เครื่องวิเคราะห์สัญญาณตรรก

LOGIC ANALYZER

โดย

นายจิตกร ดีสกุล

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เครื่องวิเคราะสัญญาณตรรก

LOGIC ANALYZER

โดย

นายจิตกร ดีสกุล 36013006

อาจารย์ที่ปรึกษา

อาจารย์ เกรียงไกร วงศ์โรจนภรณ์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมโทรคมนาคม

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องวิเคราะห์สัญญาณตรรก

LOGIC ANALYZER

ผู้จัดทำ

นายจิตกร ดีสกุล 36013006

_____ อาจารย์ที่ปรึกษา

(อาจารย์ เกรียงไกร วงศ์โรจนภรณ์)

เครื่องวิเคราะร์สัญญาณตรรก
LOGIC ANALYZER

โดย

นายจิตกร ดีสกุล 36013006

อาจารย์ที่ปรึกษา

อาจารย์ เกรียงไกร วงศ์โรจนภรณ์

บทคัดย่อ

ปริญญานิพนร์ฉบับนี้ เป็นการสร้างเครื่องวิเคราะห์สัญญาณตรรก โดยแนวทางการพัฒนาระบบ จะใช้บอร์ด JR 180 ซึ่งบอร์ดนี้จะใช้ไมโครโปรเซสเซอร์เบอร์ Z80180 ซึ่งมีขนาด 8 บิท เป็นที่สามารถตรวจสอบสภาวะของสัญญาณทางตรรกในเครื่องคอมพิวเตอร์ หรืออุปกรณ์อิเลคทรอนิคส์ต่างๆ ทั่วไปได้ถึง 40 ช่องสัญญาณ โดยมีอัตราการสุ่มสัญญาณ 50 เมกกะเฮิทซ์ และสามารถลดทอนอัตราการสุ่มสัญญาณได้ด้วยอัตรา 5 - 2 - 1 (50 - 20 - 10 เมกกะเฮิทซ์) การควบคุมการทำงานและการแสดงผล จะกระทำโดยอาศัยเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป ซึ่งทุกเครื่องจะมีพอร์ทการเชื่อมต่อข้อมูลแบบอนุกรม เมื่อทำการใช้เครื่องวิเคราะห์สัญญาณตรรกตรวจสอบข้อมูลได้อย่างถูกต้อง ก็จะทำให้การวิเคราะห์ปัญหาที่เกิดขึ้น สามารถกระทำได้อย่างรวดเร็วและแม่นยำ ซึ่งจะทำให้ระยะเวลาในการแก้ปัญหาที่เกิดขึ้นน้อยลงจากเดิมมาก

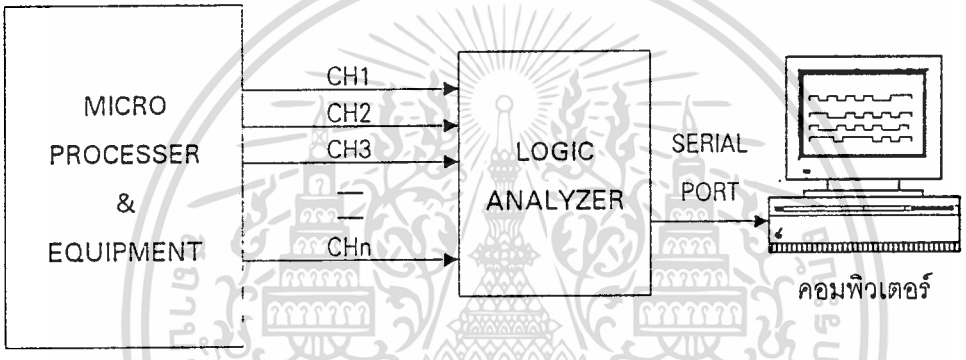
ABSTRACT

This paper presents the developing of Logic Analyzer . It uses a microcontroler bord (JR180) with use microprocessor 8 bit (Z80180) . The Logic Analyzer can measuring logic state in the personal computer and electronic equipments . Logic Analyzer has an input maximum 40 chanals for input measurment , use high sampling rate (50 MHz) , and decreasing sampling rate in 5 - 2 - 1 . It can use personal computer to control Logic Analyzer and to display the logic state . A Personal computer is a general computer which has serial port (RS-232C) . When using the Logic Analyzer the problem can be easily analized and the solving time can be greatly reduced

สารบัญ

บทที่ 1
บทนำ

ในปัจจุบันระบบสัญญาณดรรก (Digital Logic Signal) ถูกนำมาใช้แทนระบบสัญญาณแบบต่อเนื่อง (Analog Signal) ในระบบสัญญาณแบบดรรกจะมีจำนวนเส้นของข้อมูล (Data) และตำแหน่ง (Address) หลายเส้น ในการวิเคราะห์สัญญาณเหล่านั้น ถ้าใช้เครื่องมือวัดธรรมดา เช่น ออสซิโลสโคป หรือมัลติมิเตอร์ มาทำการวัดสัญญาณจะมีความยุ่งยากมาก และสามารถตรวจสอบความผิดพลาดเฉพาะทางอุปกรณ์พื้นฐานเท่านั้น ส่วนทางด้านโปรแกรมจะไม่สามารถทำการวิเคราะห์ได้ ด้วยเหตุนี้จึงได้ใช้เครื่องวิเคราะห์สัญญาณดรรกเข้ามาช่วย เครื่องวิเคราะห์สัญญาณดรรกสามารถที่จะทำการวัดสัญญาณหลายๆ สัญญาณพร้อมกัน ซึ่งทำให้ผู้ใช้สามารถทำการวิเคราะห์จุดบกพร่องได้ง่ายดังรูปที่ 1.1



รูปที่ 1.1 การใช้เครื่องวิเคราะห์สัญญาณดรรก

จากรูปที่ 1.1 เป็นการวิเคราะห์สัญญาณจากแผงวงจรไมโครโปรเซสเซอร์และอุปกรณ์อื่นๆ จากรูปสามารถทำการวัดได้ที่ละหลายๆ ช่องสัญญาณ ซึ่งสัญญาณที่วัดได้จะแสดงบนจอภาพของเครื่องคอมพิวเตอร์

วัตถุประสงค์ของปริญญานิพนธ์นี้ก็เพื่อต้องการพัฒนาอุปกรณ์ที่เรียกว่า เครื่องวิเคราะห์สัญญาณดรรก ให้มีราคาไม่แพงเกินไป สามารถใช้งานได้ง่าย มีประสิทธิภาพสูง และสามารถตรวจจับสัญญาณได้สูงถึง 40 ช่องสัญญาณ ซึ่งสามารถรองรับเครื่องคอมพิวเตอร์ส่วนบุคคลในปัจจุบัน (32 Bit) ใช้อัตราการสุ่มสัญญาณสูง 50 แมกกะเฮิร์ท

เครื่องวิเคราะห์ที่ทำการพัฒนาขึ้นมีขอบเขตในการพัฒนา 2 ส่วนด้วยกันคือ

- วงจรของเครื่องวิเคราะห์สัญญาณดรรก ซึ่งพัฒนาโดยใช้ไมโครโปรเซสเซอร์ Z80180 หรือ HD64180

- โปรแกรมควบคุมระบบของเครื่องวิเคราะห์สัญญาณตรรก โดยที่เครื่องวิเคราะห์ต้องอาศัยเครื่องคอมพิวเตอร์ส่วนบุคคลในการแสดงผล

บทที่ 2
ทฤษฎีเครื่องวิเคราะห์สัญญาณตรรก

2.1 โครงสร้างและหลักการทำงาน

โครงสร้างและหลักการทำงานทั่วไปของเครื่องวิเคราะห์สัญญาณตรรกศาสตร์นั้นสามารถแสดงดังผังการทำงานในรูปที่ 2.1 ซึ่งประกอบไปด้วยส่วนย่อยๆ ดังนี้

2.1.1. ส่วนสัญญาณด้านขาเข้าแบบขนาน (PARALLEL DATA INPUT)

วงจรประกอบไปด้วย วงจรศักดาไฟฟ้าขีดเริ่มเปลี่ยน (THRESHOLD VOLTAGE) , วงจรแปลงจาก ECL-TTL , วงจรหน่วงสัญญาณ , วงจรอินเวอร์เตอร์ และวงจรสุ่มตัวอย่างสัญญาณ (SAMPLE / LATCH) ซึ่งจุดประสงค์ของส่วนสัญญาณด้านขาเข้าแบบขนานนี้ เป็นตัวปรับระดับของสัญญาณที่เข้ามาให้มีระดับที่เหมาะสมเพื่อนำไปประมวลผลต่อไป

2.1.2. ส่วนเลือกสัญญาณ (WORD RECOGNIZER)

วงจรส่วนเลือกสัญญาณ จะทำการนำสัญญาณทางตรรก (LOGIC) ของสัญญาณที่เข้ามาจากขั้ววัดสัญญาณ (PROBE) มาทำการเปรียบเทียบกับสัญญาณ ขาเข้าที่ได้เลือกไว้ (QUALIFIER INPUT) ถ้าสัญญาณทั้งสองตรงกัน วงจรส่วนเลือกสัญญาณจะให้ระดับสัญญาณขาออกเป็นสูง (HIGH) และถ้าสัญญาณทั้งสองไม่ตรงกัน ระดับสัญญาณขาออกจะเป็นต่ำ (LOW) ซึ่งสัญญาณที่ได้จากวงจรส่วนเลือกนี้จะนำไปเป็นสัญญาณกระตุ้นแบบไม่ได้จังหวะ (ASYNCHRONOUS TRIGGER PLUSE)

2.1.3. ส่วนหน่วยเก็บความจำแบบขนาน (PARALLEL ACQISTION MEMORY)

เป็นวงจรที่ประกอบไปด้วยหน่วยความจำแบบแรม (RAM) และวงจรนับตำแหน่ง (ADDRESS COUNTER) หน่วยความจำแรม จะทำหน้าที่เป็นตัวเก็บสัญญาณตรรกขาเข้าซึ่งผ่านการปรับระดับของสัญญาณแล้ว และวงจรนับตำแหน่ง จะเป็นตัวนับตำแหน่งของหน่วยความจำ เพื่อทำการเก็บข้อมูลของสัญญาณตรรกที่เข้ามาแบบขนาน หรือเป็นตัวนับตำแหน่งของหน่วยความจำแรมที่ต้องการนำข้อมูลออกมาพิจารณา ส่วนนี้จะทำการนับเมื่อมีสัญญาณกระตุ้นจากส่วนหน่วงสัญญาณกระตุ้น (TRIGGER DELAY)

2.1.4. ส่วนหน่วงสัญญาณกระตุ้น (TRIGGER DELAY)

ส่วนหน่วงสัญญาณกระตุ้นเป็นวงจรซึ่งประกอบไปด้วย วงจรกระตุ้น (TRIGGER) และวงจรหน่วงต่างๆ เช่น TRIGGER DELAY COUNTER , TRIGGER GATE STAGE , DELAYED GATE

STAGE และ อื่นๆ ส่วนหน่วงสัญญาณกระตุ้นจะรับสัญญาณกระตุ้น (TRIGGER PLUSE) มาทำการหน่วงเวลาแล้วส่งสัญญาณไปทำการกระตุ้นวงจร PARALLEL ACQUISITION MEMORY

2.1.5. ส่วนคาบของสัญญาณนาฬิกา (TIME BESE)

เป็นส่วนที่ประกอบด้วยวงจรผลิตความถี่มาตราฐาน และวงจรหารความถี่ ซึ่งทำหน้าที่จ่ายสัญญาณนาฬิกาให้ไมโครโปรเซสเซอร์ , รอม , แรม , วงจรนับตำแหน่ง และหน่วยประมวลผลต่างๆ

2.1.6. ส่วนรับสัญญาณขาเข้าแบบอนุกรม (SERIAL / SIGNATURE INPUT)

เป็นวงจรที่ทำหน้าที่เปรียบเทียบสัญญาณขาเข้า ปรับแต่งระดับสัญญาณให้เหมาะสม เพื่อส่งเข้าสู่วงจรในส่วนอนุกรมอื่นๆ ต่อไป

2.1.7. ส่วนกำเนิดสัญญาณ (SIGNATURE GENERATER)

เป็นส่วนที่รับสัญญาณแบบอนุกรมเข้ามาทำการถอดรหัส (DECODE) ให้เปลี่ยนเป็นสัญญาณขาออก (16 - BIT OUTPUT) และส่งเข้าไปยังแรมแสดงผลในหน่วยเอ็มพียู (MPU) ต่อไป

2.1.8. ส่วนแปลงสัญญาณอนุกรม (SERIAL DATA AQUISITION)

เป็นส่วนที่ประกอบไปด้วย ตัวกำเนิดสัญญาณโบด์เรต (BAUD RATE GENERATOR) , USART (PROGRAMMABLE COMMUNICATION INTERFACE) และดาต้าบัสบัฟเฟอร์ ส่วนนี้จะรับสัญญาณเข้ามาแบบอนุกรม แล้วแปลงเป็นสัญญาณขาออกแบบขนานและวงจรนี้จะเป็นตัวจ่ายสัญญาณนาฬิกาให้กับอุปกรณ์ภายในและภายนอก โดยสัญญาณที่ส่งออกจากส่วนนี้จะผ่านเข้าสู่ดาต้าบัสเอ็มพียูต่อไป

2.1.9. ส่วนคีบอร์ดและเอ็นพียู (KEYBOARD AND MPU)

- คีบอร์ด ทำหน้าที่เป็นตัวส่งสัญญาณ 2 ข้อมูลไปยังเอ็มพียู โดยผ่านแลตช์และเกตเพื่อจัดเวลาให้เหมาะสม

- เอ็มพียู ทำหน้าที่เป็นหน่วยประมวลผลต่างๆ ได้แก่ ไมโครโปรเซสเซอร์ , แรม , รอม , วงจรถอดรหัสให้กับตำแหน่งหน่วยความจำ

2.1.10. ส่วนควบคุมการแสดงผล (DISPLAY CONTROL)

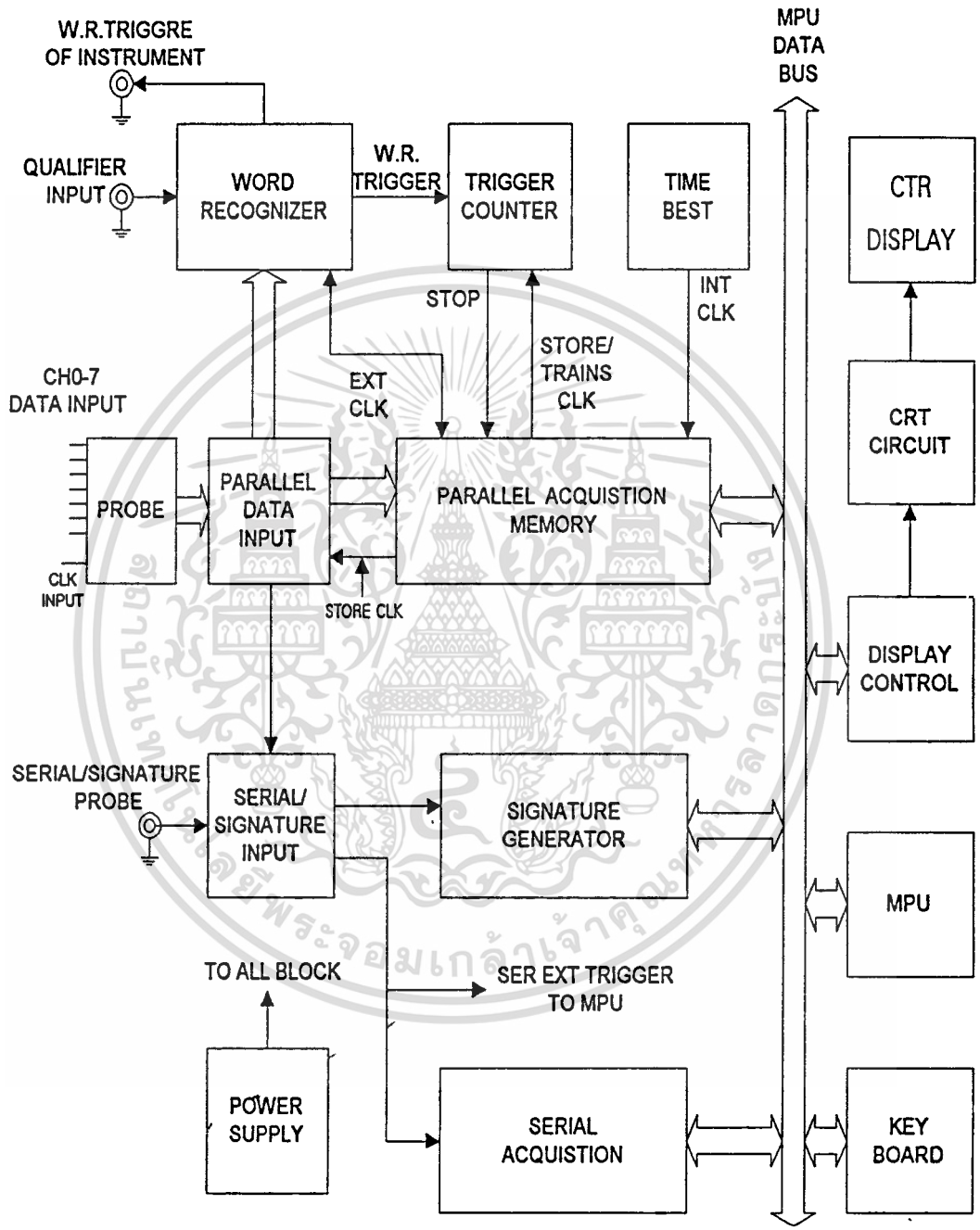
เป็นวงจรที่ประกอบขึ้นด้วยหน่วยความจำแรม ซึ่งเป็นตัวเก็บข้อมูลที่จะแสดงผล ซึ้งได้รับการ ประมวลผลจากเอ็มพียู เพื่อจะส่งไปยังจอภาพต่อไป

2.1.11. ส่วนจอภาพ (DISPLAY)

เป็นหน่วยที่ใช้แสดงผลของสัญญาณตรรกที่ได้จากการวัด

2.1.12. ส่วนแหล่งจ่ายกำลัง (POWER SUPPLY)

เป็นหน่วยที่ทำหน้าที่จ่ายกำลัง หรือจ่ายกระแสไฟตรงให้กับวงจรอิเลคทรอนิกส์ทุกๆ ส่วนนั้นเอง



รูปที่ 2.1 โครงสร้างพื้นฐานของเครื่องวิเคราะห์สัญญาณตรรก

2.2 แนวการออกแบบ

การออกแบบจะอาศัยแนวความคิดและทฤษฎีจากหัวข้อที่ 2.1 เพื่อกำหนดรูปแบบของเครื่องวิเคราะห์สัญญาณตรรกใหม่ดังนี้

2.2.1. เครื่องวิเคราะห์สัญญาณตรรกที่สร้างขึ้น จะต้องติดต่อใช้งานร่วมกับเครื่องคอมพิวเตอร์ส่วนบุคคลเพื่ออาศัยจอภาพและจานแม่เหล็ก ทำการแสดงผลของสัญญาณตรรกที่วัดได้และทำการเก็บข้อมูลที่วัดได้เพื่อนำมาใช้ในภายหลัง

2.2.2. เครื่องวิเคราะห์สัญญาณตรรกที่สร้างขึ้น จะรับหรือส่งข้อมูลระหว่างตัวมันกับเครื่องคอมพิวเตอร์ส่วนบุคคลผ่านทางพอร์ทอนุกรม (SERIES PORT) เท่านั้น ทั้งนี้เนื่องจากเครื่องคอมพิว เตอร์ส่วนบุคคลทั้วๆ ไปหลายยี่ห้อจะมีพอร์ทอนุกรม ถ้าสร้างเครื่องวิเคราะห์สัญญาณตรรกแล้วติดตั้งลงไปในเครื่องคอมคอมพิวเตอร์เลย จะทำให้ผูกติดกับเครื่องคอมพิว-
-เตอร์แต่ละรุ่นแต่ละยี่ห้อมากเกินไป ทำให้ไม่สะดวกในการใช้งาน

2.2.3 ช่องสัญญาณที่ทำการวัดมี 40 ช่องสัญญาณ ซึ่งสามารถจะรองรับเครื่องคอมพิวเตอร์ส่วนบุคคลในปัจจุบัน ซึ่งในปัจจุบันเครื่องคอมพิวเตอร์ส่วนบุคคลมีการส่งข้อมูลทีละ 32 บิท

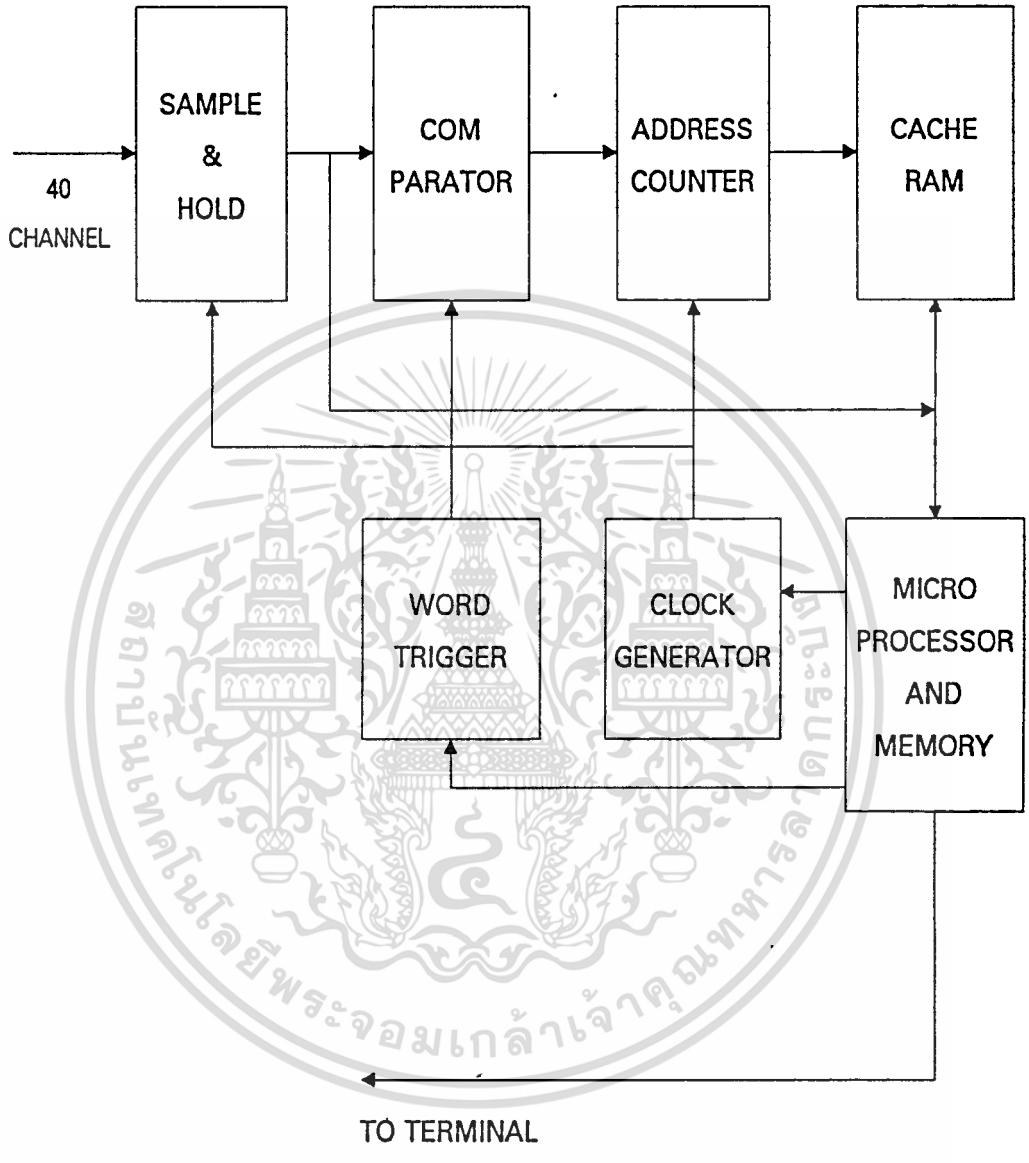
2.2.4 อัตราการสุ่มสัญญาณในการวัด เมื่อใช้สัญญาณภายในเครื่อง (INTERNAL CLOCK) จะต้องมีความถี่สูงกว่าความถี่ของเครื่องคอมพิวเตอร์บุคคลที่จะทำการวัด และสามารถทำการลดความถี่ที่จะใช้เป็นสัญญาณสุ่มตัวอย่างได้พอสมควร ในที่นี้จึงเลือกใช้การสุ่มข้อมูลในอัตรา 50 เมกกะเฮิทซ์ และสามารถจะลดทอนด้วยอัตรา 5-2-1 เช่น 50 , 20 , 10 , 5 เมก-
-กะเฮิทซ์เป็นต้น

2.2.5 สามารถกำหนดสภาวะของสัญญาณตรรก (LOGIC) เพื่อให้เริ่มเก็บข้อมูลได้ทั้ง 40 ช่องสัญญาณ (DOUBLE WORD TRIGGER) ทำให้ผู้ใช้แยกการใช้งานของช่องสัญญาณด้านขาเข้าไปจับ สัญญาณจากบัสแอดเดรสหรือบัสข้อมูลได้ตามต้องการ

จากขอกำหนดดังกล่าวข้างต้นสามารถเขียนผังการทำงานได้ดังรูปที่ 2.2 การทำงานของผังเครื่องวัดสัญญาณตรรกมีดังนี้

สัญญาณตรรกที่รับเข้ามาทางขาเข้า (0-40 ช่องสัญญาณ) จะผ่าน **ส่วนสุ่มสัญญาณและค้างข้อมูล** (SAMPLE AND HOLD) ส่วนนี้จะทำหน้าที่สุ่มสัญญาณที่เข้ามาด้วยความถี่กว่าความถี่ที่ต้องการวัดแล้วทำการค้างสัญญาณไว้ชั่วขณะ จากนั้นก็จะส่งสัญญาณไปยัง **ส่วนเปรียบเทียบ** (COMPARATOR) เพื่อทำการเปรียบเทียบระหว่างสัญญาณที่ต้องการวัดกับสัญญาณจาก**ส่วนกระตุ้น** (WORD TRIGGER) เมื่อข้อมูลตรงกันกับสัญญาณกระตุ้นที่ผู้ใช้กำหนด ก็จะทำการส่งสัญญาณไปให้**ส่วนนับตำแหน่ง** (ADDRESS COUNTER) ส่วนนี้จะทำการนับตำแหน่ง โดยการใช้สัญญาณนาฬิกามาจาก**ส่วนสร้างสัญญาณนาฬิกา** (CLOCK GENERATOR) ตำแหน่งที่ได้จาก

ส่วนนับตำแหน่งจะถูกใช้เป็นตำแหน่งในการเก็บข้อมูลให้กับส่วนหน่วยความจำแบบแคชแรม (CACHE RAM) หน่วยความ



รูปที่ 2.2 แผนภาพของเครื่องวิเคราะสัญญาณตรรก

จำจะทำการเก็บสัญญาณข้อมูลที่ได้จากส่วนสุ่มสัญญาณและค้างข้อมูล และในขณะนี้จะเห็นว่าสัญญาณข้อมูลที่ผู้ใช้ต้องการทราบจะถูกเก็บไว้ในหน่วยความจำแบบแคชแรม เมื่อหน่วยความจำแบบแคชแรมทำการเก็บข้อมูลจนเต็มแล้ว ก็จะทำการส่งข้อมูลเข้าไปเก็บไว้ในหน่วยความจำ (แรม) ภายในส่วนไมโครโปรเซสเซอร์และแรม (MICROPROCESSOR AND MEMORY) ส่วน

ไมโครโปรเซสเซอร์จะทำหน้าที่ในการควบคุมระบบการทำงานที่ผ่านมาทั้งหมด เมื่อถึงตอนนี้สัญญาณที่มารอ อยู่ที่ส่วนนี้ก็จะถูกส่งออกไปยังคอมพิวเตอร์ส่วนบุคคล เพื่อที่จะทำการเก็บลงจานแม่เหล็ก (HARD DISK) และทำการแสดงผลของสัญญาณตรรกที่วัดได้

2.3. โครสร้างและหลักการทำงานของไมโครโปรเซสเซอร์แบบชิพเดียวตระกูล Z80180

จากแผนผังในรูปที่ 2.2 ไมโครโปรเซสเซอร์เป็นเสมือนหัวใจของระบบเครื่องวิเคราะห์สัญญาณตรรก คือไมโครโปรเซสเซอร์ททำหน้าที่เป็นตัวควบคุมระบบการทำงานทั้งหมด ในที่นี้ใช้ไมโครโปรเซสเซอร์ Z80180 ของบริษัท Zilog เนื่องจากความก้าวหน้าทางเทคโลยี CMOS และหน่วยการทำงานซึ่งทำงานด้วยไมโครโค้ด (MICRO-CODE) ไมโครโปรเซสเซอร์มีขนาด 8 บิท ซึ่งทำให้เกิดเป็นข้อได้เปรียบ คือมีประสิทธิภาพการทำงานสูง ค่าใช้จ่ายต่ำและกินไฟขณะทำงานน้อย ขณะที่ซอฟแวร์ก็ยังสามารถใช้ร่วมกับซอฟแวร์ของไมโครโปรเซสเซอร์ของผู้ผลิตอื่นๆ ได้

การทำงานของไมโครโปรเซสเซอร์ตระกูลนี้ ได้พัฒนาให้สามารถทำงานที่ความถี่สูงได้ และในระบบไปป์ไลน์นึ่ง (Pipelining) ขณะที่ชุดคำสั่งก็มีเพิ่มมากขึ้น และยังมีส่วนจัดการหน่วยความจำ (Memory Management Unit หรือ MMU) ซึ่งมีที่วางทั้งแบบ 1 เมกกะไบท์ และแบบ 512 กิโลไบท์

ค่าใช้จ่ายของระบบลดลงเนื่องจากว่า ฟังก์การทำงานหลักที่สำคัญพร้อมทั้ง MMU ได้ถูกรวมไว้อยู่ในชิพแล้ว เช่น มีตัวควบคุมการทำ DMA 2 แชนแนล (Direct Memory Access Controller หรือ DMAC) , ตัวสร้างสภาวะรอ (Wait state generrator) , การทำรีเพรชไดนามิกแรม (Dynamic Ram Refresh) , มีส่วนอินเตอร์ในการสื่อสารข้อมูลอนุกรมแบบอะซิงค์โครนัส 2 แชนแนล (Asynchronous Series Communication Interface หรือ ASCI) , มีสัญญาณนาฬิกาสำหรับพอร์ท I/O แบบอนุกรม (Clock Series I/O Port หรือ CS I/O) , มีตัวนับเวลาแบบรีโหลดซึ่งสามารถโปรแกรมได้ 2 แชนแนล (Programmable Reload Timer หรือ PRT) , ตัวควบคุมในการอินเทอร์รัพ 12 ชนิด และการอินเตอร์ เฟสกับบัส 2 แบบ์ คือแบบ 80xx และ 68xx ในโหมดการทำงานแบบกินไฟน้อย (Low Power Consumtion) เราสามารถใช้ซอฟแวร์ควบคุมให้ CPU ทำงานได้โดยกินไฟน้อยมาก

* Z80180 (Z180) จะสามารถใช้งานร่วมกับ HD64180 ซึ่งผลิตและจำหน่ายโดยบริษัท ฮิตาชิ ได้ *

ส่วนสำคัญของไมโครโปรเซสเซอร์ HD64180

- ความถี่ในการทำงานสูงถึง 8 MHz
- ส่วน MMU ซึ่งมีหน่วยความจำ 2 แบบ คือ 1 เมกกะไบท์ หรือ 512 เมกกะไบท์
- ส่วน DMAC 2 แชนแนล ซึ่งควบตุมการถ่ายโอน (Tranfer) ระหว่างหน่วยความจำกับหน่วยความจำ , หน่วยความจำกับ I/O , หน่วยความจำกับหน่วยความจำแมบป์ (Memory

Mapped)

- มีขา WAIT และตัสร้างสภาวะรอ สำหรับอุปกรณ์อินเตอร์เฟสที่ทำงานช้า
- สามารถทำการโปรแกรมการรีเฟรสไดนามิกแรม (Refresh DRAM)
- มีส่วน อินเตอร์เฟสสำหรับการสื่อสารข้อมูล แบบอนุกรมแบบอะซิงค์โครนัส 2 ช่องสัญญาณ ซึ่งสามารถโปรแกรมอัตราการส่งข้อมูล (Buad Rate) และสัญญาณแฮนด์เชคกิ้ง (Hand--shaking)
- มีส่วนสัญญาณนาฬิกาสำหรับพอร์ท I/O แบบอนุกรม ซึ่งทำงานด้วยความเร็วสูง (400 กิโลบิทต่อวินาที ที่สัญญาณนาฬิกาของระบบ 8 MHz)
- มีส่วนตั้งเวลา (Timer) 16 บิทรีโหลดซึ่งสามารถโปรแกรมได้
- ส่วนควบคุมการอินเทอร์จาก 12 แหล่ง โดยที่จะเกิดการอินเทอร์ภายใน 8 แหล่ง และ 4 แหล่งจะเกิดการอินเทอร์ภายนอก

บทที่ 3
โปรแกรมควบคุมการทำงาน

โฟร์ชาร์ทการทำงานของโปรแกรมควบคุม

โฟร์ชาร์ทการทำงานของโปรแกรมควบคุม เป็นการวางรูปแบบการทำงานเพื่อช่วยในการเขียนซอฟแวร์ ซึ่งโฟร์ชาร์ทการทำงานของโปรแกรมควบคุมแสดงดังรูปที่ 3.1 และมีรายละเอียดดังต่อไปนี้

1. การเริ่มต้น (START UP)
2. ทำการกำหนดค่าเริ่มต้น (INITIALIZE) ของ CPU ที่เราใช้งาน (HD64180)
3. ทำการกำหนดค่าเริ่มต้นให้กับ IC 8255 (PROGRAMMABLE PERIPHERAL INTERFACE)
4. ทำการกำหนดค่าเริ่มต้น (SET PROTOCAL) เป็นการกำหนดคุณสมบัติการของการติดต่อสื่อสารแบบอนุกรม เช่น กำหนดจำนวนบิทที่ใช้ (8 บิท) , บิทเริ่มต้น (START BIT) , บิทหยุด (STOP BIT) และคุณสมบัติอื่นๆ ที่จำเป็นต่อการสื่อสารแบบอนุกรม
5. เป็นการรับคำสั่งจากแป้นพิมพ์ (KEY BOARD) ว่าเราจะทำการใช้งาน หรือเลือกทำงานอะไรก่อน
6. รับคำสั่งกระตุ้น " T " (TRIGGER WORD) เมื่อได้รับคำสั่งกระตุ้นทางแป้นพิมพ์ ก็จะเข้าไปสู่โปรแกรมย่อย (SUBROUTINE) ในการเซ็ตค่าของสัญญาณกระตุ้น ซึ่งเป็นขอมูลเริ่มต้น ที่จะกระตุ้นและการกำหนดให้เครื่องวัดสัญญาณทางตรรกเริ่มตรวจจับและเก็บข้อมูลเพื่อทำงานต่อไป
7. รับคำสั่งการสุ่มสัญญาณ " S " (SAMPLING RATE) จะไปทำงานในโปรแกรมย่อยเพื่อทำการปรับค่าความถี่ในการสุ่มสัญญาณตรวจจับข้อมูล
8. รับคำสั่งวิเคราะห์สัญญาณ " Z " (LOGIC ANALYZER) เป็นโปรแกรมย่อยหลักของเครื่องวัดสัญญาณตรรภ ซึ่งจะทำการรับข้อมูลจากการตรวจจับ แล้วทำการแสดงผลของสัญญาณ ตรรกที่วัดได้ออกทางจอภาพ
9. รับคำสั่งอ่านค่าจากหน่วยความจำ " R " (READ DATE FORM CACHE RAM) ซึ่งเป็นคำสั่งในการอ่านข้อมูลจาก CACHE RAM ไปยัง SRAM เพื่อทำการแสดงผล

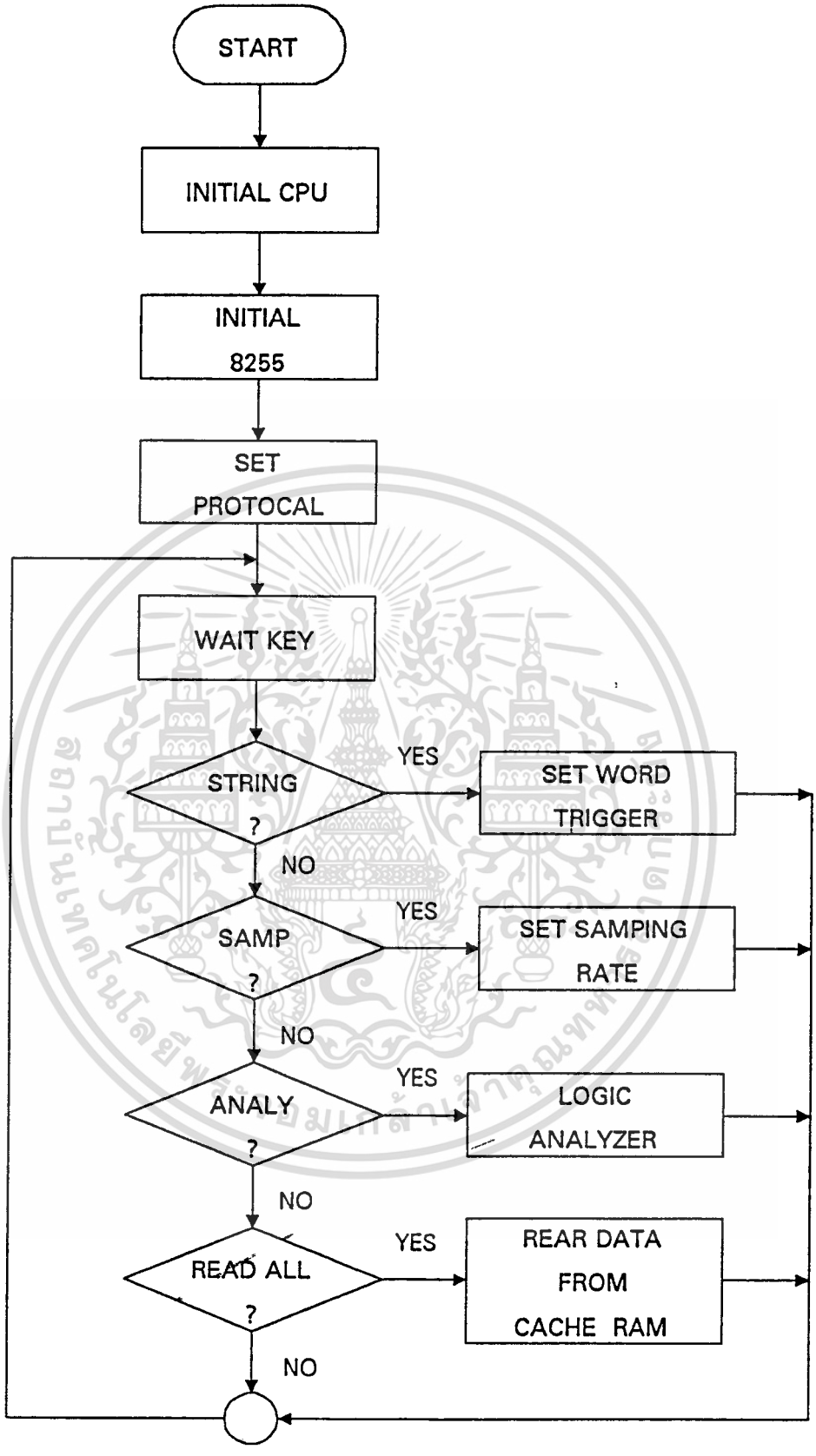
ภายในโปรแกรมการทำงานของเครื่องวัดสัญญาณตรรก จะมีโปรแกรมย่อยที่สำคัญอยู่หนึ่งโปรแกรม คือ โปรแกรมย่อยของ " Z " (LOGIC ANALYZER) ซึ่งเป็นโปรแกรมการนำดขเอาข้อมูลที่ตรวจจับได้ออกมาแสดงสภาวะตรรกทางจอภาพนั้นเอง ซึ่งการทำงานของโปรแกรมย่อยนี้แสดงดังรูปที่ 3.2 และมีการทำงานดังต่อไปนี้

1. เริ่มต้นทำการสำรองข้อมูลในหน่วยความจำ ที่จะเก็บข้อมูลสำหรับเครื่องวัดสัญญาณทางตรรก และทำการส่งคำสั่งเพื่อให้เครื่องวัดสัญญานตรรกส่งข้อมูลจาก CACHE RAM ไปยังแรมของระบบแล้วทำการส่งข้อมูลออกพอร์ทอนุกรม

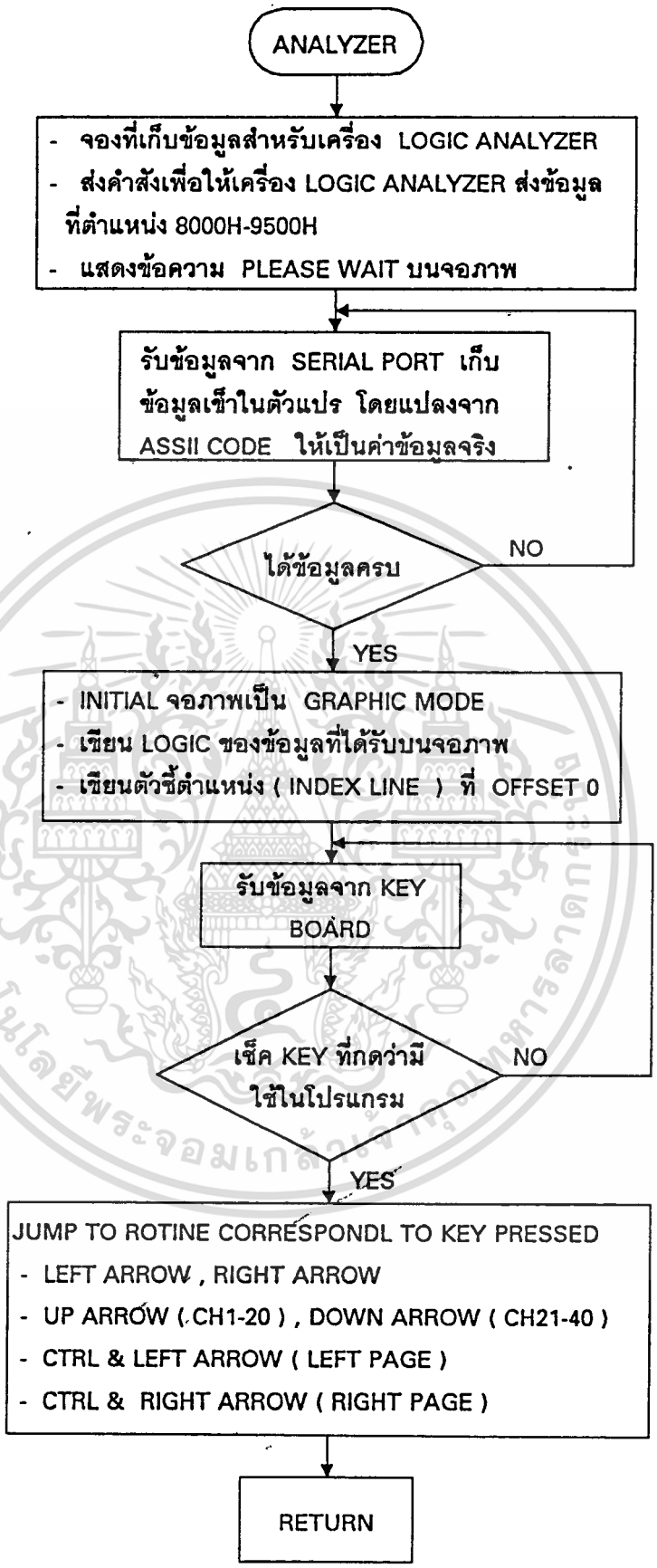
2. รับข้อมูลจาก SERIAL PORT (COM 1) เก็บข้อมูลเข้าไปในตัวแปรโดยจะแปลง ข้อมูลที่ได้รับให้อยู่ในรูป ASCII CODE ให้ได้ข้อมูลจริงที่จะแสดงให้ผู้ใช้ทราบ

3. ตรวจว่าได้รับข้อมูลมาครบหรือยัง ถ้ายังไม่ครบให้กลับไปรับใหม่แล้วทำการกำหนดค่าเริ่มต้นให้กับจอภาพเป็น GRAPHIC MODE ต่อจากนั้นเขียนข้อมูลที่ได้ให้อยู่ในสภาวะตรรก (PLUSE TRAIN) ขึ้นบนจอภาพ

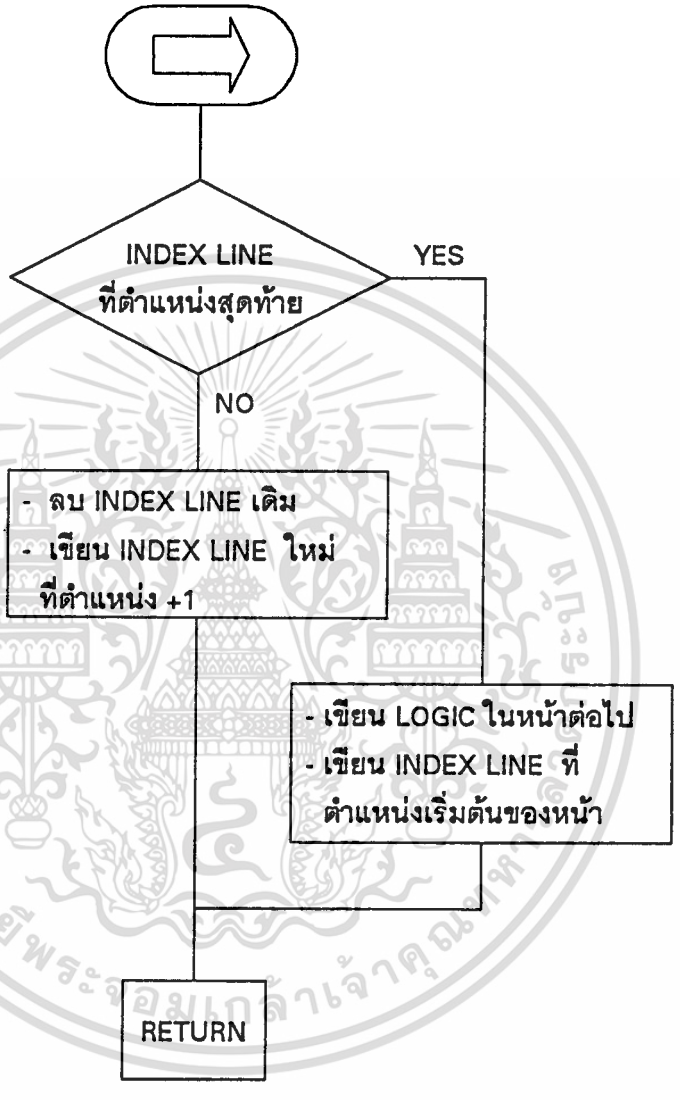
4. รอรับคำสั่งจากแป้นพิมพ์จากผู้ใช้เครื่อง ว่าต้องการตรวจสอบหรือใช้งานอะไร เมื่อได้รับคำสั่งจากแป้นพิมพ์ ก็จะไปทำยังโปรแกรมย่อยนั้น ๆ



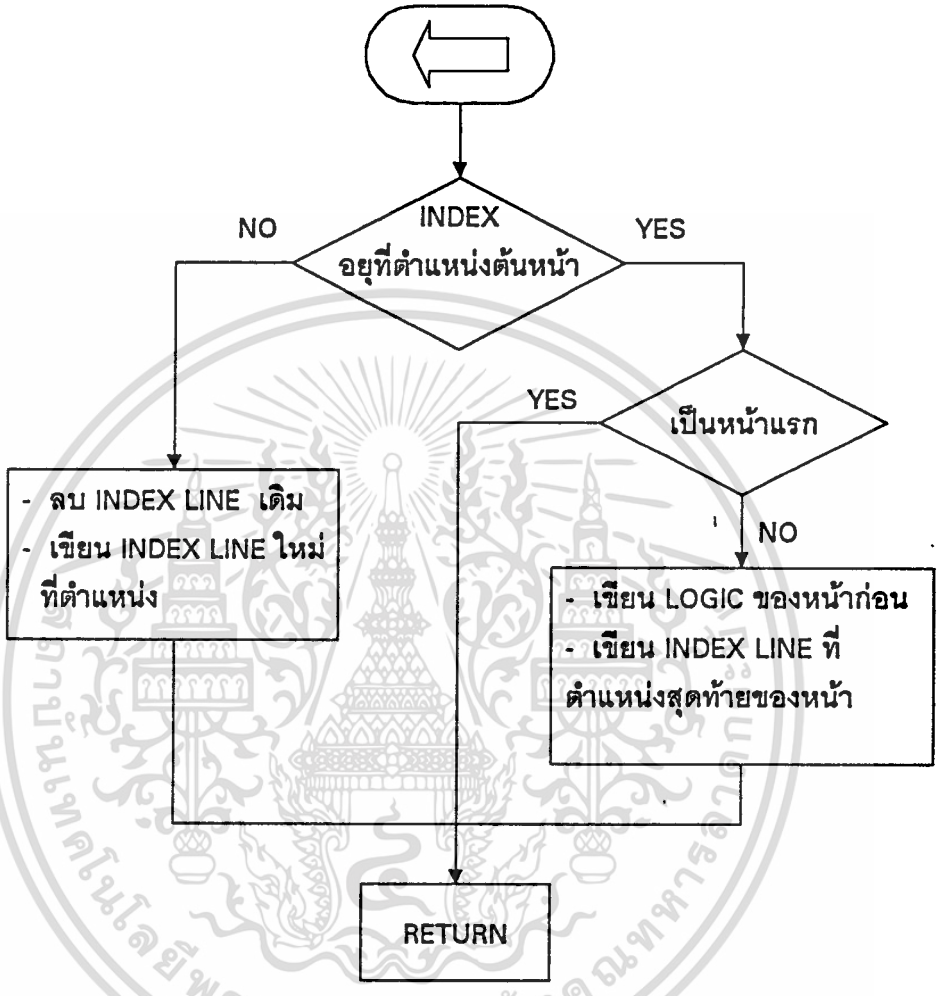
รูปที่ 3.1 แผนภาพการเขียนโปรแกรมควบคุม



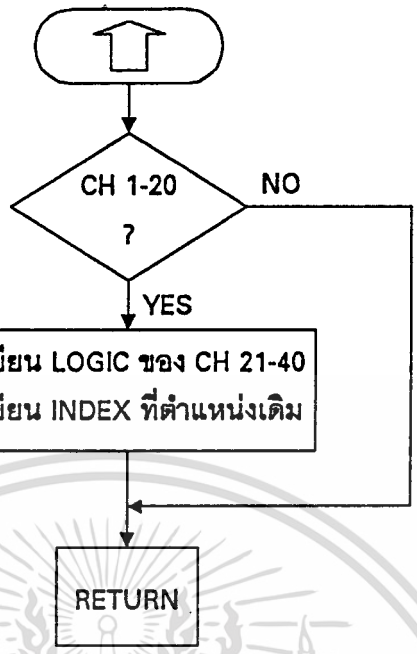
รูปที่ 3.2 แผนภาพการเขียนโปรแกรมตัวขับรูทีน (LOGIC ANALYZER)



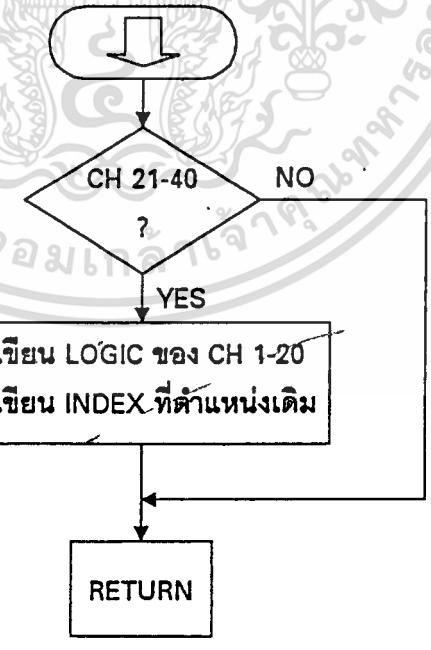
รูปที่ 3.3 แผนภาพการเขียนโปรแกรมขับรูทีน LEFT ARROW



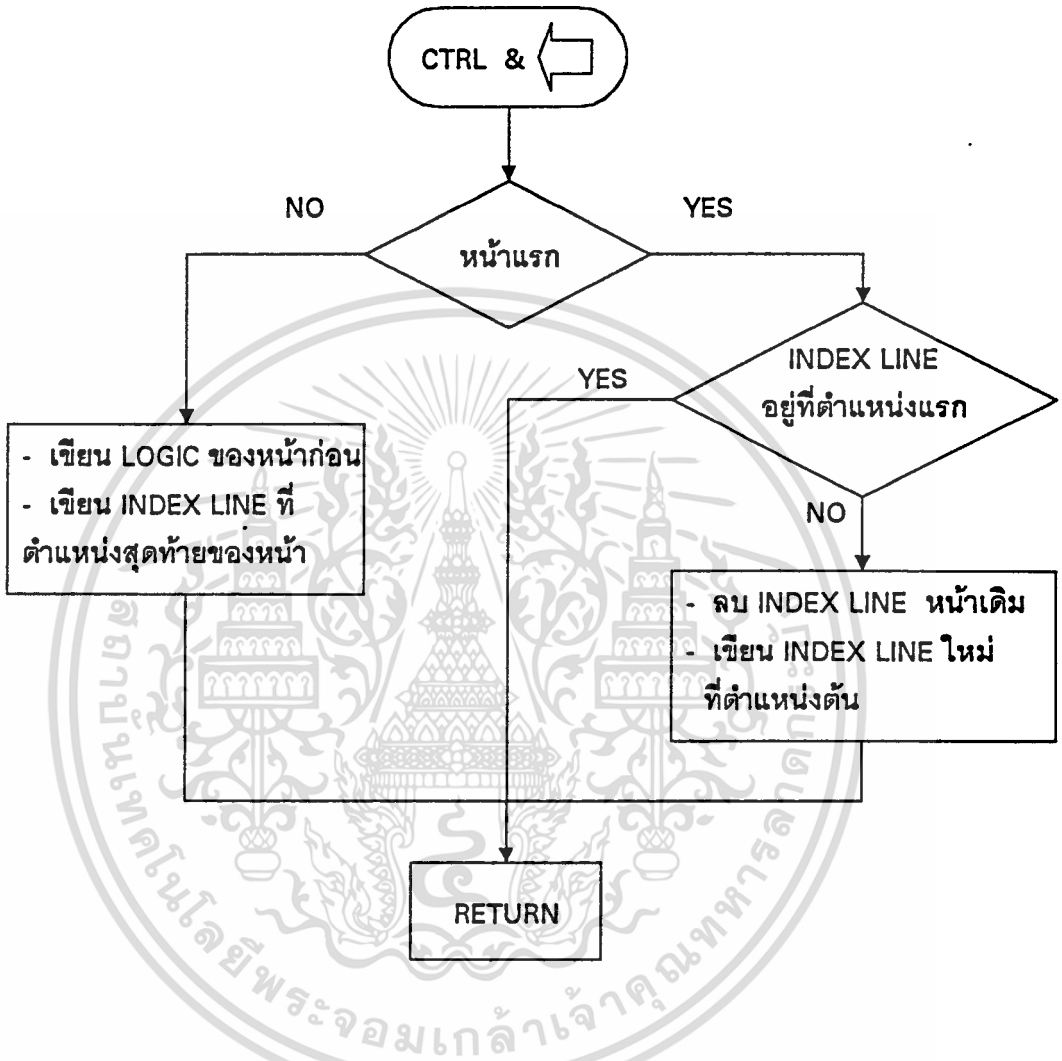
รูปที่ 3.4 แผนภาพการเขียนโปรแกรมซับรูทีน RIGHT ARROW



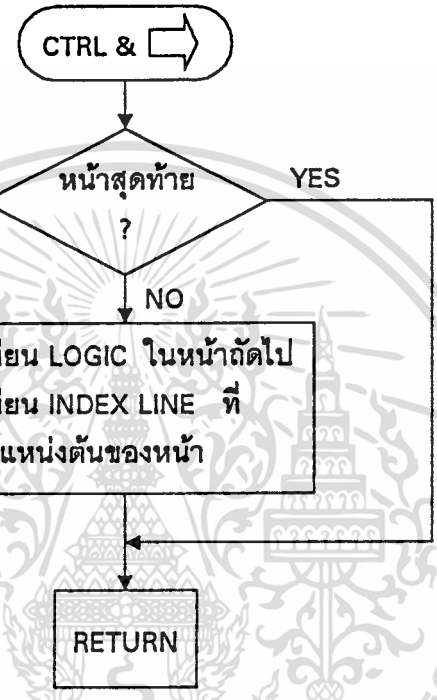
รูปที่ 3.4 แผนภาพการเขียนโปรแกรมขับรูทีน UP ARROW



รูปที่ 3.6 แผนภาพการเขียนโปรแกรมขับรูทีน DOWN ARROW



รูปที่ 3.7 แผนภาพการเขียนโปรแกรมขับรูทีน CTRL & LIFT ARROW



รูปที่ 3.8 แผนภาพการเขียนโปรแกรมขับรูทีน CTRL & RIGHT ARROW

บทที่ 4
วงจรเครื่องวิเคราะห์สัญญาณตรรก

4.1 วงจรรับอินพุทและเปรียบเทียบสัญญาณทริกเกอร์

เครื่องวิเคราะห์สัญญาณตรรกนี้ สามารถทำการวัดสัญญาณได้ 40 ช่องสัญญาณ . วงจรรับอินพุทและเปรียบเทียบสัญญาณทริกเกอร์ในส่วนนี้ จะมีทั้งหมด 5 ชุด ประกอบด้วย CH1-CH8 , CH9-CH16 ,CH17-CH24 ,CH25-CH32 และ CH33-CH40 ซึ่งแต่ละชุดมีการทำงานเหมือนกันดังนั้นเพื่อเป็นการสะดวกในการทำความเข้าใจจะทำการอธิบายเพียง 1 ชุด คือ CH1-CH8 ดังในรูปที่ 4.1

จากรูปที่ 4.1 สัญญาณที่ต้องการวัดจะถูกนำเข้ามาทางขั้วต่อ (DB9) ป้อนไปให้ ไอ.ซี U1(74F245) ทำหน้าที่เป็นบัฟเฟอร์(BUFFER) สัญญาณขาออกส่งไปที่ ไอ.ซี U2 (74F373) จะทำหน้าที่ค้างข้อมูลและสัญญาณกระตุ้น(SAMPLING & HOLD) ไอ.ซี U2 จะเป็นวงสุ่มสัญญาณ สัญญาณนาฬิกาที่ใช้ในการสุ่ม (SMCLK) ซึ่งจะต่อเข้าที่ขา 11 สัญญาณที่ขาออก สัญญาณที่ได้รับจากไอ.ซี. U2 จะปรากฏอยู่ที่ขา Q หรือไม่ขึ้นอยู่กับการควบคุมของสัญญาณ DOIR ซึ่งต่ออยู่กับขา 6 ของไอ.ซี. U64 (74F86) ในรูปที่ 4.5 ถ้าสัญญาณ DOIREN เป็น " 1" สัญญาณด้านขาออกไอ.ซี. U64 จะเป็น "0" ซึ่งเป็นการบังคับให้ไอ.ซี. U2 ส่งสัญญาณที่ค้างไว้ออกมา สัญญาณที่ส่งออกมาส่วนหนึ่งจะถูกส่งไปยังแคชแรมและอีกส่วนหนึ่งจะป้อนเป็นสัญญาณขาเข้าให้กับแนนเกตไอ.ซี. U3 , U4 (74F00) สัญญาณขาเข้าอีกด้านของแนนเกตได้มาจากไอ.ซี. U5 แนนเกตทั้งแปดตัวจะทำงานร่วมกับ ไอ.ซี. U7 ซึ่งทำหน้าที่ผลิตสัญญาณกระตุ้น (WORD TRIGGER) ตัวอย่างเช่น ต้องการตั้งให้เครื่องเริ่มทำการเก็บข้อมูลเมื่อมีข้อมูล 0AAH เข้ามา ทำโดยการส่งข้อมูล 0AAH มาค้างไว้ที่ขาออกไอ.ซี. U5 สัญญาณนี้จะถูกป้อนให้แนนเกต เมื่อสัญญาณที่ทำการวัดเป็น 0AAH จะทำให้สัญญาณขาออกแนนเกตเป็นดังนี้ ข้อมูล 55H จะถูกส่งไปยังขาเข้าของไอ.ซี. U7 (P0 - P7)

	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	
O/P U2	1	0	1	0	1	0	1	0	AAH
O/P U5	1	1	1	1	1	1	1	1	FFH
O/P U3,U4	0	1	0	1	0	1	0	1	55H
O/P U6	0	1	0	1	0	1	0	1	55H

ตารางที่ 4.2 การตั้งทริเกอร์สัญญาณที่มีบิทที่เราสนใจ

ข้อมูลของเวิร์ดทริก 0AAH จะถูกนำมาทำการคอมพลีเมนต์แบบหนึ่ง (1' COMPLEMENT) ผลที่ได้จะเป็น 55H ข้อมูลนี้จะถูกส่งไปค้างไว้ที่ขาออกไอ.ซี. U6 (74F374) หลังจากนั้นจะถูกป้อนให้ไอ.ซี. U7 (74F521) เพื่อเปรียบเทียบกับสัญญาณที่ได้จากแนนเกต เมื่อใดที่สัญญาณทั้งสองเท่ากัน ไอ.ซี. U7 จะส่งสัญญาณ "0" ออกมาที่ขา 19 แต่ถ้าสัญญาณทั้งสองไม่เท่ากันก็จะส่ง "1 " ออกมาแทน ในทำนองเดียวกัน ถ้าต้องการตั้งเวิร์ดทริกทั้ง 40 ช่องสัญญาณ ก็ทำการเอาสัญญาณที่ได้จากขา 19 ของไอ.ซี. U7, U14 ,U21 ,U28 , U35 ในแต่ละกลุ่มมาเข้าน็อร์เกต (ไอ.ซี. U36) ในที่นี้จะได้สัญญาณออกจากไอ.ซี. U36 เป็น "1" ซึ่งก็คือสัญญาณทริกเกอร์ของสัญญาณทั้ง 40 ช่องสัญญาณ ในช่วงเวลาต่อมาเมื่อมีสัญญาณช่องใดช่องหนึ่งเปลี่ยนแปลง จะทำให้สัญญาณขาออกของไอ.ซี. U36 เปลี่ยนจาก "1" เป็น "0" เมื่อมาทำการพิจารณาดูที่จุดนี้จึงเสมือนว่า เมื่อใดก็ตามที่ข้อมูลที่เข้ามามีค่าเท่ากับสัญญาณกระตุ้นที่ตั้งไว้ ที่จุดขาออกของไอ.ซี. U36 จะให้สัญญาณนาฬิกาออกมา 1 ลูกคลื่น ซึ่งสัญญาณนี้จะถูกใช้เป็นสัญญาณกระตุ้นต่อไป

	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	
O/P U2	1	0	1	0	D	D	D	D	AAH
O/P U5	1	1	1	1	0	0	0	0	FFH
O/P U3,U4	0	1	0	1	1	1	1	1	55H
O/P U6	0	1	0	1	1	1	1	1	55H

ตารางที่ 4.2 การตั้งทริเกอร์สัญญาณที่มีบิทที่เราไม่สนใจ

ในบางกรณี จากตารางที่ 4.2 การตั้งเวิร์ดทริกไม่จำเป็นต้องตั้งทุกบิท โดยทำการละบิทที่เราไม่สนใจ (DON'T CARE) บิทที่เราทำการระไว้โดยไม่สนใจ จะทำการส่งตรรก "0" ป้อนเข้าที่สัญญาณขาเข้าของแนนเกตทำให้สัญญาณตรรกที่ขาออกของแนนเกตเป็นหนึ่งเสมอ

ดังนั้นสัญญาณที่ขาออกของไอ.ซี. U6 จะต้องจ่าย "1" ออกมา เพื่อให้สอดคล้องกัน

4.2 วงจรทริกเกอร์

สัญญาณออกของไอ.ซี. U36 ในสภาวะปกติที่สัญญาณขาเข้าทั้งสองไม่เท่ากัน ส่งผลให้เกิดสัญญาณเคลียร์ขึ้นที่ขาไอ.ซี. U37 (74F163) เมื่อใดก็ตามที่สัญญาณเท่ากัน จะทำการส่งสัญญาณ "1" ออกไปที่ขา 1 ของไอ.ซี. U37 เพื่อทำการยกเลิกการเคลียร์ทำให้ไอ.ซี. U37 เริ่มทำการนับสัญญาณนาฬิกาที่มีความถี่ 50 แมกกะเฮิทซ์ของไอ.ซี. U37 จะทำงานร่วมกับ ไอ.ซี. U39 เพื่อทำหน้าที่ตรวจสอบว่าสัญญาณที่ได้จากการเปรียบเทียบข้อมูลของเวิร์ดทริกนั้นเท่ากันจริงหรือไม่

จากวงจรทริกเกอร์ในรูปที่ 4.6 เมื่อสัญญาณตรรกที่ขา CLR ของ ไอ.ซี. U37 ซึ่งถูกตั้งไว้ให้ทำหน้าที่เป็นตัวนับเลขฐานสิบหก (โดยทำการต่อขา A ,B ,C ,D ลงกราวด์) จะเริ่มทำการนับและให้สัญญาณออกที่ขา Qa ,Qb ,Qc และQd สัญญาณเหล่านี้จะนำไปเปรียบเทียบกับสัญญาณที่ได้จากไอ.ซี. U38 ซึ่งไอ.ซี. U38 จะเป็นตัวค้างการตั้งโปรแกรมการนับเพื่อตรวจสอบความยาวของสัญญาณ CLR เมื่อสัญญาณจากไอ.ซี. U37 และไอ.ซี. U38 เท่ากัน จะทำให้สัญญาณกระตุ้นขาออกเปลี่ยนจาก "0" เป็น "1" ช่วงเวลาต่อมาไอ.ซี. U37 ก็จะทำการนับต่อจึงทำให้สัญญาณเปลี่ยนกลับมาเป็น "0"

4.3 วงจรผลิตและหารความถี่

ดังที่ได้กล่าวมาแล้วเมื่อใช้อัตราการสุ่มสัญญาณนาฬิกาภายใน จะเลือกใช้การสุ่มสัญญาณในอัตรา 50 และ 20 แมกกะเฮิทซ์ และสามารถลดทอนลงด้วยอัตราการลดทอน 5 - 2 - 1 เช่น 50 ,20 ,10 ,5 ,2 , 1 แมกกะเฮิทซ์ เป็นส่วนที่ใช้สร้างสัญญาณนาฬิกาที่ใช้ในการสุ่มสัญญาณเข้ามาในระบบ ซึ่งสัญญาณนาฬิกาเหล่านั้นได้มาจากการหารความถี่ที่ได้จากก้อนผลึกสะท้อนแสง ซึ่งมีความถี่ 20 และ 50 แมกกะเฮิทซ์ และสามารถเลือกความถี่ในการสุ่มสัญญาณจากวงจรผลิตสัญญาณนาฬิกาภายนอก (POD CLK) การจะเลือกใช้สัญญาณนาฬิกาจากก้อนผลึกใดก้อนหนึ่งหรือสัญญาณนาฬิกาจากภายนอก

จากวงจรผลิตและหารความถี่ในรูปที่ 4.7 สามารถทำได้โดยการเลือกสัญญาณ S0-S6 ที่ให้กับ ไอ.ซี. U57 (74ACT153) ดังตารางที่ 4.3 โดยที่เอาท์พุทไอ.ซี. U61 ขาที่ 7 จะส่งสัญญาณไปให้ไอ ซี. U62 จะเป็นวงจรหารความถี่ด้วย 10 และ 2 ส่งไปให้ไอ.ซี. U58 ,U59 และU60 (74LS390) ซึ่งเป็นวงจรหารสิบ ตามลำดับ สัญญาณที่ได้จาก การเลือกและหารแล้วจะป้อนไปยังไอ.ซี. U56 เป็นตัวเปิดสัญญาณตามที่ต้องการออกไปซึ่งได้จากการเลือกสัญญาณ S3 , S4 และ S5 ที่ป้อนให้ไอ.ซี. U56 นั้นเอง

S7	S6	S5	S4	S3	S2	S1	S0	CLOCK (OUT)
X	0	1	1	1	0	1	1	10 Hz
X	0	1	1	1	1	1	0	20 Hz
X	0	1	1	0	0	0	1	50Hz
X	0	1	1	0	0	1	1	100Hz
X	0	1	1	0	1	1	0	200Hz
X	0	1	0	1	0	0	1	500Hz
X	0	1	0	1	0	1	1	1KHz
X	0	1	0	1	1	1	0	2KHz
X	0	1	0	0	0	0	1	5KHz
X	0	1	0	0	0	1	1	10KHz
X	0	1	0	0	1	1	0	20KHz
X	0	0	1	1	0	0	1	50KHz
X	0	0	1	1	0	1	1	100KHz
X	0	0	1	1	1	1	0	200KHz
X	0	0	1	0	0	0	1	500KHz
X	0	0	1	0	0	1	1	1MHz
X	0	0	1	0	1	1	0	2MHz
X	0	0	0	1	0	0	1	5MHz
X	0	0	0	1	0	1	1	10MHz
X	0	0	0	1	1	1	0	20MHz
X	0	0	0	0	0	0	0	50MHz
X	0	1	0	1	1	1	0	CLK IN -
X	0	0	0	1	1	1	0	CLK IN +

ตารางที่ 4.3 การเลือกความถี่ในการสุ่มสัญญาณ

4.4 วงจรผลิตสัญญาณ AQCLK และ WRC

สัญญาณนาฬิกาที่ใช้การสุ่มข้อมูลจากสายนำสัญญาณ CH1 - CH40 (STCLK , STCLK#) จะถูกป้อนมายังขาสัญญาณเข้าของแอนด์เกทและออร์เกท ดังวงจรในรูปที่ 4.6

ในสภาวะเริ่มต้นก่อนที่จะเกิดสัญญาณ TRIGGER ฟลิปฟลอป ไอ.ซี. 65 จะอยู่ในสภาวะปกติตรรกที่ขา Q จะเป็น 0 ทั้งคู่ ทำให้เกิดการปิดกันสัญญาณ STCLK และ SYCLK# ไม่ให้ผ่านแอนเกตและออร์เกตออกไป เมื่อเกิดสัญญาณ TRIGGER มากระตุ้นฟลิบฟลอปทำให้ตรรกของขา Q เปลี่ยนเป็น "1" และ Q# เปลี่ยนเป็น "0" ส่งผลให้เกิดสัญญาณนาฬิกาขึ้นที่จุดสัญญาณขาออกที่ออร์เกต (ไอ.ซี. U66) คือสัญญาณ AQCLK และสัญญาณ WR# สัญญาณนาฬิกา AGCLK จะถูกส่งไปเป็นสัญญาณนาฬิกาที่ขา 11 ซึ่งเป็นขา CP ของไอ.ซี. U2 นั้นคือการสุ่มสัญญาณของไอ.ซี. U2 จะเป็นไปตามสัญญาณนาฬิกา AGCLK ซึ่งสามารถเลือกได้ตามต้องการ

สัญญาณข้อมูลที่สุ่มได้จากไอ.ซี. U2 ส่วนหนึ่งจะถูกส่งผ่านไปเป็นสัญญาณขาเข้าของแคชแรมขนาด 16 บิท x 8 กิโลไบท์สองตัว บัสข้อมูลแต่ละเส้นของแคชแรม จะต้องรับสัญญาณที่สุ่มเข้ามาจากสายนำสัญญาณแต่ละช่องสัญญาณดังในรูปที่ 4.8

สัญญาณอีกส่วนหนึ่งในวงจรแผ่นนี้ คือ วงจรนับตำแหน่งแอดเดรสให้กับแคชแรม ได้แก่ ไอ.ซี. U50, 51, 52 และ53 (74ACT163) สัญญาณนาฬิกาของไอ.ซี. U50 ได้จากสัญญาณ AQCLK เช่นกันวงจรนับตำแหน่งแอดเดรสนี้ จะทำงานสอดคล้องกับการสุ่มสัญญาณของ U2, 9, 16, 23,30 และสัญญาณ WR# ของแคชแรมเสมอ

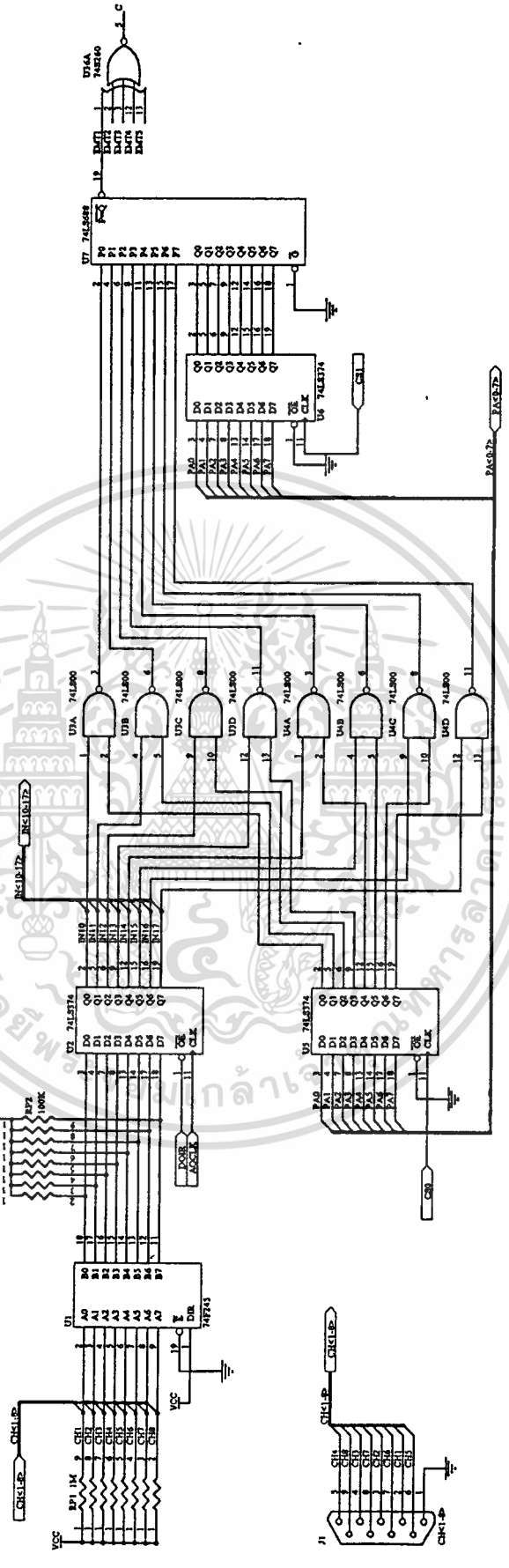
เมื่อวงจรนับตำแหน่งแอดเดรสจนครบ 8 กิโลไบท์ จะทำให้ตรรกในสายสัญญาณ CA0 - CA12 เป็น "1" หมดทุกเส้น สัญญาณส่วนนี้ส่วนหนึ่งจะถูกส่งไปยังบัสแอดเดรสของแคชแรม อีกส่วนหนึ่งต่อไปยังแอนด์ในรูปวงจรแผ่นที่ 4.8 เมื่อ CA0-CA12 เป็นตรรก "1" ทั้งหมด จะทำให้สัญญาณขาออกของแอนด์เกทไอ.ซี. U55 ขา 8 เป็น "1" ส่งผลให้เกิดการกระตุ้นฟลิบฟลอบไอ.ซี. U65 ให้เปลี่ยนสถานะ สัญญาณที่ขา Q จากเดิมตรรก "0" ไปเป็นตรรก "1" ซึ่งเป็นการปิดเกทไม่ยอมให้สัญญาณนาฬิกา STCLK ผ่านออกไปนั้น หมายความว่า สัญญาณ AQCLK จะไม่มีอีกต่อไปทำให้ส่วนสุ่มสัญญาณหยุดทำงาน และส่วนนับตำแหน่งแอดเดรสหยุดทำงานไปโดยอัตโนมัติ ในขณะนี้ข้อมูลที่สุ่มเข้ามาจะถูกเก็บเข้าไปไว้ในแคชแรมเรียบร้อยแล้ว พร้อมที่จะถูกอ่านออกมาเพื่อแสดงผลต่อไป

4.5 วงจรควบคุม

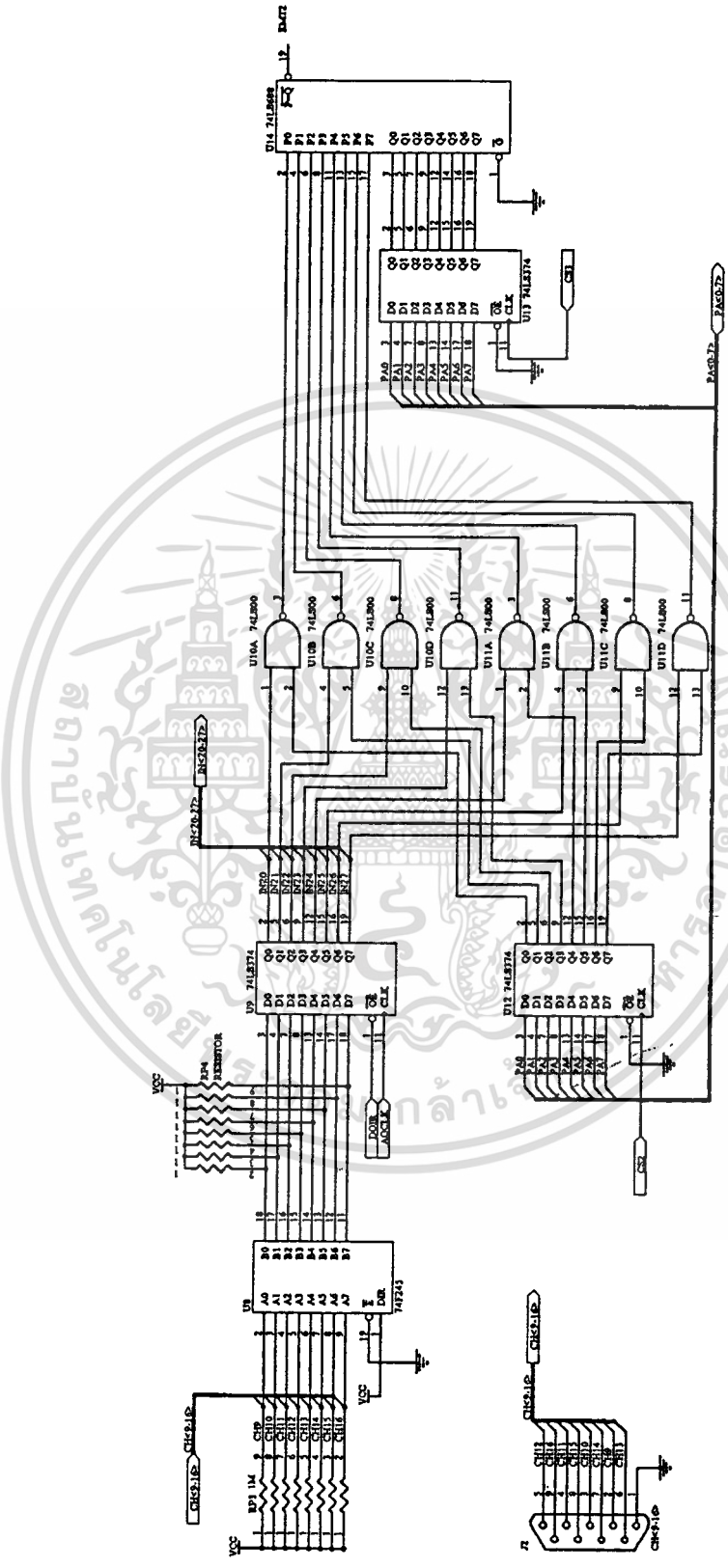
ส่วนควบคุมระบบทั้งหมดนี้ได้แก่ วงจรในรูปที่ 4.9 จะต่ออยู่กับบอร์ด JR180 บอร์ด JR180 ประกอบได้ด้วยไมโครโปรเซสเซอร์ Z80180 พอร์ทอนุกรมของ Z80180 จะต่อเข้ากับไอ.ซี.(MAX232) ซึ่งทำหน้าที่เปลี่ยนระดับสัญญาณจาก TTL ให้เป็นมาตราฐาน RS-232C เพื่อเชื่อมต่อข้อมูลเข้ากับเครื่องคอมพิวเตอร์ส่วนบุคคล หน่วยความจำแรมของระบบมี 32 กิโลไบท์ (62256) และ หน่วยจำรอมสำหรับเก็บโปรแกมจัดระบบ ไอ.ซี.PPI ,8255 ทำหน้าที่เป็นบัฟเฟอร์ระหว่างบัสของระบบที่กล่าว

มาแล้วข้างต้น กับบัสของระบบไมโครโปรเซสเซอร์ Z80180 โดยพอร์ท A และพอร์ท C จะถูกทำตัวเสมือนกับบัสข้อมูลของตรรกภายนอก โดยพอร์ท A ถือเป็นข้อมูลที่ส่งออกไปยังส่วนตรรก และพอร์ท B ถือเป็นข้อมูลที่รับเข้ามาจากส่วนตรรก สำหรับพอร์ท B0 ถึงพอร์ท B3 จะถูกนำมาถอดรหัสโดยใช้ไอ.ซี. U68 คือ CS0-CS14 และ พอร์ท B4,B5 คือ CS15,CS16 ตามลำดับ

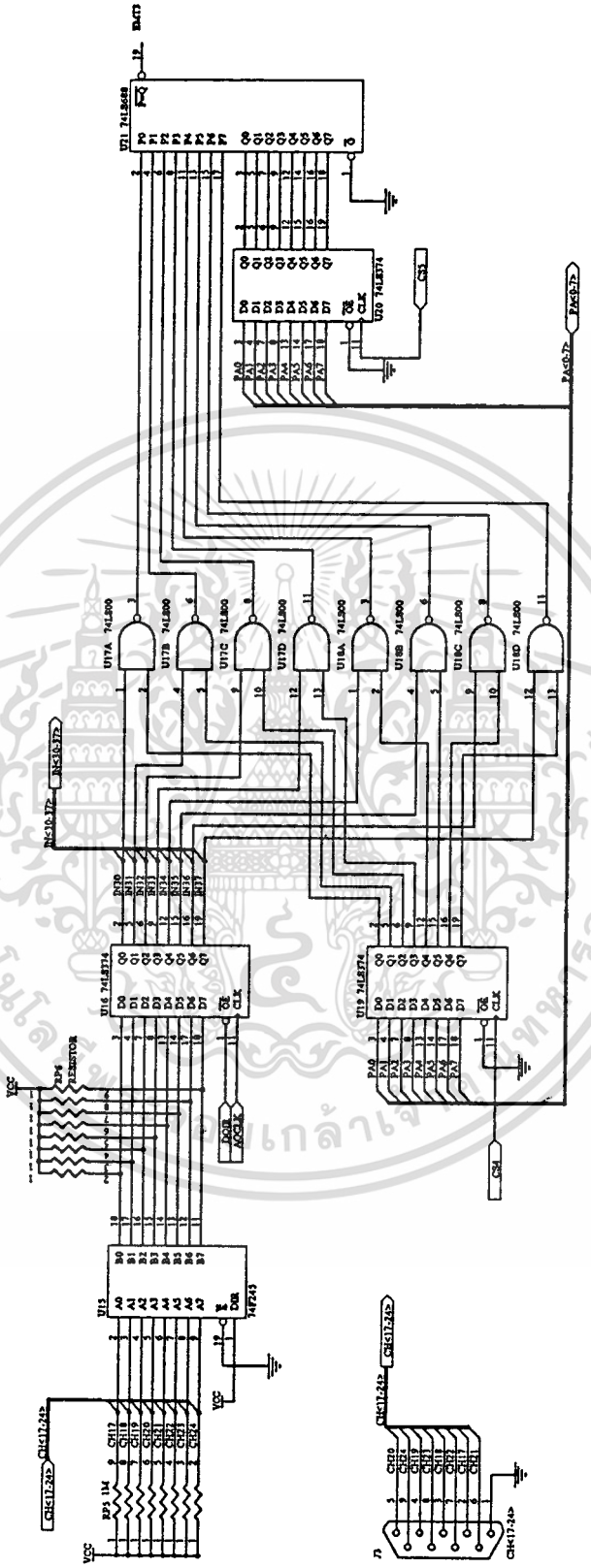
สัญญาณที่ออกมา(CS0-CS16)จะถูกนำไปควบคุมไอ.ซี. ที่ใช้ในการค้างข้อมูล และพอร์ท B6 จะใช้ควบคุมไอ.ซี. U69 ที่ทำหน้าที่ผลิต DOIREN ,STEP ,CNTCLR และ CNTCLK ที่ผลิตจากพอร์ท A การอ่านข้อมูลจากแรมแบบแคชของส่วนตรรกเข้ามายังแรมของไมโครโปรเซสเซอร์ ทำโดยส่งสัญญาณ STEP มา 1 ลูกคลื่น ส่วนนับตำแหน่งแอดเดรสจะเพิ่มค่า ทำให้แรมแบบแคชส่งข้อมูลมายังพอร์ท C ไมโครโปรเซสเซอร์จะรับข้อมูลเข้ามาเก็บไว้ในแรม กระทำเช่นนี้จนกระทั่งครบ 8 กิโลไบท์ ผลที่ได้คือ ข้อมูลจะถูกอ่านเข้ามาไว้ในแรมของระบบไมโครโปรเซสเซอร์ทั้งหมด CNTCLK เป็นสัญญาณที่ทำหน้าที่รีเซต ไอ.ซี. 65 ให้เอาท์พุท Q เป็น " 0 " CNTCLR เป็นสัญญาณที่ทำหน้าที่รีเซต ไอ.ซี. U50,51,52และ53 ให้เอาท์พุท Q เป็น " 0 " จากนี้บอร์ด JR180 จะส่งข้อมูล ที่เก็บไว้ที่ตำแหน่ง 8000H-9500H ผ่านทางพอร์ทอนุกรม RS-232C ให้เครื่องคอมพิวเตอร์ส่วนบุคคล ทำการแสดงผลต่อไป



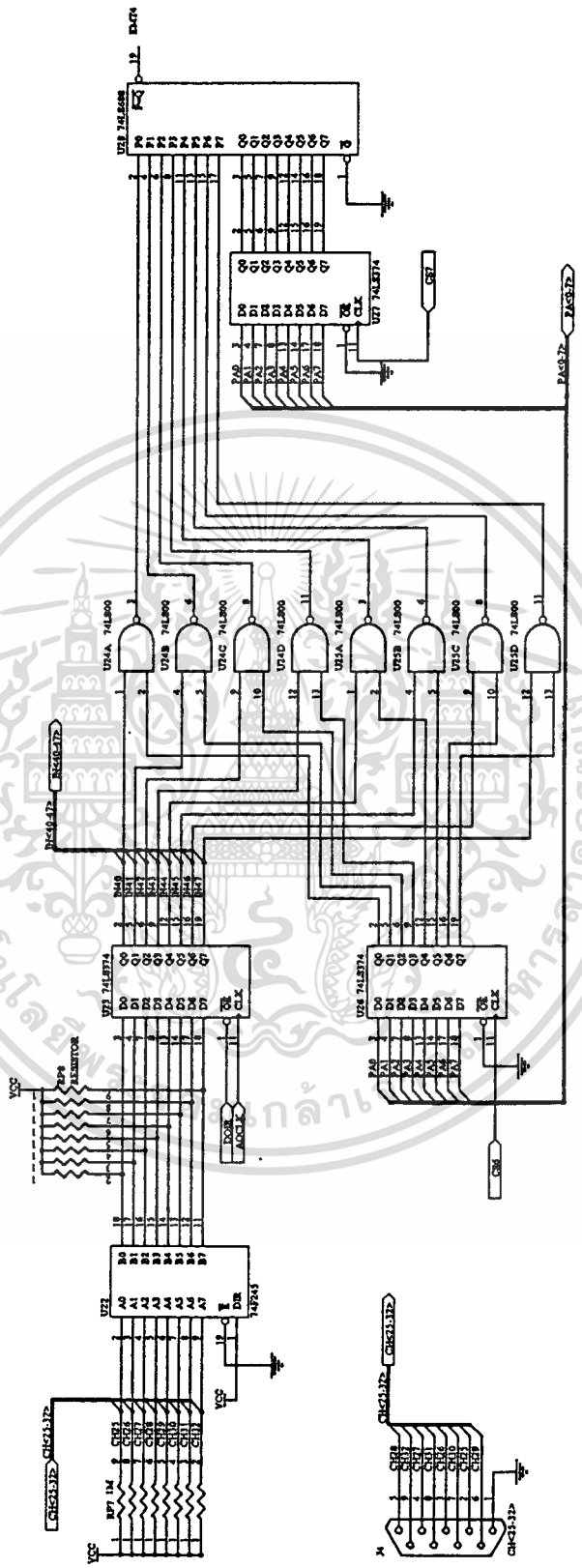
รูปที่ 4.1 วงจรรับสัญญาณอินพุท และ เปรียบเทียบสัญญาณทริกเกอร์ CH1-CH8



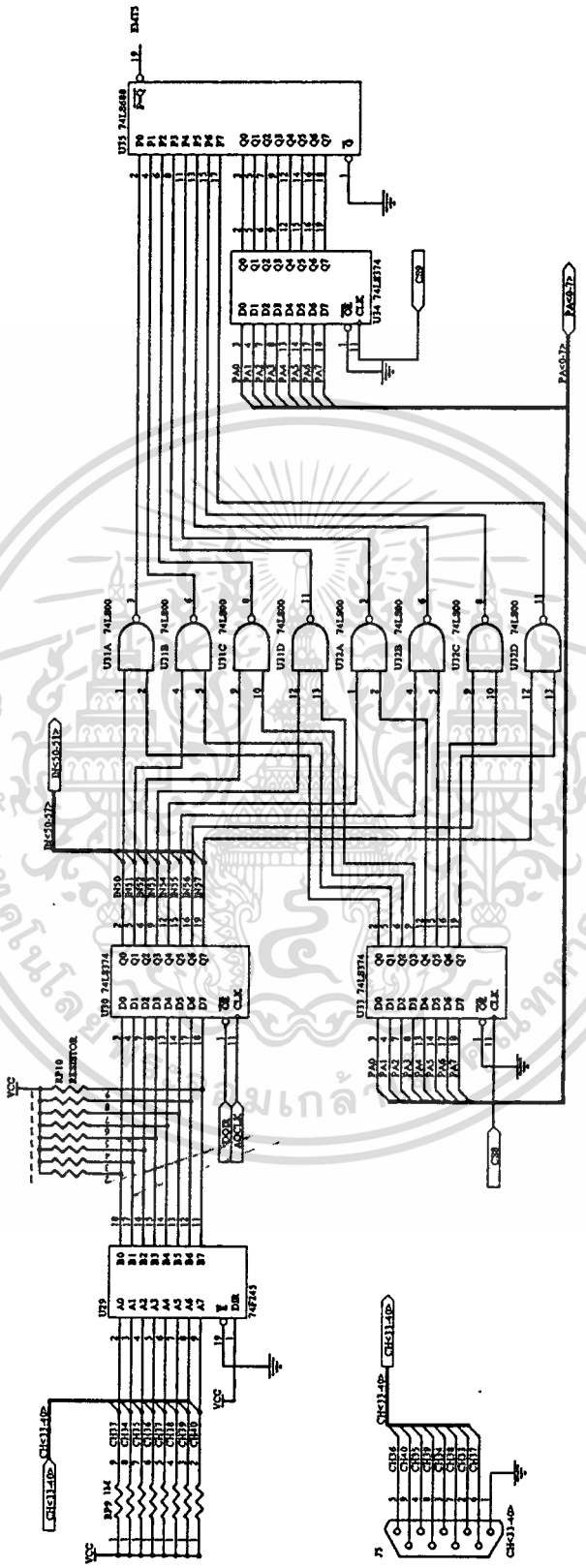
รูปที่ 4.2 วงจรรับสัญญาณอินพุท และ เปรียบเทียบสัญญาณดิจิตอล์ CH9-CH16



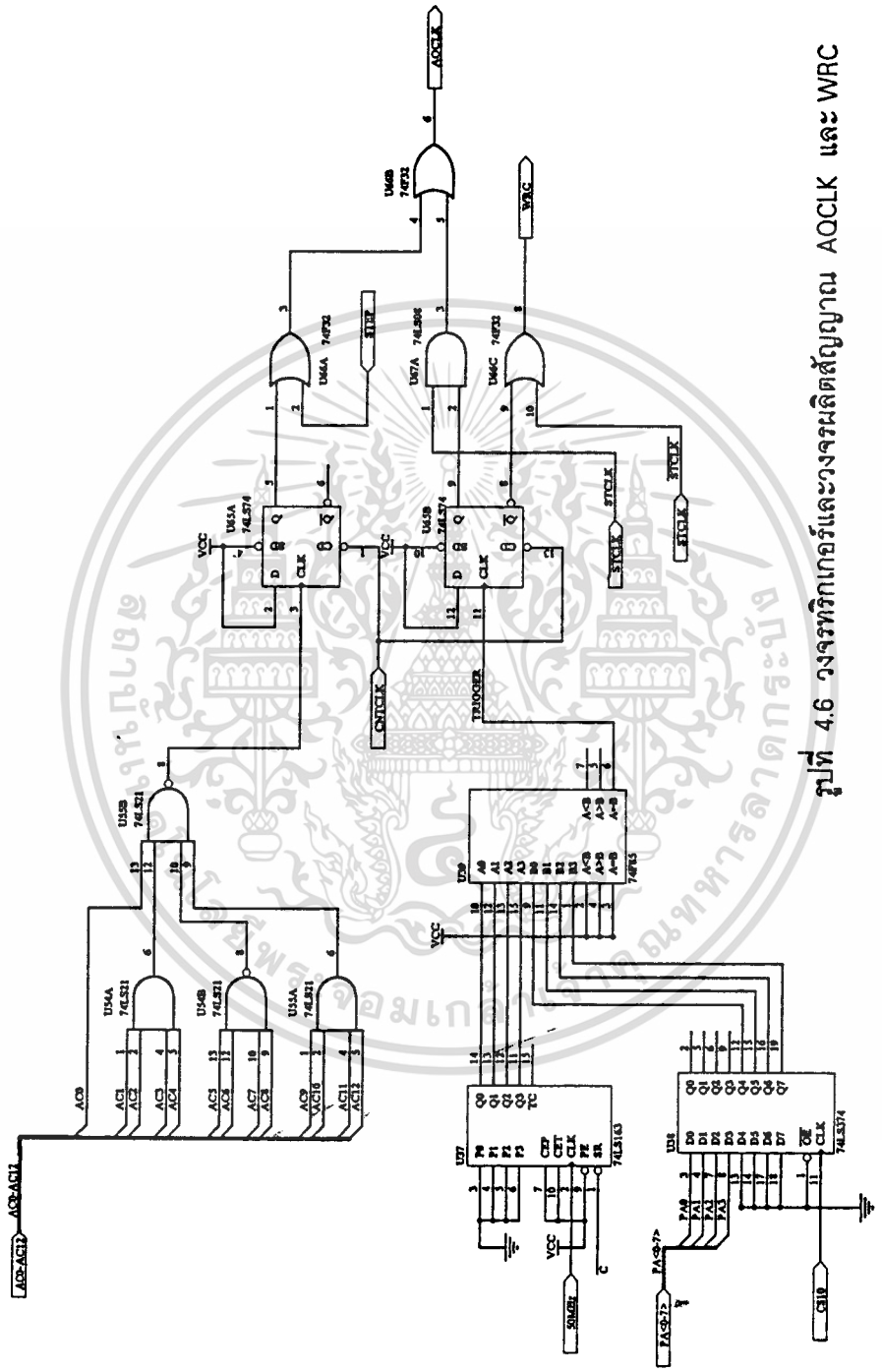
รูปที่ 4.3 วงจรรับสัญญาณอินพุท และ เปรียบเทียบสัญญาณทริกเกอร์ CH17-CH24



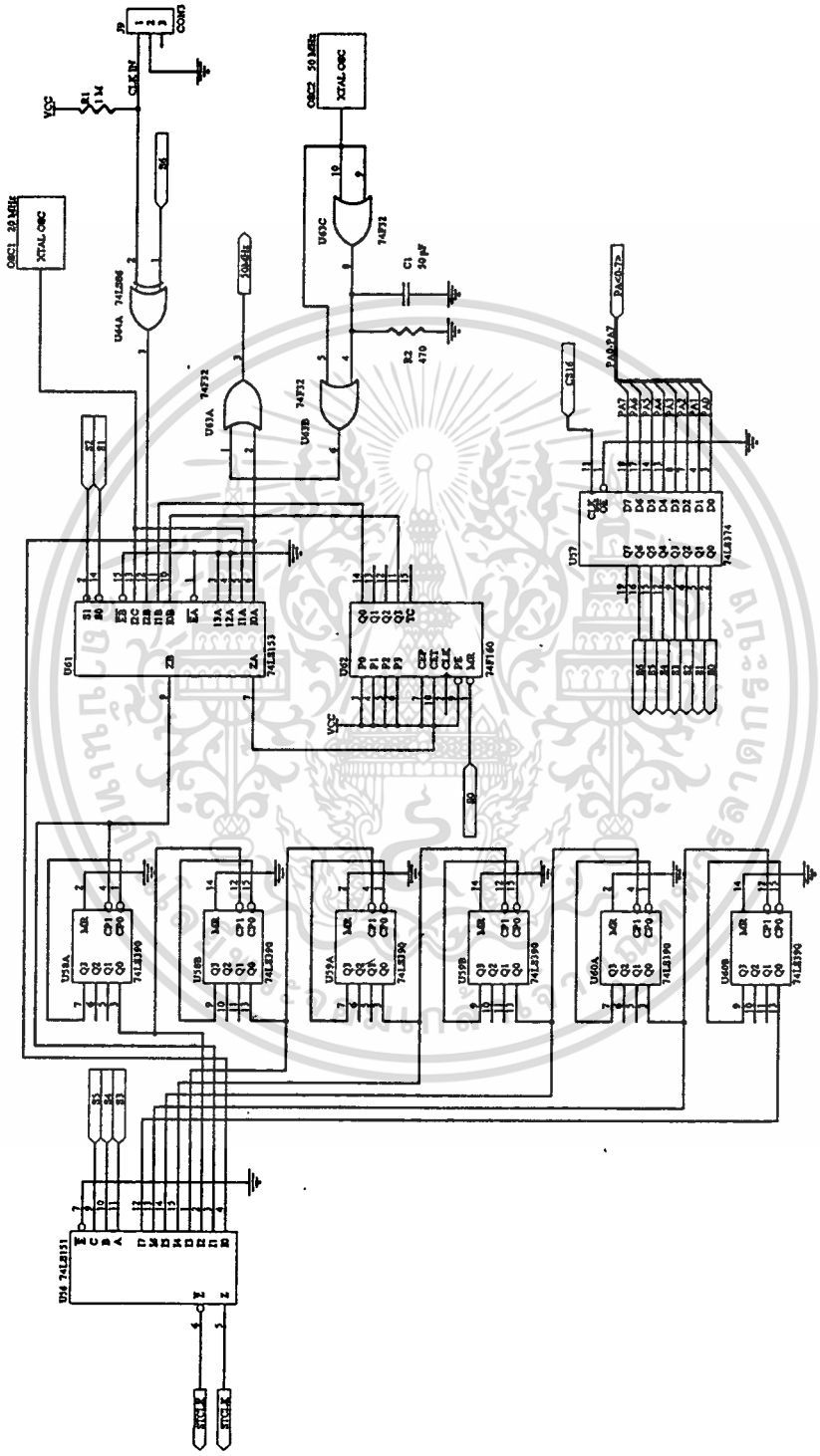
รูปที่ 4.4 วงจรรับสัญญาณอินพุท และ เปรียบเทียบสัญญาณทริกเกอร์ CH25-CH32



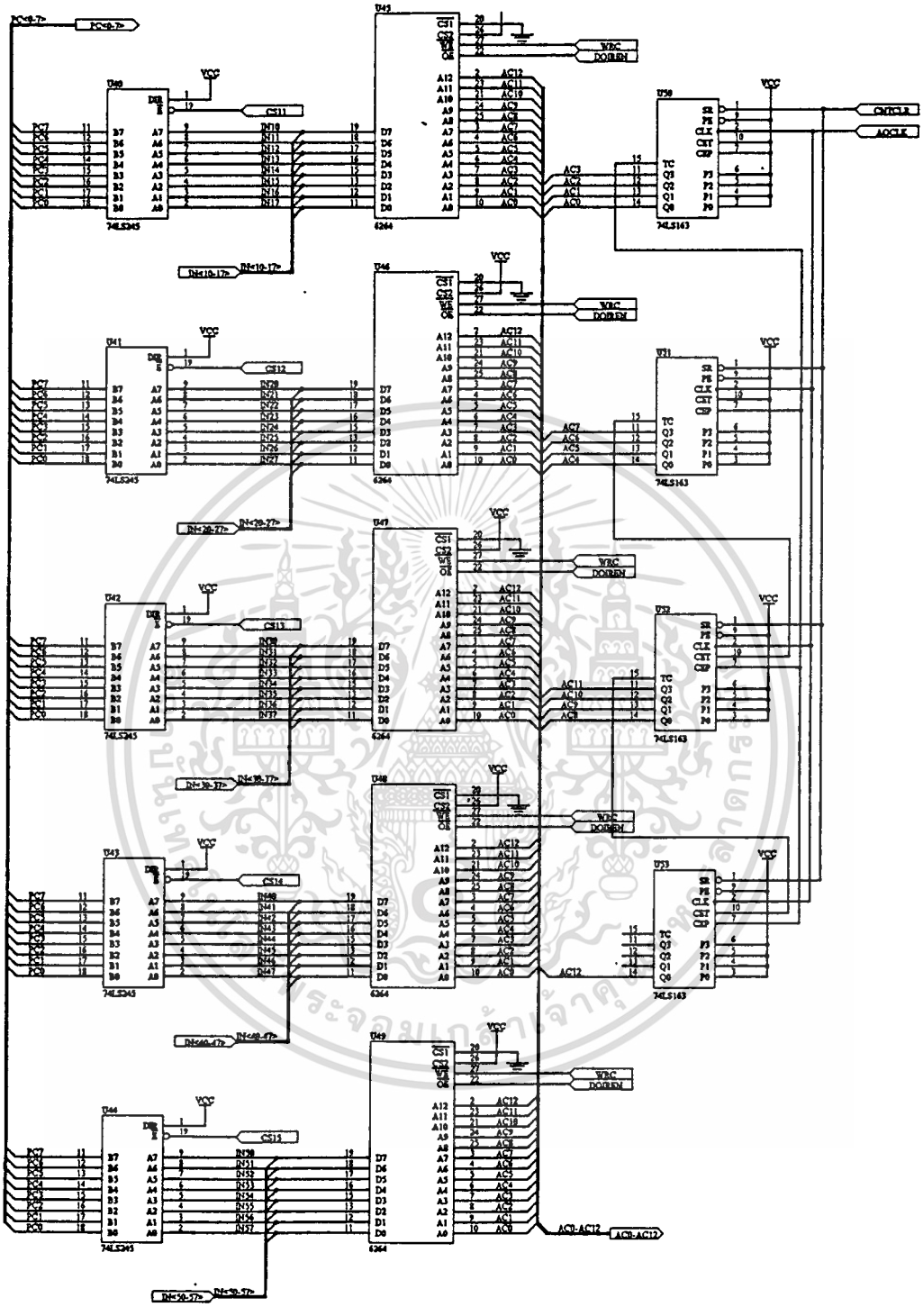
รูปที่ 4.5 วงจรรับสัญญาณอินพุท และ เปรียบเทียบสัญญาณดิจิทอล์ CH33-C40



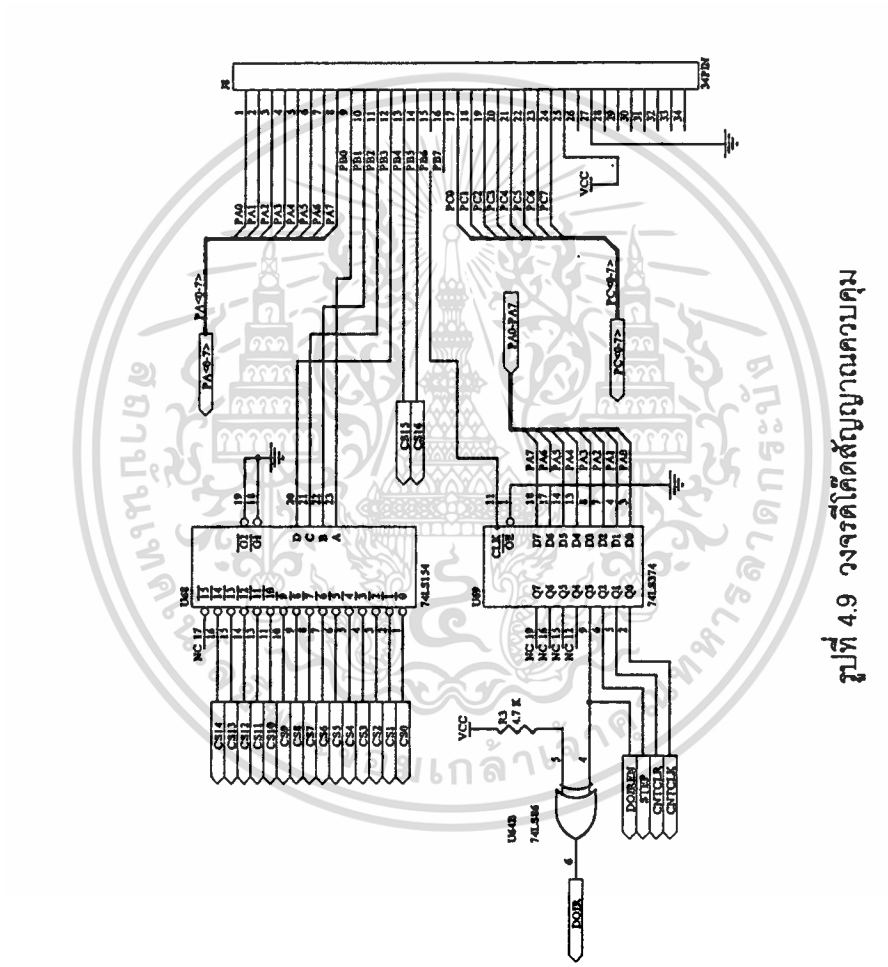
ภาพที่ 4.6 วงจรทริกเกอร์และวงจรผลิตสัญญาณ AOCLK และ WRC



รูปที่ 4.7 วงจรผลิตและหาความถี่



รูปที่ 4.8 วงจรแคชแรมและนับตำแหน่ง



รูปที่ 4.9 วงจรดีโคดสัญญาณควบคุม

บทที่ 5
การใช้เครื่องวิเคราะห์สัญญาณตรรก

ทำการรันโปรแกรมที่ชื่อ LOGIC.EXE บนเครื่องคอมพิวเตอร์ แล้วเปิดเครื่อง LOGIC ANALYZER ทำการกด SPACE BAR 2 ครั้ง เพื่อทดสอบการเชื่อมต่อสัญญาณ หลังจากการเชื่อมต่อสัญญาณเสร็จแล้ว LOGIC ANALYZER จะส่งครื่องหมายพรอมบ์ READY> ไปปรากฏที่จอเครื่องคอมพิวเตอร์เป็น READY> แสดงว่าพอร์ทสื่อสารทั้งสองได้เชื่อมต่อเข้าหากันเรียบร้อยแล้ว นั้นก็หมายว่าเครื่องวิเคราะห์สัญญาณตรรกพร้อมที่จะรับคำสั่งต่อไป ข้อความบนจอภาพจะเป็นดังรูปที่ 5.1

```
LOGIC ANALYZER KIT

BY JITTAKORN DEESAKUL

**********

COMMUNICATION LINK SETUP COMPLETE

TYPE H FOR DISPLAY HELP

READY>
```

รูปที่ 5.1 แสดงข้อความบนจอภาพเมื่อเครื่องพร้อมที่จะรับคำสั่ง

5.1 การทดลองคำสั่ง H (HELP)

เครื่องวิเคราะห์สัญญาณทางตรรกจะแสดงรายละเอียดของคำสั่งที่ใช้งาน แสดงขึ้นบนจอภาพดังในรูปที่ 5.2

```
READY>H
COMMAND USED FOLLOW
D SSSS FFFF          for dump data form memory
R                    for read data form CACH RAM
S                    for set sampling rate
Z                    for set trigger word
F5                   for display pluse train
READY>
```

รูปที่ 5.2 รายละเอียดของชุดคำสสั่ง "H"

5.2 การทดลองคำสั่ง T (TRIGGER WORD)

คำสั่งนี้จะใช้สำหรับการตั้งเวิร์ดทริกเกอร์ตามที่ผู้ใช้ต้องการ กล่าวคือ เมื่อใดก็ตามที่ข้อมูลที่รับเข้ามามีค่าเท่ากับเวิร์ดทริกเกอร์ที่ตั้งเอาไว้แล้ว เครื่องวิเคราะห์สัญญาณตรรกจะเริ่มทำการตรวจรับข้อมูล ดังที่ได้กล่าวมาแล้วว่า เราจะแบ่งสัญญาณ 40 ช่องสัญญาณ เป็น 5 กลุ่ม ดังนั้นเวิร์ดทริกเกอร์ก็จะมีอยู่ 5 กลุ่ม เช่นกัน (CH1-8 ,CH9-16 ,CH17-24 ,CH25-CH32 ,CH33-40) เครื่องวิเคราะห์สัญญาณตรรกจะแสดงรายละเอียดของคำสั่งที่ใช้งาน ซึ่งแสดงบนจอภาพดังรูปที่ 5.3 ซึ่งแสดงถึงการเลือกเวิร์ดทริกเกอร์ตามที่ผู้ใช้ต้องการ

```
READY>T
Set trigger bit , 1 for logic 1, 0 for logic 0 and D for don't care
Channel 1-8 for          bit 7 ..... bit 0
Channel 9- 16 for        bit 7 ..... bit 8
Channel 17-24 for        bit 7 ..... bit 0
Channel 25-32 for        bit 7 ..... bit 0
Channel 33-40 for        bit 7 ..... bit 0
Select channel for setting      : 1
set trigger bits for CH1-CH8 : DDDDDDDD
READY>T
Set trigger bit , 1 for logic 1, 0 for logic 0 and D for don't care
Channel 1-8 for          bit 7 ..... bit 0
Channel 9- 16 for        bit 7 ..... bit 8
Channel 17-24 for        bit 7 ..... bit 0
Channel 25-32 for        bit 7 ..... bit 0
Channel 33-40 for        bit 7 ..... bit 0
Select channel for setting      : 2
set trigger bits for CH1-CH8 : 00010011
READY>T
Set trigger bit , 1 for logic 1, 0 for logic 0 and D for don't care
Channel 1-8 for          bit 7 ..... bit 0
Channel 9- 16 for        bit 7 ..... bit 8
```

รูปที่ 5.3.1 รายละเอียดของชุดคำสั่ง "T"

```
Channel 17-24 for          bit 7 ..... bit 0
Channel 33-40 for          bit 7 ..... bit 0
Select channel for setting      : 3
set trigger bits for CH1-CH8  :  10000000
READY>T
Set trigger bit , 1 for logic 1, 0 for logic 0 and D for don't care
Channel 1-8 for            bit 7 ..... bit 0
Channel 9- 16 for          bit 7 ..... bit 8
Channel 17-24 for          bit 7 ..... bit 0
Channel 25-32 for          bit 7 ..... bit 0
Channel 33-40 for          bit 7 ..... bit 0
Select channel for setting      : 4
set trigger bits for CH1-CH8  :  DDDDDDDD
READY>T
Set trigger bit , 1 for logic 1, 0 for logic 0 and D for don't care
Channel 1-8 for            bit 7 ..... bit 0
Channel 9- 16 for          bit 7 ..... bit 8
Channel 17-24 for          bit 7 ..... bit 0
Channel 25-32 for          bit 7 ..... bit 0
Channel 33-40 for          bit 7 ..... bit 0
Select channel for setting      : 5
set trigger bits for CH1-CH8  :  11110000
READY>
```

รูปที่ 5.3.2 รายละเอียดของชุดคำสั่ง "T"

5.3 การทดลองคำสั่ง S (SAMPLING RATE)

คำสั่งนี้จะเป็นการตั้งค่าอัตราการสุ่มสัญญาณให้กับเครื่องวัดสัญญาณตรรก ดังในรูปที่ 5.4 ซึ่งสามารถเลือกอัตราการสุ่มสัญญาณได้ 2 แบบ

1. เราเลือกอัตราการสุ่มสัญญาณจากภายนอก โดยใช้คำสั่ง S และ E+ หรือ S และ E- (เครื่องหมายบวก (+) แสดงว่าเราไม่กลับเฟสของสัญญาณ และเครื่องหมายลบ (-) แสดงว่าเราผ่านวงจรกลับเฟสของสัญญาณ

2. เราเลือกอัตตราการสุ่มสัญญาณจากภายในที่ความถี่ 5 เมกกะเฮิทซ์

```
READY>S E+

READY>S E-

READY>S 5000000
```

รูปที่ 5.4 รายละเอียดของชุดคำสั่ง "S"

5.4 การทดลองคำสั่ง R (READ CACHE RAM)

คำสั่งนี้จะเป็นการสั่งให้เครื่องวัดสัญญาณตรรก นำสัญญาณที่ตรวจจับได้ซึ่งเก็บอยู่ที่แคชแรม ไปเก็บที่ แรมบนบอร์ด JR180

```
READY>R
```

รูปที่ 5.5 รายละเอียดของชุดคำสั่ง "R"

5.5 การทดลองคำสั่ง D (DUMP DATA)

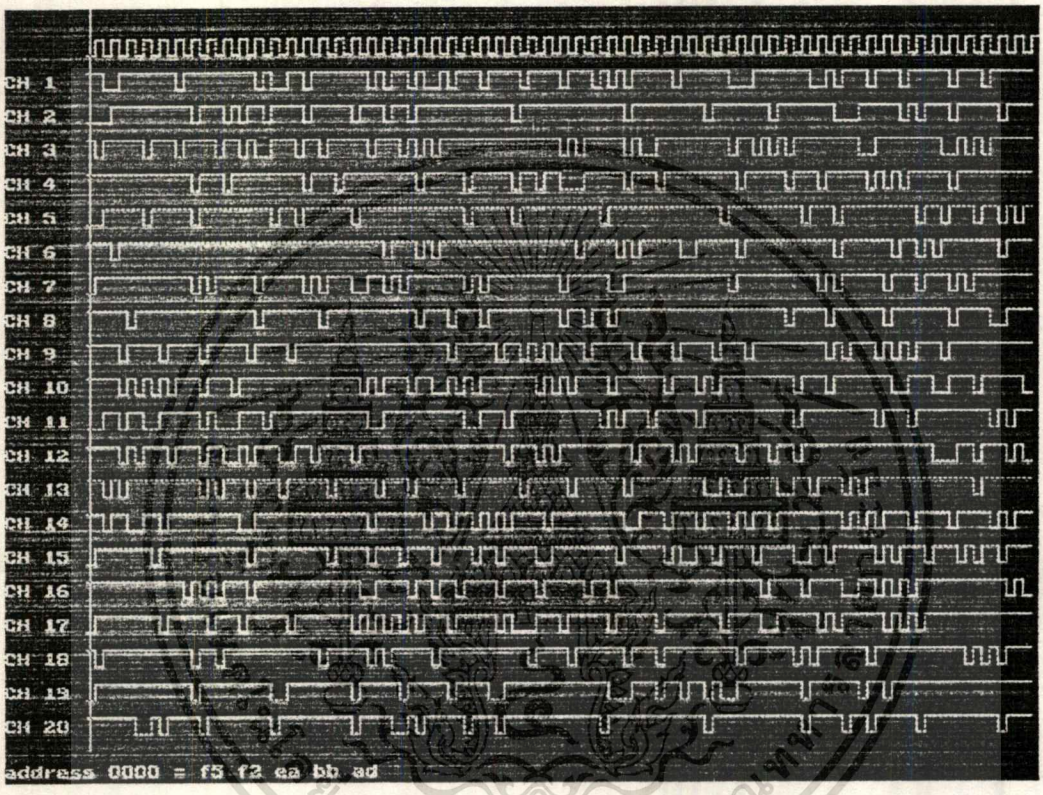
คำสั่งนี้จะเป็นการแสดงค่าในหน่วยความจำขึ้นมาในจอภาพ โดยจะทำการดัมหน่วยความจำ ตำแหน่งที่ SSSS H ไปที่ตำแหน่ง FFFF H จากรูปที่ 5.6 จะทำการดัมข้อมูลจากหน่วยความจำที่ ตำแหน่ง 8000H ไปที่ตำแหน่ง 80CFH

```
READY>D 8000 80CF

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

8000  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8010  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8020  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8030  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8040  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8050  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8060  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8070  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8080  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
8090  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
80A0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
80B0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
80C0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
```

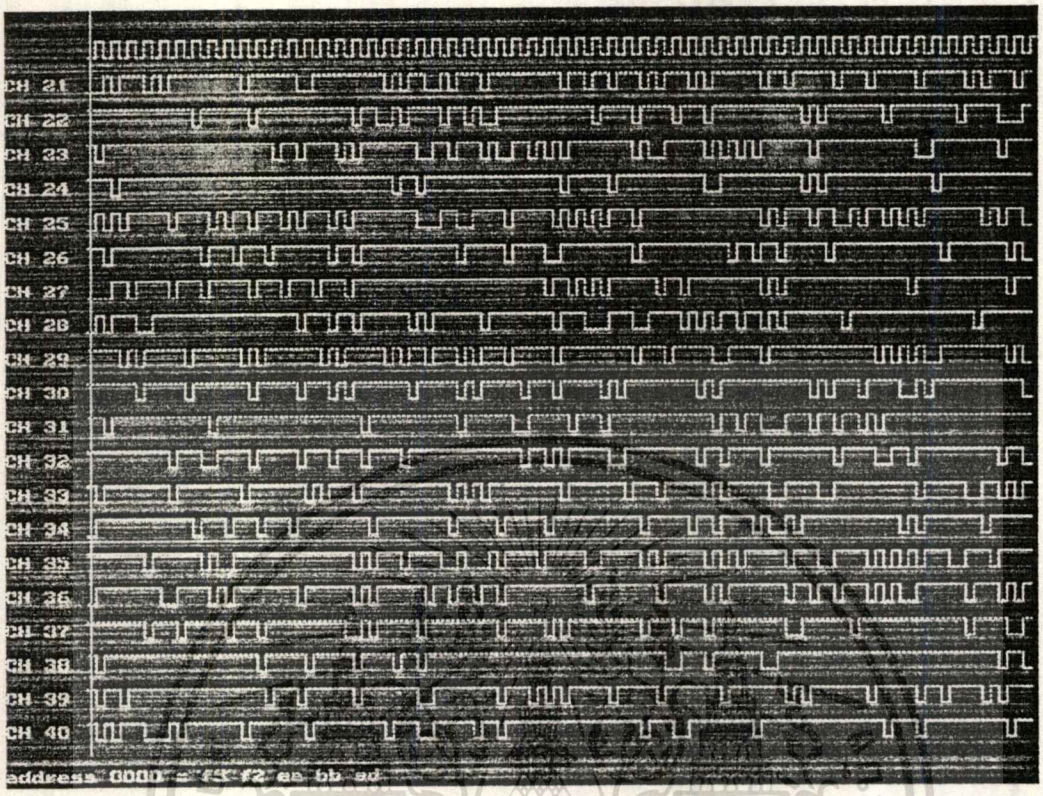
รูปที่ 5.5 รายละเอียดของชุดคำสั่ง "D"

5.5 การทดลองคำสั่ง F5 (DISPLAY PLUSE TRAIN)

คำสั่งนี้จะเป็นการแสดงรูปของสัญญาณแต่ละช่องสัญญาณบนจอภาพ หลังจากกด F5 ดังรูปที่ 5.6 แสดงผลสภาวะตรรก CH1-CH20 และรูปที่ 5.7 แสดงผลสภาวะตรรก CH21-CH40



รูปที่ 5.6 รายละเอียดของชุดคำสั่ง "F5" ที่ CH1-CH20



รูปที่ 5.7 รายละเอียดของชุดคำสั่ง "F5" ที่ CH21-CH40

บทที่ 6
สรุปผลและวิจารณ์

ผลที่ได้จากการทดลองใช้งานเครื่องวัดสัญญาณตรรกชุดนี้ สามารถทำการตรวจวัดได้ถูกต้องตามทฤฎีที่ออกแบบไว้ดังที่กล่าวไว้แล้วในบทที่ 2 ซึ่งผลการทดลองเป็นไปดังนี้คือ

1. สามารถทำการตรวจวัดสัญญาณตรรกได้ถึง 40 ช่องสัญญาณ แต่ละช่องสัญญณแยกกันโดยเด็ดขาด

2. อัตราการสุ่มสัญญาณสามารถจัดตั้งได้ เมื่อใช้สัญญาณจากภายใน ซึ่งอัตราสูงสุดที่กำหนดไว้คือ 50 เมกกะเฮิทซ์ และสามารถลดทอนลงด้วยอัตรา 5-2-1 เช่น 50, 20 ,10 ,5 ,2 ,1 เมกกะเฮิทซ์เป็นต้น

3. การกำหนดข้อมูลเริ่มต้นในการทำงาน (WORD TRIGGER) สามารถกำหนดได้ ความถี่ 40 ช่องสัญญาณ โดยสามารถกำหนดสถานะทางตรรกเป็นได้ทั้งตรรก 1,0 หรือไม่กำหนด (DON'T CARE) ซึ่งทั้งขึ้นอยู่ความต้องการของผู้ใช้งาน

4. การควบคุมการทำงานและแสดงผล กระทำผ่านทางคอมพิวเตอร์ส่วนบุคคล โดยผ่านทางพอร์ทอนุกรม RS-232

หนังสืออ้างอิง

[1] วิชิต ตั้งสุขนิรันดร " การพัฒนาไมโครโปรเซสเซอร์ให้เป็นลอจิกแอนนาไลเซอร์ "
:วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทารลาดกระบัง
:ประจำปี 2534

[2] ธันวา ศีรประมโมง "การเขียนปรแกรมภาษา ซี สำหรับวิศวกรรม. "
:มหาวิทยาลัยเทคโนโลยีมหานคร พิมครั้งที่สาม

[3] วิบูล ชื่นแขก " ไมโครโปรเซสเซอร์ "
:วิทยาลัยเทโนโลยีอุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
:พิมครั้งที่สอง พศ.2532

ส่วนของโปรแกรมแสดงผล (DISPLAY)

และ

โปรแกรมบนเครื่องวิเคราะห์สัญญาณตรรก

```
/* this program was written for logic analyzer machine*/
#include <graphics.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>

#define F4 62
#define ESC 27
#define DATA_READY 0x100
#define TRUE    1
#define FALSE   0
#define R_ARROW 77
#define LR_ARROW 116
#define L_ARROW 75
#define LL_ARROW 115
#define U_ARROW 0x48
#define D_ARROW 0x50
#define INDEX_RANGE 100
#define INDEX_SPACE 10
#define SETTINGS ( 0xE0 | 0x03 | 0x04 | 0x00)

/************************  prototype  ****************************/
void analyzer(void);
void gr_init(void);
void new_color(void);
void disp_logic(unsigned int start_add,unsigned char page,unsigned char data[500][5]);
void write_ch(unsigned int space,unsigned char page);
void draw_ref(unsigned int a,unsigned int b);
char pointer_manager(unsigned int x);
void show_add(unsigned int x,unsigned char data[500][5]);
void read_data(unsigned char data[500][5]);
void send_comm(void);
char hi_lo(unsigned char chan,unsigned char page,unsigned char new_data,unsigned char old_data);
unsigned char receive_comm(void);
unsigned char which_byte(unsigned char chan,unsigned char page);

//************************  main program  *********************************

int main(void)
```

```
{
        int in, out, status, DONE = FALSE;
        bioscom(0, SETTINGS,1);
        cprintf("... SERIAL COMMUNICATION SETUP ...\n");
        while (!DONE)
        {
                status = bioscom(3, 0, 1); //load check status of serial port
                if (status & DATA_READY)
                        if ((out = bioscom(2, 0, 1) & 0x7F) != 0)
                                putch(out);
                        if (kbhit())
                        {
                                if ((in = getch()) == '\x1B')
                                        { DONE = TRUE; }
                                else
                                        if ( in == 63)
                                        { analyzer(); }
                                                bioscom(1, in, 1);
                        }
        }
        return 0;
}

void analyzer(void) {
        unsigned char data[500][5],page;
        unsigned int size,address = 0,index_pos = 0;
        void far *index;
        char ch = 0,flag = 1,second_key = 0;

        printf("please wait ... \n");
        read_data(data);
        gr_init();
        line(0,0,0,getmaxy()-20);
        size = imagesize(0,0,0,getmaxy()-20);
        index = malloc(size);
        getimage(0,0,0,getmaxy()-20,index);
        putimage(0,0,index,XOR_PUT);
        new_color();
        page = 0;
        for(;ch != 0xf;)
        {
```

```
if(flag)
{
        cleardevice();
        disp_logic(address,page,data);
        putimage(52+index_pos*5,0,index,XOR_PUT);
}
flag = 0;
show_add(address+index_pos,data);
second_key = 0;
if (!(ch = getch()))
{
        ch = getch();
        second_key = 1;
}
if (second_key)
        switch (ch)
        {
                case LR_ARROW: index_pos = INDEX_RANGE;
                case R_ARROW:
                        if (pointer_manager(index_pos) != 1)
                        {
                                putimage(52+index_pos*5,0,index,XOR_PUT);
                                index_pos++;
                                putimage(52+index_pos*5,0,index,XOR_PUT);
                        }
                        else
                        {
                                if (address<0x7b0)
                                        address += INDEX_RANGE-INDEX_SPACE;
                                index_pos = INDEX_SPACE;
                                flag = 1;
                        }
                        break;
                case LL_ARROW: index_pos = 0; flag = 1;
                case L_ARROW:
                        if (pointer_manager(index_pos) != -1)
                        {
                                putimage(52+index_pos*5,0,index,XOR_PUT);
                                index_pos--;
                                putimage(52+index_pos*5,0,index,XOR_PUT);
                        }
```

```
                                        else
                                                if (address>=INDEX_RANGE-INDEX_SPACE)
                                                {
                                                        address -= INDEX_RANGE-INDEX_SPACE;
                                                        flag = 1;
                                                        index_pos = INDEX_RANGE-INDEX_SPACE;
                                                }
                                        break;
                                case D_ARROW: page = 1; cleardevice(); flag = 1;  break;
                                case U_ARROW: page = 0; cleardevice(); flag = 1;  break;
                        }
                        else
                                switch(ch)
                                {
                                        case ESC: ch = 0xf; break;
                                }
        }
//      outp(0x3f9,1);
        closegraph();
}

void disp_logic(unsigned int start_add,unsigned char page,unsigned char data[500][5])
{
        unsigned char offset,chan;
        unsigned char new_data[5],old_data[5];
        unsigned int space_y,gen_pur;
        char up =-10,down =10,transition,byte_num;
        space_y = getmaxy()/22;
        write_ch(space_y,page);
        draw_ref(50,space_y+textheight("c") );
        for (offset = 0 ;offset <117 ;offset++)
        {
                for (gen_pur=0;gen_pur<=4;gen_pur++)
                {
                        new_data[gen_pur] = data[offset+start_add][gen_pur];
                        if (offset==0)
                                old_data[gen_pur]=new_data[gen_pur];
                }
                for (chan=0;chan<=19;chan++)
                {
                        byte_num = which_byte(chan,page)+(page*2);
```

```
                moveto(50+(offset*5),(chan+2)*space_y+textheight("c"));
                switch(transition= hi_lo(chan,page,new_data[byte_num],old_data[byte_num]))
                {
                        case -2:    break;
                        case -1:    linerel(0,up); moverel(0,down); break;
                        case  1:    linerel(0,up); break;
                        case  2:    moverel(0,up); break;
                }
                linerel(5,0);
        }
        for (gen_pur=0;gen_pur<=4;gen_pur++)
                old_data[gen_pur] = new_data[gen_pur];
    }
}

void show_add(unsigned int x,unsigned char data[500][5])
{
    char screen_out[80];
    setviewport(0,getmaxy()-10,getmaxx(),getmaxy(),1);
    clearviewport();
    sprintf(screen_out,"address %04x = %02x %02x %02x %02x %02x ",x,
    data[x][4],data[x][3],data[x][2],data[x][1],data[x][0]);
    outtextxy(0,0,screen_out);
    setviewport(0,0,getmaxx(),getmaxy(),1);
}

void read_data(unsigned char data[500][5])
{
    int  x,y;
    unsigned char temp,temp1;
    send_comm();
    for (x=0;x<=500;x++)
    {
                for (y=0;y<=4;y++)
                {
                        do temp = receive_comm();
                        while(temp==0x20);
                        if (temp<0x40)
                                temp -= 0x30;
                        else
                                temp -= 0x37;
```

```
                                        temp1 = receive_comm();
                                        if (temp1<0x40)
                                                temp1 -= 0x30;
                                        else
                                                temp1 -= 0x37;
                                        data[x][y] = (temp << 4) + temp1;
                                }
        }
}

void send_comm(void)
{
        int in, out, status;
        char output[50] = "rLr\n";
        char temp;
        temp = 0;
        do {
                if (bioscom(3,0,1) & 0x100)
                        delay(100);
                bioscom(1,output[temp],1);
                temp++;
        } while(output[temp] != '\n');
}

unsigned char receive_comm(void)
{
        char temp;
        do {
        temp = bioscom(3,0,1);
        } while ((temp & 0x100) != 0);
        temp = bioscom(2,0,1);
        return temp;
}

void draw_ref(unsigned int a,unsigned int b)
{
        char x;
        moveto(a,b);
        for (x=0;x<59;x++)
        {
                linerel(5,0);
```

```
                linerel(0,-10);
                linerel(5,0);
                linerel(0,10);
        }
}

char pointer_manager(unsigned int x)
{
        switch (x)
        {
                case INDEX_RANGE: return(1);
                case 0:     return(-1);
                default:    return(0);
        }
}

unsigned char which_byte(unsigned char chan,unsigned char page)
{
        div_t x;
        chan += page*4;
        x = div(chan,8);
        return(x.quot);
}

char hi_lo(unsigned char chan,unsigned char page,unsigned char new_data,unsigned char old_data)
{
        div_t x;
        chan += page*4;
        x = div(chan,8);
        if (((old_data^ new_data) >> x.rem) & 0x01 )
        {
                if ((old_data >> x.rem) & 0x01)
                        return(-1);
                else
                        return(1);
        }
        else
        {
                if ((old_data >> x.rem) & 0x01)
                        return(2);
                else return(-2);            //-2 = low to low , 2 = high to high
```

```
        }
}

void gr_init(void)
{
        int gdriver = DETECT, gmode, errorcode;
        /********************** int graphics mode ***************************/
        initgraph(&gdriver,&gmode,"");
        /****************** read result of intialization ********************/
        errorcode = graphresult();
        if (errorcode != grOk)
        {
                printf("Graphics error: %s \n",grapherrormsg(errorcode));
                printf("Press any key to halt:");
                getch();
                exit(1);
        }
}

void new_color(void)
{
        unsigned char i,j,k,l;
        for (i=0;i<=3;i++)
        for (j=0;j<=3;j++) setpalette(k++,i+j*4);
}

void write_ch(unsigned int space,unsigned char page)
{
        char x;
        char screen_out[80];
        for (x=1;x<=20;x++)
        {
                if (page)
                        sprintf(screen_out,"CH %d",x+20);
                else
                        sprintf(screen_out,"CH %d",x);
                outtextxy(0,(x+1)*space,screen_out);
        }
}
```

```
;**********
;Use        : CPU Z80180
;           :   32K ROM Monitor  Address 0000H-7FFFH
;           :   32K RAM User Area Adress 8000H-FFFFH
;                       - Logic  Area  Address  8000H-AFFFH
;                       - Buffer Area  Address  B000H-EFFFH
;                       - Stack  Area  Address  F000H-FFFFH
;           : RS-232 CH1
;           : Watch Dog
;           : Cross-32 Complier
;**********
;
;**********
            CPU     "Z180.TBL"
            HOF     "INT8"
;**********
;
;**********
;PPI        EQUATE
;**********
PA          EQU     80H
PB          EQU     81H
PC          EQU     82H
CTRL        EQU     83H
TX          EQU     07H
RX          EQU     05H
;
;**************
;ADDRESS    EQUATE
;**************
;
BRATE       EQU     0F000H
;ERR        EQU     00171H
BRKSRV      EQU     0B300H
DRATE       EQU     0B202H
CHRBUF      EQU     0B000H      ;USE 8000H-8010H
FIRST       EQU     0B103H
LAST        EQU     0B114H
CHSET       EQU     0B107H
FTRIG       EQU     0B108H
;
;*****************
;CHARACTER          EQUATES
;*****************
;
CR          EQU     0DH
LF          EQU     0AH
;
```

```
;*****************
;           START MAIN PROGRAM
;*****************
;
            ORG     0
START:      LD      B,0FFH      ;delay for wake up
WAKE:       DJNZ    WAKE
            JP      START1      ;jump pass interrupt table
BPOINT:     JP      BRKSRV      ;break point service routine
            DFB     0,0,0,0,0
M0010:      JP      WARM        ;user monitor
            DFB     0,0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0020:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0030:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0040:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0050:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0060:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0070:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0080:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M0090:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00A0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00B0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00C0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00D0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00E0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
M00F0:      DFB     0,0,0,0, 0,0,0,0
            DFB     0,0,0,0, 0,0,0,0
;*****************
START1:     DI
            LD      B,50H       ;Clear all buffer
            LD      HL,0FE00H
START2:     LD      (HL),00
            INC     HL
            DJNZ    START2
```

```
;*********************
; INITIAL PARAMETER
;*********************
;INNITIAL CPU
        LD A,0H             ;Set Operation Mode Control Register
        OUT0 (3EH),A        ;TO Z180
;Set Register
        LD A,0F8H           ;Save Logical Address
        OUT0 (3AH),A
        LD A,08H
        OUT0 (39H),A        ;Physical bank = 10000H
        OUT0 (38H),A        ;Physical com area1 = 17000H
;Set I/O wait state = 4 state : INT0 = 6
        LD   A,30H
        OUT0 (32H),A        ;DMA/WAIT Control Register
;Set refresh = 10 cycles
        LD   A,81H
        OUT0 (36H),A        ;Out to Refresh control register
;Interrupt vecter
        LD   SP,0FFFFH      ;Set stack
        LD   HL,0F000H
        LD   A,H
        LD   I,A            ;Load interrupt vector high
        LD   A,L
        LD   B,00H
        OUT0 (33H),A        ;Load interrupt vector low  A => B
        IM   1              ;Set interrupt mode 1
;Enable interrupt
        LD   A,07H
        OUT0 (34H),A        ;( Mpu P17 )
;Innitial port
        LD   A,04H          ;Set Status Register CH1
        OUT0 (05H),A        ;Receive Interrupt Enable
        OUT0 (0E0H),A       ;Trigger  Watch DOG
        LD   A,89H          ;Control word of 8255
        OUT  (CTRL),A       ;All port A,B, are Output, port C is Input
        LD   HL,BRATE       ;Set baud rate = 1200
;
UPLINK: LD   A,75H          ;Set Protocal 1st, 8 bits 2 stop
        OUT0 (01H),A        ;Send format of communication
        LD   A,02H          ;Set baud rate = 38400
        LD   C,03H
        OUT  (C),A
        CALL GETCHR         ;wait char form terminal
        CALL GETCHR
        CP   20H            ;"space" ?
        JR   NZ,UPLINK      ;no cont wait
        LD   HL,SIGNON      ;display sign on mssg
        CALL SNDOUT
;
WARM:   LD   SP,0FFFFH      ;set stack
        LD   A,08H          ;set STEP,CNTCLK,CNTCLR = "0"
        OUT  (PA),A
        LD   B,3FH
        CALL ENABLE
        LD   A,0CH          ;set STEP,DOIEN = "1"
        OUT  (PA),A
        LD   B,3FH
        CALL ENABLE
        LD   A,0FFH
        OUT  (PA),A
;
RDY:    LD   HL,READY       ;display prompt
        CALL SNDOUT
;
AA:     CALL GETCHR         ;wait command from terminal
        CALL LC2UC          ;convert lowercase to upper case
        LD   (CHRBUF),A     ;save in chrbuf
        CALL SNDCHR         ;echo back
        CP   "D"
        JP   Z,DUMP         ;dump memory
        CP   "H"
        JP   Z,HELP         ;display help
        CP   "R"
        JP   Z,RDALL        ;read data from cache RAM
        CP   "S"
        JP   Z,SAMP         ;set sampling rate
        CP   "T"
        JP   Z,STRIG        [;set word trig
        CP   "Z"
        JP   Z,ANALY
        CP   "L"
        JP   Z,DISPLAY      ;display logic status
        JP   RDY
;
SIGNON: DFB  CR,LF,CR,LF,CR,LF
        DFB  CR,LF,"        LOGIG ANALYZER KIT",CR,LF
        DFB  CR,LF,"     BY JITTAKORN DESAKUL",CR,LF
        DFB  CR,LF,"  **************************",CR,LF
        DFB  CR,LF,CR,LF
        DFB  CR,LF,"COMMMUNICATION LINK SETUP  COMPLETE",CR,LF
        DFB  CR,LF,"      TYPE 'H' FOR DISPLAY HELP",CR,LF
        DFB  CR,LF,CR,LF,CR,LF,04H
READY:  DFB  CR,LF,"READY>",04H
;
;*******
```

```
;       Start Logic Analyzer
;*******
;
ANALY:  LD      HL,ANALYPAT
        CALL    SNDOUT
        IN      A,(PB)          ;disable decode
        AND     0C0H
        OR      7FH
        OUT     (PB),A
        PUSH    AF
        LD      A,0FH           ;set frequency count(50Mz)= 15 cycle
        OUT     (PA),A
        LD      B,7EH
        CALL    ENABLE
        CALL    DELAY1
        POP     AF
        LD      A,0BH           ;set STEP,CNTCLK = "0"
        OUT     (PA),A
        LD      B,3FH
        CALL    ENABLE
;       CALL    DELAY1
;       OR      11010000B
;       OUT     (PB),A
        CALL    GETCHR
        JP      WARM
;
ANALYPAT: DFB   CR,LF," Logic Analyzer: ",04H
;
;*******
;Convert Binary to Hex ASCII
;IN:    (A) = FB
;OUT:   (H) = 46  (F)
;       (L) = 42H (B)
;*******
;
BN2HEX: LD      B,A
        AND     0F0H
        RRCA
        RRCA
        RRCA
        RRCA
        CALL    NASCII
        LD      H,A
        LD      A,B
        AND     0FH
        CALL    NASCII
        LD      L,A
        RET
;
NASCII: CP      10D
        JR      C,NAS1
        ADD     A,7
NAS1:   ADD     A,"0"
        RET
;
;***********
;       Send CR and LF to Terminal
;***********
;
CRLF:   LD      A,CR
        CALL    SNDCHR
        LD      A,LF
        CALL    SNDCHR
        RET
;
;**********
;       Time Delay
;**********
;
DELAY0: NOP
        RET
DELAY1: LD      B,0H
DEL1:   DJNZ    DEL1
        RET
DELAY2: LD      B,0AH
DEL2:   DJNZ    DEL2
        RET
DELAY3: LD      HL,00FFH
LOOP:   DEC     HL
        LD      A,H
        OR      L
        JP      NZ,LOOP
        RET
;
;**********
;       Dump data from memory and send to terminal OK
;**********
;
DUMP:   CALL    GETCMD  ;Get command line
        LD      DE,CHRBUF       ;First char "D"
        INC     DE
        LD      A,(DE)  ;Next space
        CP      20H
        JP      NZ,ERR
        INC     DE      ;Next start address
        LD      A,(DE)          ;Get ASCII
```

```
LD      H,A         ;First ASCII in H
INC     DE
LD      A,(DE)
LD      L,A         ;Second in L
CALL    HEX2BN      ;ASCII to Hex (in HL result in Acc)
LD      HL,FIRST    ;Save hi-addr in first
LD      (HL),A      ;Save
LD      DE,CHRBUF+4 ;Next convert low addr
LD      A,(DE)      ;Get ASCII
LD      H,A
INC     DE
LD      A,(DE)
LD      L,A
CALL    HEX2BN
LD      HL,FIRST+1  ;Save low-addr in first+1
LD      (HL),A
LD      DE,CHRBUF+6
LD      A,(DE)
CP      20H
JP      NZ,ERR
INC     DE
LD      A,(DE)
LD      H,A
INC     DE
LD      A,(DE)
LD      L,A
CALL    HEX2BN
LD      HL,LAST
LD      (HL),A
LD      DE,CHRBUF+9
LD      A,(DE)
LD      H,A
INC     DE
LD      A,(DE)
LD      L,A
CALL    HEX2BN
LD      HL,LAST+1
LD      (HL),A
LD      HL,FIRST+1
LD      A,(HL)
AND     0F0H
LD      (HL),A
CALL    CRLF
LD      HL,AMAP
CALL    SNDOUT
CALL    SNDADD
CALL    SPACE2
CALL    SPACE1
;
DUMP3:  LD      A,(FIRST)
        LD      H,A
        LD      A,(FIRST+1)
        LD      L,A
        LD      B,10H       ;Display hex 10h byte
        PUSH    HL
;
DUMP1:  PUSH    HL
        LD      A,(HL)
        PUSH    BC
        CALL    BN2HEX
        POP     BC
        LD      A,H
        CALL    SNDCHR
        LD      A,L
        CALL    SNDCHR
        LD      A,B
        CP      9H
        JR      NZ,SPC1
        CALL    SPACE1
;
SPC1:   CALL    SPACE1
        POP     HL
        INC     HL
        PUSH    BC
        LD      B,00H
        LD      C,05H
        IN      A,(C)
        AND     80H
        POP     BC
        JP      Z,CNT
        PUSH    BC
        LD      B,00H
        LD      C,09H
        IN      A,(C)
        POP     BC
        JP      WARM
;
CNT:    DJNZ    DUMP1
        CALL    SPACE3
        POP     HL
        LD      B,10H
;
DASCII: LD      A,(HL)
        CP      20H
        JR      C,J1
        CP      80H
```

```
        JP      C,J2
;
J1:     LD      A,2EH
J2:     CALL    SNDCHR
        INC     HL
        LD      A,B
        CP      0H          ;DON SURE PAGE.12
        JR      NZ,J3
        CALL    SPACE1
;
J3:     DJNZ    DASCII
        LD      A,H
        LD      (FIRST),A
        LD      A,L
        LD      (FIRST+1),A
        PUSH    HL
        CALL    CRLF
        POP     HL
        LD      A,(LAST)
        LD      D,A
        LD      A,(LAST+1)
        LD      E,A
        XOR     A
        SBC     HL,DE
        JR      NC,DUMP2
        CALL    SNDADD
        CALL    SPACE2
        CALL    SPACE1
        JP      DUMP3
;
DUMP2:  JP      WARM
;
DISPLAY:
        LD      HL,8000H
        LD      BC,1000H
;
DISP1:  PUSH    HL
        LD      A,(HL)
        PUSH    BC
        CALL    BN2HEX
        POP     BC
        LD      A,H
        CALL    SNDCHR
        LD      A,L
        CALL    SNDCHR
        CALL    SPACE1
        POP     HL
        INC     HL
        DEC     C
        XOR     A
        OR      C
        JR      NZ,DISP1
        OR      B
        JR      NZ,DISP2
        JP      WARM
;
DISP2:  DEC     B
        JP      DISP1
;
EREMSG: DFB   CR,LF,"Syntax error ,type H for help",CR,LF,40H
;
;*******
;       TEST
;*******
;
ERR:    LD      HL,EREMSG
        CALL    SNDOUT
        JP      WARM
;
TESTMSG: DFB        CR,LF,"TEST",CR,LF,04H
;
;**********
;       Get one chr form terminal
;**********
;
GETCHR: CALL GCHR
        JR Z,GETCHR
        AND 7FH
        OR A
        SCF
        RET
;
GCHR:   PUSH BC
        LD B,00H
        LD C,05H
        IN A,(C)
        AND 80H
        POP BC
        RET Z
        PUSH BC
        LD B,00H
        LD C,09H
        IN A,(C)
        POP BC
        OR A
        RET
```

```
;
;*******
;       Get address ret when press CR
;       Add save in CHRBUFC
;*******
;
GETADD: LD      HL,CHRBUF
        LD      B,10H
;
GADD1:  LD      (HL),00H
        INC     HL
        DJNZ    GADD1
        LD      HL,CHRBUF
;
GADD3:  PUSH    HL
        CALL    GETCHR
        CALL    LC2UC
        CALL    SNDCHR
        POP     HL
        CP      0DH
        JR      NZ,GADD2
        LD      (HL),0FFH
        RET
;
GADD2:  CP      08H
        JR      Z,GADD3
        LD      (HL),A
        INC     HL
        JR      GADD3
;
;*******
;       Get data form terminal save in chrbuf
;       CR -> RET,SPACE -> JP TO RDY
;*******
;
GETDAT: LD      HL,CHRBUF
        LD      B,10H
;
GDAT1:  LD      (HL),00H
        INC     HL
        DJNZ    GDAT1
        LD      HL,CHRBUF
        LD      B,3H
;
GDAT3:  PUSH    HL
        CALL    GETCHR
        CALL    LC2UC
        POP     HL
        CP      0DH
        JR      NZ,GDAT2
        LD      (HL),0FFH
        RET
;
GDAT2:  CP      08H
        JR      Z,GDAT3
        CP      1BH
        PUSH    AF
        PUSH    BC
        LD      B,00H
        LD      C,09H
        IN      A,(C)
        POP     BC
        POP     AF
        JR      Z,GDAT4
        DJNZ    J4
        PUSH    AF
        LD      A,08H
        CALL    SNDCHR
        CALL    SNDCHR
        DEC     HL
        DEC     HL
        LD      B,2H
        POP     AF
;
J4:     LD      (HL),A
        CALL    SNDCHR
        INC     HL
        JR      GDAT3
;
GDAT4:  POP     HL
        JP      WARM
;
;*******
;       Get command line
;*******
;
GETCMD: LD      HL,CHRBUF
        LD      B,10H
;
GCMD1:  LD      (HL),00H
        INC     HL
        DJNZ    GCMD1
        LD      HL,CHRBUF
        LD      (HL),A
        INC     HL
;
```

```
GCMD3:  CALL    GETCHR
        CALL    LC2UC
        CALL    SNDCHR
        CP      0DH
        JR      NZ,GCMD2
        LD      (HL),0FFH
        RET
;
GCMD2:  CP      08H
        DEC     HL
        JR      Z,GCMD3
        INC     HL
        LD      (HL),A
        INC     HL
        JR      GCMD3
;
;*******
;       Send help mssg to terminal
;*******
;
HELP:   LD      HL,HMSSG
        CALL    SNDOUT
        CALL    CRLF
        JP      WARM
;
HMSSG:  DFB     CR,LF,"User RAM area start at address 8000:",CR,LF
        DFB     CR,LF,"Command used follow:",CR,LF
        DFB     CR,LF,"D SSSS FFFF for dump data from memory"
        DFB     CR,LF,"H        for display this message"
        DFB     CR,LF,"R        for read data from cache RAM"
        DFB     CR,LF,"S        for set sampling rate"
        DFB     CR,LF,"T        for set triggering word"
        DFB     CR,LF,"Z        for write cache RAM"
        DFB     CR,LF,"F5       for display pluse train",04H
;
;*******
;       Convert hex ASCII to Binary
;IN:    (H) = 44 (D)
;       (L) = 37 (7)
;OUT:   (A) = D7
;*******
;
HEX2BN: LD      A,L
        CALL    A2HEX
        LD      B,A
        LD      A,H
        CALL    A2HEX
        RRCA
        RRCA
        RRCA
        RRCA
        OR      B
        RET
;
A2HEX:  SUB     "0"
        CP      10D
        JR      C,A2HEX1
        SUB     7
;
A2HEX1: RET
;
;*******
;       Convert lower case to upper case
;IN:    CHR lower case in Acc
;OUT:   CHR upper case in Acc
;*******
;
LC2UC:  CP      "a"
        JR      C,EXIT
        CP      "z"+1
        JR      NC,EXIT
        SUB     "a"-"A"
EXIT:   RET
;
;*******
;       Read data from cache RAM OK
;*******
;
RDALL:  LD      HL,8000H    ;Start add for data save
        LD      A,00H       ;Set DOIREN,STEP,CNTCLR,CNTCLK = "0"
        OUT     (PA),A
        PUSH    AF
        LD      B,3FH
        CALL    ENABLE
        POP     AF
        OR      04H         ;Set STEP = "1"
        OUT     (PA),A
        LD      B,3FH
        CALL    ENABLE
        LD      BC,1000H    ;Start counter 1000 byte
;
RDAL1:  PUSH    BC
        CALL    PLUSE       ;Inc counter
        POP     BC
        LD      A,7BH       ;Get data form pod 1 (U45)
        OUT     (PB),A
```

```
        IN      A,(PC)
        LD      (HL),A
        INC     HL
        LD      A,7CH       ;Get data from pod 2 (U46)
        OUT     (PB),A
        IN      A,(PC)
        LD      (HL),A
        INC     HL
        LD      A,7DH       ;Get data form pod 3 (U47)
        OUT     (PB),A
        IN      A,(PC)
        LD      (HL),A
        INC     HL
        LD      A,7EH       ;Get data from pod 4 (U48)
        OUT     (PB),A
        IN      A,(PC)
        LD      (HL),A
        INC     HL
        LD      A,6FH       ;Get data form pod 5 (U49)
        OUT     (PB),A
        IN      A,(PC)
        LD      (HL),A
        INC     HL
        DEC     C           ; DEC COUNT
        XOR     A
        OR      C
        JR      NZ,RDAL1    ; CHK COUNT
        OR      B
        JR      NZ,RDAL2
        JP      WARM
;
RDAL2:  DEC     B
        JP      RDAL1
;
PLUSE:  LD      A,02H       ;Set STEP = "0" ,CNTCLR = "1"
        OUT     (PA),A
        LD      B,3FH
        CALL    ENABLE
;       CALL    DELAY1
        LD      A,06H       ;Set STEP = "1" ,CNTCLR = "1"
        OUT     (PA),A
        LD      B,3FH
        CALL    ENABLE
        RET
;
;*******
;       Set sampling rate
;*******
SAMP:   CALL    GETCMD      ;GET COMMAND LINE
        LD      DE,CHRBUF
        INC     DE
        LD      A,(DE)
        CP      20H         ;CHK SPACE
        JP      NZ,ERR
        INC     DE
        LD      A,(DE)
        CP      "1"         ;BGN WITH 1?
        JP      Z,ONE
        CP      "2"         ;BGN WITH 2?
        JP      Z,SECND
        CP      "5"         ;BNG WITH 5?
        JP      Z,FIVE
        CP      "E"
        JP      Z,EXT1
        JP      NZ,ERR
;
ONE:    INC     DE          ;CARACTOR 4
        XOR     A
        LD      B,00H
        LD      C,10H
        CALL    CHKZERO
        LD      A,B
        OR      C
        LD      C,A
        JP      SMAP
;
SECND:  INC     DE          ;CHK TAILLING ZERO
        XOR     A
        LD      B,A
        LD      C,20H
        CALL    CHKZERO
        LD      A,B
        OR      C
        LD      C,A
        JP      SMAP
;
FIVE:   INC     DE          ;CHK TAILLING ZERO
        XOR     A
        LD      B,A
        LD      C,50H
        CALL    CHKZERO
        LD      A,B
        OR      C
        LD      C,A
;
SMAP:   CP      11H         ;10 Hz
```

```
        JR      NZ,S21
        LD      A,3BH
        LD      (DRATE),A
        JP      SSSS
;
S21:    CP      21H     ;20 Hz
        JR      NZ,S51
        LD      A,3EH
        LD      (DRATE),A
        JP      SSSS
;
S51:    CP      51H     ;50 Hz
        JR      NZ,S12
        LD      A,31H
        LD      (DRATE),A
        JP      SSSS
;
S12:    CP      12H     ;100 Hz
        JR      NZ,S22
        LD      A,33H
        LD      (DRATE),A
        JP      SSSS
;
S22:    CP      22H     ;200 Hz
        JR      NZ,S52
        LD      A,36H
        LD      (DRATE),A
        JP      SSSS
;
S52:    CP      52H     ;500 Hz
        JR      NZ,S13
        LD      A,29H
        LD      (DRATE),A
        JP      SSSS
;
S13:    CP      13H     ;1 kHz
        JR      NZ,S23
        LD      A,2BH
        LD      (DRATE),A
        JP      SSSS
;
S23:    CP      23H     ;2 kHz
        JR      NZ,S53
        LD      A,2EH
        LD      (DRATE),A
        JP      SSSS
;
S53:    CP      53H     ;5 kHz
        JR      NZ,S14
        LD      A,21H
        LD      (DRATE),A
        JP      SSSS
;
S14:    CP      14H     ;10 kHz
        JR      NZ,S24
        LD      A,23H
        LD      (DRATE),A
        JP      SSSS
;
S24:    CP      24H     ;20 kHz
        JR      NZ,S54
        LD      A,26H
        LD      (DRATE),A
        JP      SSSS
;
S54:    CP      54H     ;50 kHz
        JR      NZ,S15
        LD      A,19H
        LD      (DRATE),A
        JP      SSSS
;
S15:    CP      15H     ;100 kHz
        JR      NZ,S25
        LD      A,18H
        LD      (DRATE),A
        JP      SSSS
;
S25:    CP      25H     ;200 kHz
        JR      NZ,S55
        LD      A,1EH
        LD      (DRATE),A
        JP      SSSS
;
S55:    CP      55H     ;500 kHz
        JR      NZ,S16
        LD      A,11H
        LD      (DRATE),A
        JP      SSSS
;
S16:    CP      16H     ;1 MHz
        JR      NZ,S26
        LD      A,13H
        LD      (DRATE),A
        JP      SSSS
;
S26:    CP      26H     ;2 MHz
```

```
        JR      NZ,S56
        LD      A,16H
        LD      (DRATE),A
        JP      SSSS
;
S56:    CP      56H     ;5 MHz
        JR      NZ,S17
        LD      A,09H
        LD      (DRATE),A
        JP      SSSS
;
S17:    CP      17H     ;10 MHz
        JR      NZ,S27
        LD      A,0BH
        LD      (DRATE),A
        JP      SSSS
;
S27:    CP      27H     ;20 MHz
        JR      NZ,S57
        LD      A,0EH
        LD      (DRATE),A
        JP      SSSS
;
S57:    CP      57H     ;50 MHz
        JR      NZ,S18
        LD      A,00H
        LD      (DRATE),A
        JP      SSSS
;
S18:    JP      ERR
;
EXT1:   INC     DE
        CP      '+'
        JR      NZ,EXTL1
        LD      A,0CH
        LD      (DRATE),A
        JP      SSSS
;
EXTL1:  CP      '-'
        JP      NZ,ERR
        LD      A,4CH
        LD      (DRATE),A
;
SSSS:   IN      A,(PB)
        AND     0C0H
        OR      7FH
        OUT     (PB),A
        LD      A,(DRATE)
        OUT     (PA),A      ;OUT TO SELECT FREQUENCY
        LD      B,5FH
        CALL    ENABLE
        CALL    CRLF
        JP      WARM
;
CHKZERO:
        LD      A,(DE)
        CP      '0'
        RET     NZ
        INC     DE
        INC     B
        JR      CHKZERO
;
;*******
;       Send data in Acc
;*******
;
SNDDAT: CALL    BN2HEX  ;CONVERT TO ASCII
        PUSH    HL
        LD      A,H     ;SENT TO TERMINAL
        CALL    SNDCHR  ;SEND HI ADD
        POP     HL
        LD      A,L     ;THEN SEND LO ADD
        CALL    SNDCHR
        RET
;
BRKPAT: DFB     CR,LF,'Break at address : ',04H
REGAF:  DFB     'Register AF = ',04H
REGBC:  DFB     'Register BC = ',04H
REGDE:  DFB     'Register DE = ',04H
REGHL:  DFB     'Register HL = ',04H
REGIX:  DFB     'Register IX = ',04H
REGIY:  DFB     'Register IY = ',04H
BRKDSP: DFB     'Hit space bar to continue else go to prompt',04H
;
;*******
;       Set word trig '?'
;*******
;
STRIG:  LD      HL,TRGPAT
        CALL    SNDOUT
        CALL    GETCHR
        PUSH    AF
        CALL    SNDCHR
        POP     AF
        LD      (FTRIG),A
        CP      '1'
```

```
        JP      NZ,CH02
;
CH01:   LD      HL,TRGCH1
        CALL    SNDOUT
        JR      GTR         ;GET TRIG
;
CH02:   CP      "2"
        JP      NZ,CH03
        LD      HL,TRGCH2
        CALL    SNDOUT
        JR      GTR
;
CH03:   CP      "3"
        JP      NZ,CH04
        LD      HL,TRGCH3
        CALL    SNDOUT
        JR      GTR
;
CH04:   CP      "4"
        JP      NZ,CH05
        LD      HL,TRGCH4
        CALL    SNDOUT
        JR      GTR
;
CH05:   CP      "5"
        JP      NZ,CH06
        LD      HL,TRGCH5
        CALL    SNDOUT
        JR      GTR
;
CH06:   CALL    CRLF
        JP      STRIG
;
GTR:    LD      HL,CHRBUF   ;CLEAR CHR BUFFER
        LD      B,10H       ;10H BYTE
;
GTR1:   LD      (HL),00H    ;CLEAR ADDRESS 8000H-8010H
        INC     HL
        DJNZ    GTR1
        LD      HL,CHRBUF   ;HI POINT TO CHRBUF
;
GTR3:   PUSH    HL
        CALL    GETCHR      ;GET ONE CHR
        CALL    LC2UC       ;CONVERT TO UPPER CASE
        CALL    SNDCHR      ;ECHO BACK
        POP     HL
        CP      0DH         ;CR?
        JR      NZ,GTR2     ;IT IS NOT CR( ENTER )
        LD      (HL),0FFH   ;YES,CLOSE DATA WITH FFH
        JR      GTR4
;
GTR2:   CP      08H         ;BACK SPACE?
        JR      Z,GTR3      ;YES,GET AGAIN
        LD      (HL),A      ;NO SAVE CHR
        INC     HL
        JR      GTR3
;
GTR4:   LD      HL,CHSET    ;HEX
        LD      DE,CHRBUF
        CALL    MASKB
        IN      A,(PB)      ;DISABLE DECODER
        AND     0F0H
        OR      2DH
        OUT     (PB),A
        LD      A,(FTRIG)
        CP      "1"
        JP      Z,SNDCH1
        CP      "2"
        JP      Z,SNDCH2
        CP      "3"
        JP      Z,SNDCH3
        CP      "4"
        JP      Z,SNDCH4
        CP      "5"
        JP      Z,SNDCH5
        JP      WARM        ;TEST
;
;***********
;       Send    channal     1
;***********
SNDCH1: LD      A,(CHSET)   ;SEND WORD TRIG
        OUT     (PA),A
        LD      B,70H       ;SEND TO U5
        CALL    ENABLE
        CALL    DELAY1
        LD      A,(CHSET)
        CPL
        OUT     (PA),A
        LD      B,071H      ;SEND TO U6
        CALL    ENABLE
        CALL    CRLF
        JP      WARM
;
;***********
;       Send command 2
;***********
```

```
SNDCH2:  LD      A,(CHSET)  ;SEND WORD TRIG
         OUT     (PA),A
         LD      B,72H      ;SEND TO U12
         CALL    ENABLE
         CALL    DELAY1
         LD      A,(CHSET)
         CPL
         OUT     (PA),A
         LD      B,73H      ;SEND TO U13
         CALL    ENABLE
         CALL    CRLF
         JP      WARM
;
;**********
;        Send channal 3
;**********
SNDCH3:  LD      A,(CHSET)  ;SEND WORD TRIG
         OUT     (PA),A
         LD      B,74H      ;SEND TO U19
         CALL    ENABLE
         CALL    DELAY1
         LD      A,(CHSET)
         CPL
         OUT     (PA),A     ;SEND TO U20
         LD      B,75H
         CALL    ENABLE
         CALL    CRLF
         JP      WARM
;**********
;        Send channal 4
;**********
SNDCH4:  LD      A,(CHSET)  ;SEND WORD TRIG
         OUT     (PA),A
         LD      B,76H      ;SEND TO U26
         CALL    ENABLE
         CALL    DELAY1
         LD      A,(CHSET)
         CPL
         OUT     (PA),A
         LD      B,77H      ;SEND TO U27
         CALL    ENABLE
         CALL    CRLF
         JP      WARM
;**********
;        Send channal 5
;**********
SNDCH5:  LD      A,(CHSET)  ;SEND WORD TRIG
         OUT     (PA),A
         LD      B,78H      ;SEND TO U33
         CALL    ENABLE
         CALL    DELAY1
         LD      A,(CHSET)
         CPL
         OUT     (PA),A
         LD      B,79H      ;SEND TO U34
         CALL    ENABLE
         CALL    CRLF
         JP      WARM
;
ENABLE:  IN      A,(PB)     ;READ VALUE FROM PORT B
         AND     80H        ;CLEAR AT INTERES BIT
         OR      B          ;SELECT CHIP,DISABLE GATE
         OUT     (PB),A
         CALL    DELAY2
         AND     0DFH       ;ENABLE    GATE
         OR      20H
         OUT     (PB),A
         CALL    DELAY1     ;DELAY PULSE WIDTH
         AND     0C0H
         OR      7FH
         OUT     (PB),A
         LD      A,0FFH
         OUT     (PA),A
         RET
;
MASKB:   LD      A,(DE)     ;BIT 7
         CP      "0"
         JR      NZ,A70
         RES     7,(HL)
         JR      A72
;
A70:     CP      "1"
         JR      NZ,A71
         SET     7,(HL)
         JR      A72
;
A71:     RES     7,(HL)
         INC     HL
         SET     7,(HL)
         DEC     HL
;
A72:     INC     DE
         LD      A,(DE)     ;BIT 6
         CP      "0"
         JR      NZ,A60
         RES     6,(HL)
```

```
        JR      A62
;
A60:    CP      "1"
        JR      NZ,A61
        SET     6,(HL)
        JR      A62
;
A61:    RES     6,(HL)
        INC     HL
        SET     6,(HL)
        DEC     HL
;
A62:    INC     DE
        LD      A,(DE)      ;BIT 5
        CP      "0"
        JR      NZ,A50
        RES     5,(HL)
        JR      A52
;
A50:    CP      "1"
        JR      NZ,A51
        SET     5,(HL)
        JR      A52
;
A51:    RES     5,(HL)
        INC     HL
        SET     5,(HL)
        DEC     HL
;
A52:    INC     DE
        LD      A,(DE)      ;BIT 4
        CP      "0"
        JR      NZ,A40
        RES     4,(HL)
        JR      A42
;
A40:    CP      "1"
        JR      NZ,A41
        SET     4,(HL)
        JR      A42
;
A41:    RES     4,(HL)
        INC     HL
        SET     4,(HL)
        DEC     HL
;
A42:    INC     DE
        LD      A,(DE)      ;BIT 3
        CP      "0"
        JR      NZ,A30
        RES     3,(HL)
        JR      A32
;
A30:    CP      "1"
        JR      NZ,A31
        SET     3,(HL)
        JR      A32
;
A31:    RES     3,(HL)
        INC     HL
        SET     3,(HL)
        DEC     HL
;
A32:    INC     DE
        LD      A,(DE)      ;BIT 2
        CP      "0"
        JR      NZ,A20
        RES     2,(HL)
        JR      A22
;
A20:    CP      "1"
        JR      NZ,A21
        SET     2,(HL)
        JR      A22
;
A21:    RES     2,(HL)
        INC     HL
        SET     2,(HL)
        DEC     HL
;
A22:    INC     DE
        LD      A,(DE)      ;BIT 1
        CP      "0"
        JR      NZ,A10
        RES     1,(HL)
        JR      A12
;
A10:    CP      "1"
        JR      NZ,A11
        SET     1,(HL)
        JR      A12
;
A11:    RES     1,(HL)
        INC     HL
        SET     1,(HL)
        DEC     HL
```

```
;
A12:    INC     DE
        LD      A,(DE)      ;BIT 0
        CP      "0"
        JR      NZ,A00
        RES     0,(HL)
        JR      A02
;
A00:    CP      "1"
        JR      NZ,A01
        SET     0,(HL)
        JR      A02
;
A01:    RES     0,(HL)
        INC     HL
        SET     0,(HL)
        DEC     HL
;
A02:    INC     DE
        RET
;
TRGPAT:
        DFB  CR,LF,"Set Trigger bit 1, for logic 1 ,0 for logic 0 and don't
care"
        DFB  CR,LF, "Cannel 1-8   for bit7..... bit 0  ",CR,LF
        DFB  CR,LF, "Cannel 9-16  for bit7..... bit 0  ",CR,LF
        DFB  CR,LF, "Cannel 17-24 for bit7..... bit 0  ",CR,LF
        DFB  CR,LF, "Cannel 25-32 for bit7..... bit 0  ",CR,LF
        DFB  CR,LF, "Cannel 33-40 for bit7..... bit 0  ",CR,LF
        DFB  CR,LF, "Select channel for setting  : ",04H
;
TRGCH1: DFB  CR,LF,"Set trigger bit of CH1  : ",04H
TRGCH2: DFB  CR,LF,"Set trigger bit of CH2  : ",04H
TRGCH3: DFB  CR,LF,"Set trigger bit of CH3  : ",04H
TRGCH4: DFB  CR,LF,"Set trigger bit of CH4  : ",04H
TRGCH5: DFB  CR,LF,"Set trigger bit of CH5  : ",04H
;
;*******
;       Send add hex in first,first+1 to terminal
;*******
;
SNDADD: LD      HL,FIRST    ;send start address
        LD      A,(HL)
        CALL    BN2HEX      ;convert hi add
        LD      A,H
        PUSH    HL
        CALL    SNDCHR      ;send hi add
        POP     HL
        LD      A,L
        CALL    SNDCHR
        LD      HL,FIRST+1  ;next convert lo address
        LD      A,(HL)
        CALL    BN2HEX
        LD      A,H
        PUSH    HL
        CALL    SNDCHR      ;send lo out
        POP     HL
        LD      A,L
        CALL    SNDCHR
        RET
;
AMAP:
        DFB     CR,LF,"0 1 2 3 4 5 6 7  8 9 "
        DFB     "A B C D E F ,CR,LF,04H"
;
;**********
;       Send space to Terminal
;**********
;       NEXT
;
SPACE4: CALL    SPACE3      ;send 4 space to terminal
SPACE3: CALL    SPACE2      ;send 3 space to terminal
SPACE2: CALL    SPACE1      ;send 2 space to terminal
SPACE1: LD      A,20H       ;send 1 space to terminal
        CALL    SNDCHR
        RET
;
;*******
;       Send MSSSG to Terminal, MSSSG point by  HL
;*******
;
SNDOUT: LD      A,(HL)      ;get data
        CP      04H         ;eot
        RET     Z
        PUSH    BC
        PUSH    AF
        LD      B,00H
;
SOUT1:  LD      C,05H       ;
        IN      A,(C)
        AND     02H
        JR      Z,SOUT1
        POP     AF
        LD      C,07H
        OUT     (C),A       ;send out
        POP     BC
```

```
        INC     HL
        JR      SNDOUT
;
;***********
;       Send one character to terminal
;       Data in acc
;***********
;
SNDCHR: PUSH    BC
        PUSH    AF
        LD      B,00H
SOUT2:  LD      C,05H
        IN      A,(C)
        AND     02H
        JR      Z,SOUT2
        POP     AF
        LD      C,07H
        OUT     (C),A       ;SEND OUT
        POP     BC
        RET
;
        END
```