ปีการศึกษา 2533
การ์ดวิเคราะห์สัญญาณดิจิตอล

โดย
นายจิรศิลป์ จยาวรรณ เลขประจำตัว 301039
นายจิรศักดิ์ เหลืองอุไร เลขประจำตัว 301040

อาจารย์ที่ปรึกษา
อาจารย์ขนิษฐา แซ่ตั้ง

ปริญญานิพนธ์ปีการศึกษา 2533
ภาควิชา อิเล็คทรอนิคส์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร
เรื่อง การ์ดวิเคราะห์สัญญาณดิจิตอล

ผู้จัดทำ...

1. นายจิรศิลป์ จยาวรรณ เลขประจำตัว 301039
2. นายจิรศักดิ์ เหลืองอุไร เลขประจำตัว 301040

.....อาจารย์ที่ปรึกษา
(อาจารย์ขนิษฐา แซ่ตั้ง)

การ์ดวิเคราะห์สัญญาณดิจิตอล

นายจิรศิลป์ จยาวรรณ 301039
นายจิรศักดิ์ เหลืองอไร 301040
อาจารย์ขนิษฐา แซ่ตั้ง อาจารย์ที่ปรึกษา
ปีการศึกษา 2533

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาและออกแบบการ์ดวิเคราะห์สัญญาณดิจิตอลที่ทำงานร่วมกับเครื่องไมโครคอมพิวเตอร์ของ IBM ให้สามารถใช้วิเคราะห์และตรวจสอบสัญญาณ ในวงจรดิจิตอลได้ 16 ช่องสัญญาณที่ความถี่ในการสุ่มข้อมูลสูงสุด 20 MHz และ 8 ช่อง สัญญาณที่ความถี่ในการสุ่มข้อมูลสูงสุด 40 MHz โดยอาศัยหลักการของการแซมปลิ้ง (Sampling) ข้อมูลด้วยความเร็วสูง และสามารถเปลี่ยนความถี่ในการแซมปลิ้งได้ตั้งแต่ 500 เฮิรทซ์ (Hertz) ถึง 40 เมกกะเฮิรทซ์(Megahertz) ข้อมูลที่ได้จะถูกนำไปเก็บใน หน่วยความจำความเร็วสูง และจะถูกนำไปแสดงผลในรูปแบบต่างๆ ทางจอภาพหรือเครื่อง พิมพ์ของ ไมโครคอมพิวเตอร์ และสามารถเก็บข้อมูลไว้ใช้ภายหลังได้ โดยที่การทำงานทั้ง หมดจะถูก ควบคุมผ่านทางไมโครคอมพิวเตอร์และโปรแกรมควบคุมการใช้งานที่เขียนด้วย ภาษา C

LOGIC ANALYZER CARD

Mr.Chirasak Luangurai
Mr.Chirasil Chayawan
Miss Khanitdha Sae-tang Advisor
1990

Abstract

This project is a development and design of Logic Analyzer Card add on IBM PC whose ability is analyzing and checking the digital signal. It can detect 16 channel of input simultaneously by 20 MHz maximum sampliig rate and 8 channel of input by 40 MHz maximum sampling rate .Sampled datas are stored in the high-speed memory ,which showed on the microcomputer display or printer and also can be save in the storage device .The funtion of this Logic Analyzer Card is control via microcomputer and user interface of controller programme which written by C .

สารบัญ

บทที่ 1
บทนำ

ชนิดของความผิดพลาดในระบบดิจิตอล

ความผิดพลาดที่เกิดในวงจรดิจิตอล อาจจะเป็นผลมาจากทางฮาร์ดแวร์(hardware) หรือ ซอร์ฟแวร์ (software) หรือทั้งสองอย่าง ซึ่งความผิดพลาดทางซอร์ฟแวร์จะเกิดขึ้น จากการผิดพลาดของโปรแกรมการทำงาน) ส่วนความผิดพลาดทางฮาร์ดแวร์อาจเกิดจาก

1. ความผิดพลาดของข้อมูล (No data or wrong data)
2. กลิทช์ (Glitches)
3. สไป๊ค์ (Spikes)
4. เรส (Races)
5. การผิดพลาดในคาบเวลา (Timing error)
6. ริงกิ้ง (Ringing)
7. ระดับแรงดันผิดพลาด (Wrong level)

ความผิดพลาดทางฮาร์ดแวร์ (Hardware faults)

1. ความผิดพลาดของข้อมูล (No data or wrong data) อาจทำให้สายในวงจร เกิดความเสียหาย , การลัดวงจรหรือทำให้ไม่มีไฟเลี้ยงในบางวงจร

2. กลิทช์ (Glitches) เป็นสัญญาณพัลซ์เล็ก ๆ ซึ่งเป็นสัญญาณที่ไม่ต้องการจะพบใน วงจรที่ทำการออกแบบ ในวงจรที่ใช้งานจริง ๆ ต้องกำจัดออกให้หมด

3. สไป๊ค์ (Spikes) เป็นสัญญาณคล้ายๆกับกลิทช์ มักเกิดในสายส่งในระบบบัสที่มีการ คัพพลิง (coulping) โดยตัวเก็บประจุในแต่ละสายส่ง

4. เรส (Races) มีลักษณะคล้ายกับกริทช์และสไพค โดยเกิดจากการรวมสัญญาณที่มี ความเร็วต่างกันในวงจรลอจิกหนึ่งจะทำให้เกิดเป็นสัญญาณพัลซ์เล็กขึ้น

5. ความผิดพลาดในคาบเวลา(Timing error) มักเกิดขึ้นในระบบการส่งข้อมูลโดย บัส เช่นการส่งข้อมูลโดยบัสไปยังอุปกรณ์อีกชุดหนึ่งซึ่งจะมีการสุ่มข้อมูลโดยพัลซ์(seming pulse) ณ ขอบขาลงของพัลซ์ ถ้าพัลซ์ถูกทำให้มาช้ากว่าปกติด้วยเหตุผลใดเหตุผลหนึ่ง และมีผลทำให้พัลซ์ไปปรากฏที่ช่วงที่ข้อมูลมีการเปลี่ยนแปลง จะทำให้ได้ข้อมูลที่ผิดพลาดได้

6.ริงกิ้ง (Ringing) เกิดขึ้นโดยที่สัญญาณขาเข้า (input) เกิดการออสซิลเลท (oscillate) ผ่านระดับแรงดันหลายครั้งก็จะทำให้ข้อมูลที่ได้ผิดพลาด

7. ระดับแรงดันผิดพลาด (Wrong level) ในวงจรทางลอจิกแต่ละตระกูลจะมีระดับ แรงดันที่ใช้ในการทำงานแตกต่างกันซึ่งความผิดพลาดนี้อาจจะเกิดจากการโหลดค่าสูงๆ ของ เกทหรือโดยการอินเทอร์เฟส (interface)

ความผิดพลาดทางซอร์ฟแวร์ (SOFTWARE FAULTS)

มักเกิดจากความผิดพลาดของโปรแกรม ซึ่งแบ่งออกได้ 3 ชนิดคือ

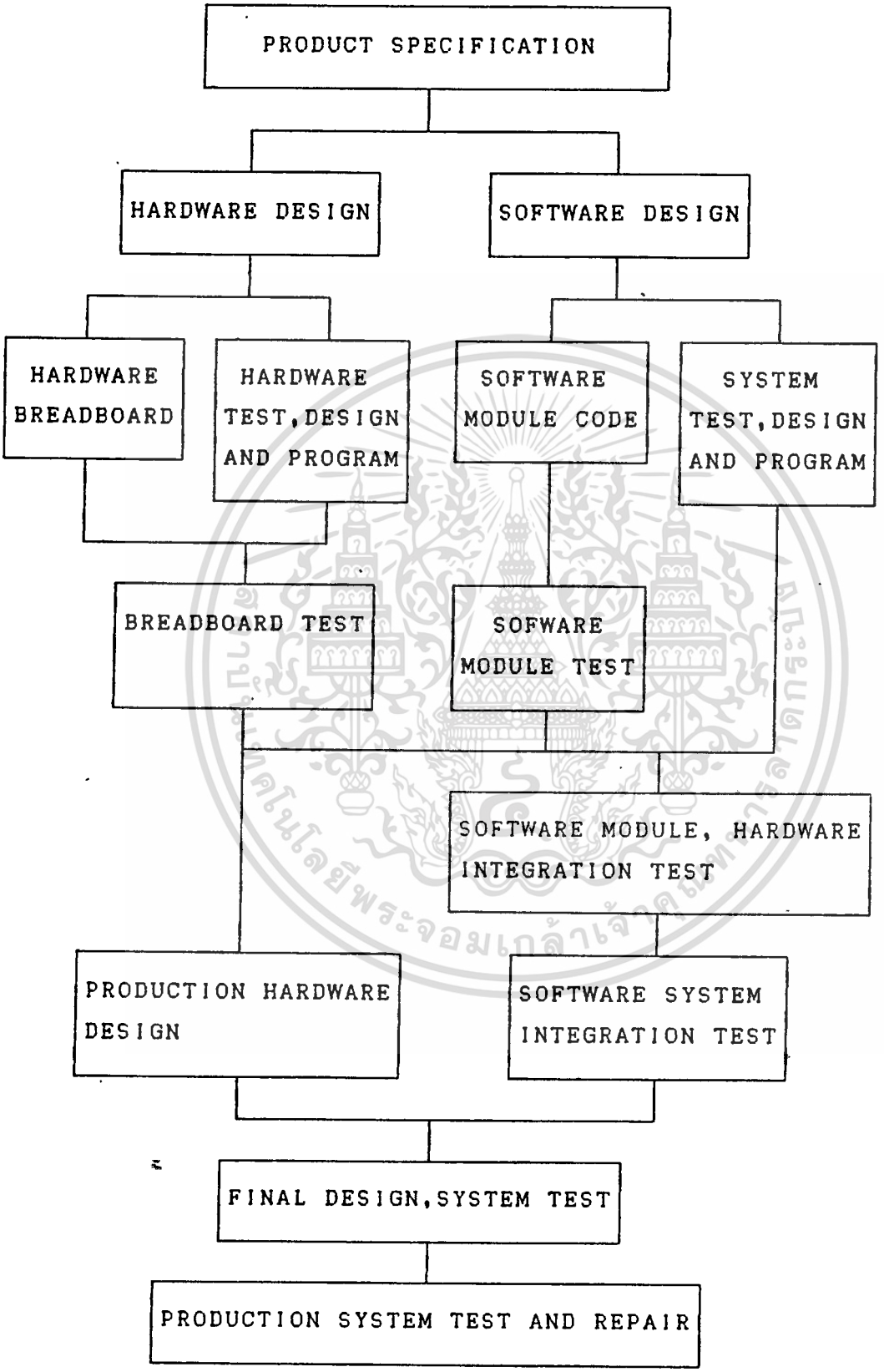
1. ความผิดพลาดของคำสั่ง (Wrong instructions) เกิดการใช้คำสั่งผิดทำให้ การทำงานผิดรูปแบบไป นอกจากนี้อาจเกิดจากความผิดพลาดของแอดเดรส (wrong address)

2. ความผิดพลาดแอบแฝง (Latent faults) ความผิดพลาดของคำสั่งหรือความผิด พลาดทางฮาร์ดแวร์ เช่น คำสั่ง JMP กับ JNZ ที่มีโค๊ดต่างกันเพียง 1 bit

JMP C3 11000011
JNZ C2 11000010

3. การผิดพลาดในคาบเวลา (timing faults) เช่น การติดต่อระหว่างผู้ส่งกับผู้ รับ ถ้าผู้รับมีระบบการทำงานที่ช้ากว่า และโปรแกรมที่ใช้ควบคุมการทำงานไม่ได้โปรแกรม ให้มีการรอคอยการทำงาน ก็จะทำให้เกิดความผิดพลาดขึ้นได้

ในการที่จะตรวจสอบข้อผิดพลาดเหล่านี้ จำเป็นที่จะต้องมีเครื่องมือที่จะสามารถวัด สัญญาณต่างๆเปรียบเทียบกัน ซึ่งลอจิกอนาไลเซอร์สามารถทำงานเช่นนี้ได้ ดังแสดงในแผน ผังภาพที่แสดงกระบวนการในการออกแบบผลิตภัณฑ์ทางด้านดิจิตอลและไมโครโปรเซสเซอร์ ซึ่งจะต้องใช้ทั้งฮาร์ดแวร์และซอฟแวร์ ประกอบกัน ลอจิกอนาไลเซอร์จะถูกใช้ในส่วนที่เป็นการทดสอบการทำงานและแก้ไขข้อบกพร่อง

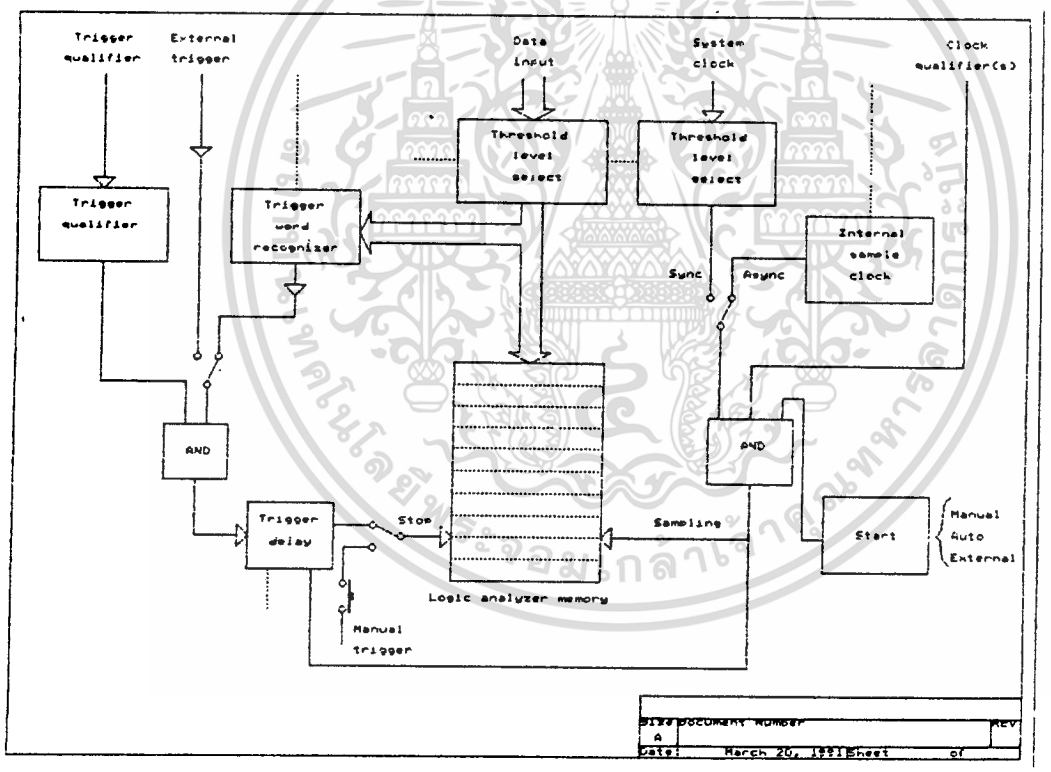


บทที่ 2
หลักการเบื้องต้นของลอจิกอนาไลเซอร์

หน้าที่หลักของลอจิกอนาไลเซอร์ก็คือ การแซมปลิ้ง (Sampling) ข้อมูลนำไปเก็บใน หน่วยความจำ เพื่อใช้สำหรับการแสดงผลและการวิเคราะห์ข้อมูลในภายหลัง ดังนั้นลอ จิกอนาไลเซอร์จะประกอบด้วยส่วนประกอบ 3 ส่วนคือ

1. ภาคควบคุมการสุ่มข้อมูล (DATA ACQUISITION)
2. ภาคควบคุมการหยุดสุ่มข้อมูล (TRIGGER FUNCTION)
3. ภาคแสดงผล (DISPLAY)

เราสามารถเขียนบล็อกไดอะแกรมแสดงส่วนประกอบ และการทำงานของลอจิกอนาไลเซอร์ได้ดังภาพที่ 2-1



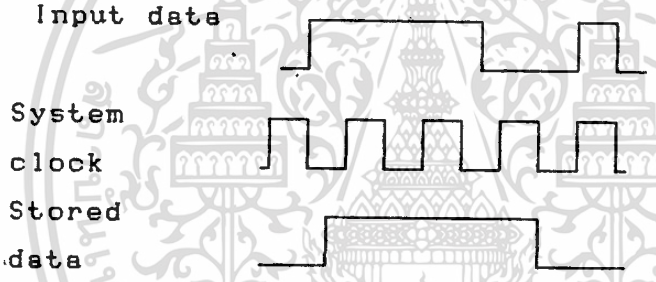
ภาพที่ 2-1 บล็อกไดอะแกรมของลอจิกอนาไลเซอร์

ภาคควบคุมการสุ่มข้อมูล (DATA ACQUISTION)

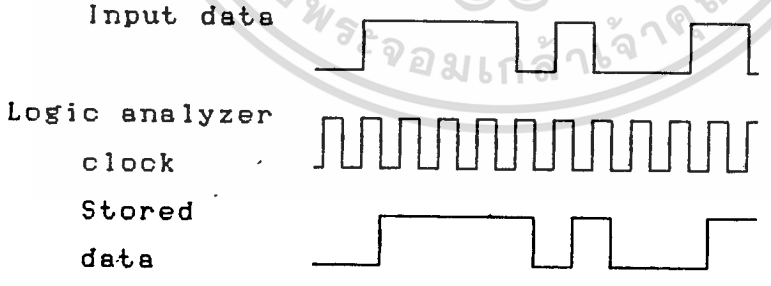
สัญญาณนาฬิกา (CLOCK) ข้อมูลจะถูกสุ่มเข้ามาทุกๆคล็อกพัลซ์ (Clock pulse) จากบล็อกไดอะแกรมจะเห็นว่าเราสามารถเลือกใช้สัญ

ญาณนาฬิกาได้ 2 แบบคือ สัญญาณนาฬิกาของระบบ (System clock) หมายถึงสัญญาณนาฬิกาของวงจรที่เราตรวจสอบซึ่งโดยทั่วไปข้อมูลเข้า จะมีความสัมพันธ์กับสัญญาณนี้ ในกรณีนี้เรียกว่าการทำงานแบบซิงโครนัส (Synchronous Mode) อีกกรณีหนึ่งที่นิยมใช้กันมากกว่าคือใช้สัญญาณนาฬิกาที่สร้างโดยตัวลอจิกอนาไลเซอร์เอง เรียกว่าการทำงานแบบอะซิงโครนัส (Asynchronous Mode) ข้อมูลที่สุ่มจะมีความถูกต้องใกล้เคียงข้อมูลจริงเพียงใดขึ้นอยู่กับความถี่ของสัญญาณ นาฬิกาที่ใช้โดยปกติจะใช้ความถี่สูงกว่าความถี่ของข้อมูลเข้า 5-10 เท่า

เหตุผลหนึ่งที่ต้องมีการทำงานแบบอะซิงโครนัส เพราะว่าสัญญาณบางอย่าง เช่น สไปค์ หรือ กลิทช์ เกิดขึ้นโดยไม่ได้สัมพันธ์กับสัญญาณนาฬิกาของระบบและมีความแคบมาก ทำให้ข้อมูลไม่ถูกแซมปลิ้งถ้าใช้วิธีซิงโครนัสต้องแก้ไขด้วยการเพิ่มความถี่สัญญาณนาฬิกาโดยการทำงานแบบอะซิงโครนัส



ภาพที่ 2-2 ก)การทำงานแบบซิงโครนัส



ภาพที่ 2-2 ข)การทำงานแบบอะซิงโครนัส

การจับสัญญาณกลิทช์ (GLITH-CAPTURING) ในการสุ่มข้อมูลตามปกตินั้นเราใช้ขอบของสัญญาณนาฬิกาเป็นตัวกำหนด เรียกว่าแซมเปิ้ลโหมด(Sample Mode) ซึ่งมีข้อจำกัดคือ ถ้าอินพุทเป็นพัลซ์เล็กๆอยู่ระ

หว่างสัญญาณนาฬิกา และมีขนาดแคบกว่าคาบของสัญญาณนาฬิกา จะไม่ถูกนำไปเก็บ ซึ่งสามารถแก้ไขได้ด้วยการทำงานแบบแลทช์โหมด (Latch Mode) (ภาพที่ 2-3) โดยการใช้มาสเตอร์-สลาฟ ฟลิปฟลอป ความเร็วสูงมาเป็นตัวเก็บพัลซ์ไว้ ก่อนที่จะนำไปเก็บในหน่วยความจำ

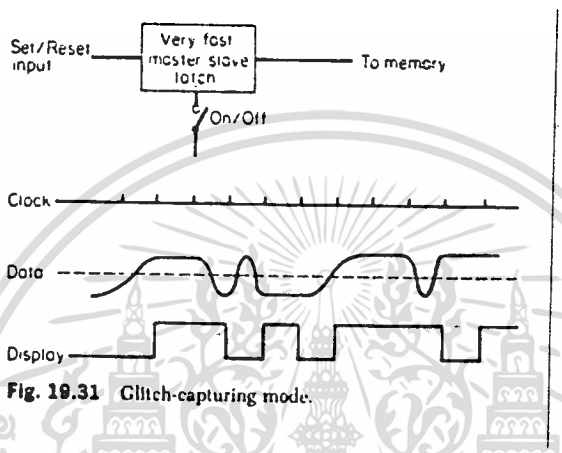
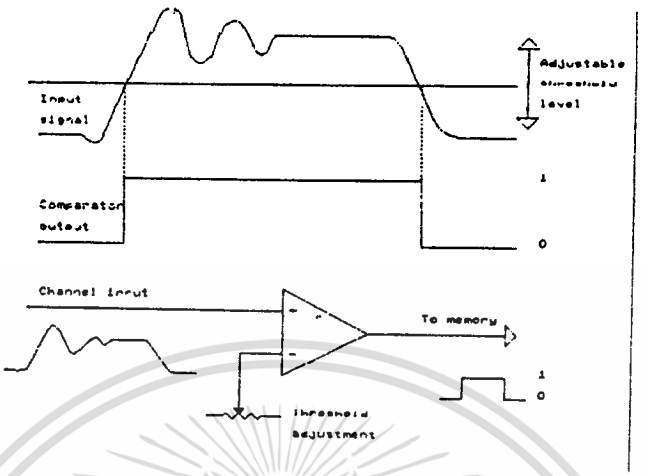


Fig. 19.31 Glitch-capturing mode.

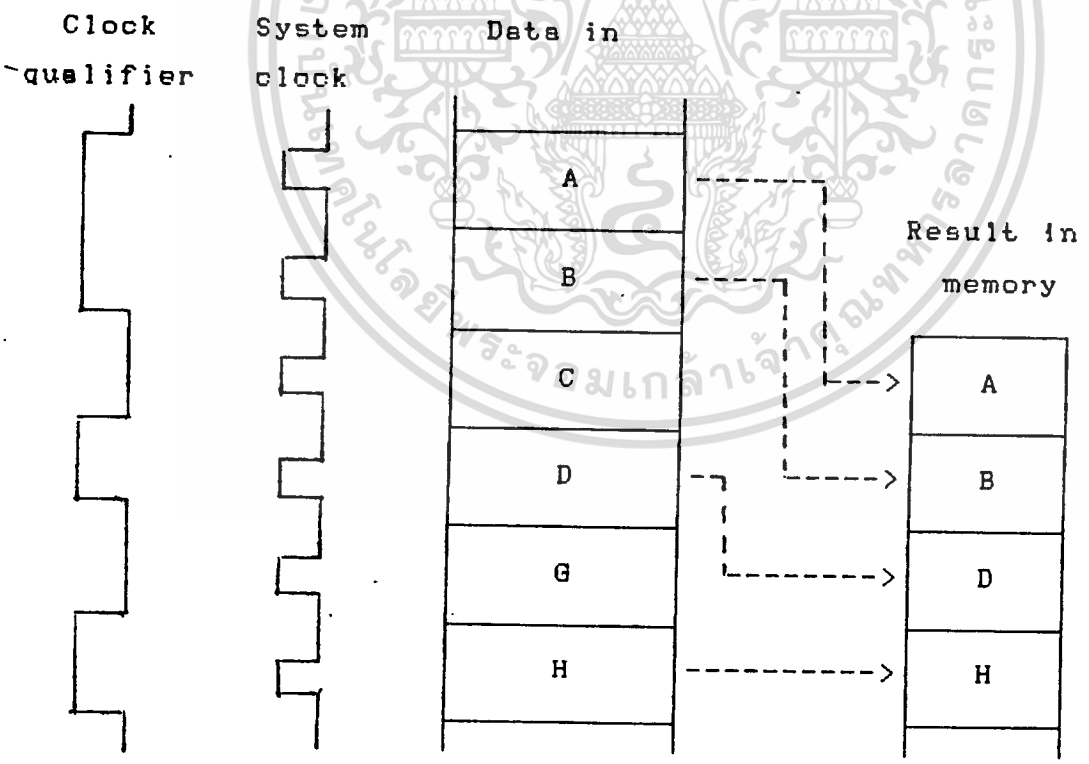
ภาพที่ 2-3 การจับสัญญาณกลิทช์

ข้อมูลเข้า (DATA INPUT) ก่อนที่จะรับข้อมูลเข้ามาเก็บในลอจิกอนาไลเซอร์ต้องเลือกช่วงของระดับอินพุทที่เหมาะสม โดยวิธีเลือกแรงดันเทรสโฮลด์ (Threshold Voltage) ซึ่งขึ้นอยู่กับว่าเป็นลอจิกตระกูลไหน (ตัวอย่างเช่น TTL มีแรงดันเทรสโฮลด์ เท่ากับ 1.4 โวลต์) ภาพที่ 2-4 แสดงถึงการใช้แรงดันเทรสโฮลด์เปลี่ยนระดับสัญญาณ การที่สามารถเปลี่ยนค่าแรงดันเทรสโฮลด์ได้นี้จะมีประโยชน์มากในการตรวจสอบความผิดพลาดที่เกิดจากแรงดันของสัญญาณผิดพลาด

คล็อกควอลิฟายเออร์ (Clock Qualifiers) ในการทำงานแบบซิงโครนัส บางครั้งเราต้องการสุ่มเฉพาะข้อมูลบางช่วงเท่านั้น จึงจำเป็นที่จะต้องควบคุมการปล่อยสัญญาณนาฬิกาให้ได้ตรงกับช่วงเวลาของข้อมูลที่ต้องการ ซึ่งทำได้โดยการนำสัญญาณคล็อกควอลิฟายเออร์มา AND กับสัญญาณนาฬิกา ดังภาพที่ 2-5 จะเห็นข้อมูลที่เราต้องการอยู่ปะปนกับข้อมูลอื่นๆ สมมติให้คล็อกควอลิฟายเออร์คือสัญญาณอ่าน-เขียนหน่วยความจำของระบบที่กำลังทดสอบ โดยถ้าสัญญาณเป็น 1 หมายถึงช่วงการอ่าน และ



ภาพที่ 2-4 การเปลี่ยนระดับสัญญาณข้อมูล



ภาพที่ 2-5 การปรับปรุงสัญญาณนาฬิกา

เราต้องการสุ่มข้อมูลช่วงนี้ เราต้องเซ็ตให้คล็อกควอลิฟายเออร์แอคทีฟที่ลอจิก 1 ดังนั้นข้อมูลที่เราจะเก็บได้ก็คือช่วงที่ระบบที่กำลังทดสอบอ่านหน่วยความจำเท่านั้น วิธีใช้คล็อกควอลิฟายเออร์มีข้อดีอีกอย่างคือ ช่วยให้ใช้หน่วยความจำของลอจิกอนาไลเซอร์ได้อย่างมีประสิทธิภาพ

การเริ่มต้นการทำงาน (Start function) อาจทำได้หลายวิธี เช่นผู้ใช้เป็นผู้ส่ง สัญญาณเริ่มต้นการทำงาน, รับสัญญาณมาจากภายนอก, หรือตั้งเป็นแบบอัตโนมัติทุกๆช่วง เวลาที่กำหนด

ภาคควบคุมการหยุดสุ่มข้อมูล (TRIGGER FUNCTION)

ในช่วงที่การสุ่มข้อมูลดำเนินอยู่ ข้อมูลจะถูกนำไปเก็บในหน่วยความจำตลอดเวลา และจะไม่มีการแสดงผลออกมาให้เห็น ดังนั้นจึงต้องมีส่วนที่ทำหน้าที่ควบคุมการหยุดสุ่มข้อมูล เรียกว่าทริกเกอร์ ส่วนนี้ทำให้เราสามารถเลือกเก็บเฉพาะกลุ่มข้อมูลที่ต้องการได้ทริกเกอร์ แบ่งออกได้เป็น 3 แบบคือ การทริกภายนอก (external triggering), การทริกภายใน (internal triggering)และการทริกด้วยมือ (manual triggering)

การทริกด้วยมือ (Manual triggering) อาจจะใช้เป็นลักษณะของสวิทช์กดเพื่อหยุดการทำงาน โดยปกติแล้วการทริกแบบนี้ไม่ค่อยได้ใช้ มักจะใช้เพื่อหยุดการสุ่มข้อมูลแบบทันที ทันใด

```
Trigger word | 1011 0111 0011 1110 |

Clock          Data Channels
  1        0100 1101 0010 0110
  2        1011 0111 0011 1110 <= Triggering occure
```

ภาพที่ 2-6 ก)ทริกเกอร์เวิร์ด

		Trigger word	
Trigger qualifier	0	1011 0111 0011 1110	
CLock	Q	Data Channels	
1	0	0100 1101 0010 0110	
2	1	1011 0111 0011 1110	<= None triggering
3	1	1010 1100 1110 1001	
4	1	0001 1101 0101 1101	
5	1	1110 0011 1011 0111	
6	0	0000 1010 1000 1000	
7	0	1100 1100 0111 0001	
8	1	1010 0010 1011 1000	
9	1	0110 1101 1100 0111	
10	0	1011 0111 0011 1110	<=Triggering occure

ภาพที่ 2-6 ข)ทริกเกอร์เวิร์ดและทริกเกอร์ควอลิฟายเออร์

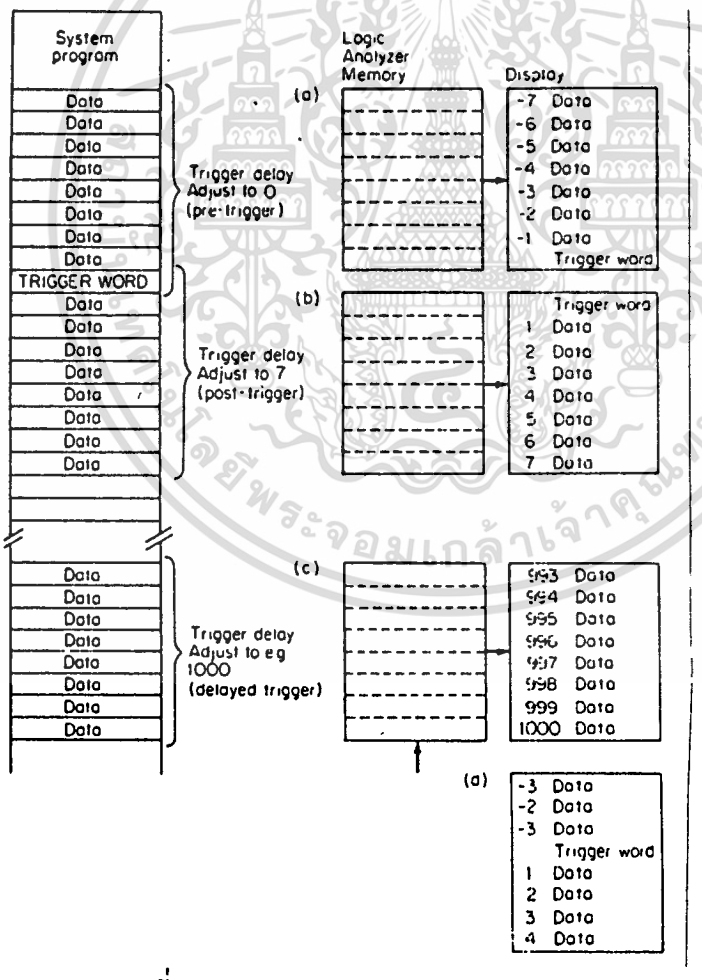
การทริกภายใน (Internal triggering) การทริกแบบนี้เกิดจากสัญญาณที่สร้างมาจากการเปรียบเทียบข้อมูลอินพุทกับทริกเกอร์เวิร์ด และการสุ่มข้อมูลจะหยุดเมื่อสัญญาณทั้งสองเหมือนกัน เราสามารถใช้ทริกเกอร์เวิร์ดหลายชุดทำงานต่อเนื่องกัน เพื่อให้ได้ข้อมูลที่ต้องการลักษณะอีกอย่างคือการใช้ทริกเกอร์ควอลิฟายเออร์มาควบคุมการทริกดังในภาพที่ 2-6 จะเห็นว่าถึงแม้ข้อมูลเข้าช่วงสัญญาณนาฬิกาลูกที่สองจะตรงกับทริกเกอร์เวิร์ดแต่ไม่มีการทริกเกิดขึ้น เพราะว่าทริกเกอร์ควอลิฟายเออร์ไม่ตรงกับที่กำหนด และการทริกเกิดขึ้นในสัญญาณนาฬิกาลูกที่สิบเพราะเงื่อนไขตรงทั้งสองอย่าง

การทริกภายนอก (External triggering) การทริกสามารถควบคุมจากภายนอกได้เช่นกัน โดยใช้สัญญาณทริกจากภายนอก

การหน่วงเวลาทริก (Trigger delay) การหน่วงเวลาการทริกทำให้เราสามารถเลือกจำนวนของข้อมูลที่อยู่ก่อนหรือหลังทริกเกอร์เวิร์ดได้ เราหน่วงเวลาการเกิดของสัญญาณทริกไปเป็นจำนวนครั้งของการสุ่มข้อมูลซึ่งเลือกได้โดยผู้ใช้ ขอยกตัวอย่างการใช้ทริกเกอร์ดีเลย์ดังภาพที่ 2-7

ถ้าไม่มีการหน่วงเวลาดังในภาพที่ 2-7ก ทันทีที่ข้อมูลเข้าและ ทริกเกอร์เวิร์ดเท่ากัน การสุ่มข้อมูลจะหยุดทันที และข้อมูลจะถูกนำออกแสดง ข้อมูลสุดท้าย คือทริกเกอร์เวิร์ด เราเรียกการทริกลักษณะนี้ว่า พรีทริกเกอร์ (pretrigger)

สมมติว่าหน่วยความจำมีความยาว 8 ข้อมูล(ภาพที่ 2-7ข) เลือกให้มีการหน่วงเวลาเท่ากับ 7 คล็อก เมื่อพบทริกเกอร์เวิร์ด ตัวนับ(counter) จะเริ่มนับถอยหลังทุกๆครั้งที่มีการสุ่มข้อมูลจนเท่ากับศูนย์ สัญญาณหยุดจะถูกส่งออกมา ซึ่งจะทำให้เราได้ข้อมูลหลังจากทริกเกอร์เวิร์ดเจ็ดข้อมูลนำไปแสดงผลแบบนี้เรียกว่า โพสต์ทริกเกอร์(post-trigger) ทำนองเดียวกันถ้าเราให้ดีเลย์เป็น 4 ก็จะได้ข้อมูลก่อนทริกเกอร์เวิร์ดสี่ข้อมูล และหลังทริกเกอร์เวิร์ดสามข้อมูล เรียกว่าเซ็นเตอร์ทริกเกอร์ (center trigger)(ภาพที่ 2-7ค)



ภาพที่ 2-7 การหน่วงเวลาทริก

การทำงานของภาคแสดงผล คือการนำข้อมูลที่เก็บอยู่ในหน่วยความจำมาจัดให้อยู่ในรูปแบบที่เหมาะสม แล้วนำไปแสดงออกทางจอภาพหรือพิมพ์ออกทางพริ้นเตอร์ รูปแบบของการแสดงผลแบ่งเป็นการแสดงสถานะ (State Display) และการแสดงเวลา (Timing Display) ในการแสดงสถานะนั้นจะอยู่ในรูปความสัมพันธ์ระหว่างสถานะทางลอจิกของแต่ละอินพุทตามลำดับของการสุ่มข้อมูล ซึ่งอาจจะแสดงในรูปของเลขฐานสอง (Binary Format) หรือเลขฐานสิบหก (Hexadecimal Format) (ภาพที่ 2-8ก) หรือ อยู่ในรูปแบบอื่นๆ รวมทั้งการทำดิสแอสเซมเบลอร์ (Disassembler) (ภาพที่ 2-8ข)ข้อมูล ด้วย

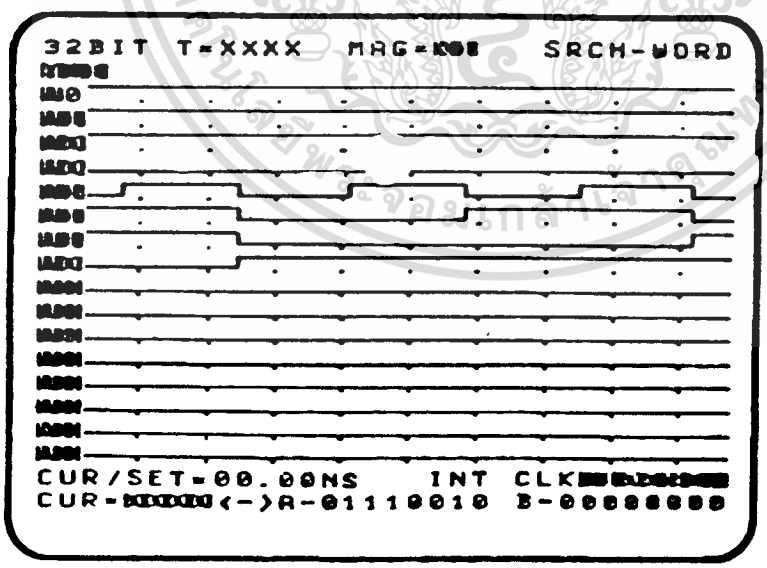
ส่วนการแสดงเวลานั้นจะนำข้อมูลมาแสดงเป็นแผนผังเวลา(Timing Diagram) ตามเวลาจริงที่ข้อมูลเกิดขึ้น (ภาพที่ 2-8ค) แบบเดียวกับออสซิลโลสโคปและเรายัง สามารถขยายขนาดของภาพที่แสดงได้ด้วย

48 BIT	T=XXXX	C=S+0000	MEM-SRC
CURS	BIN	OCT	HEX
0490S	00001111	017	OF
0491	00000000	000	00
0492	00110000	060	30
0493	00110001	061	31
0494	10010010	222	92
0495	10010010	222	92
0496	00010010	022	12
0497	00000000	000	00
0498	00000000	000	00

ภาพที่ 2-8ก การแสดงผลแบบเลขฐานสองและฐานสิบหก

```
48 BIT    T=XXXX    C=S+0000    MEM-SRC
ADDR      DATA      [MNEMONIC Z80]
0023      AF        XOR    A
0024      D381      OUT    (81H),A
0081      00        i/o-write
0026      C33300    JP     0033H
0033      DB81      IN     A,(81H)
0081      E2        i/o-read
0035      CB6F      BIT    5,A
0037      CAE500    JP     Z,00E5H
003A      C34400    JP     0044H
```

ภาพที่ 2-8ข การแสดงผลการทำดิสแอสเซมเบิ้ล



ภาพที่ 2-8ค การแสดงผลแบบแผนผังเวลา

รูปแบบของการแสดงผลแบบต่างๆที่กล่าวมาแล้วอาจจะแสดงออกทางเครื่องพิมพ์แทนจอภาพก็ได้ นอกจากนี้ถ้าเราต้องการเก็บข้อมูลไว้วิเคราะห์ภายหลังก็ทำได้โดยนำข้อมูลในหน่วยความจำไปเก็บในหน่วยความจำถาวร เช่น เทป, แผ่นจานแม่เหล็ก, ฯลฯ

บทที่ 3
การออกแบบฮาร์ดแวร์ของลอจิกอนาไลเซอร์

ในการออกแบบฮาร์ดแวร์นั้นมีสิ่งที่ต้องคำนึงถึงคือขีดความสามารถที่ต้องการ,ความ สะดวกในการใช้งาน, และความเชื่อถือได้ ซึ่งเราจะยึดถือเป็นหลักในการออกแบบโดย ตลอด

3.1 คุณสมบัติที่ต้องการ (Specification)

การกำหนดคุณสมบัติต่างๆยึดถือตามความสามารถที่เครื่องลอจิกอนาไลเซอร์ควรจะมี ตามที่ได้กล่าวถึงในบทที่ 2 และได้ปรับปรุงไปตามความเหมาะสมในการออกแบบจริง

คุณสมบัติที่กำหนดมีดังนี้

-INTERNAL CLOCK 500Hz-40MHz (1-2-5 sequence) , EXTERNAL CLOCK 20MHz maximum 1 channel with 3 channel Clock qualifier

-16 channel data input ,memory deep 2k bit /channel for20 MHz clock or 8 channel data input , memory 4k bit for 40 MHz clock

-INTERNAL ,EXTERNAL or MANUAL TRIG can be used

-16 bit TRIGGER WORD (0,1 or Don't care selectable)

-TRIGGER DELAY 0-4096 counts(samplilg clock)

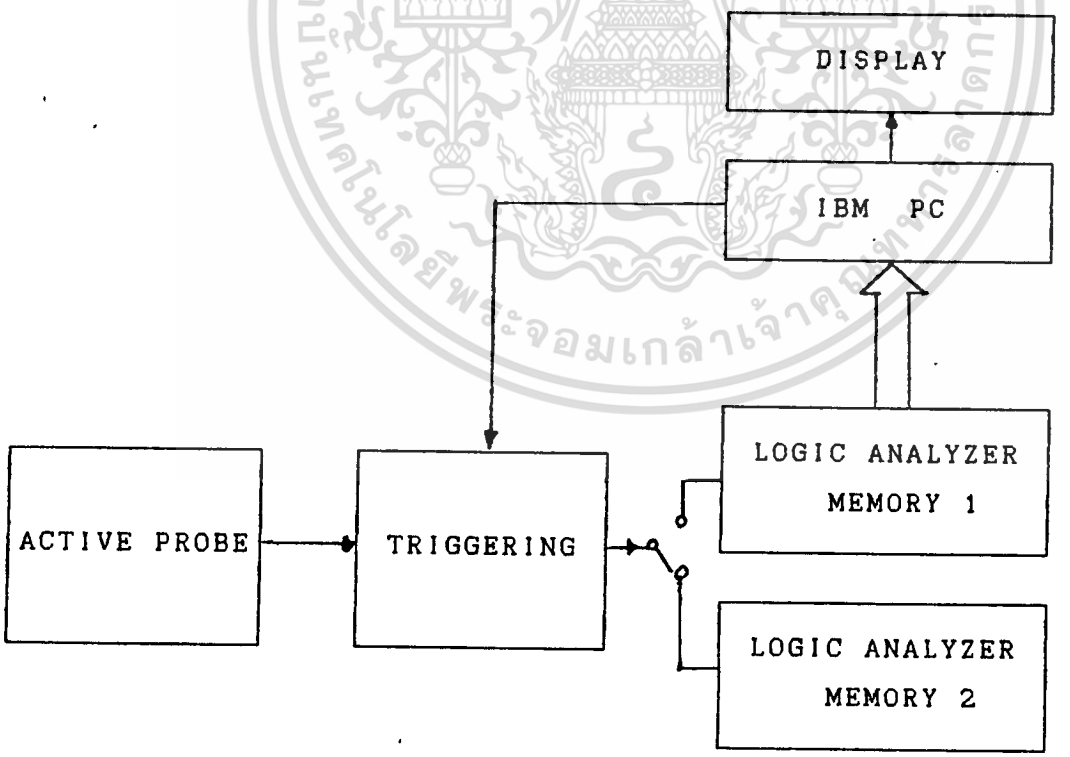
-THRESHOLD LEVEL -10V to +10V (0.1V step) and TTL LEVEL(1.4V)

-GLITCH DETECTOR supported

สำหรับการควบคุมการทำงานและการแสดงผลนั้นเลือกใช้วิธีการติดต่อผ่านทางเครื่อง IBM PC เพื่อให้เกิดความง่ายในการใช้งาน และลดความยุ่งยากในการออกแบบวงจรแสดง ผลดังนั้นลักษณะของเครื่องจะเป็นการ์ดเสียบบนสล็อตของเครื่อง IBM PC

3.2 บล็อกไดอะแกรมของเครื่องลอจิกอนาไลเซอร์

จากคุณสมบัติที่กำหนดนำมาเขียนเป็นบล็อกไดอะแกรมแสดงการทำงานได้ดังภาพที่ 3-1 การควบคุมการทำงานจะเป็นหน้าที่ของซอฟแวร์ผ่านทางสล็อต 8 บิทของเครื่อง IBM PC การทำงานจะเริ่มจากการเซ็ตเครื่องโดยการส่งข้อมูลที่จำเป็นมาที่พอร์ทต่างๆ รวมทั้งการส่งสัญญาณ START มาเพื่อเริ่มการสุ่มข้อมูล จากนั้นการสุ่มข้อมูลก็จะเริ่มขึ้น โดยสัญญาณนาฬิกาจะถูกนำไปใช้สร้างแอดเดรสของหน่วยความจำ และสร้างสัญญาณสุ่มข้อมูลตามลักษณะการทำงานที่ได้เลือกไว้การสุ่มข้อมูลจะหยุดเมื่อมีสัญญาณ TRIG หลังจากนั้นก็จะเป็นการย้ายข้อมูลจากหน่วยความจำของเครื่องไปยังหน่วยความจำของIBM PC เพื่อทำการแสดงผลและวิเคราะห์ข้อมูลต่อไป



ภาพที่ 3-1 บล็อกไดอะแกรมของเครื่องลอจิกอนาไลเซอร์

ส่วนที่สำคัญอีกส่วนหนึ่งคือหน่วยความจำ ซึ่งจะเห็นได้ว่าต้องใช้หน่วยความจำ 2 ชุด สลับกันทำงาน ทั้งนี้เนื่องมาจากการทำงานที่ความที่สัญญาณนาฬิกา 40 MHz นั้นถ้าใช้หน่วยความจำชุดเดียวจะต้องใช้หน่วยความจำที่มีเวลาเข้าถึง(access time) น้อยกว่า 25 ns ซึ่งหายาก จึงแก้ไขโดยใช้หน่วยความจำความเร็ว 50 ns สองชุดสลับกันทำงาน ซึ่งทำให้การทำงานแบบนี้ จำนวนอินพุทลดลงครึ่งหนึ่งแต่ขนาดของหน่วยความจำจะเพิ่มขึ้นเป็นสองเท่า

3.3 การออกแบบวงจร

สำหรับฮาร์ดแวร์แบ่งออกเป็น 7 ส่วนใหญ่ๆ ดังนี้

1) ส่วนเชื่อมต่อกับ IBM PC
2) ส่วนสร้างสัญญาณนาฬิกา
3) ส่วนสร้างแอดเดรสและสัญญาณควบคุมการสุ่มข้อมูล
4) ส่วนสร้างแรงดันเทรสโฮลด์
5) ส่วนรับข้อมูล
6) ส่วนสร้างสัญญาณทริก
7) หน่วยความจำ

3.3.1 ส่วนเชื่อมโยงกับ IBM PC

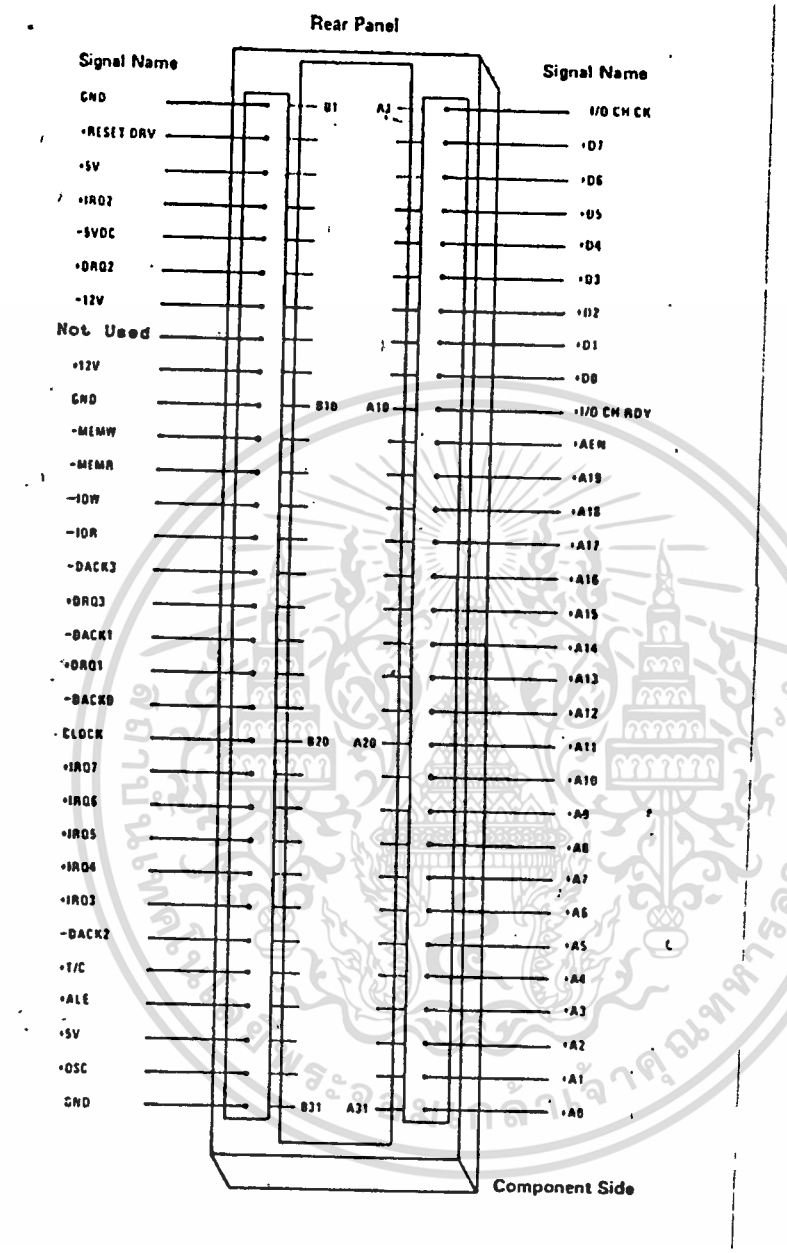
จากภาพที่ 3-2 แสดงถึงสัญญาณต่างๆบนสล็อตของเครื่อง IBM PC ซึ่งมีสัญญาณที่เรา ใช้งานคือ

1) D0-D7 คือสายข้อมูล (Data Bus)

2) A0-A9 คือสายแอดเดรสซึ่ง IBM ได้จัดเตรียมไว้สำหรับการติดต่อกับอุปกรณ์ภายนอกจำนวน 10 เส้น (ที่เหลือใช้ขยายหน่วยความจำ)

3) AEN (Address Enable) เป็นสัญญาณบ่งบอกว่ากำลังอยู่ในช่วงการทำ DMA อยู่ (AEN เป็น 1) ซึ่งอุปกรณ์อื่นๆจะใช้แอดเดรสบัสไม่ได้

4) IOR สัญญาณบอกสถานะการอ่านข้อมูลจากอุปกรณ์ภายนอก
5) IOW สัญญาณบอกสถานะการเขียนข้อมูลไปที่อุปกรณ์ภายนอก
6) GND,+12V,-12V,+5V และ -5V



ภาพที่ 3-2 สัญญาณต่างๆบนสล็อต IBM PC

Hex Range	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Register
0AX*	NMI Mask Register
0CX	Reserved
0EX	Reserved
200-20F	Game Control
210-217	Expansion Unit
220-24F	Reserved
278-27F	Reserved
2F0-2F7	Reserved
2F8-2FF	Asynchronous Communication (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C**	SDLC Communication
380-389**	Binary Synchronous Com(Secondary)
3A0-3A9	Binary synchronous Com(Primary)
3B0-3BF	IBM Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphic
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchronous Communication (Primary)

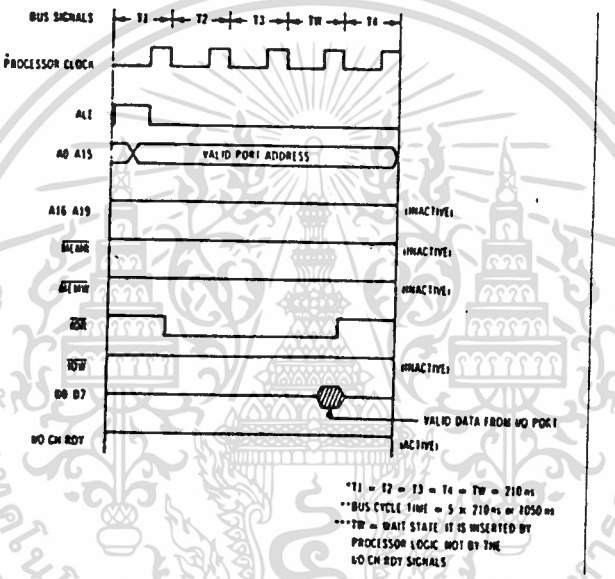
ภาพที่ 3-3 ตำแหน่งพอร์ทที่ถูกใช้โดย IBM PC

ในการเชื่อมโยงกับ IBM PC นี้จะทำผ่านทางพอร์ทอินพุทและเอาท์พุท ดังนั้นจึงต้องมีการดีโคดหมายเลขพอร์ท ซึ่งต้องไม่ให้ซ้ำของเดิมโดยใช้แอดเดรส A0-A9 มาทำการดีโคด โดยที่

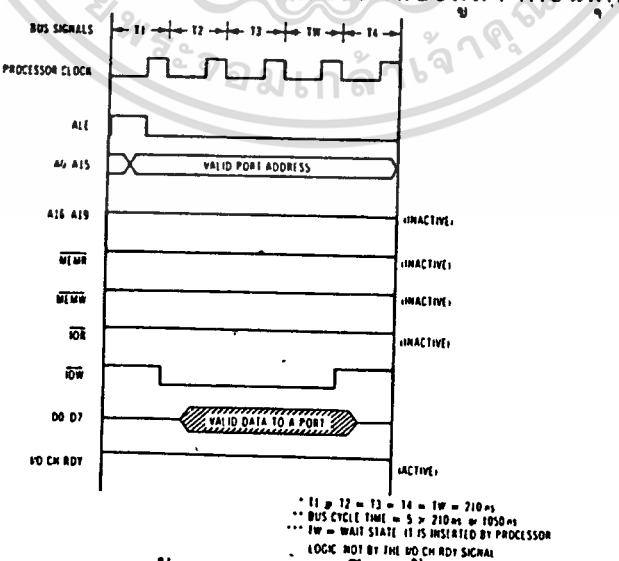
ถ้า A9 = 0 หมายถึงพอร์ทบนเมนบอร์ด

ถ้า A9 = 1 หมายถึงพอร์ทบน I/O การ์ด

นอกจากนี้ยังมีสัญญาณอื่นๆที่ต้องนำมาพิจารณาในการดีโคดตำแหน่งพอร์ทได้แก่ IOR, IOW และ AEN ซึ่งจะสามารถพิจารณาถึงความสัมพันธ์ระหว่างสัญญาณต่างๆในขบวนการอ่านและเขียน I/O พอร์ทได้จากแผนผังเวลาในภาพที่ 3-4



ภาพที่ 3-4 ก. แผนผังเวลาการอ่านข้อมูลจากอินพุทพอร์ท



ภาพที่ 3-4 ข. แผนผังเวลาการเขียนข้อมลลงเอาท์พุทพอร์ท

จากข้อมูลทั้งหมดนำมาสร้างเป็นวงจรสำหรับติดต่อกับ IBM PC ได้ดังภาพที่ 3-5 ซึ่ง ออกแบบการดีโคดแอดเดรสให้สามารถเปลี่ยนตำแหน่งได้โดยเซ็ตที่ดิพสวิตช์ให้ตรงกับกลุ่ม แอดเดรสที่ต้องการ 74LS688 จะเป็นตัวเปรียบเทียบแอดเดรสบัสว่าตรงกับค่าที่ตั้งไว้ ก็จะให้สัญญาณออกไปอีนาเบิ้ล 74LS138 เพื่อทำการดีโคดกลุ่มย่อยให้เป็นสัญญาณเลือกพอร์ทอีกที ทำให้ได้สัญญาณเลือกพอร์ท 8 เส้น (CS0-CS7) ในโครงงานนี้จะใช้อินพุทพอร์ทเพียง 4 พอร์ท และเอาท์พุทพอร์ท 11 พอร์ท แต่เนื่องจาก 8255 แต่ละตัวต้องใช้พอร์ท 1 พอร์ท สำหรับคำสั่งควบคุม จึงต้องใช้ 8255 ทั้งหมด 5 ตัว และใช้ SA0,SA1,IOR และIOW ต่อ ร่วมกับสัญญาณเลือกพอร์ท เพื่อควบคุมการเขียน-อ่านพอร์ท

ความหมายของสัญญาณในแต่ละพอร์ทมีดังนี้

อินพุทพอร์ท

-DA0..DA7 ดาต้าบัสของหน่วยความจำชุดที่ 1

-DB0..DB7 ดาต้าบัสของหน่วยความจำชุดที่ 2

-B0..B11 แอดเดรสของข้อมูลสุดท้ายที่เก็บในหน่วยความจำ

-FULL แสดงสถานะของการเก็บข้อมูลว่าเก็บเต็มหน่วยความจำหรือไม่

-STOP แสดงการหยุดสุ่มข้อมูล เมื่อมีการตรวจสอบว่าบิทนี้เป็น 1 จะทำการย้ายข้อมูลไปยังหน่วยความจำของ IBM PC

เอาท์พุทพอร์ท

-S0..S5 สัญญาณควบคุมการเลือกสัญญาณนาฬิกา

-S6 เลือกโหมดการรับอินพุทแบบ 16 บิท หรือ 8 บิท

-S7 ใช้ในโหมด 8 บิท สำหรับเลือกว่าจะรับอินพุทจากแชนเนล A หรือ B

-S8 ใช้ดิสเอเบิ้ลสัญญาณแอดเดรสของลอจิกอนาไลเซอร์ เพื่ออ้างแอดเดรสโดยตรงจาก IBM PC

-S9,S10 เลือกแหล่งสร้างสัญญาณทริก

-S11 เลือก Threshold voltage ของสัญญาณนาฬิกาและควอลิฟายเออร์

-S12 เลือก Threshold voltage ของสัญญาณข้อมูล

-TW0..TW15 Trigger word

-TI0..TI15 Trigger bit enable

-S/L เลือกโหมด Sampling หรือ Latch

-C0..C11 เวลาหน่วงของการทริก

-TH0..TH7 ระดับแรงดันอ้างอิง

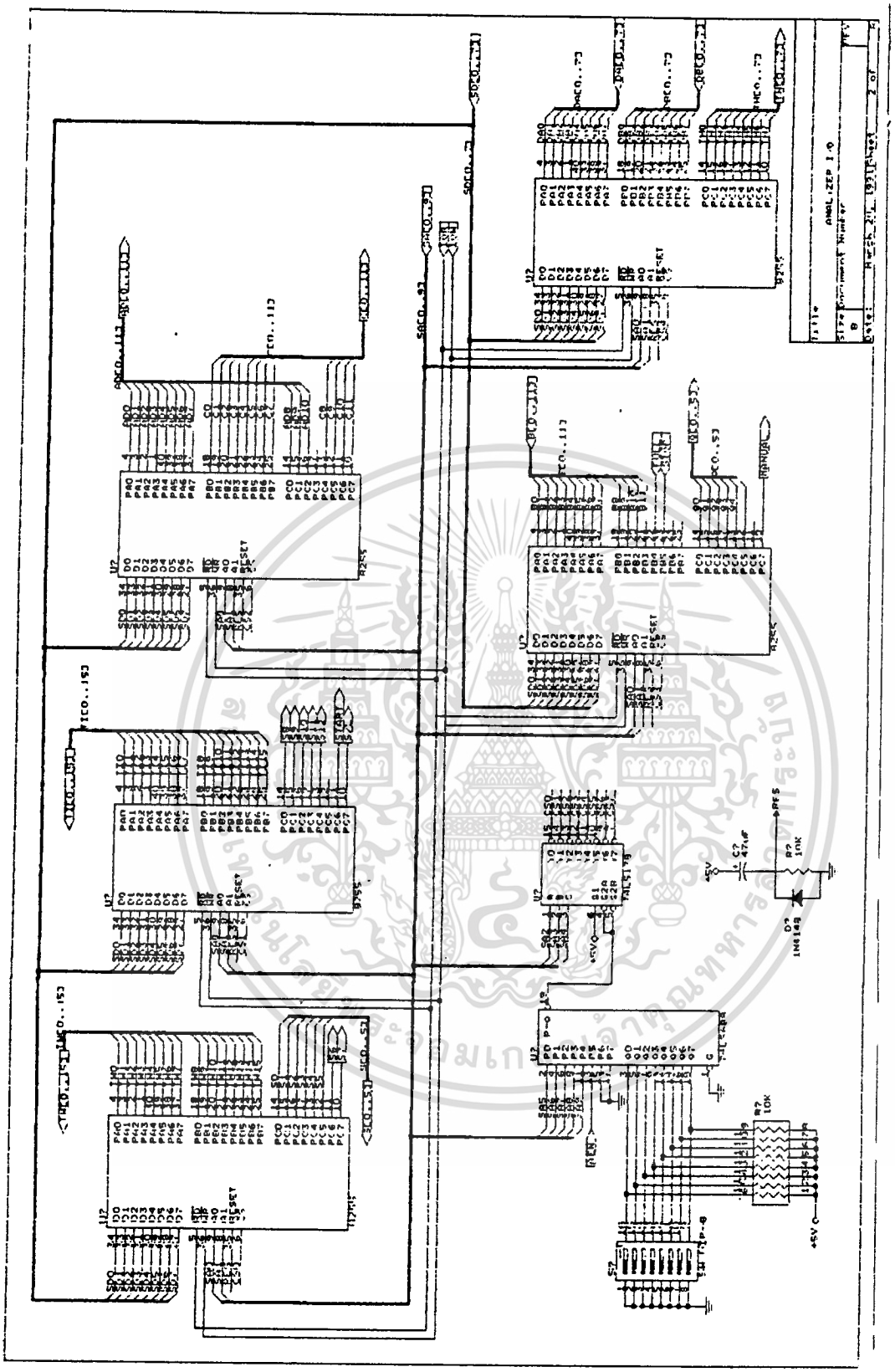
-Q0..Q5 ควบคุมคล็อกควอลิฟายเออร์

-MANUAL สัญญาณทริกที่ส่งมาจาก IBM PC

-AD0..AD10 แอดเดรสสำหรับการอ่านข้อมูลจากหน่วยความจำ

-START สัญญาณเริ่มต้นการสุ่มข้อมูล

รายละเอียดการทำงานของสัญญาณต่างๆจะได้กล่าวถึงต่อไป



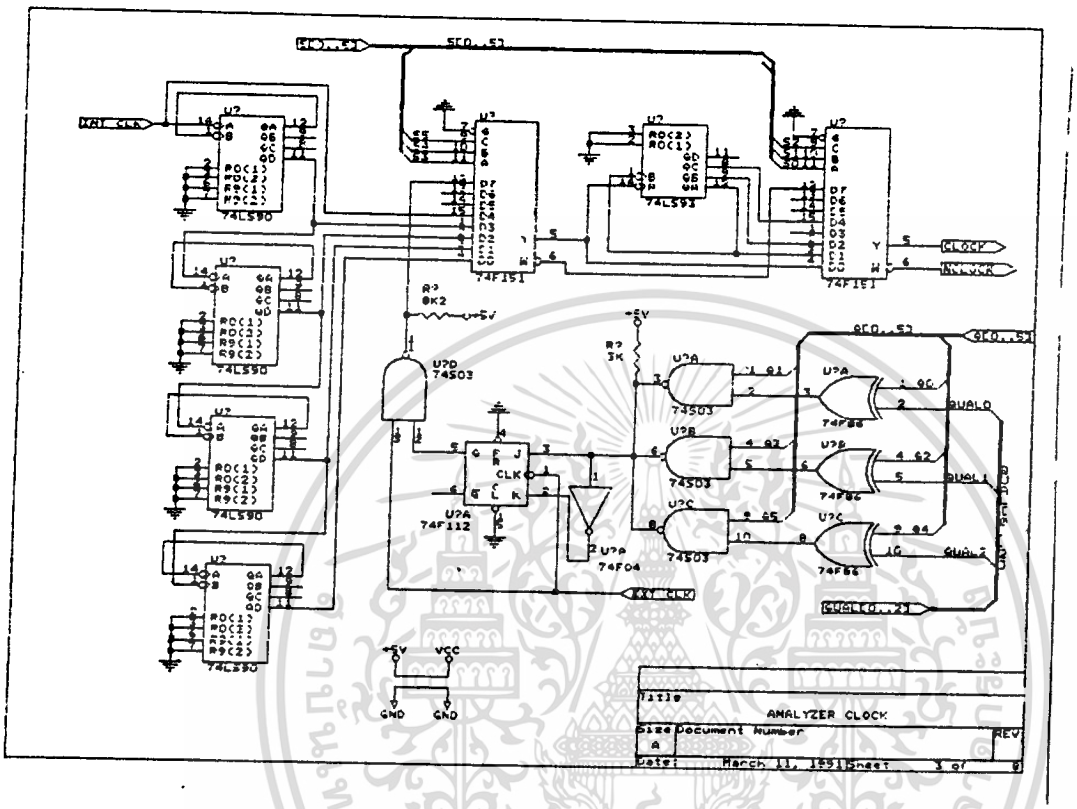
ภาพที่ 3-5 วงจรส่วนติดต่อกับ IBM PC

3.3.2 ส่วนสร้างสัญญาณนาฬิกา (CLOCK SELECTOR)

จากภาพที่ 3-6 เป็นวงจรเลือกสัญญาณนาฬิกา ซึ่งสามารถเลือกได้ทั้งสัญญาณนาฬิกาภายใน(INTERNAL CLOCK) และสัญญาณนาฬิกาภายนอก (EXTERNAL CLOCK) ซึ่งสัญญาณนาฬิกาภายในสามารถเลือกความถี่ได้ตั้งแต่ 500 Hz ถึง 40 MHz โดยสามารถเลือกความถี่ได้ในลักษณะ 5-2-1 วงจรที่ใช้สร้างความถี่เป็นวงจร OSCILLATOR ขนาด40 MHz โดยใช้ไอซีเบอร์ 74LS90 เป็นวงจรหาร 10 และใช้ไอซีเบอร์ 74LS93เป็นวงจรหาร 12 และใช้ไอซีเบอร์ 74F151 เป็นตัวเลือกความถี่ ซึ่งสัญญาณนาฬิกาที่สร้างได้จะสามารถเลือกความถี่ได้ตามตาราง

S5	S4	S3	S2	S1	S0	ความถี่สัญญาณนาฬิกา
1	0	0	0	0	0	40 เมกกะเฮิร์ท
1	0	0	0	0	1	20 "
1	0	0	0	1	0	10 "
1	0	0	1	0	0	5 "
0	1	1	0	0	1	2 "
0	1	1	0	1	0	1 "
0	1	1	1	0	0	500 กิโลเฮิร์ท
0	1	0	0	0	1	200 "
0	1	0	0	1	0	100 "
0	1	0	1	0	0	50 "
0	0	1	0	0	1	20 "
0	0	1	0	1	0	10 "
0	0	1	1	0	0	5 "
0	0	0	0	0	1	2 "
0	0	0	0	1	0	1 "
0	0	0	1	0	0	500 เฮิร์ท
1	1	1	0	0	0	-EXTERNAL CLOCK
1	1	1	1	1	1	+EXTERNAL CLOCK

ตารางที่ 3-1 แสดงความถี่ของสัญญาณนาฬิกา



ภาพที่ 3-6 แสดงวงจรเลือกความถี่สัญญาณนาฬิกา

ในส่วนของสัญญาณนาฬิกาภายนอก (EXTERNAL CLOCK) จะมีวงจรคล็อกควอลิฟายเออร์ ซึ่งใช้ในการเลือกสุ่มสัญญาณเป็นบางช่วงของสัญญาณตามควอลิฟายเออร์ที่ได้ตั้งไว้ โดยใช้สัญญาณนาฬิกาภายนอกในการสุ่มข้อมูล ดังนี้

	LOGIC	STATE
Q1,Q3,Q5	0	DON'T CARE
	1	H,L
Q0,Q2,Q4	0	L
	1	H

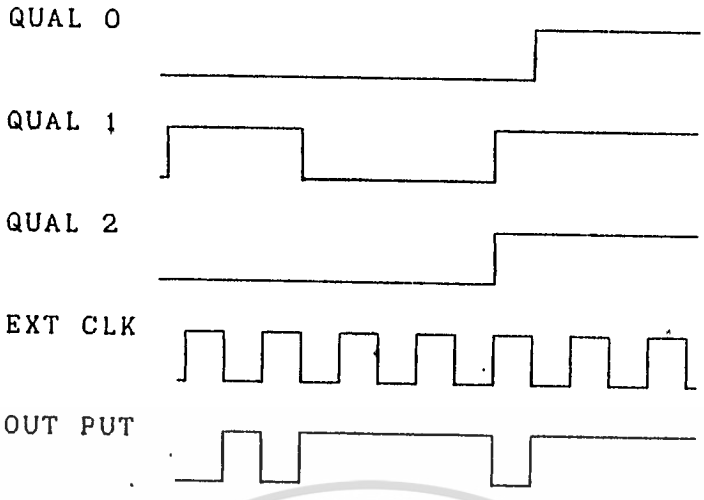
ตารางที่ 3-2 แสดงการเลือกใช้ คล็อกควอลิฟายเออร์

ซึ่งในการใช้คล็อกควอลิฟายเออร์นั้นจะมีลักษณะการทำงาน 3 สถานะ คือ X(don't care),H(High),L(Low) และจะมีควอลิฟายเออร์อยู่ 3 channels จะเห็นว่า ถ้า Q1 หรือ Q3 หรือ Q5 มีสถานะเป็น LOW ควอลิฟายเออร์นั้นจะมีสภาพเป็น don't care ถ้า Q1 หรือ Q3 หรือ Q5 มีสถานะเป็น HIGH และ Q0 หรือ Q2 หรือ Q4 มีสถานะเป็น LOW ควอลิฟายเออร์นั้นจะทำงานที่สถานะ LOW แต่ถ้า Q0,Q2,Q4 มีสถานะเป็น HIGH ควอลิฟายเออร์นั้นจะทำ งานที่สถานะ HIGH

จากวงจรจะเห็นว่าใช้ EXCLUSIVE OR เป็น GATE ในการเลือกสถานะของควอลิฟายเออร์ และใช้ NAND GATE แบบคอลเล็คเตอร์เปิดเป็นตัวรวมสัญญาณเข้าไปยัง D ฟลิป ฟลอป ซึ่งจะทำงานตามสัญญาณนาฬิกาภายนอก ที่ใช้เป็น CLOCK ให้กับ D ฟลิป ฟลอป สัญญาณที่ออกจะไปเข้า NAND GATE พร้อมกับสัญญาณนาฬิกาภายนอกได้เป็นสัญญาณนาฬิกาเข้าไปยังไอซีเบอร์ 74LS151เพื่อทำการเลือกความถี่

ตัวอย่างการทำงานของ คล็อกควอลิฟายเออร์

ถ้าให้ ควอลิฟายเออร์ 0 ทำงานที่สถานะ L
ควอลิฟายเออร์ 1 ทำงานที่สถานะ H
ควอลิฟายเออร์ 2 ทำงานที่สถานะ X



ภาพที่ 3-7 ตัวอย่างการทำงานของ คล็อกควอลิฟายเออร์

3.3.3 ส่วนสร้างแอดเดรสและสัญญาณควบคุมการสุ่มข้อมูล

วงจรในส่วนนี้ประกอบด้วยวงจรที่สำคัญ 2 ส่วนคือ

1. วงจรหน่วงเวลาทริก
2. วงจรสร้างสัญญาณแอดเดรส

1. วงจรหน่วงเวลาทริก

ประกอบด้วย

1) ไอซีเบอร์ 74F161 3 ตัว ใช้เป็นวงจรนับค่าเวลาหน่วง

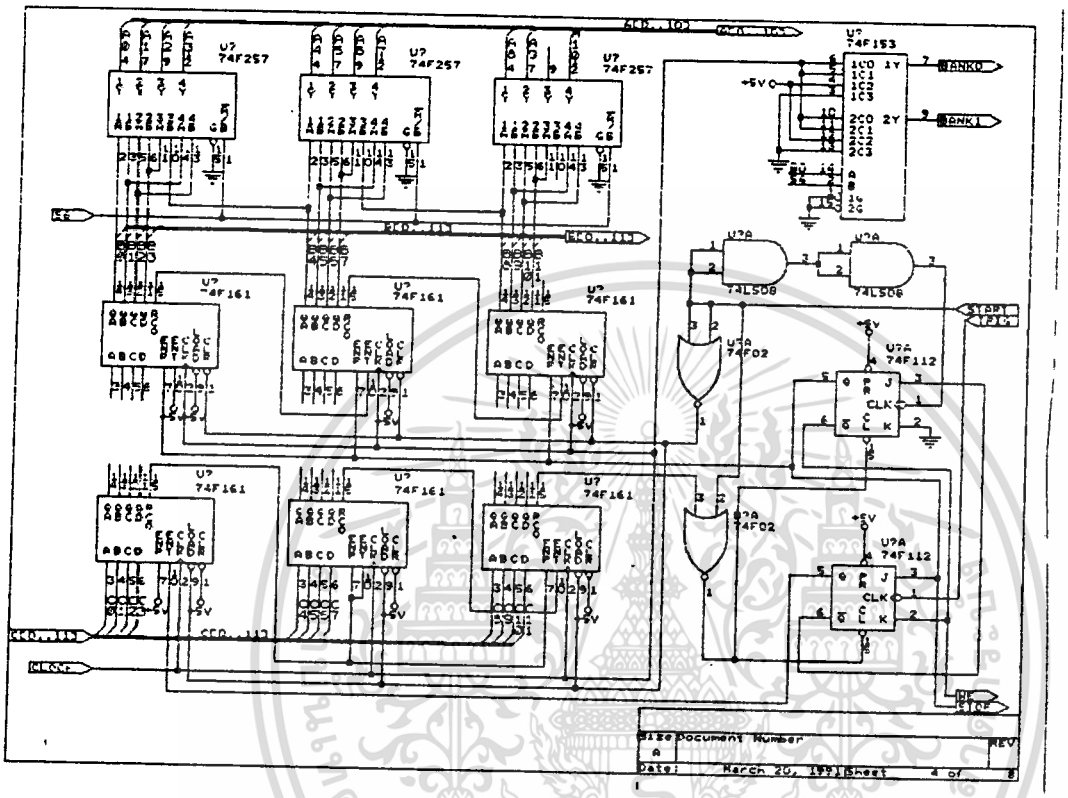
2) ไอซีเบอร์ 74F151 1 ตัว ใช้เป็นวงจรสร้างสัญญาณเคลียร์ (CLEAR)

จากภาพที่ 3-8 C0 ถึง C11 เป็นอินพุทของไอซีเบอร์ 74F161 ทั้ง 3 ตัวซึ่งจะทำการเซทค่าเวลาหน่วง ซึ่งมีขั้นตอนการทำงานดังนี้

1. สัญญาณ START ที่มาจาก PORT จะมีสถานะเป็น HIGH ทำให้ขา LOAD ของไอซีเบอร์ 74F161 เป็น LOW ทำให้ข้อมูลค่าเวลาหน่วงถูกส่งเข้าไปยัง 74F161 และ D ฟลิป-ฟลอปทั้งสองตัวถูกเคลียร์

2. เอ้าท์พุท(Q) ของ D type ฟลิป ฟลอป ทั้ง 2 ตัวมีสถานะเป็น LOW และ Q เป็น HIGH ดังนั้นที่อินพุทของ D type ฟลิป-ฟลอปตัวแรกจึงมีสถานะเป็น HIGH และ สัญญาณ WE เป็น HIGH

4. เมื่อเอ้าท์พุทของ Dฟลิป-ฟลอป เป็น LOW ทำให้ขา ENTของ 74F161 เป็น LOW 74F161 จึงยังไม่ทำงาน



ภาพที่ 3-8 แสดงวงจรสร้างสัญญาณแอดเดรส

5. เมื่อสัญญาณ START เป็น LOW ทำให้ขา CLK ของ D ฟลิป - ฟลอปเปลี่ยนจากHIGH เป็น LOW ทำให้เอ้าท์พุทของ D ฟลิป-ฟลอป ตัวแรกเป็น HIGH และสัญญาณ WE เป็น LOW วงจรเริ่มทำการส่มข้อมูล

6. เมื่อสัญญาณ TRIG เปลี่ยนสถานะจาก HIGH เป็น LOW ทำให้เกิดการทริกขึ้นที่ขา CLK ของ D ฟลิป-ฟลอปตัวที่ 2 ขา ENT ของไอซีเบอร์ 74F161 เป็น HIGH จึงเริ่มทำการนับ

7. เมื่อทำการนับจนถึงค่า OFFF จะทำให้สัญญาณที่ขา RCO ของทั้ง 3 ตัวเป็น HIGH จึงทำให้เกิดสัญญาณเคลียร์ D Flip- Flop ทั้ง 2 ตัว และขา ENT ของไอซีเบอร์ 74F161 เป็น LOW วงจรจึงหยุดทำงาน

2. วงจรสร้างสัญญาณ แอดเดรส

ประกอบด้วย 1)ไอซีเบอร์ 74F161 3 ตัวเป็นวงจรนับ

2)ไอซีเบอร์ 74F257 3 ตัวเป็นวงจรเลือกสัญญาณแอดเดรส

ขั้นตอนการทำงาน

1.เมื่อสัญญาณ START มีสถานะเป็น LOW ทำให้เอ้าท์พุทของไอซีเบอร์74F151 เป็น LOW ส่งผลให้ไอซีเบอร์ 74F161 ถูกเคลียร์ และสัญญาณที่ขา ENP เป็นLOW ทำให้ ยังไม่มีการนับ

2.เมื่อสัญญาณ START เป็น HIGH เอ้าท์พุทของไอซีเบอร์ 74F151 เป็น HIGH ทำให้ขา CLK ของไอซีเบอร์ 74F161 เป็น HIGH ขา ENP จึงมีสถานะเป็นHIGH วงจรนับจึงเริ่มทำการนับ

3.เมื่อสัญญาณ TRIG เปลี่ยนสถานะจาก LOW เป็น HIGH จะทำให้วงจรนับในส่วนของวงจรนับค่าเวลาหน่วงทำงาน เมื่อวงจรนับค่าเวลาหน่วงนับจนถึงค่า OFFF จะทำให้เอ้าท์พุทของไอซีเบอร์ 74F151 มีสถานะเป็น LOW วงจรนับค่าเวลาหน่วงจึงหยุดนับ และทำให้ขา ENP ของไอซีเบอร์ 74F161 ในส่วนของวงจรสร้างสัญญาณแอดเดรสมีสถานะเป็น LOW วงจรนับจึงหยุดนับก็จะทำให้ได้ค่าแอดเดรสตามที่เราตั้งเวลาหน่วงไว้

3.3.4 ส่วนสร้างแรงดันเทรสโฮลด์

แรงดันเทรสโฮลด์แบ่งเป็น 2 ชุดคือแรงดันคงที่ 1.4 V สำหรับไอซีตระกูลTTL และ แรงดันปรับค่าได้ตั้งแต่ -10V ถึง +10V สำหรับไอซีตระกูลอื่น และใช้ทดสอบความผิดพลาด เนื่องจากแรงดันผิดระดับ

การปรับค่าแรงดันเทรสโฮลด์นั้น ต้องสามารถทำได้ด้วยซอฟแวร์ ดังนั้นจึงต้องใช้วิธี การของ D/A มาช่วย โดยการใช้ข้อมูลจากเอาท์พุทพอร์ท (TH0..TH7) มาแปลงเป็น สัญญาณอนาล็อกด้วยไอซี DAC0800 (ภาพที่ 3-9) เอาท์พุทที่ได้จะเป็นกระแสต้องนำไปเข้าออปแอมป์เพื่อแปลงเป็นแรงดันและขยายให้ได้ขนาดที่ต้องการ

DAC0800 ต้องการแรงดันอ้างอิงขนาด 10V ซึ่งสร้างมาจากไอซีเร็กกูเลเตอร์ LM317 จะได้กระแสเอาท์พุท ตามสมการ

$$I_{fs} = V_{ref}/R_{ref} * 255/256$$

โดยที่

$$I_{fs} = I_{out} + I_{out}$$

$$I_{out} = \frac{V_{ref}}{R_{ref}} * \left(\frac{B_1}{2} + \frac{B_2}{4} + \ldots + \frac{B_8}{256} \right)$$

เมื่อนำ I_{out} และ I_{out} มาต่อร่วมกับออปแอมป์เพื่อ

แปลงผลต่างของกระแสให้เป็นแรงดัน จะได้สมการสำหรับแรงดันเอ้าท์พุทคือ

$$V_{out} = (I_{out} - I_{out}) * R_{load}$$
$$= (2I_{out} - I_{fs}) * R_{load}$$
$$= \frac{V_{ref}}{R_{ref}} * R_{load} * \left[2\left[\frac{B_1}{2} + \frac{B_2}{4} + \ldots + \frac{B_8}{256}\right] - \frac{255}{256}\right]$$

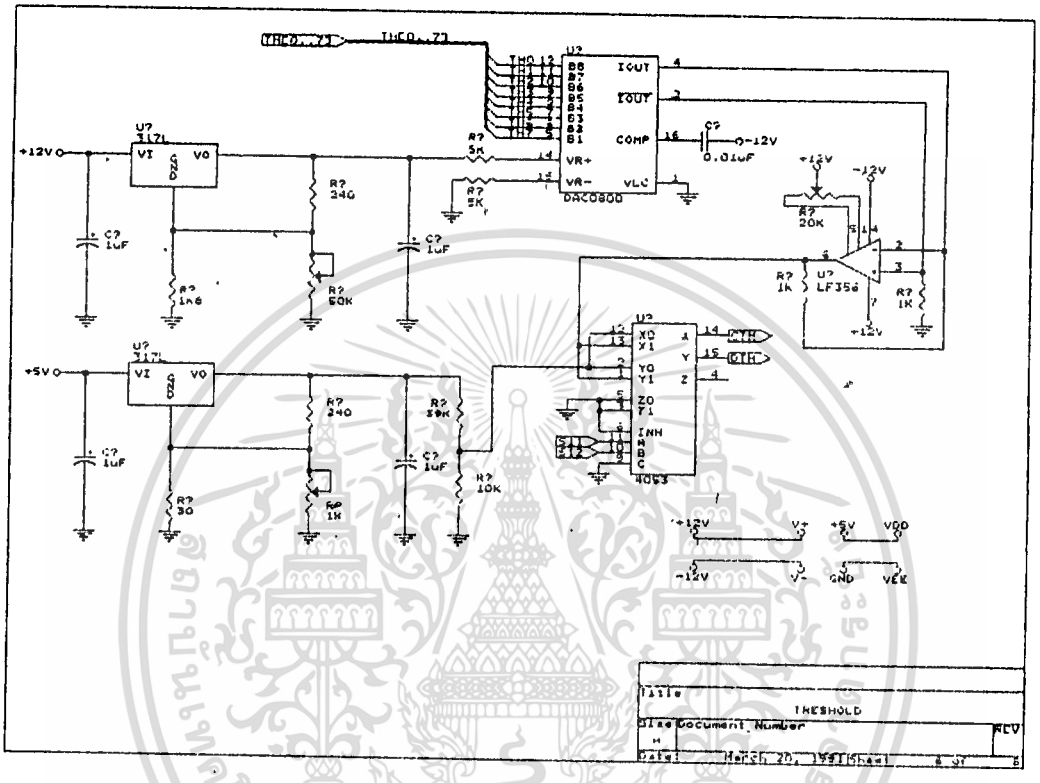
จากสมการโดยการใช้ V_{ref} เท่ากับ 10 V,Rref เท่ากับ 5 K จะได้I_{fs} ประมาณ 2 mA(โดยการปรับ V_{ref} ให้สูงกว่า 10 V เล็กน้อย จะได้ I_{fs} เท่ากับ 2 mA) ดังนั้นเพื่อให้ได้ V_{out} ตั้งแต่ -10 V ถึง +10 V ต้องใช้ R_{load}ประมาณ 5 K แต่เนื่อง จากสัญญาณอินพุทจะถูกลดทอนลงไป 5 เท่าที่ probe ดังนั้นแรงดันอ้างอิงที่จะให้กับตัวเปรียบเทียบแรงดันก็จะต้องลดลงเหลือ -2 V ถึง +2 V ดังนั้นต้องใช้ R $_{load}$ เท่ากับ 1 K เนื่องจากความไม่เท่ากันของค่าความต้านทานและแรงดันออฟเซ็ตของออปแอมป์ ทำให้ V_{out} มีค่าไม่สมมาตร จึงต้องมีการแก้ไขโดยการต่อตัวต้านทานปรับค่าระหว่างขา 1 และ ขา 5 ของออปแอมป์เพื่อปรับเอาท์พุทให้สมมาตร

ส่วนที่เป็นแรงดันคงที่ 1.4 V สร้างโดยใช้วิธีการแบบเดียวกับการสร้างแรงดัน 10 V ซึ่งแรงดันนี้จะต้องถูกลดทอนลงไป 5 เท่าก่อนเข้าตัวเปรียบเทียบแรงดันเช่นเดียวกัน และ แรงดันเทรสโฮลด์ทั้งสองแบบนี้ จะถูกนำไปใช้ในส่วนรับข้อมูล เพื่อเปรียบเทียบกับสัญญา น อินพุท,สัญญาณนาฬิกาภายนอก และสัญญาณทริกภายนอก โดยใช้ S11,S12เป็นตัวเลือกดังนี้

S11,S12 = 1 เลือกใช้แรงดันปรับค่าได้

S11,S12 = 0 เลือกใช้แรงดันคงที่

ทั้งนี้ S11 ใช้เลือกแรงดันเทรสโฮลด์สำหรับสัญญาณนาฬิกาและสัญญาณทริก S12 ใช้สำหรับสัญญาณข้อมูล โดยที่ S11,S12 ได้มาจาก 8255 นำไปควบคุมการปิด/เปิด อนาล็อกสวิทช์



ภาพที่ 3-9 วงจรสร้างแรงดันเทรสโฮลด์

3.3.5 ส่วนรับข้อมูล

ส่วนนี้มีหน้าที่ในการแปลงระดับสัญญาณจากภายนอก ให้อยู่ในระดับ TTL ซึ่งเป็นระดับที่ใช้งานในเครื่องลอจิกอนาไลเซอร์โดยใช้หลักการเปรียบเทียบสัญญาณกับแรงดันเทรสโฮลด์ที่ตั้งไว้ด้วยคอมพาราเตอร์ความเร็วสูง โดยแบ่งออกเป็นสองชุดคือสำหรับสัญญาณนาฬิกา และสำหรับข้อมูลอินพุท

สำหรับตัวเปรียบเทียบแรงดันใช้อุปกรณ์ประเภท Line Reciever (ภาพที่ 3-10) เนื่องจากมีข้อดีคือมีความเร็วสูง และให้เอาท์พุทเป็น TTL นอกจากนี้ยังทำงานแบบชมิทท์ทริกเกอร์(Schmitt Trigger) ทำให้ป้องกันความผิดพลาดในการตรวจจับสัญญาณซึ่งเกิดขึ้นจากสัญญาณรบกวน

สำหรับอุปกรณ์ที่ใช้ จะรับสัญญาณแบบคอมมอนโหมด(common mode)

ได้ประมาณไม่ เกิน 12 V แต่ในการออกแบบต้องการให้รับอินพุทสูงสุดได้ 50 V จึงต้องต่อวงจรลดทอนสัญญาณก่อน โดยใช้วงจรแบ่งแรงดันธรรมดา มีอัตราการลดทอน 5 เท่า และที่วงจรแบ่งแรงดันนี้ได้ต่อตัวเก็บประจุคร่อมตัวต้านทาน 39 K เพื่อชดเชยผลของตัวเก็บประจุแอบแฝง (stray capacitance) ซึ่งจะทำให้เกิดปรากฏการณ์ของวงจรกรองความถี่ต่ำผ่าน (Low- pass Filter) สำหรับค่าของตัวเก็บประจุที่ใช้ประมาณ 30-50 pF โดยได้มาจาก การทดลอง

ผลที่เกิดจากวงจรแบ่งแรงดันอีกอย่างหนึ่งคือทำให้ความต้านทานขาเข้ารวม (Total input resistor)มีค่าสูงขึ้นมากจากเดิมมีค่าประมาณ12 K สำหรับ 75175 และ 4 K สำหรับ 75107 มาเป็นมากกว่า 40 K

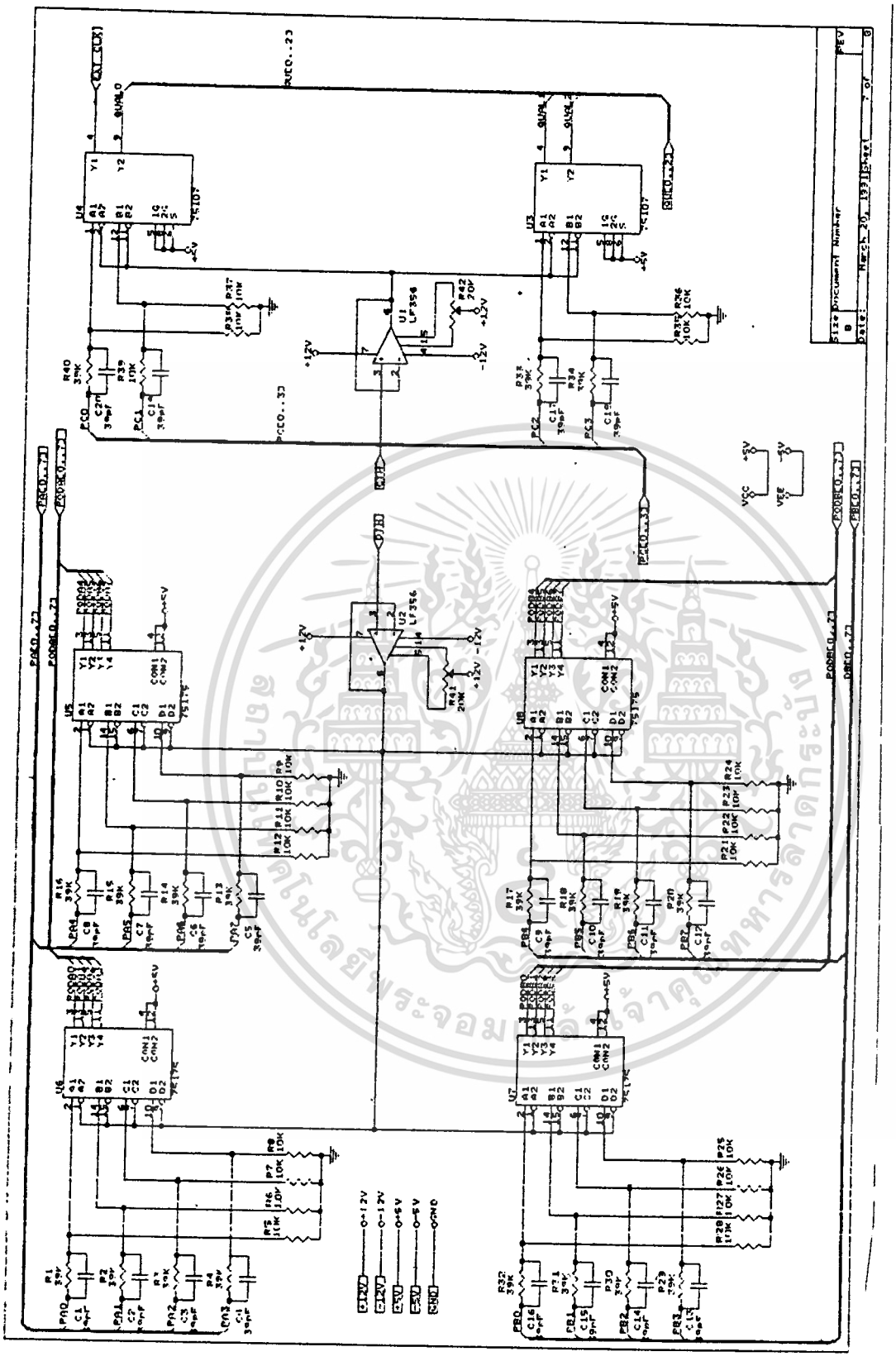
3.3.6 ส่วนสร้างสัญญาณทริก

สัญญาณทริกสามารถเลือกได้ 4 แบบ ด้วย S9,S10 ดังนี้

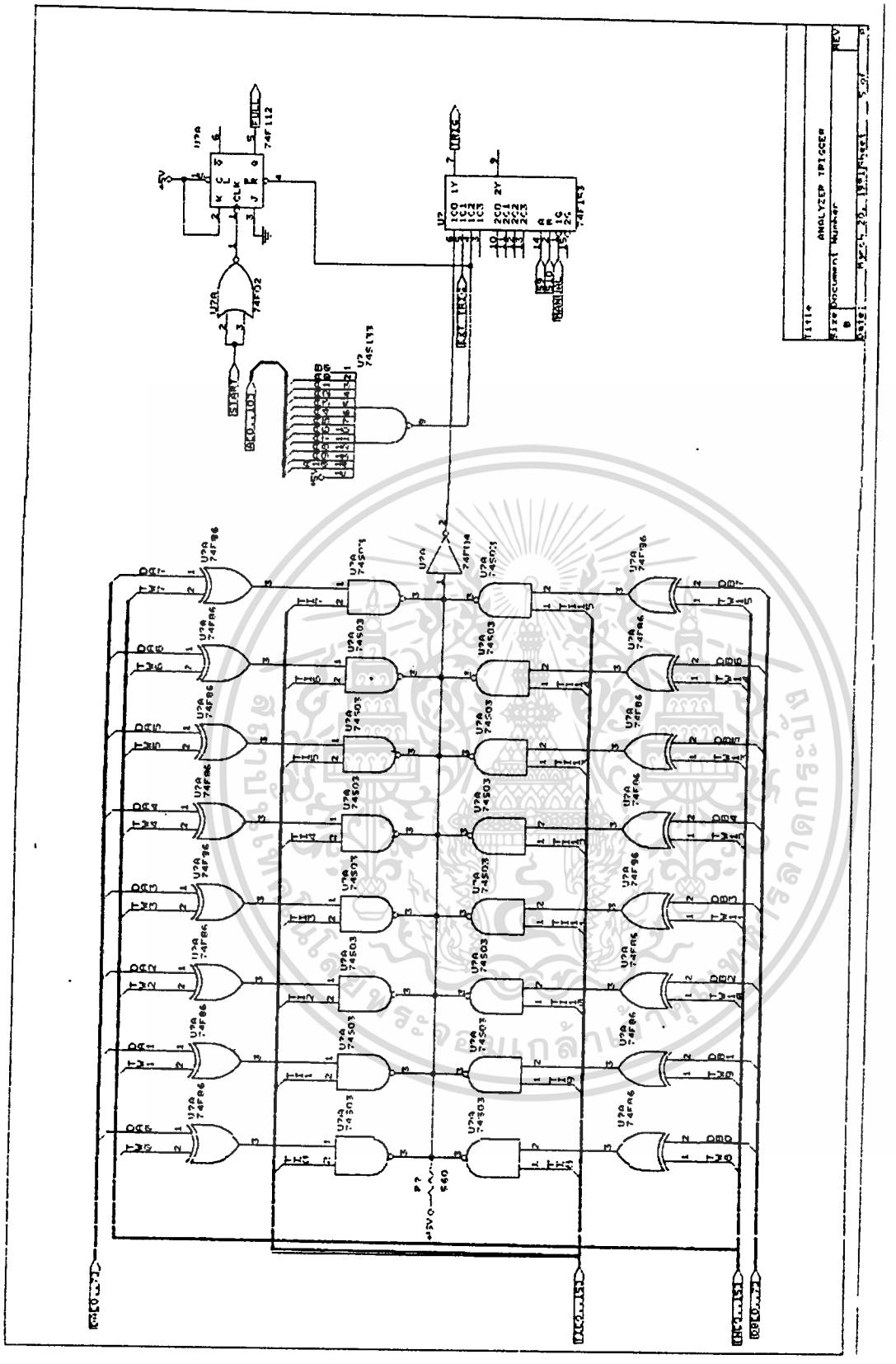
S10	S9	แหล่งสัญญาณทริก
0	0	INTERNAL
0	1	EXTERNAL
1	0	NO TRIG
1	1	Don't care

สัญญาณทริกภายใน (INTENAL) เกิดจากการเปรียบเทียบข้อมูลอินพุทกับทริกเกอร์เวิร์ด (TW0..TW15) แบบบิทต่อบิท ถ้าทุกบิทเหมือนกันจะทำให้ได้สัญญาณทริกเป็น 0 ในกรณีที่ไม่ต้องการเปรียบเทียบทุกบิทสามารถทำให้เป็น don't care ได้โดยการเซ็ต TINH สำหรับบิทนั้นให้เป็น 0

สัญญาณทริกภายนอกรับมาจากวงจรภายนอก ส่วนสัญญาณ MANUAL นั้นมาจากเอาท์พุทพอร์ท ในกรณีที่เลือก NO TRIG นั้นการสุ่มข้อมูลจะต้องหยุดเมื่อข้อมูลเต็มหน่วยความจำ จึงต้องนำสัญญาณแอดเดรสมาแนนด์กันเป็นสัญญาณทริก นอกจากนี้เมื่อมีการหยุดสุ่มข้อมูลเกิดขึ้นเรามีความจำเป็นต้องรู้ว่าเกิดการหยุดเมื่อข้อมูลเก็บเลยขนาดของหน่วยความจำหรือไม่ จึงต้องสร้าง



ภาพที่ 3-10 วงจรเปรียบเทียบสัญญาณ

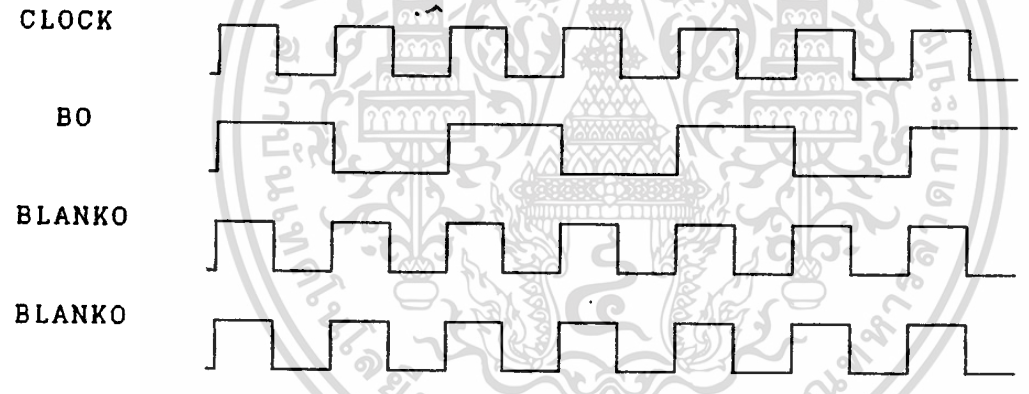


ภาพที่ 3-11 วงจรสร้างสัญญาณทริก

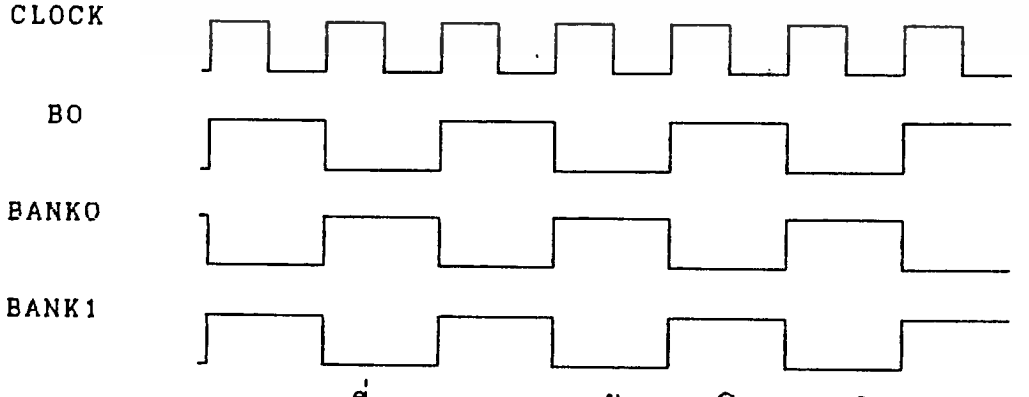
สัญญาณ FULL ซึ่งเมื่อสัญญาณนี้เป็น 1 แสดงว่าหน่วยความจำเต็ม เมื่อมีการเริ่มสุ่มข้อมูลใหม่สัญญาณ START จะถูกใช้เคลียร์สัญญาณนี้ให้เป็น 0

3.3.7 หน่วยความจำ

หน่วยความจำใช้ สเตติกแรมชนิดความเร็วสูงเบอร์ UM6116-45 มีช่วงเวลาเข้าถึง 45 ns สองชุดเรียกว่า BANK0,BANK1 ทำให้รับข้อมูลได้ 16 ช่อง 2K บิทต่อช่อง ซึ่งทำให้ใช้สัญญาณนาฬิกาได้สูงสุด 20 MHz ถ้าต้องการให้ใช้ได้กับสัญญาณนาฬิกาสูงสุด 40 MHz ต้องใช้การมัลติเพล็กซ์แรม โดยใช้สัญญาณ BANK0,BANK1 เป็นตัวควบคุมการแลตช์แอดเดรสและข้อมูลแผนผังเวลาของการทำงานทั้งสองแบบจะเป็นดังภาพที่ 3-12 ในการทำงานแบบ 16 บิท สัญญาณ BANK0 และ BANK1 จะเหมือนกับสัญญาณนาฬิกา และจะแอคทีฟที่ขอบขาขึ้น ส่วนในโหมด 8 บิท BANK0,BANK1 จะแอคทีฟสลับกัน โดยที่ BANK0 จะทำงานที่แอดเดรสคู่ส่วนBANK1จะทำงานที่แอดเดรสคี่



ภาพที่ 3-12 ก) แผนผังเวลาโหมด 16 บิท



ภาพที่ 3-12 ข) แผนผังเวลาโหมด 8 บิท

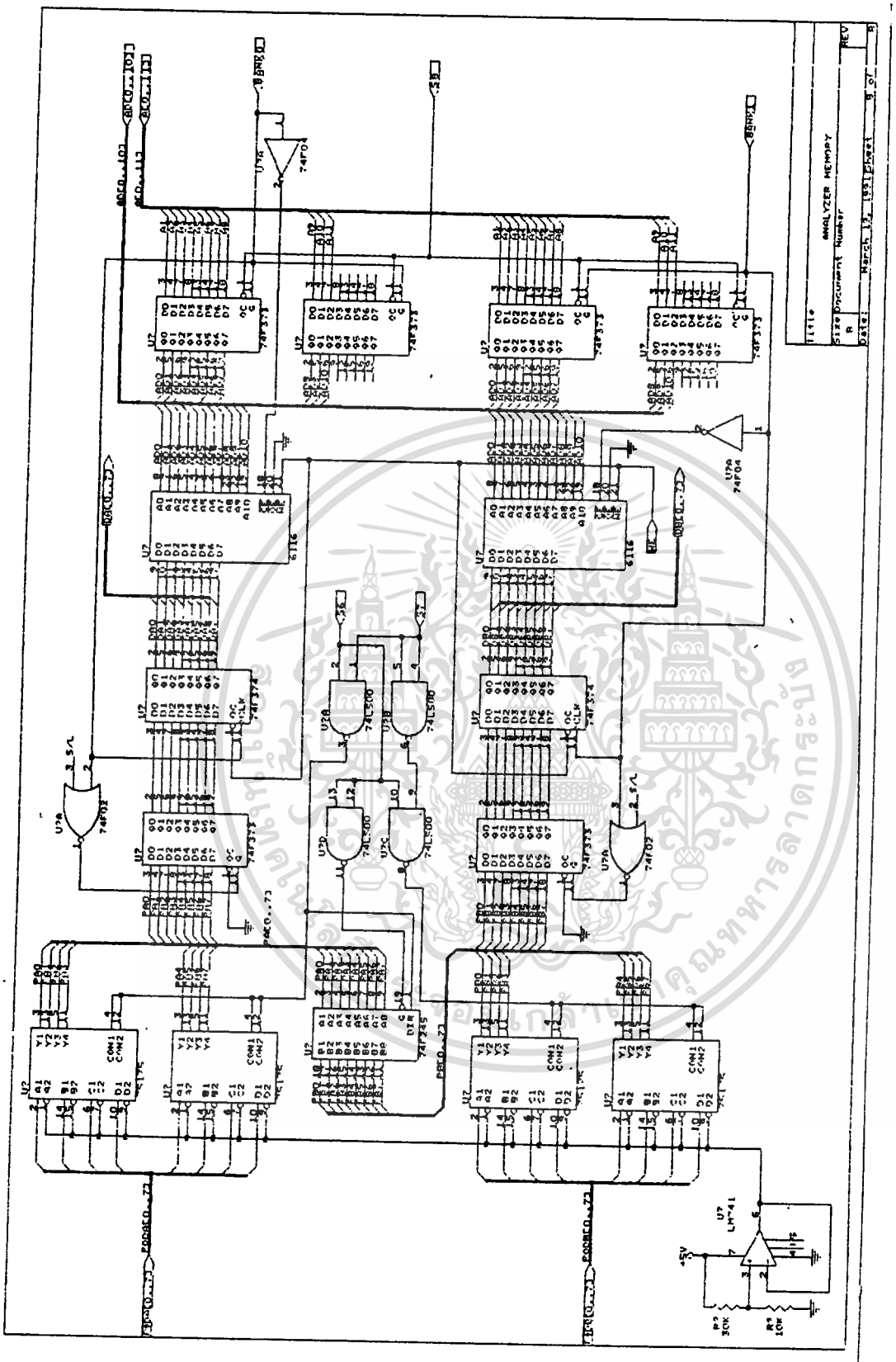
สัญญาณที่รับมาจากส่วนรับข้อมูลจะถูกตรวจจับโดย Line Receiver (75175) อีกครั้งหนึ่งเพื่อให้ได้สัญญาณที่ถูกต้อง โดยตั้งแรงดันอ้างอิงไว้ประมาณ 1.2 V เอ้าท์พุทของไอซีเบอร์นี้จะเป็นแบบ 3 สถานะทำให้สามารถเลือกได้ว่าจะเก็บข้อมูลจากอินพุทใด โดยใช้สัญญาณ S6,S7 เป็นตัวเลือก

การเลือกโหมดการทำงานเลือกโดยใช้ S6,S7 ดังนี้

S7	S6	โหมดการทำงาน	ช่องสัญญาณ
0	0	16 บิท	A,B
0	1	8 บิท	A
1	0	16 บิท	A,B
1	1	8 บิท	B

สัญญาณ S/L ใช้เลือกแบบการสุ่มข้อมูล ถ้า S/L เป็น,1 จะเป็นแบบแซมเปิ้ล ตัว 74LS373 จะทำตัวเป็นทางผ่านของสัญญาณเท่านั้น เมื่อ S/L เป็น 0 จะทำงานแบบแลทช์ 74LS373 และ 74LS374 จะทำงานร่วมกันเป็นมาสเตอร์-สลาฟฟลิปฟลอป ทำให้สามารถจับพัลซ์ที่มีขนาดแคบกว่าสัญญาณนาฬิกาได้

ในการอ่านข้อมูลจากแรมเพื่อนำไปเก็บลงหน่วยความจำของIBM PC จะต้องเซ็ตให้ S8 เป็น 1 เพื่อดิสเอเบิ้ลสัญญาณแอดเดรสจากลอจิกอนาไลเซอร์ แล้วส่งสัญญาณแอดเดรสมายัง AD0..AD10 และอ่านข้อมูลจาก DA0.



ภาพที่ 3-13 วงจรหน่วยความจำ

บทที่ 4

โปรแกรมควบคุมการทำงาน

การที่จะให้ระบบทำงานได้จะต้องมีโปรแกรมสำหรับควบคุมการทำงานของฮาร์ดแวร์ ดังนั้นจึงได้พัฒนาโปรแกรมขึ้นโดยใช้ภาษา C โดยที่การทำงานของโปรแกรมแบ่งออกเป็น 4 ส่วนใหญ่ๆ คือ

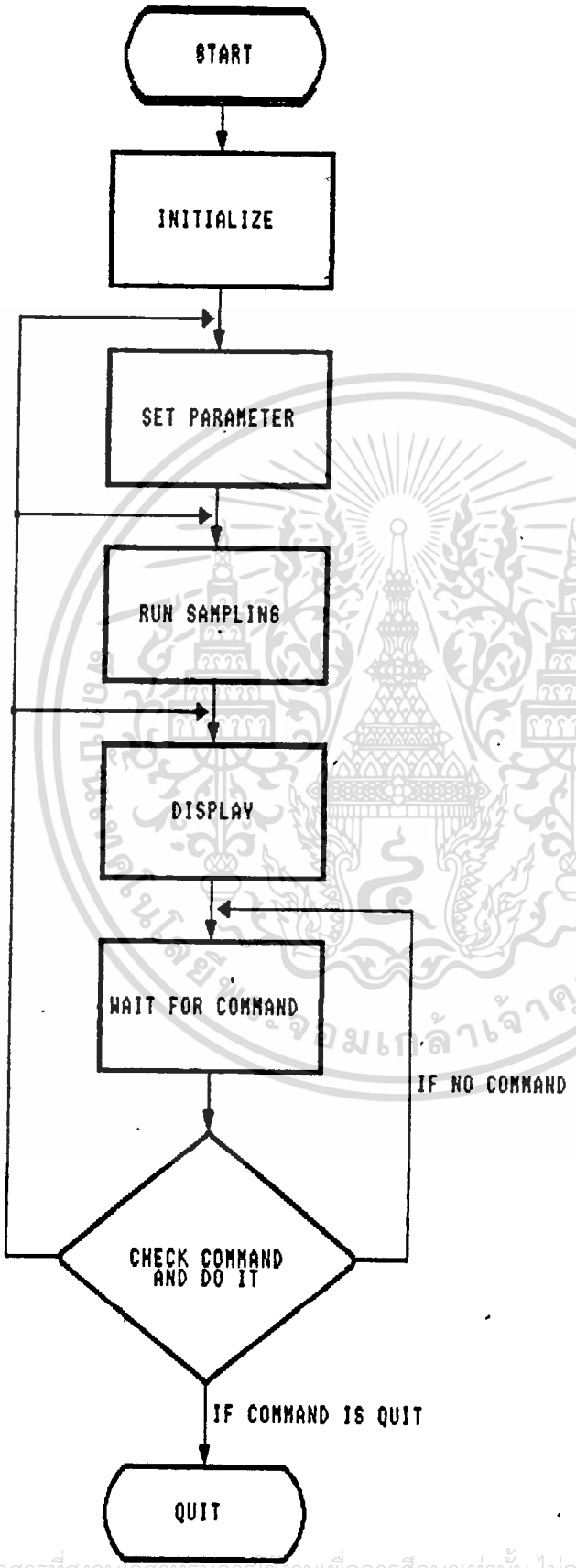
1.Initialize ทำหน้าที่ในการตรวจสอบฮาร์ดแวร์ เซ็ตค่าพารามิเตอร์เริ่มต้น

2.Menu เป็นส่วนที่ติดต่อกับผู้ใช้ ให้ทำการเลือกโหมดและตั้งค่าตัวแปรต่างๆ รวมทั้งรอรับคำสั่งจากผู้ใช้

3.Data flow control ควบคุมการรับส่งข้อมูลระหว่างการ์ดกับ CPU และจัดเตรียมข้อมูลให้เหมาะกับการนำไปแสดงผล และตรวจสอบการสุ่มข้อมูล

4.Display ทำหน้าที่ในการนำข้อมูลออกแสดงผล

ขั้นตอนการทำงานของโปรแกรมแสดงได้ดัง Flow chart ต่อไปนี้



```
/* program for logic analyzer card on IBM PC */
/* Test version                               */
#include <conio.h>
#include <stdio.h>
#include <stdarg.h>
#include <graphics.h>
#include <stdlib.h >
#include <dos.h>
#define ESC 0x1B
#define CR 0xOD
#define BS 0xO8
#define F1 59
#define F2 60
#define F3 61
#define F4 62
#define F5 63
#define F6 64
#define F10 68
#define LEFT_ARROW 75
#define RIGHT_ARROW 77
#define GR_LEFT 128
#define GR_TOP  25
#define GR_RIGHT 639
#define GR_BOTTOM 298
#define PORTRAIT 0
#define LANDSCAPE 1
#define PORT11 0xF300
#define PORT12 0xF301
#define PORT13 0xF302
#define PORT14 0xF303
#define PORT21 0xF304
#define PORT22 0xF305
#define PORT23 0xF306
```

```
#define PORT24 0xF307
#define PORT31 0XF308
#define PORT32 0xF309
#define PORT33 0xF30A
#define PORT34 0xF30B
#define PORT41 0xF30C
#define PORT42 0xF30D
#define PORT43 0xF30E
#define PORT44 0xF30F
#define PORT51 0xF310
#define PORT52 0xF311
#define PORT53 0xF312
#define PORT54 0xF313
/* This part is global variables */
typedef unsigned char byte;
typedef struct { unsigned n0 : 1; unsigned n1 : 1;
                 unsigned n2 : 1; unsigned n3 : 1;
                 unsigned n4 : 1; unsigned n5 : 1;
                 unsigned n6 : 1; unsigned n7 : 1;
               } bit;
char mode = 16, mag = 1;
static char *clk[] = {"INT CLK","EXT CLK"};
static char *rate[] = {"2 msec","1 msec","500 usec",
                       "200 usec","100 usec","50 usec",
                       "20 usec","10 usec","5 usec",
                       "2 usec","1 usec","500 nsec",
                       "200 nsec","100 nsec","50 nsec",
                       "25 nsec"};
static char *name[] = {"A0","A1","A2","A3","A4","A5",
                       "A6","A7","B0","B1","B2","B3",
                       "B4","B5","B6","B7"};
char *unit,c = 0,r = 14;
typedef union { bit  dat1;
```

```
                byte dat2;
              } bb ;
bb  da[2048],db[2048];
int cur = 0,curx,curb;
float curset = 0, time;
int mem = 2048,xpos = 0,ypos = 0;
/*-------------------------------------------------*/
bb twa,twb,tia,tib;
typedef struct {unsigned clksel : 6;
                unsigned s6     : 1;
                unsigned s7     : 1;
               } port13 ;
union { port13 pa13;
        byte   pb13;
      } p13;
typedef struct {unsigned s8   : 1;
                unsigned trig : 2;
                unsigned s11  : 1;
                unsigned s12  : 1;
                         : 1;
                unsigned start: 1;
                unsigned sl   : 1;
               } port23 ;
union { port23 pa23;
        byte   pb23;
      } p23;
byte addrl,delayl;
typedef struct {unsigned addrh  : 4;
                unsigned delayh : 4;
               } port33;
union { port33 pa33;
        byte   pb33;
      } p33;
```

```
byte saddr1;
typedef struct {unsigned saddrh : 4;
                unsigned full   : 1;
                unsigned stop   : 1;
                                : 0;
               } port42;
union { port42 pa42;
        byte   pb42;
      } p42;
typedef struct {unsigned q0     : 1;
                unsigned q1     : 1;
                unsigned q2     : 1;
                unsigned q3     : 1;
                unsigned q4     : 1;
                unsigned q5     : 1;
                                : 1;
                unsigned manual : 1;
               } port43;
union { port43 pa43;
        byte   pb43;
      } p43;
byte db0,db1,thv;
/*-------------------------------------------------*/
void TestGraphicError();
void InitialGraph();
void gprintf(int *xloc,int *yloc,char *fmt,...);
void eraseStr(int xloc,int yloc,char *str);
void gprintc(int *xloc,int *yloc,char *fmt,...);
void gprintxy(int xloc,int yloc,char *fmt,...);
void calcurset();
void header();
void timeset();
void savefile();
```

```
void loadfile();
void file();
void curmode();
void jump();
void pgraph(int mode,int direction);
/*-------------------------------------------------*/
void startup()
{ outp(PORT14,0x80); /*********************/
  outp(PORT24,0x80); /*                   */
  outp(PORT34,0x91); /* Set mode of 8255 */
  outp(PORT44,0x92); /*    */
  outp(PORT54,0x92); /*********************/
  twa.dat2 = twb.dat2 = tia.dat2 = tib.dat2 = 0;
  p13.pb13 = 0x21;
  p23.pb23 = 0x04;
  delay1 = 0xFF;
  p33.pa33.delayh = 0xF;
  p43.pb43 = 0;
  thv = 0x7F;
  outp(PORT11,twa.dat2); /************************/
  outp(PORT12,twb.dat2); /**/
  outp(PORT13,p13.pb13); /**/
  outp(PORT21,tia.dat2); /*                    */
  outp(PORT22,tib.dat2); /*  Set parameter     */
  outp(PORT23,p23.pb23); /*                    */
  outp(PORT32,delay1);   /*                    */
  outp(PORT33,p33.pb33); /*                    */
  outp(PORT43,p43.pb43); /*                    */
  outp(PORT53,thv);      /************************/
}
/*-------------------------------------------------*/
/* FUNCTION TestGraphicError()
   USAGE To test error occured after some graphic
```

```
            funtion  then  print  error messages and exit
            to DOS.                                        */
 void TestGraphicError()
 {
   int errorcode;
   errorcode = graphresult();
   if (errorcode != grOk)
   {
     closegraph();
     printf("Graphics  System Error: %s\n",grapherrormsg
           (errorcode));
     exit(1);
   }
 }
 /*-------------------------------------------------------*/
 /* FUNCTION InitialGraph()
    USAGE  To change to graphic mode ,set mode for EGAHI
           and set FONTS                                   */
 void InitialGraph()
 {
   int GraphDriver,GraphMode ;
   GraphDriver = DETECT ;
   initgraph(&GraphDriver,&GraphMode,"b:");
   TestGraphicError();
   setgraphmode(EGAHI);
   TestGraphicError();
   settextjustify(LEFT_TEXT,TOP_TEXT);
 }
 /*-------------------------------------------------------*/
 /* FUNCTION gprintf()
    USAGE  Print  variable  with  the  string  format in
           graphic mode                                    */
void gprintf(int *xloc,int *yloc,char *fmt, ...)
```

```
{
   va_list  argptr;
   char   str[140];
   struct textsettingstype textinfo;
   gettextsettings(&textinfo);
   va_start(argptr,fmt);
   vsprintf(str,fmt,argptr);
   outtextxy(*xloc,*yloc,str);
   if (textinfo.direction == HORIZ_DIR)
*yloc += textheight(str)+2;
  else  *xloc += textheight(str)+2;
  va_end(argptr);
}
/*-------------------------------------------------------*/
/* FUNCTION erasestr()
   USAGE  Erase  the  appropriate  area  of  the screen
          before writting a string to the screen     */
void erasestr(int xloc,int yloc,char *str)
{
  struct textsettingstype textinfo;
  int     xdim,ydim;
  void *textimage;
  gettextsettings(&textinfo);
  switch (textinfo.direction)
  {
    case HORIZ_DIR : xdim = textwidth(str);
     ydim = textheight(str);
     xloc--; break;
    case VERT_DIR  : xdim = textheight(str);
     ydim = textwidth(str);
     yloc++; break;
  }
  switch (textinfo.horiz)
```

```
  {
    case LEFT_TEXT   : break;
    case CENTER_TEXT : xloc -= xdim/2; break;
    case RIGHT_TEXT  : xloc -= xdim  ; break;
  }
  switch (textinfo.vert)
  {
    case BOTTOM_TEXT : yloc -= ydim  ; break;
    case CENTER_TEXT : yloc -= ydim/2; break;
    case TOP_TEXT    : break;
  }
  while (xloc<0) {xloc++; xdim--;}
  while (yloc<0) {yloc++; ydim--;}
  textimage = malloc(imagesize(xloc,yloc,xdim,ydim));
  getimage(xloc,yloc,xloc+xdim,yloc+ydim,textimage);
  putimage(xloc,yloc,textimage,XOR_PUT);
  free(textimage);
}
/*-------------------------------------------------*/
/* FUNCTION gprintc()
   USAGE  Write the output text on the screen and clear
          the old text.                              */
void gprintc(int *xloc,int *yloc,char *fmt, ...)
{
  va_list  argptr;
  char     str[140];
  struct textsettingstype textinfo;
  gettextsettings(&textinfo);
  va_start(argptr,fmt);
  vsprintf(str,fmt,argptr);
  erasestr(*xloc,*yloc,str);
  outtextxy(*xloc,*yloc,str);
  if (textinfo.direction == HORIZ_DIR)
```

```
   *yloc += textheight(str)+2;
    else  *xloc += textheight(str)+2;
    va_end(argptr);
  }
  /*-------------------------------------------------*/
  void gprintxy(int xloc,int yloc,char *fmt,...)
  {
    va_list argptr;
    char str[140];
    struct textsettingstype textinfo;
    va_start(argptr,fmt);
    vsprintf(str,fmt,argptr);
    gettextsettings(&textinfo);
    erasestr(xloc,yloc,str);
    outtextxy(xloc,yloc,str);
    va_end(argptr);
  }
  /*-------------------------------------------------*/
  void calcurset()
  {
    curset = time*curb;
    if (curset <= 9.999e-8)
       curset = curset/1e-9, unit = "nsec";
    else
       { if (curset <= 9.999e-5)
    curset = curset/1e-6, unit = "usec";
         else
    { if (curset <= 9.999e-2)
         curset = curset/1e-3, unit = "msec";
      else
         unit = " sec";
    }
       }
  }
```

```
/*-------------------------------------------------*/
void timeset()
{
  switch(r)
    { case 0: time = 2e-3; break;
      case 1: time = 1e-3; break;
      case 2: time = 5e-4; break;
      case 3: time = 2e-4; break;
      case 4: time = 1e-4; break;
      case 5: time = 5e-5; break;
      case 6: time = 2e-5; break;
      case 7: time = 1e-5; break;
      case 8: time = 5e-6; break;
      case 9: time = 2e-6; break;
      case 10:time = 1e-6; break;
      case 11:time = 5e-7; break;
      case 12:time = 2e-7; break;
      case 13:time = 1e-7; break;
      case 14:time = 5e-8; break;
      case 15:time = 2.5e-8; break;
      case 16:time = 0;    break;
    }
}
/*-------------------------------------------------*/
void header()
{
  bb a,b;
  int x=3,y=2;
  setcolor(EGA_YELLOW);
  calcurset();
  a.dat2=da[cur].dat2;
  b.dat2=db[cur].dat2;
  gprintf(&x,&y,"MODE %2d BIT    MAG=x%-2d    MEMORY"
```

```
            " %4d BYTE   CUR=%04d     CUR/SET=%08.5f %s",
             mode,mag,mem,cur,curset,unit);
    gprintf(&x,&y,"A[%d%d%d%d%d%d%d%d]    "
             "B[%d%d%d%d%d%d%d%d]        %7s=%6s",
             a.dat1.n7,a.dat1.n6,a.dat1.n5,a.dat1.n4,
             a.dat1.n3,a.dat1.n2,a.dat1.n1,a.dat1.n0,
             b.dat1.n7,b.dat1.n6,b.dat1.n5,b.dat1.n4,
             b.dat1.n3,b.dat1.n2,b.dat1.n1,b.dat1.n0,
             clk[c],rate[r]);
}
/*-------------------------------------------------*/
void changehead()
{
   setviewport(0,0,639,23,0);
   clearviewport();
   header();
   setviewport(GR_LEFT,GR_TOP,GR_RIGHT,GR_BOTTOM,0);
}
/*-------------------------------------------------*/
void border()
{
   setcolor(EGA_LIGHTRED);
   setwritemode(COPY_PUT);
   setlinestyle(SOLID_LINE,0xffff,NORM_WIDTH);
   line(0,24,639,24);
   line(0,299,639,299);
   line(127,24,127,299);
}
/*-------------------------------------------------*/
void wrtname()
{
  int x=0,y;
  char i;
```

```
  setcolor(EGA_BROWN);
   for (i=0,y=30;i<=15;i++,y+=7)
     gprintf(&x,&y,"%12s",name[i]);
 }
 /*-------------------------------------------------*/
 void bottom()
 {
   int x=12,y;
   char i;
   static char *funckey[]={"F1","F2","F3","F4","F5",
                           "F6","F10"};
   static char *command[]={"-HELP","-FILE","-PRINT",
                           "-RUN","-CURSOR","-JUMP",
                           "-MENU"};
   setfillstyle(SOLID_FILL,EGA_YELLOW);
   bar(0,334,639,349);
   for (i=0,y=336;i<=6;i++)
     {
      setcolor(EGA_CYAN);
      gprintf(&x,&y,"%s",funckey[i]);
      x += textwidth(funckey[i]);
      y = 336;
      setcolor(EGA_BLUE);
      gprintf(&x,&y,"%s",command[i]);
      x += textwidth(command[i])+26;
      y = 336;
     }
}
/*-------------------------------------------------*/
void graph1(int addr)
{
  int x=0 ,y=0 ,i;
  setcolor(EGA_BLUE);
```

```
setwritemode(COPY_PUT);
setlinestyle(SOLID_LINE,0xffff,NORM_WIDTH);
if (addr <= 1536)
{
  for (i=addr;i<=addr+511;i++,x++,y=1)
  {
    switch (da[i].dat1.n0)
    {
      case 1: if ((i-1)>0)
              if (da[i-1].dat1.n0 == 0)
                  line(x,y,x,y+12);
              putpixel(x,y,EGA_BLUE);
              break;
      case 0: if ((i-1)>0)
              if (da[i-1].dat1.n0 == 1)
                  line(x,y,x,y+12);
              putpixel(x,y+12,EGA_BLUE);
    }
    y+=17;
    switch (da[i].dat1.n1)
    {
      case 1: if ((i-1)>0)
              if (da[i-1].dat1.n1 == 0)
                  line(x,y,x,y+12);
              putpixel(x,y,EGA_BLUE);
              break;
      case 0: if ((i-1)>0)
              if (da[i-1].dat1.n1 == 1)
                  line(x,y,x,y+12);
              putpixel(x,y+12,EGA_BLUE);
    }
    y+=17;
    switch (da[i].dat1.n2)
```

```
{
  case 1: if ((i-1)>0)
          if (da[i-1].dat1.n2 == 0) line(x,y,x,y+
          putpixel(x,y,EGA_BLUE);
          break;
  case 0: if ((i-1)>0)
          if (da[i-1].dat1.n2 == 1)
              line(x,y,x,y+12);
          putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (da[i].dat1.n3)
{
  case 1: if ((i-1)>0)
          if (da[i-1].dat1.n3 == 0)
              line(x,y,x,y+12);
          putpixel(x,y,EGA_BLUE);
          break;
  case 0: if ((i-1)>0)
          if (da[i-1].dat1.n3 == 1)
              line(x,y,x,y+12);
          putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (da[i].dat1.n4)
{
  case 1: if ((i-1)>0)
          if (da[i-1].dat1.n4 == 0)
              line(x,y,x,y+12);
          putpixel(x,y,EGA_BLUE);
          break;
  case 0: if ((i-1)>0)
          if (da[i-1].dat1.n4 == 1)
```

```
            line(x,y,x,y+12);
            putpixel(x,y+12,EGA_BLUE);
    }
    y+=17;
    switch (da[i].dat1.n5)
    {
      case 1: if ((i-1)>0)
              if (da[i-1].dat1.n5 == 0)
                  line(x,y,x,y+12);
      putpixel(x,y,EGA_BLUE);
      break;
      case 0: if ((i-1)>0)
              if (da[i-1].dat1.n5 == 1)
                  line(x,y,x,y+12);
      putpixel(x,y+12,EGA_BLUE);
    }
    y+=17;
    switch (da[i].dat1.n6)
    {
      case 1: if ((i-1)>0)
              if (da[i-1].dat1.n6 == 0)
                  line(x,y,x,y+12);
              putpixel(x,y,EGA_BLUE);
              break;
      case 0: if ((i-1)>0)
              if (da[i-1].dat1.n6 == 1)
                  line(x,y,x,y+12);
              putpixel(x,y+12,EGA_BLUE);
    }
    y+=17;
    switch (da[i].dat1.n7)
    {
      case 1: if ((i-1)>0)
```

```
            if (da[i-1].dat1.n7 == 0)
                line(x,y,x,y+12);
            putpixel(x,y,EGA_BLUE);
            break;
   case 0: if ((i-1)>0)
            if (da[i-1].dat1.n7 == 1)
                line(x,y,x,y+12);
                putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (db[i].dat1.n0)
{
  case 1: if ((i-1)>0)
            if (db[i-1].dat1.n0 == 0)
                line(x,y,x,y+12);
           putpixel(x,y,EGA_BLUE);
           break;
  case 0: if ((i-1)>0)
           if (db[i-1].dat1.n0 == 1)
               line(x,y,x,y+12);
          putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (db[i].dat1.n1)
{
  case 1: if ((i-1)>0)
           if (db[i-1].dat1.n1 == 0)
               line(x,y,x,y+12);
          putpixel(x,y,EGA_BLUE);
          break;
  case 0: if ((i-1)>0)
          if (db[i-1].dat1.n1 == 1)
              line(x,y,x,y+12);
```

```
            putpixel(x,y+12,EGA_BLUE);
 }
 y+=17;
 switch (db[i].dat1.n2)
 {
   case 1: if ((i-1)>0)
           if (db[i-1].dat1.n2 == 0)
               line(x,y,x,y+12);
           putpixel(x,y,EGA_BLUE);
           break;
   case 0: if ((i-1)>0)
           if (db[i-1].dat1.n2 == 1)
               line(x,y,x,y+12);
           putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (db[i].dat1.n3)
{
  case 1: if ((i-1)>0)
          if (db[i-1].dat1.n3 == 0)
              line(x,y,x,y+12);
          putpixel(x,y,EGA_BLUE);
          break;
  case 0: if ((i-1)>0)
          if (db[i-1].dat1.n3 == 1)
          line(x,y,x,y+12);
          putpixel(x,y+12,EGA_BLUE);
}
y+=17;
switch (db[i].dat1.n4)
{
  case 1: if ((i-1)>0)
          if (db[i-1].dat1.n4 == 0)
```

```
                line(x,y,x,y+12);
            putpixel(x,y,EGA_BLUE);
            break;
    case 0: if ((i-1)>0)
             if (db[i-1].dat1.n4 == 1)
                 line(x,y,x,y+12);
            putpixel(x,y+12,EGA_BLUE);
 }
 y+=17;
 switch (db[i].dat1.n5)
 {
   case 1: if ((i-1)>0)
            if (db[i-1].dat1.n5 == 0)
                line(x,y,x,y+12);
           putpixel(x,y,EGA_BLUE);
           break;
  case 0: if ((i-1)>0)
            if (db[i-1].dat1.n5 == 1)
                line(x,y,x,y+12);
           putpixel(x,y+12,EGA_BLUE);
 }
 y+=17;
 switch (db[i].dat1.n6)
 {
   case 1: if ((i-1)>0)
            if (db[i-1].dat1.n6 == 0)
                line(x,y,x,y+12);
           putpixel(x,y,EGA_BLUE);
           break;
   case 0: if ((i-1)>0)
           if (db[i-1].dat1.n6 == 1)
           line(x,y,x,y+12);
          putpixel(x,y+12,EGA_BLUE);
}
```

```
        y+=17;
        switch (db[i].dat1.n7)
        {
          case 1: if ((i-1)>0)
                  if (db[i-1].dat1.n7 == 0)
                      line(x,y,x,y+12);
                  putpixel(x,y,EGA_BLUE);
                  break;
          case 0: if ((i-1)>0)
                  if (db[i-1].dat1.n7 == 1)
                      line(x,y,x,y+12);
                  putpixel(x,y+12,EGA_BLUE);
        }
      }
    }
  }
/*-------------------------------------------------*/
void drawcursor()
{
  setcolor(EGA_RED);
  setwritemode(XOR_PUT);
  setlinestyle(DOTTED_LINE,0xffff,NORM_WIDTH);
  line(curx,0,curx,273);
}
/*-------------------------------------------------*/
void putcur()
{
  setcolor(EGA_LIGHTRED);
  setwritemode(XOR_PUT);
  setlinestyle(SOLID_LINE,0xffff,NORM_WIDTH);
  line(curx,0,curx,273);
}
/*-------------------------------------------------*/
```

```
void erasecursor()
{
  drawcursor();
}
/*-------------------------------------------------------*/
void movcurright()
{
  erasecursor();
  curx++; curb++;
  drawcursor();
}
/*-------------------------------------------------------*/
void movcurleft()
{
  erasecursor();
  curx--; curb--;
  drawcursor();
}
/*-------------------------------------------------------*/
unsigned char readkbd(unsigned char *scancode)
{
  _AH = 0, geninterrupt(0x16);
  *scancode = _AH;
  return(_AL);
}
/*-------------------------------------------------------*/
void scrollright()
{
  if (cur < 2047)
  {
    cur++,curb++;
    clearviewport();
    graph1(cur-511);
```

```
      drawcursor();
    }
  }
  /*-------------------------------------------------*/
  void scrollleft()
  {
    if (cur > 0)
    {
      cur--,curb--;
      clearviewport();
      graph1(cur);
      drawcursor();
    }
  }
  /*-------------------------------------------------*/
  void moveleft(int *curs, int *cure)
  {
    if (*cure <= *curs)
       curx--,curb++,(*cure)--,putcur();
    else
       putcur(),curb--,(*cure)--,curx--;
  }
  /*-------------------------------------------------*/
  void moveright(int *curs, int *cure)
  {
    if (*cure >= *curs)
       curx++;curb++,(*cure)++,putcur();
    else
       putcur(),curb--,(*cure)++,curx++;
  }
  /*-------------------------------------------------*/
  void curmode()
  {
```

```
    int  curs,cure;
    char ch,code;
    curb=0;
    curs=cure=cur;
    erasecursor();
    putcur();
    changehead();
    do
    {
     readkbd(&code);
     switch (code)
     {
      case LEFT_ARROW : if (curx > 0)
                        moveleft(&curs,&cure), cur--;
                        changehead();
                        break;
      case RIGHT_ARROW: if (curx < 511)
                        moveright(&curs,&cure), cur++;
                        changehead();
                        break;
      case F5         : clearviewport();
                        curb = cur;
                        graph1(cur);
                        drawcursor();
                        changehead();
    }
   } while (code != F5);
}
/*-------------------------------------------------*/
void getfname(char *fname,int *nlength)
{
  char ch;
  int x=12,y=1;
```

```
   clearviewport();
    gprintxy(x,y,"ENTER FILE NAME : ");
    x+=144;
    do
    {
      gprintxy(x,y,"_");
      ch=getch();
      switch (ch)
      {
        case CR : break;
        case ESC: break;
        case BS : if (*nlength > 0)
    {
       x-=8, (*nlength)--;
       gprintxy(x,y,"_ ");
    }
break;
        default : if (*nlength < 12)
   {
      fname[*nlength]=ch;
      (*nlength)++;
      gprintxy(x,y,"%c",ch);
      x+=8;
    }
     }
   } while ((ch != CR)&&(ch != ESC));
}
/*-------------------------------------------------------*/
void savefile()
{
  FILE *fp,*fopen();
  char fname[13];
  int x=12,y=1,length=0,i;
```

```
    getfname(fname,&length);
    if (length != 0)
       {
        fname[length]=NULL;
        fp = fopen(fname,"w");
        if (fp == NULL)
           {
            clearviewport();
            gprintxy(x,y,"Can't open \"%s\" for writing."
                    ,fname);
           }
        else
           {
            for (i=0;i<2048;i++)
            fprintf(fp,"%c,%c",da[i].dat2,db[i].dat2);
            fclose(fp);
            y = 12;
            gprintxy(x,y,"Saving is successfull");
           }
       }
   else
       {
         clearviewport();
         gprintxy(x,y,"Select \"1\" for save file, \"2\""
                  "for load file, <ESC> to cancel.");
       }
}
/*-------------------------------------------------*/
void loadfile()
{
  FILE *fp,*fopen();
  char fname[13];
  int x=12,y=1,length=0,i;
```

```
    getfname(fname,&length);
    if (length != 0)
       {
        fname[length]=NULL;
        fp = fopen(fname,"r");
        if (fp == NULL)
            {
             clearviewport();
             gprintxy(x,y,"Can't open \"%s\" for loading"
                      ,fname);
            }
        else
            {
             for (i=0;i<2048;i++)
             fscanf(fp,"%c,%c",&da[i].dat2,&db[i].dat2);
             fclose(fp);
             y=12;
             gprintxy(x,y,"Loading is successfull");
            }
       }
   else
       {
        clearviewport();
        gprintxy(x,y,"Select \"1\" for save file, \"2\""
                 " for load file, <ESC> to cancel.");
       }
}
/*-------------------------------------------------*/
void file()
{
  char ch,code;
  int x=12,y=1;
  setcolor(EGA_LIGHTBLUE);
```

```
   setviewport(0,300,639,333,0);
   clearviewport();
   gprintxy(x,y,"Select \"1\" for save file, \"2\""
            " for load file, <ESC> to cancel.");
   do
   {
     ch=getch();
     switch(ch)
     {
       case '1' : savefile(); break;
       case '2' : loadfile();
     }
   } while (ch != ESC);
   clearviewport();
   setviewport(GR_LEFT,GR_TOP,GR_RIGHT,GR_BOTTOM,0);
   clearviewport();
   graph1(cur);
   drawcursor();
}
/*-------------------------------------------------------*
void pgraph(int mode,int direction)
{
  char m;
  int  i,j,k,msb,lsb,maxx=getmaxx(),maxy=getmaxy();
  struct viewporttype viewport;
  getviewsettings(&viewport);
  setviewport(0,0,maxx,maxy,0);
  fprintf(stdprn,"\x1BA%c",7);
  switch(direction)
  {
    case PORTRAIT:
    {
      lsb=maxx & 0x00FF;
```

```
        msb=maxx>>8;
        for(j=0;j<=maxy/8;j++)
        {
          fprintf(stdprn,"\x1B*%c%c%c",mode,lsb,msb);
          for(i=0;i<=maxx;i++)
          {
            m=0;
            for(k=0;k<8;k++)
            {
              m<<=1;
              if (getpixel(i,j*8+k)) m++;
            }
            fprintf(stdprn,"%c",m);
          }
          fprintf(stdprn,"\x0D\x0A");
        }
      }
      case LANDSCAPE:
      {
        lsb=maxy&0x00FF;
        msb=maxy>>8;
        for(j=0;j<maxx;j+=8)
        {
fprintf(stdprn,"\x1B*%c%c%c",mode,lsb,msb);
          for(i=maxy;i>=0;i--)
          {
            m=0;
            for(k=0;k<8;k++)
            {
              m<<=1;
              if (getpixel(j+k,i)) m++;
            }
            fprintf(stdprn,"%c",m);
```

```
        }
        fprintf(stdprn,"\x0D\x0A");
      }
    }
  }
  fprintf(stdprn,"\f");
}
/*-------------------------------------------------*/
void jump()
{
  char str[5];
  int  begin,x=12,y=1;
  unsigned char count=0, ch;
  setcolor(EGA_LIGHTBLUE);
  setviewport(0,300,639,383,0);
  gprintxy(x,y,"JUMP TO CURSOR:");
  x +=125;
  do
  {
    gprintxy(x,y,"_");
    ch=getch();
    switch (ch)
    {
      case BS : if (count > 0)
                {
                  x-=8, count--;
                  gprintxy(x,y,"_ ");
                }
                break;
      default : if ((ch>='0')&&(ch<='9')&&(count<4))
                {
                  str[count]=ch;
                  count++;
```

```
                gprintxy(x,y,"%c",ch);
                x+=8;
              }
  }
} while ((ch != ESC)&&(ch != CR));
if ((ch != ESC)&&(count != 0))
   {
     str[count]=NULL;
     cur=atoi(str);
     if (cur < 2048)
     {
       if ((cur >= 512)&&(cur < 1536))
       {
        begin = cur - curx;
        curb  = cur;
       }
       if ((cur >= 0)&&(cur < 512))
       {
        begin = 0;
        curb = curx = cur;
       }
       if (cur >= 1536)
       {
        begin = 1536;
        curb  = cur;
        curx  = cur - begin;
       }
       clearviewport();
       setviewport(GR_LEFT,GR_TOP,GR_RIGHT,
                   GR_BOTTOM,0);
       clearviewport();
       graph1(begin);
       drawcursor();
```

```
          changehead();
        }
        else
        {
         clearviewport();
         setviewport(GR_LEFT,GR_TOP,GR_RIGHT,
                     GR_BOTTOM,0);
        }
      }
   else
      {
        clearviewport();
        setviewport(GR_LEFT,GR_TOP,GR_RIGHT,
                    GR_BOTTOM,0);
      }
}
/*-------------------------------------------------------*/
main()
{ unsigned char ch,keycode;
  InitialGraph();
  setusercharsize(4,3,5,4);
  setbkcolor(EGA_LIGHTGRAY);
  timeset();
  header();
  border();
  wrtname();
  bottom();
  do
  {
   ch = readkbd(&keycode);
   switch (ch)
   {
     case 0: switch (keycode)
```

```
{
 case F2:file();
         break;
 case F4:setviewport(GR_LEFT,GR_TOP,
         GR_RIGHT,GR_BOTTOM,0);
         clearviewport();
         cur=curb=0;
         graph1(cur);
         curx=0;
         drawcursor();
         changehead();
         break;
 case F5:curmode();
         break;
 case F6:jump();
         break;
 case LEFT_ARROW:
         if (curx > 0)
           movcurleft(), cur--;
         else
           scrollleft();
         changehead();
         break;
 case RIGHT_ARROW:
         if (curx < 511)
             movcurright(), cur++;
         else
         scrollright();
         changehead();
         break;
case F10:break;
}
}
```

```
  } while (keycode != F10);
  closegraph();
}
```

บทที่ 5
สรุปผลและวิจารณ์

1. ในส่วนของโปรแกรมควบคุมการทำงานที่เขียนขึ้นโดยใช้ภาษา c นั้นสามารถควบคุมการทำงานได้ คือ

1.1 สามารถนำสัญญาณที่วัดได้จากส่วนรับข้อมูล และเก็บอยู่ในหน่วยความจำออกมาแสดงผลทาง IBM PC

1.2 สามารถทำงานได้ตามฟังก์ชั่นการทำงานพื้นฐานของ การ์ดวิเคราะห์สัญญาณดิจิตอล

2. ในส่วนการทำงานทางด้านฮาร์ดแวร์วงจรสามารถทำงานได้ตามที่ออกแบบไว้ในแต่ละส่วน แต่เมื่อนำวงจรในแต่ละส่วนมารวมกันจะเกิดปัญหาทางด้านความช้าเร็วในการทำงานของวงจรไม่สัมพันธ์กัน ซึ่งได้แก้ปัญหาในส่วนนี้โดยการหน่วงเวลาในบางส่วนเพื่อให้ทำการทำงานสัมพันธ์กัน

3. โครงงานชิ้นนี้ถ้าได้รับ การพัฒนาในด้านการเพิ่มความถี่ในการสุ่มข้อมูล , การเพิ่มฟังก์ชั่นต่างๆ และการพัฒนาโปรแกรมต่อไป จะทำให้มีประสิทธิภาพในการทำงานสูงขึ้น และมีราคาถูกกว่าเครื่องลอจิกอนาไลเซอร์ทั่วๆไป

เอกสารอ้างอิง

1.ทินกร ดุ๊ก และ ธานินทร์ ถาวรคาสนวงค์, "การอินเทอร์เฟสไอบีเอ็มพีซี", ฟิสิกส์ เซนเตอร์, 2530

2."Basics of logic analysis", 1985

3.Technical information center, "FAST and LS TTL data", Motorolar Inc, 1986

4.Technical information center, "Linear and Interface IC", Motorolar Inc, 1986

5.Borland International Inc, "Turbo C user guide , reference guide", 1988