บทที่ 4

การออกแบบฮาร์ดแวร์โปรเซสเซอร์

4.1 ลักษณะทั่วไปของโปรเซสเซอร์

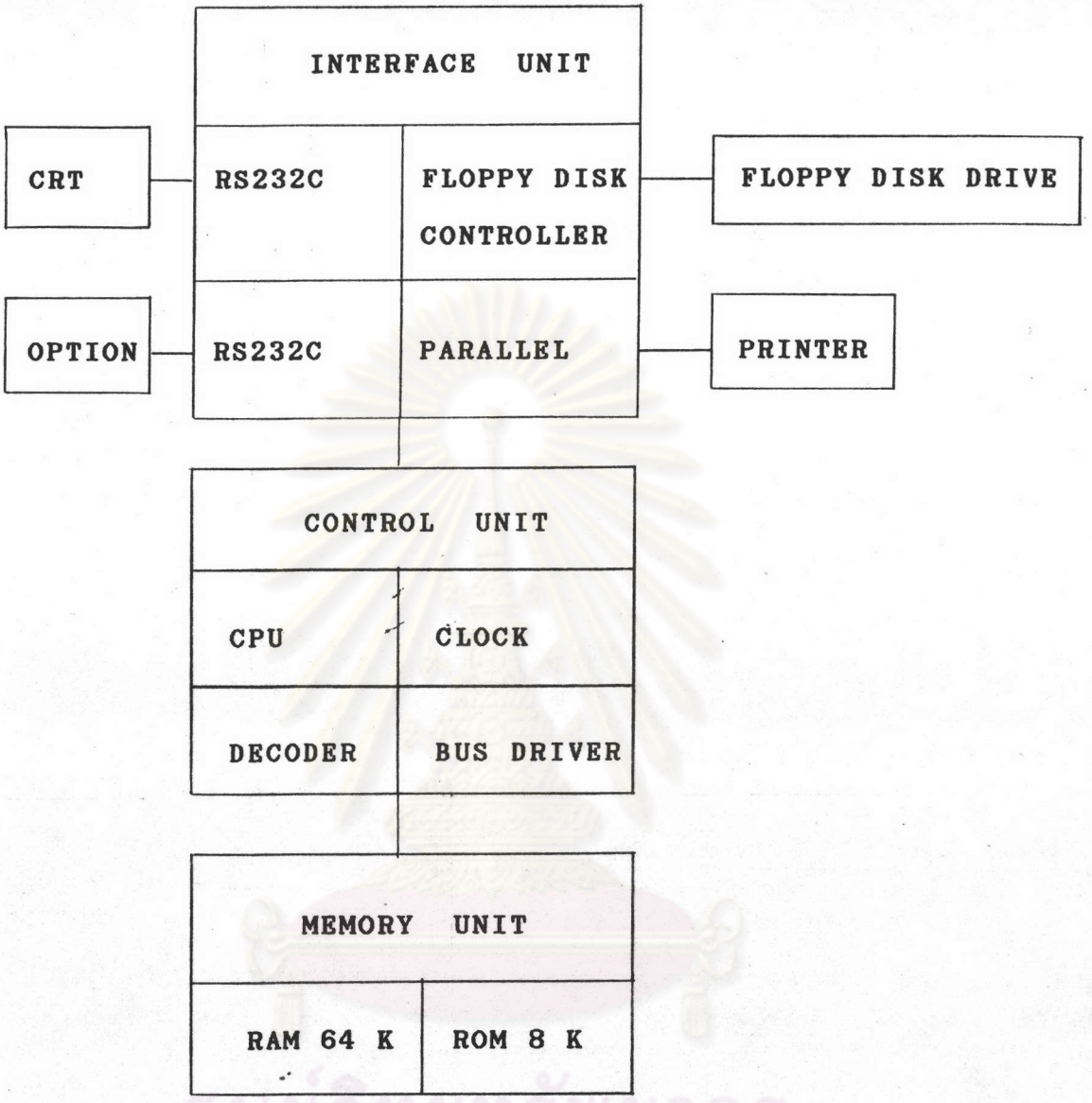
โปรเซสเซอร์ หมายถึง ส่วนอุปกรณ์ซึ่งทำหน้าที่จัดการ หรือควบคุมและประมวลผลการทำงานของระบบซึ่งภายในโปรเซสเซอร์จะประกอบด้วยส่วนที่สำคัญ 2 ส่วน ดังนี้คือ

1. ส่วนฮาร์ดแวร์ (HARDWARE) จะหมายถึงวงจรและองค์ประกอบไอซีที่ใช้ต่อร่วมกันเพื่อช่วยทำหน้าที่ต่าง ๆ เช่น เป็นส่วนควบคุม (ซีพียู) ส่วนหน่วยความจำส่วนอินเตอร์เฟสกับอุปกรณ์ภายนอก

2. ส่วนเพิร์มแวร์ (FIRMWARE) จะหมายถึง โปรแกรมที่ใช้ควบคุมการทำงานของฮาร์ดแวร์โดยตรงให้ทำงานตามหน้าที่ที่ต้องการ ปกติโปรแกรมนี้จะถูกเก็บหรือบันทึกไว้ในหน่วยความจำแบบรอม (ROM) หรือ อีพรอม (EPROM) เพื่อให้ซีพียู ติดต่อใช้งานได้สะดวกรวดเร็ว และโปรแกรมที่เก็บนี้จะมีลักษณะเป็นรหัสภาษาเครื่อง (MACHINE CODE) ของซีพียูที่ใช้

4.2 โครงสร้างของฮาร์ดแวร์โปรเซสเซอร์

จากบทที่ 3 ทำให้ทราบถึงข้อกำหนด และลักษณะของฮาร์ดแวร์โปร-เซสเซอร์ที่ออกแบบ และสร้างซึ่งจะแสดงให้เห็นในรูปที่ 4.1 จากรูปจะเห็นว่าฮาร์ดแวร์ของโปรเซสเซอร์ประกอบด้วยส่วนต่าง ๆ ดังนี้ คือ



รูปที่ 4.1 แสดงโครงสร้างของฮาร์ดแวร์โปรเซสเซอร์ที่ออกแบบ

4.2.1 ส่วนควบคุม (CONTROL UNIT)

ส่วนนี้จะทำหน้าที่ ควบคุม และประมวลผลภายในระบบ ซึ่งจะประกอบด้วย

1. ซีพียู (CPU - CENTRAL PROCESSING UNIT) ทำหน้าที่ควบคุมและประมวลผลภายในระบบโดยการอ่านคำสั่งหรือข้อมูลจากหน่วยความจำหรือ จากส่วนต่าง ๆ ภายในระบบเพื่อถอดรหัสเป็นคำสั่ง หรือข้อมูลเพื่อส่งไปควบคุม หรือให้ยังส่วนต่าง ๆ ด้วย เช่น วงจรอินเตอร์เฟส ซีอาร์ทีเทอร์มินอล วงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก และวงจรอินเตอร์เฟสเครื่องพิมพ์ จะเห็นว่าทำหน้าที่ทั้งการอ่าน (READ) และบันทึก (WRITE) ข้อมูล รวมทั้งส่งสัญญาณเพื่อควบคุม (CONTROL) องค์ประกอบต่าง ๆ ที่ใช้ต่อร่วมกันภายในระบบด้วย ซึ่งสามารถแยกสัญญาณควบคุมออกได้ต่าง ๆ ดังนี้ คือ บัสข้อมูล (DATA BUS) แอดเดรสบัส (ADDRESS BUS) บัสควบคุม (CONTROL BUS)

2. วงจรสัญญาณนาฬิกา (CLOCK) จะเป็นตัวกำเนิดสัญญาณนาฬิกาให้กับระบบเพื่อทำหน้าที่ควบคุมจังหวะการทำงานของซีพียูและองค์ประกอบภายในระบบให้ทำงานอย่างถูกต้องตามจังหวะที่ต้องการ

3. วงจรขับสัญญาณ (BUS DRIVER) วงจรนี้จะทำหน้าที่เพิ่มระดับแรงดันของสัญญาณ ให้คงที่ตามสภาวะทางตรรก (LOGIC) ที่ถูกต้อง เช่น วงจรแอดเดรสบัส บัสข้อมูล บัสควบคุม เป็นต้น

4. วงจรถอดรหัส (DECODER) วงจรนี้จะทำหน้าที่เลือกตำแหน่งหรือแอดเดรส ที่ต้องการเมื่อซีพียูจะติดต่อกับส่วนต่าง ๆ ภายในระบบ เช่น ติดต่อกับหน่วยความจำ หรือส่วนอินเตอร์เฟส เป็นต้น

4.2.2 ส่วนหน่วยความจำ (MEMORY UNIT)

ส่วนนี้จะทำหน้าที่เก็บข้อมูลหรือโปรแกรมที่จะใช้งานภายในระบบ ซึ่งในที่นี้จะใช้หน่วยความจำ 2 ประเภท คือ

1. หน่วยความจำแบบแรม (RAM) หมายถึง หน่วยความจำซึ่งสามารถอ่าน และบันทึกได้จากซีพียูโดยตรง ส่วนนี้จะทำหน้าที่เก็บข้อมูลและโปรแกรมสำหรับใช้งาน (APPLICATION PROGRAM) คือทำหน้าที่เป็น WORKING AREA นั่นเอง ขณะเดียวกันบางส่วนของหน่วยความจำยังใช้เก็บโปรแกรมที่ทำหน้าที่เป็น OPERATING SYSTEM และเก็บข้อมูลที่ใช้เกี่ยวกับซีพียู เช่น STACK AREA เป็นต้น

2. หน่วยความจำแบบรอม (ROM) หมายถึง หน่วยความจำซึ่งสามารถอ่านได้อย่างเดียว ซึ่งจะเรียกว่าเป็นแบบ READ ONLY MEMORY หน่วยความจำนี้ จะใช้ทำหน้าที่ เก็บโปรแกรมที่ใช้งาน ควบคุมฮาร์ดแวร์ภายในระบบ เช่น โปรแกรมควบคุมการรับส่งข้อมูล ระหว่างซีพียู กับอุปกรณ์ภายนอก โดยปกติโปรแกรมเหล่านี้มักจะเขียนเป็นส่วนโปรแกรมย่อย[9] ๆ ซึ่งสามารถเรียกใช้ได้เพื่อการติดต่อกับฮาร์ดแวร์โดยตรง เช่น โปรแกรมควบคุมการอ่านและบันทึกข้อมูลลงจานแม่เหล็ก หรือโปรแกรมรับส่งข้อมูลจากเทอร์มินอล และโปรแกรมส่งข้อมูลไปพิมพ์ยังเครื่องพิมพ์ เป็นต้น

4.2.3 ส่วนอินเตอร์เฟส (INTERFACE UNIT)

ส่วนนี้จะทำหน้าที่ ติดต่อรับส่งข้อมูล หรือสัญญาณควบคุมระหว่างอุปกรณ์ภายนอกกับซีพียู ซึ่งส่วนอินเตอร์เฟสนี้จะประกอบด้วย

1. วงจรอินเตอร์เฟส ซีอาร์ทีเทอร์มินอล จะทำหน้าที่รับส่งข้อมูล และสัญญาณควบคุมระหว่างเทอร์มินอลกับซีพียู ซึ่งจะใช้การอินเตอร์เฟสแบบ RS232C

2. วงจรอินเตอร์เฟสกับตัวขับจานแม่เหล็ก จะทำหน้าที่รับส่งข้อมูลหรือสัญญาณต่าง ๆ ระหว่างตัวขับจานแม่เหล็กกับซีพียู เช่น การอ่าน หรือบันทึกข้อมูลลงจานแม่เหล็ก เป็นต้น

3. วงจรอินเตอร์เฟสกับเครื่องพิมพ์ จะทำหน้าที่รับข้อมูลและส่งสัญญาณควบคุมระหว่างซีพียูกับเครื่องพิมพ์ การอินเตอร์เฟสจะเป็นแบบขนาน

4. วงจรอินเตอร์เฟส กับ อุปกรณ์ภายนอกอื่น ๆ (OPTION) เพื่อไว้ใช้ติดต่อ (COMMUNICATION) กับอุปกรณ์ภายนอกอื่น ๆ เช่น คอมพิวเตอร์ หรือ เครื่องพิมพ์แบบอนุกรม RS232C เป็นต้น

4.3 การพิจารณาเลือกองค์ประกอบไอซีสำหรับออกแบบฮาร์ดแวร์โปรเซสเซอร์

ก่อนอื่นจะต้องยึด หลักการพิจารณาออกแบบ เช่น เลือกใช้ไอซี[8] ที่เหมาะสม และที่มีจำหน่ายในท้องตลาดเมืองไทย การออกแบบไม่ยุ่งยากซับซ้อน สามารถใช้งานได้ตามต้องการ และเต็มความสามารถของอุปกรณ์ไอซี พยายามเลือกใช้ไอซีในตระกูลเดียวกัน[12] เพื่อให้ออกแบบง่าย และสิ่งที่สำคัญอีกอย่างคือ ควรจะมีขนาดเล็ก ประหยัด สามารถหาอุปกรณ์ทดแทนได้ในอนาคตและเป็นที่

นิยมในตลาดเมืองไทย สามารถหาคู่มือเพื่อศึกษาการทำงานได้ โดยจะทำการพิจารณาเลือกองค์ประกอบไอซี เพื่อออกแบบได้ตามลำดับดังนี้

4.3.1 องค์ประกอบไอซีของส่วนควบคุม

1. ออกแบบส่วนควบคุมหรือซีพียู จะเลือกใช้ไมโครโปรเซสเซอร์เบอร์ Z80A เพื่อจะให้สามารถใช้งานกับซอฟท์แวร์ในระบบซีพีเอ็มได้ และสามารถทำงานได้กับความเร็วของสัญญาณนาฬิกา ขนาด 4 เมกกะเฮิรทซ์ และมีชารีเฟรช (REFRESH) สำหรับต่อกับไดนามิคแรม (DYNAMIC RAM) ด้วย

2. ออกแบบวงจร ให้กำเนิดสัญญาณนาฬิกา จะใช้ความถี่ 4 เมกกะเฮิรทซ์ เพื่อให้ใช้งานกับ Z80A ได้เต็มตามความสามารถ และจะใช้คริสตัล (CRYSTAL) ขนาด 16 เมกกะเฮิรทซ์ มาหาร 4 เพื่อให้ความถี่เที่ยงตรงมากขึ้น โดยต่อใช้ร่วมกับไอซี 7404 และใช้ไอซี 74LS390 ทำหน้าที่เป็นเคาน์เตอร์ (COUNTER - วงจรนับ) หาร 4

3. ออกแบบบัสไดร์เวอร์ (BUS DRIVER) จะเลือกใช้ไอซีเบอร์ 74LS245 สำหรับบัสข้อมูล เพราะใช้ได้ทั้งสองทิศทางเป็นลักษณะ BI-DIRECTIONAL BUS DRIVER ทำให้ออกแบบง่าย และประหยัดไอซีได้หนึ่งตัว สำหรับแอดเดรสบัส จะให้ไอซี 74LS244 2 ตัว เพื่อให้แน่ใจว่าสัญญาณแรงพอที่จะขับวงจรทั้งหมดได้

4. วงจรถอดรหัส (DECODER) จะเลือกใช้ ไอซีเบอร์ 74LS139 เพราะสามารถเลือก (SELECT) ได้ 8 ตำแหน่ง และมีสัญญาณ STROB 2 ขา ทำให้สามารถลดจำนวนไอซีในการถอดรหัส (DECODE) ลงได้ แต่เนื่องจาก หน่วยความจำของระบบมีมากกว่า 64 กิโลไบท์ ดังนั้นจึงต้องมีวงจรสำหรับเลือก BANK เพิ่มขึ้น โดยจะใช้คำสั่งไอโอ และไอซี 74LS74 ซึ่งเป็นฟลิปฟลอปทำหน้าที่นี้

4.3.2 องค์ประกอบของไอซีส่วนหน่วยความจำ

1. หน่วยความจำแรม (RAM) เลือกใช้ ไอซีเบอร์ 4164 เพราะเป็นแบบไดนามิคแรมขนาด 64 กิโลไบท์คูณ 1 บิท เวลาที่ใช้ในการอ่าน 250 nS ทำให้สามารถใช้งาน กับสัญญาณนาฬิกาขนาด 4 เมกกะเฮิรทซ์ ได้ และทำให้ประหยัดเนื้อที่เพราะสามารถใช้เพียง 8 ตัว เท่านั้น

2. หน่วยความจำรอม (ROM) เลือกใช้ ไอซีเบอร์ 2764 เพราะสามารถใช้งานได้กับสัญญาณนาฬิกาขนาด 4 เมกกะเฮิรทซ์ได้ โดยไม่ต้อง

ใช้วงจร WAIT STATE เพื่อทำการ WAIT ซีพียู ทำให้ซีพียูทำงานได้รวดเร็วขึ้น ขณะเดียวกันยังทำให้สามารถประหยัดไอซีที่จะใช้ทำวงจร WAIT STATE ได้อีก 1 ตัว และยังทำให้สามารถแก้ไขเพิ่มเติมเฟิร์มแวร์ได้อีกในอนาคต เพราะไอซีเบอร์ 2764 มีขนาดถึง 8 กิโลไบท์

4.3.3 องค์ประกอบของไอซีส่วนอินเตอร์เฟส

1. วงจรอินเตอร์เฟส ซีอาร์ทีเทอร์มินอล ซึ่งเป็นแบบอนุกรม RS232C จะพิจารณาเลือกใช้ ไอซีเบอร์ Z80A-SIO ซึ่งเป็น SERIAL INPUT/OUTPUT และมีพอร์ท (PORT) ใช้งาน 2 พอร์ท สำหรับในที่นี้จะใช้พอร์ท B ซึ่งมีขาสัญญาณนาฬิกากำหนดความเร็วการรับส่งข้อมูลอยู่ในขาเดียวกัน และยังมีข้อดีคือมีระบบบัสสัญญาณต่าง ๆ ที่สามารถต่อใช้งานร่วมกับ Z80A-CPU ได้โดยตรง เช่น บัสข้อมูล $\overline{\text{RESET}}$ $\overline{\text{MI}}$ $\overline{\text{IORQ}}$ $\overline{\text{RD}}$ $\overline{\text{INT}}$ เป็นต้น สำหรับวงจรกำเนิดสัญญาณนาฬิกาที่ใช้กำหนดความเร็วการรับส่งข้อมูลให้กับ Z80-SIO นี้จะใช้ไอซีเบอร์ Z80-CTC ซึ่งเป็น COUNTER TIMER CIRCUIT ทำหน้าที่นี้ ซึ่งจะใช้ความถี่สัญญาณนาฬิกาขนาด 2 เมกกะเฮิรทซ์มาหาร โดยจะเลือกใช้ COUNTER MODE และยังสามารถเลือกโปรแกรมเพื่อหารความถี่ได้อีก ซึ่งสามารถใช้กำหนด BAUD RATE ตั้งแต่ 300-9600 BPS ได้และยังต่อใช้งานร่วมกับระบบบัสได้ง่าย

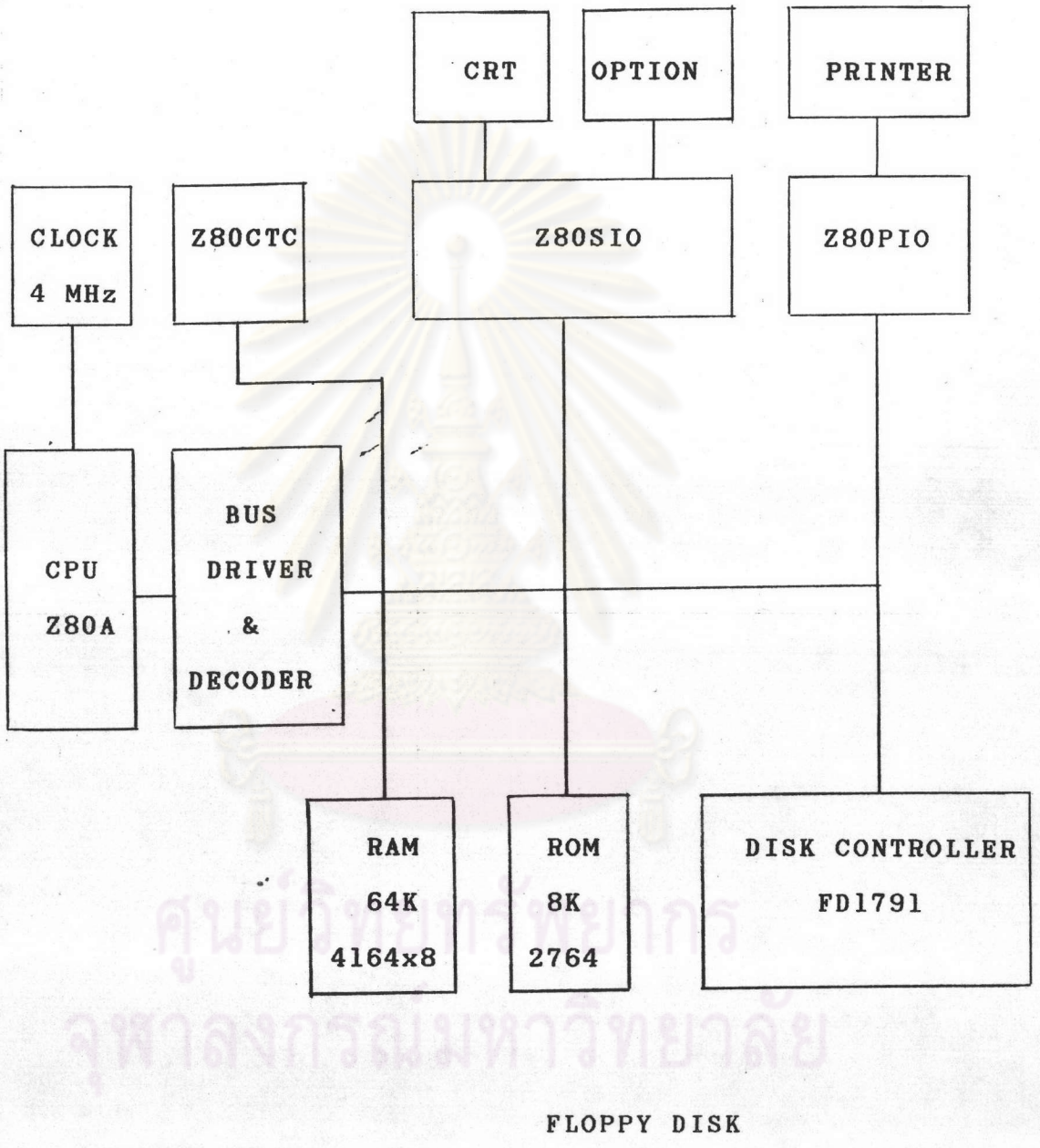
2. วงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก จะเลือกใช้ไอซีเบอร์ FD1791 เพื่อลดความยุ่งยากในการออกแบบวงจร ขณะเดียวกัน ยังสามารถใช้งานในลักษณะ IBM FORMAT ได้ และยังสามารถใช้งานกับจานแม่เหล็กได้ทั้งแบบชิงเกิลเดนซิตี้ และดับเบิลเดนซิตี้ ด้วย

3. วงจรอินเตอร์เฟส กับเครื่องพิมพ์แบบขนาน จะพิจารณาเลือกใช้ ไอซีเบอร์ Z80A-PIO เพราะมี 2 พอร์ท และออกแบบใช้งานร่วมกับซีพียูได้ง่ายเพราะมีระบบบัสที่ต่อเข้ากันได้ ขณะเดียวกันยังสามารถโปรแกรมให้แต่ละขาเป็นอินพุท หรือเอาท์พุทได้ตามต้องการ จะใช้พอร์ท (PORT) A เป็นพอร์ทควบคุมสัญญาณ และพอร์ท B เป็นพอร์ทรับข้อมูลส่งไปพิมพ์ยังเครื่องพิมพ์

4. วงจรอินเตอร์เฟส กับอุปกรณ์ภายนอก (OPTION) จะใช้แบบอนุกรม RS232C โดยจะเลือกใช้ Z80A-SIO ตรงพอร์ท A ที่เหลือจากการใช้งานครั้งแรกที่ต่อกับเทอร์มินอล ขณะเดียวกันยังมีข้อดี คือ สามารถใช้งานในลักษณะ SDLC และ HDLC ได้ และยังสามารถใช้สัญญาณนาฬิกาในการ

กำหนดความเร็วเพื่อรับและส่งข้อมูลต่างกันได้

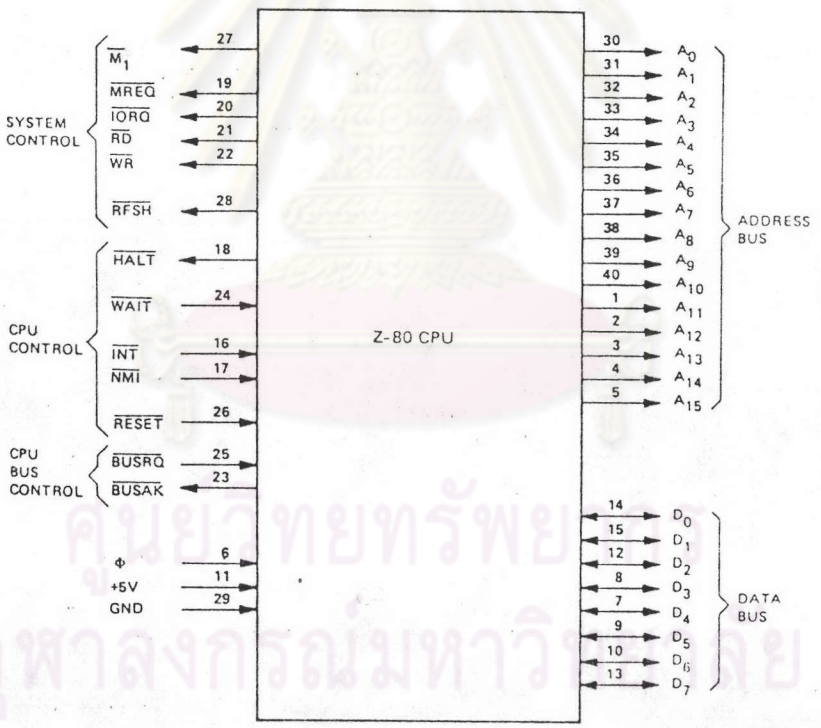
รายละเอียดแสดง องค์ประกอบ ไอซีที่ใช้ออกแบบฮาร์ดแวร์โปรเซส-เซอร์จะแสดงในรูปที่ 4.2 สำหรับวงจรสมบูรณ์จะแสดงในภาคผนวก ก



รูปที่ 4.2 แสดงโครงสร้างและองค์ประกอบของฮาร์ดแวร์โปรเซสเซอร์

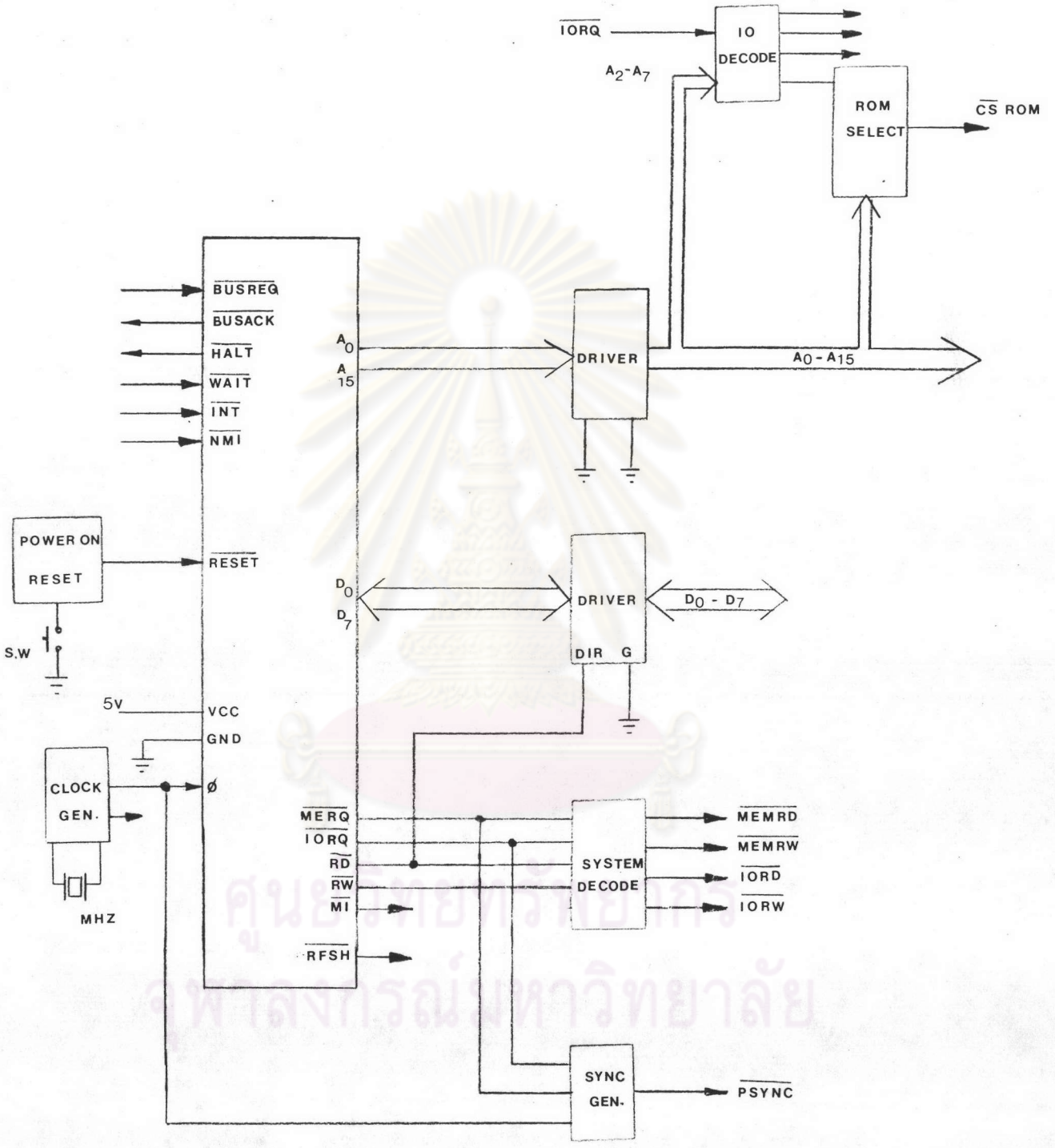
4.4 การออกแบบส่วนควบคุมซีพียู

ในที่นี้จะใช้ไมโครโปรเซสเซอร์เบอร์ Z80A ทำหน้าที่เป็นซีพียูของระบบ ลักษณะของไอซีเบอร์นี้ จะแสดงในรูปที่ 4.3 จากรูปจะเห็นว่ามีลักษณะเป็นแบบ 40 ขา โดยแบ่งขาออกทำหน้าที่ต่าง ๆ แยกได้เป็นส่วน ๆ ดังนี้ คือ แอดเดรสบัส บัสข้อมูล SYSTEM CONTROL CPU CONTROL และ CPU BUS CONTROL เป็นต้น (สำหรับรายละเอียดของไอซีเบอร์นี้หาได้จากคู่มือไอซีตระกูล Z-80 ของ INTEL)



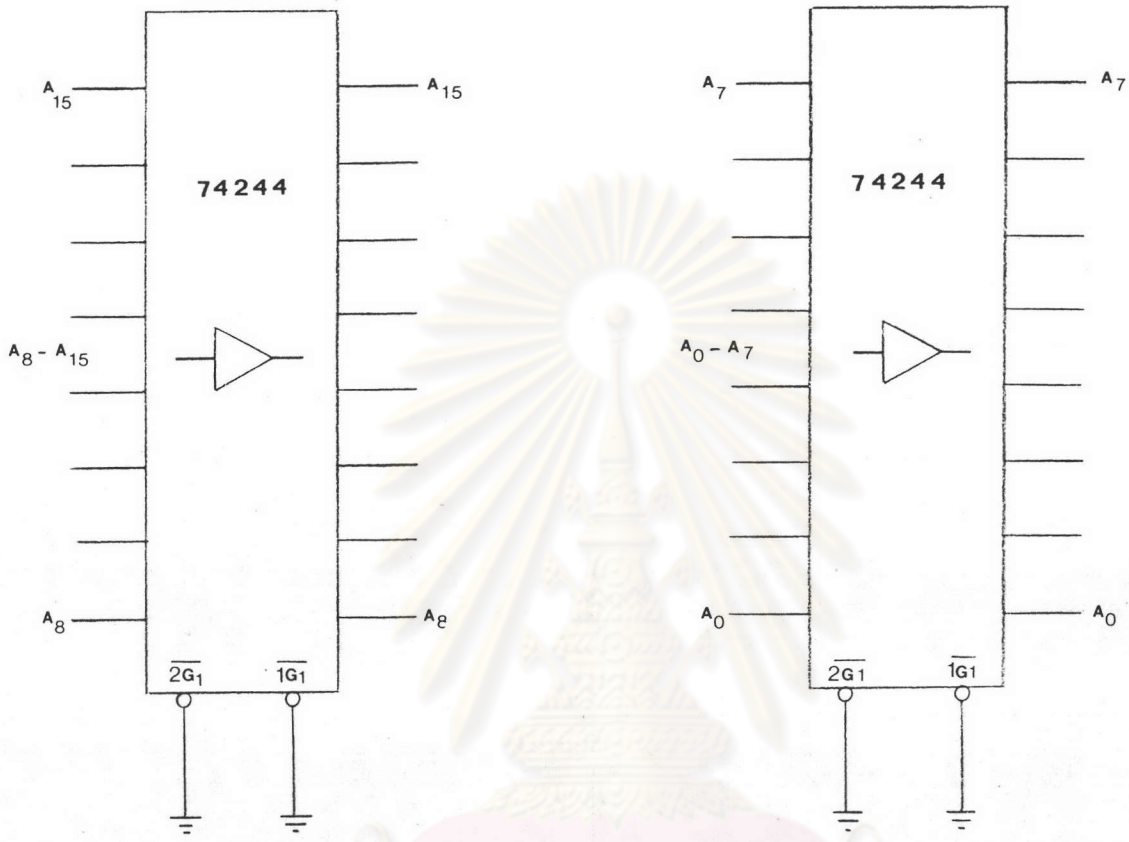
รูปที่ 4.3 แสดงลักษณะไอซีเบอร์ Z80A และหน้าที่ของขาต่าง ๆ

โดยปกติวงจรซีพียูนี้มิได้ หมายถึง เฉพาะไอซีเบอร์นี้ตัวเดียวเท่านั้น แต่จะต้องประกอบด้วยวงจรส่วนรอบนอกที่จำเป็นสำหรับการทำหน้าที่ต่าง ๆ ด้วย เช่น วงจรบัสไดร์เวอร์ ซึ่งมีทั้งแอดเดรสบัส บัสข้อมูล และวงจรถอดรหัส (DECODER) สำหรับเลือกตำแหน่งหน่วยความจำ หรือตำแหน่งของอุปกรณ์ไอโอด้วย ขณะเดียวกันจะต้องมีวงจรสำหรับสร้างสัญญาณนาฬิกา (CLOCK) เพื่อให้กับซีพียู และถ้าศึกษาดูจาก ไอซีเบอร์ Z80A-CPU แล้ว จะเห็นว่าขาสัญญาณที่จะใช้เกี่ยวกับการอ่านหรือบันทึกข้อมูลลงหน่วยความจำหรือกับอุปกรณ์ไอโอนั้น ยังไม่มีขาสัญญาณนี้โดยตรง แต่จะต้องทำการสร้างสัญญาณนี้ ขึ้นมาใหม่จากขาสัญญาณ $\overline{\text{I/ORQ}}$ $\overline{\text{MERQ}}$ $\overline{\text{RD}}$ $\overline{\text{WR}}$ ด้วยวงจร SYSTEM CONTROL DECODER เพื่อให้ได้สัญญาณ $\overline{\text{MEMRD}}$ $\overline{\text{MEMWR}}$ $\overline{\text{IORD}}$ $\overline{\text{IOWR}}$ สำหรับใช้งานกับหน่วยความจำและอุปกรณ์ไอโอด้วย รายละเอียดของวงจรซีพียู ซึ่งประกอบกับวงจรรอบนอกที่กล่าวมาข้างต้น จะแสดงเป็นบล็อกไดอะแกรม (BLOCK DIAGRAM) ในรูปที่ 4.4



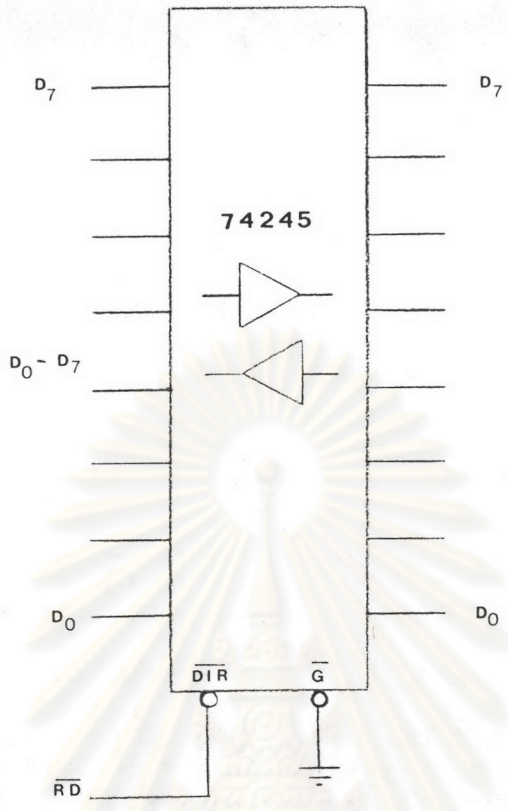
รูปที่ 4.4 แสดง BLOCK DIAGRAM ของวงจรซีพียู

4.4.1 การออกแบบวงจร BUS DRIVER



รูปที่ 4.5A แสดงวงจร ADDRESS BUS

เพื่อให้สัญญาณแอดเดรสบัสมีความแรงพอสำหรับขับวงจรทั้งหมดและทำหน้าที่เป็นตัวกั้น (BUFFER) จะใช้ไอซีเบอร์ 74LS244 ซึ่งเป็น NONINVERTED 3-STATE OUTPUTS DRIVE BUS LINE ซึ่งมีคุณสมบัติ HIGH FAN-OUT สามารถรับกระแส (SINK CURRENT) ได้ถึง 24 mA และจ่ายกระแส (SOURCE CURRENT) ได้ถึง -15 mA

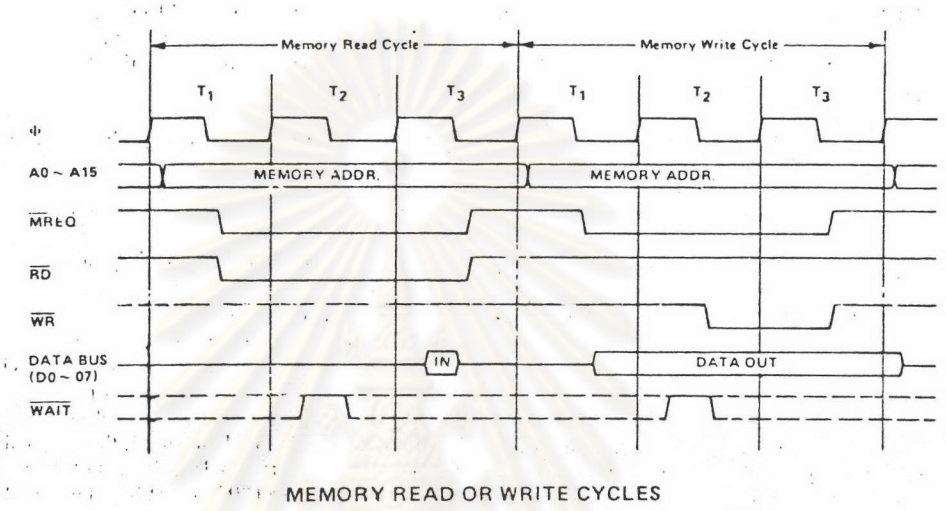


รูปที่ 4.5B แสดงวงจร DATA BUS

สำหรับสัญญาณบัสข้อมูล เพื่อให้สัญญาณมีความแรงพอที่จะขับให้สัญญาณข้อมูลคงที่ตามสภาวะ (STATE) ทั้งการอ่าน และเขียนข้อมูล ดังนั้น จำเป็นต้องใช้บัสไดร์เวอร์ (BUS DRIVER) เพิ่ม โดยจะใช้ ไอซีเบอร์ 74LS245 ซึ่งเป็น BI-DIRECTIONAL DATA BUS DRIVER ทำหน้าที่นี้ ขณะเดียวกันยังสามารถรับกระแส (SINK CURRENT) ได้ถึง 24 mA และจ่ายกระแส (SOURCE CURRENT) ได้ถึง -15 mA แต่เนื่องจากว่าต้องทำหน้าที่ ทั้งการอ่านและการบันทึกข้อมูล ดังนั้นจะใช้สัญญาณ $\overline{RD}$ จากซีพียูมาทำหน้าที่ควบคุมทิศทางการรับ และส่งข้อมูลโดยต่อเข้ากับขา $\overline{DIR}$ (DIRECTION) ของไอซี 74LS245

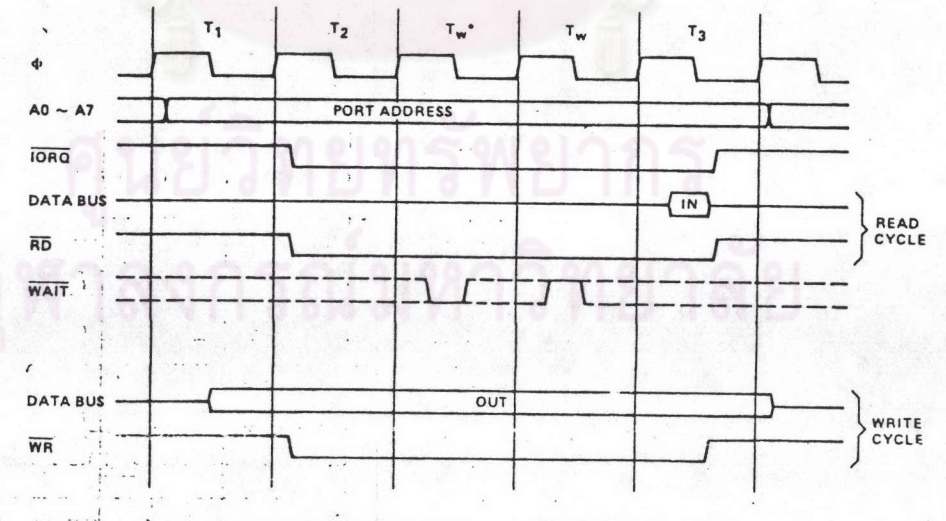
4.4.2 การออกแบบวงจร CONTROL SYSTEM DECODER

เพื่อสร้างสัญญาณในการควบคุมการอ่าน และบันทึกหน่วยความจำ และอุปกรณ์ไอโอจะสามารถศึกษาเพื่อสร้างวงจร ให้กำเนิดสัญญาณสำหรับควบคุมการอ่าน และการบันทึกได้โดยจากการสังเกตไทม์มิ่ง (TIMING) การทำงานของซีพียู ดังรูปที่ 4.6 และรูปที่ 4.7



MEMORY READ OR WRITE CYCLES

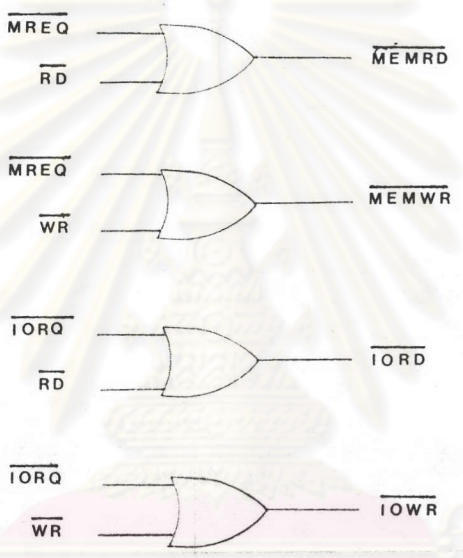
รูปที่ 4.6 แสดง TIMING ของซีพียูเมื่อทำการอ่านและบันทึกหน่วยความจำ



INPUT OR OUTPUT CYCLES WITH WAIT STATES

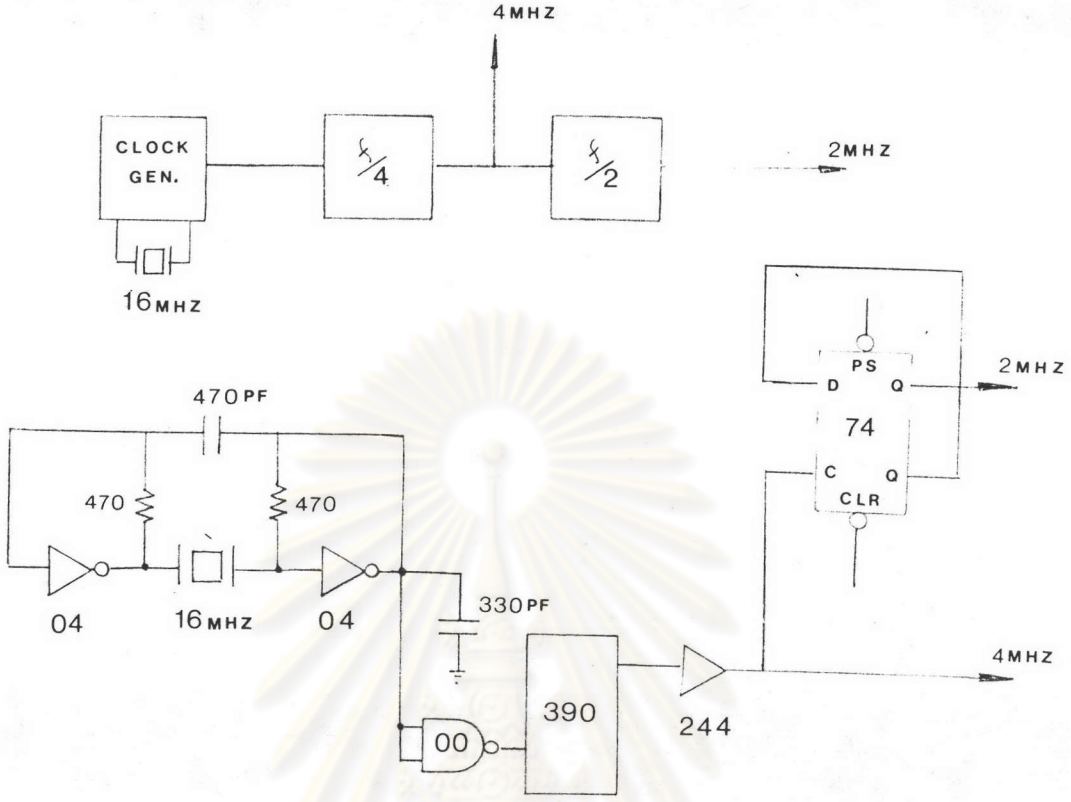
รูปที่ 4.7 แสดง TIMING ของซีพียูเมื่อทำการอ่านและบันทึกอุปกรณ์ไอโอ

จากรูปที่ 4.6 จะเห็นว่าเราสามารถนำสัญญาณ $\overline{MREQ}$ มา OR กับสัญญาณ $\overline{RD}$ เพื่อสร้างสัญญาณอ่าน $\overline{MEMRD}$ ได้ และในช่วงการบันทึกสามารถใช้สัญญาณ $\overline{MREQ}$ มา OR กับสัญญาณ $\overline{WR}$ ทำให้ได้สัญญาณ สำหรับการบันทึกหน่วยความจำคือ $\overline{MEMWR}$ ขณะเดียวกันในการสร้างสัญญาณอ่าน-บันทึก สำหรับอุปกรณ์ไอโอจากรูปที่ 4.7 สามารถใช้ $\overline{IORQ}$ มา OR กับสัญญาณ $\overline{RD}$ เพื่อสร้างสัญญาณอ่านอุปกรณ์ไอโอ $\overline{IORD}$ และนำสัญญาณ $\overline{IORQ}$ มา OR กับสัญญาณ $\overline{WR}$ เพื่อสร้างสัญญาณบันทึกอุปกรณ์ไอโอ $\overline{IOWR}$ ได้ ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 แสดงวงจรในการควบคุมการอ่าน-บันทึก
หน่วยความจำและอุปกรณ์ไอโอ

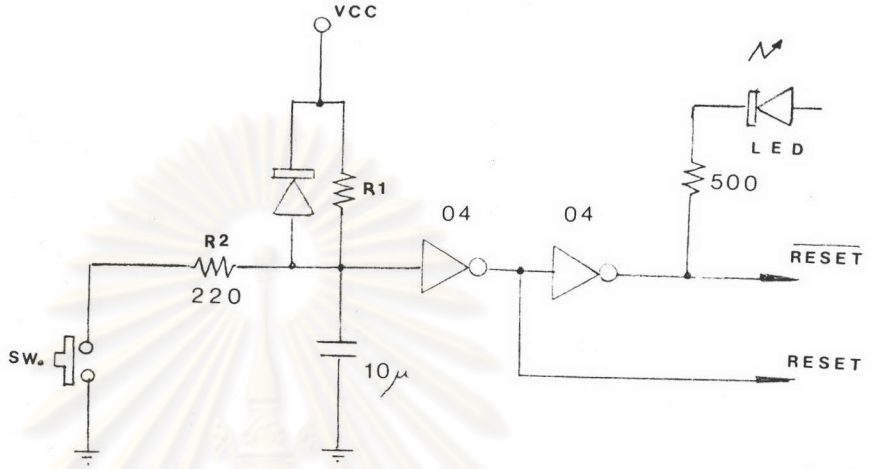
4.4.3 การออกแบบวงจรให้กำเนิดสัญญาณนาฬิกา



รูปที่ 4.9 แสดง BLOCK DIAGRAM และวงจรให้กำเนิดสัญญาณนาฬิกา

จากวงจร จะเห็นว่าใช้คริสตัล (CRYSTAL) และไอซีเบอร์ 7404 และอุปกรณ์ RC เพื่อทำหน้าที่ให้กำเนิดสัญญาณนาฬิกา แต่ความถี่ที่ออกมายังมีลักษณะไม่เป็นรูปสี่เหลี่ยมนัก จึงต้องใช้ NAND GATE ทำหน้าที่ไดร์เวอร์ และให้กับไอซีเบอร์ 74LS390 ที่เป็นเคาน์เตอร์ (COUNTER) เพื่อทำหน้าที่หาร 4 ให้ได้ความถี่ 4 เมกกะเฮิรทซ์ โดยผ่านไดร์เวอร์ ไอซี 74LS244 ให้ได้สัญญาณแรงพอที่จะขับวงจรภายในระบบ เนื่องจากบางจุดในวงจรจะใช้ความถี่ 2 เมกกะเฮิรทซ์ จึงใช้ไอซีเบอร์ 74LS74 ซึ่งเป็น FLIPFLOP ทำหน้าที่หาร 2 จะได้ความถี่ 2 เมกกะเฮิรทซ์ตามต้องการ

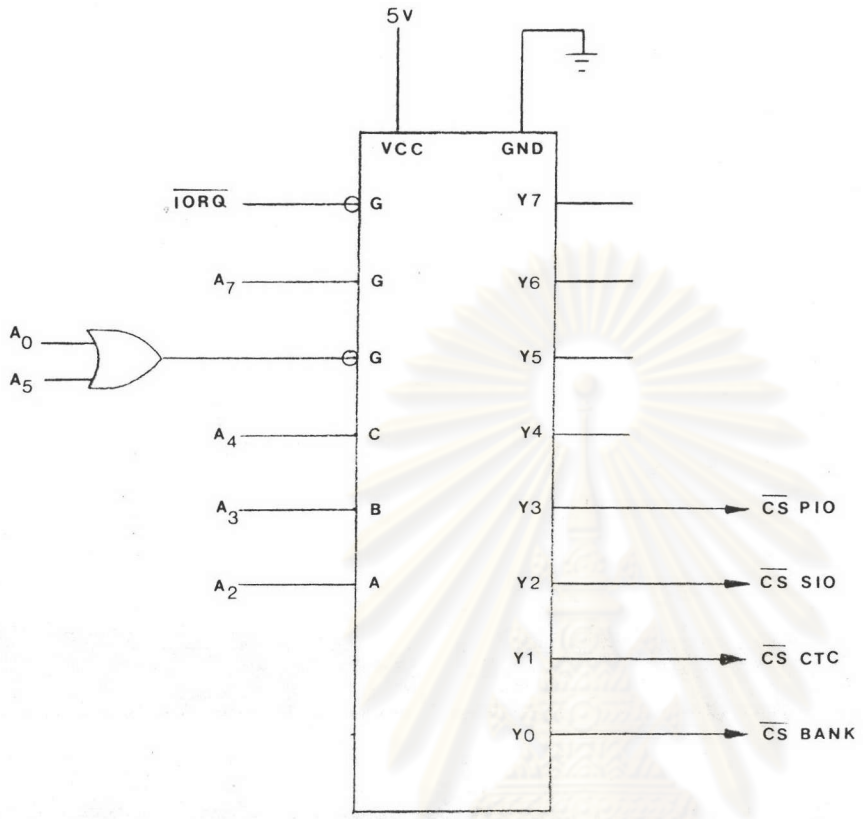
4.4.4 การออกแบบวงจร POWER ON RESET



รูปที่ 4.10 แสดงวงจร POWER ON RESET

จากวงจรจะสังเกตเห็นว่า เมื่อมีการเปิดเครื่องครั้งแรก C1 ซึ่งตอนแรกมีประจุเป็นศูนย์ค่า $\overline{\text{RESET}}$ จะเป็น $\overline{\text{RESET}}$ และค่า RESET จะเป็น RESET ทำให้ซีพียูสามารถ RESET ได้เป็นลักษณะ POWER ON RESET หลังจาก C1 เริ่ม CHARGE จนมีประจุจะมีกระแสไหลผ่าน R1 ทำให้มี VOLTAGE ตกคร่อม C1 เพื่อผ่าน INVERTER จะทำให้ $\overline{\text{RESET}}$ มีค่าตรงข้ามเป็น RESET และ RESET เป็น $\overline{\text{RESET}}$ ทำให้ซีพียูเลิกการ RESET ถ้าหากมีการกด SW.1 จะทำให้ C1 คายประจุผ่าน R2 จนหมดจนทำให้ C1 มีประจุเป็นศูนย์ ทำให้ $\overline{\text{RESET}}$ มีค่าเป็น $\overline{\text{RESET}}$ และ RESET เป็น RESET ทำให้ซีพียูอยู่ในสภาวะ RESET

4.4.5 การออกแบบวงจร I/O DECODER

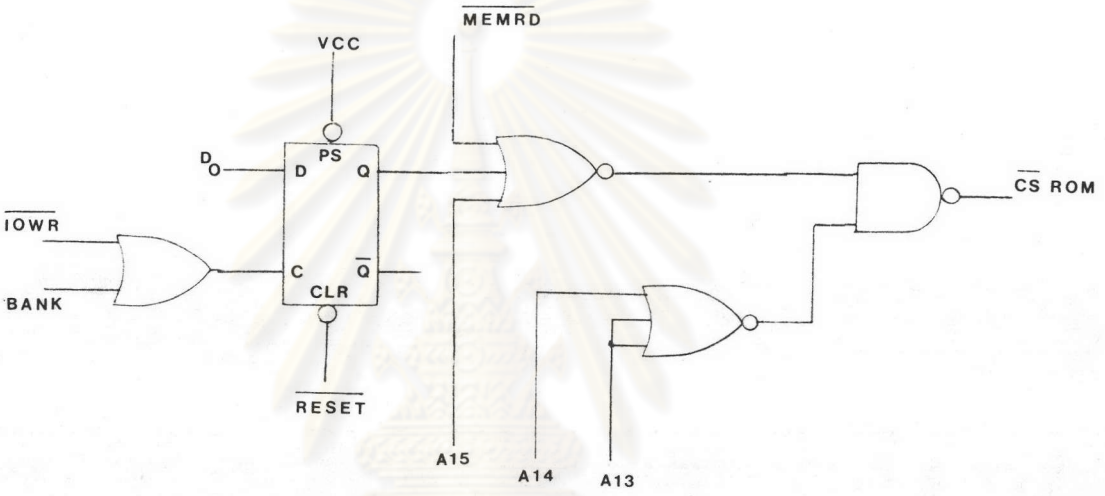


รูปที่ 4.11 แสดงวงจร I/O DECODER

พิจารณาเลือกใช้ ไอซีเบอร์ 74LS138 ซึ่งเป็น 3 TO 8 LINE DECODER/MULTIPLEXERS โดยนำ A2-A7 และ $\overline{\text{IORQ}}$ มาทำการถอดรหัส (DECODE) ดังนั้นจะเห็นว่าต้องมีการอ้างถึงคำสั่งเกี่ยวกับอินพุทและเอาท์พุท ในแอดเดรสระหว่าง 80-8F เท่านั้นที่จะเกิดการเลือกขึ้น โดยแยกดังนี้คือ แอดเดรสระหว่าง 80-83 จะเป็นการ SELECT BANK แอดเดรส 84-87 จะเป็นการเลือก Z80-CTC แอดเดรส 88-8B จะเลือก Z80-SIO และ แอดเดรส 8C-8F จะเป็นการเลือก Z80-PIO ส่วนที่เหลือจะเป็น OPTION

4.4.6 การออกแบบวงจรเลือกหน่วยความจำแบบรอม

เนื่องจาก Z80 ซีพียูมีแอดเดรสบัสเพียง 16 บิท และสามารถแอดเดรสหน่วยความจำในระบบได้สูงสุดเพียง 64 กิโลไบท์ ดังนั้น ถ้าหากจะใช้ต่อหน่วยความจำนอกเหนือเพิ่มขึ้น จึงจำเป็นจะต้องสร้างวงจรเลือก BANK ขึ้นเพื่อ DISABLE 64 กิโลไบท์แรกทิ้งก่อน โดยอาศัยคำสั่งไอโอมาใช้ร่วมกับค่าของข้อมูลในการเลือกแต่ละส่วนของหน่วยความจำ ดังรูปที่ 4.12

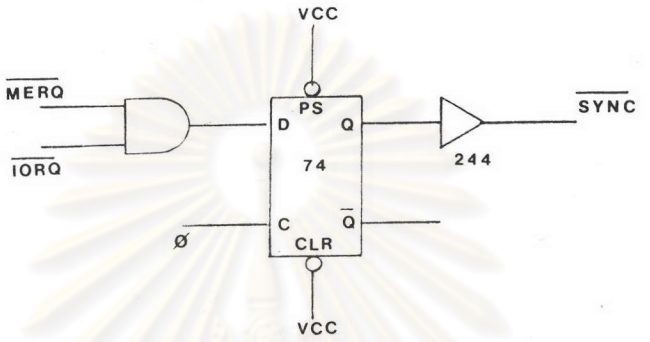


รูปที่ 4.12 วงจรเลือกหน่วยความจำรอม

จากรูป จะเห็นว่าถ้าต้องการจะเลือกใช้งานหน่วยความจำรอมจะต้องใช้คำสั่ง $\overline{\text{IOWR}}$ ตรงพอร์ท หรือแอดเดรสช่วง 80-83 เท่านั้น พร้อมกับ OUT ข้อมูลที่มีค่า D0 เป็น 0 และจะใช้อยู่ในช่วง 8 กิโลไบท์แรก คือตั้งแต่แอดเดรส 0000-1FFF เท่านั้น จะเห็นว่าครั้งแรกเมื่อเริ่ม RESET ระบบซีพียูจะทำงานในหน่วยความจำรอม ถ้าจะเลิกใช้หรือ DISABLE จะต้อง OUT ค่า D0 เป็น 1 ที่พอร์ท 80-83 และเนื่องจากรอมเป็น READ ONLY MEMORY จึงต้องใช้สัญญาณ $\overline{\text{MEMRD}}$ มาช่วยด้วย

4.4.7 การออกแบบวงจรสร้างสัญญาณ $\overline{\text{SYNC}}$

สัญญาณ $\overline{\text{SYNC}}$ จะเป็นสัญญาณที่เกิดขึ้น ทุกครั้งที่มีการใช้คำสั่งเกี่ยวกับหน่วยความจำและอุปกรณ์ไอโอ คือเกิดตามสัญญาณ $\overline{\text{MERQ}}$ หรือ $\overline{\text{IORQ}}$ ในช่วงสัญญาณนาฬิกาลูกถัดไปเพื่อใช้ควบคุมวงจรบางส่วนให้ทำงาน SYNCHRONIZE กันภายในระบบวงจรสร้างสัญญาณ $\overline{\text{SYNC}}$ จะแสดงในรูปที่ 4.13



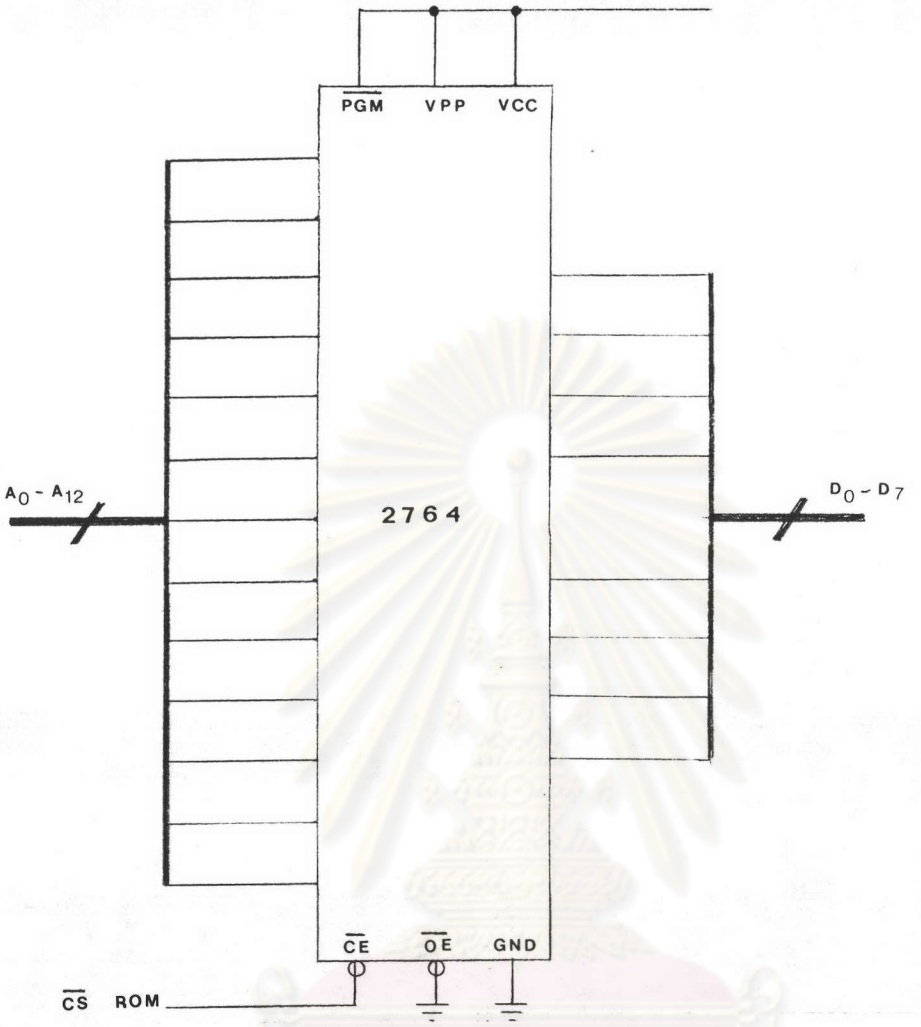
รูปที่ 4.13 แสดงวงจรสร้างสัญญาณ $\overline{\text{SYNC}}$

4.5 การออกแบบหน่วยความจำ

หน่วยความจำที่ใช้อยู่ในระบบนี้มีอยู่ 2 ประเภท คือ. ชนิดรอม และไดนามิคแรม โดยจะใช้หน่วยความจำรอม 8 กิโลไบท์ และหน่วยความจำแรม 64 กิโลไบท์ ซึ่งการออกแบบจะแบ่งเป็นส่วนดังนี้

4.5.1 การออกแบบวงจรหน่วยความจำชนิดรอม

จะพิจารณาเลือกใช้ไอซีเบอร์ 2764 ซึ่งเป็นอีพรอม (EPROM - ERASABLE PROM) มีขนาด 8 กิโลไบท์ และมีความเร็วในการทำงาน (ACESS TIME) ถึง 200 nS ทำให้สามารถใช้งานกับความเร็วของซีพียูขนาด 4 เมกกะเฮิรทซ์ ได้ดังแสดงในรูปที่ 4.14

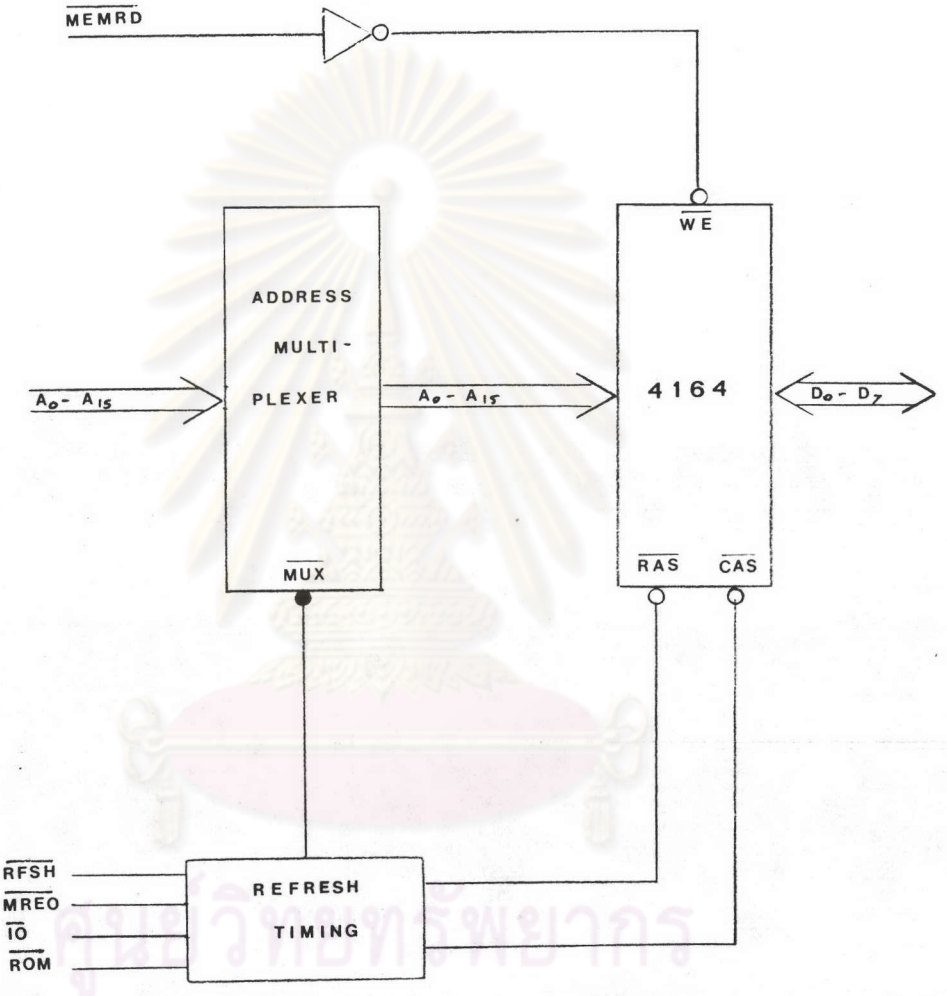


รูปที่ 4.14 แสดงวงจรหน่วยความจำรอม

จากรูปที่ 4.14 จะพบว่าขา Vpp และ ขา $\overline{PGM}$ ซึ่งเป็นขาที่ใช้งานเกี่ยวกับการบันทึกลงหน่วยความจำ (PROGRAMMING) จะต่อเข้ากับ Vcc เพราะจะใช้ทำหน้าที่อ่านอย่างเดียวเท่านั้น และขา $\overline{OE}$ ซึ่งปกติจะต่อเข้ากับสัญญาณ $\overline{MEMRD}$ จะถูกต่อลงกราวด์ ด้วยเหตุผลที่ต้องการให้มีความเร็วในการอ่านสูงขึ้น เนื่องจากไอซีเบอร์ 2764 ที่มีขายในท้องตลาดเมืองไทย มักจะมีความเร็วต่ำกว่าข้อกำหนด คือ 200 nS ดังนั้น เพื่อให้แน่ใจว่าสามารถทำงานได้เร็วทันการอ่านของซีพียู ซึ่งมีขนาดความเร็ว 4 เมกกะเฮิรทซ์ หรือ 250 nS ได้ จึงต่อลงกราวน์ไว้ .

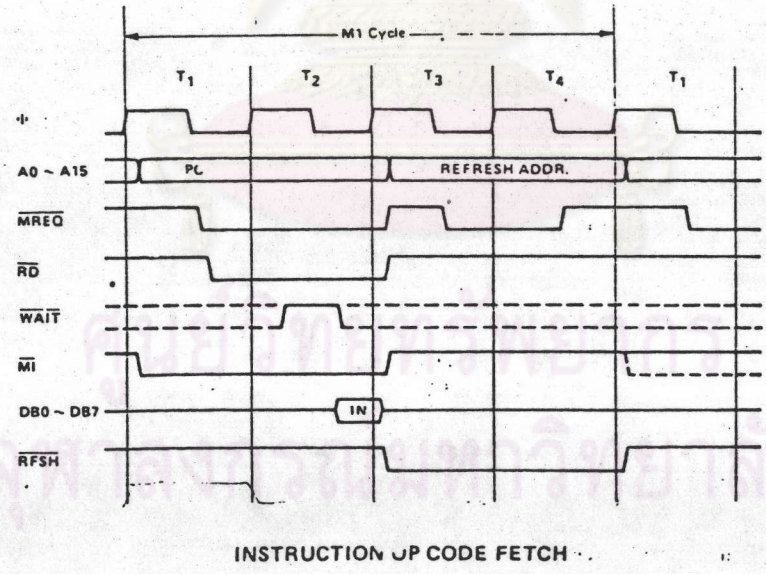
4.5.2 การออกแบบวงจรหน่วยความจำชนิดไดนามิคแรม

เมื่อออกแบบใช้หน่วยความจำแรม ที่ใช้แบบไดนามิคแรม แล้วจำเป็นอย่างยิ่ง ที่จะต้องมีวงจรที่ใช้ สำหรับสร้างสัญญาณเพื่อการรีเฟรชหน่วยความจำ และวงจรมัลติเพลกเซอร์ เพื่อ MULTIPLEX ADDRESS BUS ดังแสดงในรูปที่ 4.15



รูปที่ 4.15 แสดงบล็อกไดอะแกรมของวงจรหน่วยความจำไดนามิคแรม

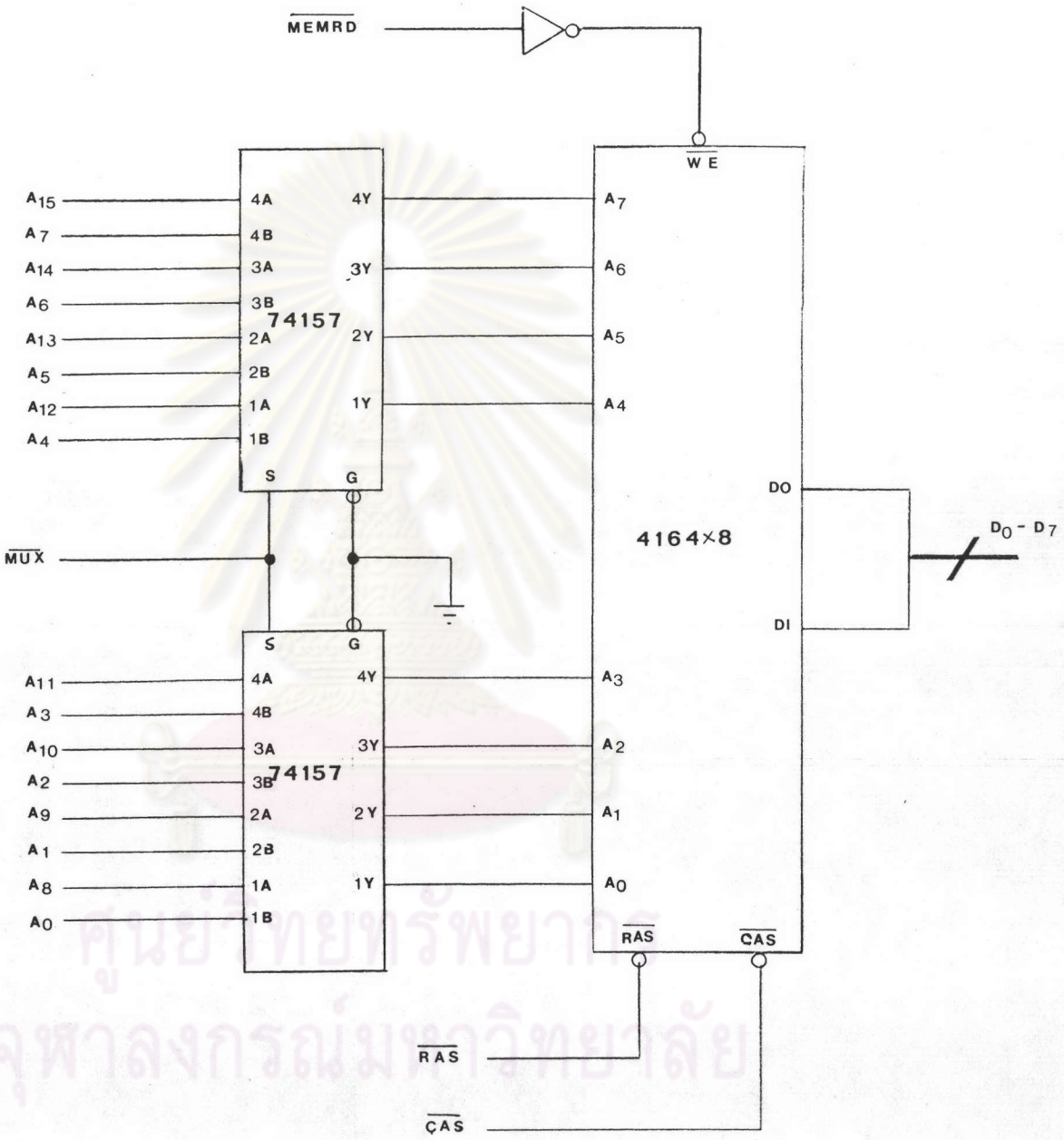
จากรูปที่ 4.15 จะพบว่าในส่วนของแอดเดรสบัสของไอซี 4164 จะมีขาแอดเดรส เพียง 8 ขาเท่านั้น ดังนั้นจึงจำเป็นต้องใช้วงจร ADDRESS MULTIPLEXER ช่วยในการมัลติเพลกเซอร์ ให้ ADDRESS A0-A15 เข้าไปครั้งละ 8 บิท คือเป็นลักษณะ ADDRESS A0-A7 ครั้งหนึ่งและ A8-A15 อีกครั้งสลับกันไปโดยมีสัญญาณ $\overline{MUX}$ เป็นตัวควบคุม และตรงสัญญาณ $\overline{WR}$ จะใช้ $\overline{MEMRD}$ โดยผ่านตัว INVERTER เป็นสัญญาณ MEMRD ต่อเข้าแทนที่จะใช้สัญญาณ $\overline{MEMWR}$ ต่อ เนื่องจากเหตุผลของการออกแบบวงจรรีเฟรช สำหรับวงจรรีเฟรชนี้จะใช้สัญญาณ $\overline{RFSH}$ ซึ่งได้จากขาของซีพียู Z80 มาใช้ แต่จะใช้ร่วมกับสัญญาณ $\overline{MREQ}$ โดยจะใช้ ADDRESS A0-A7 ทำหน้าที่รีเฟรช ในช่วงเกิดสัญญาณ $\overline{RFSH}$ และ $\overline{MREQ}$ เมื่อมีการอ่านคำสั่ง ดังแสดงในรูปที่ 4.16 สำหรับสัญญาณ $\overline{I/O}$ และ $\overline{ROM}$ จะใช้เพื่อทำหน้าที่ DISABLE ไดนามิคแรม เมื่อซีพียูทำงานในส่วนไอโอและอ่านหน่วยความจำ ROM โดยที่ช่วงนี้จะทำให้ $\overline{CAS}$ ไม่ ACTIVE คือเป็นสภาวะ "1" ทำให้ DISABLE ไดนามิคแรม



รูปที่ 4.16 แสดงไทม์มิ่งของสัญญาณขณะอ่านคำสั่ง (INSTRUCTION FETCH) และสัญญาณสำหรับรีเฟรช

4.5.3 การออกแบบวงจรมัลติเพลกเซอร์

จะพิจารณาใช้ไอซีเบอร์ 74 LS157 ซึ่งเป็น 2 LINE TO 1 LINE DATA SELECTOR/MULTIPLEXER ทำหน้าที่ MULTIPLEX ADDRESS โดยจะใช้ถึง 2 ตัว ดังแสดงในรูปที่ 4.17

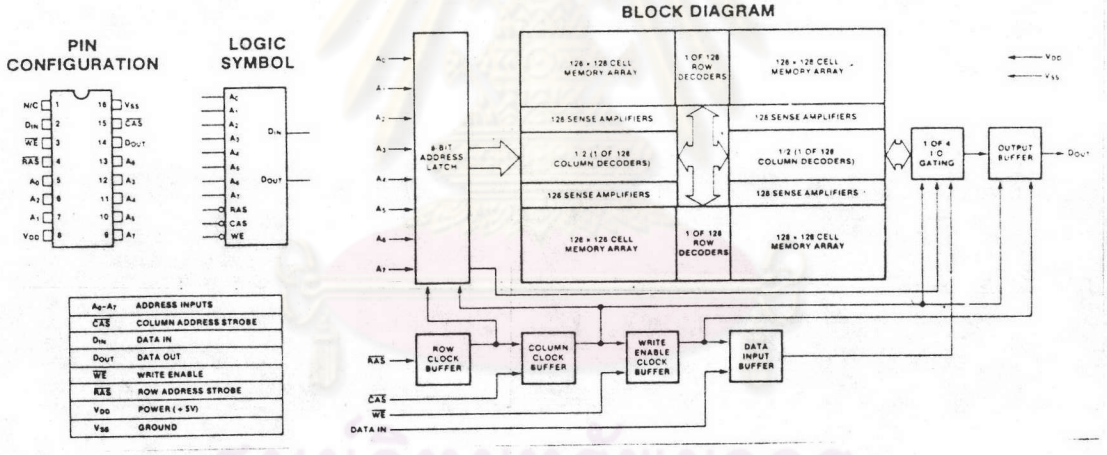


รูปที่ 4.17 แสดงวงจร MULTIPLEXER ADDRESS BUS สำหรับหน่วยความจำไดนามิคแรม

จากรูป จะเห็นว่าสัญญาณ $\overline{MUX}$ จะเป็นตัวควบคุมการเลือกแอดเดรสสู่หน่วยความจำและมีสัญญาณ $\overline{RAS}$ เป็นตัวเลือกแถว (ROW) และ $\overline{CAS}$ จะเป็นตัวเลือกคอลัมน์ (COLUMN) ในหน่วยความจำ

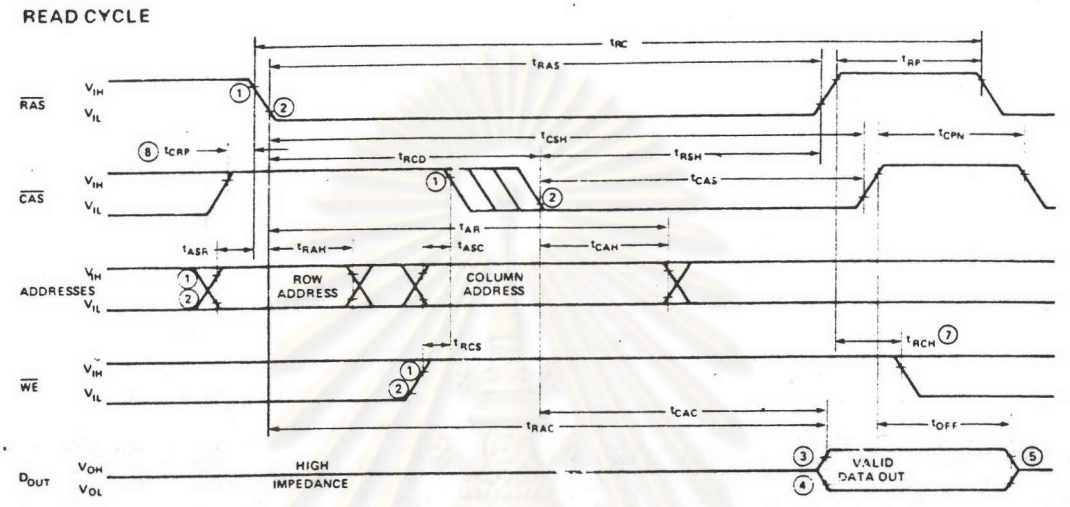
4.5.4 การออกแบบวงจรรีเฟรช หน่วยความจำไดนามิคแรม

ในที่นี้จะพิจารณาใช้ไอซีเบอร์ 4164 ซึ่งเป็นไดนามิคแรม ขนาด 64 กิโลไบท์ 1 บิท ซึ่งจะใช้ 8 ตัว ก่อนที่จะออกแบบวงจรรีเฟรช ควรจะทราบถึงลักษณะของไอซีเบอร์ 4164 และไทม์มิ่งของสัญญาณที่จะใช้ สำหรับการอ่านบันทึกหน่วยความจำและรีเฟรชด้วย รายละเอียดข้อกำหนดของไอซี 4164 จะแสดงในภาคผนวก ข

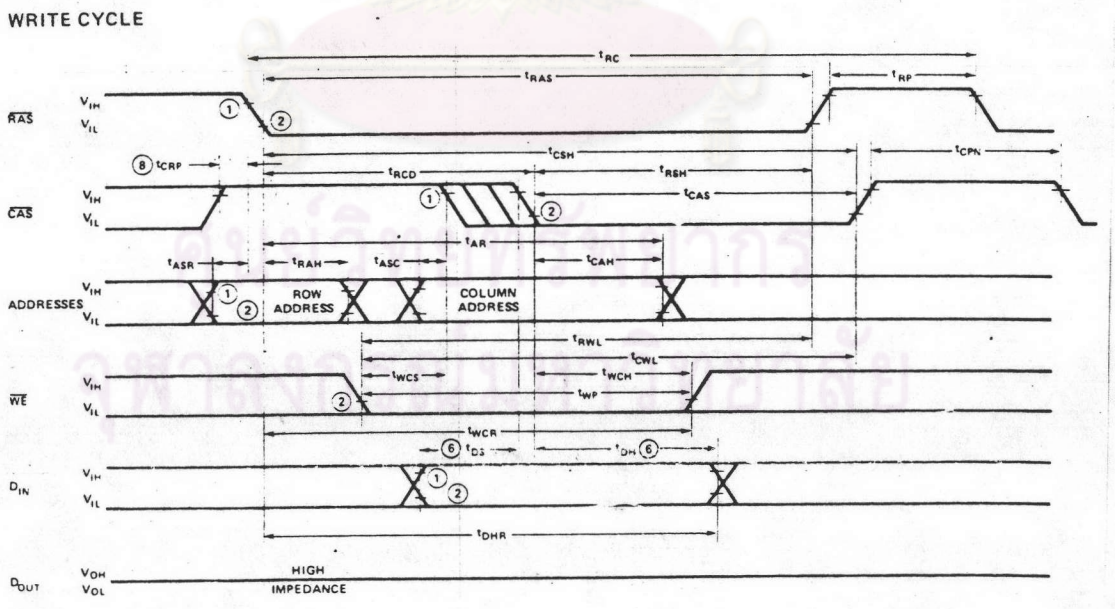


รูปที่ 4.18 แสดงลักษณะของหน่วยความจำ 4164 และ โครงสร้างภายใน

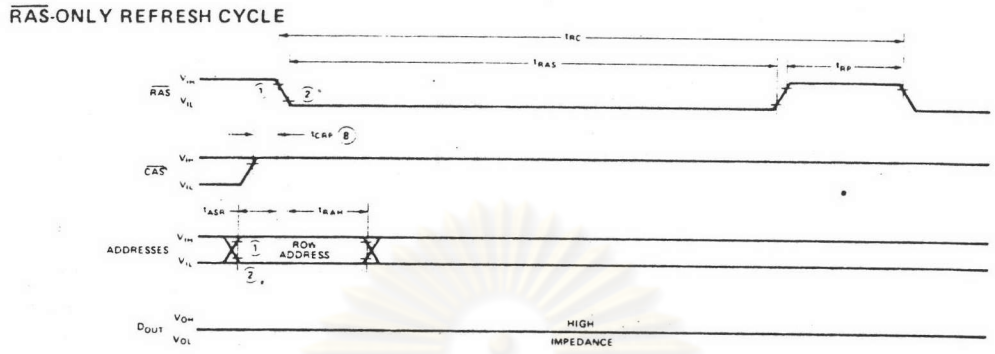
จากคุณสมบัติของไอซีเบอร์ 4164 จะทำการรีเฟรช 128 REFRESH CYCLES ภายใน 2 mS และจะใช้เฉพาะสัญญาณ $\overline{RAS}$ เท่านั้นทำการรีเฟรช และใช้สัญญาณ $\overline{CAS}$ ควบคุมการนำข้อมูลออกมาจากขาข้อมูลออก เมื่อ $\overline{CAS}$ เป็น LOW และขาข้อมูลออกจะเป็นสภาวะ HIGH IMPEDANCE เมื่อ $\overline{CAS}$ เป็น HIGH สำหรับสัญญาณและไทม์มิ่งของการอ่านการบันทึกและการรีเฟรชหน่วยความจำจะแสดงในรูปถัดไป



รูปที่ 4.19 แสดงสัญญาณและไทม์มิ่งของการอ่านหน่วยความจำ

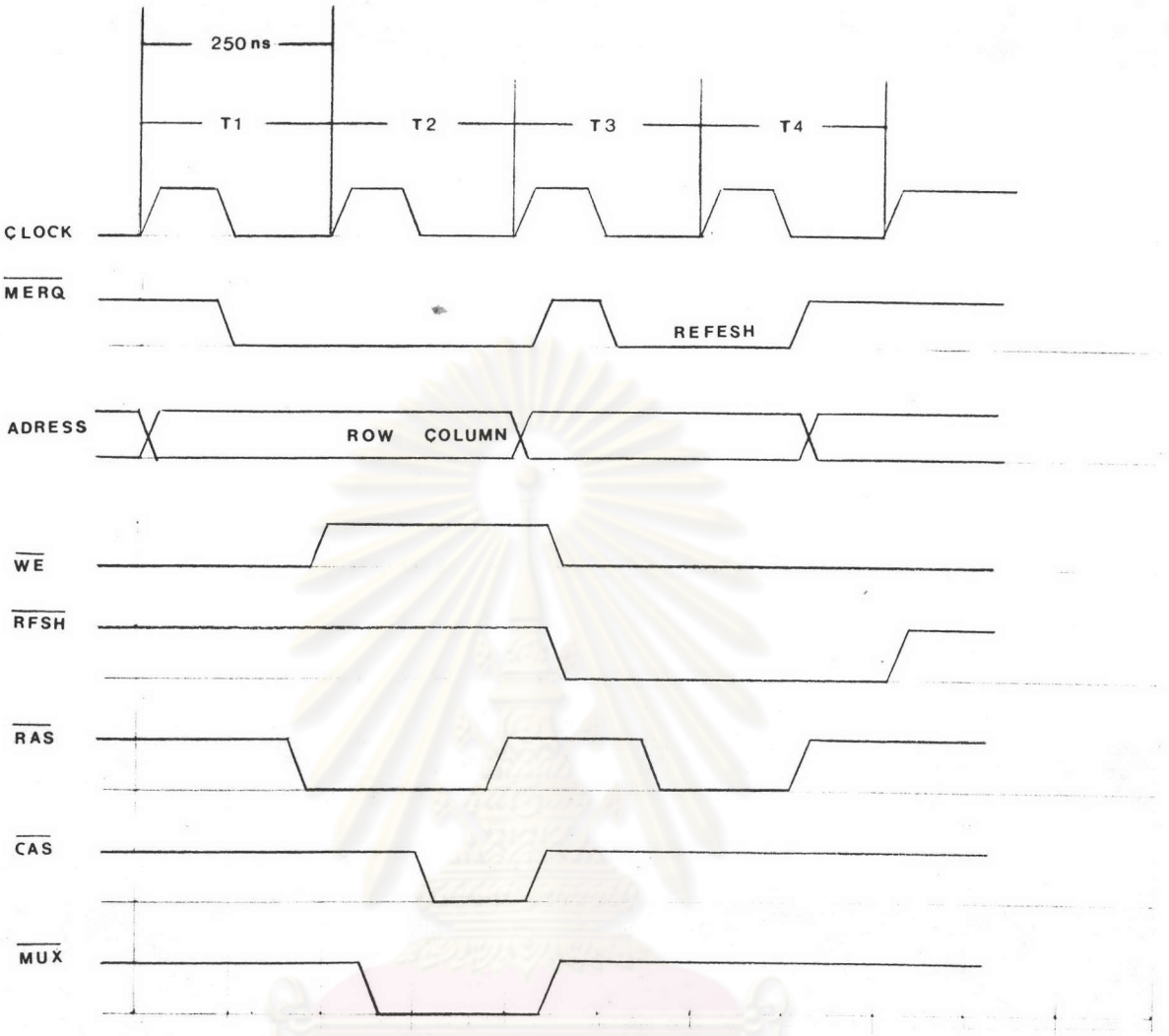


รูปที่ 4.20 แสดงสัญญาณและไทม์มิ่งของการบันทึกหน่วยความจำ



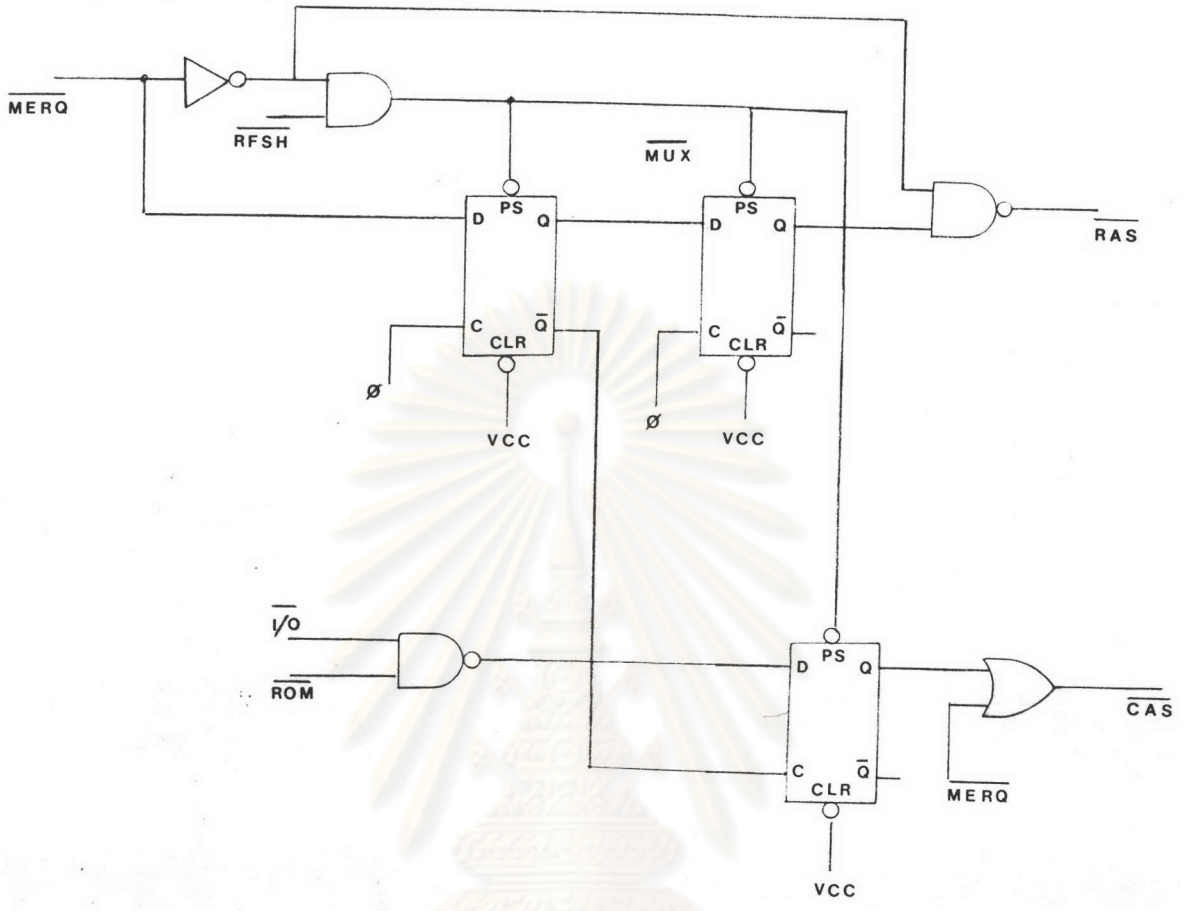
รูปที่ 4.21 แสดงสัญญาณและไทม์มิ่งของการรีเฟรช

จากรูปที่ 4.19 และ รูปที่ 4.20 จะเห็นว่าช่วงการอ่าน หรือการบันทึกหน่วยความจำ จะมีสัญญาณ $\overline{RAS}$ เกิดก่อนสัญญาณ $\overline{CAS}$ ดังนั้นสัญญาณแอดเดรสช่วงแรกจะเป็นการเลือก ROW ADDRESS ช่วงหลังจะเป็นการเลือก COLUMN ADDRESS ดังนั้น จากวงจรรูปที่ 4.17 จะต้องให้สัญญาณ $\overline{MUX}$ ในช่วงแรกเป็น HIGH เพื่อจะได้เลือก ROW ADDRESS ถูกต้องและช่วงหลังจะเป็น HIGH เพื่อเลือก COLUMN ADDRESS และจากรูปที่ 4.16 จะพบว่าซีพียูจะทำการอ่านในช่วงสัญญาณนาฬิกาสองไซเกิลแรกขณะเดียวกันในการบันทึกก็เหมือนกัน สำหรับรูปที่ 4.21 จะแสดงให้เห็นว่าจะใช้สัญญาณ $\overline{RAS}$ สำหรับการรีเฟรชโดยเฉพาะ และ แอดเดรส ช่วงนั้นจะเป็น ROW ADDRESS คือ A0-A7 แต่ที่ใช้งานจริงจะใช้เพียง A0-A6 เท่านั้น (จากคุณสมบัติของซีพียูและหน่วยความจำ) ก็สามารถรีเฟรชได้ จากเหตุผลทั้งหมดทำให้สามารถออกแบบสัญญาณและไทม์มิ่งสำหรับการรีเฟรชได้ ดังแสดงในรูปที่ 4.22



รูปที่ 4.22 แสดงสัญญาณและไทม์มิ่งของการรีเฟรชหน่วยความจำ

จากรูป จะเห็นว่าจะใช้สัญญาณ $\overline{RAS}$ สำหรับการเลือก ROW ADDRESS โดยช่วงนี้สัญญาณ $\overline{MUX}$ จะเป็น HIGH และใช้สัญญาณ $\overline{CAS}$ เลือก COLUMN ADDRESS โดยช่วงนี้สัญญาณ $\overline{CAS}$ จะเป็น HIGH และแอดเดรสสำหรับช่วงนี้จะเป็นแอดเดรสสำหรับการรีเฟรช คือ A0-A6 เท่านั้น จากข้อกำหนดของสัญญาณและไทม์มิ่งนี้สามารถออกแบบวงจรสำหรับการรีเฟรชหน่วยความจำได้ ดังแสดงในรูปที่ 4.23

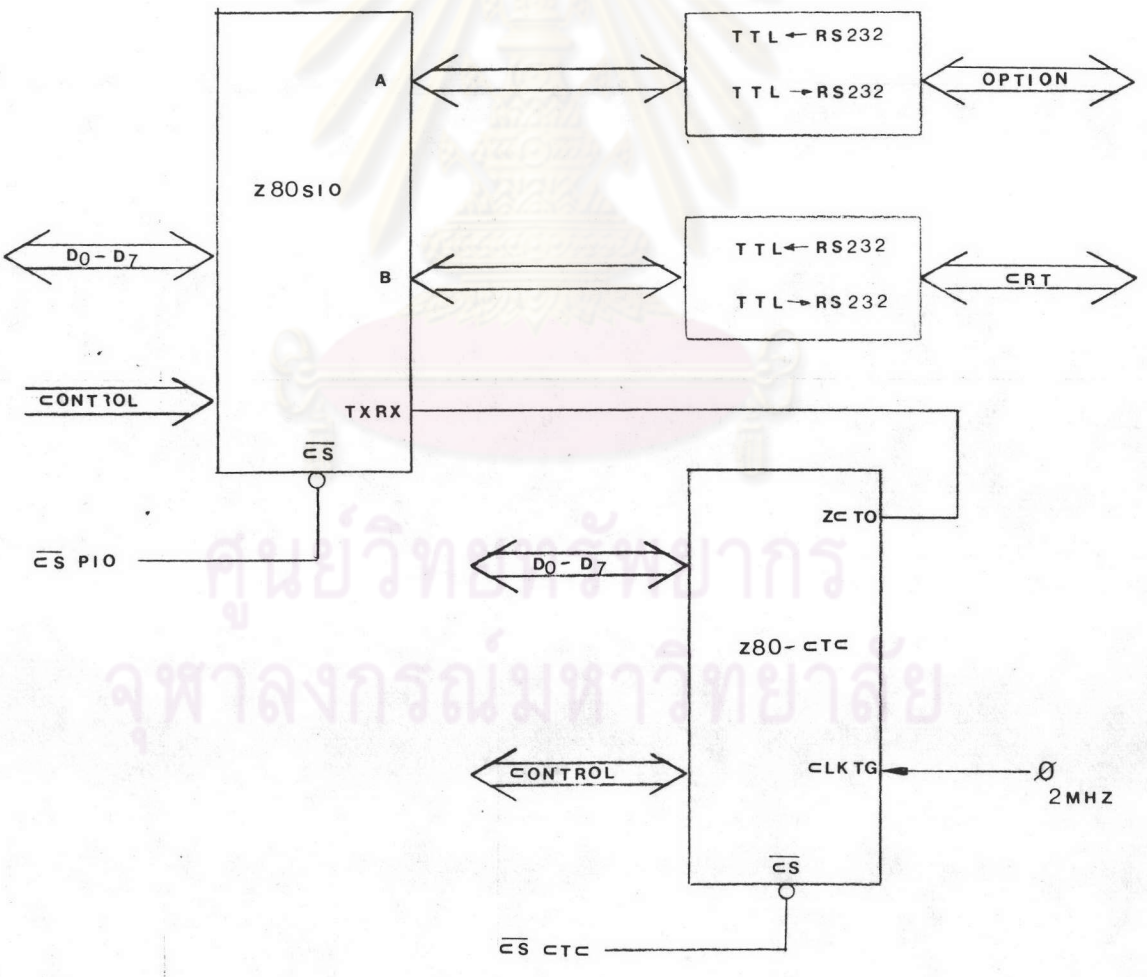


รูปที่ 4.23 แสดงวงจรสำหรับการรีเฟรช (REFRESH) หน่วยความจำ

จากรูป จะได้สัญญาณ $\overline{MUX}$ ไปควบคุมวงจรมัลติเพล็กเซอร์ (MULTIPLEXER) สัญญาณ $\overline{RAS}$ $\overline{CAS}$ ไปควบคุมการรีเฟรชหน่วยความจำ สัญญาณ $\overline{MERQ}$ $\overline{RFSH}$ และ ϕ ซึ่งเป็นสัญญาณนาฬิกาขนาด 4 เมกกะเฮิรทซ์ จะเป็นส่วนควบคุมวงจรการรีเฟรช ซึ่งมาจากส่วนซีพียู สำหรับสัญญาณ $\overline{I/O}$ และ $\overline{ROM}$ จะมาจากส่วนอินเตอร์เฟสตัวขับจานแม่เหล็ก กับสัญญาณเลือก หน่วยความจำรอมตามลำดับ เพื่อทำการ DISABLE หน่วยความจำแรมโดยให้ขา $\overline{CAS}$ เป็น HIGH ตลอด ถ้าสัญญาณ $\overline{I/O}$ หรือ $\overline{ROM}$ เกิดมีสภาวะเป็น LOW

4.6 การออกแบบวงจรอินเตอร์เฟสซีอาร์ทีเทอร์มินอล

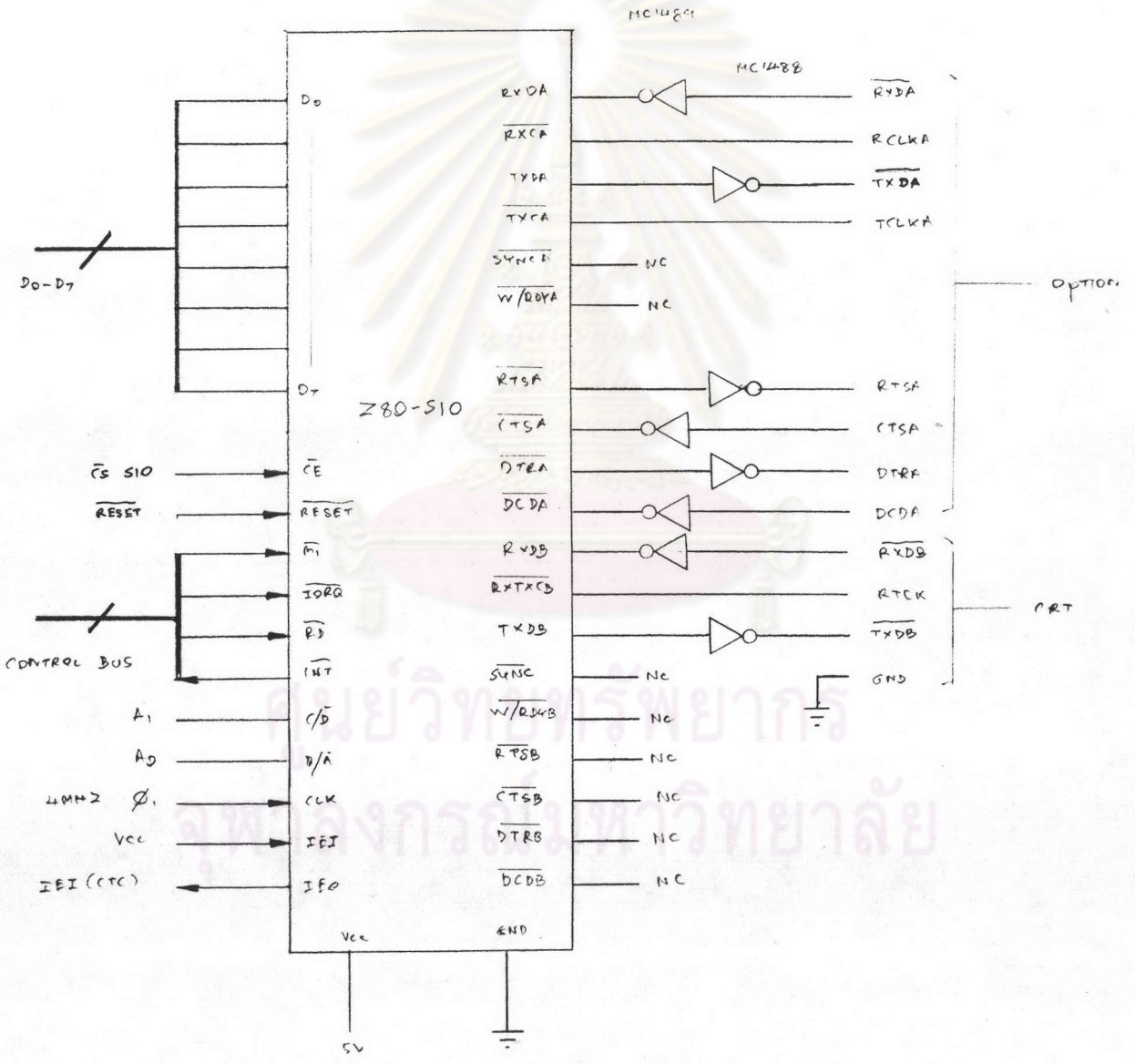
สำหรับการอินเตอร์เฟสกับซีอาร์ทีเทอร์มินอลซึ่งเป็นแบบ RS232 นี้ จะพิจารณาเลือกใช้ไอซีเบอร์ Z80A-SIO ซึ่งมี 2 พอร์ท โดยจะเลือกพอร์ท B ทำหน้าที่นี้ ส่วนที่เหลือพอร์ท A จะใช้เป็น OPTION สำหรับอินเตอร์เฟสกับอุปกรณ์ภายนอกอื่นๆ วงจรอินเตอร์เฟสซีอาร์ทีเทอร์มินอลและ OPTION จะแสดงเป็นบล็อกไดอะแกรม ดังรูปที่ 4.24 (สำหรับรายละเอียดของไอซีเบอร์ Z80A-SIO และ CTC สามารถหาดูได้จากคู่มือของไอซีตระกูล Z-80 ของ INTEL) และรายละเอียดของซอฟท์แวร์ที่ใช้ควบคุม จะอยู่ในหัวข้อการออกแบบโปรแกรมระบบ



รูปที่ 4.24 แสดงโครงสร้างของวงจรอินเตอร์เฟส RS232C

จากรูปจะพบว่า โครงสร้างของวงจรอินเตอร์เฟส RS232C สำหรับซีอาร์ทีเทอร์มินอลและ OPTION จะมีส่วนประกอบที่สำคัญ 3 ส่วนคือ วงจรอินเตอร์เฟสกับวงจรสำหรับสร้างสัญญาณนาฬิกากำหนดความเร็ว ในการรับส่งข้อมูล (BAUD RATE GENERATOR) และวงจรสำหรับการเปลี่ยนระดับสัญญาณจาก TTL (5V) เป็น RS232C(12V) ขณะส่งสัญญาณ และจาก RS232C (12V) เป็น TTL (5V) ขณะรับสัญญาณ

4.6.1 วงจรอินเตอร์เฟส RS232C

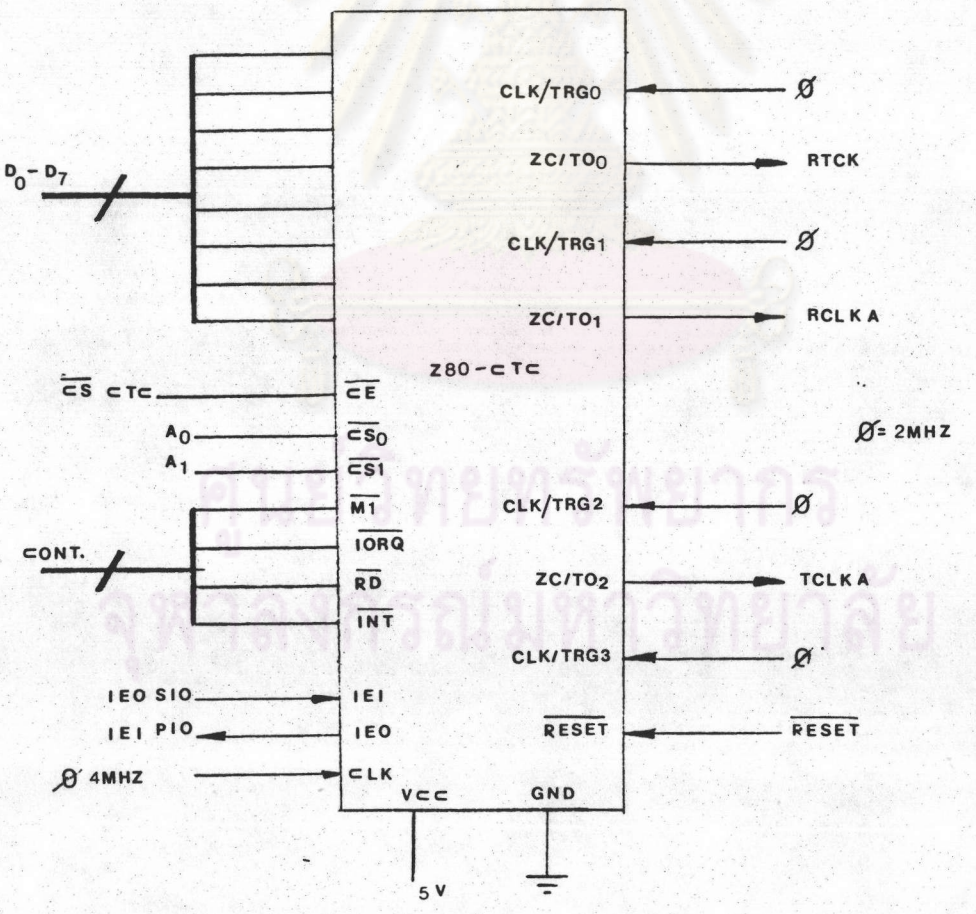


รูปที่ 4..25 วงจรอินเตอร์เฟส RS232C

จากรูปจะเห็นว่าพอร์ท A จะใช้เป็นพอร์ท RS232C สำหรับเป็น OPTION และพอร์ท B จะใช้เป็นพอร์ทต่อกับซีอาร์ทีเทอร์มินอลโดยจะใช้เพียง 3 ขาเท่านั้นในการรับส่งข้อมูลคือ ขา $\overline{RXDB}$ สำหรับรับข้อมูล ขา $\overline{TXDB}$ สำหรับส่งข้อมูลอีกขาจะเป็นกราวด์ ส่วนขา $\overline{RXTXCB}$ จะเป็นขารับสัญญาณนาฬิกากำหนดความเร็วการรับส่งข้อมูลจะต่อเข้ากับ ZC/TO_0 ซึ่งเป็นเอาท์พุทของ Z80-CTC โดยจะใช้ความเร็วในการรับส่งข้อมูลเท่ากันคือ 9600 BAUD

4.6.2 การออกแบบวงจร BAUD RATE GENERATOR

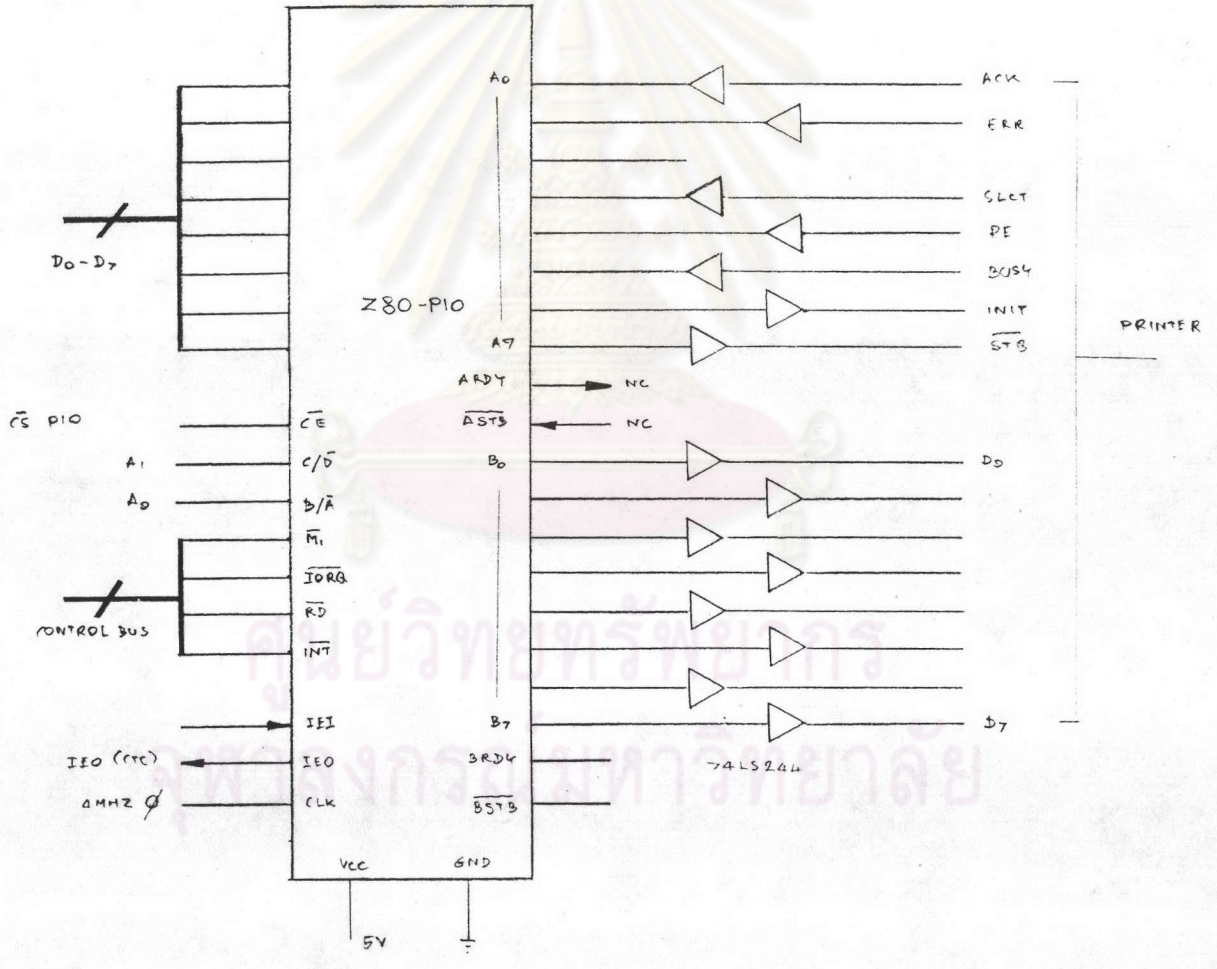
จะพิจารณาใช้ไอซีเบอร์ Z80-CTC โดยจะเลือกใช้ในลักษณะเคาน์เตอร์โหมด (COUNTER MODE) โดยจะใช้สัญญาณนาฬิกาซึ่งมีขนาดความเร็ว 2 เมกกะเฮิรทซ์ มาเข้าขา CLK/TRG และนำเอาสัญญาณนาฬิกาที่หารแล้วออกที่ขา ZC/TO_{0-2} ไปต่อเข้ากับขา $\overline{RXTXCB}$ ขา $\overline{RXCA}$ และขา $\overline{TXCA}$ ของ Z80-SIO วงจรที่ออกแบบจะแสดงในรูปที่ 4.26 (รายละเอียดของ Z80-CTC สามารถดูได้จากคู่มือไอซีตระกูล Z-80 ของ INTEL)



รูปที่ 4.26 แสดงวงจร BAUD RATE GENERATOR

4.7 การออกแบบวงจรอินเตอร์เฟสเครื่องพิมพ์

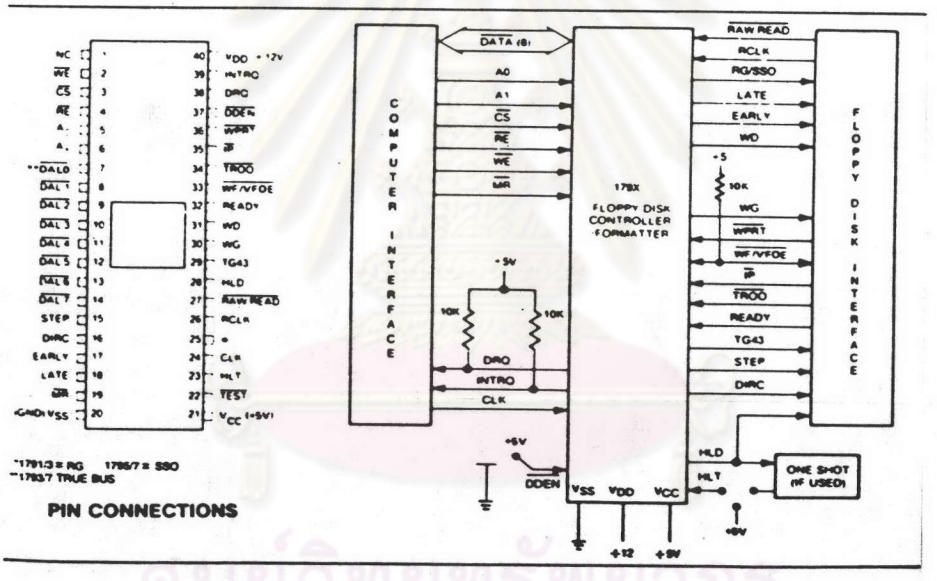
วงจรอินเตอร์เฟสเครื่องพิมพ์ที่ออกแบบนี้ จะเป็นแบบขนาน ดังนั้นจะพิจารณาเลือกใช้ Z80-PIO ซึ่งมีพอร์ทขนาน 2 พอร์ท โดยจะเลือกพอร์ท A ทำหน้าที่รับและส่งสัญญาณควบคุม ส่วนพอร์ท B จะใช้สำหรับรับส่งข้อมูลไปยังเครื่องพิมพ์ และเนื่องจากสายต่ออินเตอร์เฟสของเครื่องพิมพ์มีความยาว เพื่อป้องกันการสูญหายของข้อมูลจึงใช้ไอซีเบอร์ 74LS244 ทำหน้าที่เป็นบัสไดร์เวอร์ ดังแสดงในรูปที่ 4.27 (รายละเอียดของไอซีเบอร์ Z-80 PIO สามารถหาได้จากคู่มือไอซีตระกูล Z-80 ของ INTEL) ซอฟท์แวร์ที่ใช้ควบคุมจะกล่าวถึงในหัวข้อการออกแบบโปรแกรมระบบ



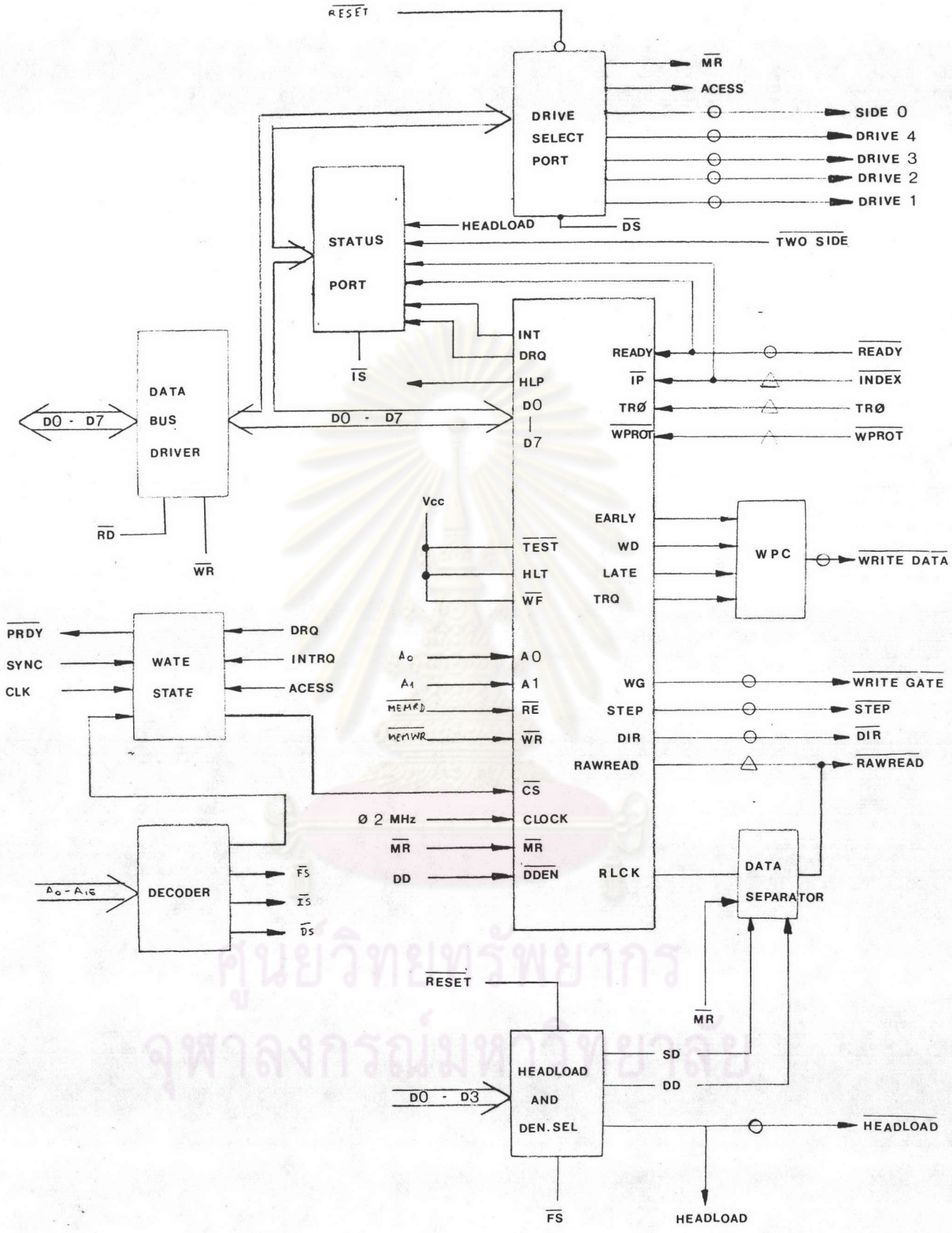
รูปที่ 4.27 แสดงวงจรอินเตอร์เฟสแบบขนาน

4.8 การออกแบบวงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก

เพื่อลดความยุ่งยาก ในการออกแบบวงจรตรรก จะใช้ ไอซีเบอร์ FD1791 ทำหน้าที่เป็นตัวควบคุมในวงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก และยังสามารถใช้งานในลักษณะไอบีเอ็มฟอร์แมททั้งแบบซิงเกิลเดนซิตี้และดับเบิลเดนซิตี้ด้วย บล็อกไดอะแกรมของวงจรที่ใช้ FD1791 จะแสดงในรูปที่ 4.28 สำหรับรายละเอียดของไอซีเบอร์ FD1791 จะแสดงในภาคผนวก ค วงจรอินเตอร์เฟสตัวขับจานแม่เหล็กที่ออกแบบจะแสดงเป็นบล็อกไดอะแกรมในรูปที่ 4.29



รูปที่ 4.28 แสดงบล็อกไดอะแกรมของไอซี FD1791



รูปที่ 4.29 แสดงบล็อกไดอะแกรมของวงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก

จากรูปที่ 4.29 จะเห็นว่าไอซีเบอร์ FD1791 จะทำหน้าที่เป็นส่วนควบคุมในวงจรอินเตอร์เฟสตัวขับจานแม่เหล็กการทำงานของวงจรจะสมบูรณ์ได้จะต้องประกอบด้วยวงจรรอบนอกซึ่งทำหน้าที่ต่าง ๆ เช่น

1. วงจร BUS DRIVER ทำหน้าที่เป็นตัวกั้น (BUFFER) และขยายสัญญาน ให้มีความแรงพอที่จะจ่ายให้กับองค์ประกอบไอซีที่ต่อร่วม เนื่องจากระบบบัสของ FD1791 เป็น INVERT BUS ดังนั้นไอซีซึ่งทำหน้าที่เป็น BUS DRIVER จะเป็น INVERTER ด้วย ในที่นี้จะใช้ไอซีเบอร์ 74LS240 2 ตัว คือ ทั้งตอนอ่านและบันทึก สัญญาณที่จะใช้ในการควบคุมก็คือ $\overline{\text{MEMRD}}$ และ $\overline{\text{MEMWR}}$ ซึ่งเกิดอยู่ในช่วงแอดเดรส FBF8-FBFF

2. วงจร WAIT STATE จะทำหน้าที่สร้างสัญญาณ $\overline{\text{PRDY}}$ เพื่อ WAIT ซีพียูขณะที่ไอซี FD1791 กำลังทำหน้าที่อ่าน-บันทึกข้อมูลจากจานแม่เหล็ก หลังจากอ่าน-บันทึกข้อมูลเสร็จจะส่งสัญญาณ DRQ และ INTRO ที่เป็น HIGH ซึ่งทำให้ ซีพียูสามารถอ่าน-บันทึกข้อมูลไปยังไอซี FD1791 ได้ ขณะเดียวกันยังสามารถใช้สัญญาณ ACESS เพื่อบอกให้ซีพียูสามารถอ่านหรือบันทึกข้อมูลได้เช่นกัน การทำงานของวงจรนี้ จะเกี่ยวกับ DATA REGISTER ดังนั้นจำเป็นต้องใช้ ADDRESS SELECT ของ DATA REGISTER มาเกี่ยวข้องด้วย ขณะเดียวกันจังหวะการทำงานของวงจรนี้ จะถูกควบคุมด้วยสัญญาณ $\overline{\text{SYNC}}$ และ CLOCK จากซีพียูด้วย

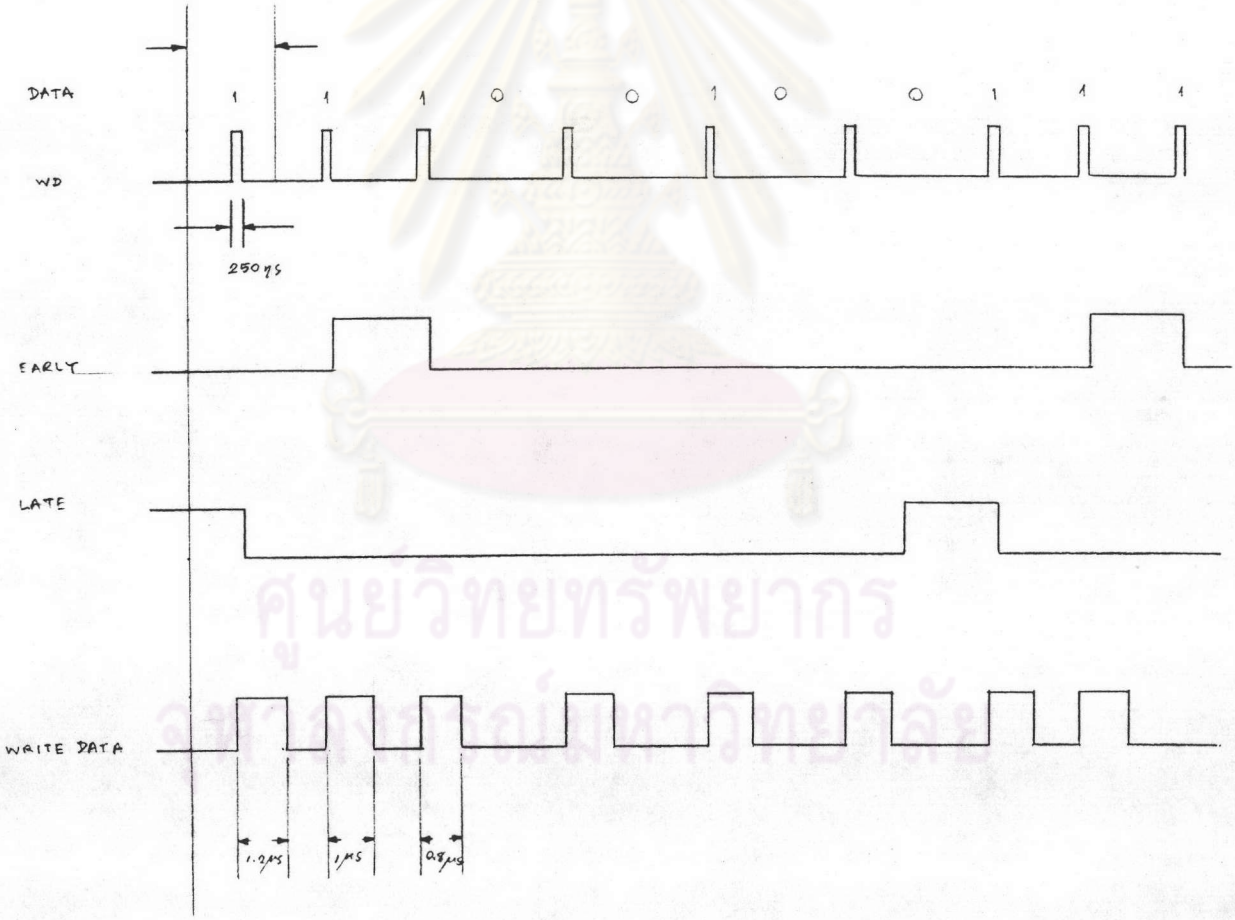
3. วงจร DECODER ทำหน้าที่สร้างสัญญาณ เพื่อเลือกองค์ประกอบให้ทำงานในหน้าที่ต่าง ๆ เช่น $\overline{\text{FS}}$ (FUNCTION SELECT) จะเลือก DENSITY และ สั่ง HEADLOAD $\overline{\text{IS}}$ (IN STATUS) จะเลือกพอร์ทเพื่ออ่านสภาวะการทำงานของตัวขับจานแม่เหล็กและ $\overline{\text{DS}}$ (DRIVE SELECT) จะเลือกพอร์ทเพื่อเลือกตัวขับจานแม่เหล็กและเลือกทำงานด้านใดของจาน ขณะเดียวกันยังทำหน้าที่เลือกไอซี FD1791 ด้วย

4. วงจรบันทึกข้อมูล (WRITE DATA) จะทำหน้าที่ชดเชยและควบคุมความถี่ ของสัญญาณบันทึกข้อมูลให้มีความถูกต้อง ในขณะที่จานแม่เหล็กหมุน

5. วงจรอ่านข้อมูลและแยกข้อมูล (DATA SEPERATER) จะทำหน้าที่แยกสัญญาณข้อมูลออกจากสัญญาณนาฬิกาโดยสร้างสัญญาณ RCLK ให้กับไอซี FD1791 จากวงจรที่ประกอบทั้งหมด จะเห็นว่าวงจรแยกข้อมูลและวงจรบันทึกข้อมูล เป็นวงจรที่น่าสนใจซึ่งจะแสดงรายละเอียดในหัวข้อต่อไป

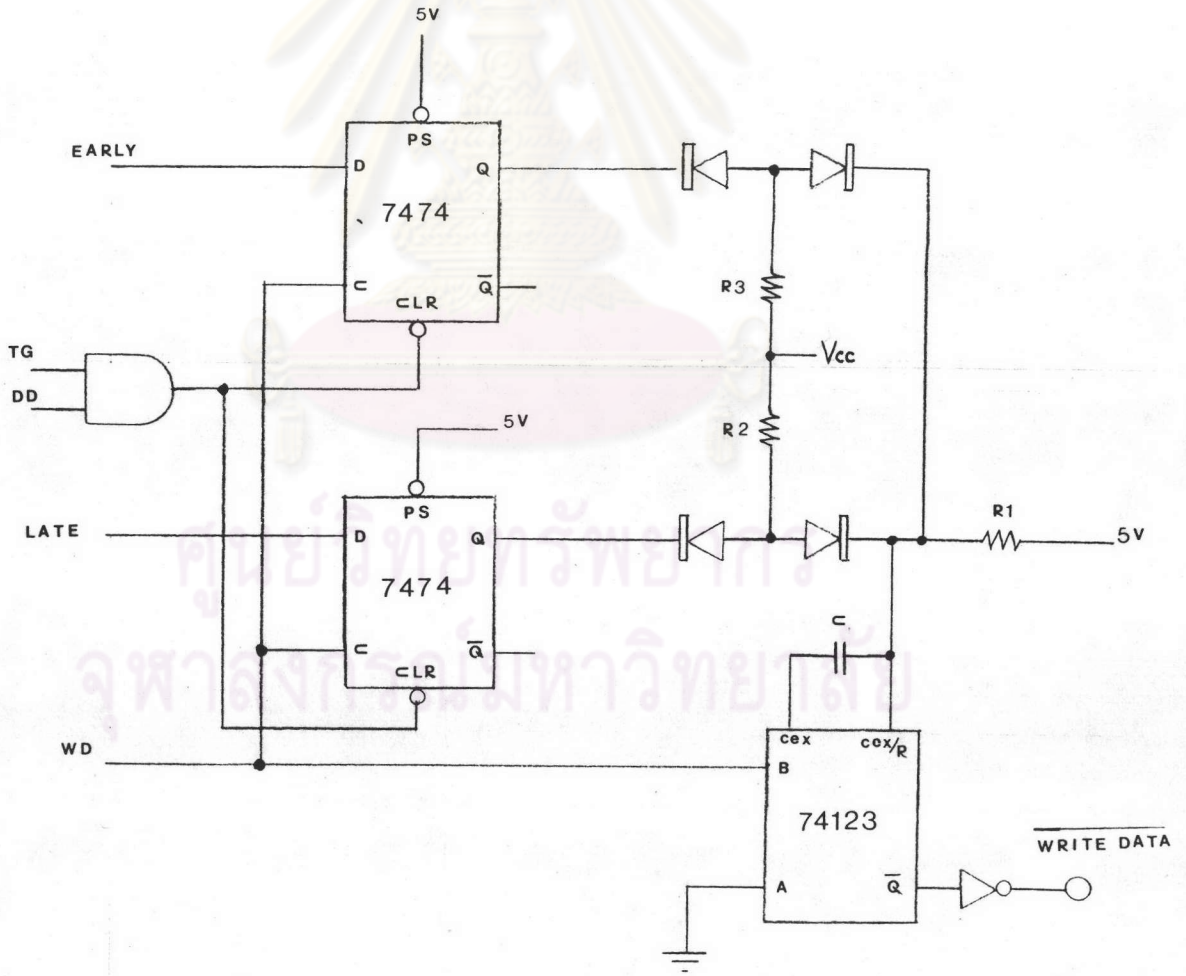
4.8.1 การออกแบบวงจรบันทึกข้อมูล (WRITE DATA)

สัญญาณข้อมูลที่บันทึกลงจานแม่เหล็ก จะออกจากขา WD (WRITE DATA) ของไอซี FD1791 เนื่องจากการหมุนของจานแม่เหล็กไม่คงที่ ดังนั้นจึงจำเป็นจะต้องใช้สัญญาณ EARLY และ LATE เพื่อใช้ควบคุมสัญญาณความถี่การบันทึกข้อมูลให้ถูกต้องตามสภาวะการหมุนของจานแม่เหล็ก โดยเฉพาะการบันทึกข้อมูลแบบ MFM (DOUBLE DENSITY) ในช่วงแทรคใน ๆ คือ ตั้งแต่ แทรค 44-76 สำหรับการบันทึกแบบ FM (SINGLE DENSITY) นั้นความถี่ของการบันทึกข้อมูลมีความถี่ต่ำกว่าแบบ MFM การบันทึกอาจใช้สัญญาณ WD โดยตรงก็ได้ เพียงแต่ผ่านวงจรโมโนสเตเบิลเพื่อขยายความกว้างของสัญญาน WD ช่วงบวกให้เหมาะสมเท่านั้น รูปแสดงสัญญาณการบันทึกข้อมูลและสัญญาณควบคุมแบบ MFM จะแสดงในรูปที่ 4.30



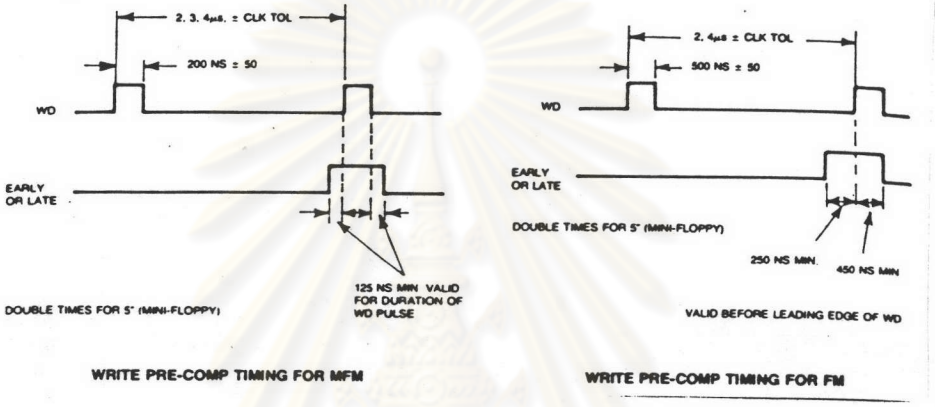
รูปที่ 4.30 แสดงสัญญาณการบันทึกข้อมูลและสัญญาณที่ใช้ควบคุม

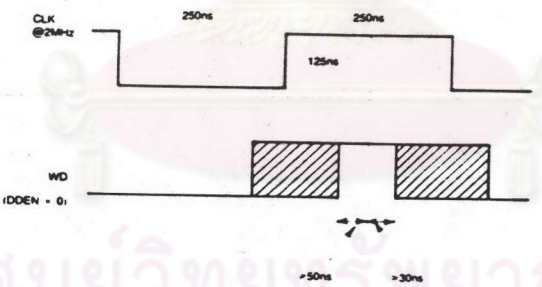
จากรูปจะพบว่าถ้าจานแม่เหล็กหมุนช้าจะมีสัญญาณ LATE เกิดขึ้น ดังนั้นการออกแบบให้สัญญาณการบันทึกข้อมูลจะช้าลงด้วย สำหรับการบันทึกแบบ MFM ซึ่งช่วงเวลาการบันทึกข้อมูลแต่ละบิทจะใช้เวลา 2 uS และ FM จะใช้ 4 uS (เทคนิคการบันทึกข้อมูลจะดูได้ในบทที่ 2) ดังนั้นควรออกแบบให้สัญญาณช่วงบวกช้าลง คือ 1.2 uS ถ้าตัวขับจานแม่เหล็กหมุนเร็วขึ้นจะทำให้สัญญาณ EARLY เกิดขึ้น ดังนั้นจะต้องออกแบบให้สัญญาณ WRITE DATA เร็วขึ้น คือ 0.8 uS ถ้าความเร็วของจานแม่เหล็กคงที่ จะใช้ความกว้างของพัลส์คงที่ด้วย คือ 1 uS สำหรับสัญญาณ WD นั้นจะมีความกว้างของช่วงบวกประมาณ 250 nS เนื่องจาก FD1791 ยังไม่มีสัญญาณการ WRITE DATA โดยตรง ดังนั้นจึงจำเป็นต้องอาศัยสัญญาณ WD EARLY LATE TG43 (TRACK GREATER THAN 43) และ DD (DOUBLE DENSITY) มาสร้างสัญญาณ WRITE DATA ดังแสดงในรูปที่ 4.31A และรูปสัญญาณ WRITE DATA จะแสดงในรูปที่ 4.31B และ 4.31C



รูปที่ 4.31A วงจรสร้างสัญญาณ WRITE DATA

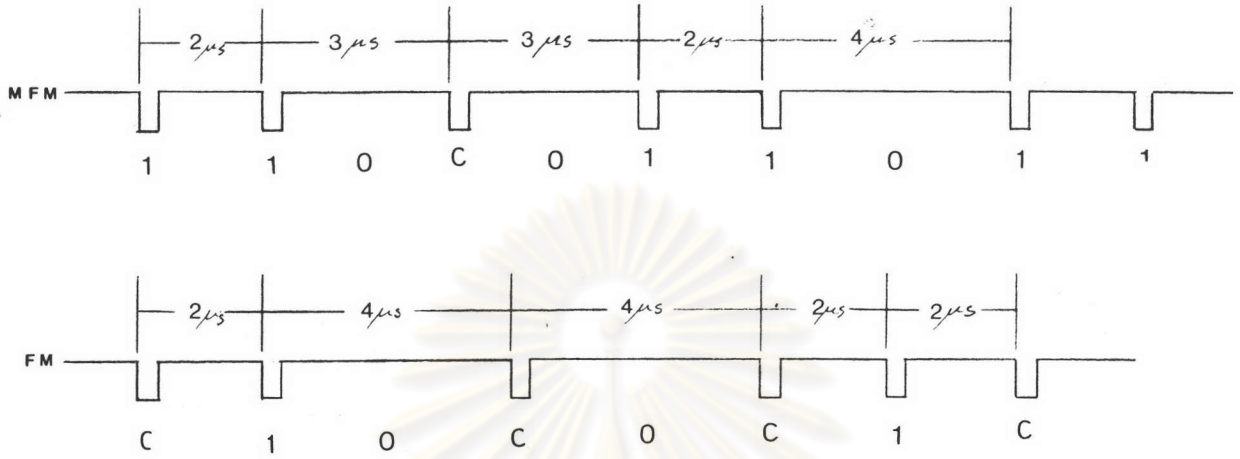
จากรูปที่ 4.31A จะเห็นว่า ใช้สัญญาณ EARLY LATE WD TG43 DD มาควบคุมความกว้างของสัญญาณ WD โดยวิธีการเลือกค่า R1 R2 R3 สำหรับต่อเป็นค่า R ให้กับวงจรโมโนสเตเบิล 74LS123 เพราะค่าความกว้างของพัลส์จะถูกควบคุมด้วยค่า RC จากวงจรจะเห็นว่าถ้าเป็นกรณี FM หรือ MFM ที่อยู่ในช่วงแทรคต่ำกว่า 43 แล้วค่า R จะคงที่ คือเท่ากับ R1//R2 แต่ถ้าเป็นกรณี MFM และทำงานอยู่ในช่วงแทรคมากกว่า 43 แล้วการทำงานจะแสดงได้ในตารางที่ 4.1





รูปที่ 4.31B แสดงสัญญาณ WRITE PRE-COM TIMING

จากรูปที่ 4.31B จะพบว่าสัญญาณ WRITE DATA (WD) จะมีอยู่ถึง 3 ขนาด คือ 2,3,4 uS สำหรับการบันทึกแบบ MFM และจะมี 2 ขนาด คือ 2,4 uS ถ้าเป็นการบันทึกแบบ FM เหตุที่ช่วงเวลาของสัญญาณ WD ต่างกันก็เนื่องมาจากการบันทึกข้อมูลที่ต่างกัน ซึ่งจะอธิบายได้ในรูปที่ 4.31 C

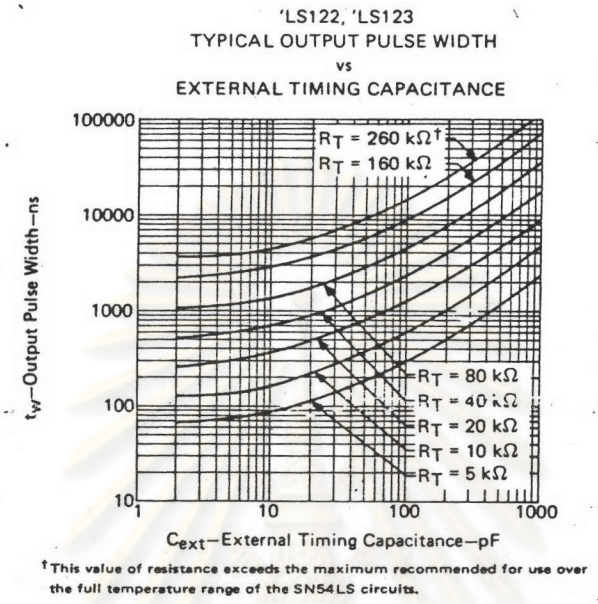


รูปที่ 4.31C แสดงสัญญาณ WD ที่ช่วงเวลาต่าง ๆ กันในการบันทึกข้อมูล

มอเตอร์	LATE	EARLY	R	PULSE WIDTH uS
หมุนปกติ	0	0	R1//R2	1
หมุนช้า	1	0	R1	1.2
หมุนเร็ว	0	1	R1//R2//R3	0.8

ตารางที่ 4.1 แสดงการทำงานของวงจรสร้างสัญญาณ WRITE DATA

กรณีในตารางนี้จะเป็นการทำงานการบันทึกข้อมูลแบบ MFM และบันทึกอยู่ในช่วงแทรค 44-76 การคำนวณ สามารถดูได้จาก คู่มือ 74LS123 ดังแสดงในรูปที่ 4.32

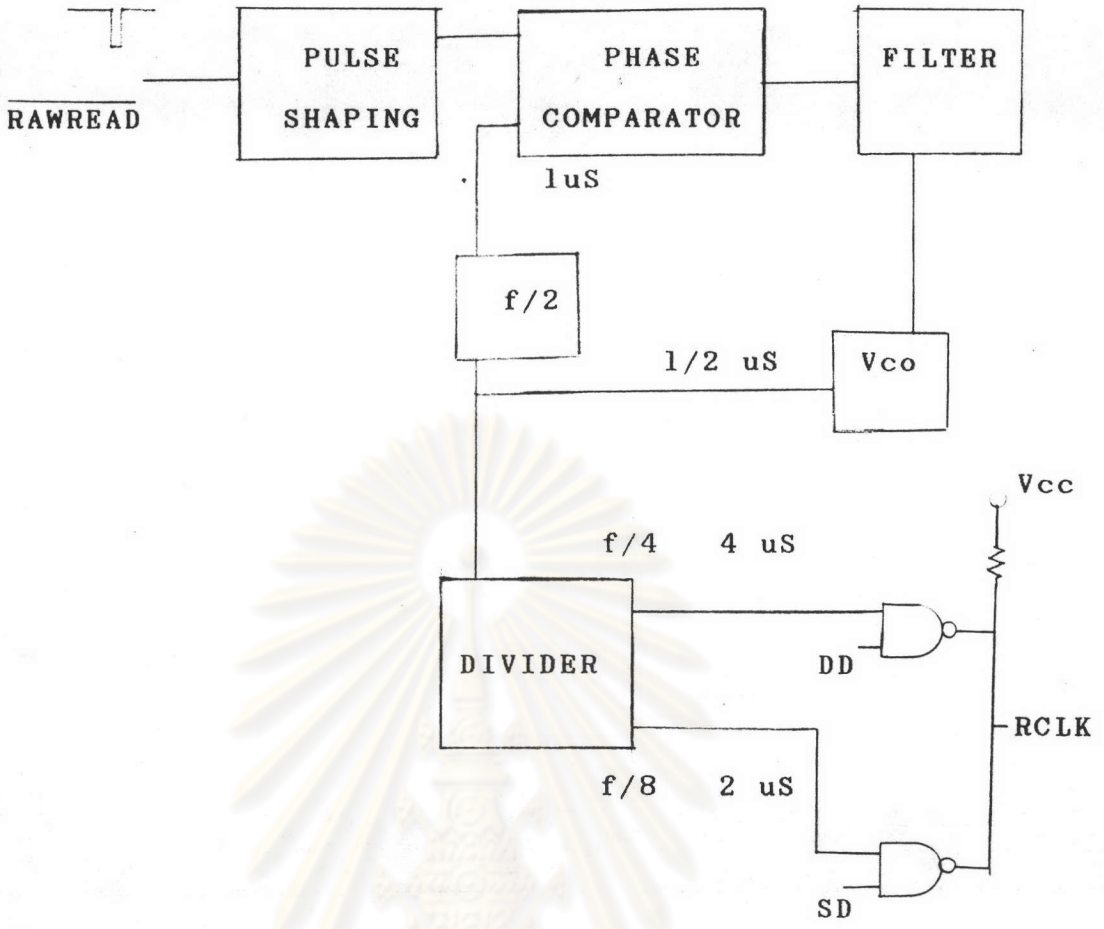


รูปที่ 4.32 แสดงกราฟการคำนวณหาค่า R

จากรูปที่ 4.32 สามารถคำนวณหาค่า R ในแต่ละช่วงเวลาได้ โดย กำหนดให้ค่า C = 115 pF ดังนั้นถ้าให้ R1 = 20 K จะได้ค่า R2 = 53.6 K และ R3 = 86.6 K จากการคำนวณทางคณิตศาสตร์

4.8.2 การออกแบบวงจรอ่านและแยกข้อมูล (DATA SEPERATOR)

ในทำนองเดียวกัน ขณะที่ FD1791 ทำการอ่านข้อมูลจากจาน-แม่เหล็กนั้นจานแม่เหล็กอาจหมุนไม่คงที่ อันเนื่องมาจากกลไกและระบบขับเคลื่อนต่าง ๆ ภายในตัวขับจานแม่เหล็กทำให้ ความถี่ของสัญญาณข้อมูลไม่คงที่ เท่ากับความถี่ที่ใช้ควบคุมในระบบ ดังนั้นจึงจำเป็นที่จะต้องหาทางแก้ปัญหานี้ ให้ข้อมูลที่อ่านมีความถูกต้องโดยอาศัยสัญญาณ RCLK เป็นตัวกำหนด เพื่อแยกสัญญาณข้อมูลออกจากสัญญาณที่อ่านเข้ามา ซึ่งสามารถแก้ปัญหานี้ได้โดยอาศัยวงจรเฟสล็อคลูป (PHASE LOCK LOOP). ดังแสดงในรูปที่ 4.33

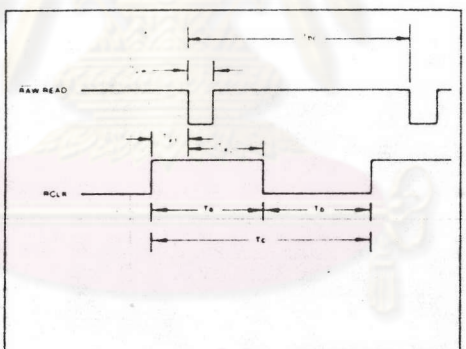


รูปที่ 4.33 แสดงวงจรเฟสล็อคลูปและวงจรสร้างสัญญาณ RCLK สำหรับใช้แยกข้อมูล

จากรูปที่ 4.33 จะพบว่าสัญญาณ $\overline{\text{RAWREAD}}$ จะเป็นสัญญาณที่อ่านมาจากตัวขับจานแม่เหล็ก ซึ่งยังมีลักษณะเป็นพัลส์เล็ก ๆ จึงต้องทำการขยายให้มีความกว้างขึ้นโดยผ่านวงจรแต่งรูปร่าง (PULSE SHAPING) ซึ่งอาจใช้ไอซีที่เป็นโมโนสเตเบิลทำหน้าที่นี้ จากนั้นจะทำการเปรียบเทียบกับความถี่ภายในระบบด้วยวงจร PHASE COMPARATOR ถ้ามอเตอร์หมุนเร็วทำให้ความถี่ที่อ่านเข้ามามีค่าสูง จะทำให้ได้แรงดัน (VOLTAGE) ซึ่งถ้าผ่านวงจรกรอง (FILTER) แล้วมีค่าสูงขึ้นเพื่อไปควบคุมวงจร OSC (OSCILLATOR) จะทำให้ OSCILLATE ความถี่สูงขึ้นจากเดิมด้วย เมื่อนำไปหารแล้วจะทำให้ได้สัญญาณ RCLK ที่ถูกต้อง

และ SYCRONIZE กับความถี่ของสัญญาณ $\overline{\text{RAWREAD}}$ ถ้ามอเตอร์หมุนช้าลงทำให้ความถี่ของข้อมูลที่อ่านเข้ามา เมื่อเปรียบเทียบกับสัญญาณความถี่ภายในระบบแล้วจะน้อยกว่าทำให้ได้แรงดันที่ต่ำกว่าเดิม เมื่อนำไปควบคุมวงจร OSC ก็จะทำให้ได้ความถี่ลดลงจากเดิมด้วย ถ้ากรณีที่ ความเร็วมอเตอร์ปกติ จะทำให้สัญญาณที่ผ่านเข้ามามีความถี่ถูกต้องตรงกับความถี่ของ OSC เมื่อเปรียบเทียบกันแล้วจะทำให้ได้แรงดันเท่าเดิม จึงไม่มีผลต่อการเปลี่ยนแปลงแรงดัน และความถี่ของ OSC จากรูปที่ 4.33 จะเห็นว่า วงจรนี้จะทำหน้าที่สร้างสัญญาณ RCLK เพื่อนำไปใช้ในการแยกสัญญาณข้อมูลออกจากสัญญาณนาฬิกาที่รวมกันมาในขณะทำการอ่านข้อมูลจากขา $\overline{\text{RAWREAD}}$ ซึ่งวงจรนี้จะประกอบด้วยวงจรต่าง ๆ เป็นส่วนประกอบ ดังนั้นการออกแบบสามารถทำตามขั้นตอนได้ดังนี้

1. ออกแบบวงจรแต่งรูปพัลส์ (PULSE SHAPING) เนื่องจากสัญญาณ $\overline{\text{RAWREAD}}$ ซึ่งดูได้จากรูปที่ 4.34A ที่อ่านมาจากตัวขับจานแม่เหล็กยังมีลักษณะเป็นพัลส์แคบ ๆ จึงขยายให้กว้าง เพื่อใช้ในการเปรียบเทียบในวงจร PHASE COMPARATOR โดยพิจารณาใช้ไอซีเบอร์ 74LS123 ซึ่งสามารถคำนวณได้จากรูปที่ 4.32 การออกแบบวงจรจะแสดงในรูปที่ 4.34B

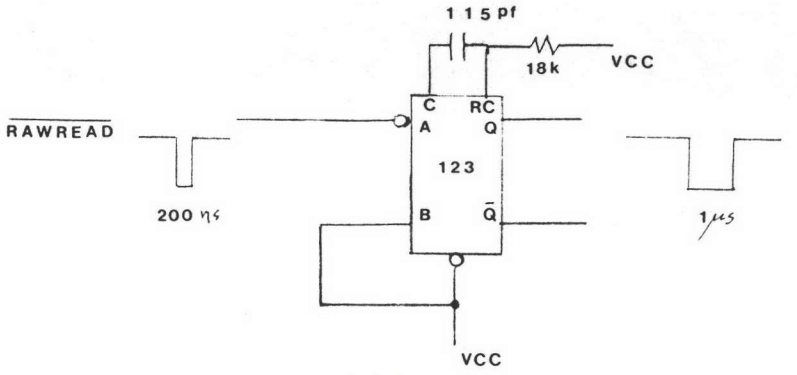


DISKETTE	MODE	$\overline{\text{DDEN}}$	CLK	NOMINAL T_a	NOMINAL T_b	NOMINAL T_c
8"	MFM	0	2 MHz	1 µs	1 µs	2 µs
8"	FM	1	2 MHz	2 µs	2µs	4 µs
5"	MFM	0	1 MHz	2µs	2µs	4 µs
5"	FM	1	1 MHz	4 µs	4 µs	8 µs

INPUT DATA TIMING:

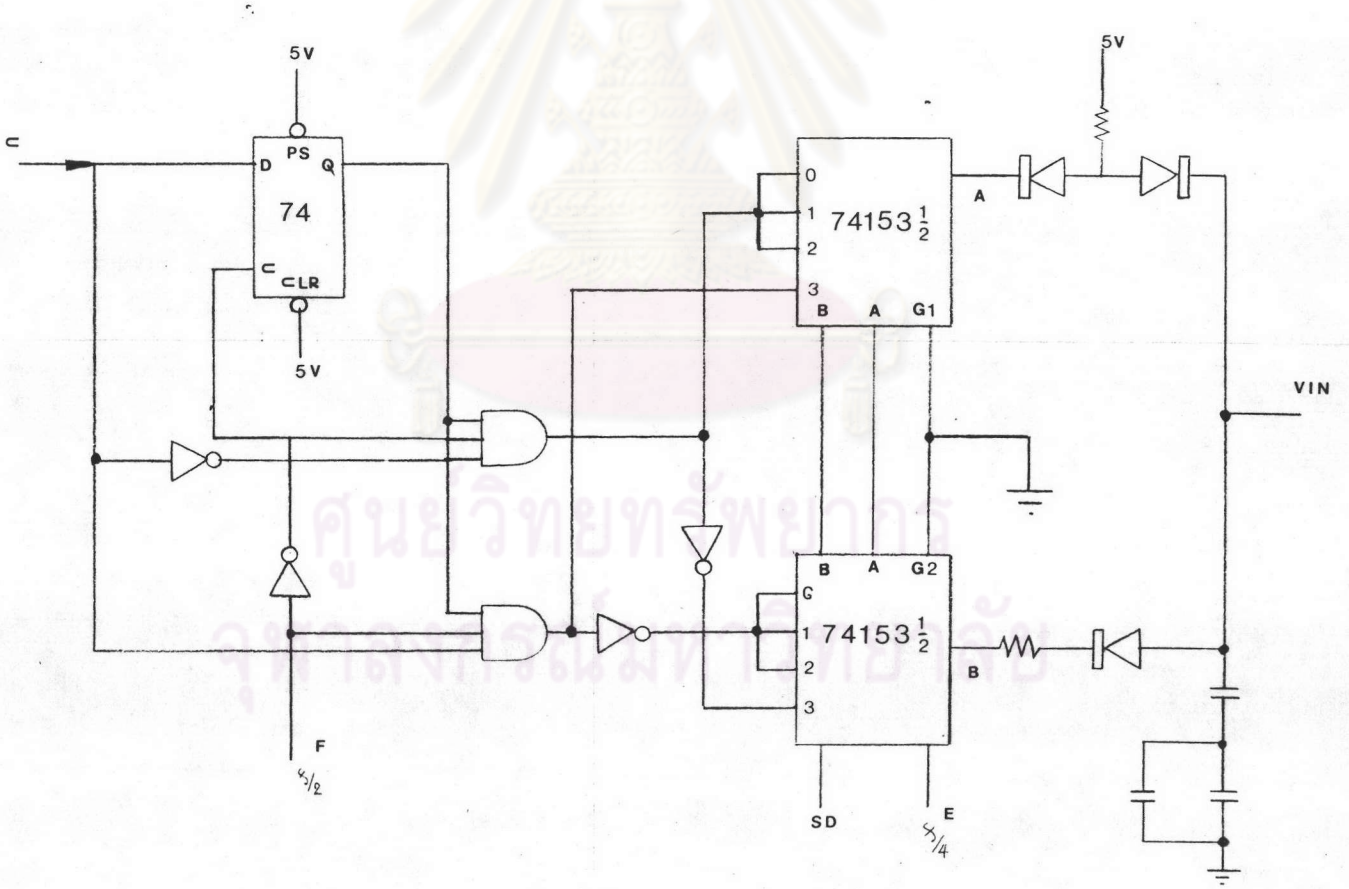
SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Tpw	$\overline{\text{Raw Read}}$ Pulse Width	100	200		nsec	See Note 1
tbc	$\overline{\text{Raw Read}}$ Cycle Time		1500		nsec	1800 ns @ 70°C
Tc	RCLK Cycle Time		1500		nsec	1800 ns @ 70°C
T_{x_1}	RCLK hold to $\overline{\text{Raw Read}}$	40			nsec	See Note 1
T_{x_2}	$\overline{\text{Raw Read}}$ hold to RCLK	40			nsec	

รูปที่ 4.34A แสดงสัญญาณ INPUT DATA TIMING



รูปที่ 4.34B วงจรแต่งรูปพัลส์สัญญาณ $\overline{\text{RAWREAD}}$

2. ออกแบบวงจรเปรียบเทียบเฟส และกรอง (PHASE COMPARATOR AND FILTER)

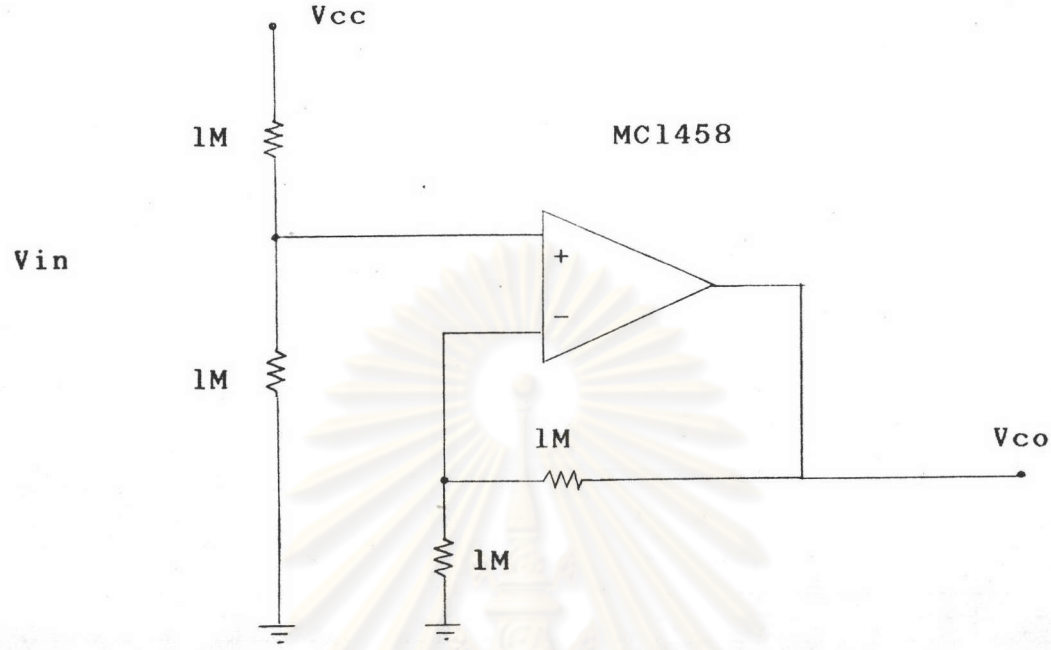


รูปที่ 4.35 วงจรเปรียบเทียบเฟสและกรอง

จากรูปที่ 4.35 จะพบว่าจะเปรียบเทียบสัญญาณ $\overline{\text{RAWREAD}}$ กับสัญญาณความถี่ของ Vco ซึ่งถ้าความถี่ของ Vco น้อยกว่า $\overline{\text{RAWREAD}}$ จะได้ OUTPUT ที่ A และ B เท่ากับ 1 ถ้าความถี่ของ Vco มากกว่า $\overline{\text{RAWREAD}}$ จะได้ OUTPUT ที่ A และ B เท่ากับ 0 ถ้าความถี่ของ Vco เท่ากับ $\overline{\text{RAWREAD}}$ จะได้ OUTPUT ที่ A เท่ากับ 0 และ B เท่ากับ 1 หลังจากนั้นจะนำสัญญาณ A และ B นี้ ไปควบคุมการประจุและคายประจุของวงจรกรอง ซึ่งจะสามารถอธิบายได้ดังรูปที่ 4.36

รูปที่ 4.36 แสดงค่า Vin ที่สภาวะต่าง ๆ และการทำงานของวงจรกรอง

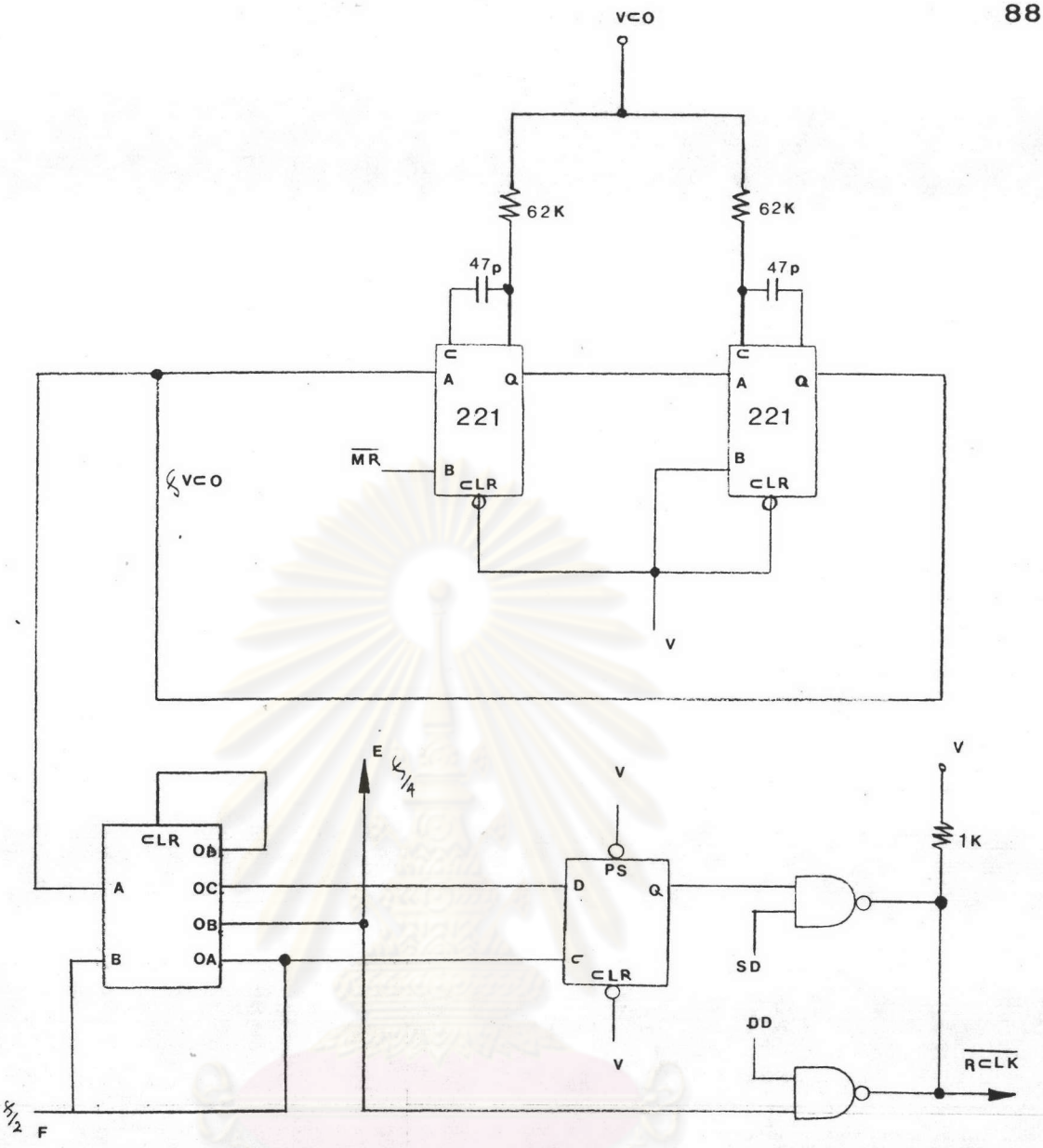
หลังจากได้ค่า Vin ตามต้องการแล้ว จะต้องผ่านวงจรขยายแรงดัน โดยใช้ OP-AMP ซึ่งแสดงในรูปที่ 4.37 และเพื่อเป็นตัวกั้นระหว่างวงจรกรองกับวงจร OSC ด้วย



รูปที่ 4.37 วงจรขยายแรงดัน Vco

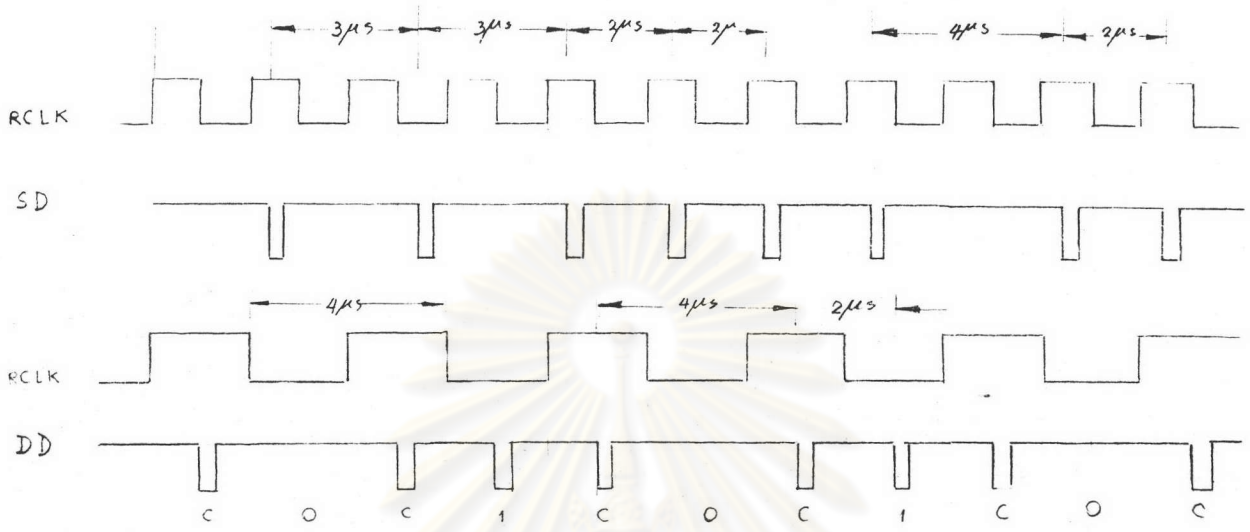
จากรูปที่ 4.37 จะได้สัญญาณ Vco เพื่อใช้ควบคุมการทำงานของวงจร OSC

3. การออกแบบวงจร OSC (OSCILLATOR) และวงจรหาร (DIVIDER) จะพิจารณาใช้ไอซีเบอร์ 74LS221 ทำหน้าที่เป็นวงจร OSC โดยต่อเป็นวงจรโมโนสเตเบิลสองตัวเป็นวงรอบ ซึ่งความถี่จะถูกควบคุมด้วย Vco ถ้า Vco สูงความถี่จะสูง ถ้า Vco ลดลงความถี่ ก็จะต่ำลงด้วย ซึ่งวงจรจะแสดงในรูปที่ 4.38



รูปที่ 4.38 วงจรสร้างความถี่และสร้างสัญญาณ RCLK

จากรูปที่ 4.38 จะเห็นว่าความถี่ที่ได้จาก OSC จะถูกหาร เพื่อนำไปใช้ ยังวงจรเปรียบเทียบ และนำไปใช้สร้างสัญญาณ RCLK ด้วย จากการเลือกของสัญญาณ FM หรือ MFM และ OSC นี้ จะมีความถี่คงที่ที่ 2 เมกกะเฮิรทซ์ หรือคาบเวลาเท่ากับ 500 nS จะเปลี่ยนแปลงขึ้นอยู่กับ Vco ที่ควบคุม รูปที่ 4.39 จะแสดงความสัมพันธ์ของสัญญาณ RCLK และ $\overline{\text{RAWREAD}}$



รูปที่ 4.39 แสดงความสัมพันธ์ของสัญญาณ RCLK และ $\overline{\text{RAWREAD}}$

4.9 การออกแบบโปรแกรมระบบ[6]

โปรแกรมระบบหมายถึงซอฟท์แวร์ที่ใช้ควบคุมการทำงานของฮาร์ดแวร์ภายในระบบให้ทำงานตามหน้าที่อย่างถูกต้อง สำหรับโปรแกรมระบบในที่นี้จะเก็บไว้ในหน่วยความจำแบบรอมขนาด 8 กิโลไบท์ ซึ่งจะประกอบด้วย โปรแกรมย่อย (SUBROUTINE) ต่าง ๆ ที่คอยทำหน้าที่ควบคุมการทำงานขององค์ประกอบฮาร์ดแวร์ที่ใช้สำหรับติดต่อรับส่งข้อมูล ระหว่างซีพียูกับอุปกรณ์ภายนอก ให้ทำงานตามความต้องการ และเป็นตัวกลาง ในการติดต่อระหว่างซอฟท์แวร์ที่ใช้งาน (APPLICATION SOFTWARE) กับอุปกรณ์ภายนอกของระบบทั้งหมด เช่น เทอร์-มินอล เครื่องพิมพ์ และตัวขับจานแม่เหล็ก รวมถึงอุปกรณ์ภายนอกที่จะต่อเข้าระบบ โปรแกรมระบบของระบบนี้จะแบ่งเป็น 2 ส่วน คือ ส่วนที่เป็นมอนิเตอร์ช่วยทำหน้าที่ ทดสอบการทำงานของหน่วยความจำ และการแสดงผลบนจอภาพและพิมพ์ผลทางเครื่องพิมพ์ ส่วนนี้จะมีขนาดประมาณ 1/4 กิโลไบท์ ส่วนที่เหลือ

จะเป็นเฟิร์มแวร์ สำหรับใช้ทำหน้าที่ ควบคุมการทำงานของตัวขับจานแม่เหล็ก เช่น การอ่านและบันทึกข้อมูล เป็นต้น ส่วนนี้จะมีขนาดประมาณ 1 กิโลไบท์ ดังนั้นจะเห็นว่า ยังมีส่วนหน่วยความจำรอม ที่เหลืออีกเกือบ 7 กิโลไบท์เก็บไว้สำหรับขยายหน้าที่ การทำงานของมอนิเตอร์ได้อีกในอนาคต รายละเอียดการทำงานของโปรแกรม ส่วนที่เป็นมอนิเตอร์ จะแสดงในผังงานที่ 4.1 ท้ายบท สำหรับโปรแกรมมอนิเตอร์จะแสดงในภาคผนวก ง และรายละเอียดการทำงานของโปรแกรมส่วนที่เป็นเฟิร์มแวร์ จะแสดงในผังงานที่ 4.2 - 4.22 ท้ายบท สำหรับโปรแกรมเฟิร์มแวร์ จะแสดงในภาคผนวก จ

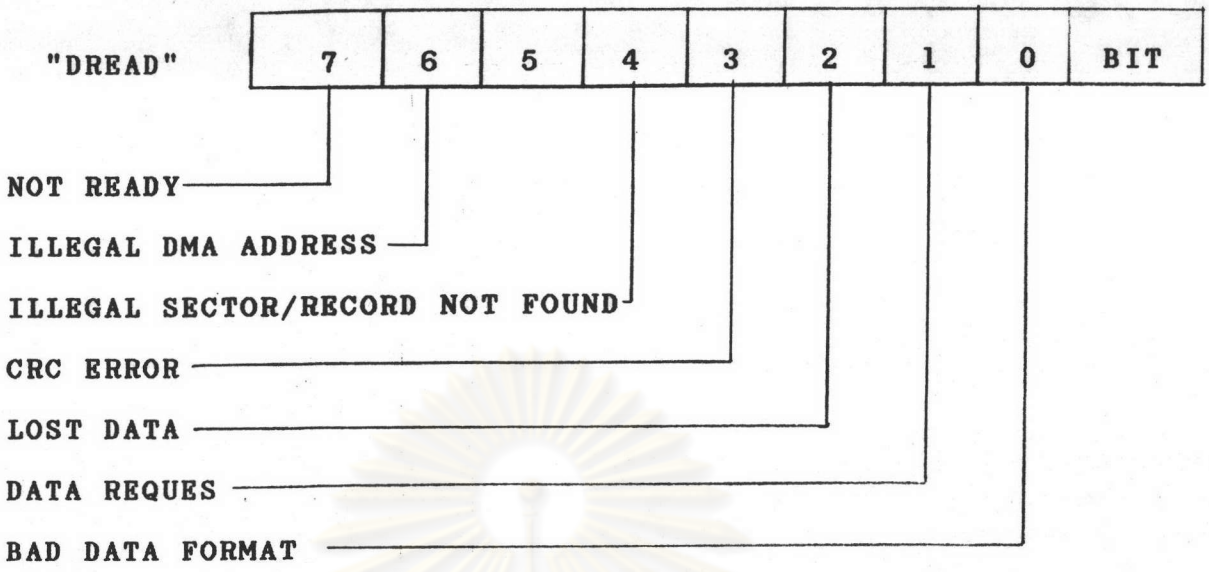
4.10 โปรแกรมย่อย (SUBROUTINE) ต่าง ๆ ที่สำคัญที่ใช้ในโปรแกรมระบบเมื่ออ้างถึงอุปกรณ์ไอโอ

เมื่อจะเขียนโปรแกรม เพื่อติดต่อกับอุปกรณ์ไอโอต่าง ๆ เช่น เทอร์มินอล เครื่องพิมพ์ และตัวขับจานแม่เหล็ก จำเป็นอย่างยิ่งที่จะต้องทราบถึงหน้าที่การใช้งานและแอดเดรสของโปรแกรมย่อย ๆ นั้น เพื่อสามารถเรียก (CALL) มาใช้งานร่วมกันกับโปรแกรมที่เขียนได้ โปรแกรมย่อยต่าง ๆ ที่ควรทราบจะแสดงในตารางที่ 4.2 สำหรับรายละเอียดการทำงานของโปรแกรมย่อยต่าง ๆ สามารถดูได้จากผังงานที่ 4.2 - 4.22 โปรแกรมย่อยเหล่านี้สามารถเรียกใช้ได้โดยคำสั่ง CALL และสามารถตรวจสอบการทำงานได้ ถ้ามีความผิดพลาดเกิดขึ้น เช่น ถ้าเกิดความผิดพลาดขึ้นในโปรแกรมย่อย จะทำให้ CARRY FLAG SET ถ้าการทำงานปกติ CARRY FLAG จะถูก CLEAR เป็น 0 การกำหนดค่า (PARAMETER) ต่าง ๆ ก่อนเรียกใช้โปรแกรมย่อย จะกำหนดในรีจีสเตอร์ C ยกเว้นการกำหนดแอดเดรส DMA จะกำหนดในรีจีสเตอร์ BC และการรับข้อมูลจากเทอร์มินอล ข้อมูลจะอยู่ในรีจีสเตอร์ A สำหรับการอ่านและบันทึกข้อมูลลงจานแม่เหล็ก รวมทั้งอ่านสภาวะของจานแม่เหล็ก หลังจากทำงานในโปรแกรมย่อยเหล่านี้แล้ว ข้อมูลแสดงสภาวะการทำงานจะอยู่ในรีจีสเตอร์ A ซึ่งสามารถดูรายละเอียดได้จากรูปที่ 4.40 และรูปที่ 4.41 กรณีเกิดความผิดพลาดของระบบขึ้นอาจจะใช้โปรแกรมย่อย DSKERR แสดงได้ โดยแสดงความผิดพลาดหรือความคลาดเคลื่อนของระบบด้วยการกระพริบความสว่างของ LED เพื่อให้รับรู้ได้

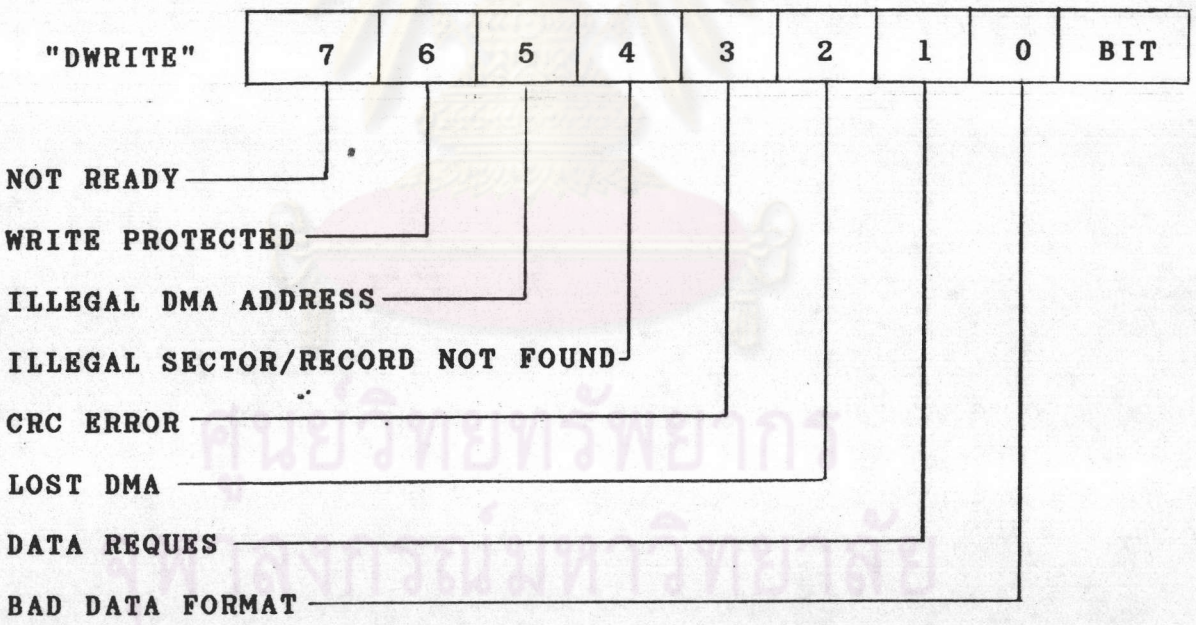
แอดเดรส	สัญญลักษณ์	หน้าที่
F800	DBOOT	เริ่มอ่าน BOOTSTRAP ROUTINE
F803	TERMIN	รับข้อมูลจากเทอร์มินอล
F806	TRMOUT	ส่งข้อมูลไปเทอร์มินอล
F809	TKZERO	กำหนดแทรคศูนย์
F80C	TRKSET	กำหนดค่าแทรค
F80F	SETSEC	กำหนดค่าเซคเตอร์
F812	SETDMA	กำหนดค่าแอดเดรส DMA
F815	DREAD	อ่านข้อมูลจากจานแม่เหล็ก
F818	DWRITE	บันทึกข้อมูลลงจานแม่เหล็ก
F81B	SELDRV	กำหนดไดร์ว
F81E	TPANIC	ทดสอบข้อมูลจากเทอร์มินอล
F821	TSTAT	อ่านสภาวะจากเทอร์มินอล
F824	DMAST	อ่านค่าแอดเดรส DMA
F827	STATUS	อ่านค่าสภาวะการทำงานของดิสค์ไดร์ว
F82A	DSKERR	แสดงค่า ERROR
F82D	SETDEN	กำหนดค่าเดนซิตี้
F830	SETSID	กำหนดด้านของแผ่น
F833	LSTOUT	พิมพ์ข้อมูล
F836	LSTSTA	อ่านสภาวะเครื่องพิมพ์

ตารางที่ 4.2 แสดงแอดเดรสและหน้าที่ของโปรแกรมย่อยต่าง ๆ

A REGISTER ERROR CODE

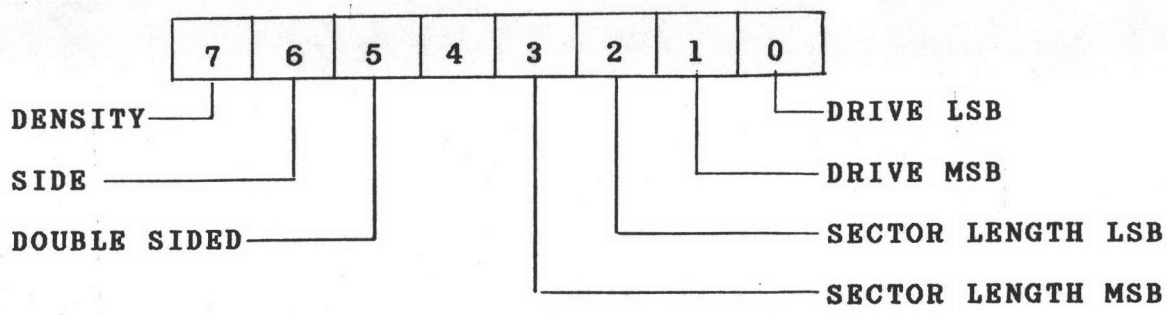


A REGISTER ERROR CODE



รูปที่ 4.40 แสดงข้อมูลและสภาวะในรีจีสเตอร์ A หลังจากเรียกใช้โปรแกรมย่อย DREAD และ DWRITE

A REGISTER BIT PATTERN



DRIVE MSB	DRIVE LSB	DRIVE No
0	0	DRIVE A
0	1	DRIVE B
1	0	DRIVE C
1	1	DRIVE D

SIDE BIT	SIDE SELECTED
0	SIDE 0
1	SIDE 1

SECTOR LENGTH MSB	SECTOR LENGTH LSB	SECTOR LENGTH	DENSITY
0	0	128	SINGLE
0	1	256	DOUBLE
1	0	512	DOUBLE
1	1	1024	DOUBLE

DENSITY BIT	
0	SINGLE
1	DOUBLE

รูปที่ 4.41 แสดงข้อมูลและสภาวะในรีจีสเตอร์ A หลังจากเรียกใช้โปรแกรมย่อย STATUS และการกำหนดค่าต่าง ๆ

4.10.1 ขั้นตอนและวิธีเรียกใช้โปรแกรมย่อยเพื่ออ่านหรือบันทึกข้อมูลจากจานแม่เหล็กมีดังต่อไปนี้

1. เลือกไดร์วก่อนว่าใช้ไดร์วไหน โดยการเรียกใช้โปรแกรมย่อยที่ชื่อ "SELDRV" ค่าไดร์วจะอยู่ในรีจีสเตอร์ C ถ้าเป็นแบบหัวดิสค์ 2 หัว ควรจะบอกด้าน (SIDE) ด้วย โดยการเรียกใช้โปรแกรมย่อย ที่ชื่อ "SETSID" โดยที่ค่ากำหนด SIDE จะอยู่ที่รีจีสเตอร์ C

2. ควรทำการINITIALIZE DISK โดยการกำหนดแทรคศูนย์ โดยเรียกใช้โปรแกรมย่อยที่ชื่อ "TRZERO" เพื่อให้แน่ใจว่าหัวอ่านอยู่ในตำแหน่งเริ่มต้นเพื่อการใช้งานต่อไป

3. กำหนดค่าแอดเดรส DMA โดยการเรียกใช้โปรแกรมย่อย "SETDMA" ค่าที่กำหนดอยู่ใน BC รีจีสเตอร์เพื่อบอกให้รู้ว่าจะอ่านหรือบันทึกข้อมูลจากแอดเดรสใดในหน่วยความจำ

4. กำหนดค่าแทรค ที่ต้องการอ่านหรือบันทึก โดยการเรียกโปรแกรมย่อย "TRKSET" ค่าที่กำหนดแทรคจะอยู่ในรีจีสเตอร์ C ถ้าเป็นการอ่านหรือบันทึกค่าแทรคเดิมไม่ต้องกำหนดใหม่

5. กำหนดค่าเซคเตอร์ที่ต้องการอ่านหรือบันทึก โดยการเรียกโปรแกรมย่อย "SETSEC" ค่าที่กำหนดเซคเตอร์จะอยู่ในรีจีสเตอร์ C แต่ถ้าเป็นการอ่านเซคเตอร์เดิมไม่ต้องกำหนดใหม่

6. เมื่อต้องการอ่านข้อมูล เรียกใช้โปรแกรมย่อย "DREAD" และถ้าต้องการบันทึกใช้ข้อมูลเรียกใช้โปรแกรม "DWRITE"

4.10.2 ตัวอย่างโปรแกรมบันทึกข้อมูลลงจานแม่เหล็ก

```
0000'           0001        ORG     2000H
                0002 ;--------------------------------
     (F809)     0003 TKZERO  EQU     0F809H      ; recalibrate head routine
     (F80C)     0004 TRKSET  EQU     0F80CH      ; set track routine
     (F80F)     0005 SETSEC  EQU     0F80FH      ; set sector routine
     (F812)     0006 SETDMA  EQU     0F812H      ; set DMA. address routine
     (F815)     0007 DREAD   EQU     0F815H      ; disk read routine
     (F818)     0008 DWRITE  EQU     0F818H      ; disk write routine
     (F81B)     0009 SELDRV  EQU     0F81BH      ; select drive routine
     (F824)     0010 DMAST   EQU     0F824H      ; read recent DMA. address routine
                0011 ;--------------------------------
                0012 ;  Program for write data from    ;
                0013 ;  memory address 3000H   to     ;
                0014 ;  diskette track 45             ;
                0015 ;           sector 8 to 17       ;
                0016 ;      256 bytes/sector          ;
                0017 ;--------------------------------
2000  310050    0018 WRITE:  LD      SP,5000H    ; set stack pointer
2003  0E00      0019         LD      C,0         ; select drive 0
2005  CD1BF8    0020         CALL    SELDRV
2008  CD09F8    0021         CALL    TKZERO      ; recalibrate head
200B  0E2D      0022         LD      C,45
200D  CD0CF8    0023         CALL    TRKSET      ; seek to track 45
2010  0E08      0024         LD      C,8         ; begin write at sector 8
2012  060A      0025         LD      B,10        ; and write 10 sectors
2014  C5        0026         PUSH    BC
2015  010030    0027         LD      BC,3000H    ; set write data address
2018  CD12F8    0028 WRNXT:  CALL    SETDMA
201B  C1        0029         POP     BC
201C  C5        0030         PUSH    BC
201D  CD0FF8    0031         CALL    SETSEC      ; set sector
2020  CD18F8    0032         CALL    DWRITE      ; and write that sector
2023  C1        0033         POP     BC
2024  3810      0034         JR      C,ERROR     ; disk write error ?
2026  05        0035         DEC     B           ; decrement counter
2027  280E      0036         JR      Z,DONE      ; counter = 0 -> Ok.
2029  0C        0037         INC     C           ; bump to next sector
202A  C5        0038         PUSH    BC
202B  CD24F8    0039         CALL    DMAST       ; read DMA. address
202E  210001    0040         LD      HL,100H     ; add sector size to current
2031  09        0041         ADD     HL,BC       ;  DMA. address
2032  44        0042         LD      B,H
2033  4D        0043         LD      C,L
2034  18E2      0044         JR      WRNXT
                0045 ;--------------------------------
2036  76        0046 ERROR:  HALT                ; force processor halt when error
                0047 ;--------------------------------
2037  C30000    0048 DONE:   JP      0           ; goto monitor when ready
                0049 ;--------------------------------
203A  (0000)    0050         END

Errors          0
```

4.10.3 ตัวอย่างโปรแกรมอ่านข้อมูลจากจานแม่เหล็ก

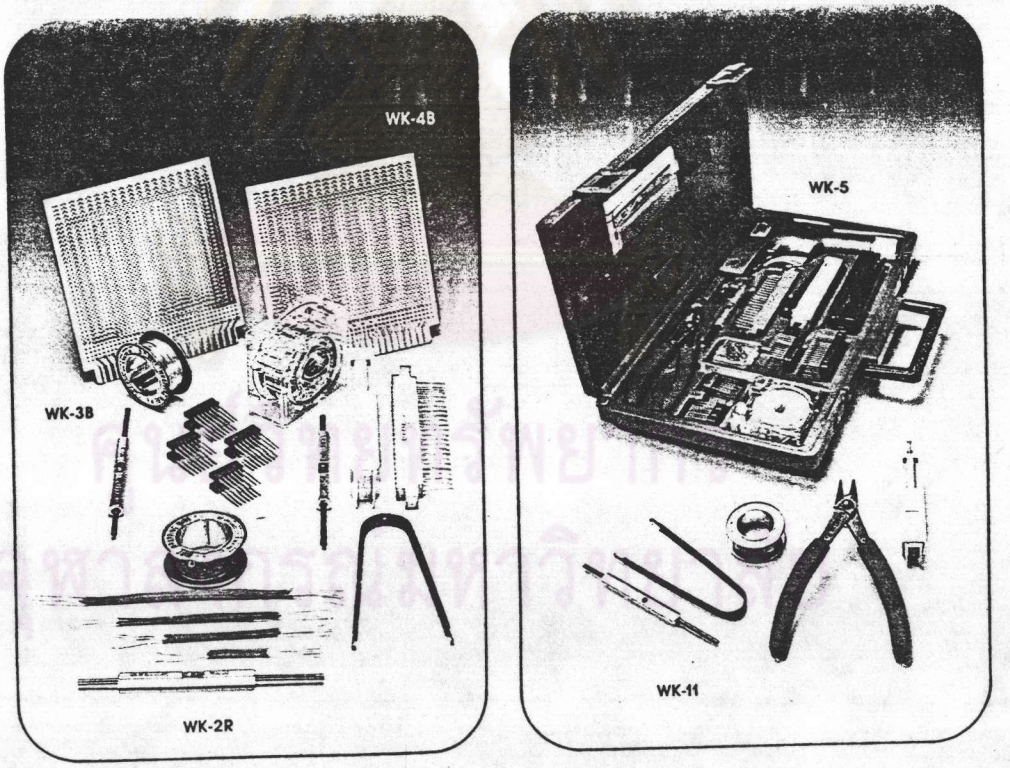
```
0000'                0001          ORG     2000H
                     0002 ;---------------------------------
      (F809)         0003 TKZERO   EQU     0F809H          ; recalibrate head routine
      (F80C)         0004 TRKSET   EQU     0F80CH          ; set track routine
      (F80F)         0005 SETSEC   EQU     0F80FH          ; set sector routine
      (F812)         0006 SETDMA   EQU     0F812H          ; set DMA. address routine
      (F815)         0007 DREAD    EQU     0F815H          ; disk read routine
      (F818)         0008 DWRITE   EQU     0F818H          ; disk write routine
      (F81B)         0009 SELDRV   EQU     0F81BH          ; select drive routine
      (F824)         0010 DMAST    EQU     0F824H          ; read recent DMA. address routine
                     0011 ;---------------------------------
                     0012 ;  Program for read data from    ;
                     0013 ;  diskette track 45             ;
                     0014 ;           sector 8 to 17       ;
                     0015 ;      256 bytes/sector          ;
                     0016 ;  to memory address 3000H       ;
                     0017 ;---------------------------------
2000  310050         0018 READ:    LD      SP,5000H        ; set stack pointer
2003  0E00           0019          LD      C,0             ; select drive 0
2005  CD1BF8         0020          CALL    SELDRV
2008  CD09F8         0021          CALL    TKZERO          ; recalibrate head
200B  0E2D           0022          LD      C,45
200D  CD0CF8         0023          CALL    TRKSET          ; seek to track 45
2010  0E08           0024          LD      C,8             ; begin read at sector 8
2012  060A           0025          LD      B,10            ; and read 10 sectors
2014  C5             0026          PUSH    BC
2015  010030         0027          LD      BC,3000H        ; set read address
2018  CD12F8         0028 RDNXT:   CALL    SETDMA
201B  C1             0029          POP     BC
201C  C5             0030          PUSH    BC
201D  CD0FF8         0031          CALL    SETSEC          ; set sector
2020  CD15F8         0032          CALL    DREAD           ; and read that sector
2023  C1             0033          POP     BC
2024  3810           0034          JR      C,ERROR         ; disk read error ?
2026  05             0035          DEC     B               ; decrement counter
2027  280E           0036          JR      Z,DONE          ; counter = 0 -> Ok.
2029  0C             0037          INC     C               ; bump to next sector
202A  C5             0038          PUSH    BC
202B  CD24F8         0039          CALL    DMAST           ; read DMA. address
202E  210001         0040          LD      HL,100H         ; add sector size to current
2031  09             0041          ADD     HL,BC           ;  DMA. address
2032  44             0042          LD      B,H
2033  4D             0043          LD      C,L
2034  18E2           0044          JR      RDNXT
                     0045 ;---------------------------------
2036  76             0046 ERROR:   HALT                    ; force processor halt when error
                     0047 ;---------------------------------
2037  C30000         0048 DONE:    JP      0               ; goto monitor when ready
                     0049 ;---------------------------------
203A  (0000)         0050          END

Errors               0
```

4.11 การสร้างโปรเซสเซอร์

ในการสร้างโปรเซสเซอร์นี้ จะไม่ใช้วิธีการต่อวงจร แบบทำปรินท์ เพราะต้นทุนสูงและเสียเวลาในการออกแบบวงจรปรินท์ ขณะเดียวกันยังแก้ไขหรือดัดแปลงวงจรได้ยาก ไม่เหมาะกับงานออกแบบสร้างเครื่องต้นแบบ ที่ใช้ในการทำวิจัย แต่จะพิจารณาเลือกใช้วิธีการต่อวงจร ด้วยการพันสายหรือต่อสายวงจรแบบ WIREWRAP[16] โดยจะทำการสร้างหรือออกแบบวงจรทีละส่วน เริ่มจากส่วนซีพียู หน่วยความจำ อินเตอร์เฟสเทอร์มินอล และเครื่องพิมพ์ จนกระทั่งถึงส่วนอินเตอร์เฟสตัวขับจานแม่เหล็ก ตามลำดับ ขณะเดียวกันหลังจากต่อวงจรเสร็จแต่ละส่วนจะทำการทดลองเป็นขั้น ๆ ซึ่งจะกล่าวรายละเอียดต่อไป

4.11.1 อุปกรณ์ที่ใช้ในการสร้าง

หลังจากพิจารณาถึงวิธีการในการต่อสายวงจรแล้ว ควรจะทราบถึงอุปกรณ์ที่ใช้ในการสร้างหรือต่อวงจร ซึ่งจะแสดงดังรูปที่ 4.42



รูปที่ 4.42 แสดงถึงอุปกรณ์ที่ใช้ในการสร้างต่อวงจร

จากรูปที่ 4.42 จะพบว่า อุปกรณ์ที่ใช้ ในการสร้างวงจรแบบ WIREWRAP จะประกอบด้วย

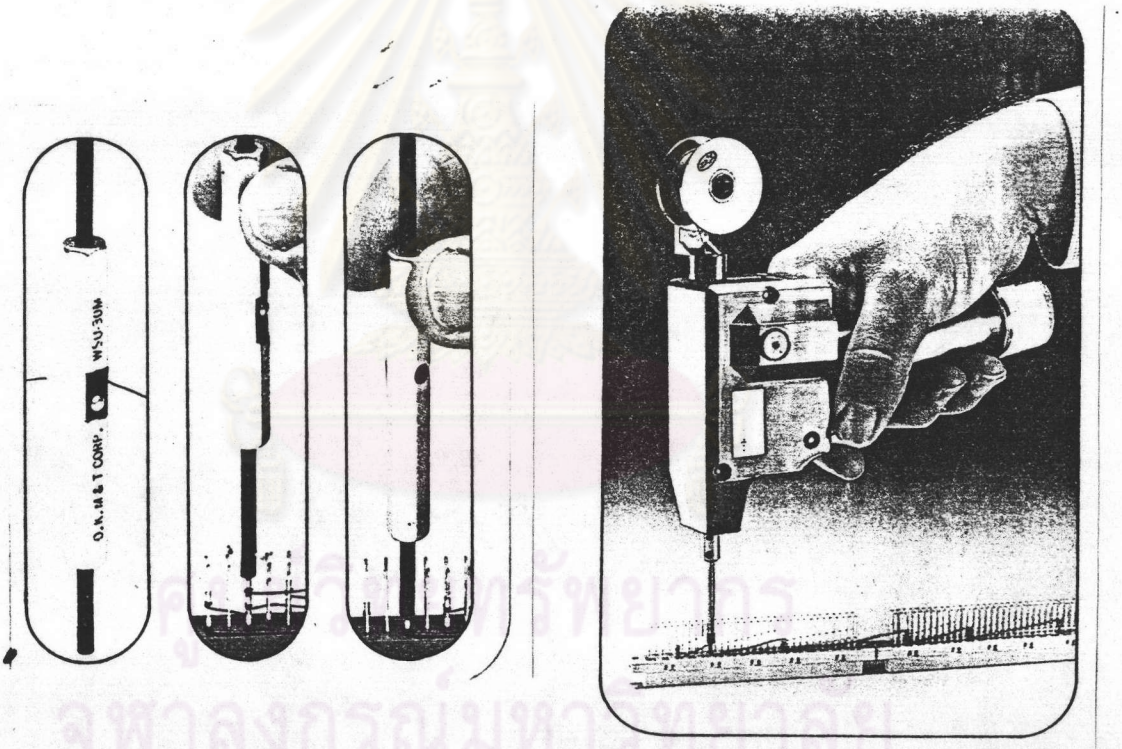
1. เครื่องมือใช้ในการพันสาย (WRAP TOOL) ขณะเดียวกันจะทำหน้าที่ปลอกสายด้วย

2. สายไฟใช้สำหรับในการพัน (WIREWRAP)

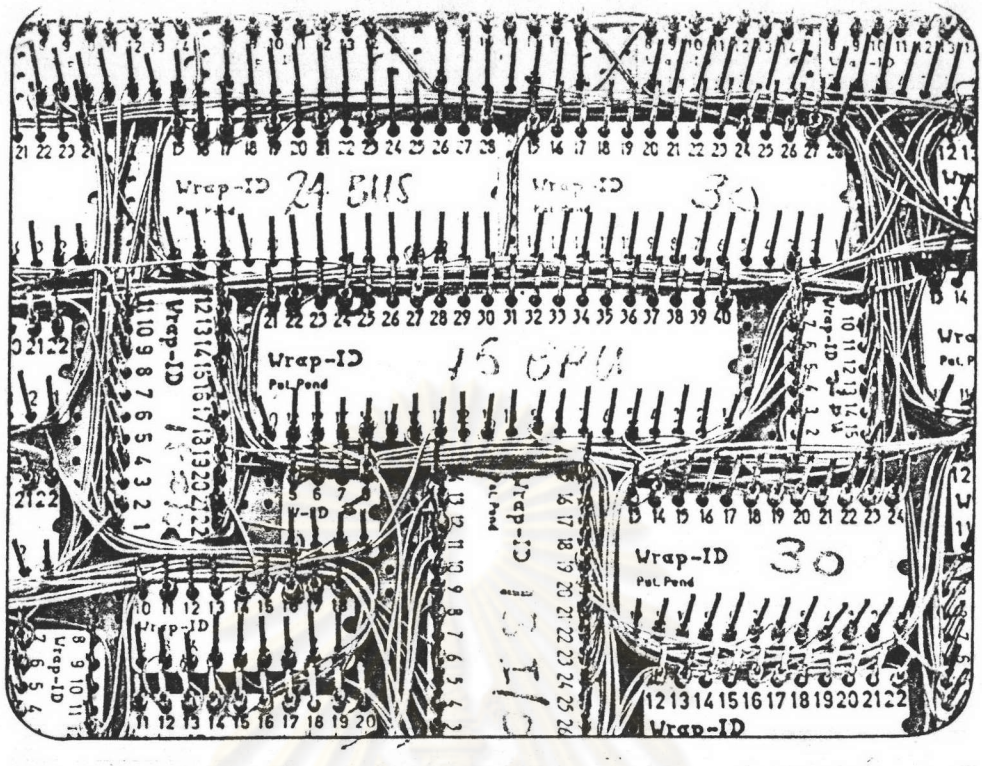
3. SOCKET สำหรับเสียบไอซีจะเป็นแบบWIREWRAP SOCKET ขาจะยาวกว่าธรรมดา

4. แผ่นพีซีบอร์ด (PC BOARD) สำหรับยึดขา SOCKET และจัดวางองค์ประกอบไอซี

5. คีมตัดสาย



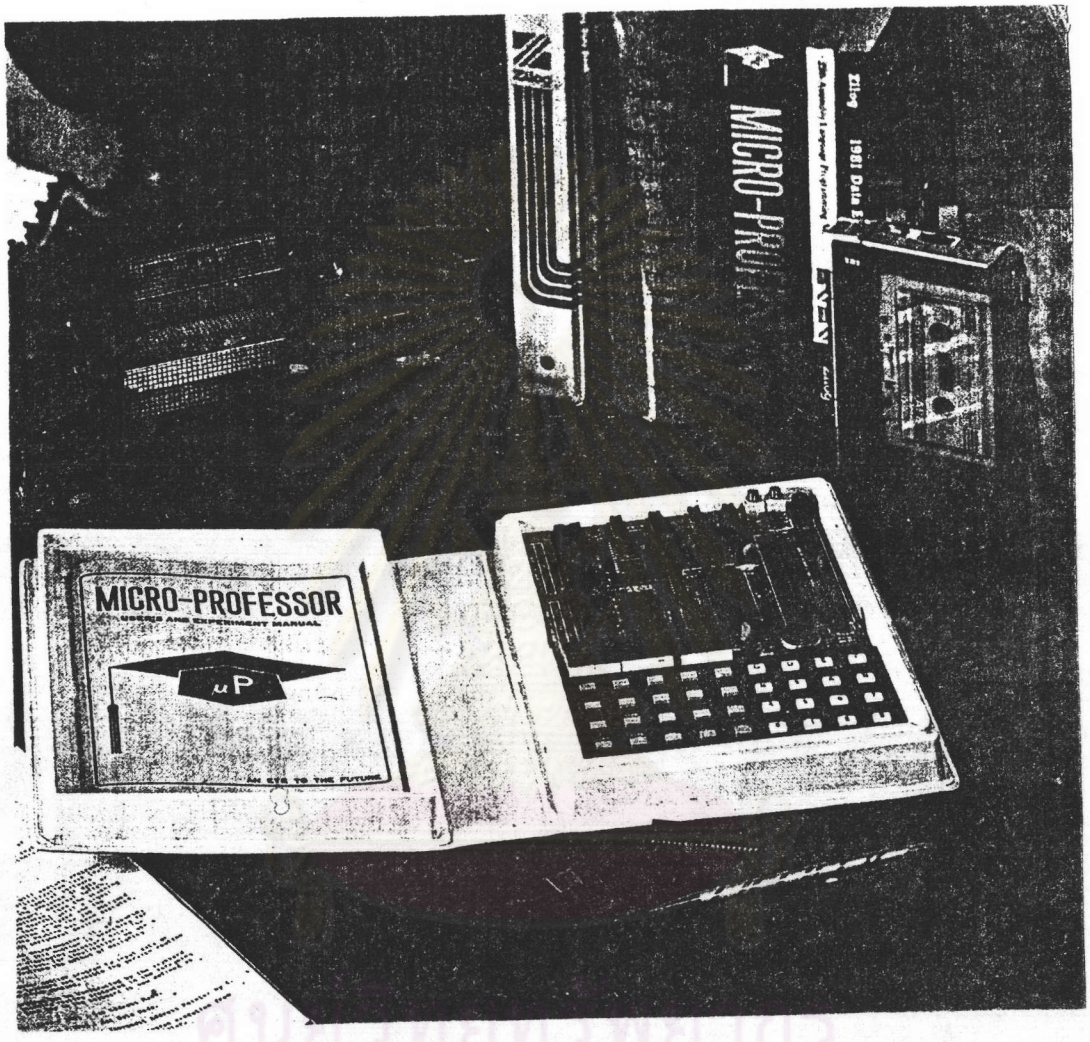
รูปที่ 4.43 แสดงลักษณะวิธีการต่อสายโดยวิธีการ WIREWRAP



รูปที่ 4.44 แสดงวงจรที่ถูกสร้างโดยวิธีการ WIREWRAP

4.11.2 การสร้างและการทดสอบ

เพื่อให้ง่ายต่อการสร้างวงจรและทดสอบ จะแบ่งแผ่นปริ๊นท์ออกเป็น 2 แผ่น สำหรับการจัดวางช็อกเกตของวงจรซีพียู หน่วยความจำ อินเตอร์-เฟสเทอร์มินอล และเครื่องพิมพ์ไว้ 1 แผ่น ส่วนอีก 1 แผ่น จะใช้เป็นวงจรอินเตอร์เฟสตัวขับจานแม่เหล็กโดยเฉพาะ และการทดสอบจะใช้เครื่องไมโคร-โปรเซสเซอร์ แบบซิงเกิลบอร์ด ยี่ห้อ MPF-I มาช่วย เพราะใช้ซีพียู Z-80 เหมือนกัน ซึ่งจะดูได้จากรูปที่ 4.45 จะพบว่ามีแป้นกดข้อมูล (KEYBOARD) และหน่วยความจำขนาด 2 กิโลไบท์ มีมอนิเตอร์ขนาด 2 กิโลไบท์ และสามารถแสดงผลได้ 6 หลัก เป็นแอดเดรส 4 หลัก ข้อมูล 2 หลัก และยังมีระบบบัสที่สามารถต่อขยายใช้งานได้ เป็นตัวต่อ (CONNECTOR) แบบ 40 ขา ทำให้สามารถนำมาใช้กับวงจรที่ออกแบบแต่ละส่วนได้ เพื่อการทดสอบและสร้างวงจรได้รวดเร็วขึ้น แทนที่จะรอให้สร้างเสร็จทั้งหมดทุกส่วนของวงจรก่อน เพราะถ้าหากเกิดปัญหาในส่วนใดแล้ว จะแก้ไขยากและเสียเวลาด้วย สำหรับขั้นตอนการสร้างจะสามารถจัดเป็นข้อ ๆ ได้ดังนี้



รูปที่ 4.45 แสดงลักษณะของเครื่องไมโครโปรเซสเซอร์ MPF-I

4.11.3 ขั้นตอนการสร้างวงจรต่าง ๆ

1. จัดวางอุปกรณ์ช็อกเกตไอซี ให้เหมาะสม ตามตำแหน่งที่สัมพันธ์กันของวงจรแต่ละส่วน

2. ต่อสายวงจรที่เกี่ยวกับ Vcc และ GROUND ก่อน และค่อยเดินสายสัญญาณทีหลัง ควรต่อให้สายสัญญาณความถี่สูง เช่น แอดเดรสบัส บัสข้อมูล สัญญาณควบคุม และสัญญาณนาฬิกาสั้นที่สุด และอยู่ห่างหรือตั้งฉากกับสายไฟดีซี ควรเลือกใช้สายหลายสีเพื่อจะตรวจสอบวงจรได้ง่าย โดยใช้สายสีเดียวกันอยู่ในสัญญาณประเภทเดียวกัน เช่น Vcc สีแดง GROUND สีดำ และสัญญาณแอดเดรสสีฟ้า สัญญาณข้อมูลสีเหลือง เป็นต้น เมื่อทำการต่อสายเส้นใดในวงจรแล้ว ควรจะทำเครื่องหมายไว้ในวงจร ที่อยู่ในกระดาษด้วย (เพื่อป้องกันการเผลอหรือต่อสายซ้ำ)

3. หลังจากต่อสายวงจรใดเสร็จควรทำการตรวจสอบ โดยวิธีการใช้โอห์มมิเตอร์หรือบัสเซอร์ (BUZZER) ให้แน่ใจว่าต่อสายไม่ผิด และควรทำเครื่องหมายไว้ในวงจรว่าผ่านการตรวจสอบแล้ว

4. ควรต่อไฟดีซี และลองวัดไฟดู ในขณะที่ยังไม่มีไอซีเสียบอยู่เพื่อป้องกันความผิดพลาด โดยวัดที่ขา Vcc และ GROUND ของแต่ละช็อกเกตว่าถูกต้องหรือไม่

4.11.4 ขั้นตอนการสร้างและทดสอบส่วนซีพียู

1. ทำตามขั้นตอนที่ 1-4 ของหัวข้อ 4.11.3 แล้วใส่ไอซีที่ใช้ในส่วนซีพียูลงในช็อกเกต วัดไฟดีซี และสัญญาณนาฬิกาดู ถ้าดีทำข้อ 2 ต่อไป ถ้าไม่ดีให้แก้ไขใหม่

2. ทดสอบการทำงานโดยต่อเข้ากับเครื่อง MPF-I โดยดึงซีพียูของ MPF-I ออก ทำให้เราสามารถตรวจสอบได้ง่าย เนื่องจากอาศัยภาคแสดงผลและรับข้อมูลรวมทั้งมอนิเตอร์ของ MPF-I ได้ โดยไม่ต้องเสียเวลาสร้างวงจรเพื่อทำการทดสอบ และเขียนโปรแกรมมอนิเตอร์เอง

ปัญหาของการสร้างซีพียูจะอยู่ที่วงจรสร้างสัญญาณนาฬิกามากที่สุด ควรให้สัญญาณแรงพอที่จะใช้จ่ายให้กับวงจรต่าง ๆ ได้ และรูปที่ได้ออกมาควรเป็นสี่เหลี่ยม (SQUARE WAVE) วิธีแก้ปัญหาอาจใช้ DRIVER ช่วย หรือต่อสายสัญญาณนาฬิกาให้สั้นที่สุด

4.11.5 ขั้นตอนการสร้าง และทดสอบวงจรหน่วยความจำรอม และไดนามิคแรม

1. ทำตามขั้นตอนที่ 1-4 ในหัวข้อ 4.11.3 แล้วทดลองนำอีพรอมของ MPF-I มาใส่และทดสอบ เพราะสามารถใช้แทนกันได้เบอร์ 2732 กับ 2764 แทนกันได้

2. ทดสอบหน่วยความจำไดนามิคแรม โดยวิธีการเขียนโปรแกรมทดสอบขึ้นมาโดยอาศัยโปรแกรมมอนิเตอร์ของ MPF-I เป็นหลัก โปรแกรมที่เขียน (โปรแกรมอีพรอม) นี้จะอยู่ในแอดเดรส 0000H ของอีพรอมที่จะทดสอบ โดยมีโปรแกรมมอนิเตอร์ของ MPF-I อยู่ที่ส่วนแอเดรส 1100H ของอีพรอม โปรแกรมที่เขียนนี้จะทำหน้าที่ย้าย (MOVE) ข้อมูลและโปรแกรมมอนิเตอร์ของ MPF-I จากอีพรอมไปเก็บยังแรมที่แอดเดรส 3000H (เพราะตอนแรกทำงานในอีพรอมซึ่งมีขนาด 8K ไม่สามารถอ้างถึง 8K แรกของแรมได้) หลังจากนั้นจะกระโดด (JUMP) ไปทำงานในแรมตรงแอดเดรส 3000H ก่อนจะถึงส่วนโปรแกรมมอนิเตอร์ MPF-I จะเป็นคำสั่งให้ DISABLE อีพรอมทิ้งและทำหน้าที่ย้ายข้อมูลที่เป็นมอนิเตอร์ของ MPF-I ไปยังแอดเดรส 0000H เป็นอันเสร็จขั้นตอนนี้ ขั้นตอนการทำงานจะสามารถแสดงได้ในรูปที่ 4.46 และทดลองใช้งานกับแป้นกดข้อมูลของ MPF-I ดู โดยเขียนโปรแกรมทดสอบการทำงานของหน่วยความจำแรมทั้ง 64 กิโลไบท์ได้ เพราะตอนนี้มอนิเตอร์ MPF-I จะอยู่ในแรมทั้งหมดแล้ว อย่าลืมว่าก่อนจะทดสอบหน่วยความจำนี้ซีพียูจะใช้วงจรที่ออกแบบแล้วโดยมีความเร็วของสัญญาณนาฬิกา 4 เมกกะเฮิรทซ์

ปัญหา ของการสร้างวงจรนี้ คือ ข้อมูลมักจะหายไปหรือหยุดชะงักเมื่อทดสอบหน่วยความจำคือ เกิดการผิดพลาดของข้อมูลที่บันทึกหรืออ่านไม่ตรงกัน

วิธีแก้ปัญหา ให้ใช้ LOGIC ANALYZER ตรวจดูมักจะพบว่าสัญญาณการรีเฟรชไม่เป็นไปตามเวลาที่ต้องการ คือ สัญญาณ RAS เกิดเกือบพร้อมกับสัญญาณ CAS ตามข้อกำหนด RAS จะเกิดก่อน CAS อย่างน้อย 35-40 nS วิธีแก้อาจจะใช้คอนเดนเซอร์ค่าน้อยๆ หน่วง CAS ไว้

	EPROM 8K	RAM 64K	
0000H	PROGRAM MOVE DATA FROM ADDRESS 1000H TO 3000H	MPF-I MONITOR	0000H
1000H	DISABLE ROM AND MOVE DATA FROM ADDRESS 3100H TO 0000H AND JUMP TO 0000H		
1100H	MPF-I MONITOR	DISABLE ROM AND MOVE DATA FROM ADDRESS 3100H TO 0000H AND JUMP TO ADDRESS 0000H	3000H
		MPF-I MONITOR	3100H

รูปที่ 4.46 แสดงขั้นตอนของการย้ายโปรแกรม MPF-I MONITOR จากอีพรอมไปยังแรม

4.11.6 ขั้นตอนการสร้างวงจรอินเตอร์เฟสซีอาร์ที

1. ทำตามขั้นตอนที่ 1-4 ของหัวข้อที่ 4.11.3 ใส่ไอซีที่ใช้ในส่วนนี้ลงไปในซ็อกเกตวัดไฟดีซีและสัญญาณต่าง ๆ และต่อซีอาร์ทีเทอร์มินอล

2. เขียนโปรแกรมโดยใช้มอนิเตอร์ของ MPF-I โดยเตรียมโปรแกรมจากแป้นกดข้อมูลของ MPF-I ให้โปรแกรมนี้ทำการ INITIALIZE CTC SIO ตามข้อกำหนดที่ต้องการ และทดลองส่ง-รับข้อมูลดูทีละตัว แต่ต้องให้ BAUD RATE ถูกต้องด้วย

ปัญหา การสร้างวงจรนี้มักจะมีปัญหาจากการส่งข้อมูลหรือรับข้อมูลผิด อันเนื่องมาจาก BAUD RATE ไม่ถูกต้อง เพราะใช้ CTC เป็นตัวกำหนดข้อสำคัญอีกอย่างคือ ความถี่เข้าขา CLKTG ของ CTC ควรจะต่ำกว่าความถี่ของสัญญาณนาฬิการะบบที่เข้ายังขา CLK ของ CTC และบางครั้งกดข้อมูลค่าเดียวแต่ปรากฏว่าบนจอมี 2 ตัวได้

วิธีแก้ปัญหา กรณีที่ส่งข้อมูลหรือรับข้อมูลผิดพลาด อาจมาจากการคำนวณ BAUD RATE แล้วเมื่อนำมาใช้งาน ค่าที่คำนวณอาจใช้ไม่ได้เนื่องจากความคลาดเคลื่อนของระบบ ดังนั้นควรลองเปลี่ยนค่าดูเรื่อย ๆ ทั้งทางบวกและลบจะได้ค่าที่เหมาะสมกับการใช้งานจริง กรณีที่กดข้อมูลตัวเดียวแต่บางครั้งปรากฏบนจอ 2 ตัว อาจแก้ไขได้โดยเปลี่ยนบัสไดร์เวอร์ของระบบซีพียูเสียใหม่

4.11.7 ขั้นตอนการสร้างวงจรอินเตอร์เฟสเครื่องพิมพ์

1. ทำตามขั้นตอนที่ 1-4 ของหัวข้อที่ 4.11.3 ใส่ไอซีในส่วนของเครื่องพิมพ์ในช็อกเกตวัดไฟดีซีและสัญญาณต่าง ๆ และต่อเครื่องพิมพ์

2. เขียนโปรแกรมโดยใช้มอนิเตอร์ของ MPF-I ช่วยโดยเตรียมโปรแกรม จากแป้นกดข้อมูลของ MPF-I ให้โปรแกรมนี้ ทำการ INITIALIZE PIO ตามข้อกำหนดที่ต้องการและทดลองส่งข้อมูลไปพิมพ์ดู

ปัญหา การสร้างวงจรนี้มักจะมีปัญหาคือข้อมูลที่ส่งไปพิมพ์ผิด ๆ ถูก ๆ บางครั้งปิดเปิดเครื่องพิมพ์ ก็ทำให้ระบบซีพียูหยุดชะงักไม่ทำงาน

วิธีแก้ปัญหา ใส่บัสไดร์เวอร์ให้กับวงจรเอาท์พุทของบัสข้อมูล และสัญญาณที่อ่านตรวจสอบสภาวะบางจุดควรทำการ PULL UP กับ Vcc เสียเพราะเมื่อต่อสายอินเตอร์เฟสกับเครื่องพิมพ์ยาว มักจะเกิดการสูญหายหรือผิดพลาดของข้อมูลได้เนื่องจากข้อมูลที่ส่งเป็นสัญญาณ TTL ซึ่งมีขนาด 5 V

4.11.8 ขั้นตอนการสร้างและทดสอบวงจรอินเตอร์เฟสตัวขับจานแม่เหล็ก

1. ทำตามขั้นตอนที่ 1-4 ของหัวข้อที่ 4.11.3 แล้วใส่ไอซีในส่วนนี้ให้ครบแล้ววัดไฟดีซี และสัญญาณต่าง ๆ ที่ใช้ภายในระบบ เช่น สัญญาณนาฬิกา สัญญาณ SYNC และสัญญาณ CHIP SELECT ส่วนต่าง ๆ โดยสามารถเขียนโปรแกรมจาก MPF-I และทดลองใช้ LOGIC PROB จับดูได้

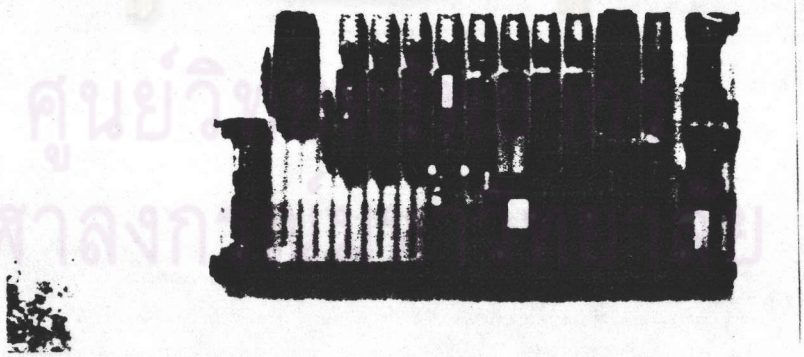
2. ทำการโปรแกรม อีพรอมใหม่ โดยคราวนี้ ใส่โปรแกรมเฟิร์มแวร์ สำหรับควบคุมตัวขับจานแม่เหล็กเข้าไปด้วย โดยรวมกับโปรแกรมมอนิเตอร์ของ MPF-I แต่ตอนย้ายข้อมูลลงหน่วยความจำแรมนั้นให้โปรแกรมมอนิเตอร์ของ MPF-I อยู่ที่แอดเดรส 0000H แต่ให้โปรแกรมเฟิร์มแวร์ (FIRMWARE) ที่ใช้ควบคุม ตัวขับจานแม่เหล็กไปไว้ยัง แอดเดรส F800H (ขั้นตอนการทำคล้ายกับข้อที่ 2 ในหัวข้อ 4.11.5)

3. ทดลองเขียน โปรแกรมจาก MPF-I ดู โดยเรียกใช้โปรแกรมย่อยในเพิร์มแวร์ ซึ่งสามารถดูได้จาก ตารางที่ 4.2 ทดลองทำการบันทึกข้อมูลและอ่านข้อมูลดูว่าตรงกันหรือไม่ ถ้าตรงกันแสดงว่าถูกต้อง โดยดูตัวอย่างการอ่านและบันทึกได้จาก หัวข้อ 4.10.2 - 4.10.3

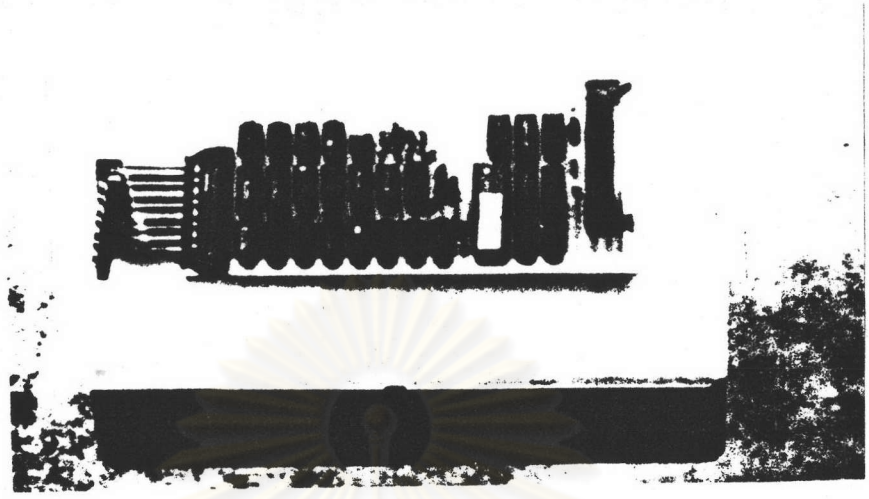
ปัญหา มักจะเกิดขึ้นเสมอกับการสร้างวงจรส่วนนี้ก็คือ การอ่านข้อมูลมาผิดหรือบันทึกข้อมูลผิดพลาด แต่ที่พบมากก็คือ การอ่านข้อมูลมาผิด

วิธีแก้ปัญหา ถ้ากรณีอ่านข้อมูลมาผิดหรืออ่านไม่ได้ ให้สังเกตดู วงจร PULSE SHAPING ซึ่งค่า R ที่ใช้มักจะต้องมีการเปลี่ยนแปลงอาจจะใช้ R ปรับค่าได้แทน และทดลองปรับดูค่าสัญญาณ RAWREAD ปกติจะมีค่าประมาณ 200 nS ในช่วงลบ ดังนั้นเมื่อผ่านวงจร PULSE SHAPING แล้วควรจะได้ 1 uS ซึ่งจะทำงานได้ถูกต้อง ส่วนวงจรบันทึกก็เหมือนกันควรจะพิจารณาค่า Rc ที่เหมาะสม

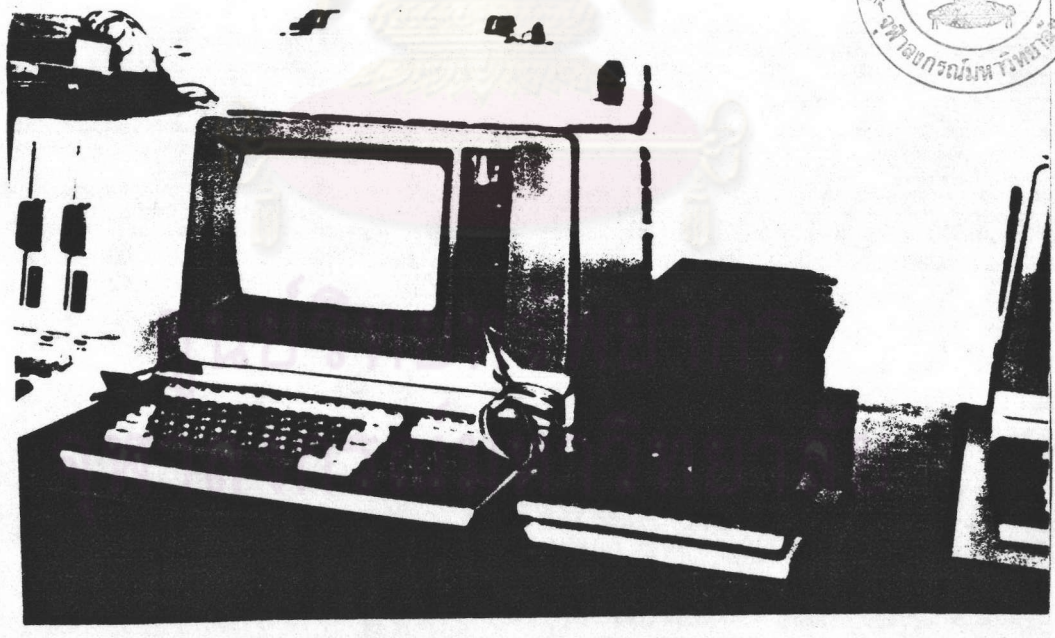
หลังจากผ่านขั้นตอน การสร้างและทดสอบหมดแล้ว ก็ควรจะทำการโปรแกรมอีพรอมใหม่ โดยใช้โปรแกรมระบบเครื่องจริง ซึ่งรายละเอียดของโปรแกรมระบบจะอยู่ในหัวข้อโปรแกรมระบบ หลังจากนั้นค่อยทำการทดสอบอีกครั้งด้วยการใช้งานกับระบบ OPERATING SYSTEM ของซีพีเอ็ม ซึ่งจะกล่าวในหัวข้อต่อไป ในการติดตั้งระบบซีพีเอ็มสำหรับรูปถ่ายตัวจริงของวงจรที่สร้างจะแสดงในรูปที่ 4.47 และรูปที่ 4.48 ส่วนระบบที่สมบูรณ์ จะแสดงในรูปที่ 4.49



รูปที่ 4.47 แสดงรูปถ่ายของบอร์ดควบคุม ซึ่งประกอบด้วย ส่วนซีพียู หน่วยความจำ ส่วนอินเตอร์เฟสเทอร์มินอล และเครื่องพิมพ์



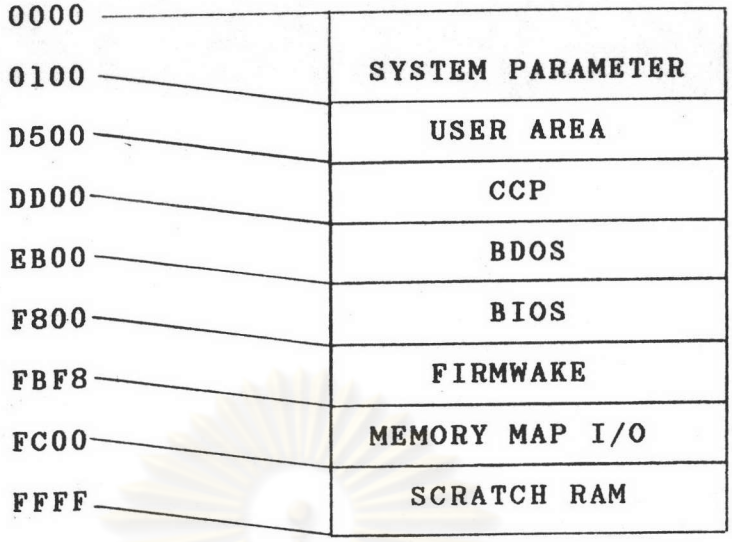
รูปที่ 4.48 แสดงรูปถ่ายของบอร์ดอินเตอร์เฟสตัวขับจานแม่เหล็ก



รูปที่ 4.49 แสดงรูปถ่ายของระบบเครื่องที่ออกแบบสร้าง

4.12 การติดตั้งระบบซีพีเอ็มกับเครื่องที่ออกแบบ

เพื่อให้เครื่องที่ออกแบบสามารถใช้งานกับซอฟท์แวร์ต่าง ๆ ในระบบซีพีเอ็มได้ จึงติดตั้งระบบซีพีเอ็มซึ่งเป็น OPERATING SYSTEM ที่ใช้กับจานแม่เหล็ก เป็นลักษณะ DOS (DOS OPERATING SYSTEM) ให้กับเครื่องที่ออกแบบ จุดประสงค์ก็เพื่อที่จะสามารถนำซอฟท์แวร์ในระบบซีพีเอ็มไปพัฒนาสร้างซอฟท์แวร์ให้กับ ระบบงานเตรียมข้อมูลได้สะดวกและรวดเร็วขึ้น OPERATING SYSTEM ของระบบซีพีเอ็มนี้[19] จะประกอบด้วย ส่วนที่เป็น CCP (CONSOLE COMMAND PROCESSOR) ทำหน้าที่ แปลคำสั่ง และรับคำสั่ง BDOS (BASIC DISK OPERATING SYSTEM) ทำหน้าที่ควบคุมการทำงานภายในจานแม่เหล็ก ไม่ขึ้นกับฮาร์ดแวร์ของเครื่องและ BIOS (BASIC INPUT/OUTPUT SYSTEM) ทำหน้าที่ควบคุมฮาร์ดแวร์ ของส่วนอินเตอร์เฟสไอโอต่าง ๆ ภายในระบบ และขึ้นอยู่กับฮาร์ดแวร์ของแต่ละเครื่องด้วย ดังนั้นจะต้องมีโปรแกรม BIOS สำหรับเครื่องที่ออกแบบโดยเฉพาะ แต่เพื่อลดความยุ่งยากในการเขียนโปรแกรมส่วนนี้ จะนำโปรแกรม CBIOS ของเครื่อง IMSAI 8080 มาพัฒนาดัดแปลงแก้ไขให้สามารถใช้งานกับฮาร์ดแวร์ของเครื่องที่ออกแบบได้รวมทั้งโปรแกรม BOOT ซึ่งทำหน้าที่อ่านโปรแกรมระบบซีพีเอ็มเข้ามาในระบบเครื่อง ก็จะทำการพัฒนาจากโปรแกรม ABOOT ของเครื่อง IMSAI 8080 ด้วย เป็นที่ทราบกันแล้วว่า การติดตั้งระบบหรือทำ SYSTEM GENERATION จะทำกับจานแม่เหล็ก ดังนั้นครั้งแรกจะพัฒนาโดยเครื่อง IMSAI 8080 ก่อน หลังจากนั้นค่อยทำการบันทึกซีพีเอ็ม ที่ผ่านการทำ SYSTEM GENERATION แล้วไว้ในแผ่นจานแม่เหล็กอีกแผ่น เพื่อจะนำไปใช้งาน กับเครื่องที่ออกแบบโดยระบบซีพีเอ็ม ที่ใช้กับเครื่องที่ออกแบบนี้ จะมีขนาด 62 กิโลไบท์ ซึ่งสามารถดูได้จากรูปที่ 4.50



รูปที่ 4.50 แสดง MEMORY MAP ของระบบซีพีเอ็ม ขนาด 62 กิโลไบท์ที่ใช้กับเครื่องที่ออกแบบ

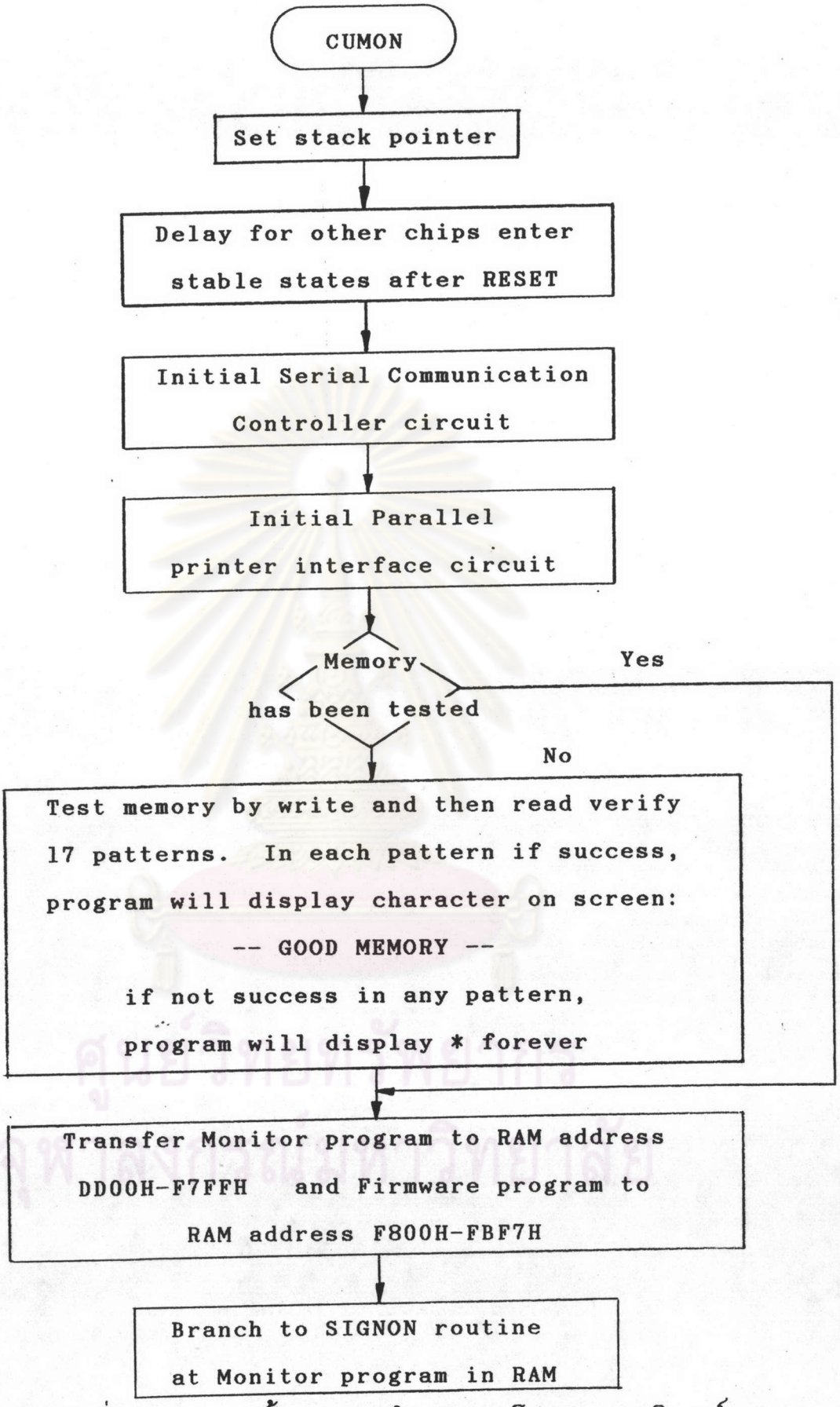
4.12.1 ขั้นตอนการติดตั้งระบบซีพีเอ็มกับเครื่องที่ออกแบบ

ก่อนอื่นจำเป็นจะต้องทราบแอดเดรสของโปรแกรม ABOOT และ CBIOS ของระบบเครื่อง IMSAI 8080 ด้วยว่าอยู่ที่ไหน เพราะเราจะนำโปรแกรมระบบซีพีเอ็มจากเครื่องนี้ มาใช้ติดตั้งกับเครื่องที่ออกแบบ สำหรับ ABOOT จะอยู่ที่แอดเดรส E000H และ CBIOS จะอยู่ที่แอดเดรส D300H เนื่องจาก เป็นระบบซีพีเอ็มขนาด 56 กิโลไบท์ (รายละเอียดของ MEMORY MAP CP/M 56 K ดูได้จากรูปที่ 3.6 ในบทที่ 3) สำหรับเครื่องที่ออกแบบนี้จะใช้ซีพีเอ็มขนาด 62 กิโลไบท์ โดยที่โปรแกรม ABOOT จะอยู่ที่ F800H และ CBIOS จะอยู่ที่ EB00H และเมื่อตอนใช้โปรแกรม MOVCPM เพื่อโหลดซีพีเอ็มเป็นที่ทราบแล้วว่า ในโปรแกรมนี้จะประกอบด้วยโปรแกรม ABOOT และ CBIOS และ CCP DBOS รวมอยู่ด้วยกัน ดังนั้นถ้าจะนำ ABOOT และ CBIOS ของเครื่องที่ออกแบบมารวมกับ CCP และ DBOS เพื่อให้ได้ระบบซีพีเอ็มใหม่ จำเป็นจะต้องทราบตำแหน่งของแอดเดรส ABOOT และ CBIOS ในโปรแกรม MOVCPM ด้วย เพื่อจะได้นำ ABOOT และ CBIOS ของเครื่องที่ออกแบบ ใส่ลงไปแทนที่ของเดิม ซึ่งเราสามารถคำนวณหา และเรียกดูได้จากคำสั่งของโปรแกรม DDT

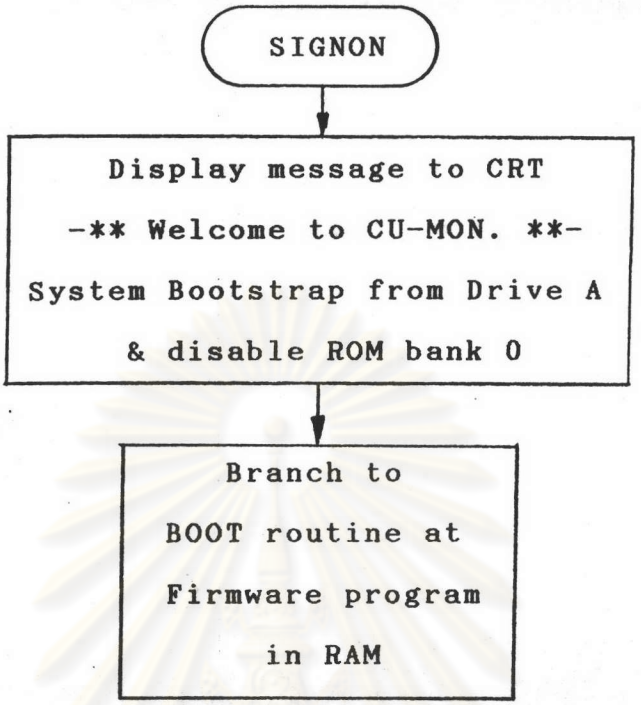
จากการตรวจดูแล้ว จะพบว่า ABOOT จะอยู่ที่แอดเดรส 0900H และ CBIOS จะอยู่ที่ 2700H ในโปรแกรม MOVCPM ดังนั้นก่อนที่จะทำการโหลดโปรแกรม ABOOT และ CBIOS ของเครื่องที่ออกแบบเข้าไปแทน สามารถคำนวณหาแอดเดรสที่จะโหลดได้จาก DDT คือใช้คำสั่ง H900, F800 จะได้แอดเดรส 1100 สำหรับ ABOOT และ H2700, EB00 จะได้แอดเดรส 3C00 สำหรับ CBIOS ซึ่งรายละเอียดขั้นตอนการทำ SYSTEM GENERATION จะแสดงในรูปที่ 4.51 หลังจากทำเสร็จแล้ว จะได้แผ่นจานแม่เหล็กที่บันทึกระบบซีพีเอ็มขนาด 62 กิโลไบท์และสามารถใช้กับเครื่องที่ออกแบบได้ หลังจากนั้นได้ทดลองนำไปใช้กับเครื่องที่ออกแบบปรากฏว่าสามารถใช้งานกับซอฟท์แวร์ในระบบซีพีเอ็มได้แสดงว่า เครื่องที่ออกแบบสร้างนี้ สามารถใช้งานใน OPERATING SYSTEM ของระบบซีพีเอ็มได้อย่างถูกต้อง

```
A>MOVCPM 62 *

CONSTRUCTING 62K CP/M Vers. 2.2
READY FOR "SYSGEN" OR
"SAVE 44 CPM62.COM"
A>
A>SAVE 44 CPM62.COM
A>
A>DDT CPM62.COM
DDT VERS 2.0
NEXT  PC
2D00 0100
-IABOOT#.HEX
-R1100
NEXT  PC
2D00 0000
-ICBIOS#.HEX
-R3C00
NEXT  PC
2D00 0000
-GO

A>
A>SYSGEN
SYSGEN VER 2.2
SOURCE DRIVE NAME (OR RETURN TO SKIP)
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B
DESTINATION ON B, THEN TYPE RETURN
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)

A>
```

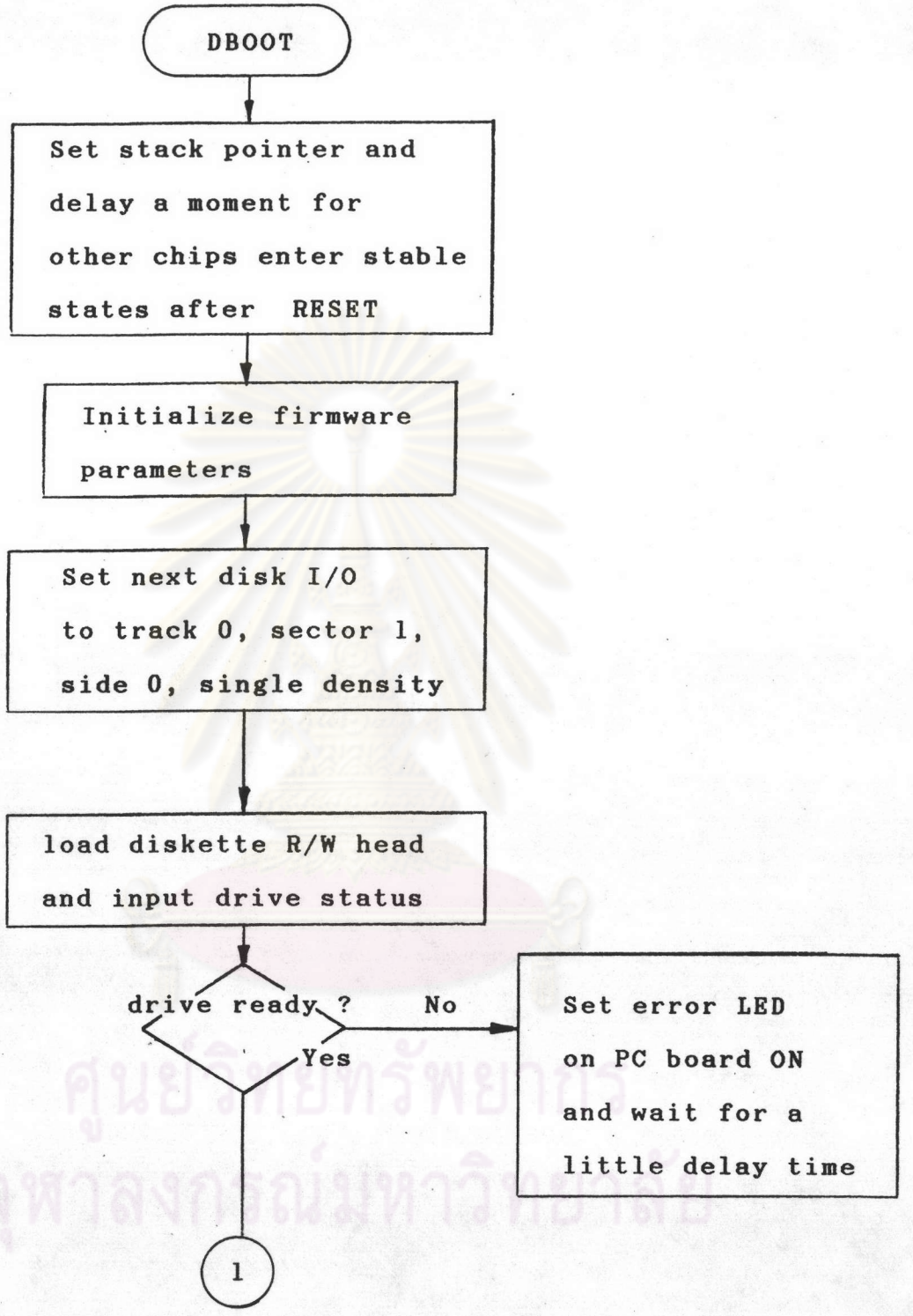
รูปที่ 4.51 แสดงขั้นตอนการทำ SYSTEM GENERATION สำหรับระบบซีพีเอ็ม 62 กิโลไบท์ ให้กับเครื่องที่ออกแบบ



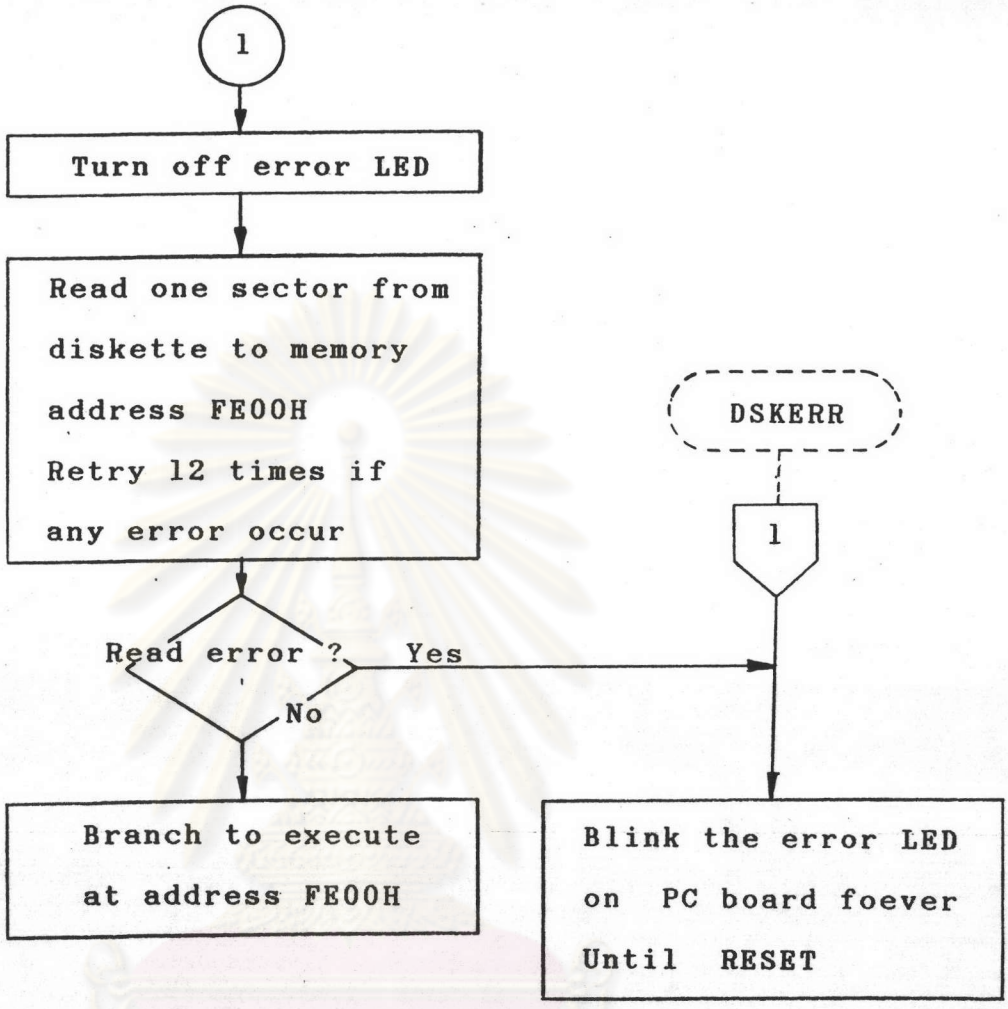
ผังงานที่ 4.1 แสดงขั้นตอนการทำงานของโปรแกรมมอนิเตอร์



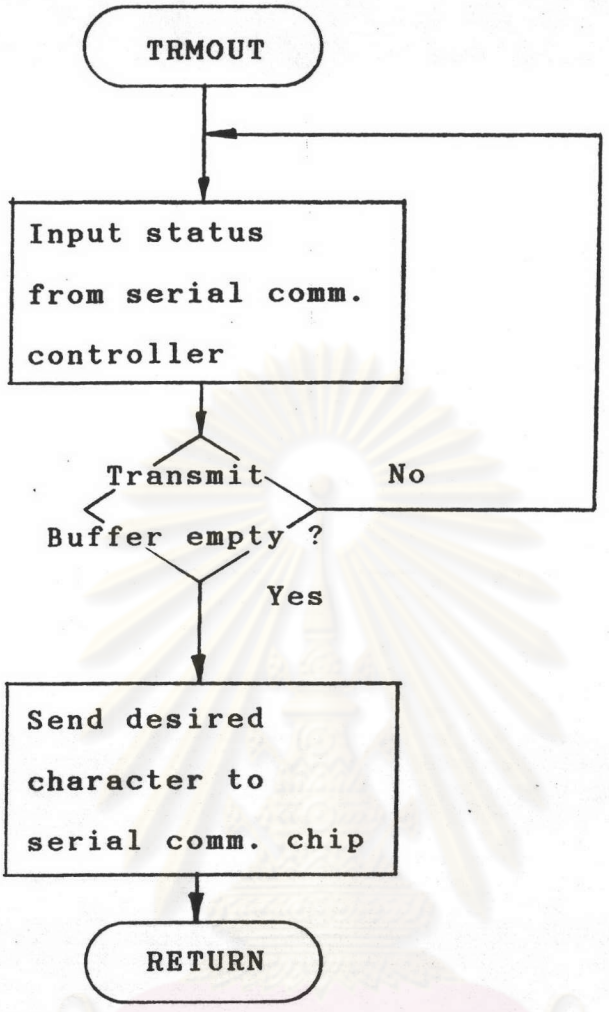
ผังงานที่ 4.1 (ต่อ) แสดงขั้นตอนการทำงานของมอนิเตอร์



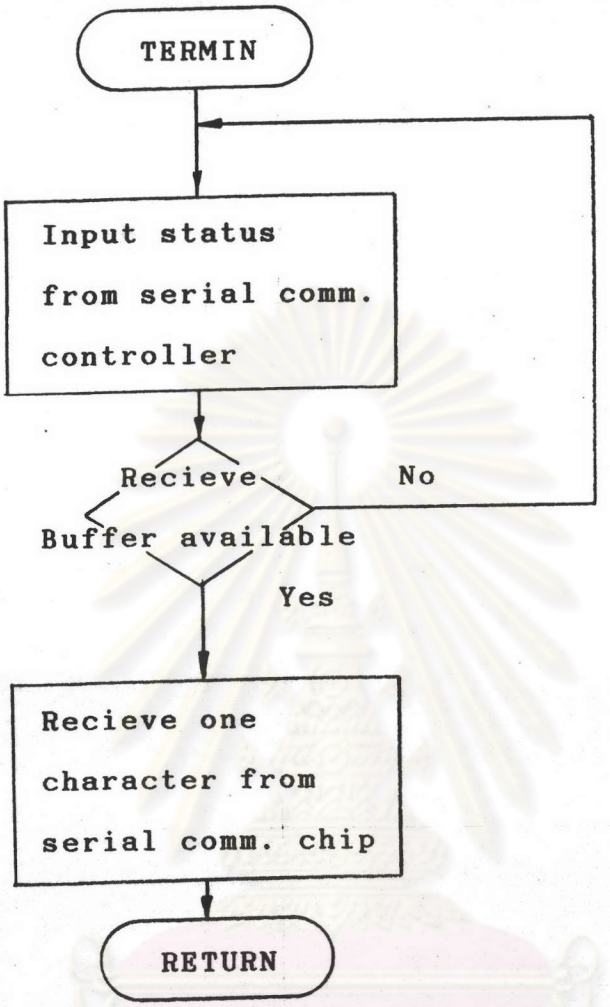
ผังงานที่ 4.2 แสดงขั้นตอนการทำงานของโปรแกรมย่อย BOOT และ DISK ERROR



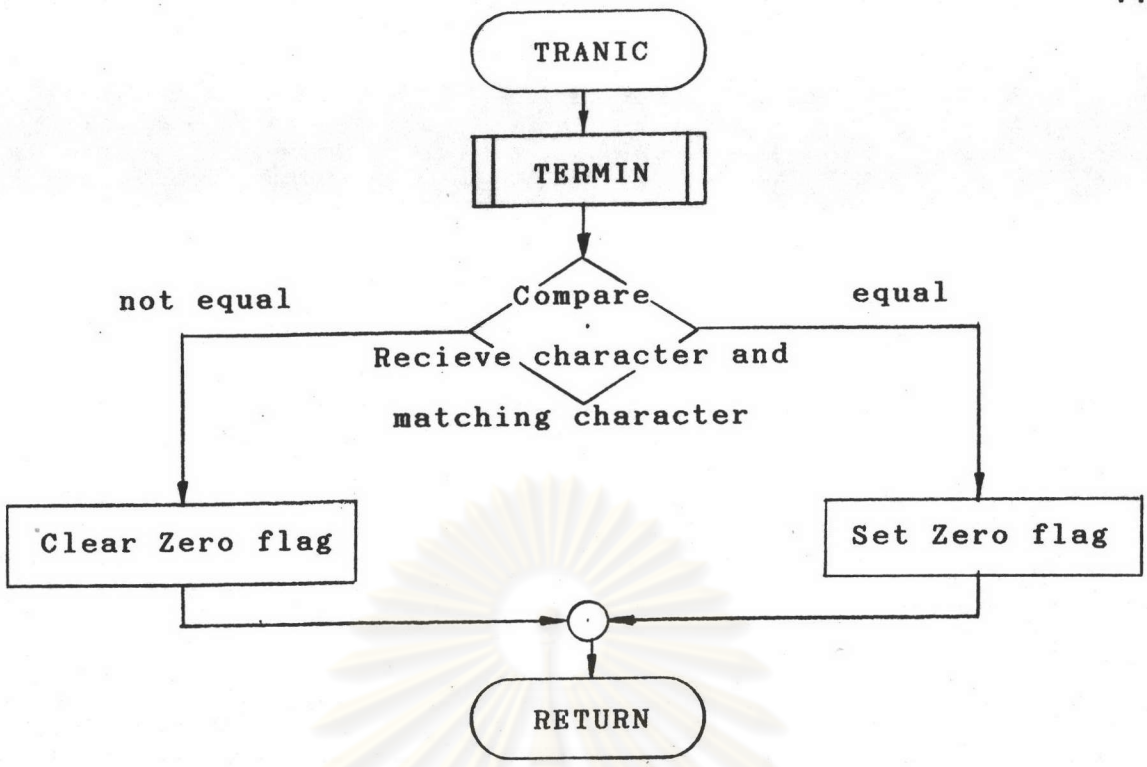
ผังงานที่ 4.2 (ต่อ) แสดงขั้นตอนการทำงานของโปรแกรมย่อย BOOT และ DISK ERROR



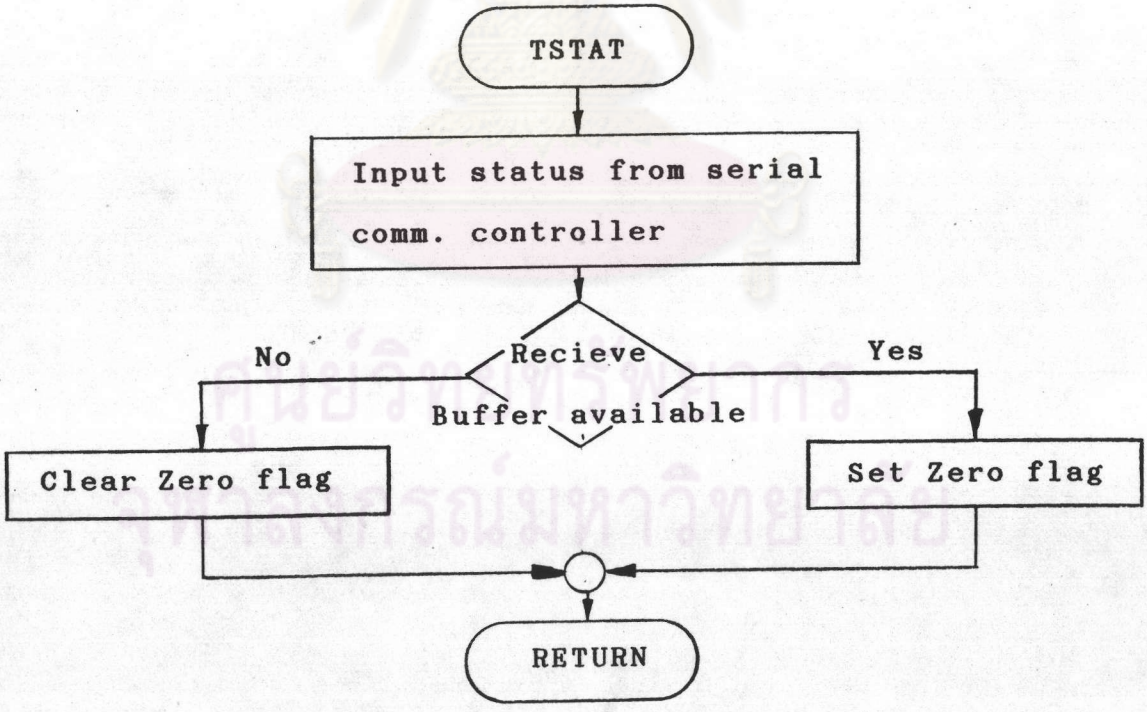
ผังงานที่ 4.3 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
สำหรับส่งข้อมูลให้กับเทอร์มินอล



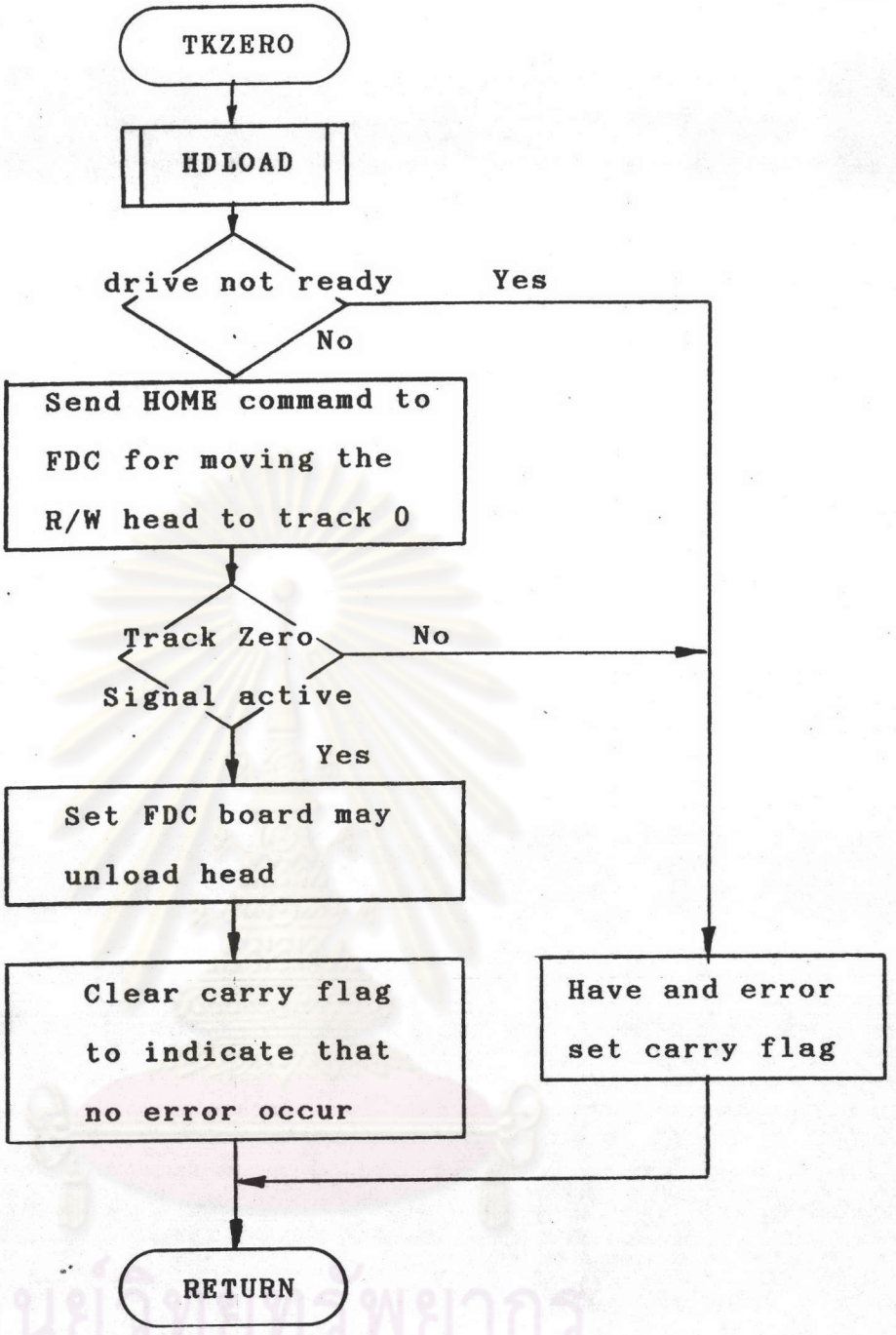
ผังงานที่ 4.4 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
สำหรับรับข้อมูลจากเทอร์มินอล



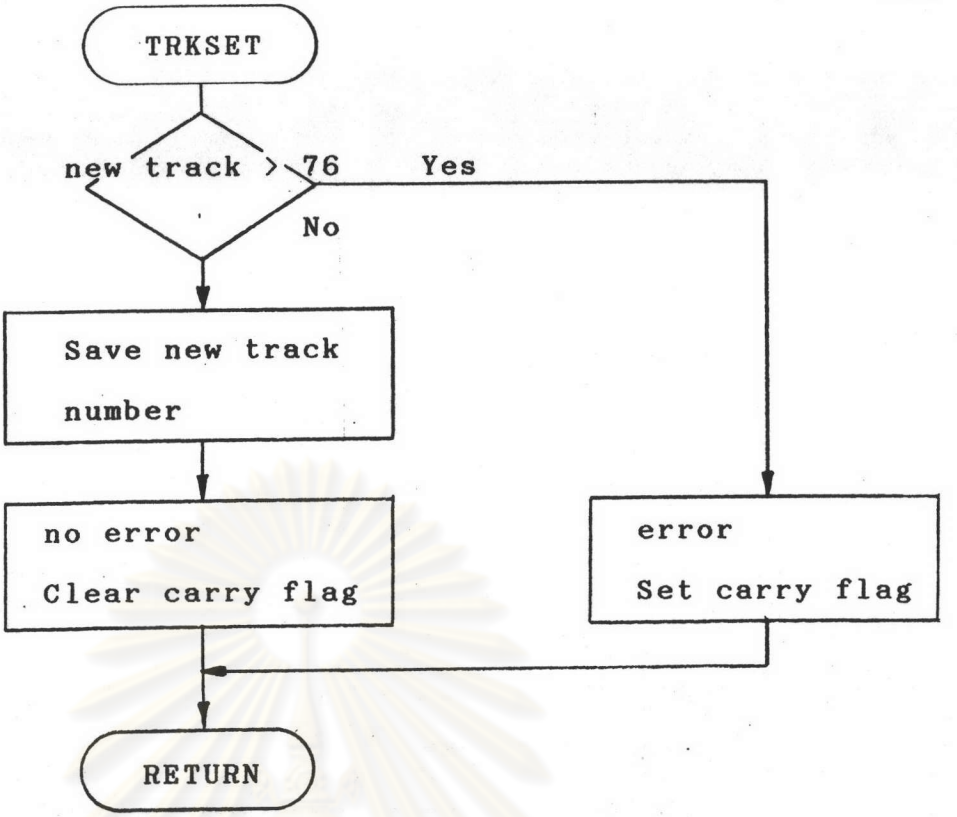
ผังงานที่ 4.5 แสดงขั้นตอนของการทำงานของโปรแกรมย่อย
เพื่อทดสอบข้อมูลจากเทอร์มินอล



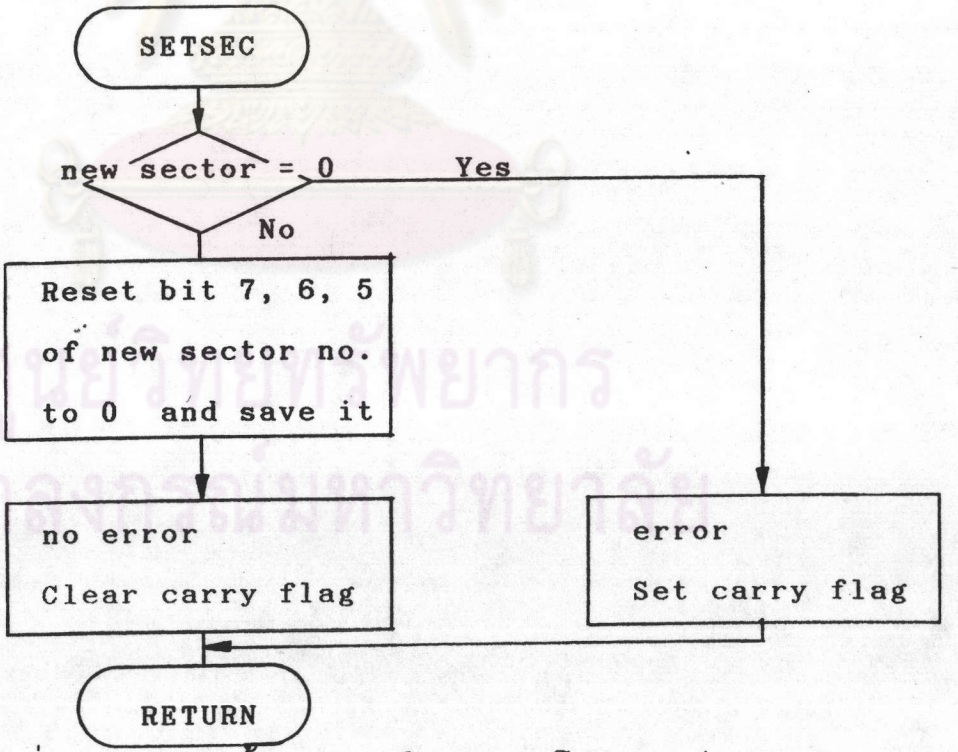
ผังงานที่ 4.6 แสดงขั้นตอน การทำงานของ โปรแกรมย่อย
ในการอ่านสภาวะ (STATUS) ของเทอร์มินอล



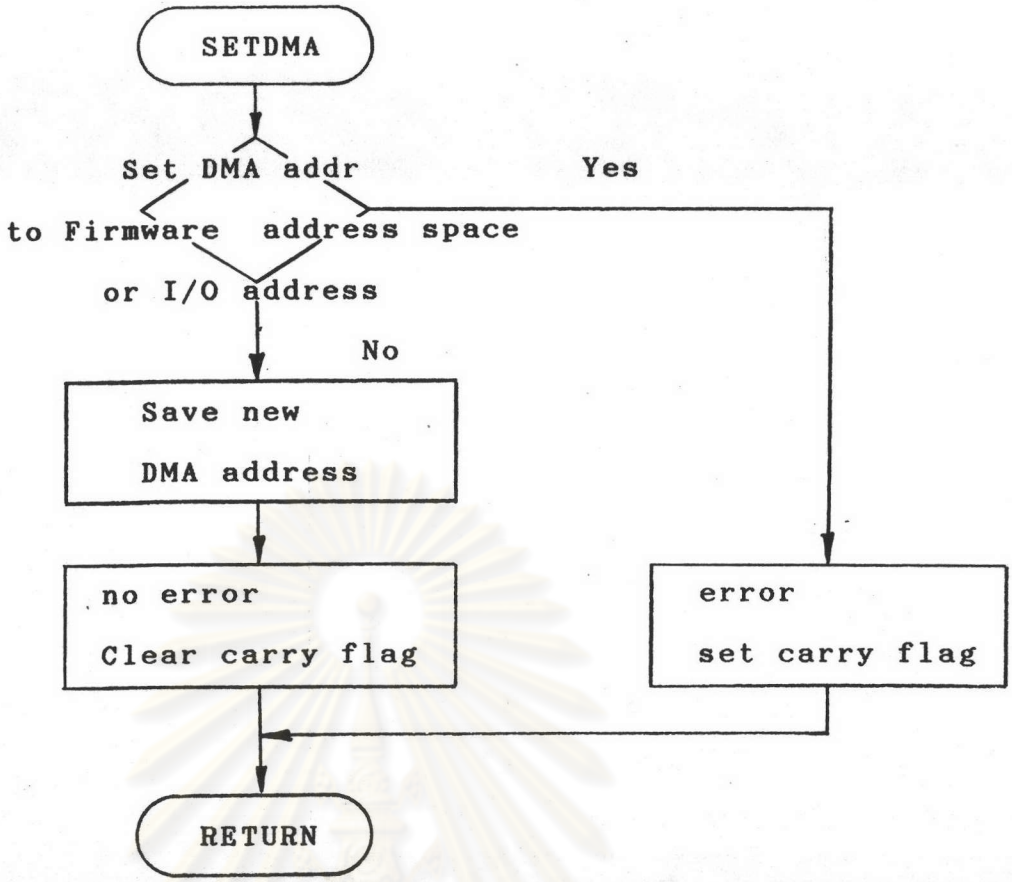
ผังงานที่ 4.7 แสดงตอนการทำงานของโปรแกรมย่อยในการ Recalibrate disk หรือ set track 0



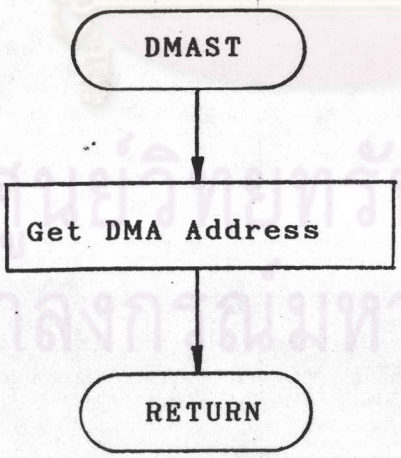
ผังงานที่ 4.8 แสดงขั้นตอนการทำงานของโปรแกรมย่อยในการกำหนดค่าแทรค



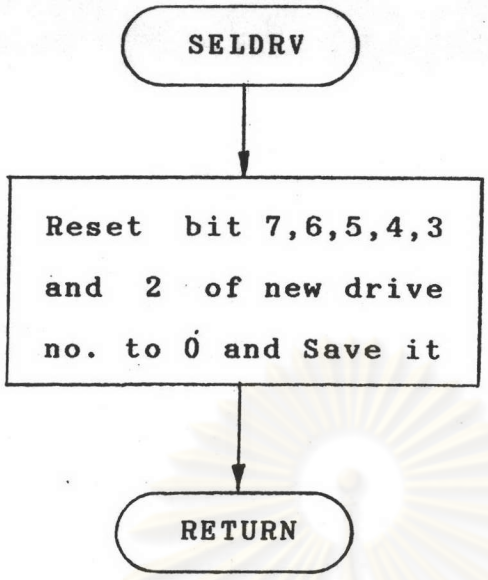
ผังงานที่ 4.9 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการกำหนดค่าเซคเตอร์



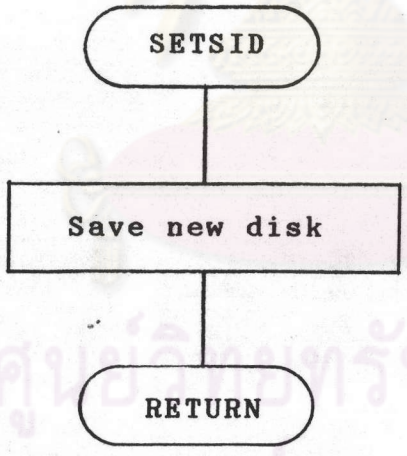
ผังงานที่ 4.10 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการกำหนดค่า DMA ADDRESS



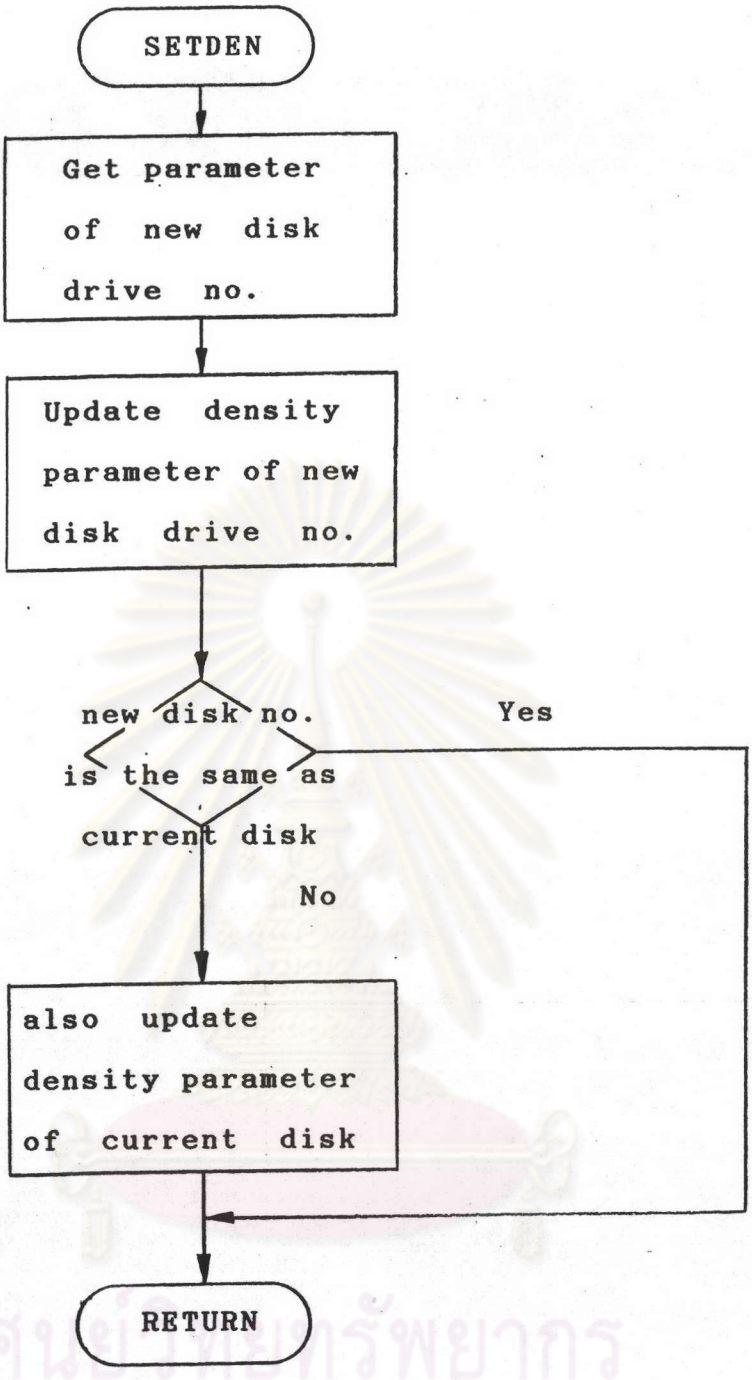
ผังงานที่ 4.11 แสดงขั้นตอนการทำงานของโปรแกรมย่อยในการอ่าน
ค่าสภาวะ (STATUS) ของ DMA Address



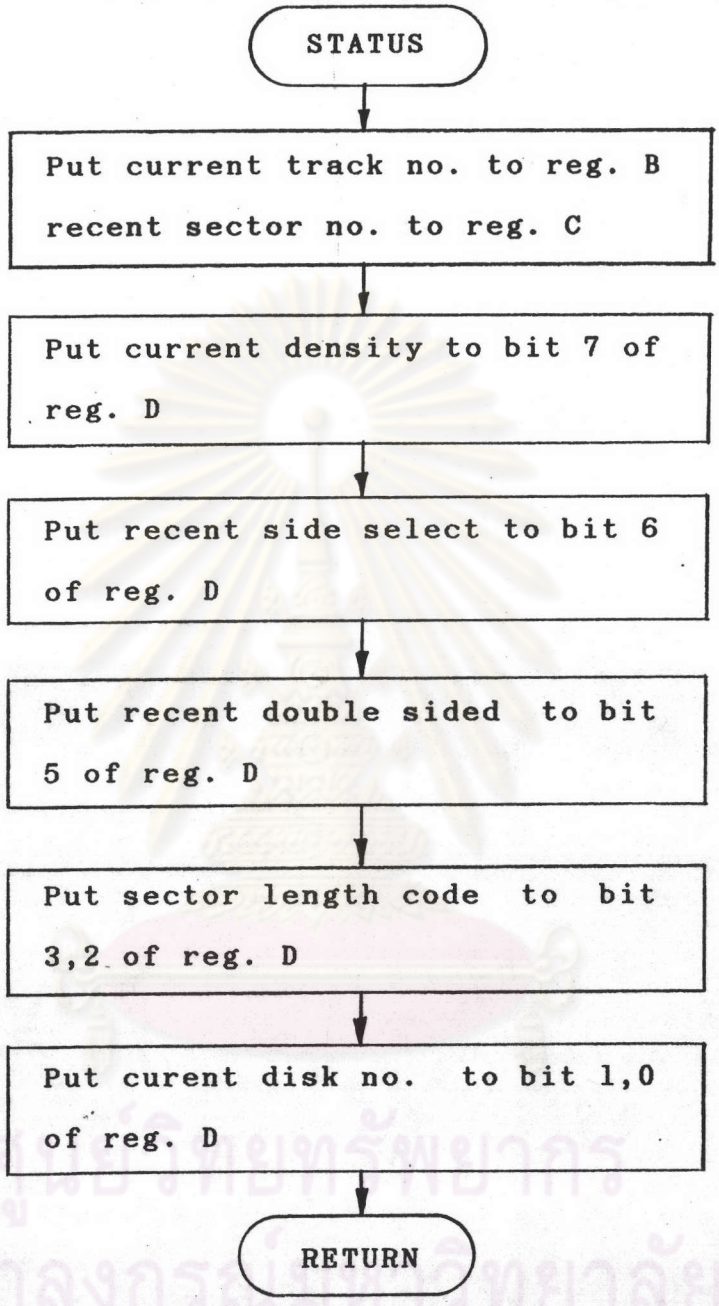
ผังงานที่ 4.12 แสดงขั้นตอนการทำงานของโปรแกรมย่อยในการเลือกไดร์ว



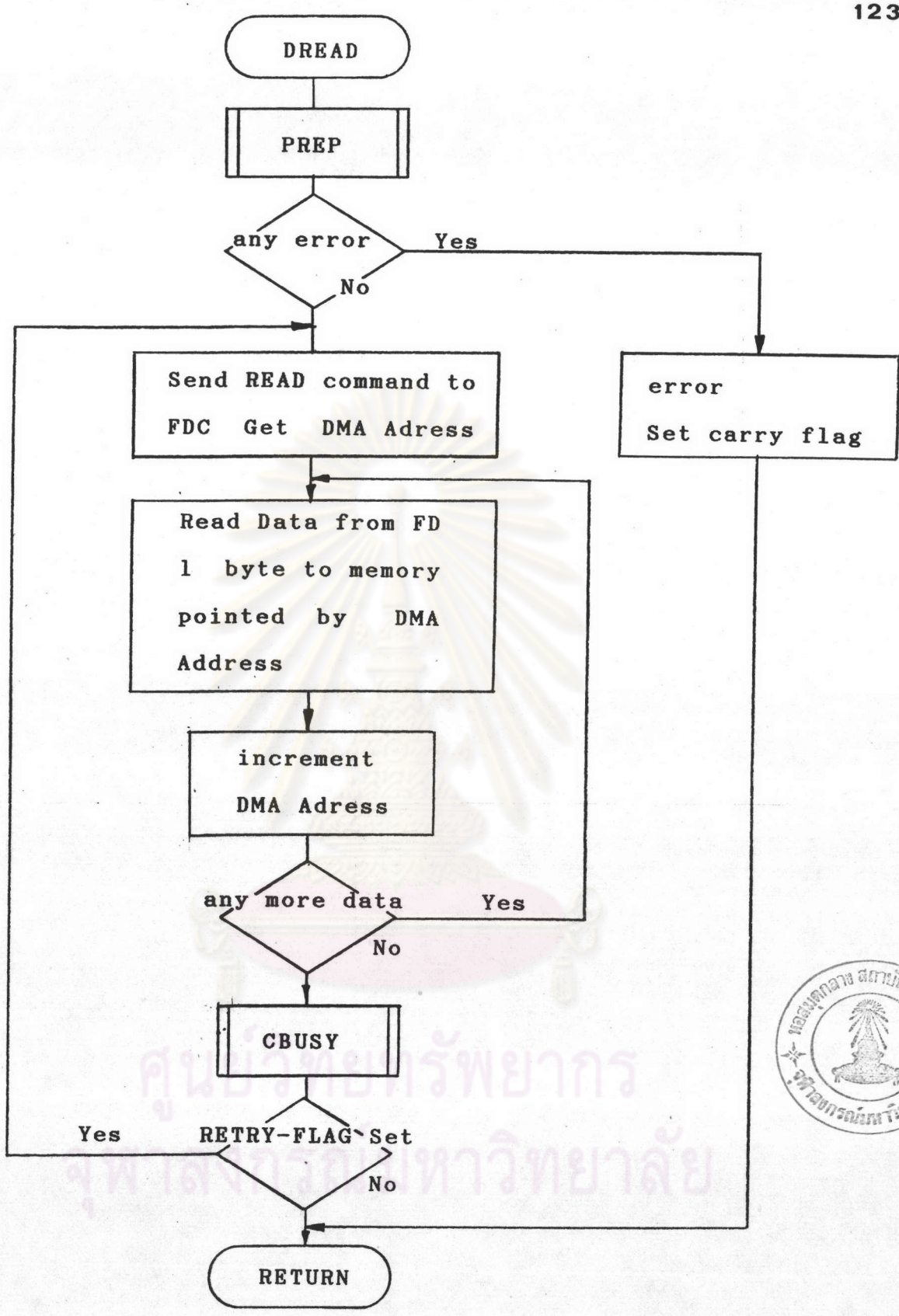
ผังงานที่ 4.13 แสดงขั้นตอน การทำงานของ โปรแกรมย่อย
ในการกำหนดค่าด้าน (SIDE) ของจานแม่เหล็ก



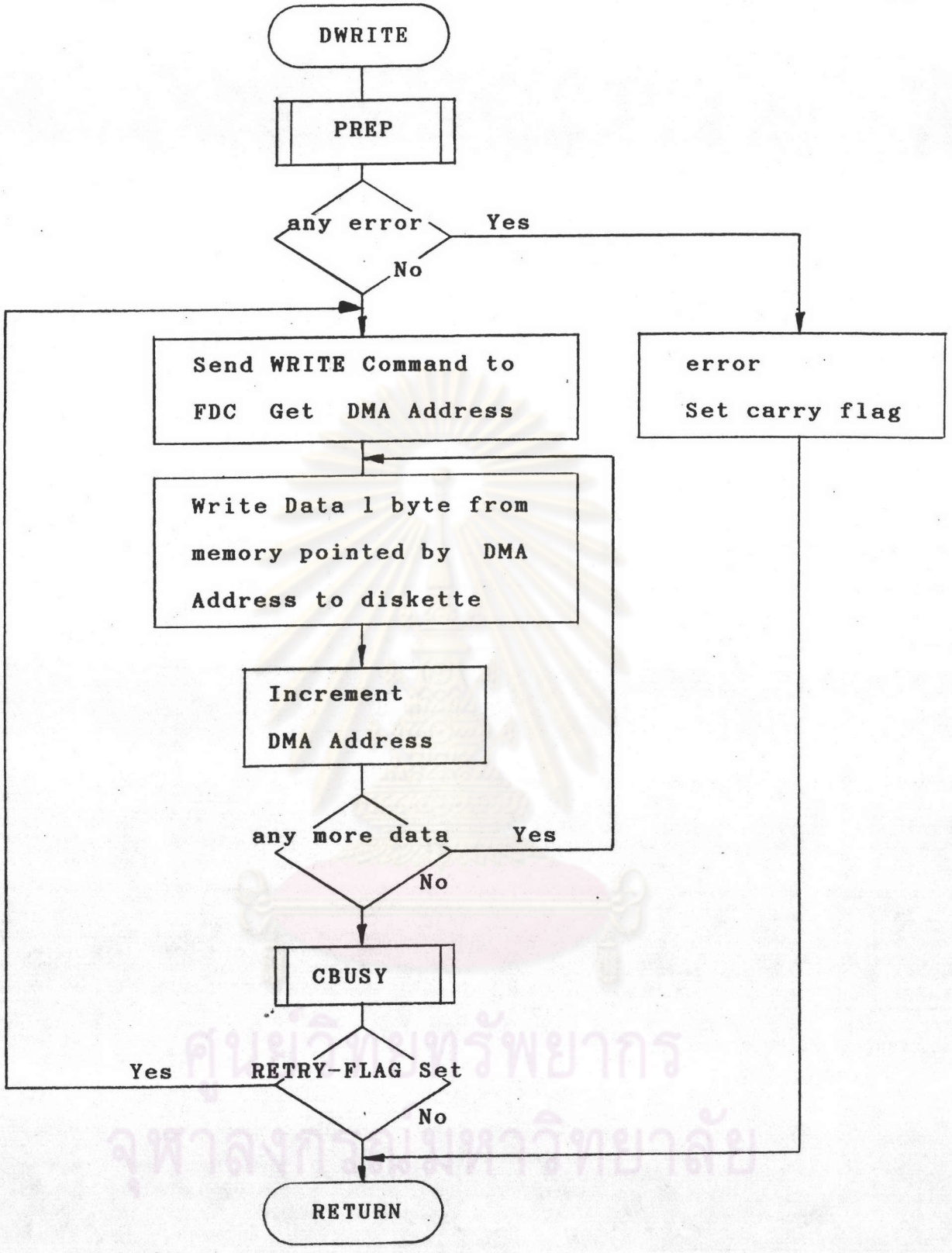
ผังงานที่ 4.14 แสดงขั้นตอน การทำงานของโปรแกรมย่อยในการกำหนดเดนซิตี้ (DENSITY) ของจานแม่เหล็ก



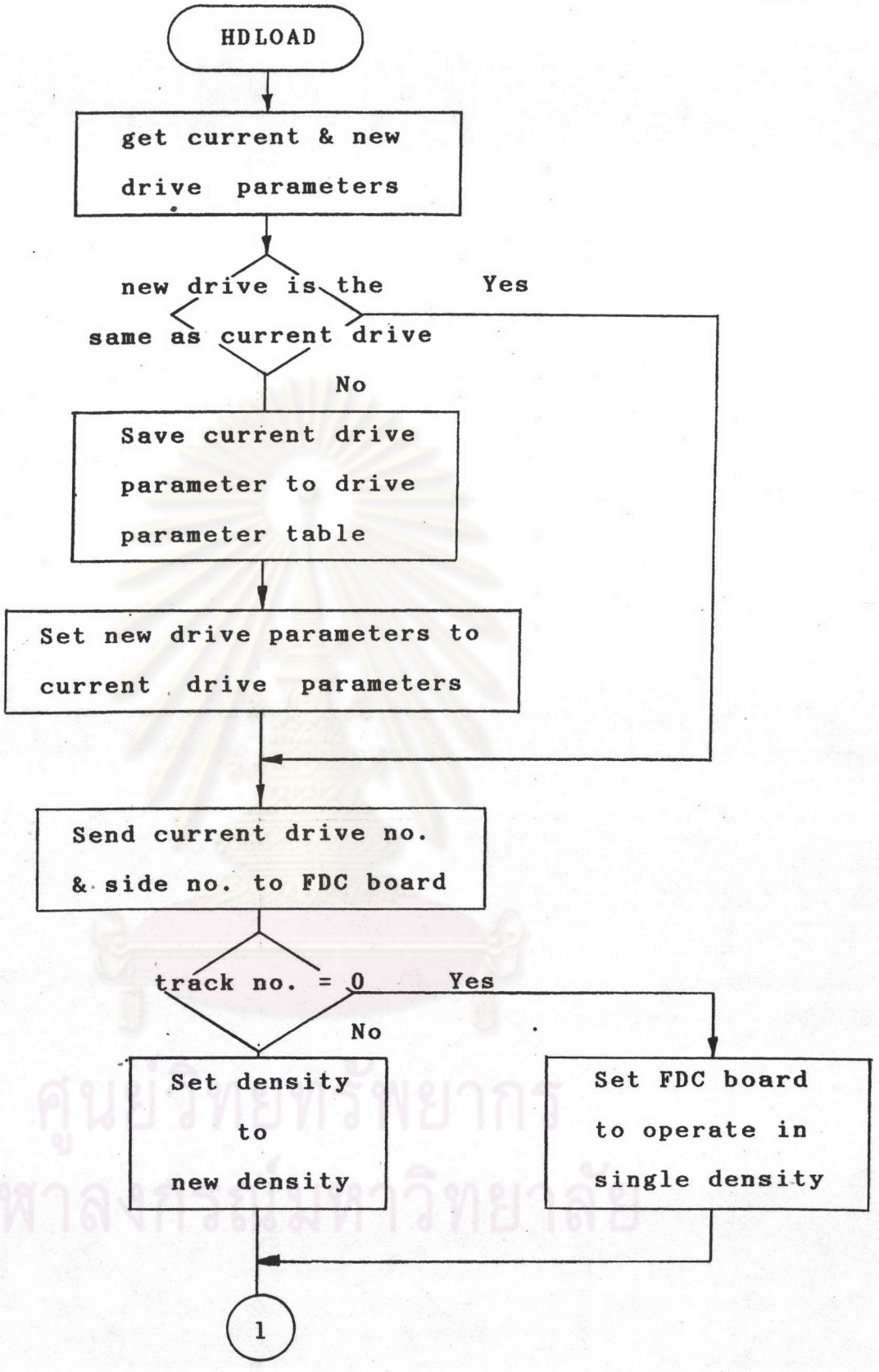
ผังงานที่ 4.15 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการอ่านสภาวะของตัวขับจานแม่เหล็ก



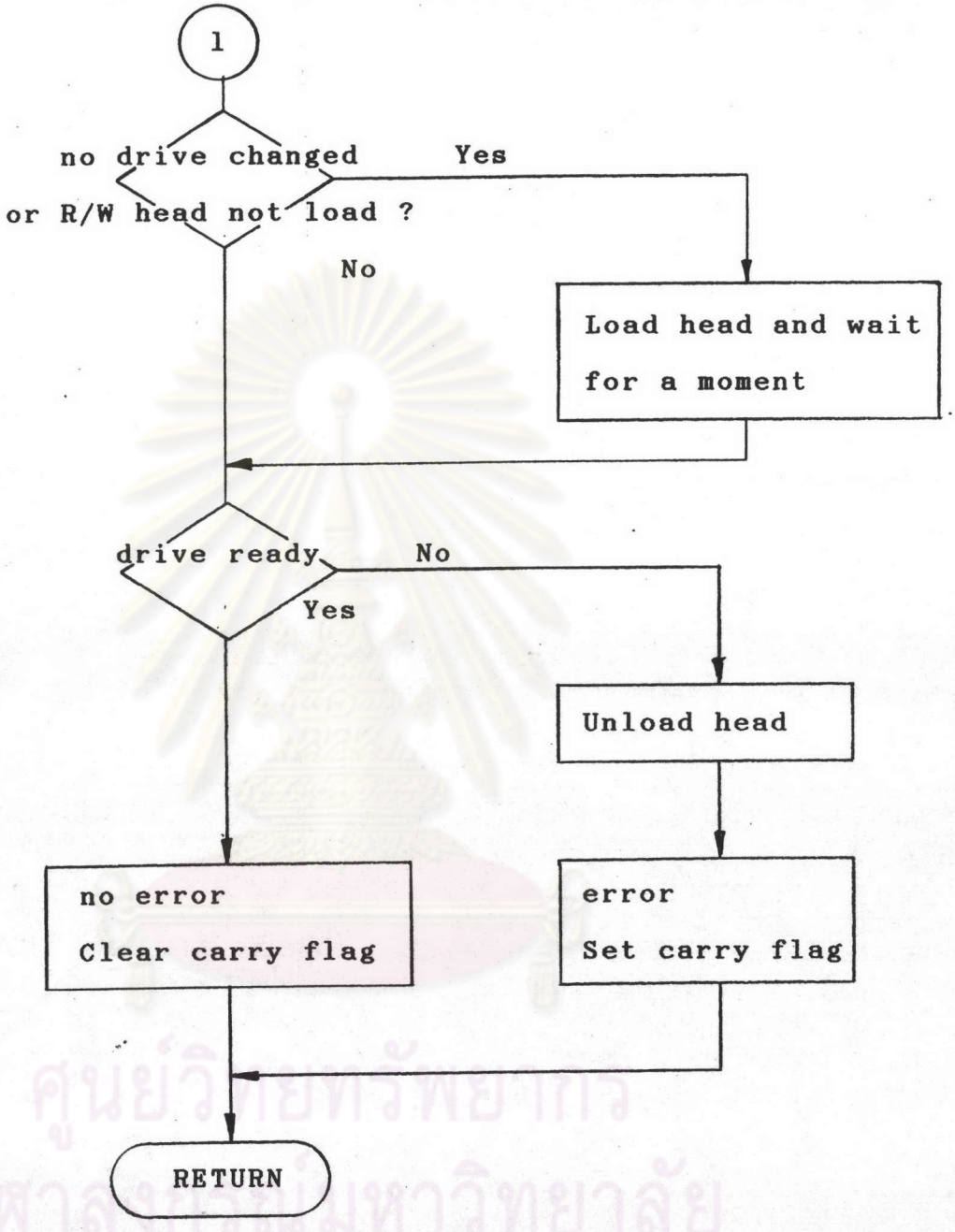
ผังงานที่ 4.16 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการอ่านข้อมูลจากจานแม่เหล็ก



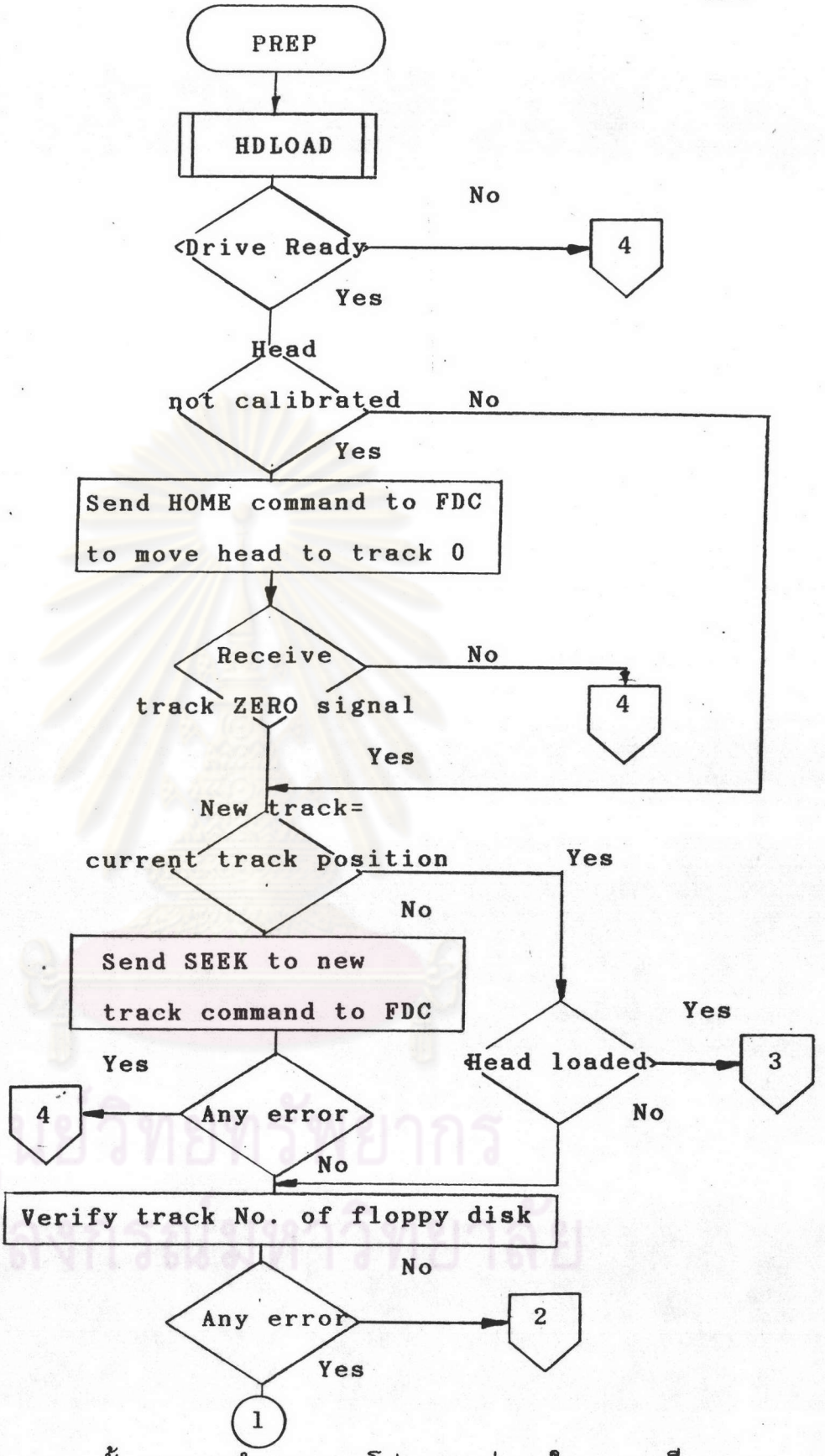
ผังงานที่ 4.17 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการบันทึกข้อมูลลงจานแม่เหล็ก



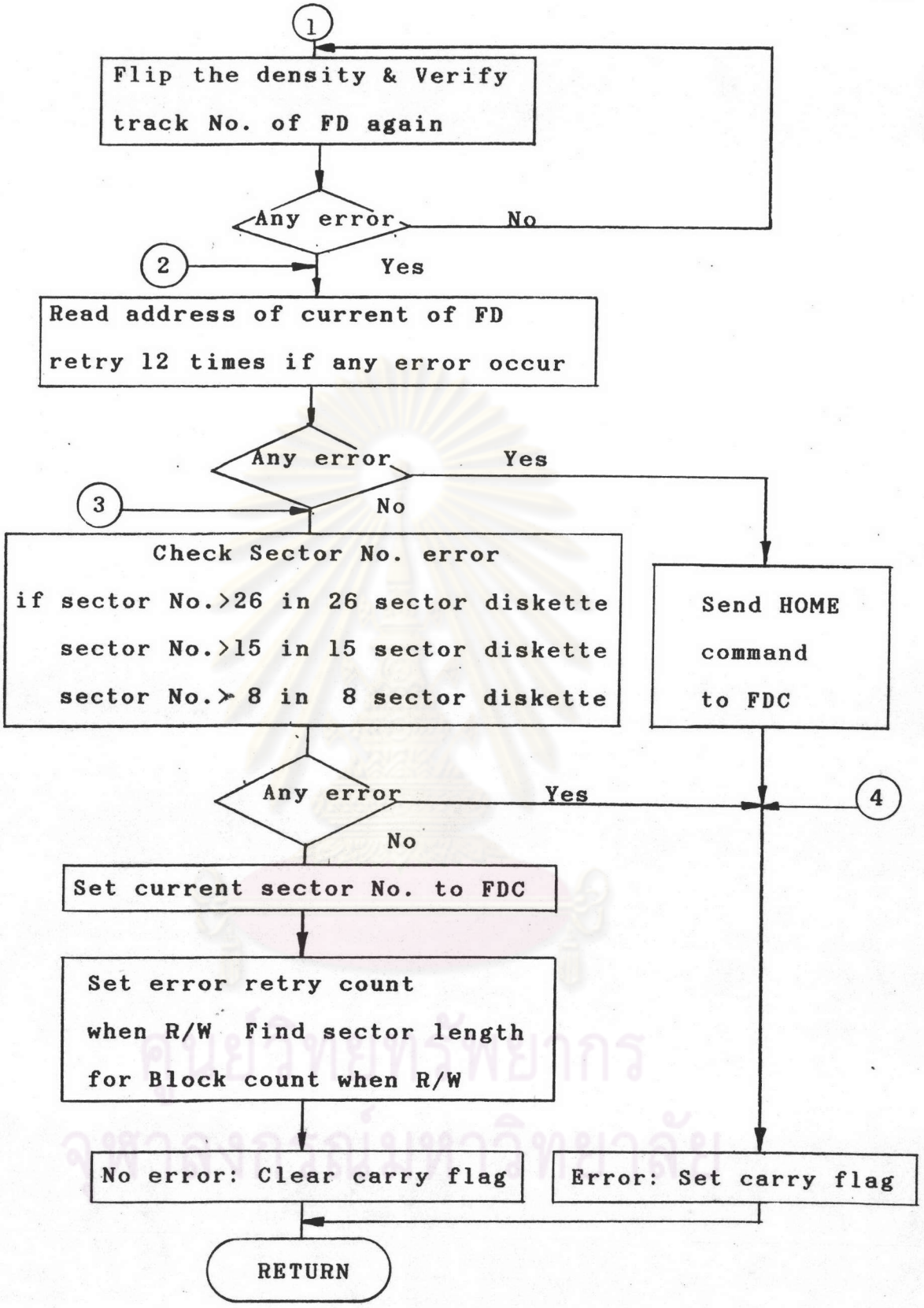
ผังงานที่ 4.18 แสดงขั้นตอนการทำงานของโปรแกรมย่อย ในการปล่อยหัวดิสค์ (HEAD LOAD)



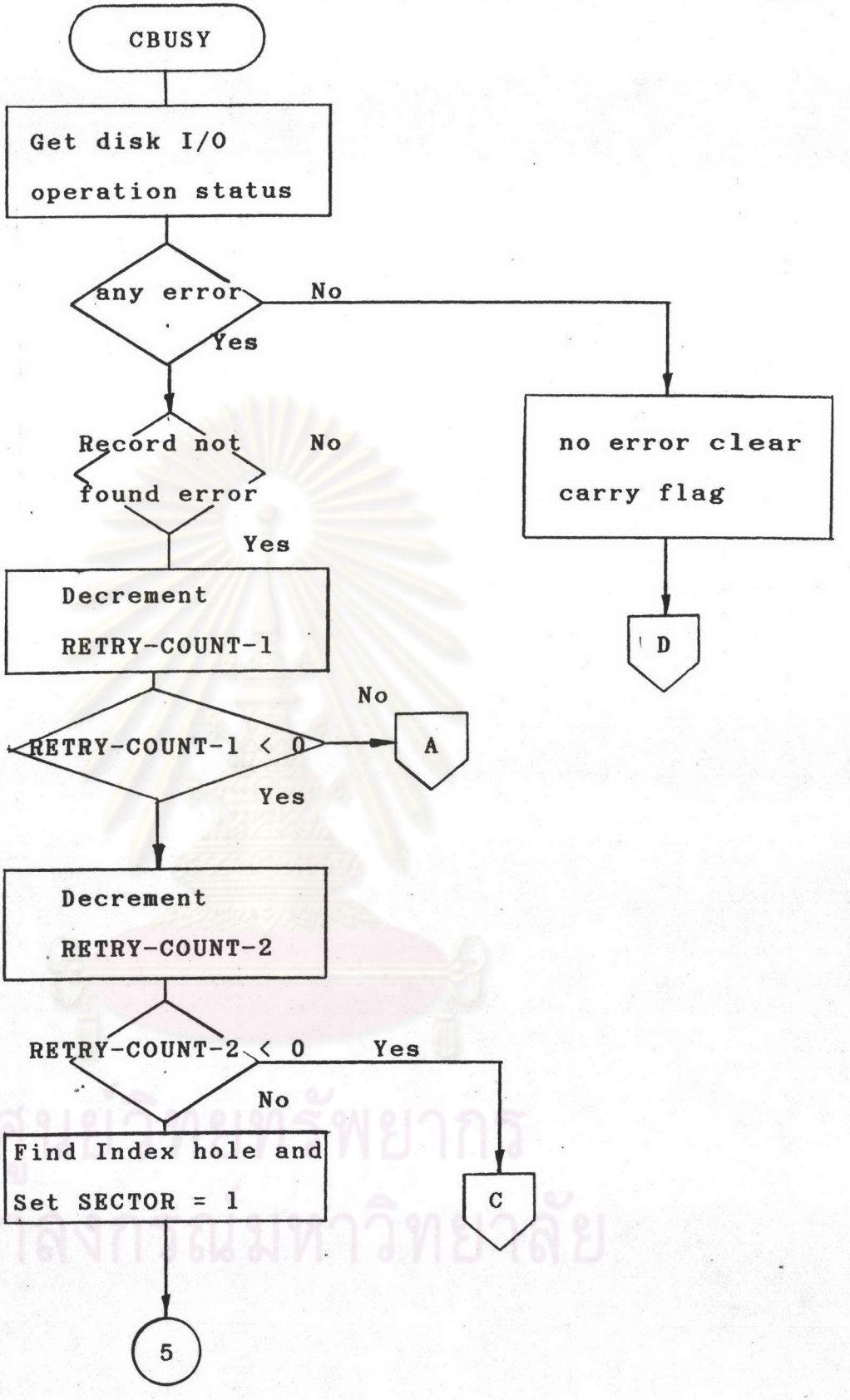
ผังงานที่ 4.18 (ต่อ) แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการปล่อยหัวดิสค์ (HEAD LOAD)



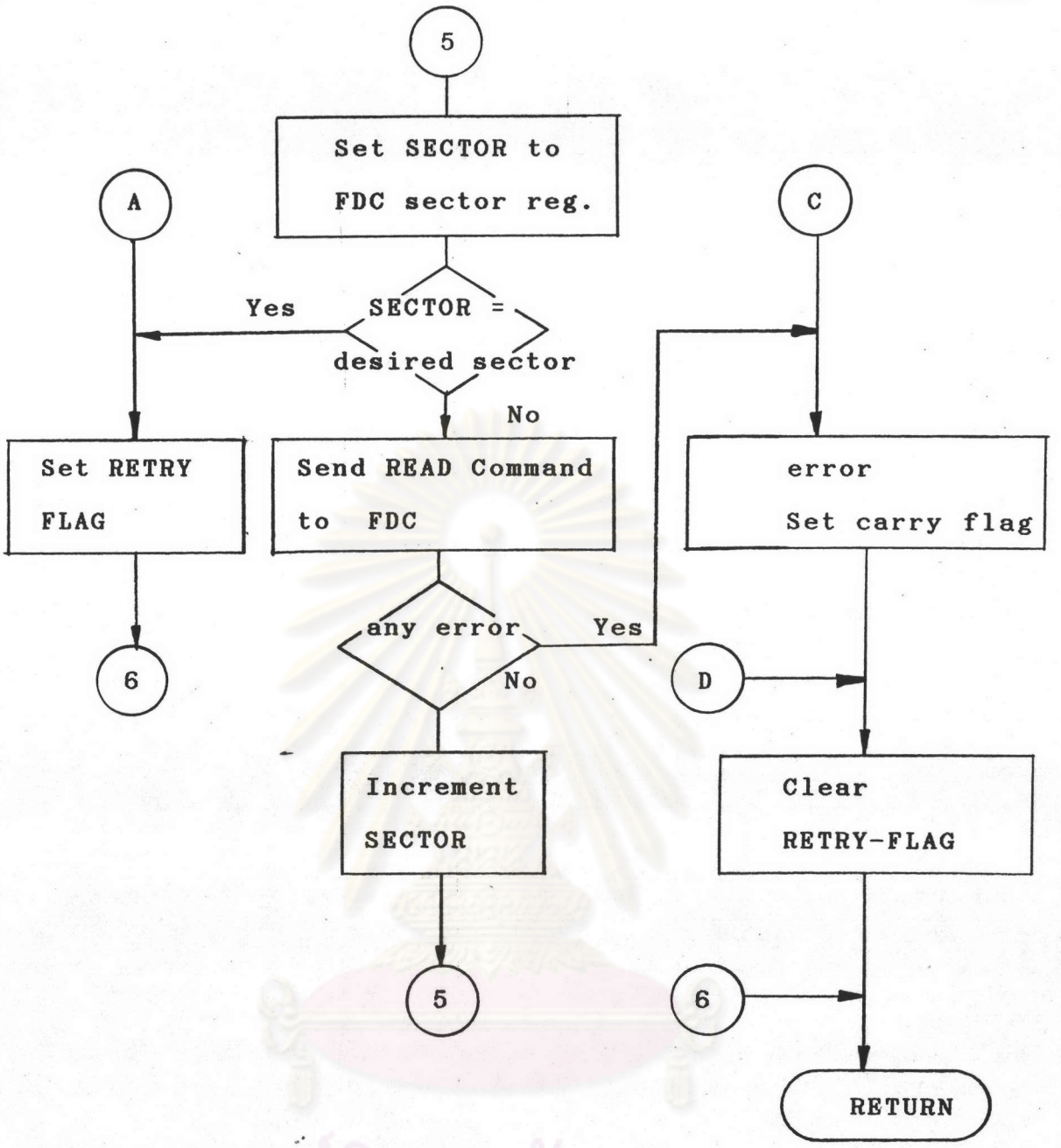
ผังงานที่ 4.19 แสดงขั้นตอนการทำงานของโปรแกรมย่อย ในการเตรียมใช้งานกับตัวขับจานแม่เหล็ก



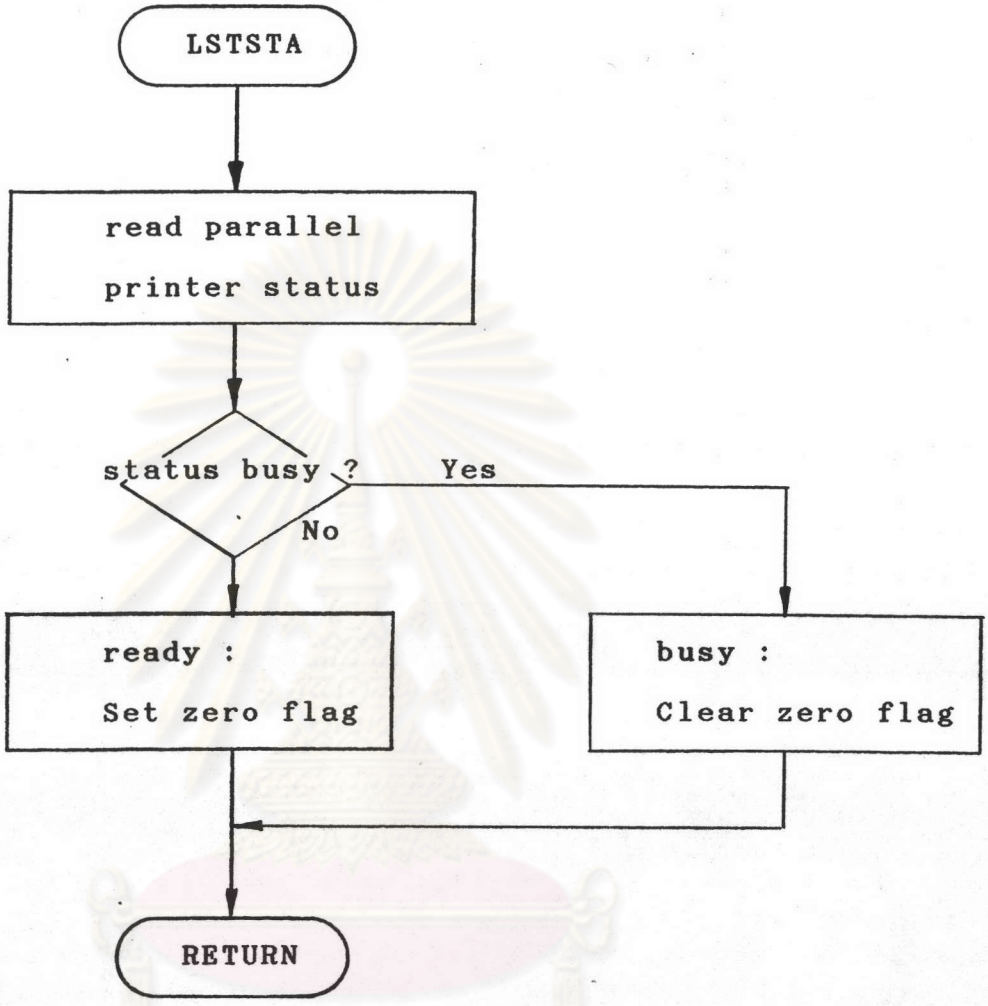
ผังงานที่ 4.19 (ต่อ) แสดงขั้นตอนการทำงานของโปรแกรมย่อย ในการเตรียมใช้งานกับตัวขับจานแม่เหล็ก



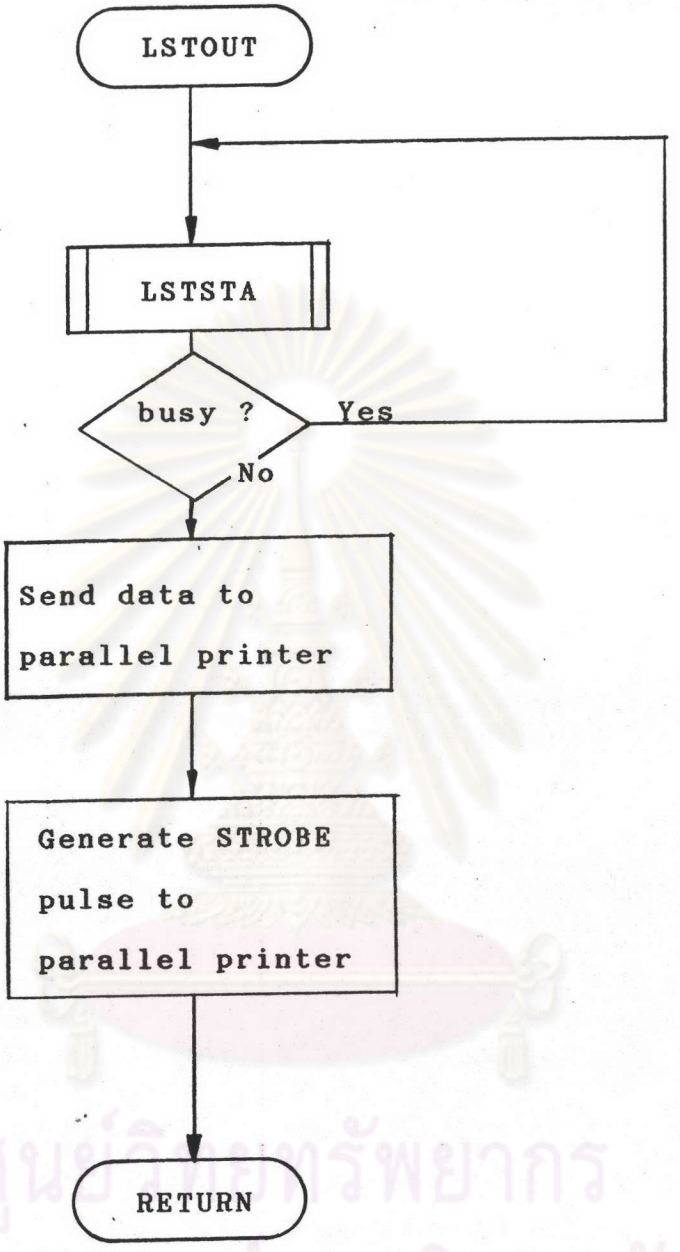
ผังงานที่ 4.20 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการตรวจสอบสภาวะของจานแม่เหล็ก



ผังงานที่ 4.20 (ต่อ) แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการตรวจสอบสภาวะ ของจานแม่เหล็ก



ผังงานที่ 4.21 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
ในการอ่านสภาวะของเครื่องพิมพ์



ผังงานที่ 4.22 แสดงขั้นตอนการทำงานของโปรแกรมย่อย
เพื่อพิมพ์ข้อมูล