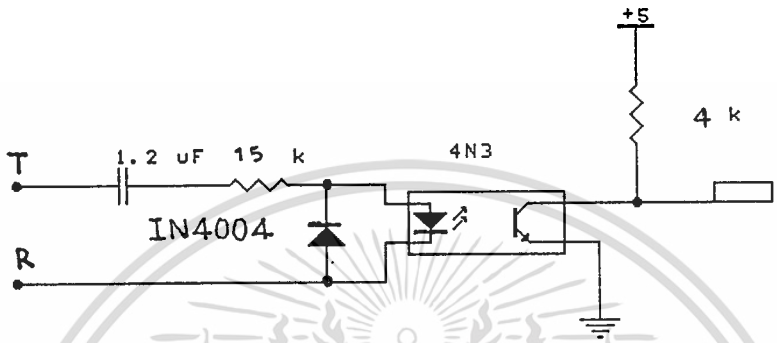
เป็นส่วนที่ใช้สำหรับตรวจจับสัญญาณกระดิ่งที่มาตามสายในขณะที่มีผู้โทรเข้ามา อาศัยหลักพื้นฐานของสัญญาณโทรศัพท์ คือ ในขณะที่สายว่าง คู่สายโทรศัพท์จะมีแรงดันไฟตรง (dc) ประมาณ 48 โวลท์ ซึ่งจ่ายมาจากชุมสายโทรศัพท์ และเมื่อมีผู้เรียกเลขหมายเข้ามา ทางชุมสายจะจ่ายสัญญาณกระดิ่งมาเป็นแรงดันกระแสสลับที่มีแรงดันประมาณ 70-100 โวลท์ ส่วนตรวจจับสัญญาณจะให้พัลส์ออกมาเมื่อมีกระดิ่งดังเข้ามา

หลักการทำงานของ Optocouple

การนำเอาออปโต้คอปเปลอร์เข้ามาใช้ในวงจร ก็เพื่อจุดประสงค์ในการแยกส่วนของแรงดันไฟสูงที่เกิดจากสายโทรศัพท์ซึ่งมีค่าประมาณ 100 V_{AC} ในขณะที่มีสัญญาณกริ่งดังเข้ามาให้เชื่อมต่อกับวงจรที่มีระดับไฟเลี้ยงขนาด 5 V_{DC} ซึ่งออปโต้คอปเปลอร์นี้จะมีความสำคัญอย่างยิ่งที่จะต้องนำมาใช้งานเพื่อให้วงจรที่มีระดับแรงดันต่างกันมาก สามารถทำงานร่วมกันได้ ในส่วนของวงจรนี้ใช้ออปโต้คอปเปลอร์เบอร์ 4N35

สำหรับรายละเอียดของ 4N35 เป็นดังนี้

- ใช้ GaAS ไดโอดซึ่งแปลงแสงอินฟราเรดไปยัง Silicon NPN Photo transistor
- การแปลง (Transfer) ของไฟกระแสตรงที่มีค่าสูง
- สามารถแยกวงจรที่มีระดับความดันที่ต่างกันได้ถึง 7.5 kVac
- ความเร็วในการสวิทช์ชิ่งสูง



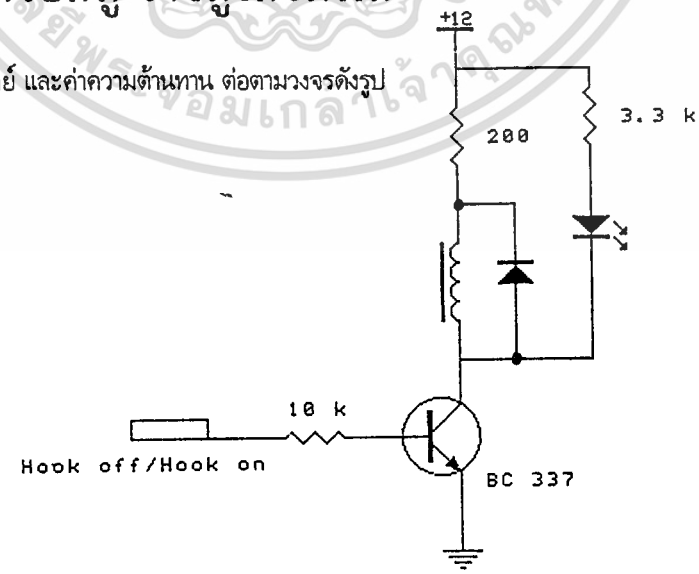
รูปที่ 7.2 วงจรตรวจจับสัญญาณกระดิ่งโดยใช้ออปโตคอปเปลอร์

- ค่าความต้านทาง 2.2 k ทำหน้าที่เป็นตัวจำกัดกระแสที่ไหลผ่านไดโอดอินฟราเรด และค่าความต้านทาง 4 k เป็นตัวกำหนดกระแสไหลผ่าน Photo transistor
- ค่าตัวเก็บประจุ 1.2 uF/250 V ทำหน้าที่กั้นไฟกระแสตรง ไม่ให้ไดโอดอินฟราเรดทำงาน เพื่อไม่ให้โฟโตทรานซิสเตอร์ทำงานในสภาวะปกติ แต่จะให้ทำงานในกรณีที่มีสัญญาณกริ่ง AC เท่านั้น

เมื่อมีสัญญาณกริ่งโทรศัพท์เข้ามาจะทำให้ ออปโต้คอปเปลอร์ทำงาน จะได้สัญญาณที่ขา collector เป็นลักษณะของสัญญาณพัลส์ ความถี่ 20 Hz ซึ่งจะนำสัญญาณที่ได้ส่งไปให้ไมโครโปรเซสเซอร์ตรวจสอบ

7.2 ส่วนควบคุมการยกหู วางหูโทรศัพท์

ส่วนนี้จะประกอบด้วยทรานซิสเตอร์ รีเลย์ และค่าความต้านทาน ต่อตามวงจรดังรูป

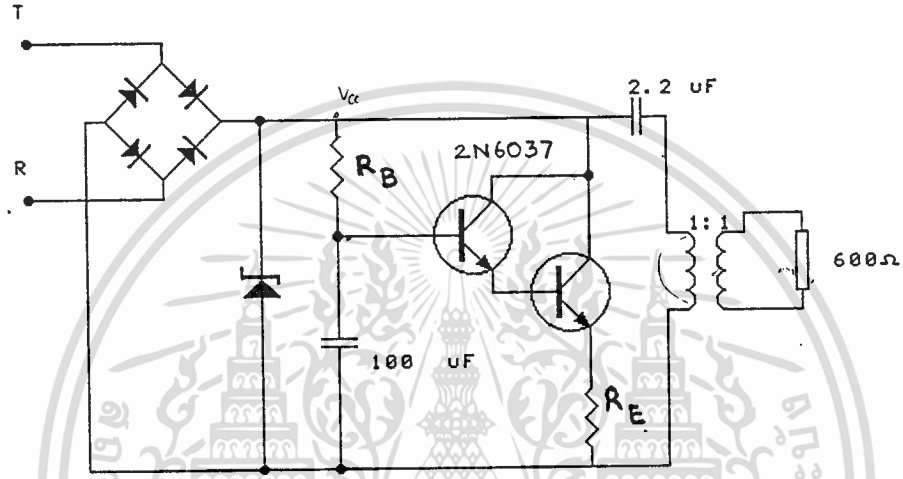


รูปที่ 7.3 วงจรส่วนควบคุมการยกหู/วางหูโทรศัพท์

เมื่อไมโครโปรเซสเซอร์ทราบว่ามีสัญญาณกริ่งโทรศัพท์เข้ามาแล้วจะทำการส่งค่า 1 (+5 V) ผ่านทาง D-Flipflop ไปยังขาเบสของทรานซิสเตอร์ ทำให้ทรานซิสเตอร์นำกระแสมีผลไปยังรีเลย์ทำการสวิทช์คู่สายโทรศัพท์ต่อเข้ากับวงจร ส่วน LED มีไว้แสดงสถานะว่ามีการยกหูโทรศัพท์แล้ว

7.3 ส่วนดึงกระแส (Current Sink)

เมื่อรีเลย์ทำงานจะทำให้คู่สายโทรศัพท์ (Tip กับ Ring) ถูกต่อเข้ากับวงจรและพร้อมกันนั้น ทางชุมสายจะรับรู้โดยจ่ายกระแสไฟตรงออกมา ประมาณ 25-30 mA เพื่อให้กระแสไหลครบ loop จึงต้องมีวงจรส่วนนี้ ขณะเดียวกันความต่างศักย์ของคู่สายโทรศัพท์จะลดลงมา 6-10 โวลท์ มีวงจรดังรูป



รูปที่ 7.4 วงจรส่วนดึงกระแส

- วงจรบริดจ์มีไว้เพื่อกำหนดทิศทางการไหลของกระแส
- R_1 และ R_2 เป็นตัวกำหนดค่าของกระแสที่ไหลผ่านวงจรโดยหาได้จากสมการ

$$Ic = \frac{\beta_0\left(V_{CC}-V_{BE}\right)}{R_B+\left(\beta_0+1\right)R_E}$$

กำหนดให้ที่ . V_{cc} = 6 โวลท์ I_c = 25 mA β_o = 1000

และให้ R_E = 100 จะได้ค่า R_B = 84 k

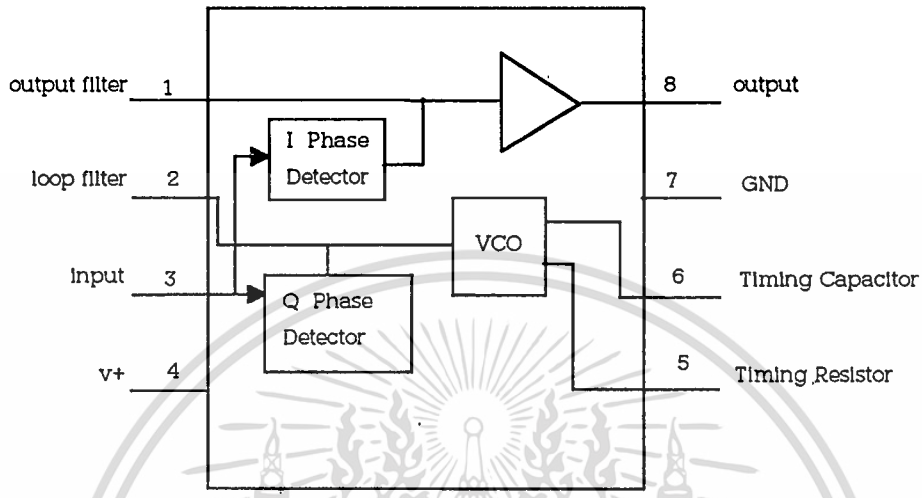
- ตัวเก็บประจุค่า 100 µF มีไว้เพื่อกำหนดค่า AC Impedance ขณะมีสัญญาณเสียงเข้ามาซึ่งเท่ากับ R_E
- ตัวเก็บประจุ 2.2 µF มีเพื่อให้เฉพาะสัญญาณเสียงถูกคัปปลิ้งไปยังวงจรถ่ายทอดสัญญาณเสียง

7.4 ส่วนตรวจจับการวางหูโทรศัพท์

เมื่อทางฝ่ายผู้เรียกทำการวางหูโทรศัพท์จะมีสัญญาณไม่ว่าง (Busy Tone) ซึ่งเป็นสัญญาณที่มีความถี่ 400 Hz ดังนั้นการตรวจสอบทางโทรศัพท์จึงใช้ IC LM567 ซึ่งเป็น Tone Decoder

7.4.1 ส่วนประกอบของ IC LM567

เป็นวงจร Phase Lock Loop คือ เป็นไอซีที่ให้สัญญาณ OUTPUT เป็น Low เมื่อความถี่ INPUT มีค่าเท่ากับความถี่ศูนย์กลาง (f_0) โดยสามารถตั้งความถี่ศูนย์กลาง และ Bandwidth เราสามารถตรวจจับความถี่ได้ตั้งแต่ 0.01 Hz - 500 kHz โดยมี Diagram ดังรูป



รูปที่ 7.5 Diagram ของ IC LM567

รายละเอียดและหน้าที่ของแต่ละขา

- OUTPUT FILTER เป็นขาที่ใช้ใส่ C เพื่อเป็นตัวกรองสัญญาณที่จะส่งออกไป โดยค่า C จะกำหนดดังนี้

  $C_3 = 2C_2$

  โดย C_3 = ค่า C ที่ขา 1 C_2 = ค่า C ที่ขา 2

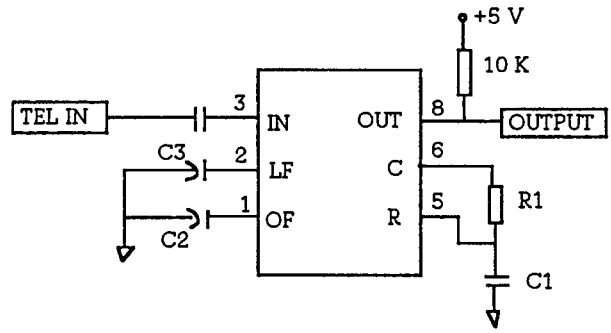
- LOOP FILTER เป็นขาที่ใช้ใส่ C เพื่อเป็นตัวกรองสัญญาณที่จะรับเข้ามา โดยค่า c จะกำหนดได้ดังนี้

  $C_2 = \frac{n}{f_0}$ ซึ่ง $500 < n < 62000$

  f_0 = ค่าความถี่ศูนย์กลาง

- INPUT เป็นขาซึ่งใช้รับสัญญาณความถี่เพื่อนำมาเปรียบเทียบกับความถี่ศูนย์กลาง
- V+ ไฟเลี้ยงบวกมีค่าได้ดังนี้ $4.75 < V+ < 9$
- TIMING RESISTOR เป็นขาที่ใช้กำหนดความถี่ศูนย์กลางร่วมกับขา Timing Capacitor
- TIMING CAPACITOR เป็นขาที่ใช้กำหนดความถี่ศูนย์กลาง โดยจะกำหนดค่าได้จาก $f_0 = \frac{1}{1.1R_1C_1}$
- GND ขา Ground ของ IC
- OUTPUT เป็นขาเอาท์พุทที่จะส่ง GND ออกมาเมื่อสัญญาณ INPUT มีค่าความถี่เท่ากับค่าความถี่ศูนย์กลาง

7.4.2 วงจรจับสัญญาณที่ใช้ในโครงงานนี้



รูปที่ 7.6 วงจรตรวจจับการวางหูโทรศัพท์

จะได้ค่าต่างๆดังนี้

- ค่าความถี่ศูนย์กลาง (f_0) = $\frac{1}{1.1R_1C_1}$

  f_0 = 400 Hz C_1 = 0.47 µF R_1 ~ 5k

- ค่า C ที่ขา 2 (Loop Filter)

  $C_2 = \frac{n}{f_0}$ ซึ่ง 500 < n < 62000

  C_2 = 2.2 µF

- ค่า C ที่ขา 1 (Output Filter)

  $C_3 = 2C_2$

  C_3 ~ 4.7 uF

7.5 วงจรแปลงความถี่ของระบบโทน (tone)

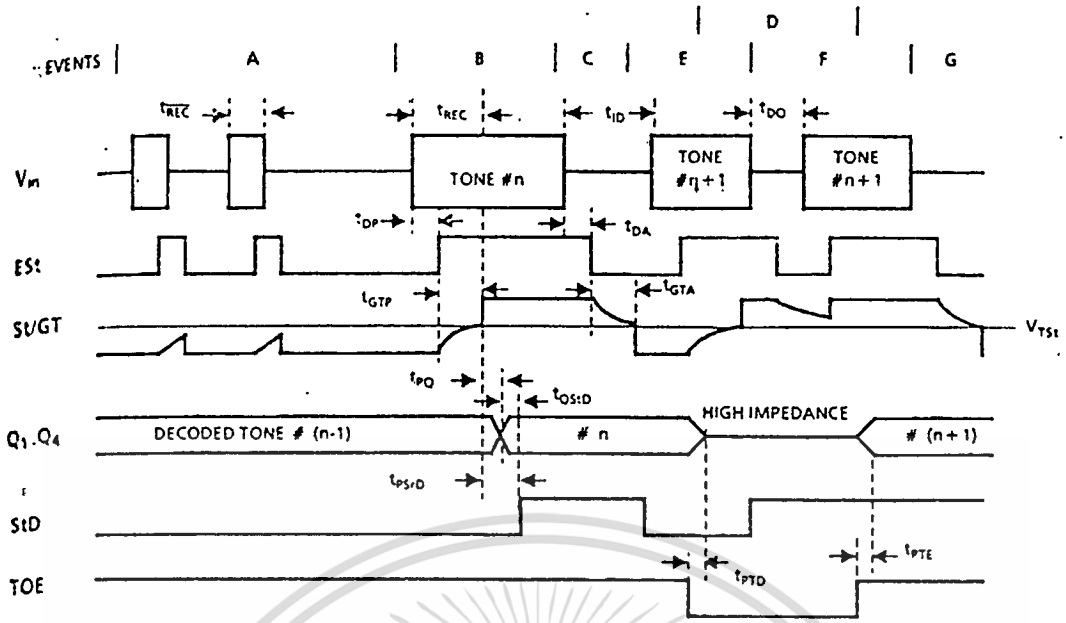
เนื่องจากโครงงานนี้มีการติดต่อระหว่างไมโครโปรเซสเซอร์ และการที่ผู้ใช้จะติดต่อจะกระทำผ่านทางปุ่มโทรศัพท์ โดยใช้ IC 8870 เป็นตัวที่ทำการถอดรหัส ให้กลายเป็นตัวเลขฐานสอง 4 บิท

7.5.1 การทำงานภายใน MT8870 มีส่วนสำคัญ 5 ส่วน ดังนี้

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

7.5.2 รายละเอียด และหน้าที่การทำงานของแต่ละขา

- IN+ นอนอินเวอร์ติ้งออปแอมป์อินพุท (Non-inverting)
- IN- อินเวอร์ติ้งออปแอมป์อินพุท (Inverting)
- GS เป็นขาที่ Feedback จากเอาท์พุทให้ผู้ใช้ได้เชื่อมตัวความต้านทานเพื่อเลือกค่า Gain ได้ตามต้องการ
- V_{REF} ค่าความต่างศักย์เปรียบเทียบที่เอาท์พุท ปกติมีค่า Vdd/2
- INH ถ้าให้ขานี้เป็น high จะไม่มีการถอดรหัส A,B,C,D ที่ปรากฏอยู่บนแป้นโทรศัพท์
- PWDN ถ้าให้ขานี้เป็น high จะหยุดการทำงานของไอซีเบอร์นี้
- OSC1 อินพุทของสัญญาณนาฬิกา
- OSC2 เอาท์พุทของสัญญาณนาฬิกา โดยใช้กับผลึก (Crystal) ที่มีความถี่ 3.5795 Mhz ต่อเข้าระหว่าง osc1 และ osc2
- Vss ไฟเลี้ยงขั้วลบ
- Q_1-Q_4 เป็นขาเอาท์พุทที่ถูกควบคุมแบบ 3 state ด้วยสัญญาณ TOE จะให้ค่าออกมาเมื่อได้รับค่าความถี่ที่ถูกต้อง และเอาท์พุทถูกอินาเบิล
- Std Delay Steering Output ให้ค่าลอจิกเป็น 1 เมื่อมีข้อมูลระดับแรงดัน
- EST Early Steering Output จะให้ค่าลอจิก 1 เมื่อข้อมูลถูกต้องและ ถ้าเป็น 0 จะไม่สัญญาณผ่านเข้ามา
- ST/GT Steering input/time output (bi-direction) ทำหน้าที่ส่งสัญญาณควบคุมวงจร RC ภายนอกเพื่อควบคุมไทม์
- Vdd ไฟเลี้ยงบวก



รูปที่ 7.7 ไทม์มิ่งไดอะแกรมของไอซีเบอร์ MT8870

7.5.3 ขั้นตอนการทำงาน

A - ตรวจพบความถี่เข้ามา แต่คาบเวลาไม่ถูกต้อง เอาท์พุทไม่เปลี่ยน

B - ความถี่ #n ถูกตรวจพบและมีคาบเวลาที่ถูกต้อง ความถี่ถูกถอดรหัส และแลทช์ไว้ที่เอาท์พุท

C - จบความถี่ #n ช่วงห่างถูกต้อง เอาท์พุทยังคงแลทช์อยู่จนกว่าจะได้รับความถี่ที่ถูกต้องใหม่

D - เอาท์พุทเปลี่ยนเป็น ไฮอิมพีแดนซ์

E - ความถี่ #n+1 ถูกตรวจพบ คาบเวลาถูกต้อง ความถี่ถูกถอดรหัสและแลตช์ไว้

F - ความถี่ #n+1 หายไป ช่วงห่างไม่ถูกต้อง เอาท์พุทยังคงแลทช์อยู่

G - จบความถี่ #n+1 ช่วงห่างถูกต้อง เอาท์พุทยังคงแลทช์อยู่ จนถึงความถี่ใหม่ที่ถูกต้อง

อธิบายคำศัพท์

Vin - สัญญาณความถี่ DTMF ที่เข้ามา

Est - Early Steering Output ใช้แสดงความถี่ที่ถูกต้อง

St/Gt - Steering input/Guard time output สำหรับต่อกับ RC ภายนอก

Q1-Q4 - เอาท์พุท BCD ขนาด 4 บิท

StD - Delayed Steering output ใช้แสดงว่าความถี่ที่ได้รับหรือหายไป มีคาบเวลาตามที่กำหนด เพื่อแสดงความถูกต้องของสัญญาณ

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

t_{REC} - คาบเวลานานสุดที่ตรวจพบความถี่ DTMF แล้วยังไม่ถูกต้อง

t_{REC} - คาบเวลาสั้นสุดที่ต้องการเพื่อแสดงว่าสัญญาณถูกต้อง

t_{ID} - เวลาสั้นสุดระหว่างสัญญาณ DTMF ที่ถูกต้อง 2 สัญญาณ

t_{DO} - เวลานานสุดที่ยอมให้สัญญาณหายไปได้ในคาบเวลาความถี่ที่ถูกต้อง

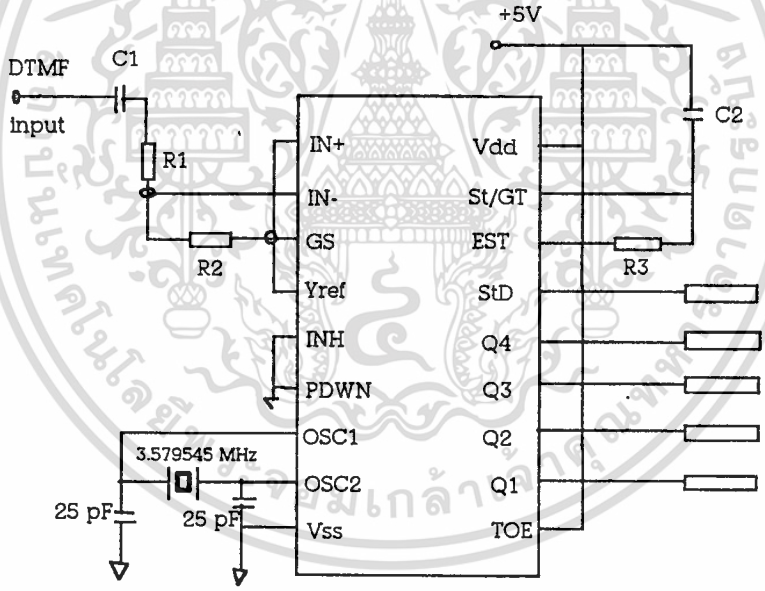
t_{DP} - เวลาที่ใช้ในการตรวจพบสัญญาณความถี่ DTMF ที่ถูกต้อง

t_{DA} - เวลาที่ใช้ในการตรวจการหายไปของสัญญาณความถี่ DTMF ที่ถูกต้อง

t_{GTP} - การ์ดไทม์ของการปรากฏความถี่ DTMF

t_{GTA} - การ์ดไทม์ของการหายไปของความถี่ DTMF

7.5.4 วงจรถอดรหัสสัญญาณ DTMF เป็นเลขฐานสองขนาด 4 บิทที่ใช้ในโครงงานนี้



รูปที่ 7.8 วงจรถอดรหัสสัญญาณ DTMF

$R_1 = R_2 = 100\ k\Omega$ $C_1 = C_2 = 100\ nF$ $R_3 = 300\ k\Omega$

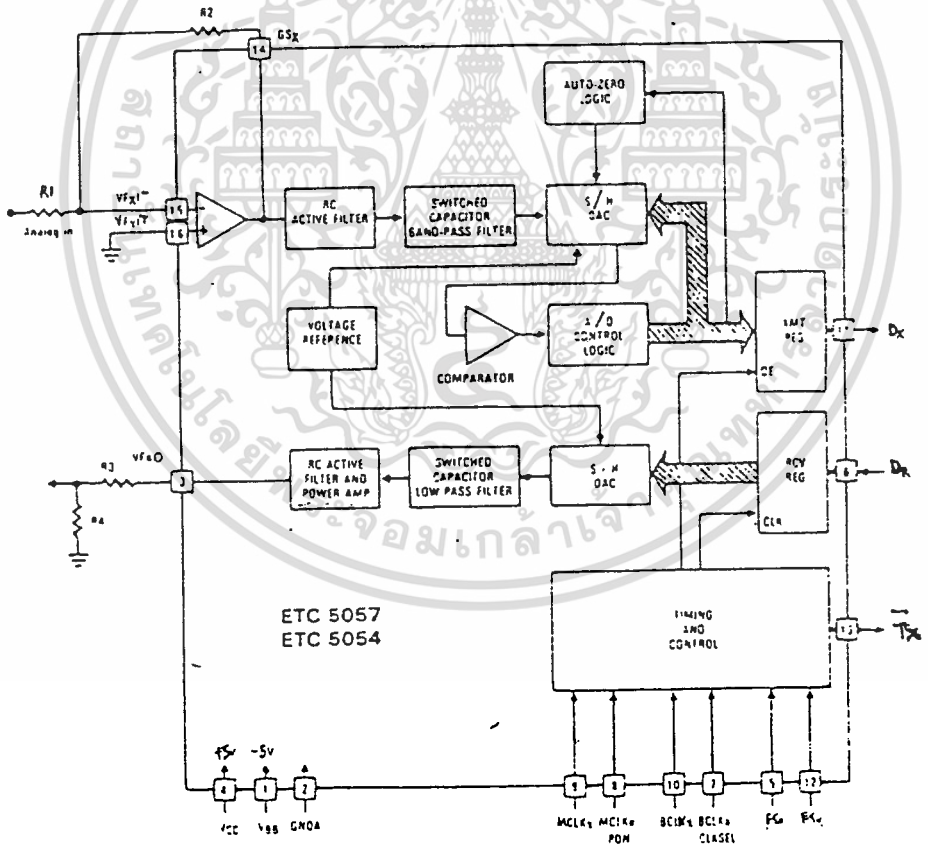
$t_{GTA} = t_{GTP} = (0.1\ u \times 300K)\ \ln(5/2.5) = 20\ ms$

7.6 ส่วนถ่ายทอดสัญญาณเสียง

วงจรในส่วนนี้จะใช้ไอซี ETC 5057 ซึ่งเป็นไอซี Serial interface codec and filter โดยจะทำการแปลงสัญญาณเสียงเป็นสัญญาณดิจิตอลขนาด 8 บิทและแปลงสัญญาณดิจิตอลกลับเป็นสัญญาณเสียง ซึ่งการใช้งานวงจร Codec จะออกแบบให้ใช้งานวงจร Codec 1 วงจรต่อ 1 คู่สายโทรศัพท์ เพื่อเป็นการป้องกันการรบกวนกันของข้อมูล ถ้ามีการใช้วงจร Codec ร่วมกันในระหว่างหลายคู่สาย ผลตามมาก็คือการมีเสียงแทรกสอดเข้ามาระหว่างคู่สาย หรือที่เรียกว่า Crosstalk

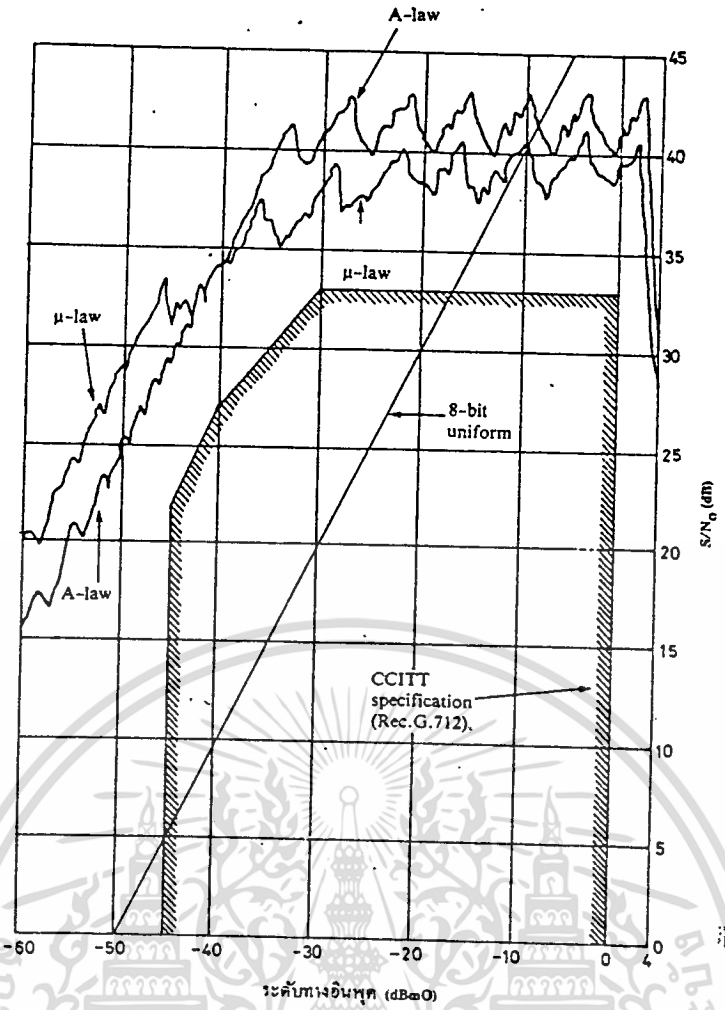
7.6.1 โครงสร้างของ ETC 5057

จากบล็อกไดอะแกรมของวงจรในภาคส่งนั้น สัญญาณอะนาลอกจะถูกแซมปลิ้งและทำการคอมแพนดิ้ง ข้อมูลดิจิตอลที่ถูกอัดแล้วจะถูกนำไปเก็บไว้ในบัฟเฟอร์ข้อมูล และถูกส่งออกไปตามจังหวะของสัญญาณนาฬิกา ส่วนในภาครับข้อมูลดิจิตอลจะผ่านเข้ามาทางบัฟเฟอร์ข้อมูล และผ่านเข้าสู่วงจรขยายข้อมูล (expanding) ภายในส่วนถอดรหัสและผ่านเข้าสู่วงจร DAC เพื่อแปลงเป็นสัญญาณอะนาลอก จากนั้นจะผ่านเข้าสู่วงจรแซมเปิ้ล และ โฮลด์ก่อนที่จะผ่านไปสู่วงจรกรองความถี่ต่ำผ่าน (smoothing filter) เพื่อให้ได้สัญญาณมีความเรียบมากขึ้น



รูปที่ 7.9 บล็อกไดอะแกรมของไอซีเบอร์ ETC5057

การคอมแพนดิ้ง คือ การปรับระดับความละเอียดของรหัสให้สัมพันธ์กับแอมปลิจูดของสัญญาณอะนาลอก ซึ่งกราฟความสัมพันธ์ที่ออกมาจะไม่เป็นเชิงเส้นอีกแล้ว แต่จะเป็นลอการิทึมแทน ดังรูป

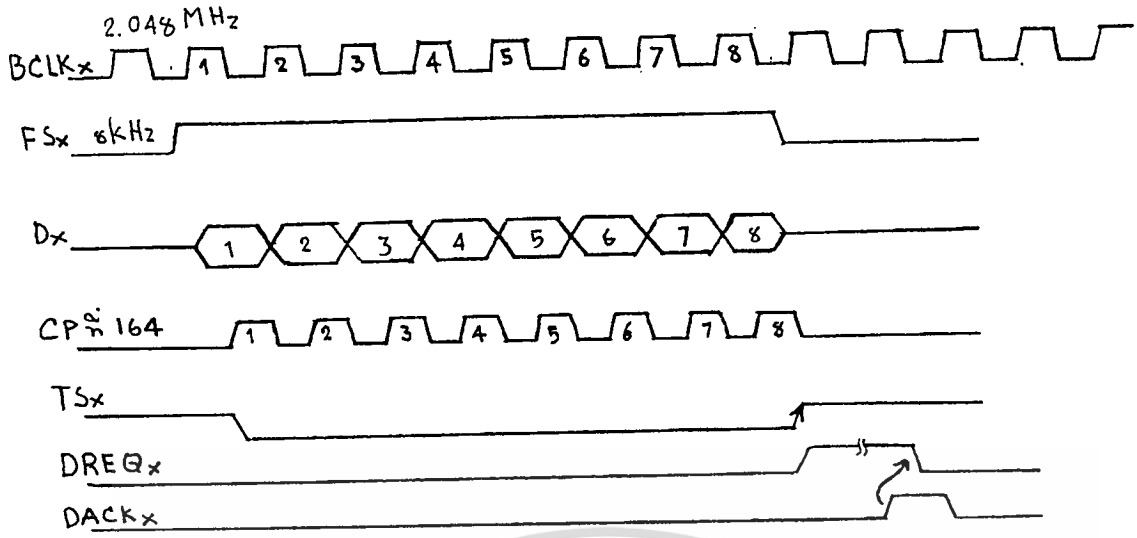


รูปที่ 7.10 เปรียบเทียบค่า S/N ระหว่างการเข้ารหัส

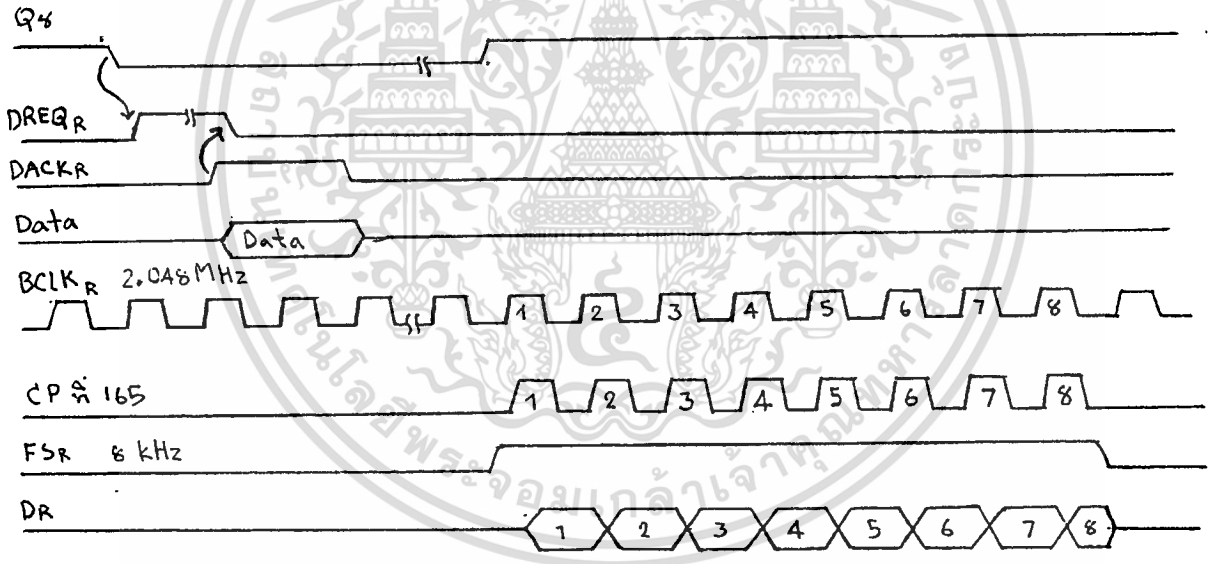
จะเห็นได้ว่าในช่วง 1/2 ถึง 1 ของแรงดันอินพุทสัมพันธ์จะมีรหัสในช่วงนี้เพียง 16 รหัส แต่ที่ระดับสัญญาณอินพุทค่าต่ำๆ เช่น ในช่วง 1/64 ถึง 1/32 รหัสช่วงนี้ก็มี 16 รหัสเช่นกัน แรงดันอินพุทยิ่งต่ำก็จะมีความละเอียดในการเข้ารหัสมากขึ้น หรือมีความละเอียดในการประมาณค่ามากขึ้น วิธีการตรงส่วนนี้เรียกว่า "การอัดข้อมูล" (Compression) ซึ่งจะปรากฏในวงจรภาคส่ง เมื่อเป็นเช่นนี้ทางภาครับก็จะต้องมีการแปลงสัญญาณกลับโดยตัวถอดรหัสซึ่งจะมีกระบวนการถอดรหัส จนในที่สุดสัญญาณก็จะถูกแปลงกลับมาอย่างเดิม ซึ่งกระบวนการที่ภาคนี้ เรียกว่า "การขยายข้อมูล" (expansion) การคอมแพนดิ้งแบ่งได้ 2 แบบคือแบบ μ-law และ A-law ซึ่งมีข้อแตกต่างกันคือ แบบ A-law จะให้ค่าอัตราส่วน S/N สูงกว่าแบบ μ-law เล็กน้อยที่ระดับสัญญาณค่าต่ำๆ แต่ในสภาวะที่เรียกว่า idle channel noise (คือ สภาวะที่ค่าของสัญญาณรบกวนเนื่องมาจากการควอนไตซ์มีค่าสูง เมื่อเทียบกับสัญญาณเสียงจริงๆ) คอมแพนดิ้งแบบ μ-law จะมีสัญญาณรบกวนน้อยกว่า

ในการสร้างวงจรส่วนแปลงสัญญาณเสียงเป็นดิจิตอล จาก Timing Diagram โดยเริ่มจากเมื่อสัญญาณ FSX เปลี่ยนจาก Low เป็น High ในขณะที่ BCIKx เป็น Low จะเป็นการนำข้อมูลที่ถูกแปลงเป็นรหัส PCM 8 บิทจากรีจิสเตอร์ภายในของ ETC 5057 ออกมายังขา Dx โดยแต่ละบิทจะถูกปล่อยออกมาตามช่วงเวลาของ BCIKx ขณะที่ข้อมูลแต่ละบิตออกมา ข้อมูลจะถูก shift เข้าไปเก็บไว้ใน 74LS164 จนครบ 8 บิทแล้ว จะมีสัญญาณ DREQ ออกมาส่งไปยัง 8237A-5 (DMA Controller) เพื่อที่จะเก็บข้อมูลลงยังหน่วยความจำ

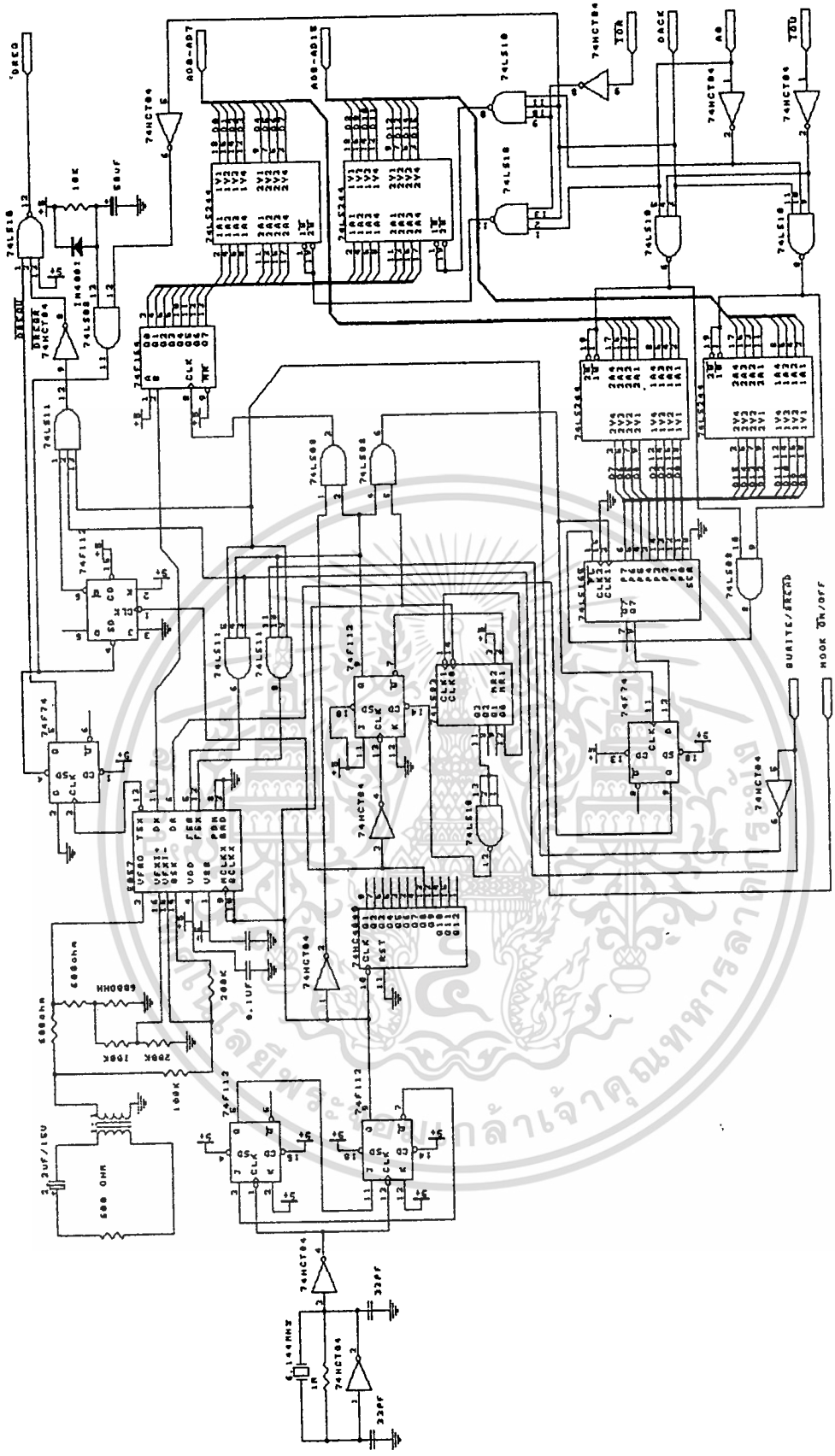
การนำสัญญาณดิจิตอล (PCM 8 bit) มาทำการแปลงเป็นสัญญาณเสียงจะเริ่มจากที่ขอบขาลงของสัญญาณ Q8 (74HC4040) จะมีสัญญาณ DREQ ร้องขอไปยัง 8237A-5 (DMA Controller) เพื่อนำข้อมูลขนาด 8 บิทเข้ามาเก็บใน 74LS165 และเมื่อสัญญาณ Q8 เปลี่ยนจาก Low เป็น High ข้อมูลจาก 74LS165 จะถูกปล่อยออกมาทีละบิทตามช่วงเวลาของ $BCIK_R$ นำมาป้อนเข้าที่ขา D_R ของ ETC 5057 แล้วจะทำการแปลงเป็นสัญญาณเสียงส่งออกไป



รูปที่ 7.11 WRITE CYCLE



รูปที่ 7.12 READ CYCLE



รูปที่ 7.13 วงจรแปลงสัญญาณเสียง

บทที่ 8

ผลการทดลองและสรุป

ผลการทดลอง

การทดลองในส่วนซิงเกิลบอร์ด Z280 และ การต่อเชื่อมกับฮาร์ดดิสต์

การทดลองเริ่มต้นจากทดลองใช้ไมโครโปรเซสเซอร์ Z280 มาสร้างเป็นซิงเกิลบอร์ด ทำให้สามารถใช้งานไมโครโปรเซสเซอร์ Z280 ร่วมกับสแตติกแรมได้ ใช้งาน I/O ของ Z280 ได้ ใช้งานส่วนจัดการหน่วยความจำของ Z280 ได้ รวมทั้งใช้อุปกรณ์ต่างๆ ของ Z280 ได้ เช่น DMA Counter/Timer รายละเอียดของวงจรที่ทดลองสามารถดูได้จากบท 2 การทดลองใช้งาน Z280

และต่อมาได้ทดลองใช้ซิงเกิลบอร์ด Z280 ทำการติดต่อกับฮาร์ดดิสต์แบบ IDE โดยใช้วงจรเหมือนกับวงจรที่จะใช้ในโครงงานเครื่องบันทึกและตอบรับโทรศัพท์ แต่เนื่องจากซิงเกิลบอร์ดที่ออกแบบมาในตอนแรกไม่ได้ออกมาเผื่อการใช้งานกับอุปกรณ์ภายนอก ดังนั้นจึงต้องมีการต่อวงจรเพิ่มเติมในส่วนควบคุมการอ่าน เขียน เล็กน้อย (ดูรูป 8.1 ประกอบ)

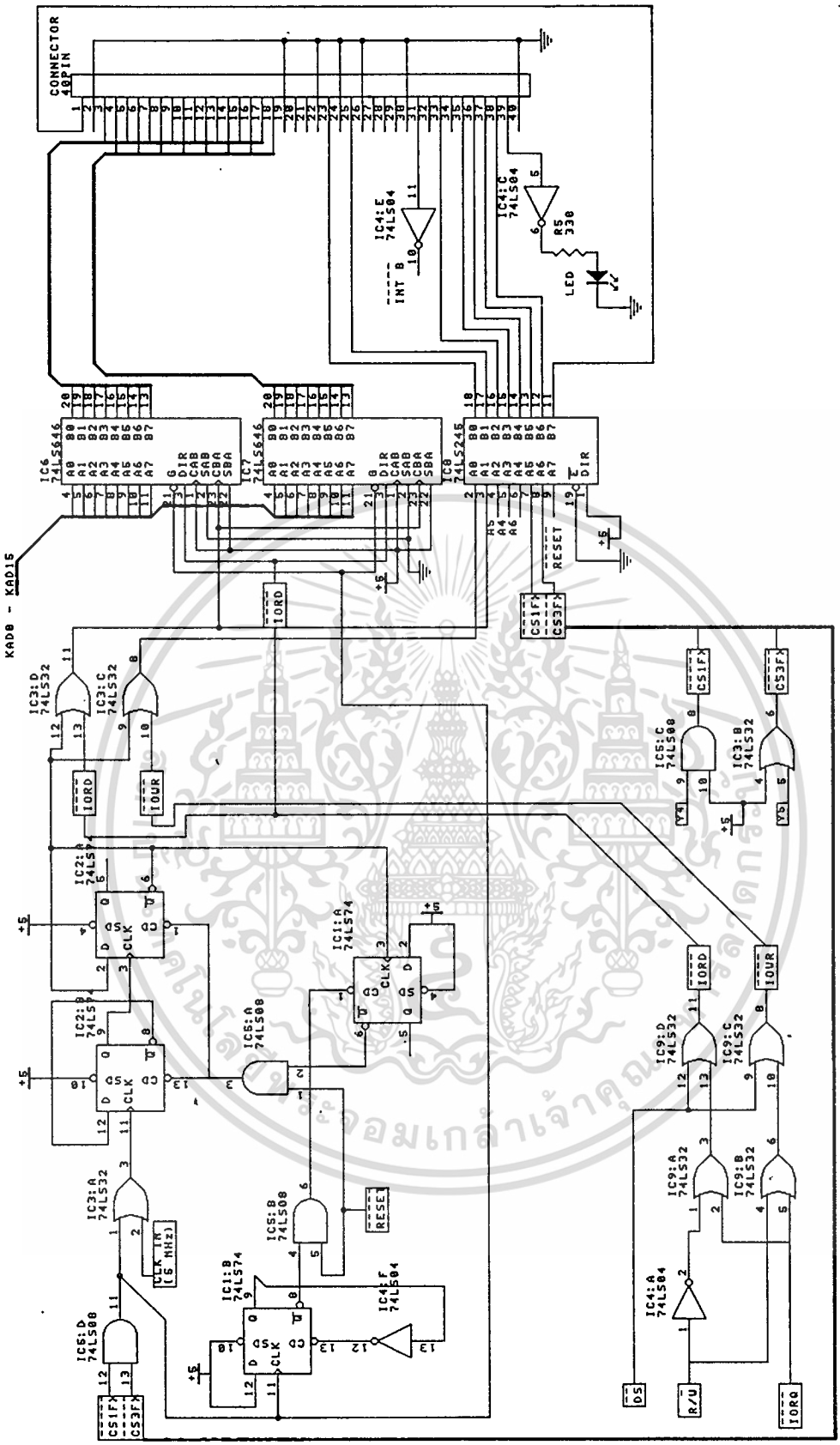
การทดลองจะทดลองติดต่อสื่อสารกับฮาร์ดดิสต์ โดยทำการ ส่งค่าสั่งไปที่รีจีสเตอร์คำสั่ง อ่านข้อมูลจากรีจีสเตอร์สถานะ ควบคุมสั่งการฮาร์ดดิสต์ในการเขียนและอ่านข้อมูลในแต่ละเซกเตอร์ได้ รวมทั้งทดลองให้ซิงเกิลบอร์ด Z280 อ่านข้อมูลในเซกเตอร์ที่ต่อเนื่องกันของฮาร์ดดิสต์ ซึ่งได้จัดเตรียมแฟ้มข้อมูลเสียงไว้ในฮาร์ดดิสต์ก่อนแล้ว และให้ข้อมูลที่อ่านได้มาออกทางลำโพงของซิงเกิลบอร์ด Z280 ซึ่งจะสามารถรับฟังเป็นเสียงตามแฟ้มข้อมูลเสียงที่เตรียมไว้ในฮาร์ดดิสต์ได้

การทดลองในส่วนวงจรโทรศัพท์

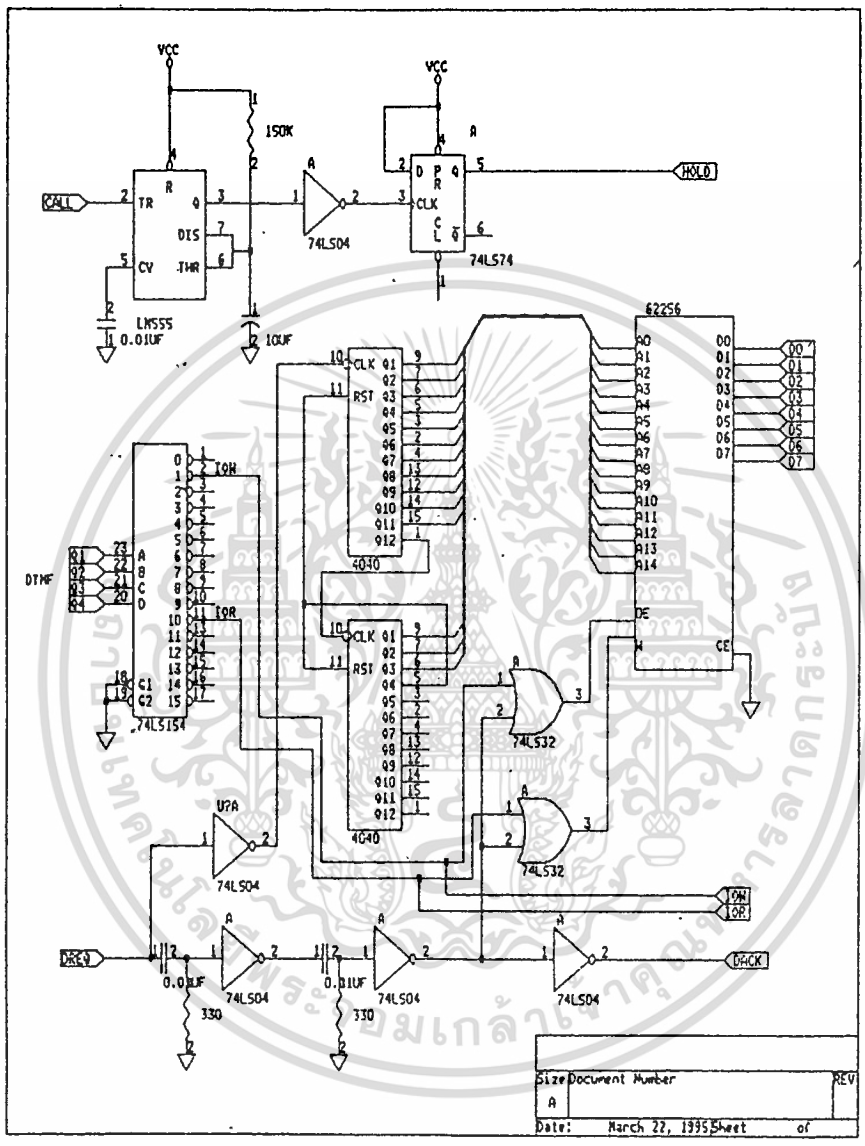
ในส่วนของวงจรอินเตอร์เฟสกับโทรศัพท์ได้ทำการป้อนสัญญาณซายน์ 0.5 V_{PP} ช่วง 300 - 4000 Hz ซึ่งเป็นความถี่ในช่วงเสียงพูดที่ใช้ในโทรศัพท์ ตัวไอซี Codec สามารถแปลงสัญญาณซายน์เป็นสัญญาณดิจิตอล และแปลงสัญญาณดิจิตอลกลับเป็นสัญญาณซายน์ได้ และทดลองการตรวจจับสัญญาณการวางหูโทรศัพท์โดยใช้ไอซี LM567 และแปลงสัญญาณ DTMF จากหน้าปัดโทรศัพท์ได้

รวมทั้งยังทดลองนำวงจรของเครื่องบันทึกและตอบรับโทรศัพท์ในส่วนของโทรศัพท์และการแปลงสัญญาณเสียง 1 คู่สายมาทดลองเก็บเสียงที่พูดจากโทรศัพท์แปลงเป็นข้อมูลแล้วเก็บลงในหน่วยความจำ และอ่านข้อมูลจากหน่วยความจำแปลงกลับเป็นเสียงแล้วส่งออกทางโทรศัพท์ได้ โดยใช้สัญญาณ DTMF จากโทรศัพท์เป็นสัญญาณควบคุมการทำงาน

เนื่องจากเวลาอันจำกัด ทำให้ไม่สามารถทดสอบและแก้ไขวงจรของ Z280 ที่ต่อเชื่อมกับไดนามิกแรมและรองรับการทำงานของ DMA ภายนอกได้ทัน ซึ่งวงจรในส่วน DMA นี้จะไปควบคุมการทำงานในการเก็บเสียงของเครื่องบันทึกและตอบรับโทรศัพท์ทั้งหมด ทำให้ไม่สามารถทดสอบการทำงานของระบบทั้งหมดได้



รูปที่ 8.1 วงจรเชื่อมต่อฮาร์ดดิสก์กับบอร์ด Z280



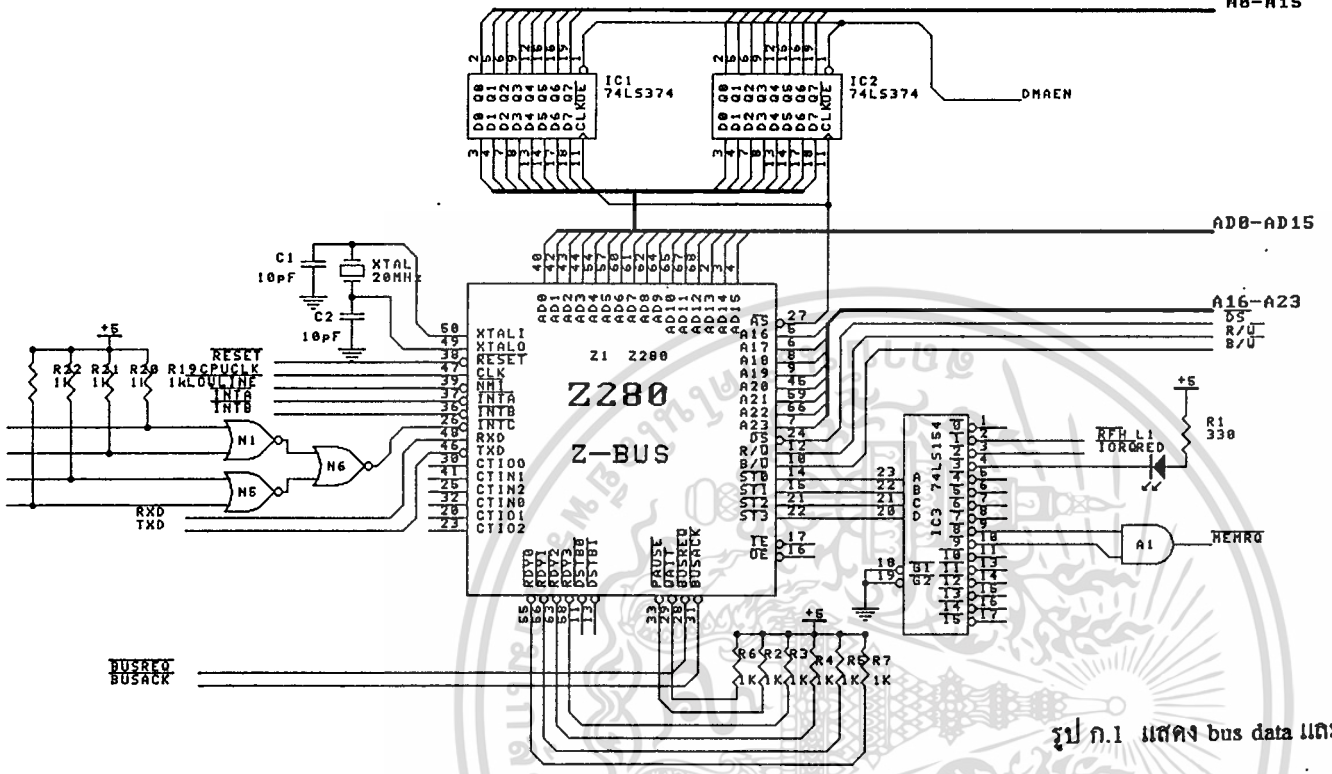
รูปที่ 8.2 วงจรทดลองในส่วนโทรศัพท์

สรุป

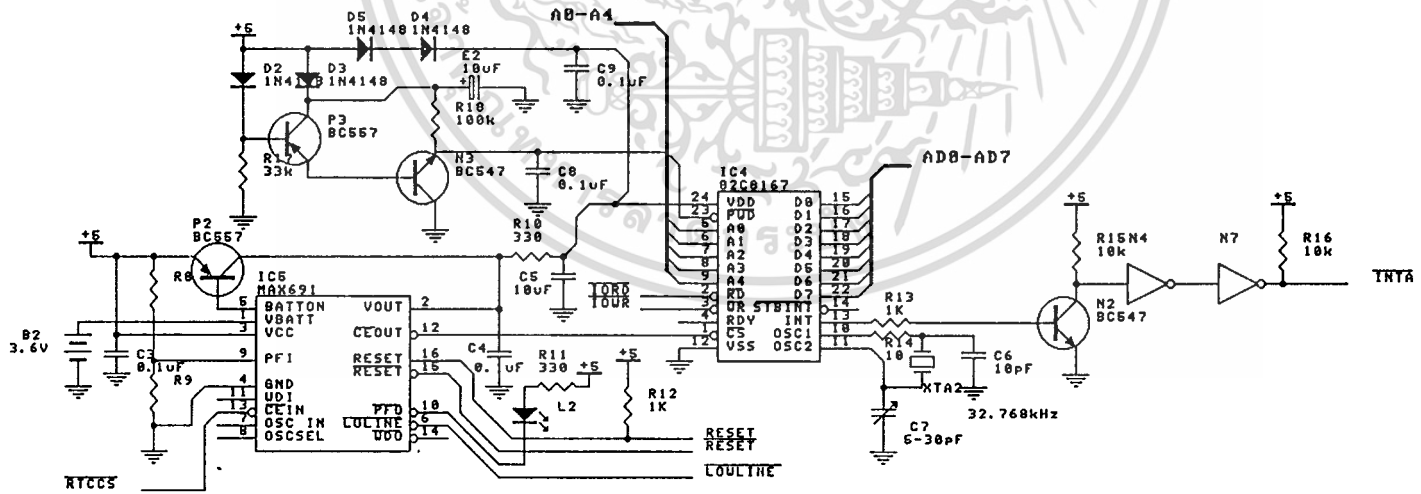
เนื่องจากโครงงานเครื่องตอบรับและบันทึกโทรศัพท์อัตโนมัตินี้เป็นโครงงานที่มีขนาดใหญ่ และมีความซับซ้อนมาก ใช้เวลาในการออกแบบวงจรทั้งหมดและการออกแบบแผ่นวงจรพิมพ์มาก ประกอบกับ ความยากลำบากในการค้นหาข้อมูลซึ่งต้องพึ่งพาแหล่งข้อมูลจากต่างประเทศเป็นจำนวนมาก รวมทั้งการประสานงานที่ไม่ดี และประสบการณ์อันน้อยของผู้จัดทำโครงงานนี้ จึงทำให้โครงงานนี้ไม่สามารถแล้วเสร็จได้ทันตามกำหนดเวลา

แต่จากข้อมูลและส่วนของโครงงานที่ได้ทำมาแล้ว เช่น ส่วนเชื่อมต่อกับฮาร์ดดิสต์ หรือส่วนโทรศัพท์ ก็สามารถนำไปใช้ประโยชน์ในการสร้างโครงงานนี้ต่อไป และอาจจะเป็นประโยชน์ต่อโครงงานอื่นได้ในอนาคต

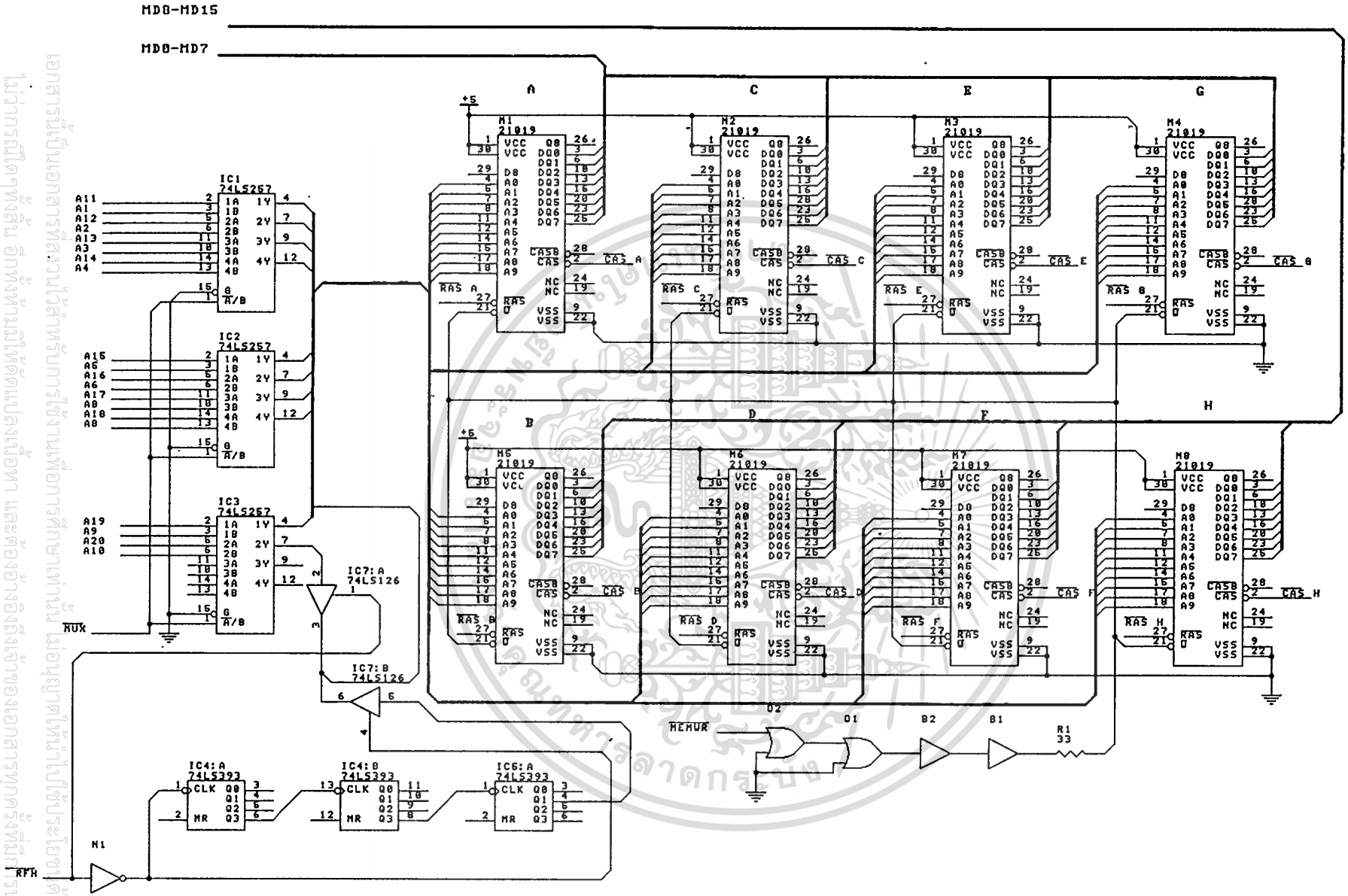
ภาคผนวก



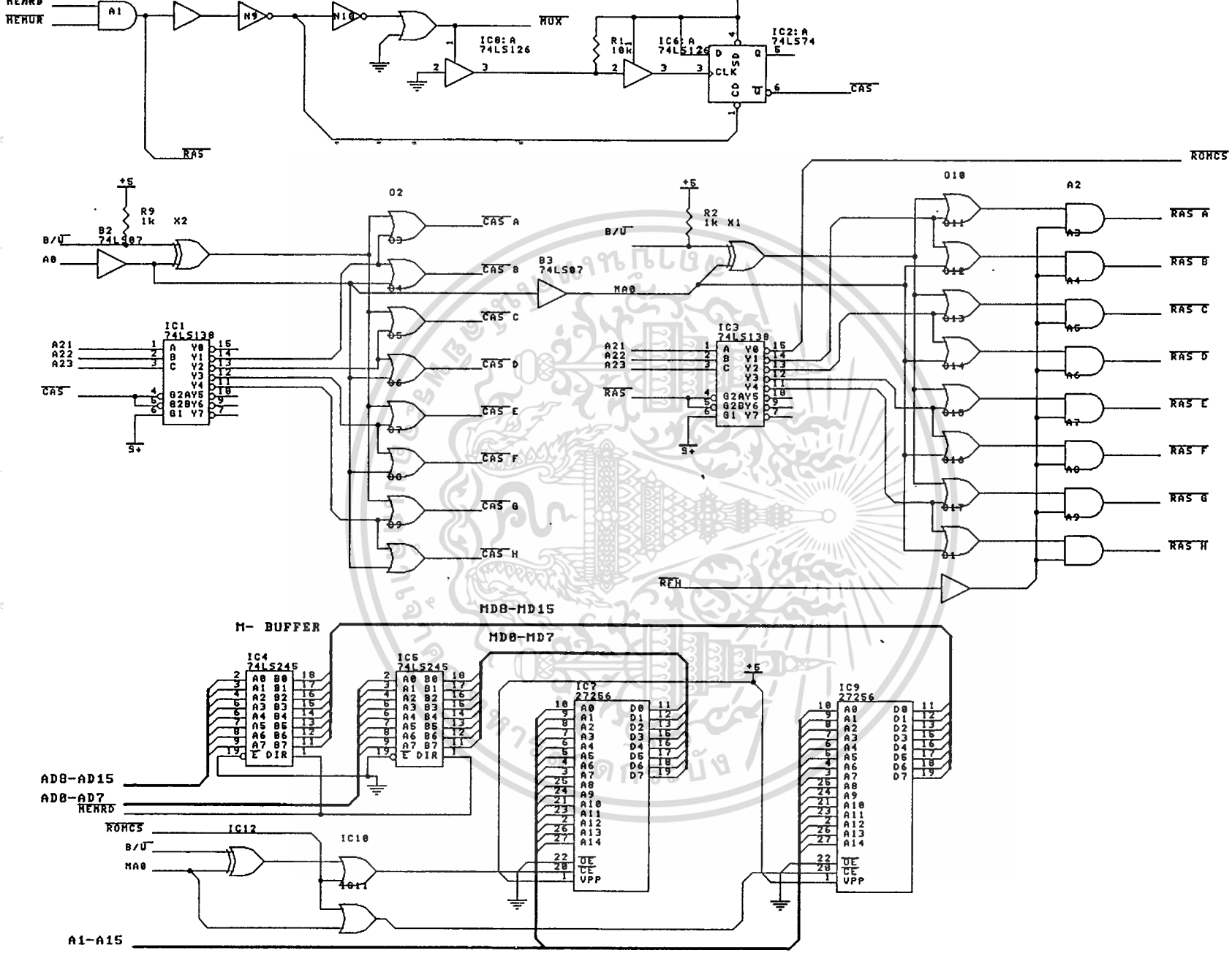
รูป ก.1 แสดง bus data และ Control bus ของระบบ



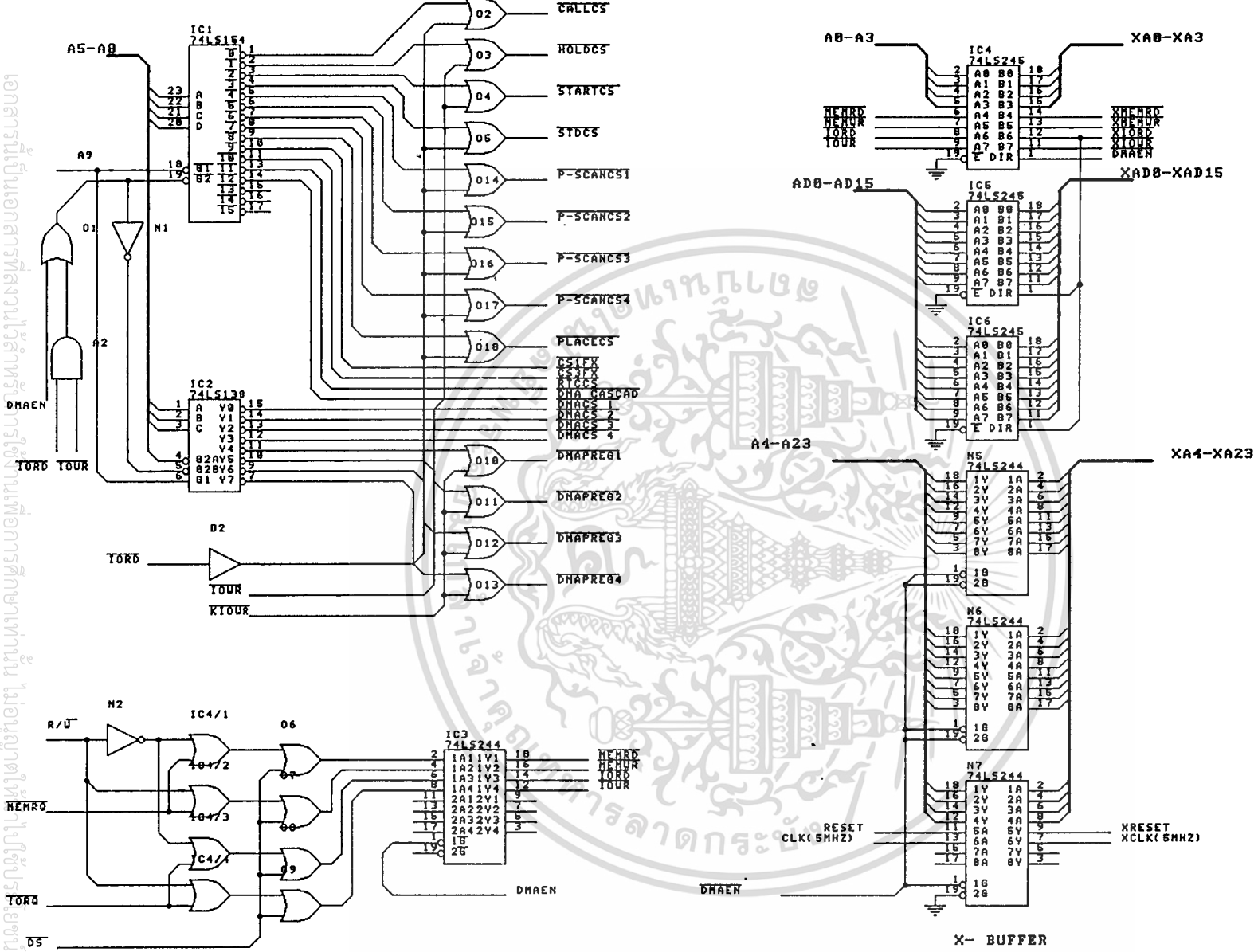
รูป ก.2 การต่อไอซี MAX691 และเมอร์ 82C8167



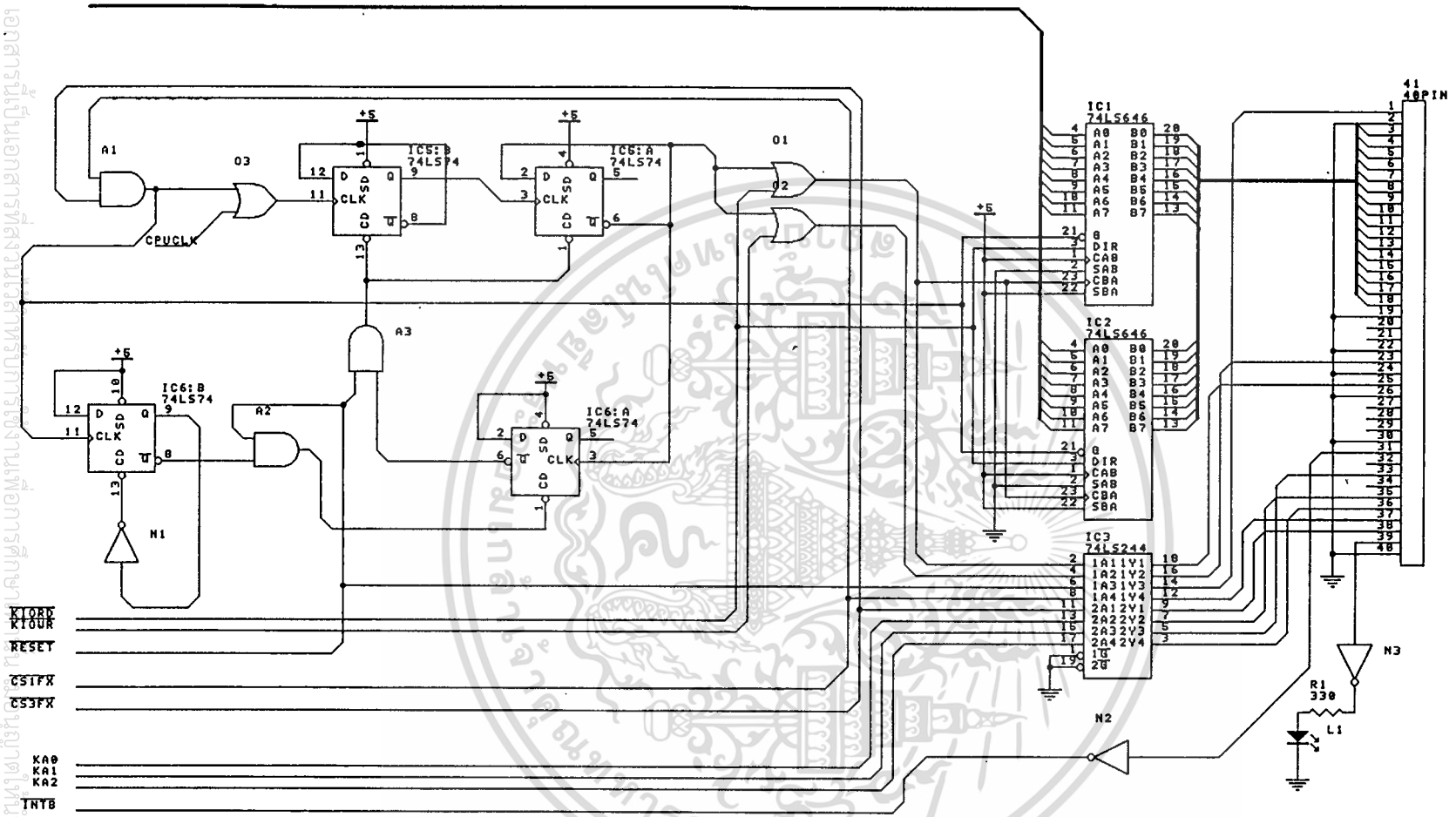
รูป ก.3 การเชื่อมต่อไดนามิกแรม



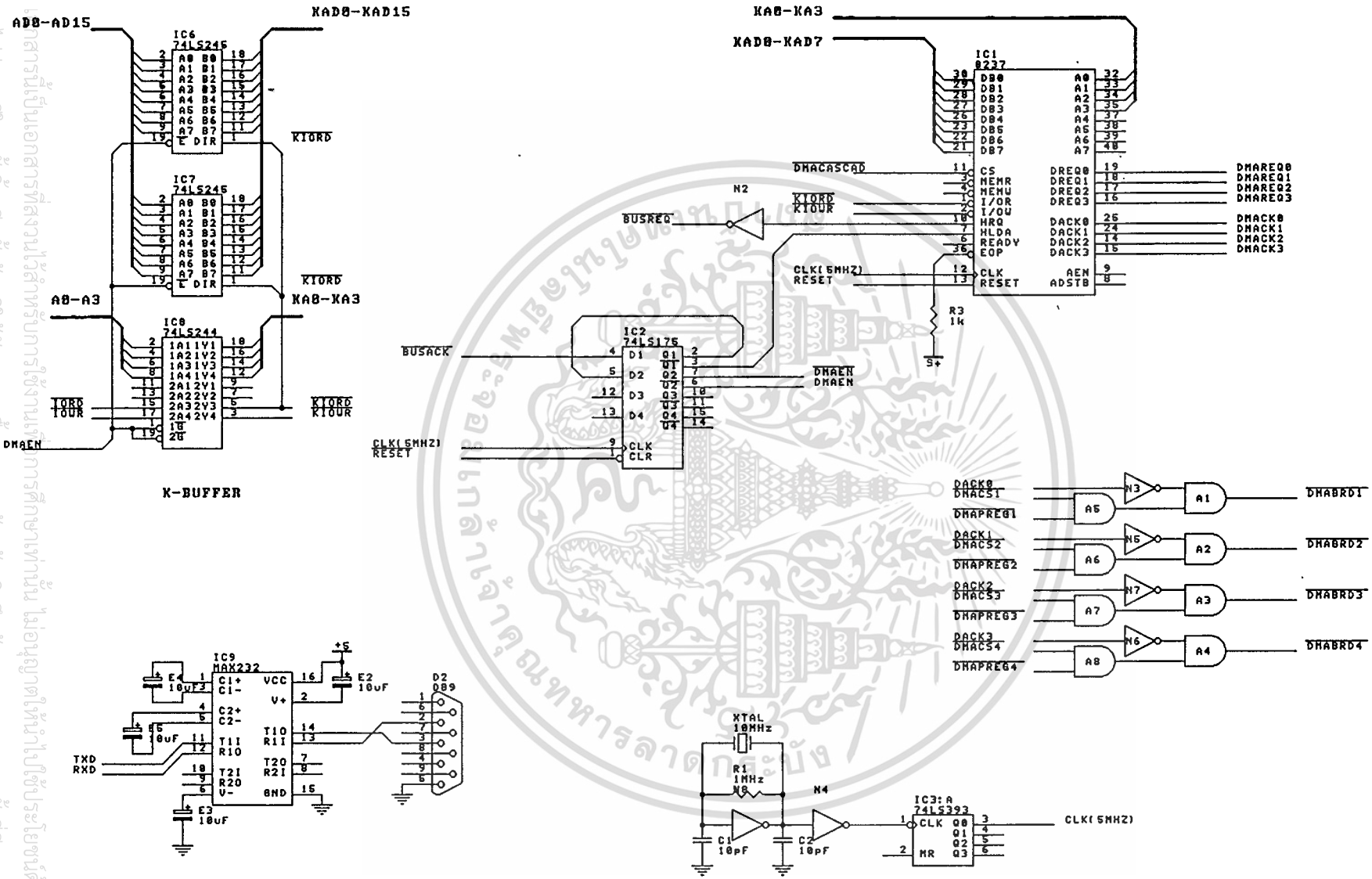
รูป ก.4 การถอดรหัสเลือกแรมในระบบ



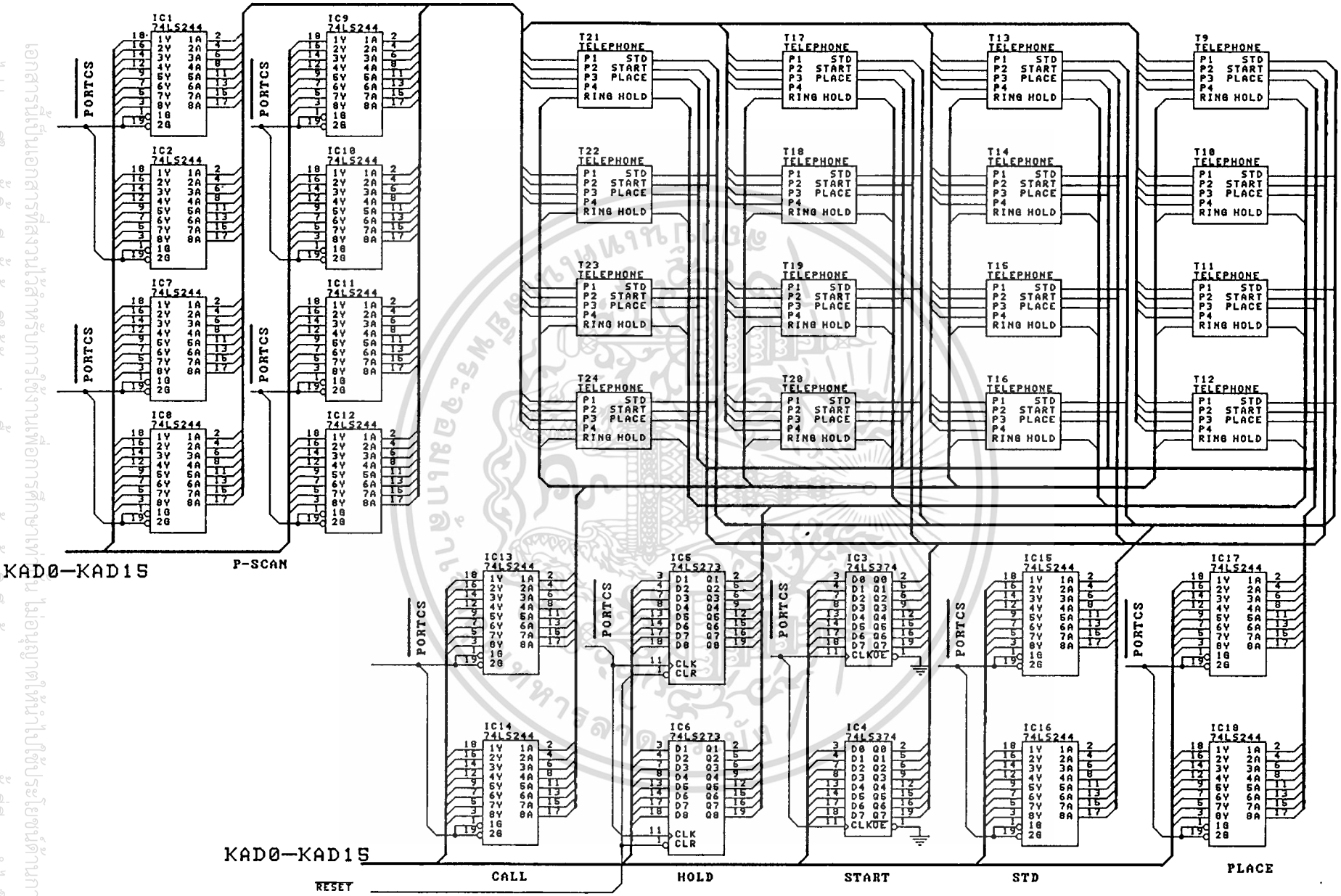
รูป ก.5 วงจรถอดรหัสเลือกพอร์ทในระบบ



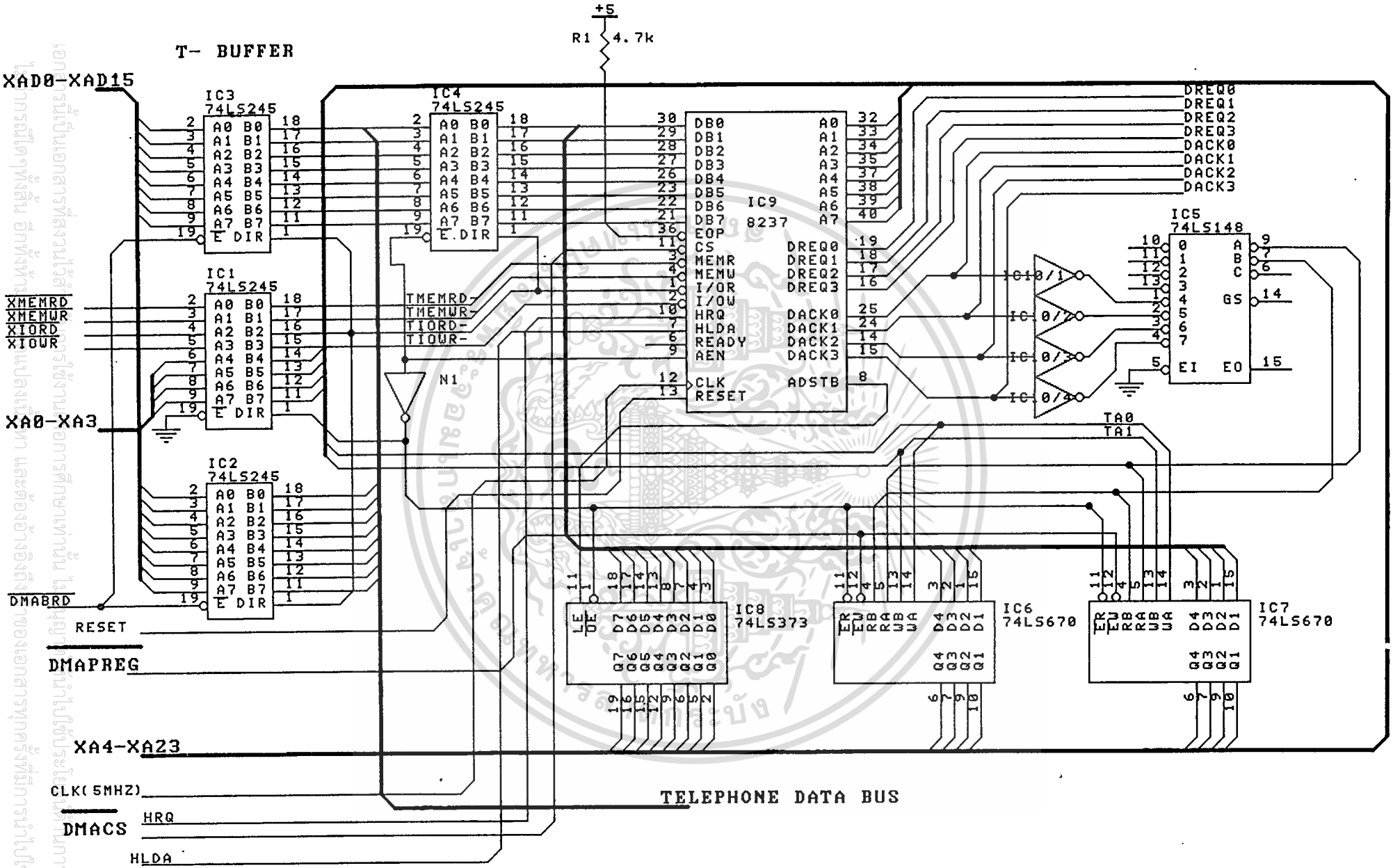
รูป ก.6 การเชื่อมต่อกับฮาร์ดดิสต์



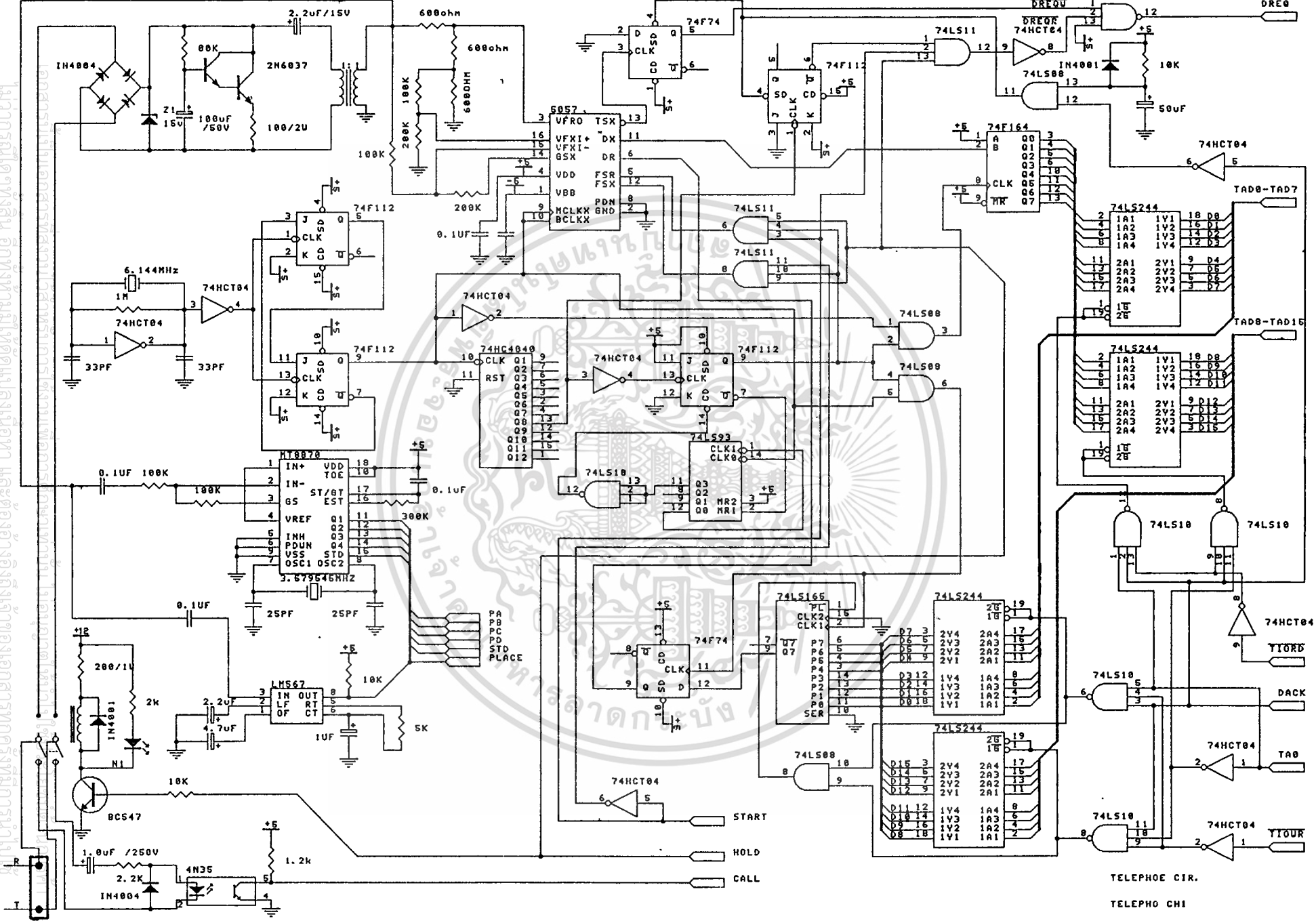
รูป ก.7 วงจรมาสเตอร์ DMA และวงจร enable / disable buffer



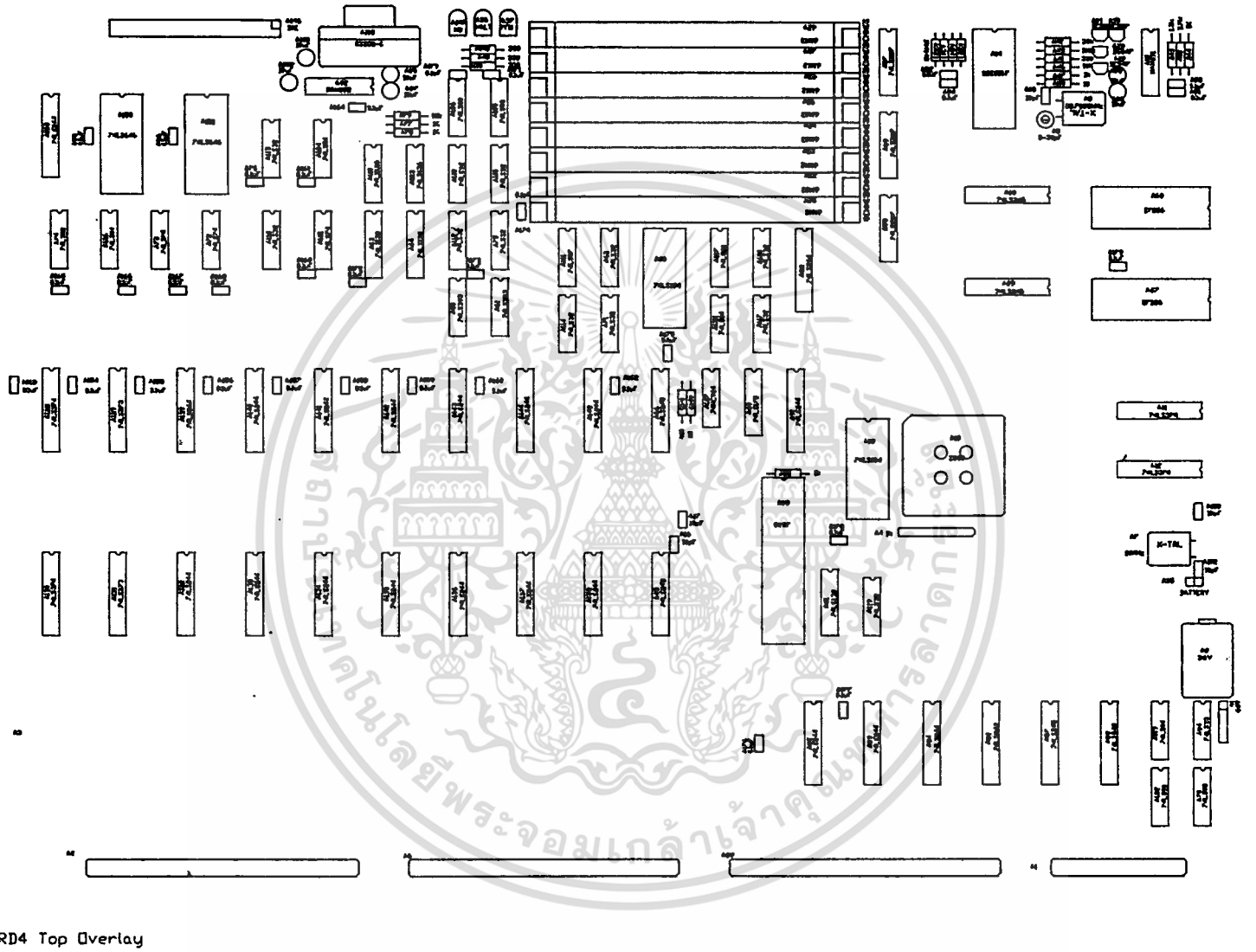
รูป ก.8 วงจร SCAN ช่องสัญญาณโทรศัพท์



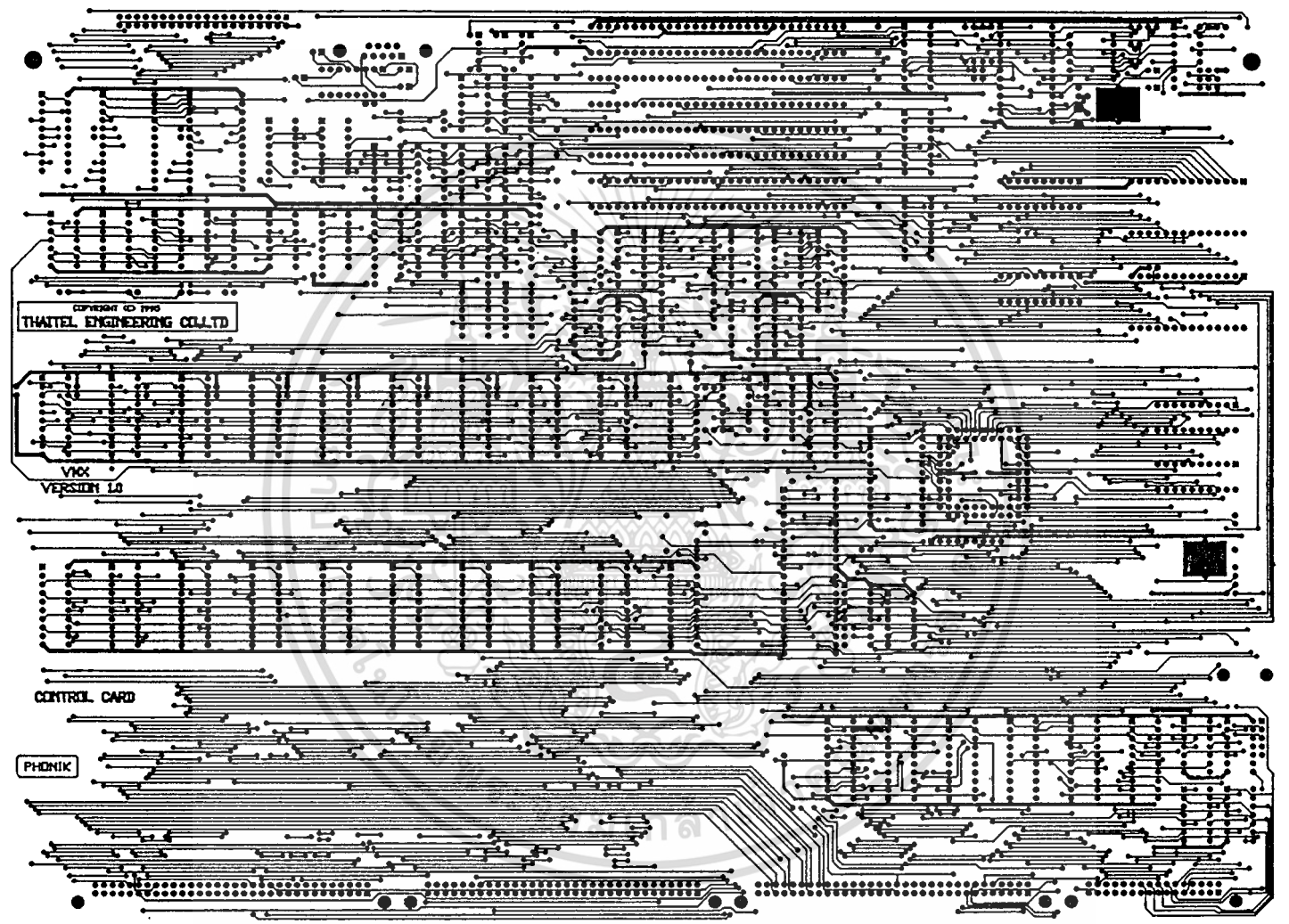
รูป ก.9 วงจร DMA Controller



รูป ก.10 วงจรส่วนเชื่อมต่อกับโทรศัพท์ และแปลงสัญญาณเสียง

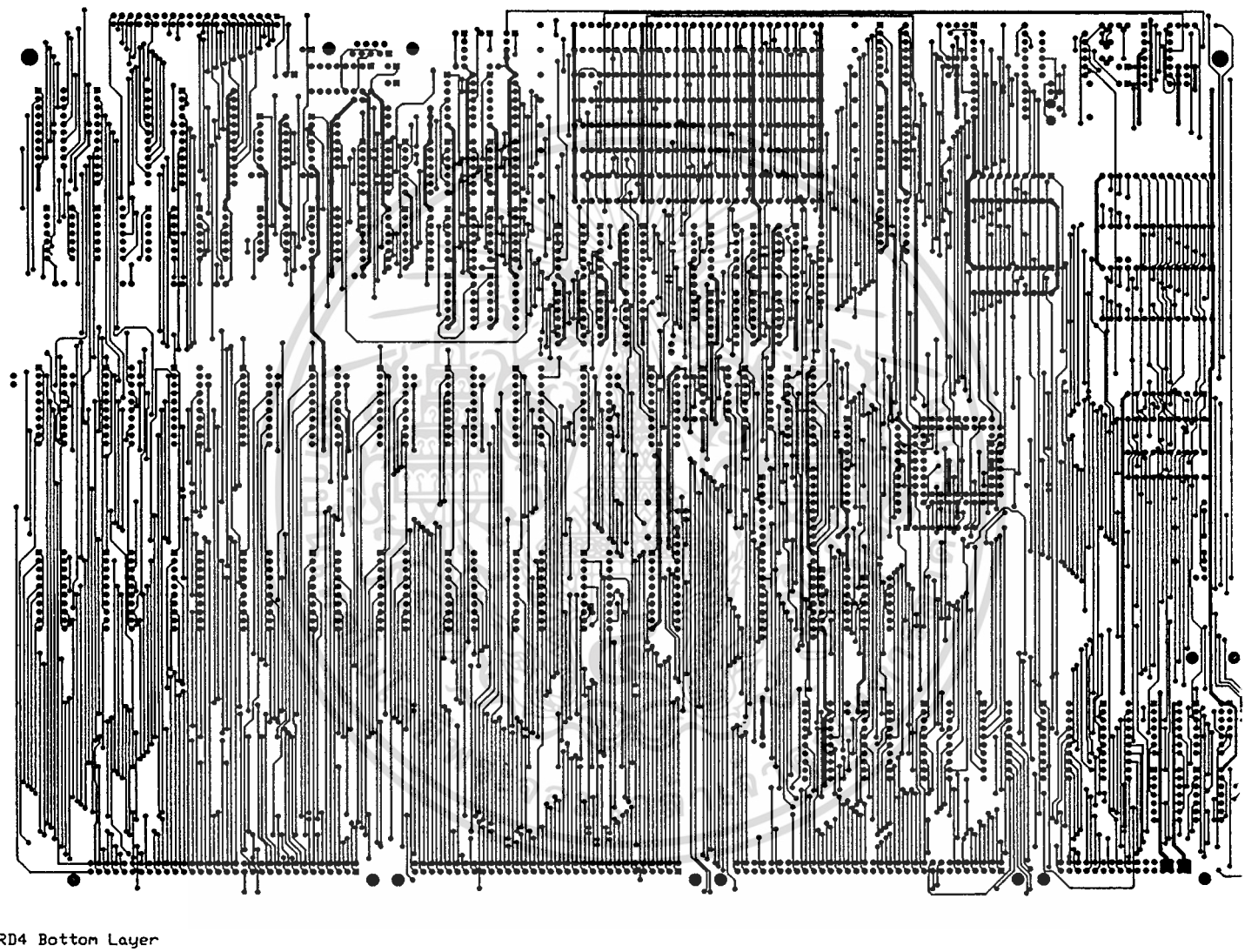


รูปที่ ข.1 รูปแสดงการลงอุปกรณ์แผ่นวงจรคอนโทรลการ์ด

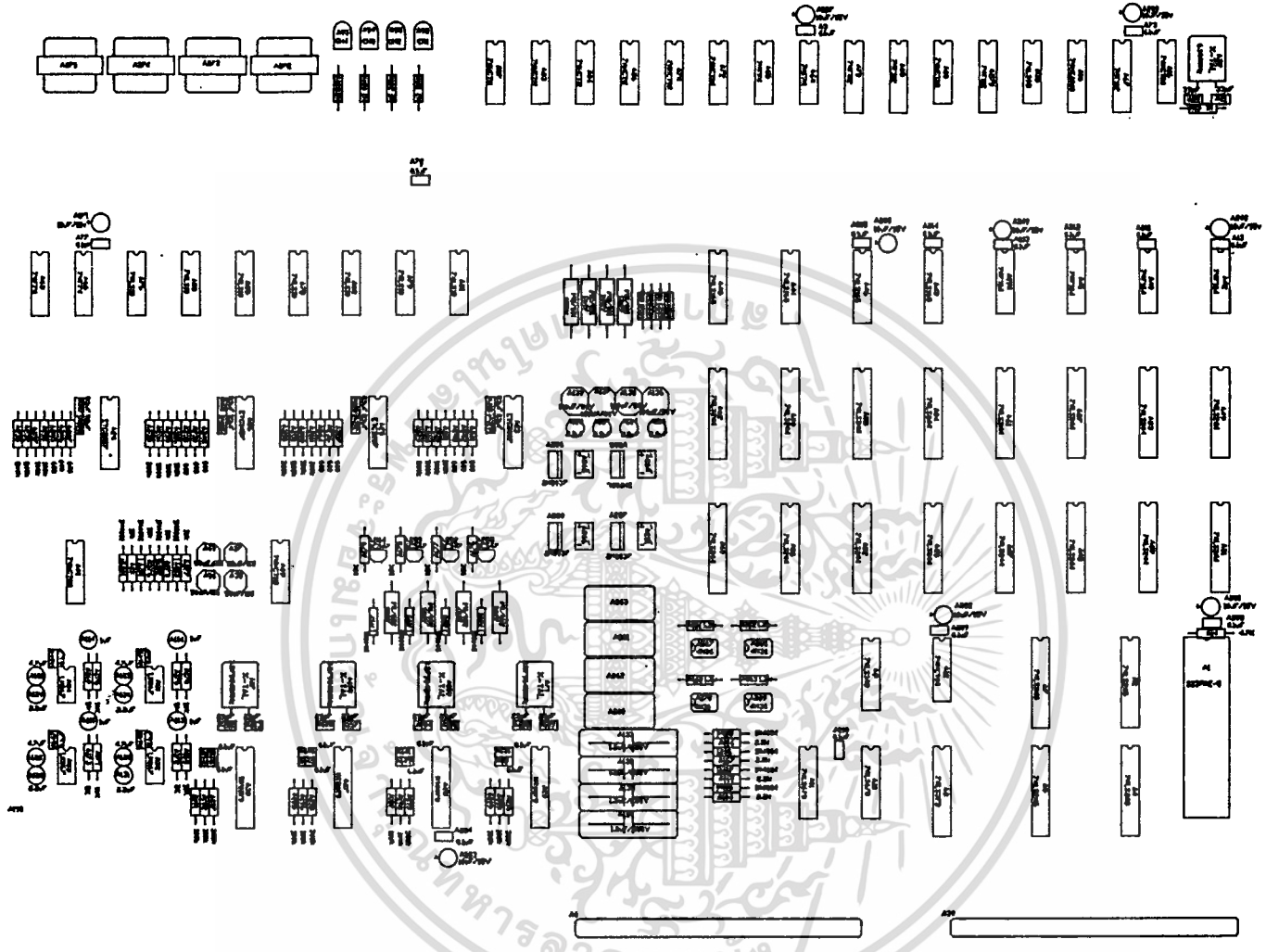


RD4 Top Layer

รูปที่ ข.2 ลายวงจรพิมพ์ด้านบนของคอนโทรลการ์ด

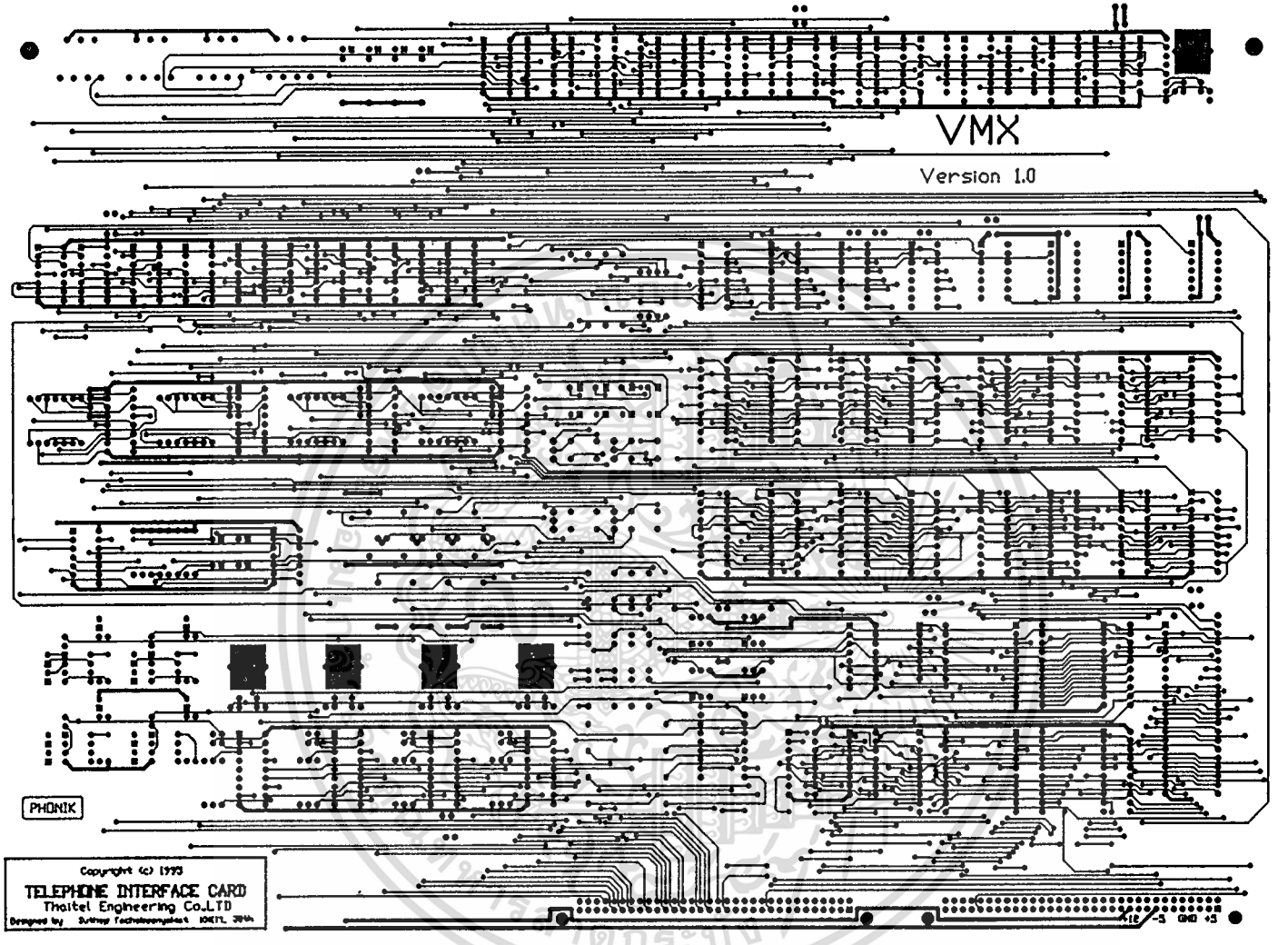


รูปที่ ข.3 ลายวงจรพิมพ์ด้านล่างของคอนโทรลการ์ด



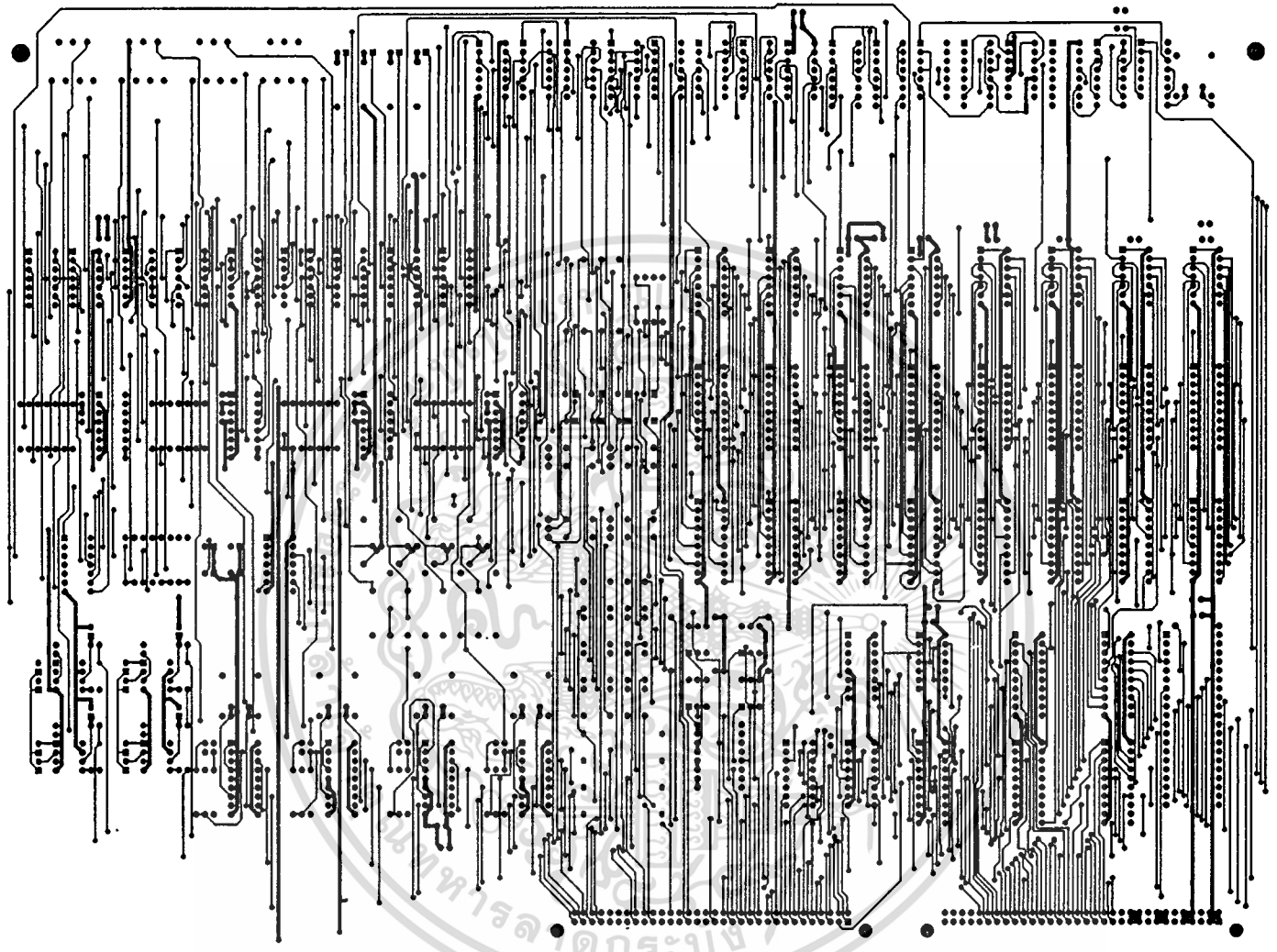
I-CARD2 Top Overlay

รูปที่ ข.4 รูปแสดงการลงอุปกรณ์แผ่นวงจรส่วนเชื่อมต่อโทรศัพท์



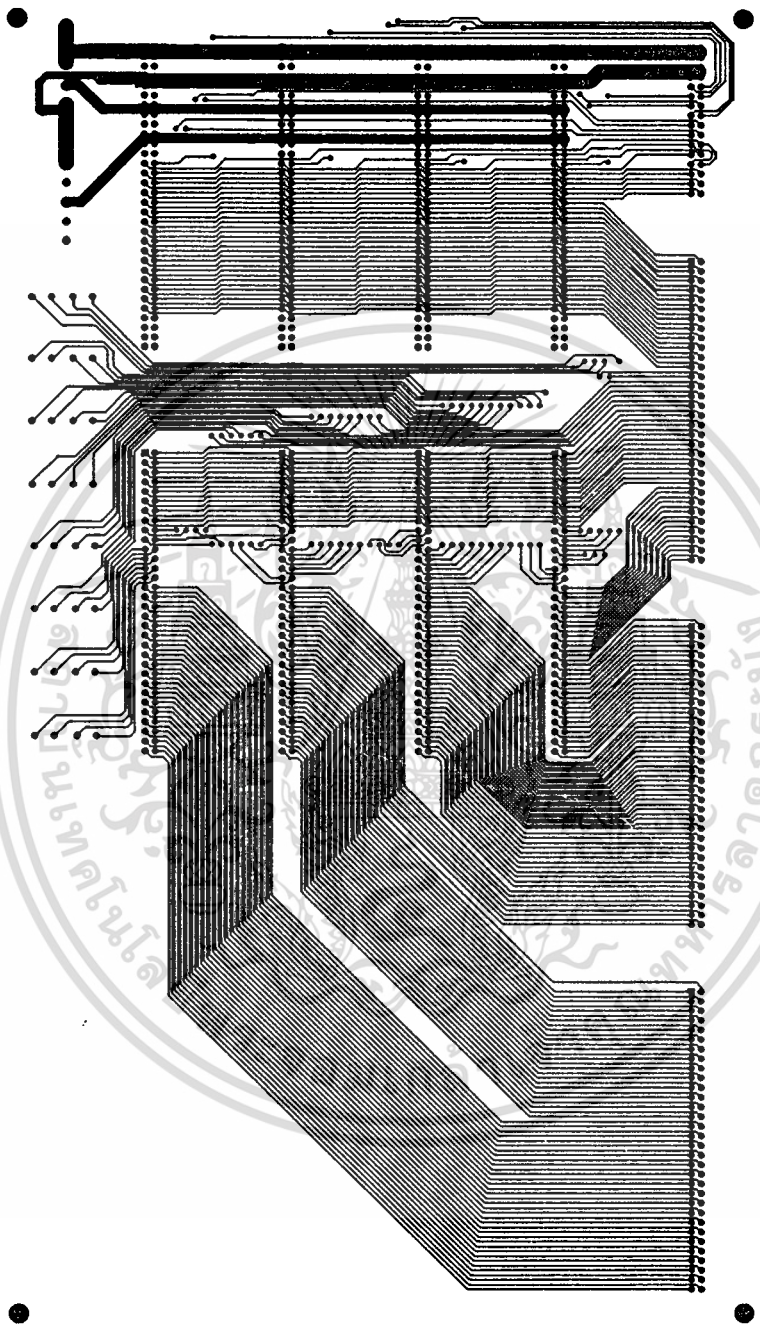
รูปที่ ข.5 ลายวงจรพิมพ์ด้านบนของแผ่นวงจรส่วนเชื่อมต่อโทรศัพท์

I-CARD2 Top Layer



I-CARD2 Bottom Layer

รูปที่ ข.6 ลายวงจรพิมพ์ด้านล่างของแผ่นวงจรส่วนเชื่อมต่อโทรศัพท์



I-CARD5 Bottom Layer

รูปที่ ข.7 ลายวงจรพิมพ์ของแผ่นเชื่อมต่อ

กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบพระคุณ ดร.กิติพล ชิตสกุล ที่ให้คำแนะนำและการสนับสนุน และขอขอบพระคุณองค์กรต่างๆ ที่สนับสนุนข้อมูลทางเทคนิค และเพื่อนๆ ที่ช่วยสนับสนุนในการหาข้อมูลด้านต่างๆ

โครงงานนี้หากมีข้อบกพร่องหรือขัดข้องประการใด คณะผู้จัดทำขอน้อมรับเพื่อนำไปเป็นข้อมูลสำหรับปรับปรุงแก้ไขต่อไป และหวังว่าโครงงานและรายงานฉบับนี้คงจะมีประโยชน์ต่อไป ไม่มากก็น้อย

ขอขอบคุณ

คณะผู้จัดทำ

13 มีนาคม 2538

บรรณานุกรม

1. Zilog Co. Ltd. , "Zilog Intellegent peripheral controllers"
2. UMC , "Microcomputer & Memory ICs Data Book 1989-1990"
3. บริษัท ซีเอ็ดยูเคชั่น จำกัด, "หนังสือคู่มือ/เทียบเบอร์ไอซี TTL"
4. สุชิน จำจด , "วิศวกรรมโทรศัพท์" , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
5. Byte , "The IDE Hard Disk Drive Interface" , March 1991
6. ธานินทร์ ถาวรศาสนวงศ์ , ทินกร ดุ๊ก , "การอินเตอร์เฟส IBM PC" , Physics center
7. Mark Minasi , "คู่มือซุบชีวิตฮาร์ดดิสต์และกู้ข้อมูล" , บริษัท ซีเอ็ดยูเคชั่น จำกัด

54148/74148 8 Data Line to 3-Line Binary (Octal) Priority Encoder

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.											SN74LS148		N			SN74148		J	N	W					
FAIRCHILD																									
MOTOROLA																									
N. S. C.																DM54148									
																DM74148									
PHILIPS																									
SIGNETICS																N74148									
																54148									
																74148									
SIEMENS																									
FUJITSU																									
HITACHI											HD74LS148		P			HD74148		D	P						
MITSUBISHI											M74LS148					M74148			P						
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS148/SN74LS148

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C	to	125°C
Input voltage	7V		SN74LS	0°C	to	70°C
		Storage temperature range		−65°C	to	150°C

recommended operating conditions

	SN54LS148			SN74LS148			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

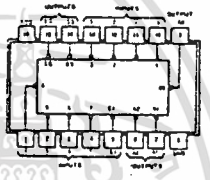
electrical characteristics over recommended operating free-air temperature range

PARAMETER*		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN,	I_I = −18mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IL} = 0.8V,	V_{IH} = 2V, I_{OH} = −400μA	2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IL} = 0.8V,	V_{IH} = 2V, I_{OL} = 8mA		0.35	0.5	V
I_I	Input current at maximum input voltage	V_{CC} = MAX,	V_I = 7V			0.2	mA
I_{IH}	High-level input current: 3 input	V_{CC} = MAX,	V_I = 2.7V			20	μA
	Any input except 3					20	
I_{IL}	Low-level input current: 3 input	V_{CC} = MAX,	V_I = 0.4V			−0.4	mA
	Any input except 3					−0.8	
I_{OS}	Short-circuit output current*	V_{CC} = MAX		−20		−100	mA
I_{CC}	Supply current:	V_{CC} = MAX, See Note 2	Condition 1		2	20	mA
			Condition 2		3	17	mA
t_{PLH}	from 0 thru 7 to output A0, A1, or A2	WAVEFORM	In-phase output	V_{CC} = 5V, T_A = 25°C, C_L = 15pF, R_L = 2kΩ	4	18	ns
t_{PHL}					5	25	
t_{PLH}	from 0 thru 7 to output A0, A1, or A2		Out-of-phase output		23	36	ns
t_{PHL}					5	29	
t_{PLH}	from 0 thru 7 to output EO		Out-of-phase output		15	18	ns
t_{PHL}					25	40	
t_{PLH}	from 0 thru 7 to output GS		In-phase output		25	55	ns
t_{PHL}					9	21	
t_{PLH}	from EI to output A0, A1, or A2		In-phase output		5	25	ns
t_{PHL}					12	25	
t_{PLH}	from EI to output GS		In-phase output		12	17	ns
t_{PHL}					4	36	
t_{PLH}	from EI to output EO		In-phase output		12	21	ns
t_{PHL}					23	35	

NOTES: 1. This is the voltage between two emitters of a multiple-emitter transistor. For SN54148/SN74148 circuits, this rating applies between any two of the eight data lines, 0 through 7.

2. I_{CC} (condition 1) is measured with inputs 7 and EI grounded, other inputs and outputs open; I_{CC} (condition 2) is measured with all inputs and outputs open.

Pin Assignment (Top View)



positive logic:
see function table

NC-No internal connection

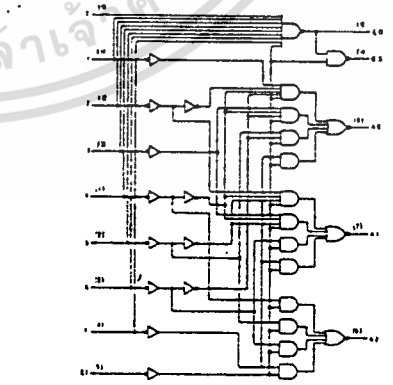
Function Table

'148

INPUTS								OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	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 = high logic level, L = low logic level, X = irrelevant

Functional Block Diagram



'148 8-LINE-TO-3LINE PRIORITY ENCODER

54164/74164 8-Parallel-Out Serial Shift Register

	Schottky TTL		High-Speed TTL		Low-Power Schottky TTL		Standard TTL		Low-Power TTL	
	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF
T.I.					SN54LS164	J W	SN54164	J W	SN54L164	J W
					SN74LS164	J N	SN74164	J N	SN74L164	J N
FAIRCHILD					9N54LS164 / 9N74LS164	D F	FM54164, FM93164	D F		
					54LS164 / 74LS164	D F	FC74164 / FC93164	D F		
MOTOROLA					SN74LS164	P	MC74164	P		
N.S.C.					DM74LS164	N			DM54L164	J W
					DM54LS164	J	DM74164	N J	DM74L164	J N
PHILIPS					N74LS164	N	N74164	N		
SIGNETICS							S54164	F D A		W
					N74LS164	A	N74164	F D A		
SIEMENS							FLJ441	D		
FUJITSU					74LS164	M				
HITACHI					HD74LS164	P	HD74164	DIP		
MITSUBISHI					M74LS164	P	M53364	P		
NEC					μPB74LS	C	μPB2164	D		
TOSHIBA							TD3503A	P		
AMD					Am74LS164					

Electrical Characteristics SN54LS164/SN74LS164

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7 V	Operating free-air temperature range	SN54'	−55°C to 125°C
Input voltage	7 V		SN74'	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommended operating conditions

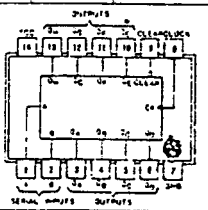
	SN54LS164			SN74LS164			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Clock frequency, f_{clock}	0		25	0		25	MHz
Width of clock or clear input pulse, t_w	20			20			ns
Data setup time, t_{setup}	15			15			ns
Data hold time, t_{hold}	5			5			ns
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN,	I_I = −18mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V,	I_{OH} = −400μA	2.7	3.5		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V,	I_{OL} = 8mA		0.35	0.5	V
I_I	Input current at maximum input voltage	V_{CC} = MAX,	V_I = 7 V			0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX,	V_I = 2.7 V			20	μA
I_{IL}	Low-level input current	V_{CC} = MAX,	V_I = 0.4 V			−0.2	mA
I_{OS}	Short-circuit output current*	V_{CC} = MAX	SN54LS	−20		−100	mA
			SN74LS	−20		−100	
I_{CC}	Supply current*	V_{CC} = MAX, See Note 1			16	27	mA
f_{max}	Maximum clock frequency	V_{CC} = 5 V, T_A = 25°C, R_L = 2kΩ	C_L = 15pF	25	36		MHz
t_{PHL}	Propagation delay time, high-to-low level Q outputs from clear input		C_L = 15pF		24	36	ns
t_{PLH}	Propagation delay time, low-to-high level Q outputs from clock input		C_L = 15pF		17	27	ns
t_{PHL}	Propagation delay time, high-to-low level Q outputs from clock input		C_L = 15pF		21	32	ns

Pin Assignment (Top View)

①



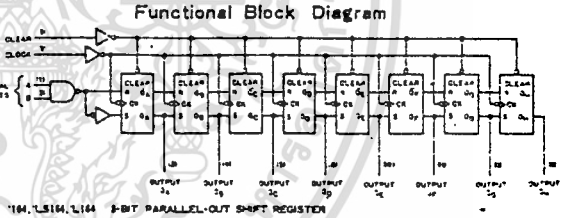
positive logic: see function table

Function Table

'164, 'LS164, 'L164 (see Note 2)

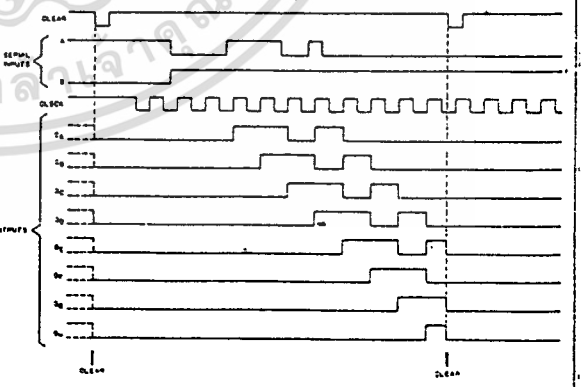
INPUTS				OUTPUTS			
CLEAR	CLOCK	A	B	Q_A	Q_B	···	Q_H
L	X	X	X	L	L		L
H	L	X	X	Q_{A0}	Q_{B0}		Q_{H0}
H	↑	H	H	H	Q_{An}		Q_{Gn}
H	↑	L	X	L	Q_{An}		Q_{Gn}
H	↑	X	L	L	Q_{An}		Q_{Gn}

Functional Block Diagram



'164, 'LS164, 'L164 8-BIT PARALLEL-OUT SHIFT REGISTER

typical clear, shift, and clear sequences



NOTES: 1. I_{CC} is measured with outputs open, serial inputs grounded, and a momentary ground, then 4.5 V, applied to clear.

2. H = high level (steady state), L = Low-level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level.
Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = the level of Q_A or Q_G before the most recent ↑ transition of the clock; indicates a one-bit shift.

54165/74165 Parallel-Load 5-Bit Shift Register

	Schottky TTL		High-Speed TTL		Low-Power Schottky TTL		Standard TTL		Low-Power TTL	
	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF	Device Type	Package C P M CF
T.I.					SN54LS165	J DI	SN54165	J W		
					SN74LS165	N DI	SN74165	N DI		
FAIRCHILD							9N54165 9N93165	J		
							9N74165, 9N93165	N DI		
MOTOROLA							MC74165	P DI		
N.S.C.									DM54L165A	J C IND FD
							DM74165	N DI	DM74L165A	J D IND FD
PHILIPS							N74165	D I		
SIGNETICS							S54165	F J B DI W		
							N74165	F J B DI		
SIEMENS							FLJ451	DI		
FUJITSU										
HITACHI										
MITSUBISHI							M53365	P DI		
NEC										
TOSHIBA										

Electrical Characteristics NS54LS165/SN74LS165

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7 V	Operating free-air temperature	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
		Storage temperature range		−55°C to 150°C

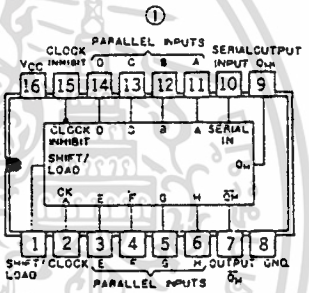
recommended operating conditions

	SN54LS165			SN74LS165			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Clock frequency, f_{clock}	0		20	0		20	MHz
Width of clock input pulse, t_w(clock)	25			25			ns
Width of load input pulse, t_w(load)	15			15			ns
Clock-enable setup time, t_{setup}	30			30			ns
Parallel input setup time, t_{setup}	10			10			ns
Serial input setup time, t_{setup}	20			20			ns
Shift setup time, t_{setup}	45			45			ns
Hold time at any input, t_{hold}	0			0			ns
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER †		TEST CONDITIONS †		MIN	TYP ‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN,	I_I = −18mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8 V,	I_{OH} = −400 μA	2.7	3.5		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8 V,	I_{OL} = 8mA		0.35	0.5	V
I_I	Input current at maximum input voltage: Shift, load input	V_{CC} = MAX,	V_I = 7V			0.3	mA
	Other inputs					0.1	
I_{IH}	High-level: Load input	V_{CC} = MAX,	V_I = 2.7V			60	μA
	Shift, load input; Other inputs					20	
I_{IL}	Low-level: Load input	V_{CC} = MAX,	V_I = 0.4 V			−1.2	mA
	Shift, load input; Other inputs					−0.4	
I_{OS}	Short-circuit output current *	V_{CC} = MAX	SN54LS	−20		−100	mA
			SN74LS	−20		−100	
I_{CC}	Supply current	V_{CC} = MAX,	See Note 2		31	36	mA
f_{max}	Maximum clock frequency			25	35		MHz
t_{PLH} / t_{PHL}	from Clear to Any output	V_{CC} = 5V, T_A = 25°C, C_L = 15 pF, R_L = 2kΩ			22	35	ns
					22	35	
t_{PLH} / t_{PHL}	from Clock to Any output				27	40	ns
					28	40	
t_{PLH} / t_{PHL}	from H to output Q_H				16	25	ns
					21	30	
t_{PLH} / t_{PHL}	from H to output $\overline{Q}_H$				21	30	ns
					16	25	

Pin Assignment (Top View)



positive logic:

Function Table

'165 (see Note 3)

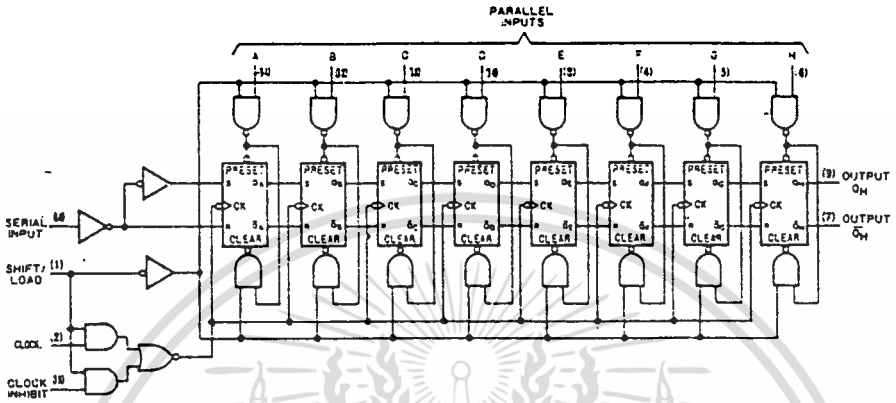
INPUTS						INTERNAL OUTPUTS		OUTPUT
SHIFT/ LOAD	CLOCK INHIBIT	CLOCK	SERIAL	PARALLEL A...H		Q_A	Q_B	Q_H
L	X	X	X	a...h		a	b	h
H	L	L	X	X		Q_{A0}	Q_{B0}	Q_{H0}
H	L	↑	H	X		H	Q_{An}	Q_{Gn}
H	L	↑	L	X		L	Q_{An}	Q_{Gn}
H	H	↑	X	X		Q_{A0}	Q_{B0}	Q_{H0}

NOTES: 1. This is the voltage between two emitters of a multiple-emitter transistor. For this circuit, this rating applies to the shift load input in conjunction with the clock clock or clock-inhibit inputs.

2. With the outputs open, clock inhibit and shift/load at 4.5V, and a clock pulse applied to the clock input, I_{CC} is measured first with the parallel inputs at 4.5V, then with the parallel inputs grounded.

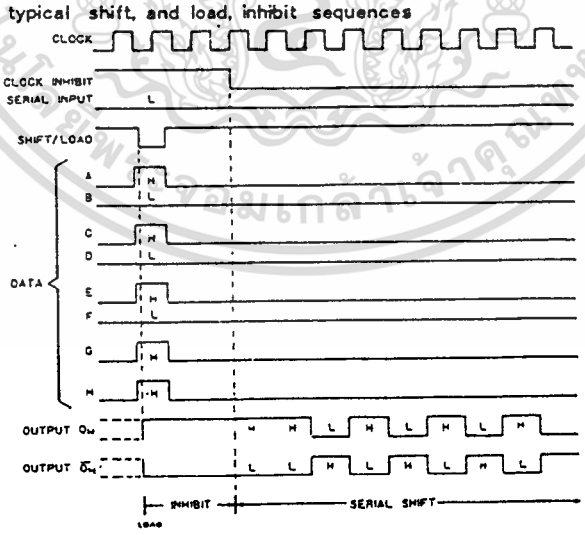
3. H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a...h = the level of steady-state input at inputs A thru H, respectively.
Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = the level of Q_A or Q_G, respectively, before the most recent ↑ transition of the clock.

Functional Block giagram



'165 PARALLEL-LOAD 8-BIT SHIFT REGISTER

typical shift, and load, inhibit sequences



54670/74670 4-by-4 Register File with 3-State Output

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF
T. I.											SN54LS670	J①			W①										
											SN74LS670	J①	N①												
FAIRCHILD											FM54LS670/FM9LS670	D①			F①										
											FJ74LS670/FC9LS670	D①	P①		D①										
MOTOROLA											SN74LS690		P①												
N. S. C.																									
PHILIPS											N74LS670		①												
SIGNETICS											N74LS670		A①												
SIEMENS																									
FUJITSU																									
HITACHI																									
MITUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS670/SN74LS670

absolute maximum rating over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS'	−55℃ to 125℃
Input voltage	7V		SN74LS'	0℃ to 70℃
Off-state output voltage	5.5V	Strange temperature range		−65℃ to 150℃

recommended operating conditions

		SN54LS670 MIN	NOM	MAX	SN74LS670 MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−1			−2.6	mA
Low-level output current, I_{OL}				4			8	mA
Width of write-enable or read-enable pulse, t_w		25			25			ns
Setup times, high- or low-level data	Data input with respect to write enable, $t_{setup}(D)$	10			10			ns
	Write select with respect to write enable, $t_{setup}(W)$	15			15			ns
Hold times, high- or low-level data (see Note 1)	Data input with respect to write enable, $t_{hold}(D)$	15			15			ns
	Write select with respect to write enable, $t_{hold}(W)$	5			5			ns
Latch time for new data, t_{latch} (see Note 2)		25			25			ns
Operating free-air temperature range, T_A		−55		125	0		70	℃

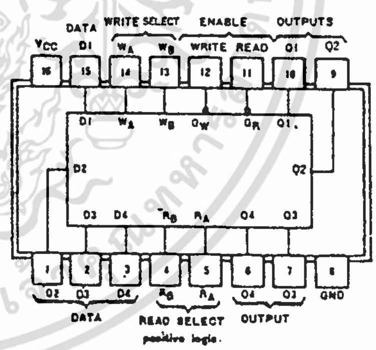
electrical characteristics over recommended operating free-air temperature range

PARAMETER*		TEST CONDITIONS †		MIN	TYP ‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage		SN54LS'			0.7	V
			SN74LS'			0.8	
V_I	Input clamp voltage	V_{CC}=MIN,	I_I=−18mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC}=MIN, V_{IH}=2V, V_{IL}=V_{IL} max	I_{OH}=−1mA SN54LS'	2.4	3.4		V
			I_{OH}=−2.6mA SN74LS'	2.4	3.1		
V_{OL}	Low-level output voltage	V_{CC}MIN, V_{IH}=2V, V_{IL}=V_{IL} max	I_{OL}=4mA		0.25	0.4	V
			I_{OL}=8mA SN74LS'		0.35	0.5	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC}=MAX, V_O=2.7V	V_{IH}=2V,			20	μA
I_{OZL}	Off-state output current, low-level voltage applied	V_{CC}=MAX, V_O=0.4V	V_{IH}=2V,			−20	μA
I_I	Input current at maximum input voltage	V_{CC}=MAX, V_I=7V	Any D,R,or W			0.1	mA
			G_W			0.2	
			G_R			0.3	
I_{IH}	High-level input current	V_{CC}=MAX, V_I=2.7V	Any D,R,or W			20	mA
			G_W			40	
			G_R			60	
I_{IL}	Low-level input current	V_{CC}=MAX	Any D,R,or W			−0.4	mA
			G_W			−0.8	
			G_R			−1.2	
I_{OS}	Short-circuit output current*	V_{CC}=MAX	SN54LS'	30		130	mA
			SN74LS'	30		130	
I_{CC}	Supply current	V_{CC}=MAX	See Note 3		30	50	mA

				MIN	TYP	MAX	UNIT
t_{PLH}	from Read select to output Any Q	V_{CC}=5V, T_A=25℃,	C_L=15pF, R_L=2KΩ		23	40	ns
t_{PHL}					25	45	
t_{PLH}	from Write enable to output Any Q				26	45	ns
t_{PHL}					28	50	
t_{PLH}	from Data to output Any Q				25	45	ns
t_{PHL}					23	40	
t_{ZH}	from Read enable to output Any Q	V_{CC}=5V, T_A=25℃	R_L=2k, C_L=15pF,		15	35	ns
t_{ZL}					22	40	
t_{HZ}			R_L=2kΩ, C_L=5pF		30	50	ns
t_{LZ}					16	35	

Pin Assignment (Top View)

①



positive logic.

NOTES:

1. Write-setup time will protect the data written into the previous address. If protection of data in the previous address is not required, $t_{setup}(W)$ can be ignored as any address selection sustained for the final 30 ns of the write-enable pulse and during $t_{hold}(W)$ will result in data being written into that location. Depending on the duration of the input conditions, one or a number of previous adresses may have been written into.
2. Latch time is the time allowed for the internal output of the latch to assume the state of new data. This is important only when attempting to read from a location immediately after that location has received new data.
3. Maximum I_{CC} is guaranteed for the following worst-case conditions: 4.5 V is applied to all data inputs and both enable inputs, all address inputs are grounded and all outputs are open

A. H=high level, L=low level, X=irrelevant, Z=high impedance(off)

B. (Q=D)=The four selected internal flip-flop outputs will assume the states applied to the four external data inputs.

C. Q_0=the level of Q before the indicated input conditions were established.

D. WOB1 = The first bit of word 0, etc.

Function Tables

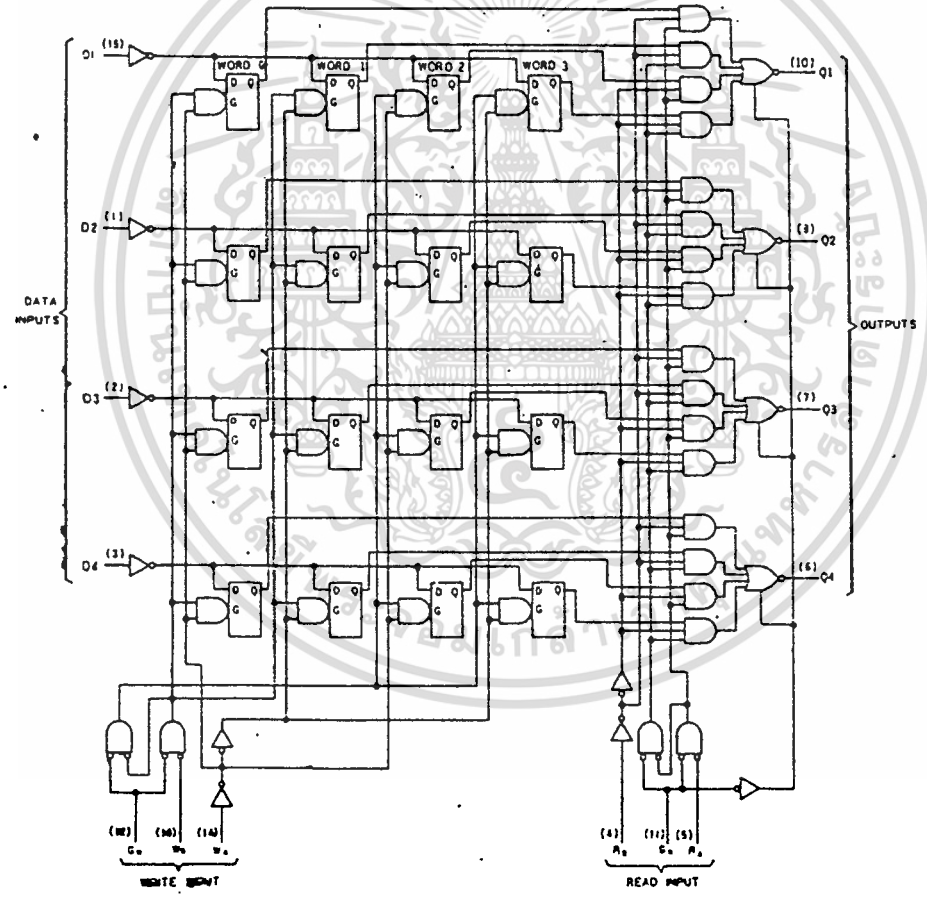
'LS670 WRITE FUNCTION TABLE
(SEE NOTES A,B,AND C)

WRITE INPUTS			WORD			
W_B	W_A	G_W	0	1	2	3
L	L	L	Q=D	Q_0	Q_0	Q_0
L	H	L	Q_0	Q=D	Q_0	Q_0
H	L	L	Q_0	Q_0	Q=D	Q_0
H	H	L	Q_0	Q_0	Q_0	Q=D
X	X	H	Q_0	Q_0	Q_0	Q_0

'LS670 READ FUNCTION TABLE
(SEE NOTES A AND C)

READ INPUTS			OUTPUTS			
R_B	R_A	G_R	Q1	Q2	Q3	Q 4
L	L	L	W0B1	W0B2	W0B3	W0B4
L	H	L	W1B1	W1B2	W1B3	W1B4
H	L	L	W2B1	W2B2	W2B3	W2B4
H	H	L	W3B1	W3B2	W3B3	W3B4
X	X	H	Z	Z	Z	Z

Functional Block Diagram



'LS670 4-BIT 4 REGISTER FILE WITH 3-STATE OUTPUT