อุปกรณ์ตรวจสอบการผิดพลาดและข้อความผิดเพี้ยนของการรับ-ส่งข้อมูลแบบอะซิงโครนัส

1. นาย วิริยะ จริยวิทยานนท์

2. นาย สมชาย ไกรศรีสิริกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

สาขา เทคโนโลยีอิเล็กทรอนิกส์

ภาควิชา เทคนิคอุตสาหกรรม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

BIT ERROR AND DISTORTION METER FOR ASYNCHRONOUS COMMUNICATION

1. MR. WIRIYA JARIYAVIDYANONT

2. MR. SOMCHAI KRAISRISIRIKUL

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIRMENTS FOR THE BACHELOR'S DEGREE

DEPARTMENT OF INDUSTIAL TECHNOLOGY

FACULITY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

หัวข้อปริญญานิพนธ์ อุปกรณ์ตรวจสอบการผิดพลาดและข้อความผิดเพี้ยน
ของการรับ-ส่งข้อมูลแบบอะซิงโครนัส

ชื่อนักศึกษา 1. นาย วิริยะ จริยวิทยานนท์
2. นาย สมชาย ไกรศรีสิริกุล

อาจารย์ที่ปรึกษา อ. ชวลิต เบญจางคประเสริฐ

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2536

ภาคเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า
เจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์นี้
เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา อุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... ประธานกรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

PROJECT REPORT BIT ERROR AND DISTORTION METER
FOR ASYNCHRONOUS COMMUNICATION

BY 1. MR. WIRIYA JARIYAVIDYANONT
2. MR. SOMCHAI KRAISRISIRIKUL

PROJECT REPORT ADVISOR
MR. CHAOVALIT BENJANGKAPRASERT

DEPARTMENT INDUSTRIAL TECHNOLOGY

ACADEMIC YEAR 1993

ACCEPTED BY THE FACULITY OF ENGINEERING , KING MONGKUT'S INSTITUTE OF TECHNOLOGY , LADKRABANG IN PARTIAL FULFILLMENT OF THE REQUIRMENTS FOR THE BACHELOR'S DEGREE

PROJECT REPORT COMITTEE

..... MEMBER
(.....)

..... MEMBER
(.....)

..... MEMBER
(.....)

..... MEMBER
(.....)

หัวข้อปริญญานิพนธ์ อุปกรณ์ตรวจสอบการผิดพลาดและข้อความผิดเพี้ยน
ของการรับ-ส่งข้อมูลแบบอะซิงโครนัส

ชื่อนักศึกษา 1. นาย วิริยะ จริยวิทยานนท์
2. นาย สมชาย ไกรศรีสิริกุล

อาจารย์ที่ปรึกษา อ. ชวลิต เบญจางคประเสริฐ

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2536

บทคัดย่อ

อุปกรณ์ตรวจสอบการผิดพลาดของการรับ-ส่งข้อมูล เป็นอุปกรณ์ตรวจสอบความผิดพลาดของเครื่องมือสื่อสารที่ใช้ในการรับ-ส่งข้อมูลแบบอะซิงโครนัส ตามรูปแบบมาตรฐานของ CCITT ที่มีความเร็วต่ำประมาณ 50 ถึง 300 บิทต่อวินาที โดยอาศัยหลักการส่งข้อมูลที่สร้างขึ้นส่งออกไปยังอุปกรณ์รับ-ส่งของเครื่องมือสื่อสารแล้วย้อนกลับเข้ามายังอุปกรณ์ตรวจสอบการผิดพลาดของการรับ - ส่งข้อมูล นำข้อมูลที่ได้มาเปรียบเทียบกับข้อมูลที่ส่งออกไป การแสดงผลจะบอกถึงจำนวนอักขระที่ผิดพลาดและจำนวนอักขระที่ส่งออกไป

ในส่วนเครื่องวัดความผิดเพี้ยน จะนำข้อมูลที่รับได้มาเปรียบเทียบกับเวลามาตรฐานที่สร้างขึ้น นำค่าความแตกต่างที่ได้มาแสดงผลมีหน่วยเป็นเปอร์เซ็นต์

เนื่องจากการรับ-ส่งข้อมูล ในกิจการบางประเภทมีระยะทางไกล ซึ่งมีทั้งในประเทศและระหว่างประเทศ เช่น ข่าวอากาศของกรมอุตุนิยมวิทยา ข่าวแผ่นการบินที่ใช้ในการควบคุมจราจรทางอากาศ ซึ่งข้อมูลเหล่านี้มีความสำคัญมาก ถ้าข้อมูลเกิดความผิดพลาดจะทำให้เกิดความเสียหาย หรือเกิดความล่าช้าในกิจการนั้นได้ ดังนั้น จึงต้องมีการตรวจสอบอุปกรณ์ต่างๆ อย่างสม่ำเสมอ เพื่อให้การสื่อสารข้อมูลเป็นไปได้อย่างต่อเนื่อง และมีความผิดพลาดน้อยที่สุด อนึ่งราคาในการซื้ออุปกรณ์ชนิดนี้จากต่างประเทศยังมีราคาสูงมาก แต่อุปกรณ์ชุดนี้สามารถสร้างได้ในราคาถูก

PROJECT REPORT BIT ERROR AND DISTORTION METER
FOR ASYNCHRONOUS COMMUNICATION

BY 1. MR. WIRIYA JARIYAVIDYANONT
2. MR. SOMCHAI KRAISRISIRIKUL

PROJECT REPORT ADVISOR
MR. CHAOVALIT BENJANGKAPRASERT

DEPARTMENT INDUSTRIAL TECHNOLOGY

ACADEMIC YEAR 1993

ABSTRACT

Bit Error and Distortion meter for Asynchronous Communication

Bit Error and Distortion meter is an equipment for detecting the error in Asynchronousedata Communication System. Its specifications refer to CCITT standards , which operates in low speed approximate 50-300 bit /second by transmitting its generated data to thetransmtitter and receiver instrumen then looping that data back to Bit Error and Distortion meter equipment. Pick out the recieved data to compare with the transmitted data then displays it in numeric to show the amount of Error Alphabet and Transmitted Alphabet.Distortion meter part will take received data to compare with generated standard time then display it in percent.

Since data transmitting and receiving in some activity takes long distance which can divided into domestic and international such as MET , Flight plan for airtrafficcontrol. These data is an importand information therefore, if there are somthing error , it will damage or relay the activity ,thus the etecing shoud be operated continuously for the most effective operation. Moreover, this equipment is cheaper than the imported equipment.

สารบัญ

บทที่ 1

8749

ไมโครคอมพิวเตอร์ชิพเดียว

เมื่อ 8749 คือ เบอร์ไอซีประเภทไมโครคอมพิวเตอร์ชิพเดียว ที่ได้เข้าสู่วงการคอมพิวเตอร์ในเมืองไทย เมื่อหลายๆ ปีก่อน และได้มีการประยุกต์ไปใช้งานในหลายๆ ด้าน จนเป็นที่รู้จักกันดีว่า 8749 คือ ไอซีเอนกประสงค์ตัวหนึ่ง

ไอซีตระกูล 8749 นี้จัดว่าเป็นชิพที่ค่อนข้างจะมีความพร้อม หรือ ความสมบูรณ์อยู่ในตัวมันเองแล้วคือไม่เพียงแต่จะประกอบด้วยส่วนซีพียูอย่างเดียว แต่ยังมีส่วนของหน่วยความจำ,ส่วนของพอร์ท , ส่วนไทเมอร์ เคาวน์เตอร์ และอื่นๆ อีกมากมาย ซึ่งทั้งหมดนี้รวมอยู่ในชิพ ซึ่งมีราคาไม่แพงเลย ก็มีหลายคนที่นำมันไปประยุกต์ในงานต่างๆ และก็ได้รับความสำเร็จไปตามๆ กัน ดังจะดูได้จากบทความในวารสารต่างๆ เช่น ทำนาฬิกาที่สามารถตั้งโปรแกรมได้, เครื่องช่วยหมุนโทรศัพท์, เครื่องกันขโมย, คีย์บอร์ดของเครื่องคอมพิวเตอร์ เป็นต้น

ขาสัญญาณต่างๆ ของ 8749

ขาสัญญาณต่างๆ ของ 8749 ที่แสดงในรูปที่ 1 นั้นมีหน้าที่ต่างกันไป ดังจะกล่าวรวมๆ อย่างย่อๆ ดังนี้

1 T_0 เป็นขาอินพุททดสอบ หรือใช้เป็นขาเอาท์พุท สัญญาณนาฬิกาของระบบ

2 $XTAL_1$ เป็นขาอินพุทของสัญญาณนาฬิกาที่ใช้เป็นฐานเวลาของระบบ หรืออาจจะต่อคริสตอลเข้ากับขา $XTAL_1$ และ $XTAL_2$ ก็ได้

3 $XTAL_2$ ต่อคริสตอลกับขานี้ เพื่อให้วงจรออสซิเลเตอร์ภายในทำงานได้

4 RESET เป็นสัญญาณอินพุทแอคทีฟศูนย์ ใช้รีเซทระบบในตัว 8749

5 SS เป็นสัญญาณอินพุทแอคทีฟศูนย์ ที่ทำงานร่วมกับ ALE เวลาตรวจสอบโปรแกรมด้วยวิธี SYNGLE STEP

6 INT เป็นอินพุทของสัญญาณอินเตอร์รัพท์ หรือใช้เป็นขาอินพุทธรรมดาก็ได้

7 EA เป็นสัญญาณอินพุท เพื่อเลือกให้ 8749 รันโปรแกรมในหน่วยความจำโปรแกรมภายใน หรือหน่วยความจำภายนอก

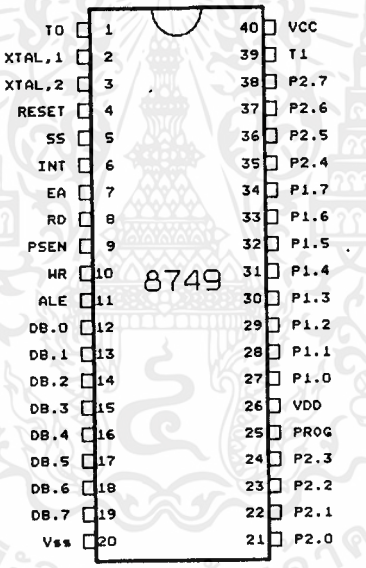
8 RD เป็นสัญญาณเอาท์พุทที่แอคทีฟศูนย์ เมื่อ 8749 อ่านข้อมูลจากหน่วยความจำข้อมูลที่ต่ออยู่ภายนอก

9 PSEN เป็นสัญญาณเอาท์พุทที่แอคทีฟศูนย์ เมื่อ 8749 เฟทช์คำสั่งจากหน่วยความจำโปรแกรมที่อยู่ภายนอก

10 WR เป็นสัญญาณเอาท์พุทที่แอคทีฟศูนย์ เมื่อ 8749 เขียนข้อมูลลงในหน่วยความจำโปรแกรมที่อยู่ภายนอก

11 ALF เป็นสัญญาณที่ใช้แลทช์แอดเดรส มัลติเพล็กซ์มาทางดาต้าบัส และใช้แลทช์ข้อมูลทางพอร์ท 2 ด้วย

12-19 $D8_0$-$D8_7$ ดาต้าบัส เป็นทางผ่านเข้าออกของข้อมูล และแอดเดรส



รูปที่ 1 แสดงการจัดขาของ 8749

20 V_{ss} กราวนด์ของระบบ 8749

21-24 P_{20}-P_{23} 4 บิทล่างของพอร์ท 2

25 PROG เป็นสัญญาณ STROBE สำหรับการต่อขยายพอร์ทด้วย 8243

26 V_{dd} ต่อกับแบตตารี่ขนาดเล็ก เพื่อเลี้ยงแรมภายใน ไม่ให้ข้อมูลสูญหายเมื่อไม่มีไฟเลี้ยง V_{cc}

27-34 P_{10}-P_{17} พอร์ท 1

35-38 P_{24}-P_{27} 4 บิทบนของพอร์ท 2

39 T_1 เป็นขาอินพุททดสอบ หรือใช้เป็นขาอินพุทของสัญญาณนาฬิกาที่ป้อนให้แก่ไทเมอร์/เคาวน์เตอร์ ภายในตัว 8749

40 V_{CC} ต่อกับไฟเลี้ยง +5 โวลต์

ส่วนประกอบต่างๆภายใน 8749

ระบบไมโครคอมพิวเตอร์ชิพเดียวของ 8749 นั้นประกอบด้วยส่วนสำคัญ 3 ส่วนด้วยกัน คือ ส่วน ซีพียู, หน่วยความจำ และหน่วยอินพุท เอาท์พุท ซึ่งแต่ละส่วนจะมีรายละเอียดดังนี้

หน่วยความจำ

ในระบบของ 8749 นี้ได้มีการแบ่งหน่วยความจำออกเป็นสองลักษณะตามชนิดของข้อมูลที่เก็บ คือ

1. หน่วยความจำข้อมูล
2. หน่วยความจำโปรแกรม

ถ้าจะพูดกันง่ายๆ แล้ว หน่วยความจำข้อมูลก็อาจจะหมายถึง หน่วยความจำส่วนที่เป็นแรม (RAM) ที่เราสามารถเขียน หรืออ่านข้อมูลจากหน่วยความจำส่วนนี้ได้ แต่ไม่สามารถจะรันโปรแกรมบนหน่วยความจำนี้ได้เลย

ส่วนหน่วยความจำโปรแกรมก็จะหมายถึง หน่วยความจำที่ภายในจะบรรจุโปรแกรม หรือข้อมูลที่เป็นค่าคงที่เอาไว้

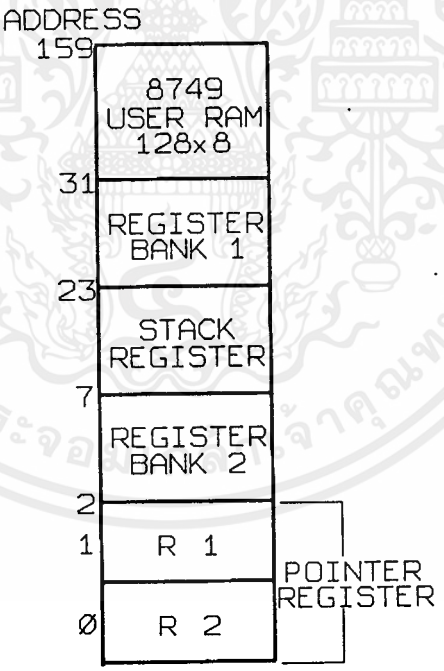
หน่วยความจำทั้งสองประเภทนี้ถูกแบ่งแยกออกจากกันด้วยคำสั่งทางซอฟท์แวร์ และ ลักษณะการติดต่อทางฮาร์ดแวร์ด้วย กล่าวคือ มีคำสั่งเฉพาะสำหรับติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่ง และมีการจัดสัญญาณ STROBE ในการติดต่อกับหน่วยความจำแต่ละชนิด แยกต่างหากกันด้วย

หน่วยความจำข้อมูล

สำหรับหน่วยความจำข้อมูลภายในตัว 8749 นั้นมีด้วยกัน 128 ไบท์ ซึ่งในจำนวนไบท์ทั้งหมดนี้จะแบ่งออกเป็น 2 ส่วน ดังแสดงในรูป 2

ส่วนรีจีสเตอร์ ได้แก่ หน่วยความจำแอดเดรส 00-07 ซึ่งเราเรียกว่า รีจีสเตอร์แบงค์ 1 (BANK 1) แต่ละแบงค์จะมีรีจีสเตอร์อยู่ 8 ตัว ซึ่งมีชื่อเรียกว่า R_0-R_7 ในการใช้งานรีจีสเตอร์เหล่านี้ เราจะสามารถใช้งานได้ทีละแบงค์เท่านั้น โดยใช้คำสั่ง SEL RB_9 หรือ SEL RB_1 ในการเลือกแบงค์ของรีจีสเตอร์

รีจีสเตอร์ทั้ง 8 ตัวนี้เป็นรีจีสเตอร์สำหรับใช้งานทั่วไป มีคำสั่งต่างๆ ที่กระทำร่วมกับรีจีสเตอร์เหล่านี้ครบ แต่จะมีรีจีสเตอร์ R_0 และ R_1 สองตัวเท่านั้น ที่มีหน้าที่พิเศษ คือ สามารถใช้เป็นตัวอ้างถึงหน่วยความจำข้อมูลได้ โดยค่าแอดเดรสจะมีขนาดเพียง 8 บิทเท่านั้น เป็นการจำกัดหน่วยความจำข้อมูลให้มีเพียง 256 ไบท์ ซึ่งจะคลุมพื้นที่ส่วนที่เป็นรีจีสเตอร์ และส่วนที่เป็นแสตคด้วย ทำให้เราสามารถติดต่อกับรีจีสเตอร์ หรือแสตคในลักษณะที่เป็นหน่วยความจำข้อมูล ได้อีกวิธีหนึ่ง การติดต่อด้วยวิธีนี้ส่วนใหญ่จะใช้กับแสตคมากกว่า ส่วนอื่น เพราะว่าคำสั่งต่างๆ ของ 8749 ที่เกี่ยวข้องกับแสตคโดยตรงมีเพียงคำสั่ง CALL กับ คำสั่ง RET (RETR) เท่านั้น

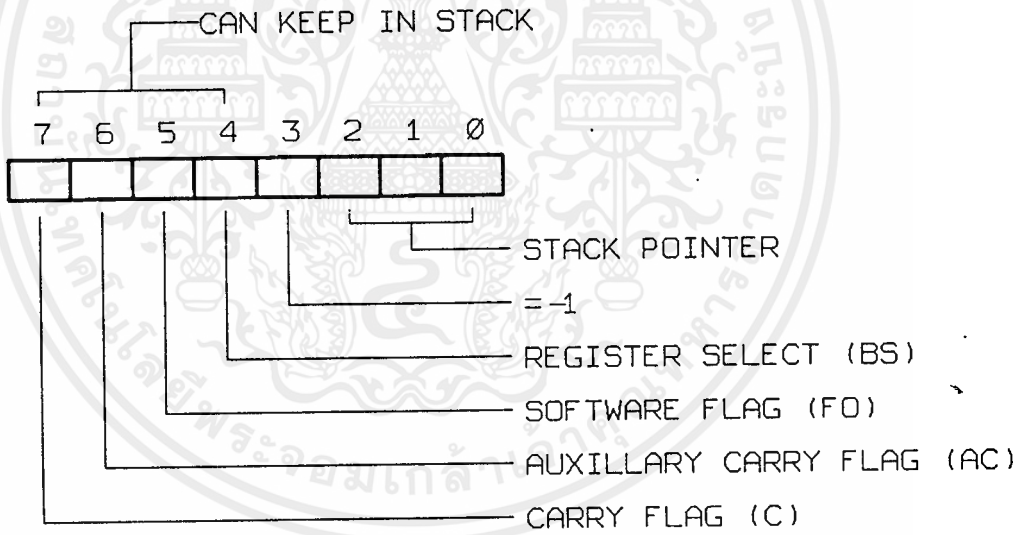


รูปที่ 2 แสดงการแบ่งหน่วยความจำข้อมูลภายในไปใช้งาน

ส่วนของแสตค ประกอบด้วยแสตคทั้งหมด 8 ชั้น แต่ละชั้นประกอบด้วยหน่วยความจำข้อมูล 2 ไบท์ เพื่อให้เพียงพอกับการเก็บค่าแอดเดรส และเก็บสถานะของโปรแกรม การที่มีแสตคเพียง 8 ชั้นนี้ทำให้มีขีดจำกัดในการเรียดโปรแกรมย่อย (SUBROUTINE) ซ้อนๆ กันได้ไม่เกิน 8 ชั้น ถ้าหากเกิน แสตคพอยเตอร์ก็จะย้อนกลับไปชี้ที่แสตคชั้นแรกใหม่

โดยปกติแล้วแสตคพอยเตอร์จะชี้ที่ S_0 เป็นแสตคชั้นแรก เมื่อมีการ CALL หรือการอินเตอร์รัพท์ ก็จะเก็บค่าแอดเดรส และสถานะของโปรแกรม (PROGRAM STATUS WORD) ลงในแสตคแล้วเพิ่มค่าแสตคพอยเตอร์ให้ชี้ไปที่ S_1 และเมื่อเอ็กซีคิ้วท์คำสั่ง RET (RETR) ก็จะลดค่าแสตคพอยเตอร์ลงไปชี้ที่ S_0 แล้ว นำค่าแอดเดรส กับสถานะของโปรแกรมที่เก็บไว้ในแสตค S_0 กลับคืนที่เดิม

PROGRAM STATUS WORD หรือ PSW เป็นรีจีสเตอร์ตัวหนึ่ง ที่ทำหน้าที่เก็บค่าของแสตคพอยเตอร์ และสถานะของโปรแกรม (สถานะของโปรแกรม ก็คือ ค่าของแฟลกต่างๆ) แต่ละบิทของ PSW มีความหมายดังรูปที่ 3



รูปที่ 3 แสดงความหมายของแต่ละบิทใน PSW

C หมายถึง แฟลกตัวทดที่ได้จากการบวก หรือโดยการเลื่อนบิท

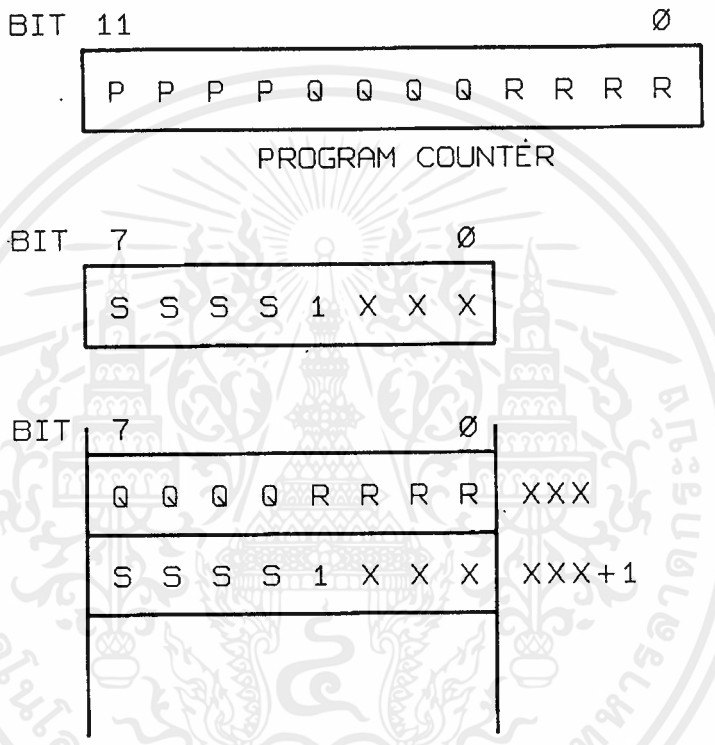
AC หมายถึง แฟลกตัวทดที่ได้จากการบวกบิทที่ 3 ของตัวเลข BCD จะใช้ทำงานร่วมกับคำสั่ง DA (DECIMAL ADJUST)

FO หมายถึง ซอฟท์แวร์แฟลกสามารถใช้งานได้ทั่วไป คำสั่งที่มีผลต่อแฟลกนี้ได้แก่ คำสั่ง CLR FO และ CPL FO

BS หมายถึง แฟลกที่แสดงการเลือกแบงค์ของรีจีสเตอร์

นอกจากแฟลก F0 แล้วยังมีซอฟท์แวร์แฟลก F1 ไม่ได้เก็บไว้ใน PSW ด้วย

บิท 0 - บิท 2 ของ PSW จะเก็บค่าของแสตคพอยเตอร์เอาไว้ เมื่อมีการเรียกโปรแกรมย่อยไม่ว่าจะโดยการCALL หรือโดยการเรียกอินเตอร์รัพท์ จะมีการเก็บค่าแอดเดรสจากโปรแกรมเคาวน์เตอร์ และค่าของ PSW ลงในแสตค ในลักษณะการจัดเรียงดังรูปที่ 4



รูปที่ 4 แสดงการเก็บค่าโปรแกรมเคาน์เตอร์ และ PSW

แต่สำหรับการ RETURN นั้น จะแบ่งออกเป็นสองแบบ คือ คำสั่ง RET ใช้สำหรับออกจากโปรแกรมย่อย จะเอาเฉพาะค่าแอดเดรสใส่คืนให้แก่โปรแกรมเคาน์เตอร์เท่านั้น ส่วนคำสั่ง RETR นั้นใช้สำหรับออกจาก INTERUPT SERVICE ROUTINE (ISR) ซึ่งจะนำเอาทั้งค่า แอดเดรส และสถานะของโปรแกรม ใส่กลับคืนที่เดิม

นอกจากหน่วยความจำข้อมูลภายในตัว 8749 แล้ว เรายังสามารถต่อขยายหน่วยความจำแรม ภายนอกได้อีก 256 ไบท์ สำหรับการติดต่อกับหน่วยความจำข้อมูลภายนอก จะยังคงใช้รีจีสเตอร์ R_0 และ R_1 ในการอ้างถึงแอดเดรสเช่นเดิม จะต่างกันตรงที่มีคำสั่งในการติดต่อกับหน่วยความจำข้อมูล แยกจากกัน คือ

ใช้คำสั่ง MOV ในการติดต่อกับหน่วยความจำข้อมูลภายใน

ใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก

หน่วยความจำโปแกรม

เนื่องจากเราไม่สามารถรันโปรแกรมใดๆ ในพื้นที่หน่วยความจำข้อมูลดังนั้น 8749 จึงจัดหน่วยความจำขนาด 4 KBYTE ไว้ภายนอกทั้งหมด

8749 สามารถเลือกที่จะรันโปรแกรมในหน่วยความจำโปรแกรมภายนอก หรือภายในก็ได้ โดยการส่งสัญญาณเลือกไปที่ขาควบคุม EA(EXTERNAL ACCESS)

ถ้าขา EA ได้รับลอจิกศูนย์ 8749 จะรันโปรแกรมในหน่วยความจำภายใน แต่เนื่องจากมีพื้นที่หน่วยความจำเพียง 1-2 KBYTE เท่านั้น จึงอาจจะรันโปรแกรมต่อที่หน่วยความจำภายนอกได้ แต่ถ้าหาก EA ได้รับลอจิกหนึ่ง 8749 ก็จะรันโปรแกรมในหน่วยความจำโปรแกรมภายนอกทั้งหมด

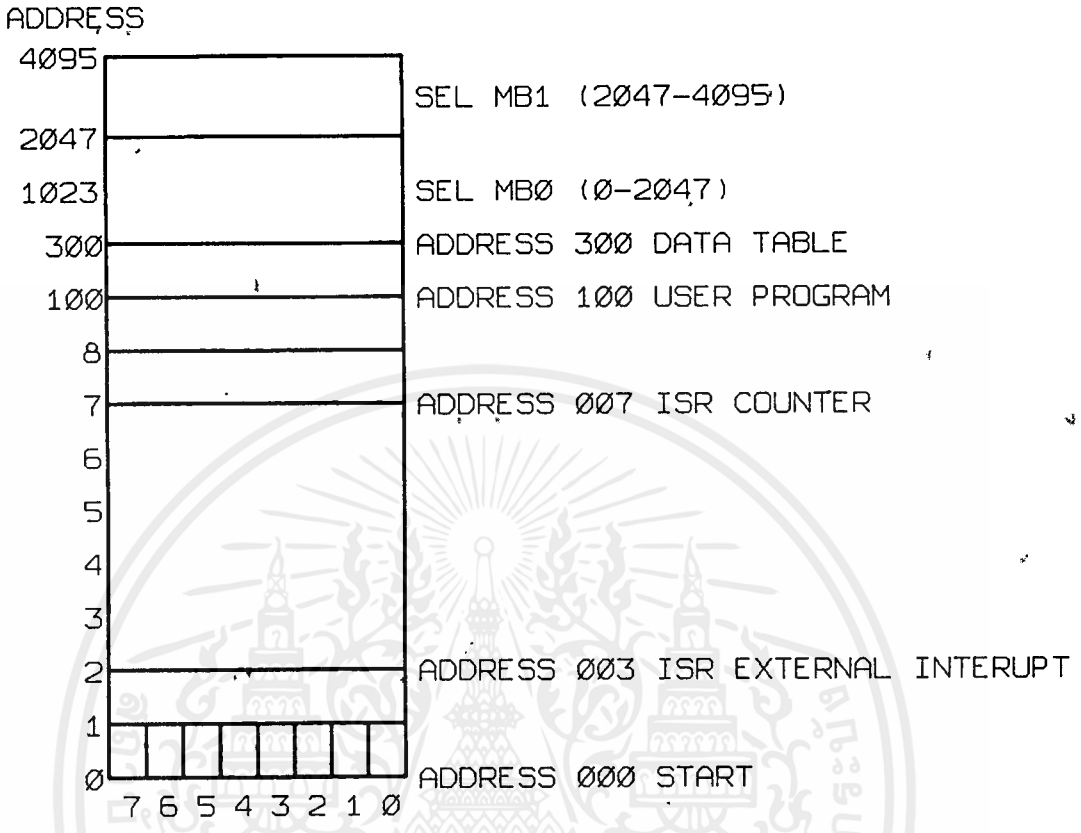
ลักษณะพิเศษของหน่วยความจำโปร์แกรมอีกอย่างหนึ่งก็คือ การแบ่งพื้นที่หน่วยความจำออกเป็นส่วนย่อยๆ หลายๆ ส่วน ดังนี้

แบ่งหน่วยความจำโปรแกรมขนาด 4 KBYTE ออกเป็น 2 แบลงค์ (BLANK) แบลงค์ละ 2 KBYTE และในแต่ละแบลงค์ยังแบ่งออกเป็น 8 เพจ (PAGE) ย่อยๆ เพจละ 256 ไบท์ การแบ่งหน่วยความจำออกเป็นส่วนๆ เช่นนี้ก็เพื่อให้มีการอ้างถึงแอดเดรสได้ง่ายในลักษณะ PAHE ADDRESSING MODE ทำให้คำสั่งมีขนาดสั้นลง นั่นคือ สามารถประหยัดเนื้อที่หน่วยความจำ (ซึ่งมีน้อยอยู่แล้ว) และสามารถทำงานได้รวดเร็วขึ้น

คำสั่งที่มี PAGE ADDRESSING MODE เช่น คำสั่ง JMPP OA หรือ คำสั่ง MOVP A OA จะไม่สามารถทำงานออกนอกเพจไปได้เลย แต่สำหรับคำสั่งที่มี ADDRESSING MODE เป็นอย่างอื่น เช่นคำสั่ง JMP หรือ CALL นั้นสามารถกระโดดข้ามไปยังเพจใดๆ ก็ได้ แต่อย่างไรก็ตาม ก็ยังไม่สามารถกระโดดข้ามเขตแบลงค์ไปได้อยู่ดี

การจะข้ามจากแบลงค์หนึ่งไปยังอีกแบลงค์นั้น ก็คือ การเซท หรือรีเซทบิทA_{11} ของโปรแกรมเคาวน์เตอร์นั่นเอง คำสั่งที่ทำหน้าที่นี้ก็คือ คำสั่ง SEL MB_0 หรือ SEL MB_1 ซึ่งจะต้องตามหลังด้วยคำสั่ง JMP หรือ CALL

ในหน่วยความจำข้อมูลภายในนั้น ได้แสดง MEMORY MAP ให้ดูด้วยว่า 8749 ใช้หน่วยความจำส่วนใดทำหน้าที่อะไร สำหรับหน่วยความจำโปรแกรมก็เช่นกัน จะมีการแบ่งพื้นที่ของหน่วยความจำโปรแกรม ไปใช้ในงานต่างๆ กันด้วย ดังรูปที่ 5



รูปที่ 5 แสดงการแบ่งพื้นที่หน่วยความจำ

1. แอดเดรส $0000 เป็นจุดเริ่มต้นของโปรแกรมหลังจากการรีเซท

2. แอดเดรส $0003 เป็นจุดเริ่มต้นของ INTERUPT SERVICE RUOTINE (ISR) ของการอินเตอร์รัพท์ จากภายนอกที่เข้ามาทางขา INT

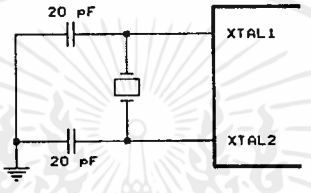
3. แอดเดรส $0007 เป็นจุดเริ่มต้นของ ISR ของการอินเตอร์รัพท์ จากตัวไทเมอร์/เคาวน์เตอร์ที่อยู่ภายใน 8749

4. แอดเดรส $0100 เป็นจุดเริ่มต้นของ USER PROGRAM แต่อาจจะไม่จำเป็นต้องเริ่มที่นี่เสมอไปก็ได้

5. แอดเดรส $0300-$03FF เป็นที่เก็บข้อมูลค่าคงที่ต่างๆ จำนวน 256 ไบท์ เราสามารถอ่านข้อมูลมาใช้ได้โดยสะดวก ด้วยคำสั่งที่ออกแบบมา ใช้กับหน่วยความจำส่วนนี้ โดยเฉพาะคือ คำสั่ง MOVP3 A,@A

สัญญาณนาฬิกาของระบบและสัญญาณควบคุม

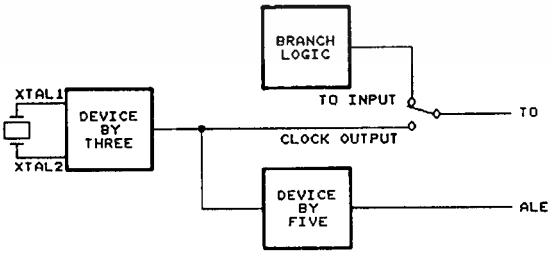
สัญญาณนาฬิกาที่เป็นฐานเวลาของระบบนี้อาจจะสร้าง จากวงจรออสซิเลเตอร์ภายนอกแล้วป้อนให้แก่ 8749 ทางขา $XTAL_1$ แต่ภายใน 8749 เองก็มีวงจรออสซิเลเตอร์อยู่ภายในอยู่แล้ว เพียงแต่คุณต่อคริสตอล และตัวเก็บประจุอีก 2 ตัว ดังรูปที่ 6 ก็จะทำให้มีสัญญาณนาฬิกาใช้เช่นกัน สำหรับความถี่ของสัญญาณนาฬิกานี้ขึ้นกับความถี่ของงคริสตอลซึ่งมีให้เลือก2 รุ่น คือ



รูปที่ 6 แสดงการต่อวงจรภายนอกเพื่อสร้างฐานเวลา

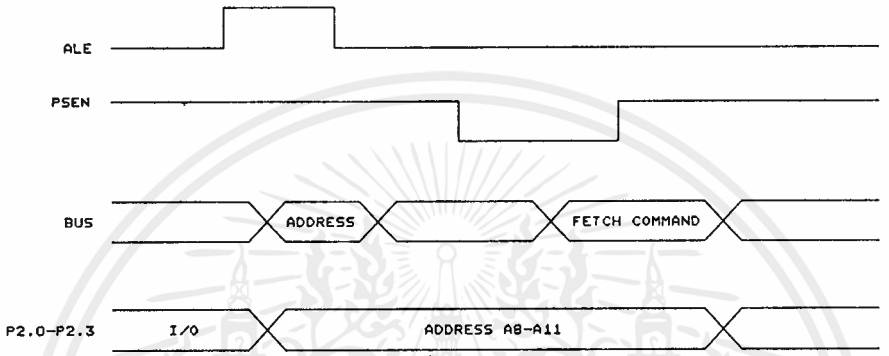
คริสตอลขนาด 1 MHz - 6 MHz โดยมากจะใช้ความถี่ 6 MHz กับชิพในอนุกรม 8749 ทั่วๆ ไป และจะใช้คริสตอลขนาด 4 MHz - 11 MHz กับไอซี 8039 และ8049

สัญญาณนาฬิกาที่ได้จากคริสตอล (OSC) ถูกหารความถี่ด้วย 3 ได้เป็นสัญญาณนาฬิกาของระบบ (CLK) เรานำสัญญาณ CLK นี้ไปหารความถี่ด้วย 5 จะได้เป็นสัญญาณ ALE ดังรูปที่ 7 ซึ่งเป็นบล็อกไดอะแกรม ของระบบสัญญาณนาฬิกาภายใน 8749



รูปที่ 7 แสดงระบบสัญญาณนาฬิกาภายใน 8749

นอกจากสัญญาณนาฬิกาที่ใช้เป็นหลักฐานเวลาของระบบแล้ว 8749 ก็ยังต้องการสัญญาณจากภายนอก เพื่อมาควบคุมการทำงาน เช่น สัญญาณ RESET, EA, INT,SS เป็นต้น และในทางตรงกันข้าม ก็ต้องส่งสัญญาณควบคุม ไปยังอุปกรณ์ภายนอกเช่นกัน สัญญาณเหล่านั้น ได้แก่ ALE, PSEN, WR, RD เป็นต้น เราจะมาทำความรู้จักกับสัญญาณควบคุมเหล่านี้กันเลย โดยเริ่มจาก



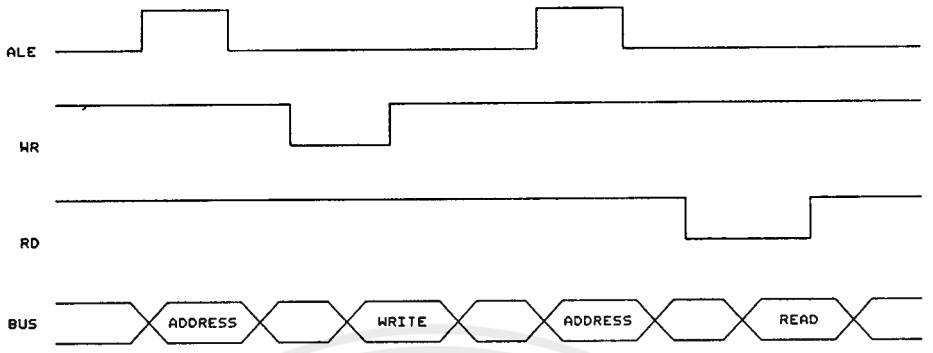
รูปที่ 8 แสดง TIMING DIAGRAM ของการเฟทช์คำสั่ง

สัญญาณควบคุมเอาท์พุท ได้แก่ สัญญาณ ALE, PSEN, WR, RD

ALE (ADDRESS LATCH ENABLE) เป็นสัญญาณนาฬิกาที่ออกมาทุกๆ แมชชีนไซเคิล (MACHINE CYCLE) หรือทุกๆ สัญญาณ CLK 5 ลูกตามที่กล่าวมาแล้ว เราจะใช้ขอบขาลงของสัญญาณ ALE ในการแลทช์ค่าของแอดเดรสไบท์ต่ำที่มัลติเพล็กซ์มาทางดาต้าบัส

PSEN (PROGRAM STORE ENABLE) เป็นสัญญาณที่แอคทีฟศูนย์และจะแอคทีฟเฉพาะเมื่อ 8749 เฟทช์คำสั่งจากหน่วยความจำโปรแกรมภายนอกเท่านั้น

ในการต่อหน่วยความจำโปรแกรมภายนอกเข้ากับ 8749 นั้น เราจำเป็นต้องแลทช์ค่าแอดเดรสที่มัลติเพล็กซ์มาทางดาต้าบัสเสียก่อน โดยใช้สัญญาณ ALE เป็น STROBE (ไอซีที่เรานิยมใช้แลทช์ค่าแอดเดรสนี้เป็น ไอซี 74LS373) ค่าแอดเดรส 8 บิทที่ได้จากการแลทช์ เมื่อนำมารวมกับแอดเดรส 4 บิทสูง (A_8-A_{11}) ซึ่งได้จากพอร์ท 2 P_{20}-P_{23} ก็จะได้แอดเดรสขนาด 12 บิท ที่จะใช้อ้างอิงหน่วยความจำโปรแกรมได้ครบ 4 KBYTE จากนั้นเราใช้สัญญาณ PSEN เป็น STROBE ในการเฟทช์คำสั่งจากหน่วยความจำเข้าไปยัง 8749 ได้



รูปที่ 9 แสดง TIMING DIAGRM ของการเขียน หรืออ่านข้อมูล

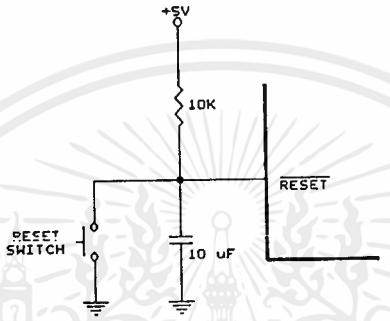
WR และ RD เป็นสัญญาณที่แอคทีฟศูนย์เช่นกัน แต่จะแอคทีฟเมื่อ 8749 มีการติดต่อกับหน่วยความจำข้อมูลภายนอก หรือเมื่อเราใช้ดาต้าบัสมาทำหน้าที่เป็นพอร์ท เราก็ใช้สัญญาณ WR และ RD ที่แอคทีฟมาเป็น STROBE ในการเขียน หรืออ่านข้อมูลกับทางบัสพอร์ทได้ด้วย

จากที่กล่าวมาแล้ว พอจะสรุปเรื่องราวของสัญญาณ PSEN, WR, RD ได้ว่า สัญญาณ STROBE ทั้งสามนี้ จะสามารถแอคทีฟได้เพียงครั้งละ หนึ่งสัญญาณเท่านั้น เช่น ในช่วงการเฟทช์คำสั่ง สัญญาณ PSEN จะแอคทีฟ ส่วน WR และ RD จะไม่แอคทีฟ เป็นต้น นอกจากนี้คำสั่งที่จะทำให้สัญญาณแต่ละเส้นแอคทีฟได้ แสดงไว้ในตาราง

สัญญาณ	สัญญาณจะแอคทีฟเมื่อ
RD	MOVX A,@R, INS A, BUS
WR	MOVX @R, A, OUTL BUS, A
PSEN	การเฟทช์คำสั่งจากภายนอก MOVP, MOVP3

สัญญาณควบคุมอินพุท ได้แก่ สัญญาณ RESET, SS, EA และสัญญาณ INT รายละเอียดต่างๆ มีดังนี้

RESET เป็นสัญญาณอินพุทที่บังคับให้ซีพียู เริ่มทำงานใหม่ โดยการป้อนลอจิกศูนย์ ที่ขาอินพุทนี้ ในการใช้งานทั่วไปแล้วจะมีสวิทช์กดติด ปล่อยดับ ทำหน้าที่ส่งสัญญาณรีเซท ซึ่งมีวงจรดังรูปที่ 10

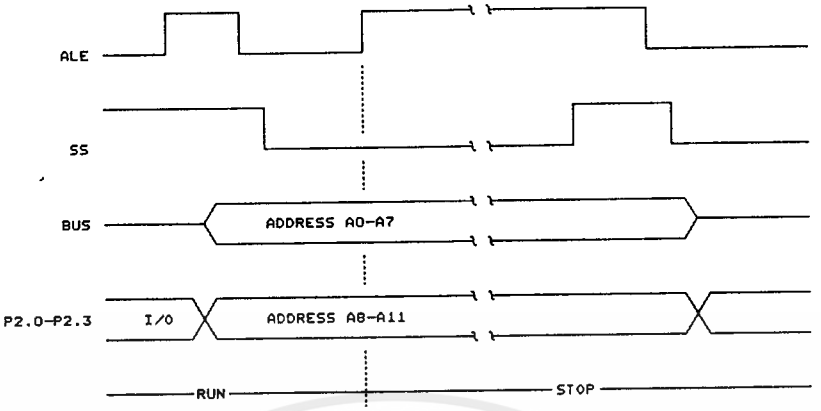


รูปที่ 10 แสดงว่าต่อวงจรภายนอกเพื่อรีเซท 8749

เมื่อ 8749 ถูกรีเซท จะทำให้

1. โปรแกรมเคาวน์เตอร์ และแสตคพอยเตอร์ ถูกรีเซทเป็นศูนย์
2. เลือกหน่วยความจำแบลงค์ศูนย์ และรีจีสเตอร์แบงค์ศูนย์
3. อินเตอร์รัพท์ ถูกดิสเอเบิ้ล
4. แฟลก F_0 และ F_1 และ ไทเมอร์แฟลก ถูกเคลียร์
5. T_0 จะกลายเป็นขาอินพุท

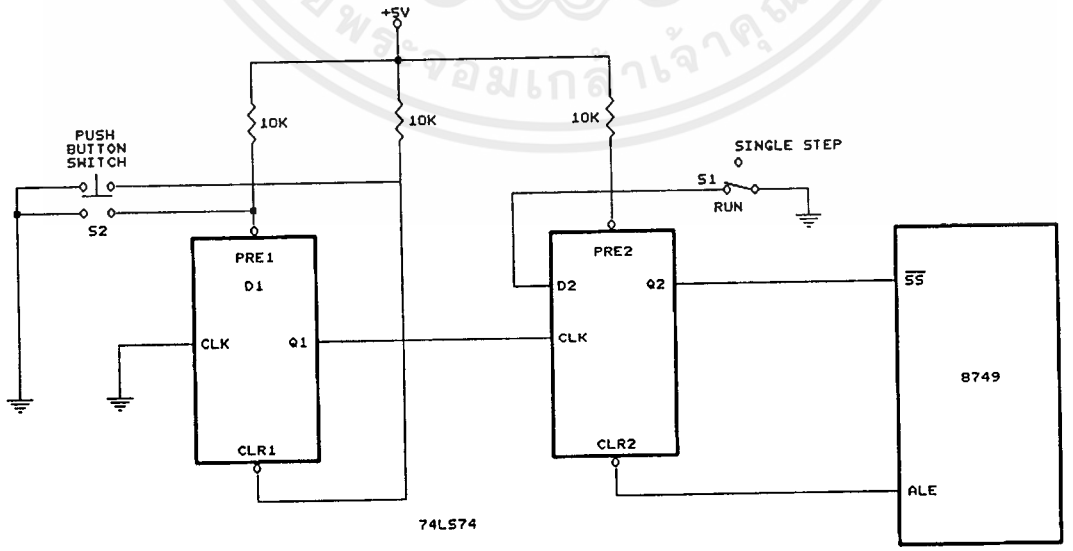
SS (SINGLE STEP) เป็นสัญญาณอินพุท ที่สามารถควบคุมให้ 8749 ทำงานทีละคำสั่ง ตามจังหวะที่เราต้องการ คือ ถ้าเราส่งสัญญาณ SS ให้มีจังหวะคล้องจองกับสัญญาณ ALE ก็จะสามารถบังคับให้ 8749 หยุดการทำงานของมันได้ตราบเท่าที่สัญญาณ SS ยังคงแอคทีฟอยู่ และในระหว่างที่ 8749 นี้หยุดทำงาน ตัวมันเองจะแสดงค่าแอดเดรสออกมาทางดาต้าบัส และ 4 บิทล่างของพอร์ท 2 (P_{20}-P_{23}) จากลักษณะนี้เองจึงเป็นการอำนวยความสะดวกแก่ผู้ใช้ ในการติดตามการทำงานของ 8749 ว่าเป็นไปตามโปรแกรมที่เราต้องการหรือไม่



รูปที่ 11 แสดง TIMING DIAGRAM ของชิงเกิ้ลเสต็ป

ขั้นตอนการทำ ชิงเกิลเสต็ปกันจาก TIMING DIAGRAM ในรูปที่ 11 ถ้าเราส่งลอจิกศูนย์ไปที่ขา SS ในช่วงที่ ALE เป็นลอจิกศูนย์ ขณะที่ 8749 กำลังเฟทช์คำสั่งอยู่นั้น 8749 จะหยุดทำงานไปชั่วขณะ พร้อมกับตอบรับการส่งสัญญาณชิงเกิ้ลเสต็ป ด้วยการให้ ALE มีลอจิกหนึ่ง ในตอนนี้ค่าแอดเดรสของคำสั่งถัดไปจะปรากฏออกมาทางดาต้าบัส และ 4 บิทล่างของพอร์ท 2

เมื่อเราให้ขา SS กลับเป็นลอจิกหนึ่ง 8749 ก็จะเปลี่ยนลอจิกของ ALE ให้เป็นศูนย์ และเริ่มทำงานต่อไป จากขั้นตอนการทำชิงเกิ้ลเสต็ป ที่กล่าวมานี้ เราสามารถต่อวงจรอีกเล็กน้อยเพื่อช่วยให้การทำชิงเกิ้ลเสต็ป เป็นไปอย่างง่ายดาย โดยใช้วงจรดังรูป 12



รูปที่ 12 แสดงการต่อวงจรภายนอกในการทำชิงเกิ้ลเสต็ป

เราสามารถจะเลือกให้ 8749 ทำงานแบบปกติ หรือแบบซิงเกิ้ลเสต็ปก็ได้ ถ้าเลือกแบบซิงเกิ้ลเสต็ป การกดสวิทช์ S_2 จะทำให้โปรแกรมดำเนินไปทีละคำสั่ง

EA (EXTERNAL ACCESS) ตามที่กล่าวมาแล้วว่า 8749 นั้นแบ่งหน่วยความจำโปรแกรมออกเป็นส่วนภายนอก กับส่วนภายใน 8749 เอง การให้ลอจิกแก่ขา EA เป็นการบอกให้ 8749 รันโปรแกรมในหน่วยความจำส่วนใด

ถ้า EA มีลอจิกหนึ่ง 8749 จะรันโปรแกรมในหน่วยความจำภายนอกทั้งหมด (ในกรณีของ 8035 และ 8039 หรือเบอร์อื่นๆ ที่ไม่มีหน่วยความจำโปรแกรมภายใน เราจำเป็นต้องต่อขา EA นี้กับลอจิกหนึ่งเสมอ)

แต่ถ้าหากให้ EA มีลอจิกศูนย์ จะเป็นการสั่งให้รันโปรแกรมจากหน่วยความจำภายใน และอาจจะรันโปรแกรมต่อมาถึงหน่วยความจำภายนอกที่เราต่อเพิ่มก็ได้

8749 จะยอมรับลอจิกของสัญญาณ EA ในช่วงที่มีการรีเซทเท่านั้น การเปลี่ยนแปลงลอจิกของ EA ในช่วงเวลาอื่น จะไม่มีผลเกิดขึ้น

โดยทั่วไปแล้ว คงจะไม่ค่อยเห็นประโยชน์จากขา EA นี้นัก แต่สำหรับในทางอุตสาหกรรมแล้ว เขาจะสามารถสั่งให้บรรจุโปรแกรมลงในหน่วยความจำภายในได้เลย เพื่อให้ 8749 สามารถทำงานได้ด้วยตัวมันเอง แต่ในกรณีที่เขาต้องการตรวจสอบระบบที่เกี่ยวข้อง เขาก็จะให้แก่ขา EA นี้ ซึ่ง 8749 ก็จะมาโปรแกรมทดสอบ ซึ่งเตรียมไว้ในหน่วยความจำโปรแกรมภายนอก ของเครื่องทดสอบนั่นเอง

จะเห็นว่า เราสามารถจะทดสอบระบบของ 8749 ได้ทั้งทางด้านฮาร์ดแวร์และทางด้านซอฟแวร์ โดยใช้ขาสัญญาณ EA และ SS ช่วยในการทดสอบ

การอินเตอร์รัพท์

อินเตอร์รัพท์ เป็นกระบวนการอย่างหนึ่งที่ระบบไมโครคอมพิวเตอร์ทั้งหลายจำเป็นต้องมีไว้ แม้แต่ระบบเล็กๆ อย่าง 8749 เองก็ยังมีระบบอินเตอร์รัพท์สองอย่างด้วยกัน คือ

1. อินเตอร์รัพท์จากอุปกรณ์ภายนอก ที่ส่งสัญญาณอินเตอร์รัพท์มาทางขา INT
2. อินเตอร์รัพท์จากภายใน เป็นการอินเตอร์รัพท์โดย ไทเมอร์/เคาวน์เตอร์ที่อยู่ภายใน 8749

เมื่อ 8749 ถูกอินเตอร์รัพท์ ไม่ว่าจะโดยอุปกรณ์ชนิดใดก็ตาม 8749 จะตรวจสอบสถานะการอินเตอร์รัพท์ของตัวมันเองเสียก่อน ว่าถูกอีนาเบิ้ลไว้แล้วหรือยัง ถ้าได้รับการอีนาเบิ้ลไว้แล้วละก็ 8749 จะหยุดทำงานที่กำลังดำเนินอยู่เดิม และก้าวไปตอบรับอุปกรณ์ที่ส่งสัญญาณอินเตอร์รัพท์มา โดยการเก็บค่าของโปรแกรมเคาวน์เตอร์ และสถานะของโปรแกรม (PSW) เข้าไปไว้ในแสตค และกระโดดไปยัง ISR ซึ่งมีอยู่สองแห่งด้วยกันตามประเภทของอินเตอร์รัพท์

การขออินเตอร์รัพท์โดยอุปกรณ์ภายนอก ด้วยการส่งสัญญาณมาทางขา INT นั้นจะทำให้ 8749 กระโดดไปยัง ISR ที่ตำแหน่ง $0003

แต่เมื่อ 8749 ถูกอินเตอร์รัพท์ด้วยไทเมอร์/เคาวน์เตอร์ ที่อยู่ภายใน จะทำให้ 8749 กระโดดไปยัง ISR อีกโปรแกรมหนึ่ง ที่ตำแหน่งแอดเดรส $0007 สำหรับรายละเอียดเกี่ยวกับตัวไทเมอร์/เคาวน์เตอร์นี้ จะกล่าวถึงอีกครั้งในภายหลัง

ถ้าจะเปรียบเทียบไปแล้ว ISR ก็มีลักษณะคล้ายกับ โปรแกรมย่อย (SUBROUTINE) ทั่วๆ ไป ยกเว้นอยู่จุดหนึ่ง คือ ที่ตอนจบของ ISR นั้น จะต้องลงท้ายด้วยคำสั่ง RETURN พิเศษ ที่เรียกว่า RETR ซึ่งต่างจากคำสั่ง RET ธรรมดาตรงที่ เมื่อทำคำสั่ง RETE นี้ นอกจากจะนำค่าแอดเดรสจากแสตคไปเก็บไว้ในโปรแกรมเคาวน์เตอร์แล้ว ยังจะเอาค่า PSW ที่เก็บในแสตค คืนกลับที่เดิมของมันด้วย

ในขณะที่ 8749 กำลังรันโปรแกรมใน ISR อยู่นั้น จะดิสเอเบิ้ลอินเตอร์รัพท์โดยอัตโนมัติ จึงทำให้ไม่สามารถมีการเรียกอินเตอร์รัพท์ซ้ำได้อีก จนกระทั่งเมื่อ 8749 พบกับคำสั่ง RETR ซึ่งเป็นคำสั่งปิดท้ายโปรแกรม ISR แล้ว 8749 จะได้การอีนาเบิ้ลอินเตอร์รัพท์ อีกครั้งหนึ่ง

หลังจากเสร็จ RETR แล้ว 8749 จะได้รับการอีนาเบิ้ลอินเตอร์รัพท์อีกครั้งและก็จะกลับไปทำงานของตนตามปกติต่อไป ในระหว่างนี้อาจจะเกิดปัญหาขึ้นได้ เมื่อสัญญาณอินเตอร์รัพท์ที่ส่งมาในตอนแรกนั้น ยังคงแอคทีฟอยู่ แม้จะจบโปรแกรม ISR แล้วก็ตาม จะทำให้ 8749 ย้อนกลับไปทำโปรแกรม ISR อีกครั้งหนึ่งทันที ที่จบคำสั่ง RETR อันเป็นความผิดพลาดที่เราไม่ต้องการ การแก้ไขปัญหา ทำได้สามลักษณะ คือ

1. คอยตรวจสอบดูก่อนว่า สัญญาณอินเตอร์รัพท์ยังคงแอคทีฟอยู่หรือไม่ ถ้าขา INT ยังมีลอจิกศูนย์ก็จะให้รออยู่ภายใน ISR ก่อน จนกระทั่งเมื่อขา INT เป็นลอจิกหนึ่งจึงจะออกจาก ISR ไปได้

การตรวจสอบสถานะลอจิกที่ขา INT นั้น เราสามารถได้ด้วยคำสั่งกระโดดแบบมีเงื่อนไข JNI ซึ่งจะกระโดดเมื่อขา INT เป็นลอจิกศูนย์ การที่เราสามารถใช้คำสั่ง JNI นี้ได้ เพราะขาสัญญาณ INT นั้นสามารถทำหน้าที่ได้สองลักษณะ คือ เมื่อ 8749 ได้รับการอีนาเบิ้ลอินเตอร์รัพท์ขา INT จะใช้เป็นอินพุทของการอินเตอร์รัพท์ แต่เมื่อ 8749 ถูกดิสเอเบิ้ลอินเตอร์รัพท์ เช่นเมื่ออยู่ใน ISR เราจะสามารถใช้ขา INTเป็นขาอินพุทธรรมดาเช่นเดียวกับ ขา T_0 หรือ T_1 ได้

2. การส่งสัญญาณตอบรับ (ACKNOWLEDGE) จาก 8749 ย้อนกลับไปบอกอุปกรณ์ที่ขออินเตอร์รัพท์ ว่าขณะนี้ 8749 รับรู้ไว้แล้ว ให้ถอนอินเตอร์รัพท์ออกไปได้ เนื่องจาก 8749 ไม่มีขาสัญญาณสำหรับ การตอบรับอินเตอร์รัพท์อยู่ ดังนั้นเราจำเป็นต้องเป็นผู้จัดให้เอง โดยอาจจะกำหนดเอาบิทใดบิทหนึ่งของพอร์ท 1 หรือ พอร์ท 2 มาใช้แทนก็ได้

3. การออกแบบให้วงจรภายนอกส่งสัญญาณอินเตอร์รัพท์ ในลักษณะพัลซ์ โดยที่ช่วงแอคทีฟของสัญญาณจะต้องไม่เกินกว่า เวลาที่ใช้ในการเอ็กซีคิ้วท์โปรแกรม ISR จนจบ และในขณะเดียวกันก็จะต้องมีช่วงกว้างไม่น้อยกว่า 2 แมชชีนไซเคิล์ ในช่วงที่ ALE แอคทีฟ

การอีนาเบิ้ลอินเตอร์รัพท์ ก็คือ การยอมให้ 8749 รับรู้อินเตอร์รัพท์ต่างๆ ได้ เราสามารถสั่งการอีนาเบิ้ลได้โดยตรงด้วยคำสั่ง EN I นอกจากนี้ 8749 ก็ยังสามารถอีนาเบิ้ลตัวมันเองโดยอัตโนมัติ เมื่อกระทำคำสั่ง RETR ด้วยเช่นกัน ดังที่กล่าวมาแล้ว

ส่วนการดิสเอเบิ้ลอินเตอร์รัพท์นั้น ทำได้สามวิธีด้วยกัน เป็นแบบอัตโนมัติเสียสองวิธี ดังนี้

1. ใช้คำสั่ง DIS I ในการดิสเอเบิ้ลโดยตรง

2. โดยการรีเซท 8749 จะทำให้อินเตอร์รัพท์ ถูกดิสเอเบิ้ลโดยอัตโนมัติ

3. ในขณะที่ 8749 กำลังอยู่ในโปรแกรม ISR 8749 จะถูกดิสเอเบิ้ลอินเตอร์รัพท์เช่นกัน เพื่อป้องกันการอินเตอร์รัพท์ซ้ำซ้อน

สำหรับเรื่องการขยายระบบอินเตอร์รัพท์ให้มากกว่าหนึ่งนั้น สามารถใช้ไอซีอินเตอร์รัพท์คอนโทรลเลอร์ เบอร์ 8259 มาช่วยก็ได้ แต่จะเพิ่มความยุ่งยากมากขึ้นจนไม่เหมาะกับระบบเล็กๆ อย่างนี้

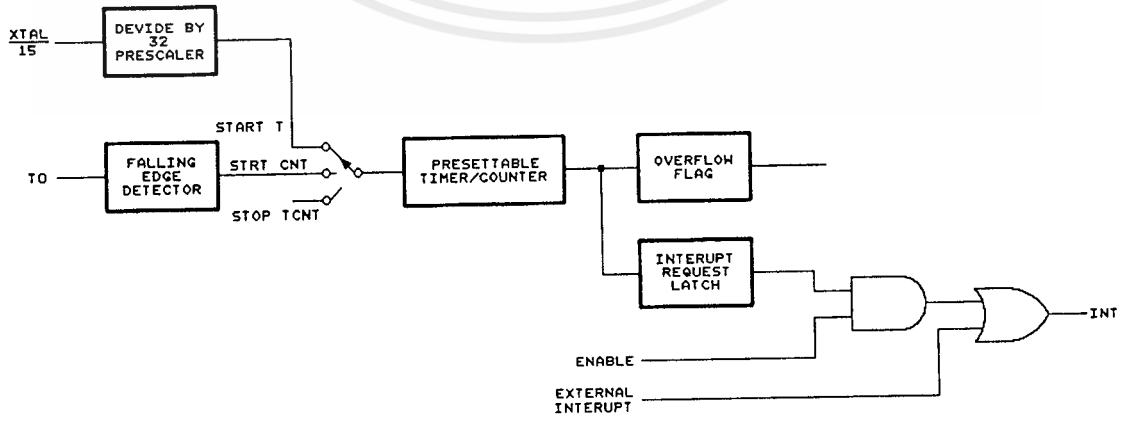
ไทเมอร์/เคาวน์เตอร์ (TIMER/CUONTER)

เราได้พูดถึงเรื่องของหน่วยความจำ และพอร์ทที่มีอยู่ภายใน 8749 มาแล้ว ยังเหลืออยู่ก็แต่ตัวไทเมอร์/เคาวน์เตอร์ ขนาด 8 บิท อีกตัวหนึ่ง ที่เรายังไม่ได้กล่าวถึง ลองมาดูซิว่า เราจะใช้งานมันได้อย่างไร

ลักษณะของตัวเคาวน์เตอร์เป็นแบบนับขึ้น มีขนาด 8 บิท สามารถตั้งโปรแกรมให้เริ่มนับที่ค่าใดค่าหนึ่งก็ได้ โดยค่าตัวเลขเริ่มต้น จะถูกเก็บไว้ในรีจีสเตอร์ที่มีชื่อว่า T เราสามารถเปลี่ยนแปลงค่าในรีจีสเตอร์ T หรือนำค่าใน T ออกมาดูก็ได้ โดยใช้คำสั่ง MOV T, A และ MOV A,T ตามลำดับ

เมื่อเราสั่งให้ไทเมอร์/เคาวน์เตอร์เริ่มทำงาน โดยใช้คำสั่ง START ค่าในรีจีสเตอร์ T จะถูกโหลดเข้าไปในไทเมอร์/เคาวน์เตอร์ (T/CNT) แล้วเริ่มการนับสัญญาณ (สัญญาณนาฬิกาที่ใช้นับนั้นอาจมาจากภายใน หรือภายนอกก็ได้ ซึ่งจะกล่าวละเอียดในตอนหลัง) และเพิ่มค่าใน T/CNT เรื่อยๆ จนเมื่อนับจากค่า $FF ไปเป็น $00 T/CNT ก็จะส่งสัญญาณอินเตอร์รัพท์ภายในไปยังซีพียู และขณะเดียวกัน ก็จะเซทค่าไทเมอร์แฟลก ให้เป็นลอจิกหนึ่ง

สำหรับไทเมอร์แฟลกนั้น เราสามารถตรวจสอบได้ด้วยคำสั่งกระโดดอย่างมีเงื่อนไข JTF และเมื่อซีพียูพบกับคำสั่งนี้แล้ว ซีพียูก็จะเคลียร์แฟลกนี้ ให้เป็นศูนย์ นอกจากนี้การรีเซทก็จะทำให้ไทเมอร์แฟลกเป็นศูนย์ได้เช่นกัน หลังจากที่ T/CNT นับถึง $00 แล้วก็ตาม แต่ T/CNT ก็จะยังไม่หยุดหน้าที่ของตน กล่าวคือ จะเริ่มต้นนับใหม่ โดยการโหลดเอาค่าเริ่มต้นจากรีจีสเตอร์ T อีกเช่นเคย การนับจะเป็นอย่างต่อเนื่องเช่นนี้ จนกว่าซีพียูจะพบกับคำสั่ง STOP หรือเมื่อ 8749 ถูกรีเซทใหม่ การนับจึงจะหยุดลงได้ และเป็นเพียงการหยุดนับเท่านั้น ค่าในรีจีสเตอร์ T จะไม่เปลี่ยนแปลง เราสามารถจะให้



รูปที่ 13 แสดงแผนผังของระบบ ไทเมอร์/เคาวน์เตอร์

เมื่อ T/CNT ส่งสัญญาณไปอินเตอร์รัพท์ซีพียู ซีพียูจะต้องตรวจสอบตัวเองก่อนว่า อรนาเบิ้ล อินเตอร์รัพท์ไว้แล้วหรือยัง และขณะนั้นจะต้องไม่มรการขออินเตอร์รัพท์จากภายนอก (ทางขา INT) ด้วย ซีพียูจึงจะสามารถตอบรับการเรียกจาก T/CNT ได้ แต่ถ้าเกิดมีอินเตอร์รัพท์พร้อมๆ กันทั้งภายในและภายนอกซีพียูได้ถูกออกแบบให้ถือการอินเตอร์รัพท์จากภายนอก มีความสำคัญมากกว่า อินเอตร์รัพท์จาก T/CNT ดังนั้นซีพียูจะกระโดดไปยัง ISR ที่แอดเดรส $0003 เสียก่อน เมื่อเสร็จสิ้นแล้วจึงค่อยกระโดดไปยัง ISR ของ T/CNT ซึ่งเริ่มต้นที่แอดเดรส $0007

การอีนาเบิ้ล และดิสเอเบิ้ลอินเตอร์รัพท์ สามารถกระทำได้โดยใช้คำสั่ง EN TCNTI และ DIS TCNTI ตามลำดับ ซึ่งจะเห็นว่า ใช้คำสั่งแยกจากอินเตอร์รัพท์จากภายนอก แต่สำหรับการอีนาเบิ้ล และดิสเอเบิ้ล โดยอัตโนมัตินั้นยังคงเหมือนกันคือ

1.เมื่อซีพียูพบกับคำสั่งRETR ก็จะได้รับการอีนาเบิ้ล อินเตอร์รัพท์

2.เมื่อมีการรีเซท หรือ ขณะอยู่ในโปรแกรม ISR ซีพียูจะถูกดิสเอเบิ้ลอินเตอร์รัพท์ โดยอัตโนมัติ

การทำงานของ T/CNT แบ่งออกเป็นสองแบบด้วยกัน คือ

1. **ไทเมอร์ โหมด**

2. **เคาน์เตอร์ โหมด**

การเลือกให้ T/CNT เป็นตัวจับเวลา (TIMER) ทำได้โดยใช้คำสั่ง START ซึ่งซีพียูจะจัดสัญญาณนาฬิกาที่มีความถี่คงที่ป้อนให้แก่ T/CNT สัญญาณดังกล่าว จะมีความถี่เท่ากับ ความถี่ของคริสตอลหารด้วย 480 เช่น ถ้าใช้คริสตอลที่มีความถี่ 6 MHz เราก็จะได้สัญญาณนาฬิกาที่ป้อนให้แก่ T/CNT มีความถี่เท่ากับ 12.5 KHz หรือค่าของ T/CNT จะนับเพิ่มขึ้นทุกๆ 80 uS นั่นเอง

ช่วงเวลาที่ T/CNT สามารถนับได้จะอยู่ในช่วง 80uS ถึง 20 mS ถ้าหากเราต้องการช่วงเวลาที่ยาวนานกว่านี้ ก็เห็นจะต้องเขียนโปรแกรมให้ช่วยในการนับอีกทอดหนึ่ง เช่น ถ้าเราต้องการออกแบบให้ 8749 เป็นนาฬิกา ซึ่งต้องอาศัยฐานเวลาขนาด 1 Hz หรือมีคาบเวลา 1 วินาที เราสามารถออกแบบให้ T/CNT นับเวลาได้นานที่สุด 2 mS ถ้าหากเราเขียนโปรแกรมให้ซีพียูนับจำนวนครั้งที่มันถูกอินเตอร์รัพท์โดย T/CNT เราก็จะทราบว่า เมื่อซีพียูถูกอินเตอร์รัพท์ครบ 50 ครั้ง จะเป็นเวลาประมาณ 1 วินาที เป็นต้น

เราได้พูดถึงช่วงเวลาที่ ไทเมอร์สามารถนับได้มากที่สุดไปแล้ว ทีนี้มาดูช่วงเวลาที่น้อยที่สุด ที่ไทเมอร์จะทำงานได้ โดยการโหลดค่า $FF ใส่ลงในรีจีสเตอร์ T แล้วใช้คำสั่ง STRT T หลังจากนั้นอีก 80 uS ไทเมอร์ก็จะส่งสัญญาณมาอินเตอร์รัพท์ซีพียู

ถ้าเราคิดว่า ช่วงเวลาขนาด 80 uS นั้นมากเกินไป ต้องการให้มีความละเอียดมากขึ้น ก็สามารถทำได้โดยการป้อนสัญญาณนาฬิกา ที่มีความถี่สูงขึ้น เข้ามาทางขา T_1 แล้วใช้คำสั่ง STRT CNT สั่งให้ T/CNT ทำหน้าที่เป็นเคาวน์เตอร์แทน เราสามารถนำสัญญาณนาฬิกาจากภายนอกมาป้อนให้ที่อินพุทของเคาวน์เตอร์ได้ และสัญญาณดังกล่าวไม่จำเป็นต้องมีคาบเวลา หรือความถี่แน่นอนเหมือนในกรณีไทเมอร์โหมด เราอาศัยลักษณะขอบขาลง(HIGH TO LOW TRANSITION) ของสัญญาณอินพุท T_1 เป็นตัวเพิ่มค่าของเคาวน์เตอร์ สัญญาณดังกล่าวนี้ จะต้องมีลักษณะสำคัญ 2 อย่าง คือ มีช่วงเวลาที่เป็นลอจิกหนึ่งนานมากกว่า 500 nS และจะต้องมีความถี่น้อยกว่า ความถี่ของ ALE หารด้วย 3 นั่นคือ ถ้าเราใช้คริสตอลขนาด 6 MHz จะได้ความถี่ของ ALE เป็น 400 KHz และสัญญาณที่ป้อนให้แก่เคาวน์เตอร์ต้องมีความถี่ไม่เกิน 133 KHz

ตามที่เคล่ามาก่อนหน้านี้ว่า T/CNT ใน 8749 นั้น ทำงานแบบต่อเนื่องไม่ว่าจะอยู่ในโหมดใด ด้วยเหตุนี้จึงสามารถส่งสัญญาณไปอินเตอร์รัพท์ซีพียูได้อย่างสม่ำเสมอ จนกว่าเราจะสั่งให้ T/CNT หยุดทำงาน โดยใช้คำสั่ง STOP TCNT หรือโดยการรีเซทระบบของ 8749 ก็ได้

<u>ขาอินพุทสำหรับการทดสอบ และ การใช้งานขา V_{DD}</u>

ขาอินพุทสำหรับการทดสอบ

ในตัวของ 8749 นั้น นอกจากจะมีพอร์ท 2 พอร์ทสำหรับติดต่อกับภายนอกแล้ว ยังมีขาสัญญาณบางขาที่ถูกออกแบบให้เป็นอินพุทขนาดหนึ่งบิทได้ ขาสัญญาณเหล่านั้นได้แก่ T_0 T_1 และ INT

จากที่ผ่านมา เราก็ได้รู้จักกับขาสัญญาณทั้งสามนี้บ้างแล้ว ว่าขาสัญญาณเหล่านี้สามารถใช้งานได้หลายหน้าที่ โดยใช้ซอฟท์แวร์ควบคุม

ขา T_0 สามารถเป็นได้ทั้งอินพุท และเอาท์พุท โดยปกติแล้ว ขา T_0 จะเป็นอินพุท ที่เราสามารถตรวจสอบลอจิกของ T_0 ได้ด้วยคำสั่ง JNTO หรือ JTO เมื่อเราต้องการให้ T_0 เป็นเอาท์พุท จะใช้คำสั่ง ENTO CLK เป็นการสั่ง 8749 ให้เอาท์พุทสัญญาณนาฬิกาที่เป็นฐานเวลาของระบบ (มีความถี่เท่ากับ ความถี่ของคริสตอล หารด้วย 3) ออกมาทางขา T_0 นี้

เมื่อ T_0 กลายเป็นเอาท์พุทไปแล้วเราจะไม่สามารถใช้คำสั่งใดๆ เพื่อเปลี่ยน T_0 กลายเป็นเอาท์พุทไปแล้ว เราจะไม่สามารถใช้คำสั่งใดๆ เพื่อเปลี่ยน T_0 ให้กลายเป็นอินพุทอีกครั้งหนึ่ง นอกจากจะอาศัยการรีเซท 8749 เท่านั้น

ขา T_1 นั้น เราใช้เป็นขาอินพุทของสัญญาณนาฬิกาที่ป้อนให้แก่เคาวน์เตอร์ตามที่กล่าวมาแล้ว นอกจากนั้นเรายังสามารถใช้ T_1 เป็นขาอินพุทธรรมดาได้เช่นกัน และสามารถตรวจสอบลอจิกของสัญญาณที่ขา T_1 ได้ด้วยคำสั่ง JNT1 และ JT1

ขา INT นั้น เป็นขาสัญญาณอินพุทเท่านั้น แต่สามารถจะเลือกใช้ขานี้ ในฐานะที่เป็นขาอินเตอร์รัพท์ หรือขาอินพุททดสอบธรรมดา

ในกรณีที่ อินเตอร์รัพท์ถูกอีนาเบิ้ล ขา INT ก็จะเป็นขาอินพุทสำหรับการขออินเตอร์รัพท์ ตรงกันข้ามถ้าอินเตอร์รัพท์ถูกดิสเอเบิ้ลขา INT ก็จะกลายเป็นขาอินพุทธรรมดา ที่เราสามารถตรวจสอบลอจิกที่อินพุทนี้ได้ด้วยคำสั่ง JNI

เรื่องของขาอินพุททดสอบ ที่อินเทลออกแบบมาให้นี้ นอกเหนือจากพอร์ทแบบขนานสองพอร์ทที่ให้มาด้วย ทำให้ผมนึกถึงแฟลกพิเศษที่อินเทลออกแบบมาให้ด้วย คือ ซอฟท์แวร์แฟลก F_0 และ F_1 เพราะโดยทั่วไปแล้ว ถ้าไม่มีแฟลกทั้งสองนี้ เราก็สามารถใช้หน่วยความจำข้อมูลไบท์ใดไบท์หนึ่งมาทำหน้าที่แทนแฟลกตัวนี้ได้ เช่นเดียวกับที่เราสามารถกำหนดให้บิทใดบิทหนึ่งของพอร์ทแบบขนาน มาทำหน้าที่แทน ขาอินพุททดสอบใดก็ได้

สิ่งเล็กๆ น้อยๆ เหล่านี้ ที่อินเทลตระเตรียมไว้ให้ ดูเหมือนของเด็กเล่นธรรมดาแต่จริงๆ แล้ว กลับกลายเป็นสิ่งที่ทำให้ ระบบเล็กๆ อย่าง 8749 มีความคล่องตัวมากขึ้น

การใช้งานขา V_{DD}

ไอซีเบอร์นี้ต้องการไฟเลี้ยง 5 โวลต์ที่ขา V_{cc} และต่อขา V_{ss} ลงกราวด์ไป ที่นี้ก็เหลืออยู่อีกขาหนึ่งคือ ขา V_{DD} ซึ่งไม่ได้ใช้ประโยชน์อะไร เราก็มักจะต่อขา V_{DD} นี้เข้ากับไฟเลี้ยง 5 โวลต์ แต่ถ้าหากคุณต้องการจะใช้งานขานี้แล้วละก็ จำเป็นจะต้องศึกษากันหน่อยซิว่าขา V_{DD} นี้ใช้ทำอะไร

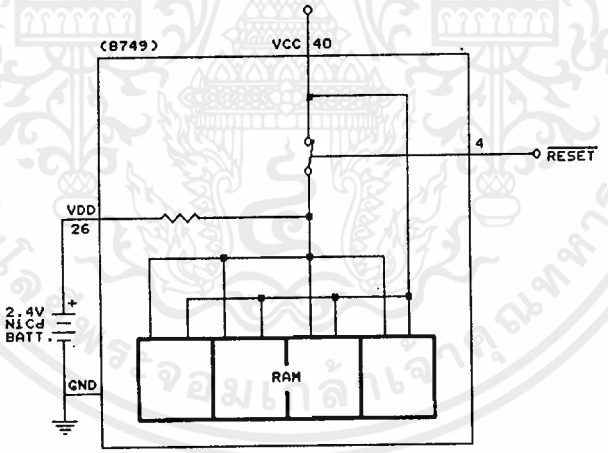
ถ้าเราต่อแบตตารี่ ชนิด NI-CD เข้ากับขา V_{DD} แล้ว กระแสจากแบตตารี่จะช่วยรักษาข้อมูลที่อยู่ในหน่วยความจำข้อมูลภายใน ไว้ไม่ให้สูญหาย เมื่อเราเลิกจ่ายไฟเลี้ยง V_{cc} ให้แก่ 8749 และในขณะที่มีไฟเลี้ยง V_{cc} ป้อนให้แก่ระบบ กระแสไฟบางส่วนก็จะไหลเข้าไปชาร์จแบตตารี่ NI-CD ที่ขา 26 ด้วย

เนื่องจาก แรมภายใน 8749 นั้นถูกออกแบบให้มีลักษณะพิเศษ จนเราไม่จำเป็นต้องป้อนไฟเลี้ยงสูงมากนักในภาษาข้อมูลในแรม เพียงแต่อาศัยแบตตารี่ NI-CD จำนวน 2 เซลก็เพียงพอแล้ว ซึ่งจะได้แรงดันประมาณ 2.4 โวลต์

ตารางในรูปที่ 14 แสดงปริมาณกระแสที่เลี้ยงแรมภายใน ซึ่งขึ้นอยู่กับจำนวนไบท์ของแรม

กระแส I_{DD} ที่ 2.2 V_{DD}	จำนวนไบท์ของแรม
1.5 mA	0 - 32
2.5 mA	0 - 64
4.5 mA	0 - 128
8.5 mA	0 - 256

รูปที่ 14 แสดงปริมาณกระแสที่จ่ายให้แก่แรมภายใน



รูปที่ 15 แสดงวงจรภายในส่วน V_{DD}

จะเห็นว่าสัญญาณรีเซทเป็นตัวการสำคัญในการเปิด ปิด สวิทช์ที่อยู่ภายใน ถ้า RESET เป็นลอจิกศูนย์ สวิทช์ภายในก็จะเปิดวงจรออก กระแสไฟฟ้าจากแบตตารี่ จะไหลเข้าไปให้แก่แรม หรือในขณะที่ RESET เป็นลอจิกหนึ่ง สวิทช์ก็จะปิดวงจร มีกระแสไหลจาก V_{CC} ไปเลี้ยงแรม และไหลออกไปชาร์จแบตตารี่ภายนอกด้วย

การใช้งานขา V_{DD} จะอยู่ในเรื่องการป้องกันข้อมูลสูญหายเมื่อไฟตก ซึ่งการออกแบบระบบนี้ คุณจะต้องมีวงจรทางฮาร์ดแวร์ สำหรับตรวจจับการเปลี่ยนแปลงของแรงดันไฟเลี้ยง V_{CC} ซึ่งถ้าวงจรตรวจพบการลดลงของแรงดัน V_{CC} จนถึงระดับหนึ่ง ขอเรียกว่า ระดับเตรียมพร้อม (เป็นระดับไฟเลี้ยงต่ำสุดที่ 8749 ยังคงทำงานอยู่ได้) ก็จะส่งสัญญาณไปอินเตอร์รัพท์ 8749 ให้เก็บข้อมูลที่สำคัญลงในแรม และพร้อมจะ STAND BY ทุกเมื่อ

ส่วนทางด้านซอฟท์แวร์นั้น เราอาจจะต้องออกแบบโปรแกรมให้ 8749 สามารถตรวจสอบตัวเองได้ว่า นี้เป็นการเริ่มต้นทำงานครั้งแรก หรือเป็นการเริ่มต้นการทำงานหลังจากที่ไฟตกลงไป ซึ่งถ้าหากเป็นกรณีหลัง 8749 ก็จะได้สามารถดำเนินงานที่ทำค้างไว้ได้โดยอาศัยข้อมูลที่เก็บไว้ในหน่วยความจำข้อมูลภายใน

ชุดคำสั่งของ 8749

ชุดคำสั่งของ 8749 แบ่งออกเป็นกลุ่มๆ เพื่อให้ง่ายแก่การศึกษา ดังนี้

1. กลุ่มคำสั่ง ควบคุม
2. กลุ่มคำสั่ง เคลื่อนย้ายข้อมูล
3. กลุ่มคำสั่ง อินพุทเอาท์พุท
4. กลุ่มคำสั่ง เกี่ยวกับแอคคิวมูเลเตอร์
5. กลุ่มคำสั่ง เกี่ยวกับไทเมอร์/เคาวน์เตอร์
6. กลุ่มคำสั่ง เกี่ยวกับการกระโดด
7. กลุ่มคำสั่ง เกี่ยวกับโปรแกรมย่อย
8. กลุ่มคำสั่ง เกี่ยวกับแฟลก

กลุ่มคำสั่งควบคุม ได้แก่

- คำสั่งอีนาเบิ้ล และดิสเอเบิ้ลอินเตอร์รัพท์ (EN I และ DIS I)
- คำสั่งเลือกแบงค์ของรีจีสเตอร์ (SEL RB_0 หรือSEL RB_1)
- คำสั่งเลือกแบลงค์ของรีจีสเตอร์ (SEL MB_0 หรือ SEL MB_1)
- คำสั่งเลือกแบลงค์ของหน่วยความจำ (SEL MB_0 หรือ SEL MB_1)
- คำสั่งเอาท์พุทสัญญาณนาฬิกาของระบบออกมาทางขา T_0 (ENTO CLK)
- คำสั่ง NO OPERATION (NOP)

กลุ่มคำสั่งเคลื่อนย้ายข้อมูล ได้แก่

- คำสั่งเคลื่อนย้ายข้อมูลระหว่างรีจีสเตอร์ หรือหน่วยความจำข้อมูลภายในกับแอคคิวมูเลเตอร์ (MOV)

- คำสั่งเคลื่อนย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์กับหน่วยความจำข้อมูลภายนอก(MOVX)

- คำสั่งอ่านข้อมูลจากหน่วยความจำโปรแกรม ที่อยู่ในเพจเดียวกัน (MOVP) และคำสั่งอ่านข้อมูลจากหน่วยความจำเพจ 3 (MOVP3)

- คำสั่งแลกเปลี่ยนข้อมูลระหว่างแอคคิวมูเลเตอร์ กับรีจีสเตอร์ หรือกับหน่วยความจำข้อมูลภายใน (XCH) และคำสั่งแลกเปลี่ยนข้อมูลระหว่างแอคคิวมูเลเตอร์กับหน่วยความจำข้อมูลภายในโดยแลกเปลี่ยนเฉพาะข้อมูล 4 บิทล่างเท่านั้น (XCHD)

กลุ่มคำสั่งเคลื่อนย้ายข้อมูล ได้แก่

- คำสั่งอ่านข้อมูล หรือเขียนข้อมูล ระหว่างแอคคิวมูเลเตอร์ กับพอร์ท 1 หรือพอร์ท 2 (IN และ OUTL)

- คำสั่งอ่านเขียนข้อมูลระหว่าง แอคคิวมูเลเตอร์ กับดาต้าบัส (INS และ OUTL)

- คำสั่งเกี่ยวกับการ AND หรือ การ OR ข้อมูล กับข้อมูลเดิมในพอร์ท 1 หรือ พอร์ท 2 หรือดาต้าบัส (ANL และ ORL)

- คำสั่งเกี่ยวกับพอร์ทส่วนขยายที่ใช้ไอซี เบอร์ 8243 (รายละเอียดของไอซี 8243 จะกล่าวถึงในตอนท้าย) อันได้แก่ คำสั่งอ่าน เขียนข้อมูลกับพอร์ทนี้ และคำสั่ง AND, OR ข้อมูลกับพอร์ทนี้ (MOVD, ANLD, ORLD)

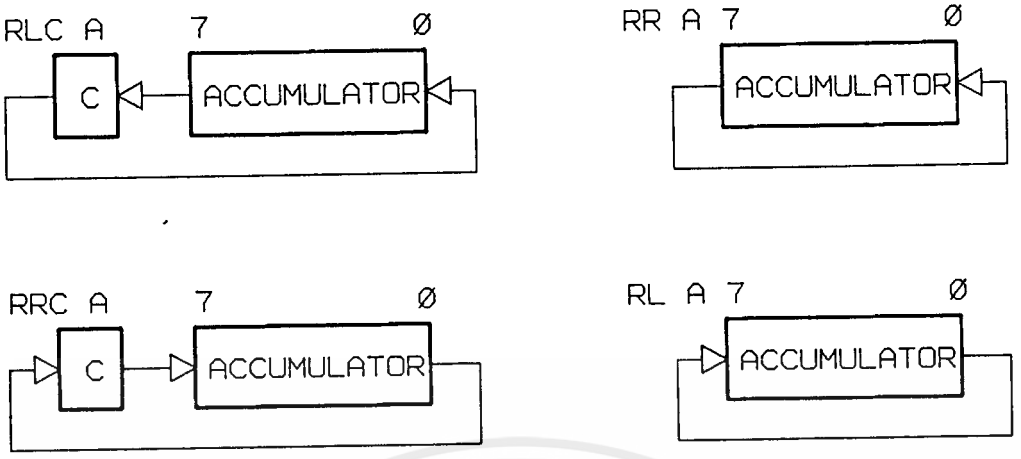
ตัวอักษร D ที่ห้อยท้ายนั้น หมายถึง การกระทำเกี่ยวกับข้อมูลเฉพาะ 4 บิทล่างเท่านั้น

กลุ่มคำสั่งเกี่ยวกับแอคคิวมูเลเตอร์ ได้แก่

- คำสั่งเกี่ยวกับการบวกตัวเลข แบบไม่คิดตัวทด และแบบที่คิดตัวทดด้วย (ADD และ ADDC)

- คำสั่งเกี่ยวกับการกระทำทางลอจิก คือ การ AND, OR, EX-CLUSIVE OR, COMPLEMENT และ CLEAR (ANL, ORL, XRL, CPL และ CLR ตามลำดับ)

- คำสั่งเกี่ยวกับการหมุน และการเลื่อนบิท มีลักษณะดังรูปที่ 16



รูปที่ 16 แสดงคำสั่งเกี่ยวกับการเลื่อนบิท

- คำสั่งเพิ่ม หรือลดค่าของแอคคิวมูเลเตอร์ หรือรีจีสเตอร์ หรือหน่วยความจำข้อมูลภายใน (INC และ DEC)
- คำสั่งปรับค่าตัวเลขเป็นฐานสิบ DECIMAL ADJUST (DA)
- คำสั่งสลับค่าข้อมูลในแอคคิวมูเลเตอร์ ระหว่าง 4 บิทล่าง กับ 4 บิทบน (SWAP)

กลุ่มคำสั่งเกี่ยวกับ ไทเมอร์/เคาวน์เตอร์ ได้แก่

- คำสั่งเคลื่อนย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์ กับรีจีสเตอร์ T(MOV)
- คำสั่งเริ่ม หรือหยุดการทำงานของ ไทเมอร์/เคาวน์เตอร์ (STRT และ STOP)
- คำสั่งอีนาเบิ้ล และดิสเอเบิ้ล ไทเมอร์อินเตอร์รัพท์ (EN TCNTI และ DIS TCNTI)

กลุ่มคำสั่งเกี่ยวกับการกระโดด ได้แก่

- คำสั่งกระโดด (JMP)
- คำสั่งกระโดดทางอ้อม ภายในเพจเดียวกัน (JMPP)
- คำสั่งตรวจสอบขาอินพุททดสอบ T_0, T_1, INT (JTO, JNTO, JT1, JNT1 และ JNI ตามลำดับ)
- คำสั่งตรวจสอบแฟลกต่างๆ เช่น แฟลกตัวทด ซอฟท์แวร์แฟลก (JC,JNC,JFOและJFI)
- คำสั่งทดสอบค่าในแอคคิวมูเลเตอร์ว่าเป็นศูนย์หรือไม่ (JZ และ JNZ)
- คำสั่งตรวจสอบลอจิกหนึ่งของบิทใดบิทหนึ่งในแอคคิวมูเลเตอร์ (JB)

- คำสั่งเกี่ยวกับการวนลูป โดยการลดค่ารีจีสเตอร์ลงหนึ่ง แล้วตรวจสอบว่าเป็นศูนย์ หรือไม่ ถ้าค่าในรีจีสเตอร์ไม่เท่ากับศูนย์ ก็จะกระโดดไป (DJNZ)

กลุ่มคำสั่งเกี่ยวกับโปรแกรมย่อย ได้แก่

- คำสั่งเรียกโปรแกรมย่อย (CALL)
- คำสั่ง RETURN มีสองคำสั่งด้วยกัน คือ RET และ RETR

กลุ่มคำสั่งเกี่ยวกับแฟลก ได้แก่

- คำสั่ง CLEAR และ COMPLEMENT แฟลกตัวทด และ ซอฟท์แวร์แฟลก F_0, F_1 (CLR และ CPL)

การขยายระบบ

แม้ไอซีในอนุกรม 8749 บางตัว เช่น 8748 จะมี EPROM ขนาด 1 KBYTE เป็นหน่วยความจำโปรแกรมภายในที่เราสามารถเขียนโปรแกรมของเราใส่ลงไปได้ ทำให้สามารถจะทำงานได้โดยลำพัง แต่สำหรับไอซีเบอร์อื่นๆ เช่น 8035, 8039, 8749 และ 8049 เป็นต้น เราจำเป็นต้องต่อหน่วยความจำโปรแกรมภายนอกเข้ากับไอซีเหล่านี้ จึงจะสามารถทำงานได้

การต่อวงจรภายนอก หรือการขยายระบบที่ผมจะพูดถึง มีด้วยกัน 3 เรื่อง คือ

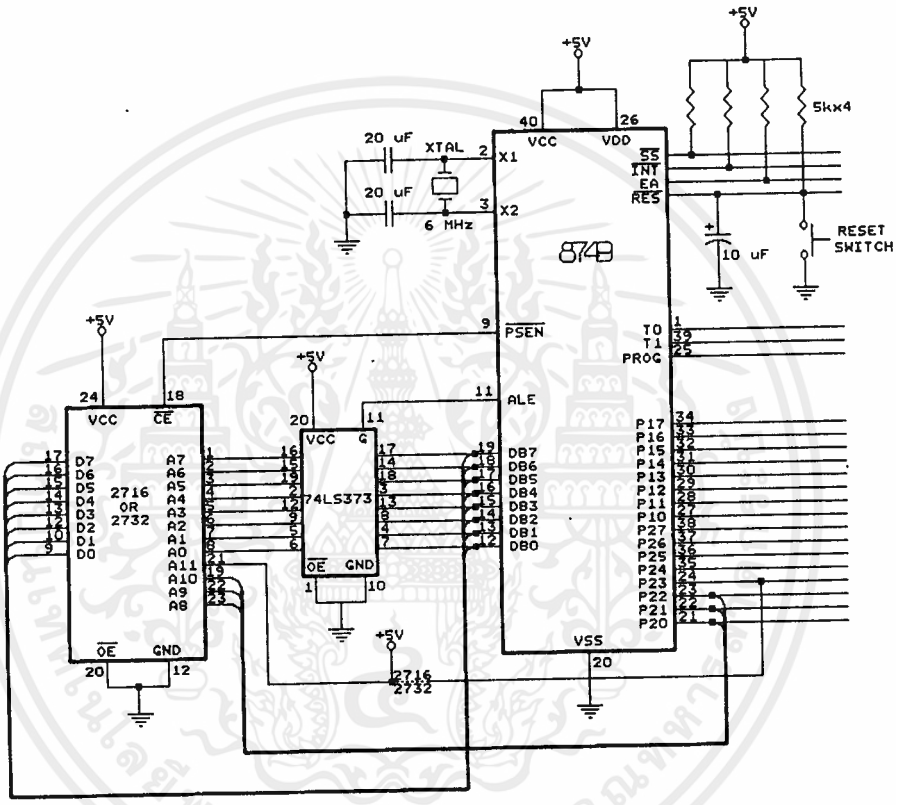
การต่อขยายหน่วยความจำโปรแกรมภายนอก

ไอซี 74LS373 ใช้สัญญาณ ALE ในการแลทช์ค่าของแอดเดรส ให้แยกออกมาจากดาต้าบัส ได้เป็นแอดเดรสบัส A_0-A_7 ส่วนแอดเดรสบัส A_8-A_{11} นั้นได้มาจาก 4 บิทล่างของพอร์ท 2 (P_{20}-P_{23})

ส่วนสัญญาณ PSEN นั้นเป็นสัญญาณที่แสดงให้รู้ว่า 8749 กำลังเฟทช์คำสั่งจากหน่วยความจำโปรแกรมภายนอก เราอาจใช้สัญญาณนี้เป็นตัวอีนาเบิ้ล (CHIP ENABLE) หน่วยความจำโปรแกรมภายนอกก็ได้

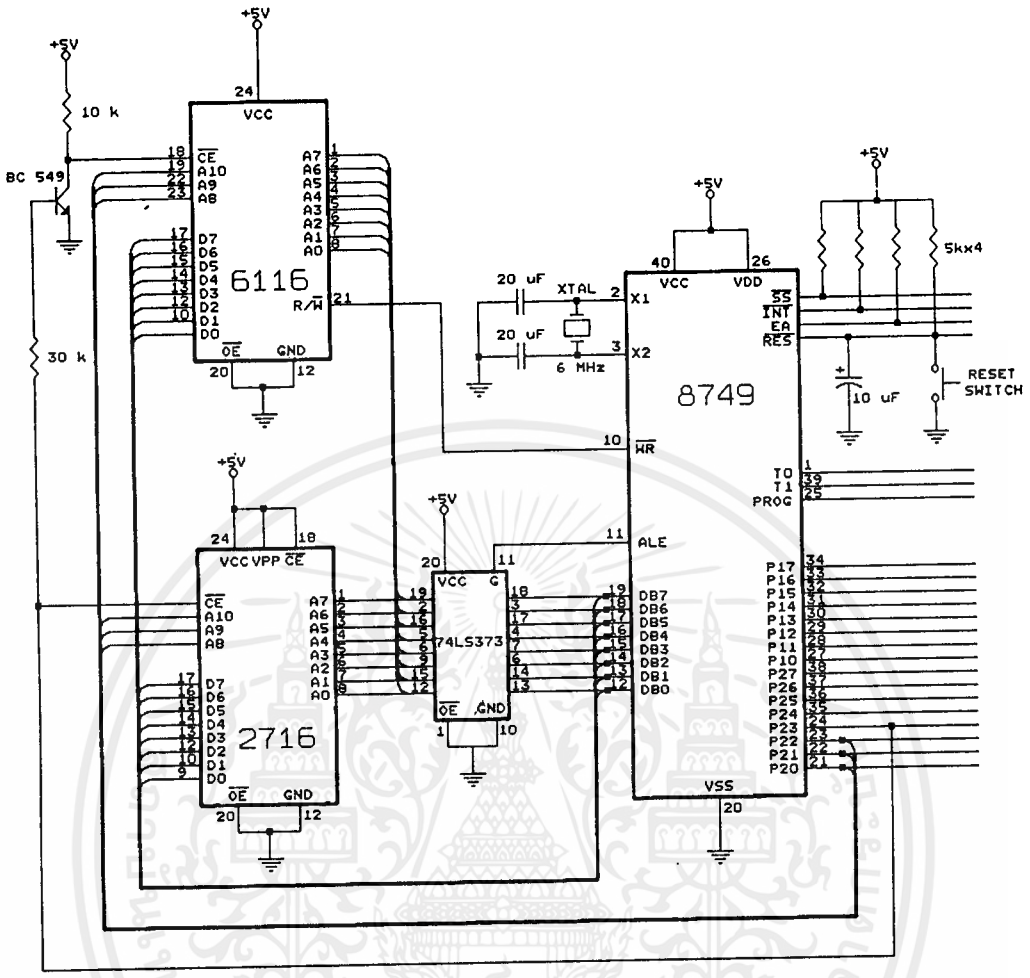
การต่อขยายหน่วยความจำข้อมูลภายนอก

จากที่กล่าวมาแล้วข้างต้นว่า เราสามารถต่อขยายหน่วยความจำข้อมูลภายนอกได้เป็น 256 ไบท์ หรือ 1 เพจ โดยเป็นอิสระกันกับหน่วยความจำข้อมูลภายใน วิธีการต่อแรมไว้ภายนอกนี้ ต้องมีการจัดแอดเดรสบัส ดาต้าบัส และสัญญาณควบคุมให้กับแรมด้วย เช่นเดียวกับกรณีขยายหน่วยความจำโปรแกรม จะแตกต่างกันที่สัญญาณควบคุมที่ใช้ไม่ใช่ PSEN แต่เป็นสัญญาณ WR หรือ RD แทน



รูปที่ 18 แสดงการต่อหน่วยความจำโปรแกรมภายนอก

ในวงจรนี้ผมใช้รอมเบอร์ 2716 เป็นหน่วยความจำที่มีแอดเดรสในช่วงแบลงค์แรก (000-7FF) และใช้แรมเบอร์ 6116 มาเป็นหน่วยความจำในช่วงแบลงค์หลัง (800-FFF) มาถึงตอนนี้ หลายคนอาจจะฉงนว่า หน่วยความจำแรมที่ต่อเพิ่มขึ้นนั้น เป็นหน่วยความจำโปรแกรม หรือหน่วยความจำข้อมูลกันแน่



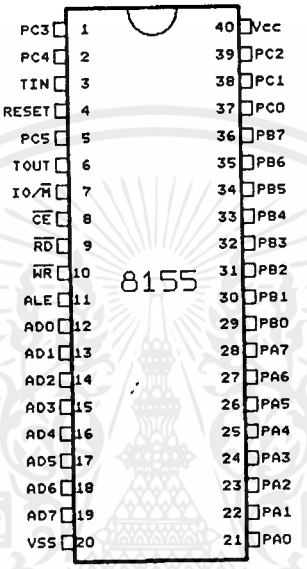
รูปที่ 19 แสดงการต่อหน่วยความจำข้อมูลภายนอก

ก็เห็นจะต้องให้คุณดูที่วงจรซึ่งจะพบว่า ผมไม่ได้ต่อขาควบคุมใดกับขา CE ของหน่วยความจำเลย ไม่ว่าจะเป็น PSEN, WR หรือ RD แต่ใช้เพียงขาแอดเดรส A_{11} เป็นตัวเลือกแบลงค์ของหน่วยความจำเท่านั้น และใช้สัญญาณ WR เพื่อต่อกับขา R/W ของแรม เพื่อเลือกว่าจะอ่าน หรือเขียนข้อมูลลงแรม

เราทราบกันมาแล้วว่า 4 บิทล่างของพอร์ท 2 (P_{20}-P_{23}) สามารถทำหน้าที่เป็นพอร์ท 2 และเป็นส่วนหนึ่งของแอดเดรสบัสด้วย แต่ในขณะที่ 8749 ติดต่อกับหน่วยความจำข้อมูลอยู่ พอร์ท 2 ส่วนนี้จะทำหน้าที่เป็นพอร์ทไบตลอดไซเคิล ดังนั้น หากผมเอาท์พุทแอดเดรส A_8-A_{11} ออกมาทางขา P_{20}-P_{23} นี้ก็เท่ากับว่า ผมสามารถเลือกเพจของหน่วยความจำข้อมูลได้ และก็สามารถใช้รอม 2716 ในวงจรในฐานะเป็นหน่วยความจำข้อมูลได้เช่นกัน

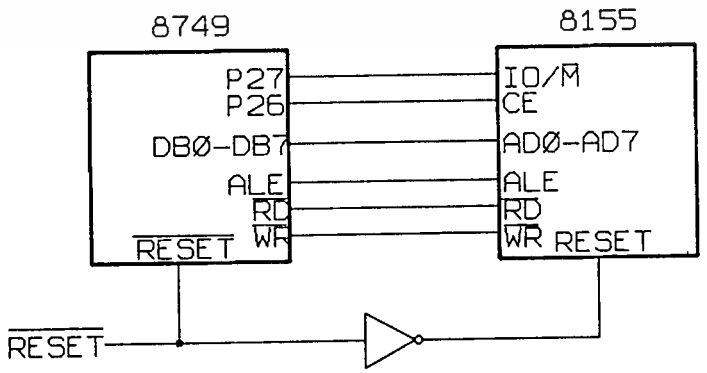
การต่อขยายพอร์ท โดยใช้ 8155 หรือ 8243

ก่อนอื่นขอกล่าวถึงการต่อขยายพอร์ทโดยใช้ 8155 เสียก่อน เพราะไอซีนี้มีฟังก์ชั่นการทำงานต่างๆ มากมาย ไม่ว่าจะเป็นพอร์ทแบบขนานทั้ง 3 พอร์ท, ไทเมอร์/เคาวน์เตอร์ ขนาด 14 บิท และหน่วยความจำแรมขนาด 256 ไบท์ ดังจะแสดงให้เห็นการจัดขาของ 8155 ไว้ในรูปที่ 20



รูปที่ 20 แสดงการจัดขาของ 8155

ไอซีเบอร์นี้ถูกออกแบบมาสำหรับใช้กับ ซีพียู 8085 และก็สามารถนำมาต่อกับ 8749 ได้อย่างสะดวก ดังจะเห็นได้จาก ระบบบัส (แอดเดรสบัสกับดาต้าบัสถูกมัลติเพล็กซ์รวมกัน), หน่วยความจำแรมขนาด 256 ไบท์และสัญญาณควบคุมต่างๆ เป็นต้น



รูปที่ 21 ลักษณะการต่อวงจรระหว่าง 8749 กับ 8155

จากวงจรนี้ จะเห็นว่าเราต้องอาศัยเอาท์พุทพอร์ท 2 บิท มาใช้งานเป็นสัญญาณควบคุม ขา CE และ IO/M ของ 8155 เมื่อผมต้องการติดต่อกับ 8155 ผมก็จำเป็นต้องส่งลอจิกศูนย์ออกมาให้ที่ขา CE ของ 8155 แล้วเลือกว่า ต้องการติดต่อกับหน่วยความจำหรือหน่วยอินพุท เอาท์พุท จากนั้นก็สามารถใช้งาน 8155

บทที่ 2

8255

พอร์ทแบบขนานที่โปรแกรมได้

ในการนำเอาไมโครโปรเซสเซอร์ไปใช้งานนั้น จำเป็นต้องให้ไมโครโปรเซสเซอร์สามารถติดต่อกับโลกภายนอกได้ซึ่งก็คือ ให้มันสามารถส่งสัญญาณมาควบคุมอุปกรณ์ต่าง ๆ ได้ เช่น สเต็ปปิ้งมอเตอร์, ควบคุมอุปกรณ์ไฟฟ้า ต่าง ๆ ส่วนที่ทำได้ไมโครโปรเซสเซอร์สามารถติดต่อกับโลกภายนอกได้ที่รู้จักกันดี คือ พอร์ท (Port) ซึ่งก็มีอยู่หลายลักษณะด้วยกัน เช่นเป็นไอซีแบบไตรสเตท (Tri-state) เบอร์ 74LS244 หรือพวกแลทต์ (Latch) เช่น 74LS374 เหล่านี้สามารถนำมาต่อใช้งานกับ CPU ได้โดยง่ายที่สุด โดยตัว CPU จะเป็นตัวควบคุมการอ่านเขียนพอร์ท หากเป็นการอ่านข้อมูลจากพอร์ทก็มักจะใช้ไอซีแบบไตรสเตทเป็นพอร์ทอินพุทโดยตัว CPU จะส่งสัญญาณไปเปิดเกทของไตรสเตทนี้ให้ข้อมูลเข้าสู่สายข้อมูลแบบ (Data Bus) และเข้าสู่ไมโครโปรเซสเซอร์หรือ CPU ต่อไป แต่สำหรับพอร์ทเอ้าพุทก็จะใช้แลทต์ฟลิบ-ฟลอบ ทำหน้าที่รับสัญญาณข้อมูลจากไมโครโปรเซสเซอร์มาแลทต์ไว้ที่ตัวมัน (CPU ส่งสัญญาณมาทริก) เพื่อให้อุปกรณ์ภายนอกนั้นรับสัญญาณจากตัวแลทต์นี้ไปอีกทีหนึ่ง ที่ทำเช่นนี้เพราะตัวไมโครโปรเซสเซอร์หรือ CPU นี้สามารถทำงานเร็วมาก ซึ่งในช่วงของการส่งข้อมูลออกพอร์ทจะใช้เวลาไม่กี่ไมโครโปรเซท ($_{uS}$) ซึ่งอาจทำให้อุปกรณ์ภายนอกรับไม่ทัน

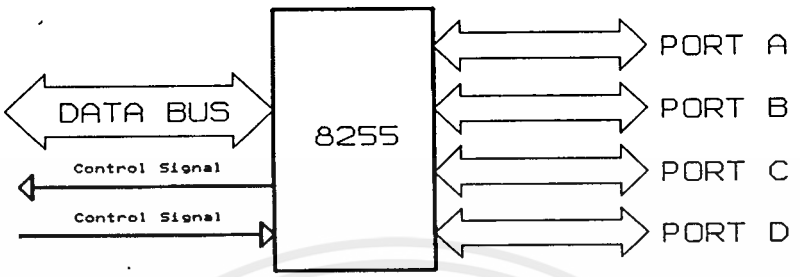
มีบริษัทต่าง ๆ เล็งเห็นความสำคัญของการติดต่อระหว่าง CPU กับอุปกรณ์ภายนอกนี้จึงได้ทำไอซีสำเร็จรูปขึ้นหลายเบอร์ และที่จะขอนำมากล่าวในที่นี้ก็คือไอซีเบอร์ 8255 ซึ่งเป็นของบริษัท อินเทล ซึ่งได้ออกแบบมาใช้กับ CPU เบอร์ 8080 แต่เราสามารถนำมาประยุกต์ใช้กับเบอร์อื่นๆ ได้โดยไม่ยาก

สาเหตุที่ 8255 เป็นที่นิยมมากก็เพราะว่ามันสามารถถูกโปรแกรมให้ทำงานในลักษณะต่างๆ ไม่ว่าจะเป็น อินพุท, เอ้าพุท หรือแม้แต่แบบ แฮนด์เชคกิ้ง (Handshaking) ได้ ทั้งยังราคาถูกอีกต่างหาก

ลักษณะทั่วไปของ 8255

เป็นไอซีขนาด 40 ขา ตัวแบน โดยแยกเป็นลักษณะของบล็อกง่ายๆ ดังรูปที่ 1 คือจะมีพอร์ทให้ใช้งานได้ถึง 3 พอร์ท (เป็นขนาด 8 บิท) พอร์ท A, พอร์ท B, พอร์ท C โดยพอร์ท C นี้สามารถแยกได้เป็น 2 ส่วน คือ พอร์ท C บนตั้งแต่ PC_4-PC_7 จำนวน 4 บิทและพอร์ท C ล่างตั้งแต่ PC_0-PC_3

โดยพอร์ททุกพอร์ท (A,B,C) สามารถโปรแกรมได้ให้เป็น อินพุทหรือเอ้าพุท ซึ่งจะได้กล่าวถึงการโปรแกรมในรายละเอียดต่อไป



รูปที่ 1 แสดงบล็อกเส้นทางของพอร์ท 8255

ในรูปที่ 2 จะเห็นโครงสร้างภายในที่แสดงถึงกลุ่มควบคุมที่มีอยู่ 3 กลุ่มคือ

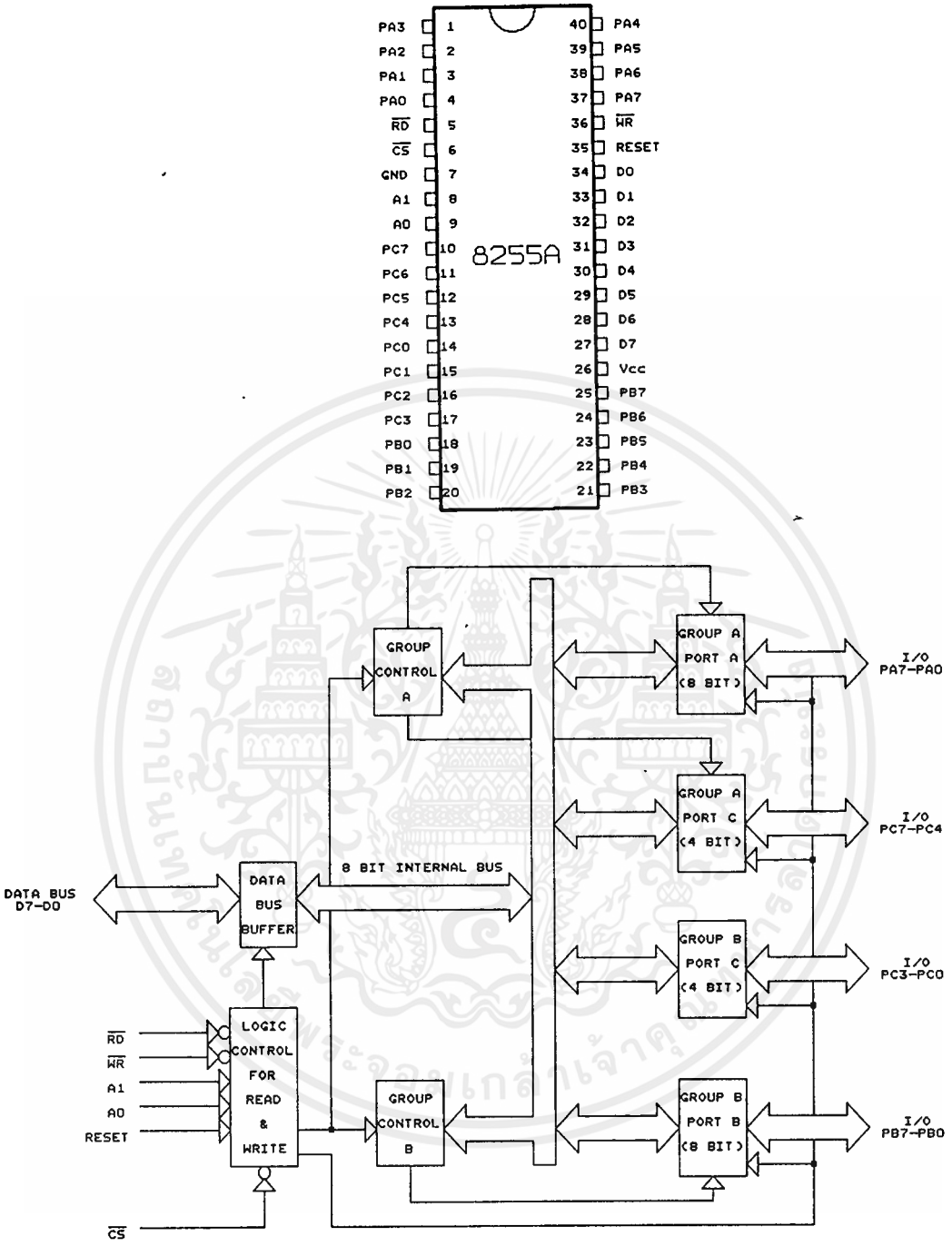
- กลุ่มควบคุมชุด A จะควบคุมพอร์ท A และพอร์ท C บน
- กลุ่มควบคุมชุด B จะควบคุมพอร์ท B และพอร์ท C ล่าง
- กลุ่มควบคุมโลจิกการเขียนและอ่าน

การทำงานของ 8255 จะใช้สัญญาณควบคุมจากตัวไมโครโปรเซสเซอร์มาควบคุมโดยจะมีการส่งคำสั่ง (Control word) มาที่กลุ่มควบคุมชุด A,B แล้วกลุ่มควบคุมชุดนี้ก็จะสั่งต่อไปที่พอร์ทเพื่อให้เป็นไปตามข้อกำหนดของคำสั่งนั้นๆ เช่น ให้พอร์ท A เป็นอินพุท พอร์ท B เป็นเอ้าพุทเหล่านี้เป็นต้น ส่วนกรณีเมื่อมีการอ่านเขียนพอร์ทจาก CPU นั้น กลุ่มควบคุมโลจิกการเขียนอ่านจะเป็นตัวที่ส่งสัญญาณไปบอกแก่กลุ่มควบคุมชุด ในแต่ละชุดอีกที ทั้งนี้แล้วแต่ว่า CPU จะมีการอ่านเขียนพอร์ทของกลุ่มควบคุมชุดใด

ต่อไปเรามาดูถึงความหมายของ ขาต่างๆ ของไอซี 8255 เพื่อจะได้ต่อใช้งานได้อย่างถูกต้องต่อไป

D0-D7 เป็นขาข้อมูลของ 8255 ที่ใช้ติดต่อกับตัวไมโครโปรเซสเซอร์ซึ่งข้อมูลที่จะเข้าออกสู่พอร์ทต่างๆ ของ 8255 จะต้องผ่านขาข้อมูลนี้

CS เป็นขาอินพุทที่รับสัญญาณโลจิก "0" จากภายนอกเพื่อแสดงว่าต้องการเลือกใช้ไอซีเบอร์นี้ หากได้รับโลจิก "1" ก็จะทำให้ไอซีตัวนี้ไม่ทำงานคือไม่รับรู้สัญญาณใดๆ ทั้งสิ้น



รูปที่ 2 แผนผังวงจรภายในและการจัดขาของไอซี 8255

RD เป็นขาอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์ โดยหากมีโลจิกเป็น "0" จะเป็นการแสดงว่า CPU ต้องการที่จะอ่านข้อมูลจากตัว 8255

WR เป็นขาอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์ โดยหากมีโลจิกเป็น "0" ก็จะเป็นการแสดงว่า CPU ต้องการที่จะเขียนข้อมูลจากตัว 8255

A0-A1 เป็นอินพุทที่รับแอดเดรสจากตัวไมโครโปรเซสเซอร์ ที่ถอดรหัสตำแหน่งของ 8255 เรียบร้อยแล้ว โดยจะมีตำแหน่งใช้งาน 4 ตำแหน่งเพื่ออ่านเขียนรีจีสเตอร์ (พอร์ท) ของ 8255 ที่มีอยู่ด้วยกัน 4 ตัว

RESET เป็นขาอินพุทที่รับสัญญาณจากภายนอกเข้ามาทำการรีเซทตัว 8255 โดยหากได้รับโลจิก "1" จะทำให้พอร์ททุกพอร์ทเป็นอินพุทพอร์ทหมด ทั้งนี้เพื่อไม่ต้องการให้มีสัญญาณออกไปกวนต่อ ระบบภายนอกเพื่อ 8255 ได้รับสัญญาณรีเซท

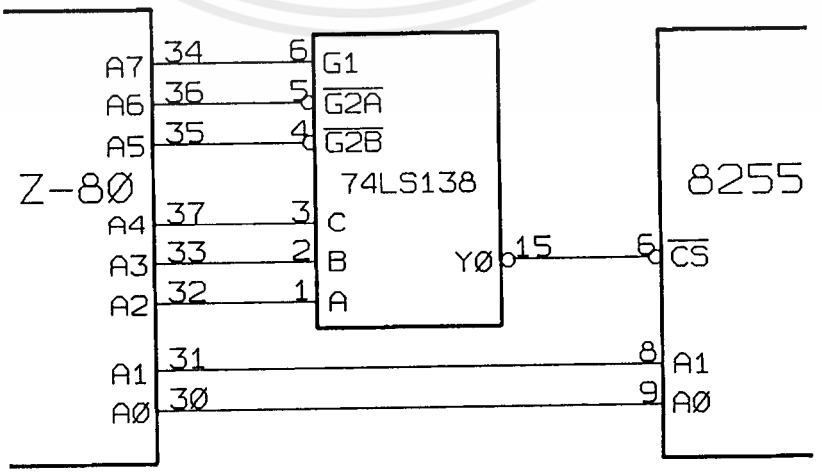
PA_0-PA_7 เป็นขาสัญญาณพอร์ท A ที่ใช้ติดต่อกับโลกภายนอก

PB_0-PB_7 เป็นขาสัญญาณพอร์ท B ที่ใช้ติดต่อกับโลกภายนอก

PC_0-PC_7 เป็นขาสัญญาณพอร์ท C ที่ใช้ติดต่อกับโลกภายนอก ซึ่งพอร์ทนี้จะแบ่งออกเป็น 2 กลุ่มคือ PC_0-PC_3 และ PC_4-PC_7 ซึ่งสามารถโปรแกรมแยกกันได้อีกต่างหาก

การต่อใช้งาน 8255

หากดูที่ขาของ 8255 ที่รูปที่ 2 แล้วจะสังเกตเห็นได้ว่าเราสามารถต่อขา 8255 บางส่วนได้โดยตรงกับขาไมโครโปรเซสเซอร์เลย (Z-80) เช่นขา D0-D7, A0-A1 เป็นต้น หากแต่บางขาเราจำเป็นต้องมีการดัดแปลงสัญญาณที่ได้จาก CPU (ซึ่งกรณีนี้เราใช้เบอร์ Z-80) เสียก่อน โดยหากเราถอดรหัสแอดเดรสของ 8255 ให้เป็นพอร์ทที่แอดเดรส 10H,11H,12H,13H เราสามารถทำ ดีโค้ดเดอร์ได้โดยง่ายดังรูปที่ 3

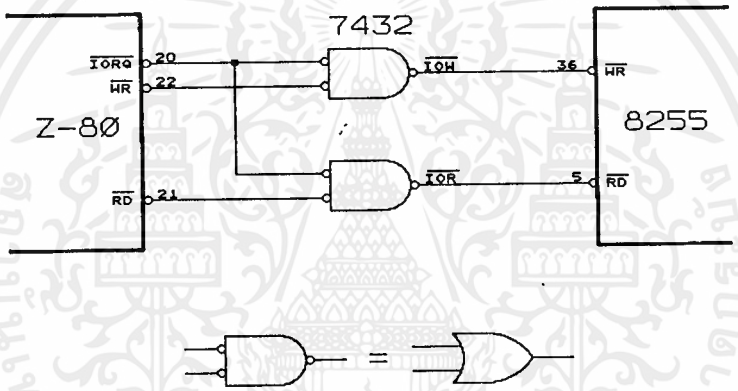


รูปที่ 3 แสดงการดีโค้ดแอดเดรสพอร์ทให้ 8255

สังเกตได้ว่า CS จะแอดตีฟทุกครั้งหาก Z-80 อ้างพอร์ทที่แอดเดรส 000100XX โดยค่าของ XX คือ A0,A1 ที่เราจะต่อตรงเข้ากับ 8255 เพื่อทำการเลือกรีจีสเตอร์ควบคุมและพอร์ททั้งสามของ 8255

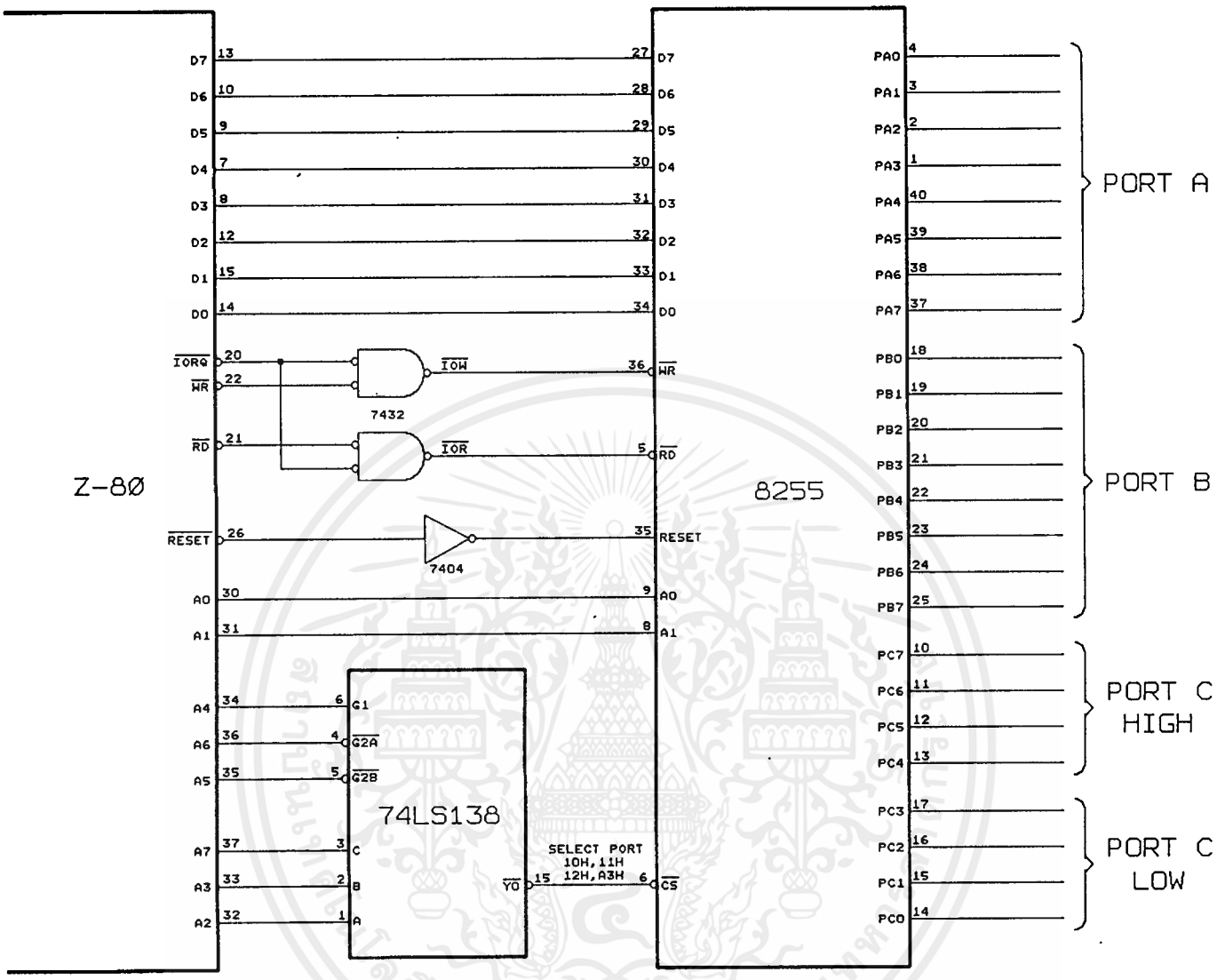
สัญญาณการควบคุมอีกส่วนที่สำคัญก็คือ สัญญาณควบคุมการอ่านเขียนพอร์ทของ 8255 หรือออกสู่พอร์ทที่ต้องการ และเช่นเดียวกันหาก RD ได้รับโลจิก "0" ก็จะเป็นการอ่านข้อมูลพอร์ทจากตัว 8255 เข้าสู่ CPU

ซึ่งหากต่อ 8255 เข้ากับ ตัว Z-80 แล้วเราจะเป็นต้องทำสัญญาณการอ่านเขียนพอร์ทของ Z-80 ให้ถูกต้องเสียก่อนโดยต่อดังรูปที่ 4



รูปที่ 4 การเชื่อมต่อสัญญาณอ่านเขียนพอร์ทระหว่าง Z-80,8255

และสุดท้ายคือสัญญาณ RESET ซึ่งของ 8255 จะรับแอดตีฟที่โลจิก "1" ซึ่งตัว Z-80 จะให้สัญญาณ RESET แอดตีฟ "0" ฉะนั้นเราจะต้องนำมาผ่านอินเวอร์เตอร์ก่อนก็จะได้ลักษณะการต่อร่วมกับ CPU ดังนี้



รูปที่ 5 แสดงการต่อ 8255 ร่วมกับ Z-80

การโปรแกรม 8255

เราได้ทราบมาแล้วในรูปที่ 2 ว่าโครงสร้างภายในของ 8255 มีกลุ่มควบคุมชุดอยู่ 3 กลุ่ม ซึ่งทั้งสามกลุ่มนี้จะทำงานร่วมกันดังที่กล่าวมา และเราสามารถจะควบคุมการทำงานของพอร์ทจาก CPU ได้โดยสั่งงานมาที่กลุ่มควบคุมดังกล่าว แต่ตัว CPU จะมองเห็น 8255 เป็น 4 พอร์ทด้วยกัน โดยแต่ละพอร์ทเสมือนเป็น รีจีสเตอร์ ที่ CPU สามารถจะทำการ อ่าน/เขียน ได้ แต่ละพอร์ทจะอยู่คนละแอดเดรสกัน ดังที่เราได้ทำการดีโค๊ดให้ 8255 ที่แอดเดรส 10H,11H,12H,13H (ตามสัญญาณ A0-A1) และเราจะได้ตำแหน่งของพอร์ท 8255 แต่ละตัวดังนี้

10H ==========> พอร์ท A

11H ==========> พอร์ท B

12H ==========> พอร์ท C

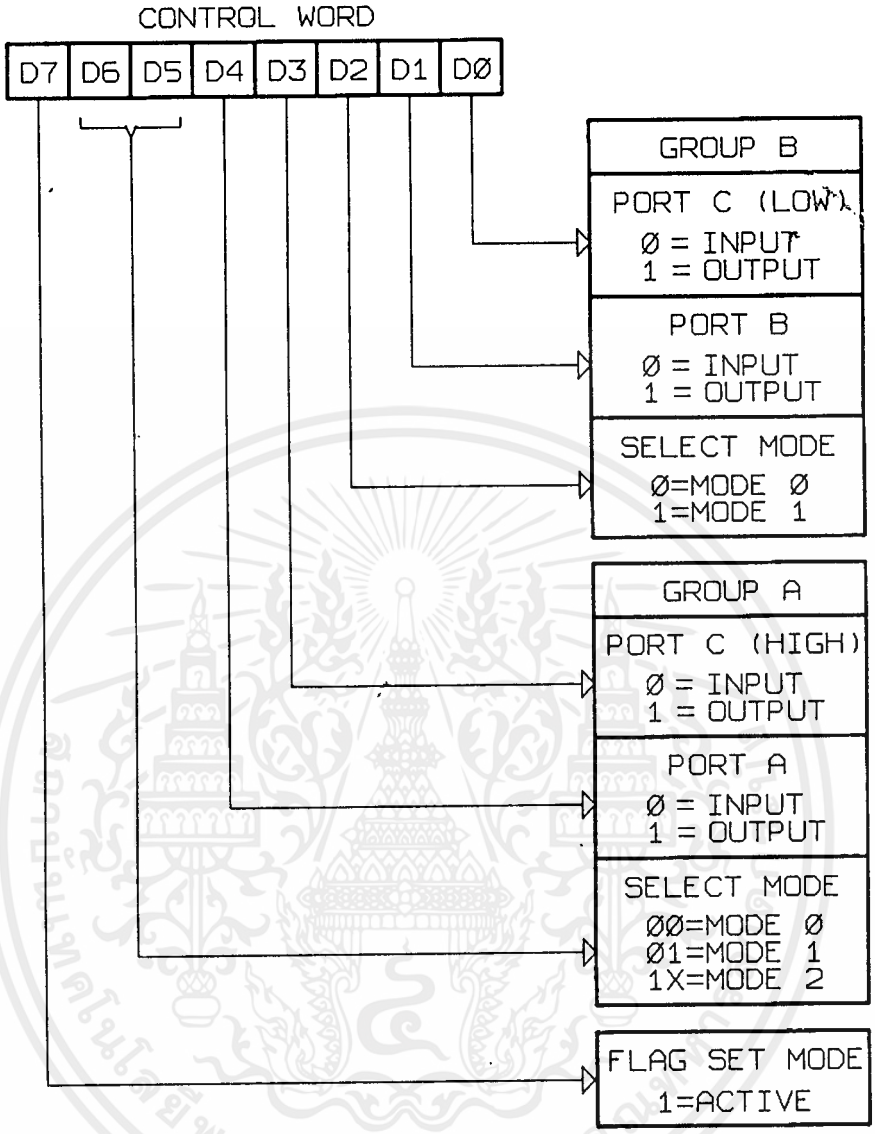
13H ==========> พอร์ท Control

ซึ่งหากมีการอ่านเขียนไปยังพอร์ทดังกล่าวก็จะใช้ร่วมกับสัญญาณ RD,WR โดย WR หมายถึงเอ้าพุท ข้อมูลและ RD แอดดีฟหมายถึงอินพุทข้อมูล ดังนั้นเราจะได้โลจิกที่ขาของ 8255 ในลักษณะต่างๆ ดังนี้

RD	WR	A1	A0	ความหมาย
1	0	0	0	เขียนพอร์ท A ซึ่งเป็นข้อมูล
0	1	0	0	อ่านพอร์ท A ซึ่งเป็นข้อมูล
1	0	0	1	เขียนพอร์ท B ซึ่งเป็นข้อมูล
0	1	0	1	อ่านพอร์ท B ซึ่งเป็นข้อมูล
1	0	1	0	เขียนพอร์ท C ซึ่งเป็นข้อมูล
0	1	1	0	อ่านพอร์ท C ซึ่งเป็นข้อมูล
1	0	1	1	เขียนข้อมูล ซึ่งเป็นรหัสควบคุม
0	1	1	1	อ่านเข้ามา ซึ่งไม่มีความหมายใด

รูปที่ 6 แสดงตารางช่องโลจิกเมื่อทำการติดต่อกับ 8255

การใช้งานเราจะต้องส่งรหัสควบคุม (Control code) เข้าไปยังพอร์ทควบคุม (หรือเรียกอีกอย่างว่า รีจิสเตอร์ควบ) ซึ่งจะเป็นข้อมูลขนาด 1 ไบท์ส่งไปที่ แอดเดรส 13H (กรณีนี้เราถอดรหัสไว้ที่ 13H) โดยความหมายของแต่ละบิทที่เราส่งไปโปรแกรมการทำงานเป็นดังนี้



รูปที่ 7 ความหมายของแต่ละบิทในรหัสควบคุม

บิท D7 เป็นบิทที่แสดงว่า ในไบท์นี้เป็นรหัสควบคุม ถ้าเป็น "1" โดยแต่ละบิทจะมีผลต่อการเปลี่ยนแปลงโหมดต่างๆ ของ 8255 หากเป็น "0" จะเป็นการเซตบิทของพอร์ท C

บิท D6,D5 เป็นการเลือกโหมดของพอร์ท A ซึ่งจะมีอยู่ด้วยกัน 3 โหมด คือ 0,1,2

บิท D4 เป็นการกำหนดให้พอร์ท A ให้เป็น อินพุทหรือเอ้าพุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็แสดงว่าให้เป็น เอ้าพุท

บิท D3 เป็นการกำหนดให้พอร์ท C บน ให้เป็นอินพุทหรือเอ้าพุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็แสดงว่าให้เป็น เอ้าพุท

บิท D2 เป็นการกำหนดโหมดการทำงานของพอร์ท B โดยหากเป็น "0" หมายถึง เลือกให้พอร์ท B ทำงานในโหมด 0 หากเป็น "1" เป็นการเลือกให้พอร์ท B ทำงานในโหมด 1

บิท D1 เป็นการกำหนดให้พอร์ท B ให้เป็น อินพุทหรือเอ้าพุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็แสดงว่าให้เป็น เอ้าพุท

บิท D0 เป็นการกำหนดให้พอร์ท C ล่าง ให้เป็น อินพุทหรือเอ้าพุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็แสดงว่าให้เป็น เอ้าพุท

การโปรแกรมจะเริ่มจากการส่งค่ารหัสควบคุม 1 ไบท์ ดังที่กล่าวนี้ไปสู่พอร์ทควบคุม หลังจากนั้นหากต้องการเรียกไปถึงพอร์ทใดก็สามารถอ้างได้ตามแอดเดรสทันที เช่น ต้องการดปรแกรมให้พอร์ทA,B,C ทั้งหมดให้เป็น เอ้าพุทพอร์ท เราก็จะได้รหัสควบคุมเป็น 1000000 หรือ 80H เราก็จะสั่งเป็น

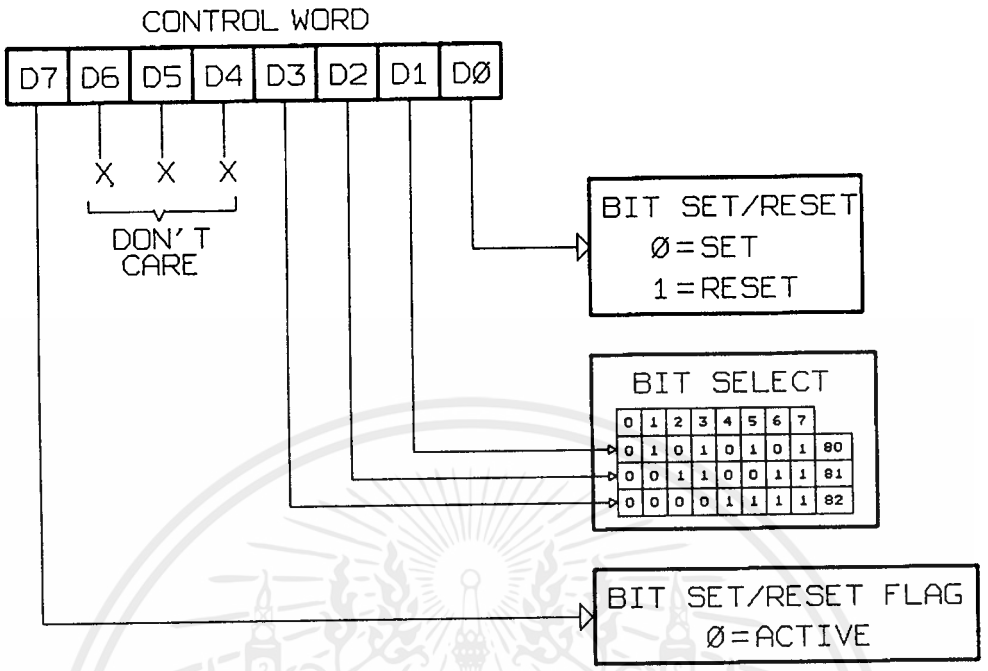
```
LD A,080H      ;กำหนดรหัสควบคุม
OUT (013H),A ;เป็นการส่งรหัสควบคุมสู่รีจีสเตอร์ควบคุม 8255
```

จากนี้ทั้งสามพอร์ทก็จะเป็น เอ้าพุทพอร์ทให้เราตามต้องการเมื่อเราจะส่งค่าออกไปก็สามารถกระทำได้โดยง่าย เช่นต้องการส่งค่า 088H ออไปที่พอร์ท B,C เราจะทำดังนี้

```
LD A,088H      ;ค่าข้อมูลที่ต้องการส่ง
OUT (011H),A   ;ส่งออกไปพอร์ท B
OUT (012H),A   ;ส่งออกไปพอร์ท C
```

<u>กรณีพิเศษของรหัสควบคุม</u>

ปกติรหัสที่เราต้องส่งไปถึงพอร์ทควบคุมหรือรีจีสเตอร์ควบคุมนี้จะต้องเป็นการเซทโหมด, พอร์ท อินพุทเอ้าพุทและในรหัสนั้น บิท 7 จะต้องเป็น "1" เสมอ <u>ทีนี้หาก บิท 7 นี้เป็น "0"</u> บ้างจะเกิดอะไรขึ้น ถ้าหากบิท 7 เป็น "0" และถูกส่งไปที่ แอดเดรสของพอร์ทควบคุมแล้ว 8255 จะถือว่าเป็นคำสั่งของการ เซท/รีเซท บิทของพอร์ท C ทันที โดยจะมี ฟอร์แมทดังรูปที่ 8 นี้



รูปที่ 8 แสดงถึงรหัสควบคุมที่ใช้ในการเซท/รีเซท บิท

ตัวอย่างเช่น

```
LD A,80H          ;รหัสควบคุมให้ทั้งสามพอร์ทเป็น OUTPUT
OUT (13H),A       ;ส่งสู่รีจีสเตอร์ควบคุม
LD A,00001001B    ;รหัสควบคุมใช้เซทบิทที่ 4 ของพอร์ท C เป็น "1"
OUT (013H),A      ;บิท 4 พอร์ท C เป็น "1"
DEC A             ;เปลี่ยนเป็น รีเซท
OUT (013H),A      ;กำหนดให้ บิท 4 พอร์ท C เป็น "0"
```

จะเห็นว่าสามารถนำไปประยุกต์ใช้งานได้ เช่นเป็นตัวสร้าง พัลซ์,และกำหนดใช้งาน เปิด-ปิด อุปกรณ์ด้วย พอร์ท C ที่มีคำสั่งไม่ยุ่งยากและเป็นอิสระเป็นต้น

การทำงานในโหมด 0

จัดว่าเป็นโหมดพื้นฐานที่นิยมใช้กันมากที่สุด เนื่องด้วยความง่ายตรงไปตรงมา คือทั้งสามพอร์ท เราสามารถจะให้พอร์ทใดเป็น อินพุท, เอ้าพุท ได้ โดยเฉพาะพอร์ท C ยังแยกให้เป็น 2 ชุดๆ ละ 4 บิท ซึ่งในแต่ละชุดนี้ก็สามารถจะโปรแกรมให้เป็นอินพุทหรือเอ้าพุทได้เสมือน 4 พอร์ท คือ พอร์ท A, พอร์ท B, พอร์ท C บนและพอร์ท C ล่าง (แต่โปรแกรมแยกเฉพาะบิทในแต่ละพอร์ทไม่ได้)

1	Ø	Ø	X	X	Ø	X	X

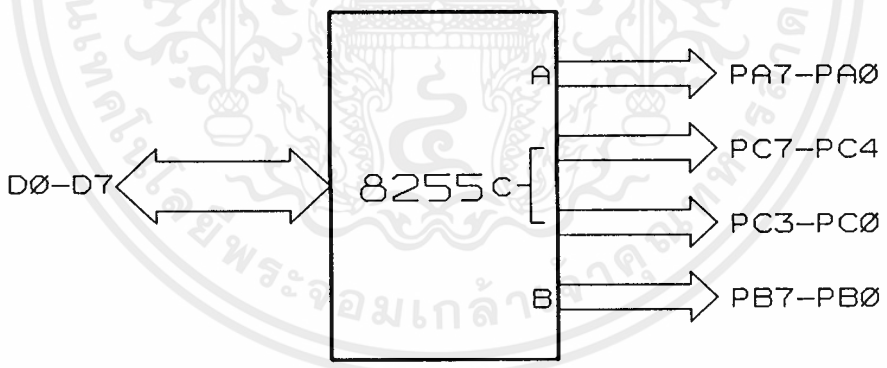
BIT 7 6 5 4 3 2 1 Ø

รูปที่ 9 แสดงรหัสคำสั่งของโหมด 0

ซึ่งหากเราดูที่ รหัสคำสั่ง แล้วจะเห็นว่ามีอยู่ 4 บิท ที่ถูกกำหนดตายตัวคือ บิท 7,บิท 5, บิท 6 และบิท 2 ส่วนที่เหลืออีก 4 บิท ก็คือข้อกำหนดว่าจะให้พอร์ทใดเป็น อินพุท/เอ้าพุท นั้นเอง ซึ่งหากเราให้พอร์ทใดเป็น อินพุทเราก็ใส่โลจิก "1" ที่บิทนั้น หรือหากต้องการให้พอร์ทใดเป็น เอ้าพุทก็ให้ใส่ "0" ที่บิทนั้น จะเห็นได้ว่าจะมีความเป็นไปได้ในการกำหนดลักษณะของพอร์ทในโหมดนี้อยู่ 16 อย่าง ด้วยกัน

ตัวอย่างเช่น 1) ต้องการให้ทุกพอร์ทเป็น เอ้าพุทหมด เราจะได้รหัสคำสั่งเป็น L:-

1	Ø	Ø	Ø	Ø	Ø	Ø	Ø



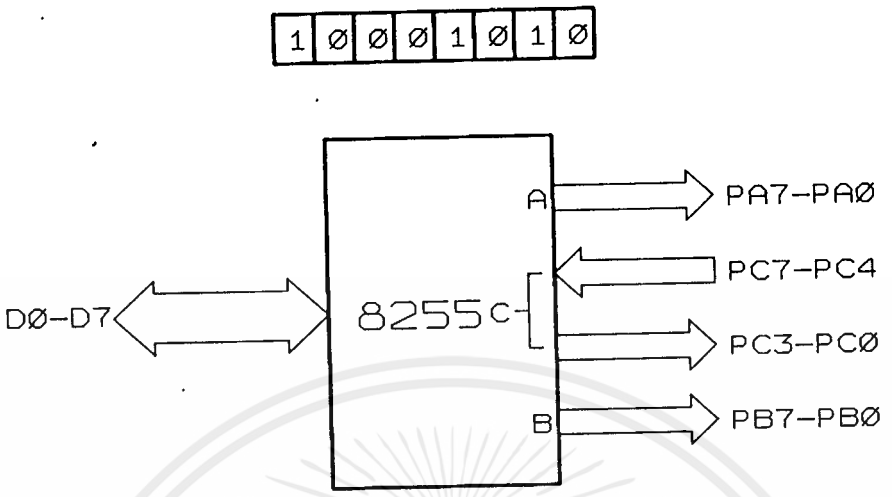
ตัวอย่าง 2) ต้องการให้ พอร์ท A ===> เอ้าพุท

พอร์ท B ===> อินพุท

พอร์ท C บน ===> อินพุท

พอร์ท C ล่าง ===> เอ้าพุท

เราจะได้รหัสคำสั่งเป็น :-



รูปที่ 10 แสดงตัวอย่างพอร์ทที่ถูกโปรแกรมในโหมด 0

หน้าถอยหลัง, บังคับจำนวนสเตปในการหมุนแต่ละครั้งยิ่งถ้าเราใช้ สเตปปิ้งมอเตอร์ 2 ตัว เพื่อควบคุมในลักษณะของแกน X,แกน Y แล้ว เราก็สามารถจะจดจำตำแหน่งต่างๆ นำไปประยุกต์ใช้งานเป็นเครื่องเขียนภาพ (PLOTER) และแขนหุ่นยนต์ในอุตสาหกรรม เป็นต้น

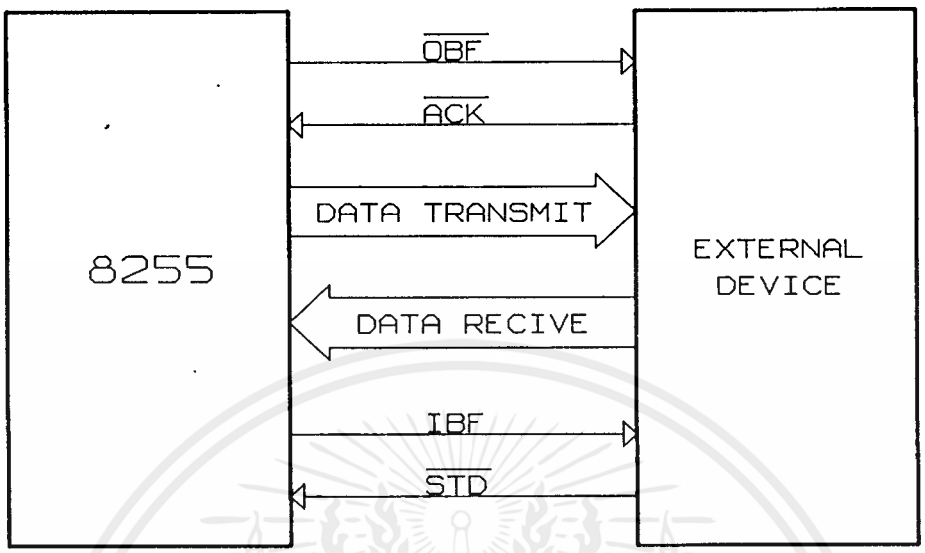
```
                2000      8                ORG     2000H
                          9
                0023      11  STPINT       EQU     23H
                0020      12  STPOUT       EQU     20H
                3311      13  DATA         EQU     1133H
2000    3E  80    14  STP          LD      A,80H
2002    D3  23    15               OUT     (STPINT),A     ;INITIAL
2004    01  3311  16               LD      BC,DATA     ;B=1 PHASE C=2 PHASE
2007    78        17               LD      A,B
2008    D3  20    18  STPI         OUT     (STPOUT),A
200A    CD  1320  19               CALL    STPDEL
200D    07        20               RLCA
```

```
200E    47          21              LD      B,A             ;ALTERNATE PHASE
200F    79          22              LD      A,C
2010    48          23              LD      C,B
2011    18  F5      24              JR      STP1
                    25
2013    08          26  STPDEL      EX      AF,AF'
2014    21  0020    27              LD      HL,2000H
2017    2B          28  STPDEL1     DEC     HL
2018    7C          29              LD      A,H
2019    B5          30              OR      L
201A    20  FB      31              JR      NZ,STPDEL1
201C    08          32              EX      AF,AF'
201D    C9          33              RET
 0 Error (s) Detected.
 30 Absolute Bytes. 7 Symbols Detectd.
```

รูปที่ 11 แสดงโปรแกรมการกระตุ้น One-rwo excetation

การทำงานในโหมด 1

เป็นความสามารถพิเศษที่บริษัทผู้ผลิตเพิ่มขึ้นมาให้แก่ผู้ใช้คือ นอกจาก 8255 จะถูกใช้เป็นพอร์ทแบบขนานที่สามารถโปรแกรมให้เป็น อินพุท-เอ้าพุทพอร์ทได้แล้ว ยังสามารถให้ทำงานในเรื่องของการสื่อสารอัตโนมัติได้อีกด้วย นั่นคือ ในโหมด 1 เป็นจะใช้ในการรับ-ส่งข้อมูลแบบมีการตรวจสอบสัญญาณก่อน (Handshaking) โดยใช้อินพุทเอ้าพุทของพอร์ท A และพอร์ท B เป็นหลัก ส่วนพอร์ท C บนจะใช้เป็นตัวตรวจสอบสัญญาณ (Handshake) ของพอร์ท A และพอร์ท C ล่างใช้เป็นตัวตรวจสอบสัญญาณสำหรับพอร์ท B ฉะนั้นการรับ-ส่งข้อมูล นอกจากมีข้อมูลแล้วก็จะมีสัญญาณควบคุมเพิ่มด้วยดังแสดงในรูปที่ 12



รูปที่ 12 แสดงสัญญาณต่างเมื่อ 8255 อยู่ในโหมด 1

วิธีการทำ Handshake นี้จะมีประโยชน์มากเพราะปกติอุปกรณ์ภายนอกมักจะทำงานได้ช้ากว่าตัวไมโครโปรเซสเซอร์อยู่แล้ว ด้วยวิธีการนี้จะทำให้ตัวไมโครโปรเซสเซอร์สามารถติดต่อกับอุปกรณ์ภายนอกด้วย 8255 ได้อย่างมีประสิทธิภาพ

ในรูปที่ 12 นั้นจะแสดงให้เราเห็นถึงบล็อกของการติดต่อระหว่างอุปกรณ์ภายนอกกับ 8255 ซึ่งได้กำหนดให้พอร์ท A เป็นเอ้าพุทพอร์ท B เป็น อินพุทพอร์ท (เราอาจกำหนดให้อยู่ในลักษณะอื่นก็ได้) และเช่นเดียวกับโหมด 0 ที่เราได้กล่าวมาแล้วคือ ก่อนที่เราจะใช้งาน 8255 อันดับแรกเลยเราต้องทำการโปรแกรมมันก่อนโดยการส่งคำสั่งควบคุม ขนาด 1 ไบท์ไปที่รีจีสเตอร์ควบคุมของ 8255 ก่อน ซึ่งหากเรากำหนดให้พอร์ทเป็นดังรูปที่ 12 เราจะได้คำสั่งควบคุมเป็น

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	X	1	1	X

ในบิทอื่น ๆ นอกจากบิทที่ 3,0 นั้นนักศึกษาสามารถเข้าใจได้เพราะเคยกล่าวมาแล้ว ส่วนบิทที่ใส่เครื่องหมาย X นั้นเราต้องมาพิจารณาเพื่อใส่ค่าต่อไป คือในลักษณะการทำงานโหมด 1 นั้น เราจะมีตารางที่แสดงถึงการใช้ พอร์ท C เพื่อเป็นสัญญาณควบคุมทั้งพอร์ท A และ พอร์ท B ในกรณีอินพุทและเอ้าพุทดังนี้

PIN	INPUT	OUTPUT
PCØ	$INTR_B$	$INTR_B$
PC1	IBF_B	$\overline{OBF}_B$
PC2	$\overline{STB}_B$	$\overline{ACK}_B$
PC3	$INTR_A$	$INTR_A$
PC4	$\overline{STB}_A$	I/O
PC5	IBF_A	I/O
PC6	I/O	$\overline{ACK}_A$
PC7	I/O	$\overline{OBF}_A$

ตารางที่ 1 แสดงถึงขาของพอร์ท C ที่ถูกใช้เป็นสัญญาณควบคุมของพอร์ท A,B ในการทำงานโหมด 1 ทั้งกรณีอินพุทและเอ้าพุท

ฉะนั้น จากตารางเราจะเห็นว่าในบิทที่ 3 นั้นขึ้นอยู่ว่าผู้ใช้ต้องการกำหนดให้ PC6, PC7 เป็นอินพุทหรือเอ้าพุท (โดยใส่ 1 เมื่อต้องการเป็นอินพุทและใส่ 0 เมื่อต้องการให้เป็น เอ้าพุท) ส่วนในบิทที่ 0 นั้นไม่สนใจเพราะเราไม่สามารถกำหนดอะไรได้ (PC0-PC2) ถูกนำไปเป็นสัญญาณควบคุมแก่พอร์ท B แล้วและ PC3 ถูกนำไปเป็นสัญญาณควบคุมแก่พอร์ท A แล้วเช่นกัน

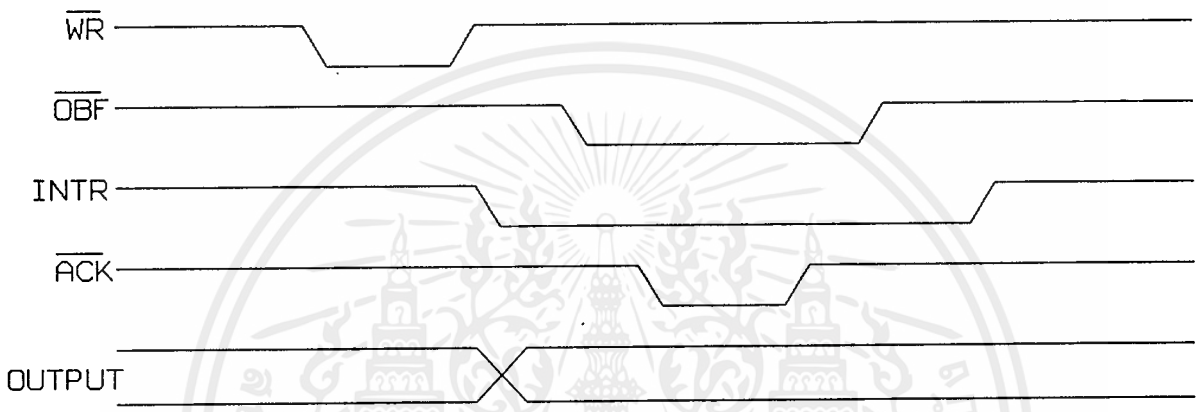
ลำดับสัญญาณในกรณีเอ้าพุทพอร์ทโหมด 1

การเอ้าพุทข้อมูล หมายถึงการส่งข้อมูลจากตัว CPU ออกมาสู่ตัว 8255 ที่พอร์ทนั้น ๆ เพื่อรอสัญญาณการรับข้อมูลจากอุปกรณ์ภายนอกรับเอาข้อมูลนี้ไป สัญญาณที่ใช้ในกรณีเอ้าพุทข้อมูลที่ตัว 8255 มีดังนี้ :-

- OBF (OUTPUT BUFFER FULL) เป็นเอ้าพุท จะแอคตีฟที่โลจิก 0 เป็นตัวบอกว่าขณะนี้ที่ตัว 8255 มีข้อมูลจาก CPU อยู่ยังไม่ถูกอ่านจากอุปกรณ์ภายนอก

- ACK (ACKNOWLEDGE) เป็นอินพุทแอคตีฟที่โลจิก 0 เป็นสัญญาณจากอุปกรณ์ภายนอกที่ส่งเพื่อรับเอาข้อมูลจาก 8255 ไป

- INTR (INTERRUPT REQUEST) เป็นสัญญาณเอ้าพุทจากตัว 8255 แอคตีฟที่โลจิก 0 ปกติเราจะใช้ไปทริกให้กับ CPU ที่ขา INT เพื่อบอกให้ CPU ทราบว่าข้อมูลได้ถูกอ่านจาก 8255 ไปแล้ว ซึ่งเราสามารถ เซท/รีเซท ว่าต้องการให้เกิดสัญญาณที่ขา INTR นี้หรือไม่ก็ได้โดยซอฟร์แวร์



รูปที่ 13 แสดงถึงลำดับสัญญาณที่ 8255 ส่งข้อมูลออกมาเพื่อให้อุปกรณ์ภายนอกอ่านข้อมูลไป

<u>อธิบายได้ดังนี้</u> เมื่อ CPU ส่งสัญญาณการเขียนข้อมูลเข้ามาเก็บไปที่ ตัว 8255 (WR) จะเป็นผลทำให้ OBF แสดงภาวะว่ามีข้อมูลเข้ามาแลทช์ไว้ที่พอร์ท A (อาจเป็นพอร์ทอื่นตามที่เราโปรแกรมไว้) โดยจะให้โลจิกออกเป็น 0 และสัญญาณ INTR ก็จะเป็น 0 ด้วย (หากเราทำการเซทไว้ก่อน) ที่นี้สัญญาณก็จะห่างอยู่อย่างนี้ต่อไปเรื่อย ๆ หากไม่มีสัญญาณการอ่านข้อมูลจากอุปกรณ์ภายนอกมาอ่านสัญญาณนั้นคือ ACK

เมื่ออุปกรณ์ภายนอกต้องการอ่านเอาข้อมูลจากตัว 8255 ก็จะทำการตรวจดูสัญญาณที่ขา OBF ของ 8255 ก่อนว่าเป็น 0 หรือไม่เพราะหากได้ 0 เป็นการแสดงว่าข้อมูลมีอยู่ในพอร์ท พร้อมที่จะทำการอ่านได้ จึงส่งสัญญาณแอคตีฟโลจิกศูนย์มาที่ขา ACK ดังแสดงในรูปที่ 13 เป็นการอ่านเอาข้อมูลจาก 8255 ออกไปนั่นเอง จากนั้นสัญญาณ OBF, INTR ก็จะกลับคืนเป็นดลจิก 1 ดังเดิมซึ่งจะทำให้ CPU ทราบได้ว่ามีการอ่านเอาข้อมูลที่พอร์ทไปเรียบร้อยแล้ว CPU สามารถที่จะทำการส่งข้อมูลเข้ามาใหม่ได้

ลำดับสัญญาณในกรณีอินพุท โหมด 1

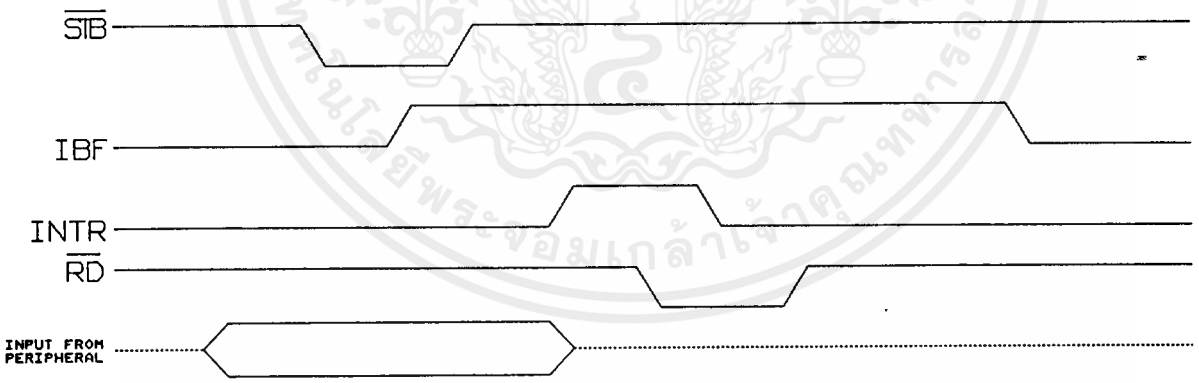
การอินพุทพอร์ท หมายถึง การที่อุปกรณ์ภายนอกส่งข้อมูลเข้ามาเก็บไว้ที่ตัว 8255 (แต่ต้องตรวจดูก่อนเช่นกันว่า 8255 ว่างหรือไม่) เพื่อให้ตัว CPU มาอ่านเอาข้อมูลนั้นไปสู่ตัว CPU มาอ่านเอาข้อมูลนั้นไปสู่ตัว CPU ต่อไป

รูปลำดับสัญญาณและความหมายที่ต้องใช้ในกรณีอินพุทมีดังนี้ ¦-

- IBF (INPUT BUFFER FULL) เป็นสัญญาณเอ้าพุท ที่แอคตีฟโลจิก 1 จะเป็นตัวแสดงว่าขณะนี้ข้อมูลในตัว 8255 เต็มอยู่หรือไม่ หมายความว่า ข้อมูลถูกอ่านด้วย CPU ไปหรือยัง หากยัง ที่ขาสัญญาณนี้ก็จะแสดงโลจิกเป็น 1 อยู่ แต่หากมีการอ่านเอาข้อมูลไปแล้วก็จะแสดงโลจิกเป็น 0 เพื่อให้อุปกรณ์ภายนอกสามารถส่งข้อมูลตัวต่อไปเข้ามาได้ทันที

- STB (STROBE INPUT) เป็นสัญญาณ อินพุท ที่แอคตีฟที่โลจิก 0 จะเป็นขาที่รับสัญญาณจากอุปกรณ์ภายนอก เพื่อแลทช์ข้อมูลที่ส่งมาจากภายนอกเข้าสู่ตัว 8255

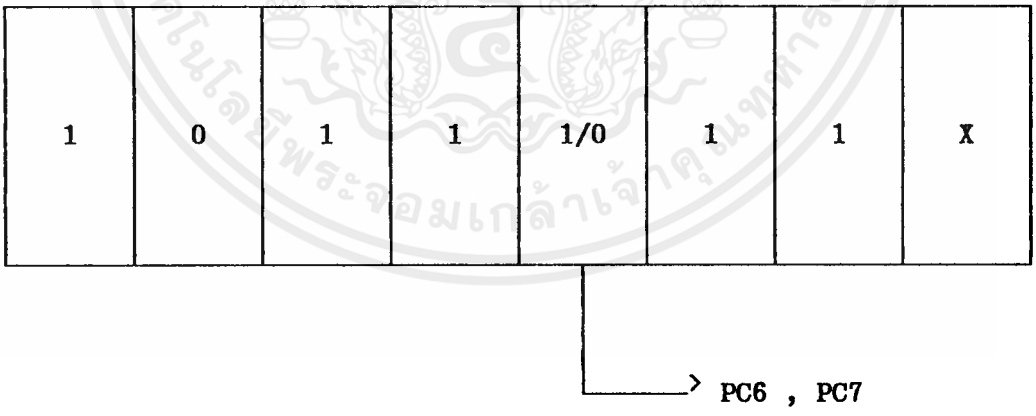
- INTR (INTERRUPT REQUEST) เป็นสัญญาณเอ้าพุท แอคตีฟที่โลจิก 1 มักใช้เป็นสัญญาณทริกอินเตอร์รัพท์ให้กับ CPU เพื่อบอกให้ทราบว่าอุปกรณ์ภายนอกส่งข้อมูลมาแล้ว

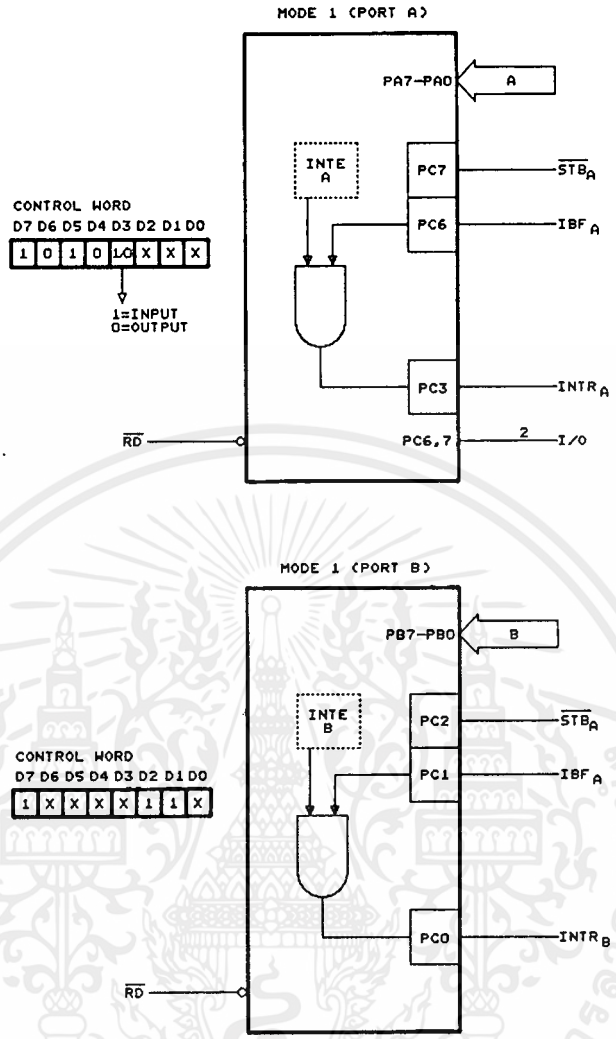


รูปที่ 14 แสดงลำสัญญาณที่เกิดขึ้นในกรณีการทำงานโหมด 1 อินพุท

อธิบายได้ดังนี้ เมื่ออุปกรณ์ภายนอกต้องการส่งข้อมูลเข้าสู่ตัว 8255 ก็ต้องตรวจดูก่อนว่าที่พอร์ทนั้นมีข้อมูลตกค้างอยู่หรือว่างหรือไม่โดยการตรวจที่ขาสัญญาณ IBF ว่าเป็นโลจิก 0 หรือไม่เพราะหากเป็นโลจิก 0 จะหมายถึงว่าง แต่หากเป็นโลจิก 1 จะหมายถึงว่าข้อมูลยังคงมีอยู่ เมื่อตรวจดูและทราบว่าเป็น 0 ก็สามารถส่งข้อมูลไปได้โดยส่งสัญญาณ พัลซโลจิก 0 ไปที่ขา STB เพื่อบอกให้ 8255 ได้ทำการแลทช์ข้อมูลที่ส่งให้นั้น และเมื่อแลทช์ข้อมูลได้แล้วก็เป็นผลทำให้ IBF มีโลจิกไปเป็น 1 เพื่อแสดงว่าได้แลทช์ข้อมูลเข้าสู่พอร์ทแล้ว พร้อมกันนั้นก็จะให้สัญญาณ INTR เป็นโลจิก 1 เพื่อแสดงให้ตัว CPU ทราบว่าได้แลทช์ข้อมูลจากอุปกรณ์ภายนอกได้แล้ว (หากเราต่อเป็นสัญญาณอินเตอร์รัพท์สู่ตัว CPU) และสัญญาณก็จะค้างลักษณะนี้ต่อไปจนกว่าจะมีสัญญาณการอ่านข้อมูลจาก CPU มาอ่านเอาข้อมูลจากพอร์ท 8255 ไปคือสัญญาณ RD และเมื่อ CPU ส่งสัญญาณ RD มาอ่านแล้วก็จะเป็นผลให้สัญญาณ IBF กลับเป็นโลจิก 1 และ INTR กลับเป็นโลจิก 0 ดังเดิม ซึ่งหากอุปกรณ์ภายนอกตรวจดูที่สัญญาณ IBF ก็จะทราบว่ามันสามารถส่งข้อมูลตัวต่อไปมาที่ 8255 ได้แล้ว ขอให้สังเกตสัญญาณในรูปที่ 14 ประกอบด้วย

ตัวอย่าง เมื่อเราโปรแกรมให้ พอร์ท A เป็น อินพุทและพอร์ท B เป็นอินพุท ก็สามารถแสดงถึงรูปของขาสัญญาณที่ใช้งานและคำสั่งควบคุมส่งไปให้ 8255 ดังรูปที่ 15 นี้





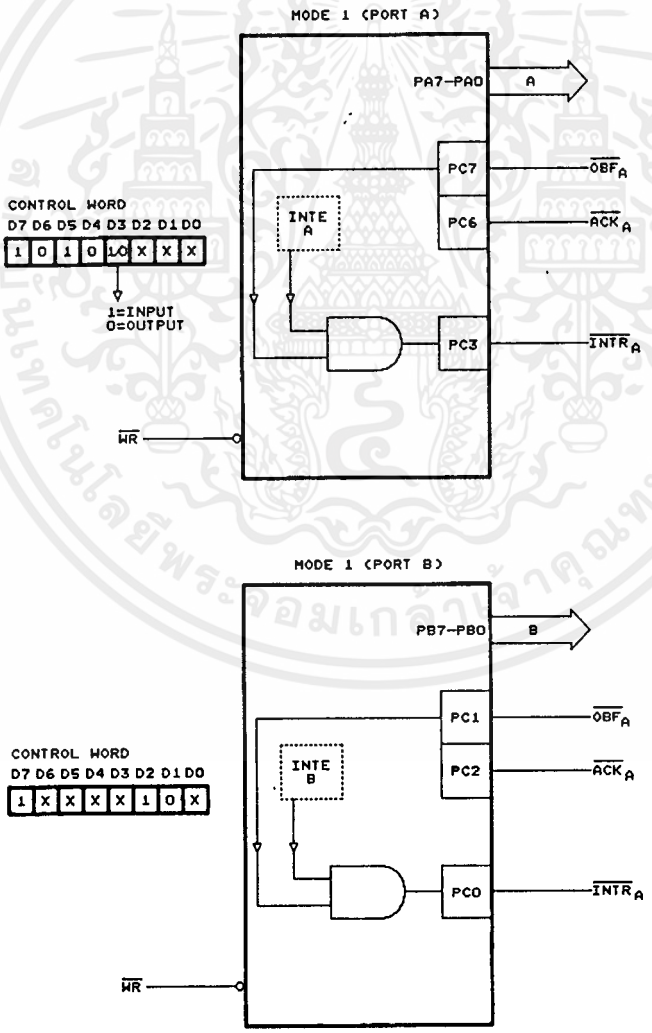
รูปที่ 28 แสดงถึงขาสัญญาณเมื่อ 8255 ถูกโปรแกรมให้พอร์ท A เป็นอินพุทและพอร์ท B เป็นอินพุท

ปกติสัญญาณ INTR มักจะไม่ใช้เพราะเราสามารถอ่านข้อมูลจากพอร์ท C ของ 8255 มาตรวจดูสถานะการทำงาน (IBF,OBF) ได้โดยการตรวจเช็คบิทเหล่านั้น แต่หากเราต้องการใช้สัญญาณ INTR นี้เราก็สามารถทำได้ โดยเราสามารถจะเซทหรือรีเซทบิทเพื่อจะกำหนดให้เกิดสัญญาณ INTR หรือไม่ก็ได้ไม่ว่ากรณีของอินพุท ซึ่งวิธีการก็คือการส่งคำสั่งควบคุมที่บิท 7 = 0 ไปที่รีจิสเตอร์ควบคุม ซึ่งเหมือนกับการใช้คำสั่งเซทบิทพอร์ท C ดังที่ได้กล่าวมาแล้วในตอนต้น

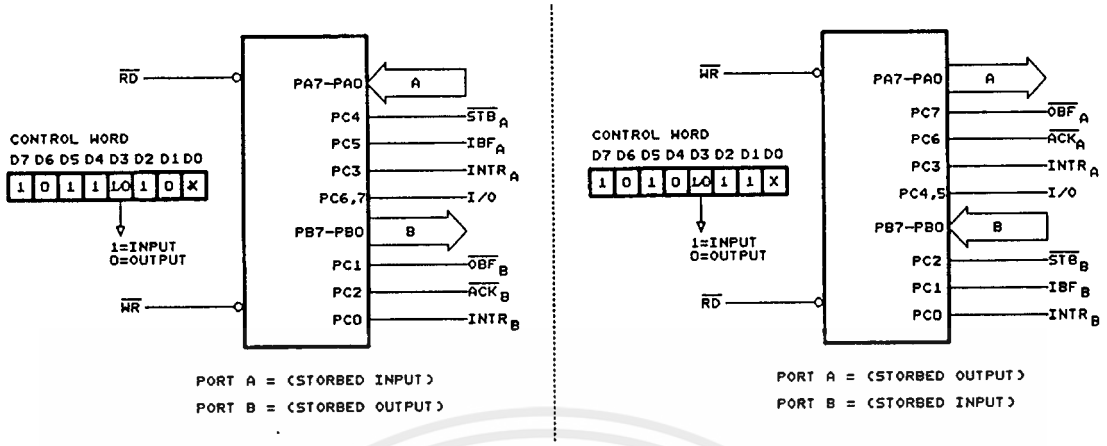
โดยจะขอสรุปว่าหากต้องการให้มีการส่งสัญญาณ INTR เกิดขึ้นก็ส่งค่า 1 ไปที่บิทนั้น หากไม่ต้องการให้เกิดสัญญาณ INTR ก็ส่ง 0 ไปที่บิทนั้น ทั้งนี้เพราะจากรูปที่ 15 นั้นเอ้าพุทที่ได้สัญญาณ INTR คือผลจากการ AND กันของสัญญาณ INTE (เป็นเสมือนตัวกำหนดสัญญาณ อินเตอร์รัฟท์) กับ IBF (กรณีอินพุท) หรือ OBF (กรณีเอ้าพุท) และเราสามารถจะเปิดปิดเกทเพื่อให้สัญญาณอินเตอร์รัฟท์ INTE ผ่าน

ได้หรือไม่ โดยกำหนดเซทหรือรีเซทที่ขาพอร์ท C ของสัญญาณ ACK,STB แล้วแต่กรณีว่าเป็นอินพุทหรือเอ้าพุท ดังจะสรุปดังนี้

PORT	กำหนดให้พอร์ทเป็น	สัญญาณ INT	ให้ทำการ SET/RESET ที่
A	INPUT	INTE A	PC4 (STBA)
A	OUTPUT	INTE A	PC6 (ACKA)
B	INPUT	INTE B	PC2 (STBB)
B	OUTPUT	INTE B	PC2 (ACKA)

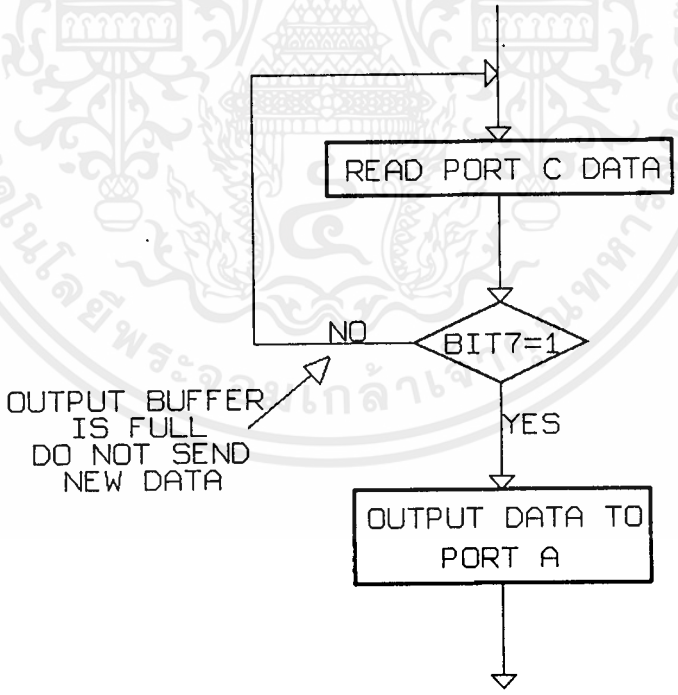


รูปที่ 16 แสดงถึง 8255 ในโหมด 1 ที่โปรแกรมพอร์ท A และ พอร์ท B เป็นเอ้าพุท

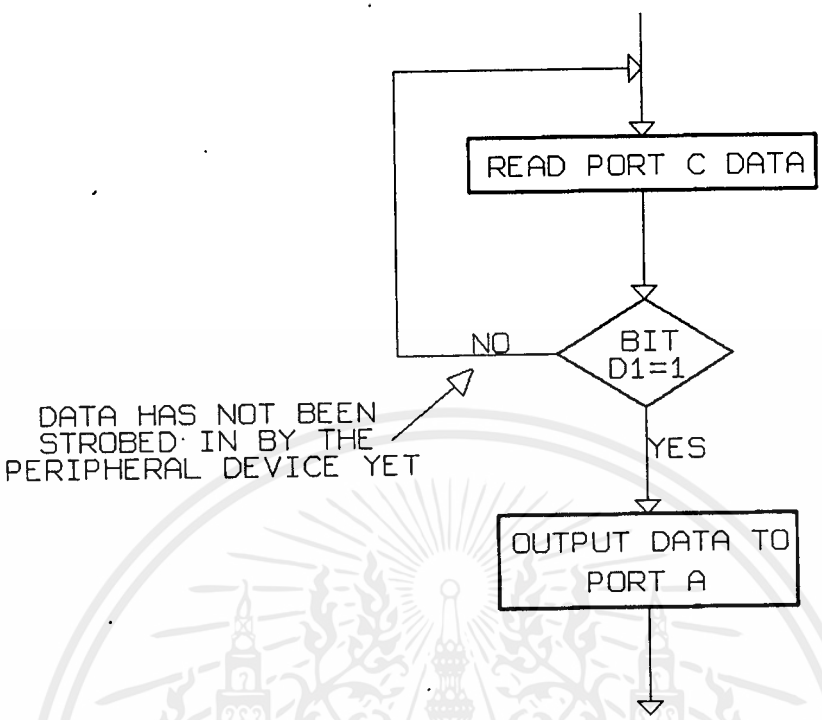


รูปที่ 17 แสดงถึงลักษณะของพอร์ท A,B ที่ถูกโปรแกรมผสมกัน ในโหมด 1

ในการโปรแกรมเพื่อส่งข้อมูลออกไปสู่ 8255 (เอ้าพุท) และรับข้อมูลจาก 8255 เข้าสู่ CPU สามารถทำได้โดยง่ายดังแสดงโฟลชาร์ทข้างล่างนี้



รูปที่ 18 แสดงโฟลชาร์ทของการส่งข้อมูลออกไปที่พอร์ท A



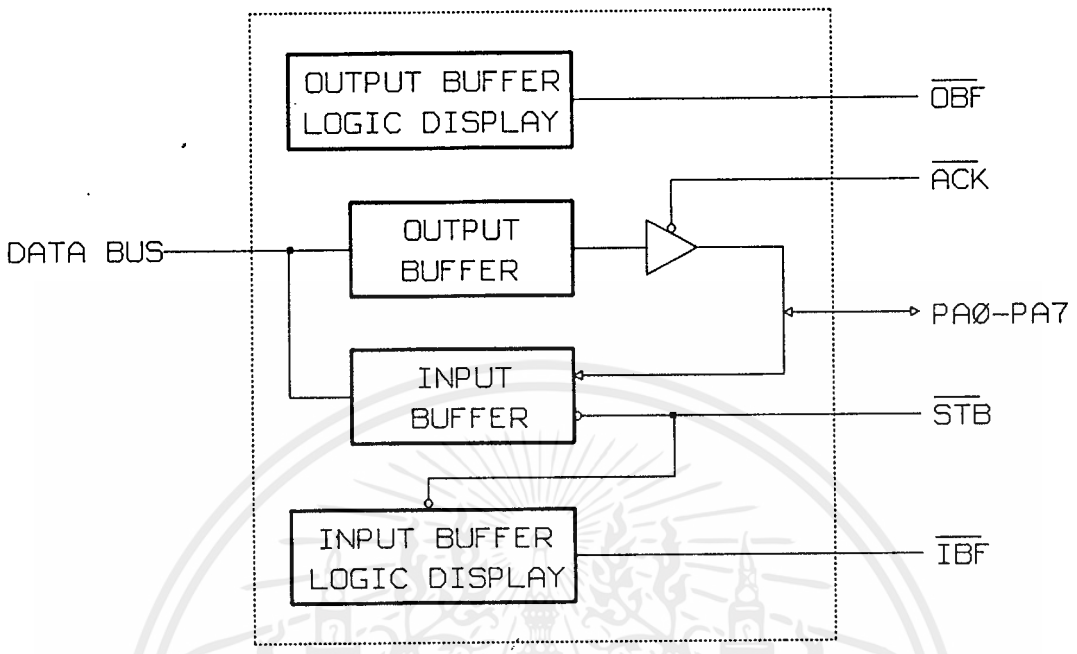
รูปที่ 19 แสดงโฟลชาร์ทและโปรแกรมของการรับข้อมูลเข้าจากพอร์ท

<u>การทำงานในโหมด 2</u>

การทำงานในโหมดที่ 2 นี้จะสามารถทำได้เฉพาะพอร์ท A เท่านั้น เนื่องจากว่าการทำงานในโหมดนี้ คือการที่กำหนดให้พอร์ท A เป็นได้ทั้งอินพุทและเอ้าพุทได้ในพอร์ทเดียว ซึ่งทำให้ต้องให้สายสัญญาณควบคุมมากขึ้นเป็น 5 เส้น ซึ่งก็ใช้พอร์ท C เป็นขาสัญญาณควบคุมนี้ ทำให้ไม่เพียงพอแก่พอร์ท B ที่จะให้เป็นโหมด 2 ฉะนั้นพอร์ท B จึงสามารถทำงานได้เฉพาะโหมด 0,1 เท่านั้น

การทำงานในโหมดนี้คือ การใช้พอร์ท A เป็นทั้งอินพุทแลทช์ข้อมูลและเอ้าพุทแลทช์ข้อมูล โดยเอ้าพุทแลทช์ก็จะหมายถึง การที่พอร์ท A รับข้อมูลจาก CPU มาทำการแลทช์ไว้เพื่อรอการอ่านข้อมูลนี้ไปด้วยอุปกรณ์ภายนอก ส่วนกรณีของอินพุทแลทช์ก็หมายถึงการเก็บข้อมูลที่อุปกรณ์ภายนอกส่งมาแลทช์ไว้เพื่อรอให้ CPU ทำการอ่านข้อมูลนี้ไปนั่นเอง

การทำงานโดยทั่วไปเหมือนกันกับการทำงานในโหมด 1 ที่ได้กล่าวมา เพียงแต่เป็นการรวมเอาพอร์ทการรับส่งไว้เป็นช่องเดียว นั่นคือเมื่อกระทำการเอ้าพุทก็จะมีสัญญาณ OBF,ACK,INTR ใช้ติดต่อควบคุม และเมื่อกระทำการอินพุทก็จะมีสัญญาณ IBF,STB,INTR ใช้ในการติดต่อควบคุมการทำงาน โดยจะขอกล่าวเป็นลำดับดังนี้

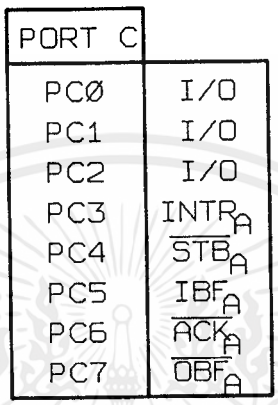


รูปที่ 20 แสดงถึงบล็อกไดอะแกรมการทำงานในโหมด 2 ของพอร์ท A

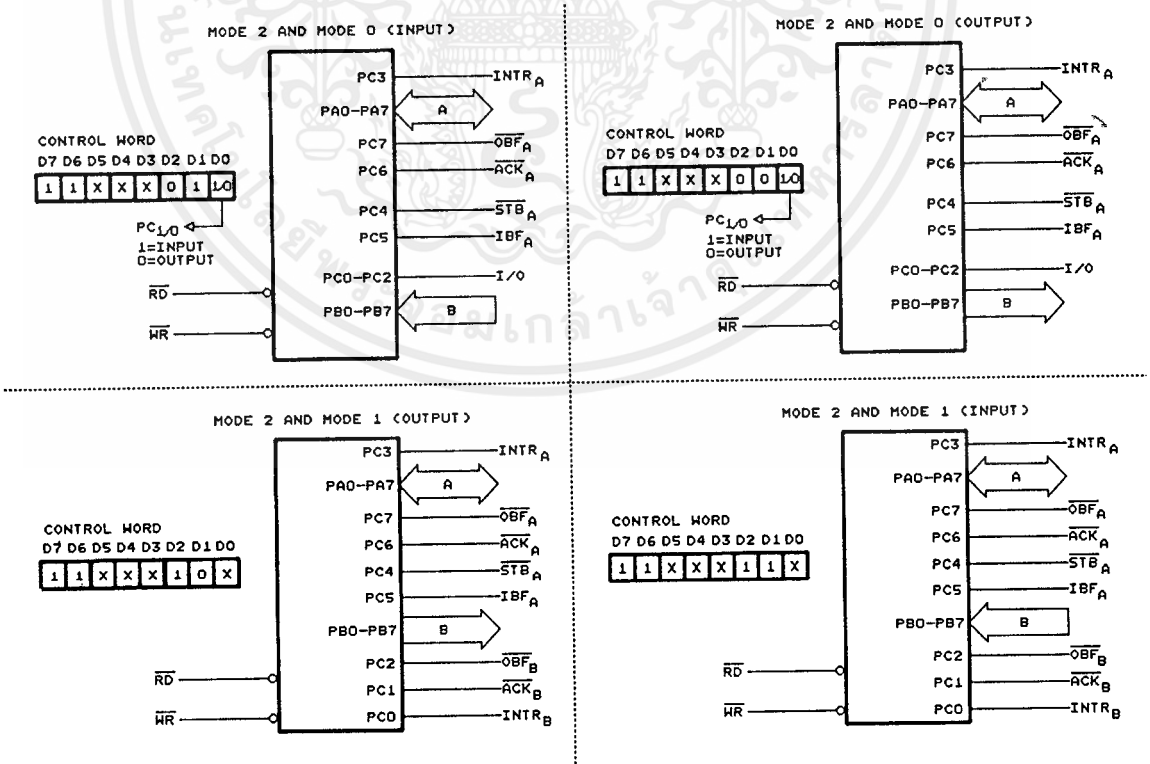
การส่งข้อมูลจาก CPU ไปสู่ 8255 เพื่อออกสู่ภายนอกนั้นขั้นแรกก็ต้องตรวจดูก่อนว่า พอร์ท A ว่างหรือไม่ โดยการอ่านค่า บิท PC7 (OBF) มาดูว่าเป็น 1 หรือไม่ หากเป็น (ว่าง) ก็สามารถส่งข้อมูลออกไปแลทช์ไว้ที่ 8255 ได้ และเมื่ออุปกรณ์ภายนอกต้องการรับข้อมูลไปก็จะตรวจที่ PC7 (OBF) นี้ว่าได้โลจิกเป็น 0 หรือไม่ หากเป็น 0 (หมายถึงมีข้อมูลอยู่) ก็จะทำการอ่านออกไป โดยส่งสัญญาณ ACK มาที่ บิท PC6 (ACK) เพื่ออ่านเอาข้อมูลไปเป็นผลทำให้สถานะของ PC7 (OBF) กลับคืนเป็น 1 อีกครั้ง เพื่อแสดงตัวว่า 8255 ว่างที่จะรับข้อมูลตัวต่อไปจาก CPU แล้ว

การรับข้อมูลจาก อุปกรณ์ภายนอกนั้น ก่อนที่จะทำการส่งข้อมูลจากอุปกรณ์ภายนอกมาที่พอร์ท A นั้นอุปกรณ์ภายนอกจะทำการตรวจดูก่อนว่าพอร์ท A ว่างหรือไม่ โดยการตรวจที่ บิท PC5 (IBF) ว่าเป็นโลจิก 0 หรือไม่ หากเป็น 0 แสดงว่าว่างก็จะส่งสัญญาณ STB มาที่ขา PC4 เป็นการบอกให้ 8255 ได้ทำการแลทช์ข้อมูลของอุปกรณ์ภายนอกที่ส่งมาไว้ที่พอร์ท A เมื่อแลทช์แล้ว PC4 (IBF) จะเปลี่ยนโลจิกไปเป็น 1 ทันที เพื่อบอกให้อุปกรณ์ภายนอกทราบว่า ข้อมูลถูกแลทช์เข้าสู่พอร์ท A เรียบร้อยแล้ว <u>อย่าเพิ่งส่งข้อมูลมาอีก</u> ถึงตอนนี้เมื่อ CPU (IBF) ดูโดยการอ่านพอร์ท C ก็จะทราบว่ามีข้อมูลส่งมาแล้ว จึงทำการอ่านข้อมูลไปได้ (ส่ง RD มาอ่าน) เป็นผลทำให้ PC4 (IBF) กลับโลจิกเป็น 0 อีกครั้ง เพื่อให้อุปกรณ์ภายนอกส่งข้อมูลตัวต่อไปมาที่ 8255 อีก

ขาสัญญาณควบคุมของพอร์ท C ที่ใช้ในการทำสัญญาณควบคุมต่างๆ นั้น แสดงดังรูปที่ 20 สังเกตุเห็นได้ว่า PC0-PC2 นั้นเราสามารถกำหนดให้เป็น อินพุทหรือเอ้าพุทได้อีกต่างหาก ในกรณีที่ พอร์ท B ถูกโปรแกรมในโหมด 0 (เพราะไม่ต้องมีสัญญาณควบคุม) แต่หากพอร์ท B ถูกโปกแกรมในโหมด 1 จะทำให้ต้องใช้ PC0-PC2 นี้เป็นสัญญาณควบคุมของ พอร์ท B



รูปที่ 20 แสดงถึงพอร์ท C ที่ถูกใช้เป็นสัญญาณควบคุมในโหมด 2



รูปที่ 22 แสดงการโปรแกรมโหมด 2 ร่วมกับโหมด 0,1

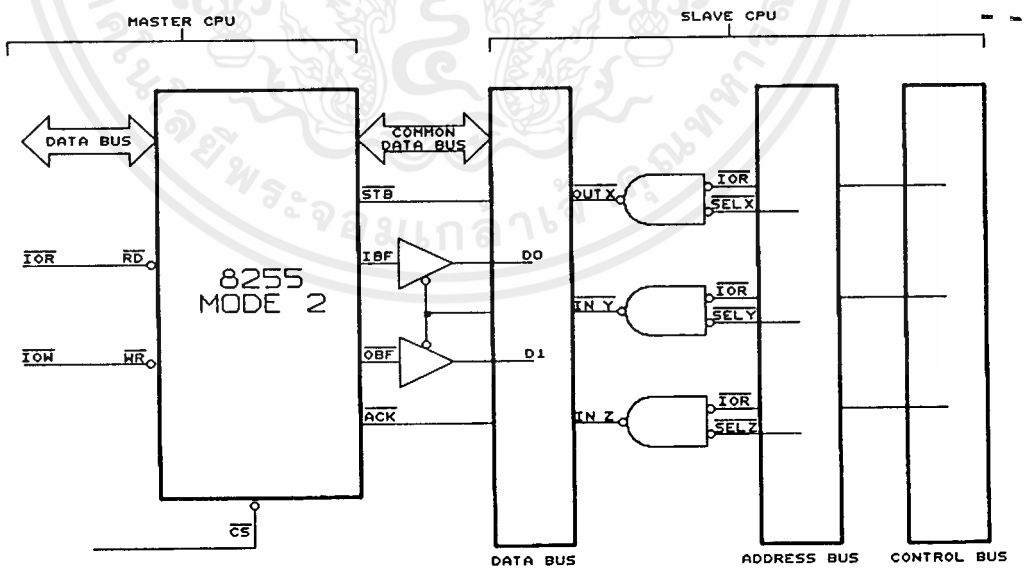
ส่วนสัญญาณ INTR นั้นเราสามารถกำหนดให้มีการเกิดอินเตอร์รัพท์หรือไม่ก็ได้ โดยการเซท/รีเซทบิทดังนี้ กรณี อินพุท สามารถทำการ เซท/รีเซท ได้ที่ บิท PC4 (STB)

กรณี เอ้าพุท สามารถทำการ เซท/รีเซท ได้ที่ บิท PC6 (ACK)

เราสามารถโปรแกรมให้พอร์ท A ในโหมด 2 ทำงานร่วมกับพอร์ท B ซึ่งเป็นโหมด 0,1 ก็ได้ดังนี้

ตัวอย่างการใช้งานในโหมด 2

สมมุติเรามี ไมโครโปรเซสเซอร์บอร์ดอยู่ 2 ชุด โดยในบอร์ดที่หนึ่งมี 8255 อยู่ ซึ่งขอเรียกบอร์ดนี้ว่า มาสเตอร์บอร์ด (Master board) ส่วนในบอร์ดที่ 2 เป็นบอร์ดที่อาจไม่มี 8255 อยู่ก็ได้ ซึ่งจะขอเรียกว่า สเลฟบอร์ด (Slave board) หากเราต้องการส่งข้อมูลไปมาระหว่าง 2 บอร์ดนี้ โดยอาจส่งข้อมูลจาก มาสเตอร์บอร์ดไปสู่สเลฟบอร์ดหรือในทางกลับกัน เราจะส่งข้อมูลจากสเลฟบอร์ดไปสู่มาสเตอร์บอร์ด ก็สามารถเขียนโปรแกรมการควบคุมได้โดยไม่ยาก โดยบอร์ดทั้งสองมีการต่อถึงกัน ดังแสดงในรูปตัวอย่างรูปที่ 23 นี้



รูปที่ 23 แสดงตัวอย่างการส่งสายสัญญาณเพื่อส่งข้อมูลไปมาระหว่างไมโครโปรเซสเซอร์

จากรูป เมื่อเราโปรแกรมให้ 8255 ทำงานในโหมด 2 แล้ว พอร์ท A ก็จะเป็นพอร์ทส่งรับข้อมูล 2 ทิศทาง ซึ่งถูกต่อโดยตรงกับสาย DATA BUS ของสเลฟบอร์ด และ IBF,OBF ถูกต่อเข้าสู่บิท D0,D1 ของสเลฟบอร์ด โดยสามารถอ่านค่าได้เมื่อ สเลฟ อินพุทพอร์ทมาที่ INY

มาสเตอร์ส่งข้อมูลให้สเลฟ มาสเตอร์บอร์ดจะตรวจดูที่ พอร์ท C ขาสัญญาณ OBF ก่อนว่าเป็น 1 หรือไม่ หากเป็น 1 แสดงว่าพอร์ท A ว่าง ก็จะส่งข้อมูลมาสู่พอร์ท A เป็นผลทำให้ OBF เป็นโลจิก 0 ซึ่งทางสเลฟบอร์ดก็สามารถทราบได้โดยการอินพุทพอร์ท INY เข้ามาดูที่บิท D1 (OBF) เมื่อทราบว่าเป็น 0 ก็จะส่งสัญญาณเข้าพอร์ท INZ เกิดสัญญาณ ACK ไปรับเอาข้อมูลมาเก็บที่ สเลฟได้ และ OBF ก็จะกลับเป็น 1 ใหม่ เพื่อรอให้มาสเตอร์ส่งข้อมูลมาที่ 8255 อีก

สเลฟส่งข้อมูลให้มาสเตอร์ สเลฟบอร์ดก็จะอ่านข้อมูลอินพุทพอร์ท INY เพื่อตรวจดูว่า IBF เป็น 0 หรือไม่ (ว่างหรือไม่) หากเป็น 0 ก็สามารถส่งข้อมูลออกไป และจะสั่งเข้าพอร์ท INX ออกไปด้วย เพื่อให้เป็นสัญญาณ STB ให้ 8255 ทำการแลทช์ข้อมูลนั้นไว้ในพอร์ท A เป็นผลทำให้ IBF เป็น 1 ซึ่งหากมาสเตอร์ตรวจที่สัญญาณ IBF นี้ดูก็จะทราบว่ามีข้อมูลจากสเลฟมา ก็จะทำการสัญญาณการอ่าน RD มาอ่านเอาข้อมูลไปเก็บไว้ ทำให้ IBF เป็น 0 อีกครั้ง เพื่อรอการรับข้อมูลจากสเลฟเข้ามาใหม่ ซึ่งสเลฟบอร์ดสามารถส่งข้อมูลตัวต่อๆ ไปเข้ามาได้

จากตัวอย่างนี้เราสามารถนำไปประยุกต์ทำเป็น PRINT BLFFER ให้หรือใช้กับการส่งข้อมูลมาตรฐานบางประการ เช่น IEEE 488

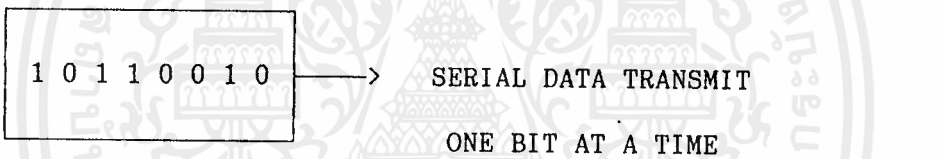
บทที่ 3

8251

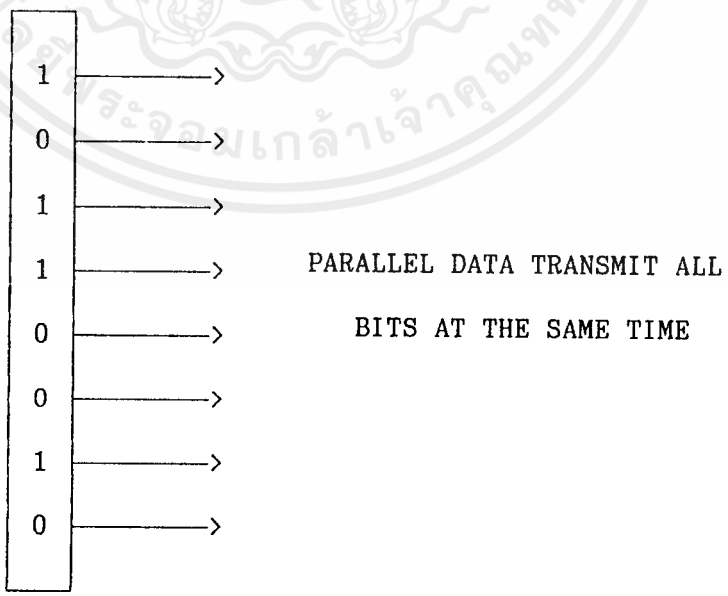
รับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลที่ได้กล่าวถึงในตอนต้นนั้น ข้อมูลทุกๆ บิทจะถูกรับหรือส่งออกไปในเวลาเดียวกัน เช่นการอ่านหรือเขียนข้อมูลลงในหน่วยความจำ ซึ่งเราเรียกการรับการส่งข้อมูลในลักษณะนี้ว่า "การรับส่งข้อมูลแบบขนาน (PARALLEL COMMUNICATION)"

สำหรับในบทนี้จะกล่าวถึงการรับส่งข้อมูลในอีกรูปแบบหนึ่ง ซึ่งเป็นการรับส่งข้อมูลทีละบิทแทนที่จะทำการส่งข้อมูลพร้อมกันทุกบิทในเวลาเดียวกัน การรับส่งข้อมูลแบบนี้มีชื่อว่า "การรับส่งข้อมูลแบบอนุกรม (SERIAL COMMUNICATION)"



รูปแสดงบิทต่างๆ ของข้อมูลที่จะทำการส่งแบบอนุกรม โดยที่ข้อมูลนี้จะถูกส่งทีละ 1 บิท



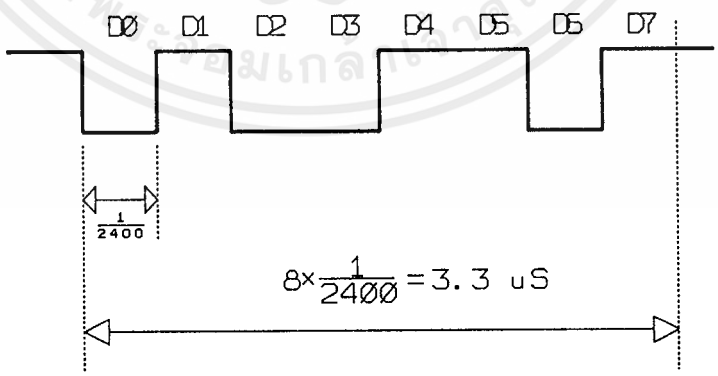
รูปแสดงบิทต่างๆ ของข้อมูลที่จะทำการส่งแบบขนาน โดยที่ทุกบิทของข้อมูลจะถูกส่งออกไปในเวลาเดียวกัน

สำหรับการรับส่งข้อมูลแบบขนานนั้น ถึงแม้ว่าจะมีความเร็วสูงกว่าแบบอนุกรมอยู่มากก็ตาม แต่ก็ต้องใช้จำนวนสายในการส่งผ่านข้อมูลเป็นจำนวนมากกว่าแบบอนุกรม ทำให้สิ้นเปลืองค่าใช้จ่ายในการวางสายไปโดยไม่จำเป็น และยังมีการลดทอนของสัญญาณมากกว่าแบบอนุกรมอีกด้วย ทำให้เกิดความผิดพลาดในการส่งผ่านข้อมูลขึ้นได้ง่าย ดังนั้นในการส่งผ่านข้อมูลในระยะทางไกลๆ เรามักจะเลือกใช้การรับส่งข้อมูลแบบอนุกรม เพื่อลดจำนวนของสายส่งซึ่งจะช่วยในการลดค่าใช้จ่ายในการวางสายลงได้อย่างมาก ถึงแม้ว่าการรับส่งข้อมูลแบบนี้จะมีความยุ่งยากและช้ากว่าการรับส่งข้อมูลแบบขนานอยู่บ้างก็ตาม

BAUD RATE

สิ่งสำคัญมากสิ่งหนึ่งในการรับส่งข้อมูลแบบอนุกรมนี้ก็คือ ความถี่ที่ใช้ในการส่งข้อมูลซึ่งจะต้องสัมพันธ์กันระหว่างอุปกรณ์ที่ทำการรับและส่งข้อมูล และความถี่ที่ใช้นี้มีชื่อเรียกว่า "BAUD RATE" ซึ่งมีความหมายถึง "อัตราการรับส่งข้อมูลเป็นจำนวนบิทใน 1 วินาที" ถ้าหากว่าเครื่องส่งใช้ BAUD RATE ที่ไม่สัมพันธ์กับเครื่องรับแล้ว ก็จะทำให้การรับส่งข้อมูลเกิดผิดพลาดขึ้นได้

โดยทั่วไปค่าของ BAUD RATE นั้นจะใช้ค่าต่างๆ ดังต่อไปนี้ คือ 110,150,300,1200, 2400,4800 และ 9600 สำหรับในบทนี้จะสมมติว่า เราต้องการที่จะส่งข้อมูลแบบอนุกรมด้วยอัตรา 2400 BAUD (2400 บิท/วินาที) และข้อมูลที่ต้องการจะส่งก็คือ OB2H หรือ 10110010B ซึ่งเราสามารถที่จะแสดงได้ในรูปของสัญญาณดังรูปที่ 1



รูปที่ 1 : แสดงรูปสัญญาณของข้อมูลที่ถูกส่งไปตามสายส่งแบบอนุกรม

จากรูปที่ 1 จะเห็นว่าความกว้างของสัญญาณของแต่ละบิทจะเท่ากับ 1/BAUD RATE วินาที ซึ่งจาก BAUD RATE ที่เราต้องการที่จะใช้คือ 2400 BAUD นั้นจะทำให้ความกว้างของแต่ละบิทมีค่าเท่ากับ 1/2400 วินาที หรือ เท่ากับ 416 microseconds ซึ่งจากความกว้างของแต่ละบิทที่จะส่งไปตามสายส่งนี้ ทำให้เราสามารถที่จะคำนวณเวลาที่จะต้องใช้ในการรับส่งข้อมูลแต่ละไบท์ (8 บิท) ได้ดังนี้คือ เท่ากับ 8 416 microseconds หรือ 3328 microseconds อย่างไรก็ตาม เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้ จึงมีการเพิ่มบิทต่างๆ ลงไปในแต่ละไบท์ของข้อมูล เพื่อช่วยในการตรวจสอบความถูกต้องของข้อมูลที่เครื่องรับได้รับเข้ามา (แต่ไม่ได้หมายความว่าเมื่อเพิ่มบิทต่างๆ เหล่านี้เข้าไปแล้วจะทำให้การส่งผ่านข้อมูลมีความถูกต้อง 100%) สำหรับบิทต่างๆ ที่เพิ่มเข้ามานี้ ก็คือ START, STOP และ PARITY BIT ซึ่งจะทำให้ข้อมูลในแต่ละไบท์ที่ส่งออกไปนี้มีมากกว่า 8 บิท และเวลาที่ใช้ในการรับส่งข้อมูลก็จะมากขึ้นตามไปด้วย

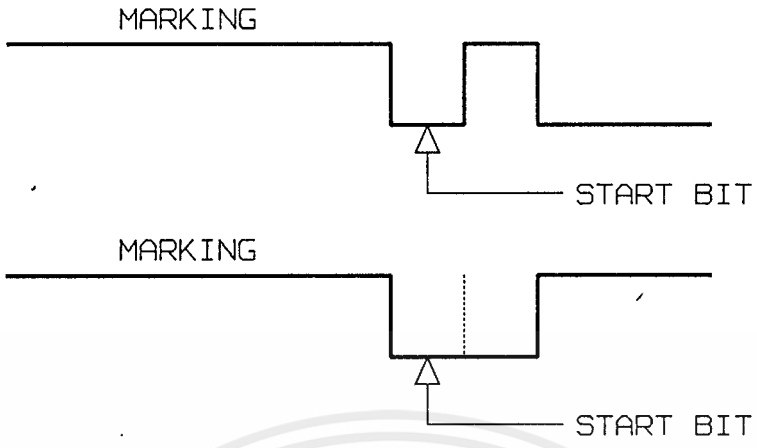
START BIT

ในการส่งผ่านข้อมูลแบบอนุกรมนี้ เราจำเป็นที่จะต้องทำให้อุปกรณ์ที่จะรับข้อมูลทราบว่า ข้อมูลที่ส่งมานั้นเริ่มต้นที่จุดใด ดังนั้นเราจึงจำเป็นที่จะต้องเพิ่มข้อมูล 1 บิทลงไปก่อนหน้าข้อมูลจริง (ACTUAL DATA) ที่จะทำการส่ง (การส่งอนุกรมจะส่งบิท D0 เป็นบิทแรก และ D7 เป็นบิทสุดท้าย) คือทำการเพิ่มบิทนี้ลงไปหน้าบิท D0 นั่นเอง และเรียกบิทนี้ว่า "START BIT"

หน้าที่ของ START BIT นี้นอกจากจะใช้ในการบอกว่าข้อมูลนั้นเริ่มต้นที่ใดแล้ว ยังทำงานร่วมกับ STOP BIT (ซึ่งจะกล่าวถึงต่อไป) เพื่อช่วยในการแยกข้อมูลแต่ละชุดออกจากกัน และความกว้างของบิทนี้จะเท่ากับความกว้างของบิทอื่นๆ ในข้อมูลที่จะส่ง (D0-D7)

เมื่ออุปกรณ์ที่จะส่งข้อมูลยังไม่ได้ทำการส่งข้อมูลใดๆ ออกมานั้น สายส่งจะอยู่ในสภาวะที่ไม่มีการรับส่งข้อมูลใดๆ เกิดขึ้นในที่นี้เราจะสมมติให้ MARKING ของสายส่งเป็นลอจิก "1" START BIT ที่จะเพิ่มเข้าไปนี้จะมีลอจิกที่ตรงข้ามกับลอจิกของ MARKING ดังนั้นในกรณีนี้ START BIT จะมีลอจิกเป็น "0"

สำหรับ START BIT นี้จะมีความกว้างเท่ากับ 1 บิทของข้อมูล เช่น ใน 1 บิทของข้อมูลมีความยาวเท่ากับ 416 microseconds START BIT ก็จะมีความกว้างของสัญญาณเท่ากับ 416 microseconds ด้วย ในรูปที่ 2 จะแสดงให้เห็นถึง START BIT ที่เพิ่มเข้าไปก่อนหน้าข้อมูล (ก่อนหน้า D0)



รูปที่ 2 การเพิ่ม START BIT เข้าไปก่อนหน้าบิท D0 ในกรณีที่บิท D0 เป็น "1" และ "0" ตามลำดับ

PARITY BIT

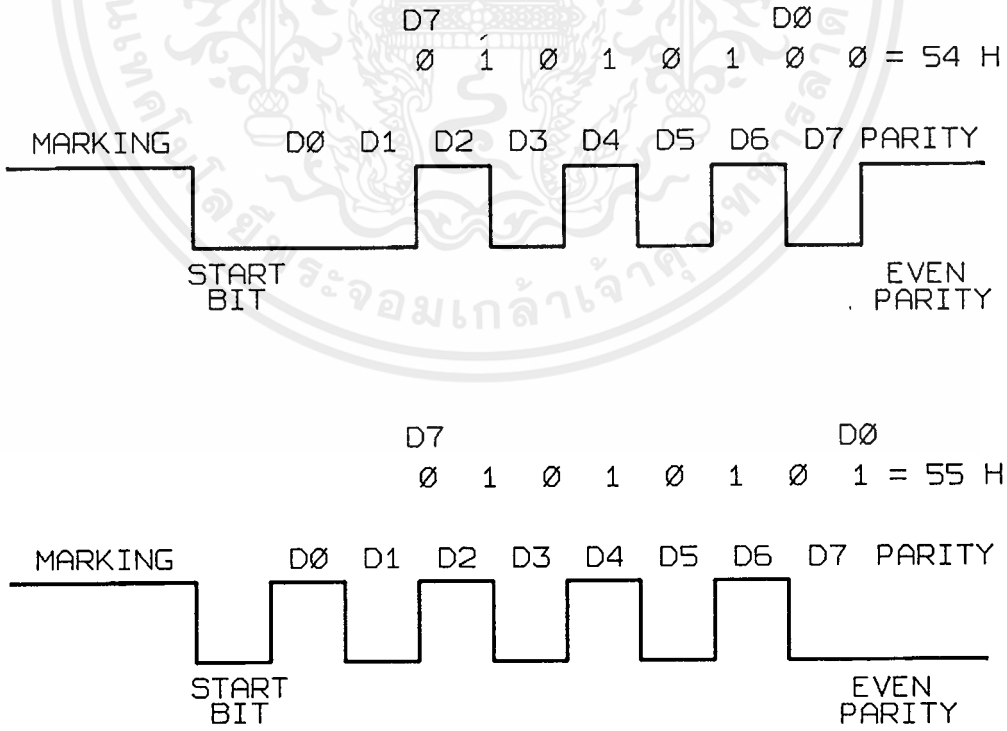
สำหรับบิทนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่า ข้อมูลที่ได้รับเข้ามานั้นมีความถูกต้องเหมือนกับข้อมูลที่ถูกส่งออกมาหรือไม่ (ถึงแม้ว่าการตรวจสอบบิทนี้จะไม่พบความผิดพลาด แต่ก็ไม่ได้หมายความว่าข้อมูลที่รับเข้ามานี้จะมีความถูกต้อง 100%) โดยที่บิทนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่าข้อมูลที่ส่งออกมาแต่ละไบท์นั้นมีจำนวนบิทที่เป็น "1" อยู่เป็นจำนวนคี่ หรือ จำนวนคู่ เช่น ข้อมูล 54H หรือ 01010111B จะมีจำนวนบิทที่เป็น "1" อยู่เป็นจำนวนคี่เป็นต้น สำหรับบิทที่ใช้ในการตรวจสอบนี้เรียกว่า "PARIRY BIT"

PARITY BIT นี้จะถูกส่งออกมาโดยอุปกรณ์ส่งข้อมูล ซึ่งบิทนี้จะเป็น "1" หรือ "0" นั้นขึ้นอยู่กับข้อมูลที่ส่งออกมา (D0-D7) ว่ามีจำนวนบิทที่เป็น "1" เป็นจำนวนคู่หรือคี่ และยังขึ้นกับอุปกรณ์รับส่งข้อมูลด้วยว่าถูกออกแบบ (โปรแกรม) ไว้ให้รับส่ง PARITY BIT ในลักษณะของ PARITY คู่ หรือ PARITY คี่ อีกด้วย

ในกรณีที่อุปกรณ์รับส่งข้อมูลถูกออกแบบไว้ให้เป็น PARITY คู่ อุปกรณ์ส่งข้อมูลจะทำการส่ง PARITY BIT เป็นลอจิก "1" ออกไปเมื่อจำนวนบิทที่เป็น "1" ของข้อมูล (D0-D7) เป็นจำนวนคี่ และจะทำการส่ง PARITY BIT เป็นลอจิก "0" เมื่อจำนวนบิทที่เป็น "1" ของข้อมูลเป็นจำนวนคู่ (คือ ทำให้จำนวนบิทที่เป็น "1" ของข้อมูล (D0-D7) รวมกับ PARITY BIT แล้วเป็นจำนวนคู่นั่นเอง) สำหรับ PARITY คี่ก็เช่นกัน คือ PARITY BIT จะเป็น "1" ในกรณีที่จำนวนบิทที่เป็น "1" ของข้อมูลเป็นจำนวนคู่และจะเป็น "0" ในกรณีที่เป็นจำนวนคี่ ในที่นี้จะสมมติว่าอุปกรณ์ถูกออกแบบไว้สำหรับ PARITY

คู่ และเราต้องการที่จะส่งข้อมูลออกไปให้กับส่วนรับข้อมูลเป็นจำนวน 2 ไบท์ คือ 54H และ 55H เมื่อเราส่งข้อมูล 54H ออกไปซึ่งมีจำนวนบิทที่เป็น "1" เป็นจำนวนคี่ ดังนั้นในกรณีนี้อุปกรณ์ส่งข้อมูลก็จะทำการส่ง PARITY BIT เป็นลอจิก "1" ตามออกมาด้วย เพื่อจะให้จำนวนบิทที่เป็น "1" ของข้อมูล (54H) รวมกับ PARITY BIT แล้วได้เป็นจำนวนคู่ ส่วนข้อมูล 55H นั้นจำนวนบิทที่เป็น "1" นั้นเป็นจำนวนคู่อยู่แล้ว ดังนั้นอุปกรณ์ส่งข้อมูลก็จะส่ง PARITY BIT เป็น "0" ให้กับส่วนรับข้อมูล ดังในรูปที่ 3 สำหรับส่วนรับข้อมูลนั้น เมื่อทำการรับข้อมูลเข้ามาแล้ว ก็จะตรวจสอบสัญญาณว่าจำนวนบิทที่เป็น "1" ของข้อมูลรวบกับ PARITY BIT นั้นเป็นจำนวนคู่หรือไม่ ถ้าหากว่าเป็นจำนวนคี่ก็แสดงว่าข้อมูลที่ได้รับเข้ามานี้มีความผิดพลาดเกิดขึ้น (แต่ไม่ได้หมายความว่า ถ้าเป็นจำนวนคู่แล้ว ข้อมูลที่รับเข้ามาจะถูกต้องเสมอไป)

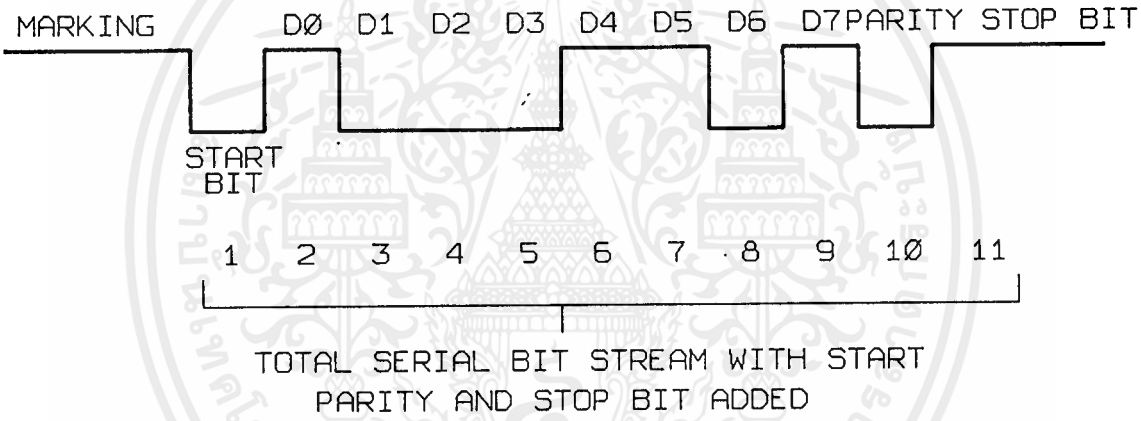
สิ่งสำคัญอีกสิ่งหนึ่งก็คือ ถ้าอุปกรณ์ส่งข้อมูลทำการส่งในลักษณะ PARITY คู่หรือคี่ก็ตาม ส่วนรับข้อมูลก็จะต้องทำการรับในลักษณะ PARITY เดียวกับอุปกรณ์ส่งข้อมูลด้วย เช่นในกรณีที่อุปกรณ์ส่งข้อมูลทำการส่งข้อมูลในลักษณะของ PARITY คู่อุปกรณ์ข้อมูลก็จะต้องทำการรับข้อมูลในลักษณะของ PARITY คู่ด้วย เป็นต้น



รูปที่ 3 การเพิ่ม PARITY BIT ลงไปในข้อมูลแต่ละไบท์

STOP BIT

สำหรับบิทสุดท้ายที่เพิ่มเข้าไปนี้ จะใช้ในการตรวจสอบจุดสิ้นสุดของข้อมูลบิทนี้ จะถูกเพิ่มเข้าไปหลัง PARITY BIT ถ้าอุปกรณ์รับข้อมูลตรวจไม่พบบิทนี้ก็แสดงว่าข้อมูลที่ได้รับเข้ามานั้นมีความผิดพลาดเกิดขึ้น สำหรับ STOP BIT นี้อาจมีจำนวนบิทเป็น 1, 1.5 หรือ 2 บิทก็ได้ รูปที่ 4 จะแสดงข้อมูลทั้ง 8 บิทที่ส่งออกมารวมทั้ง START, STOP และPARITY BIT ด้วย ซึ่งจะเห็นว่าสิ่งที่ส่งออกมาในแต่ละไบท์นั้น ไม่ได้มีเพียงข้อมูล 8
บิทเท่านั้น แต่อาจจะมีได้ถึง 12 บิท (กรณีที่ส่ง STOP BIT ออกมา 2 บิท) ดังนั้นถ้าเราทำการส่งด้วยอัตรา 2400 BAUD เราจะต้องใช้เวลาทั้งหมดเป็น 12 416 microseconds หรือ 4.99 microseconds ไม่ใช่ 3328 microseconds ดังที่ได้คำนวณไว้ในตอนต้น



รูปที่ 4 รูปแบบของข้อมูลแต่ละไบท์ในการรับส่งข้อมูลแบบอนุกรม

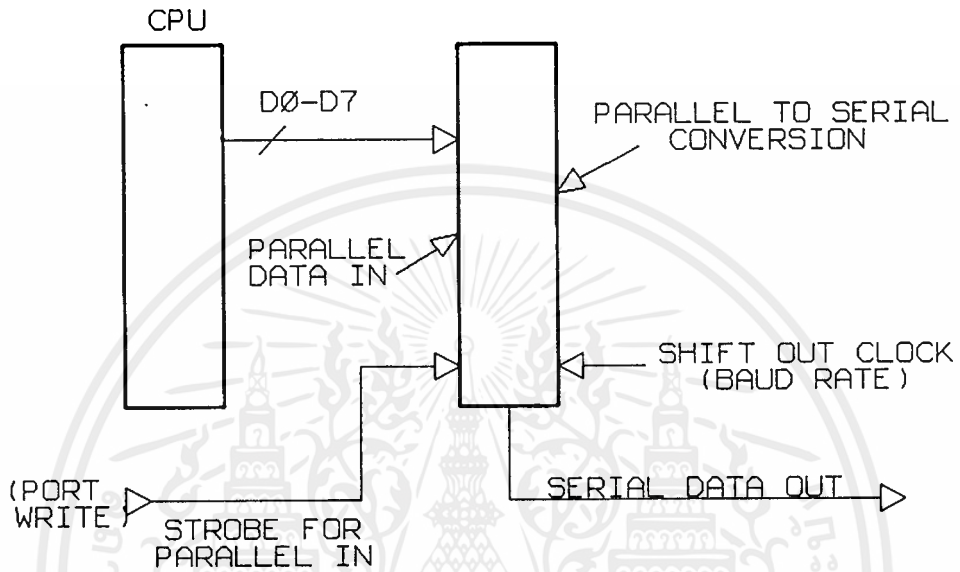
การเปลี่ยนข้อมูลจากแบบขนานเป็นข้อมูลแบบอนุกรม

โดยทั่งไปแล้ว การรับส่งข้อมูลภายในระบบมักจะเป็นการรับส่งข้อมูลแบบขนาน เนื่องจากมีความเร็วในการส่งผ่านข้อมูลที่สูงกว่าแบบอนุกรมมาก และยังมีความยุ่งยากน้อยกว่าอีกด้วย ดังนั้นในการรับส่งข้อมูลในระยะทางไกลๆ ที่จำเป็นจะต้องใช้การส่งผ่านข้อมูลแบบอนุกรม จึงจำเป็นที่จะต้องทำการเปลี่ยนรูปแบบของข้อมูลจากแบบขนาน ไปเป็นแบบอนุกรมก่อนที่จะทำการส่งข้อมูลออกไปตามสายส่งสำหรับหลักการง่ายๆ ที่ใช้ในการเปลี่ยนรูปแบบของข้อมูลนั้นมีขั้นตอนดังต่อไปนี้ คือ

1. ทำการเก็บข้อมูลแบบขนาน (ในที่นี้มีจำนวน 8 บิท) ไว้ใน SHIFT REGISTER

2. เลื่อนข้อมูลทั้ง 8 บิทออกไปให้กับอุปกรณ์รับข้อมูลทีละบิท โดยที่จะทำการส่งข้อมูลแต่ละบิทออกไปด้วยอัตราเดียวกับ BAUD RATE ที่ได้กำหนดไว้

รูปที่ 5 จะแสดงบล็อกไดอะแกรมของการทำงานทั้ง 2 ขั้นตอน คือข้อมูลแบบขนานนั้นจะถูกส่งจาก CPU ให้กับ SHIFT REGISTER จากนั้นจึงทำการเลื่อนข้อมูลออกทีละบิทด้วยอัตราของ BAUD RATE ที่กำหนด (โดยเลื่อนบิท D0 ก่อนและ D7 เป็นบิทสุดท้าย)



รูปที่ 5 บล็อกไดอะแกรมของการเปลี่ยนข้อมูลจากขนานเป็นอนุกรม

หลักการเบื้องต้นของการรับส่งข้อมูลแบบอนุกรม

สำหรับหลักการเบื้องต้นของการรับส่งข้อมูลแบบอนุกรมที่ได้กล่าวถึงนั้นสามารถที่จะสรุปเป็นข้อๆ ได้ดังนี้

1. ข้อมูลแบบขนานจากระบบจะถูกเปลี่ยนให้เป็นข้อมูลแบบอนุกรมเพื่อเตรียมที่จะส่งออกไปให้กับส่วนรับข้อมูล

2. ข้อมูลจะถูกส่งออกไปด้วยอัตราคงที่ค่าหนึ่ง ซึ่งเรียกว่า "BAUD RATE" คือ ถ้าทำการส่งข้อมูลด้วยอัตรา 1200 BAUD ก็แสดงว่าเป็นการส่งข้อมูลด้วยอัตรา 1200 บิทต่อ 1 วินาที ซึ่งก็คือ การส่งข้อมูลโดยใช้ความถี่ 1200 Hz นั่นเอง

3. ข้อมูลอนุกรมจะถูกส่งออกไปทีละบิท โดยทำการส่งบิท D0 เป็นบิทแรก และบิท D7 เป็นบิทสุดท้าย

4. ในขณะที่ยังไม่มีการส่งข้อมูลเข้าไปในสายส่ง สายส่งจะถูกทำให้ อยู่ในสภาวะลอจิกใดลอจิกหนึ่ง และเราเรียกสภาวะนี้ว่า "MARKING"

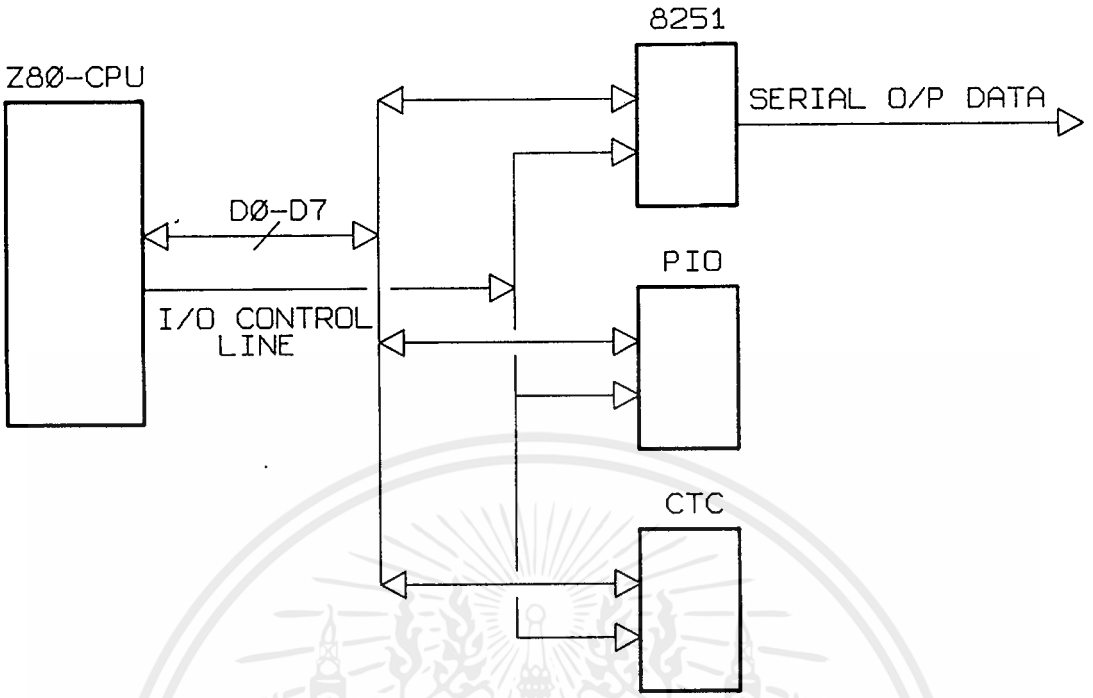
5. อุปกรณ์ส่งข้อมูลจะเพิ่มข้อมูลอีก 1 บิทเข้าไปหน้าบิท D0 ของข้อมูลที่จะส่งให้กับเครื่องรับ บิทที่เพิ่มเข้าไปนี้เรียกว่า "START BIT" สำหรับบิทนี้จะมีลอจิกตรงข้ามกับลอจิกของ MARKING เช่น ถ้าลอจิกของ MARKING เป็น "1" ลอจิกของ
บิทนี้ก็จะเป็น "0"

6. อุปกรณ์ส่งข้อมูลจะทำการเพิ่ม PARITY BIT เข้าไปหลังบิท D7 ของข้อมูล เพื่อใช้ในการตรวจสอบความผิดพลาดของข้อมูลที่เครื่องรับ (สำหรับบิทนี้เครื่องส่งอาจจะเพิ่มเข้าไปหรือไม่ก็ได้ ขึ้นอยู่กับผู้ออกแบบว่าต้องการที่จะเพิ่มบิทนี้เข้าไปหรือไม่)

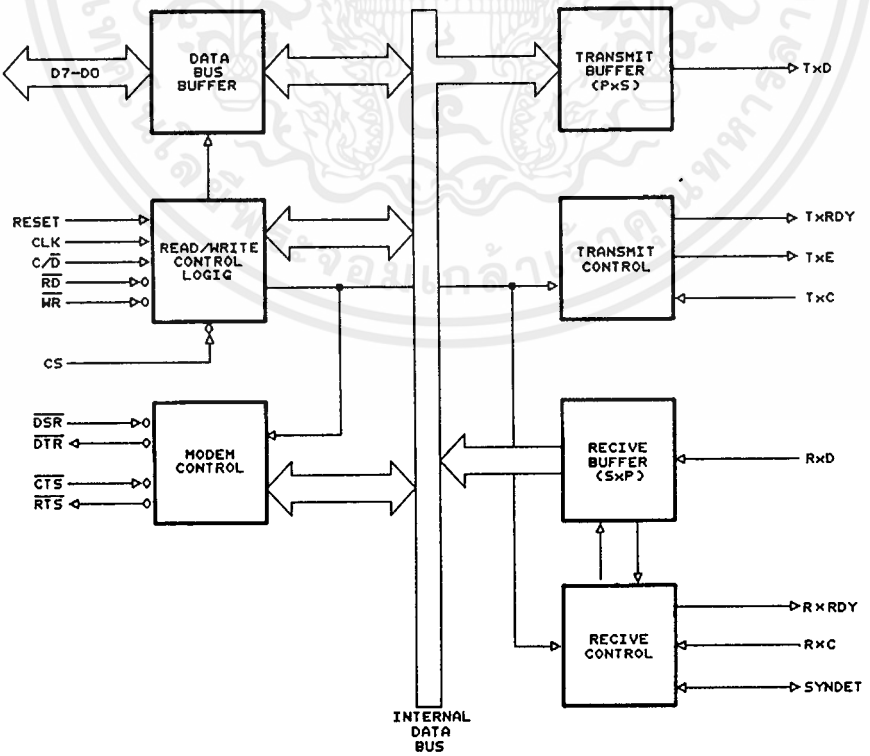
7. สำหรับบิทสุดท้ายที่ถูกเพิ่มเข้าไปหลัง PARITY BIT เรียกว่า "STOP BIT" ซึ่งอาจจะมีจำนวน 1, 1.5 หรือ 2 บิทก็ได้ และลอจิกของบิทนี้จะเป็นลอจิกเดียวกับลอจิกของ MARKING

8251 USART

ดังที่ได้กล่าวไว้แล้วว่า การส่งข้อมูลแบบอนุกรมไปตามสายส่งได้นั้น เราจำเป็นที่จะต้องทำการเปลี่ยนรูปแบบของข้อมูลเสียก่อน แต่โดยวิธีที่กล่าวมาแล้วนั้นเป็นวิธีที่ยังมีประสิทธิภาพไม่ดีพอ เนื่องจากว่า CPU จะต้องทำการรับส่งข้อมูลเองในช่วงเวลาที่เหมาะสม และในส่วนรับข้อมูล CPU ก็จะต้องทำการตรวจสอบความผิดพลาดของข้อมูลที่รับเข้ามาเองทุกอย่าง ทำให้เกิดความยุ่งยากในการออกแบบ และในส่วนรับข้อมูลก็ยังจะต้องมีอุปกรณ์ที่ใช้ในการเปลี่ยนข้อมูลกลับมาเป็นแบบขนานอีก ทำให้เกิดความสิ้นเปลือง ดังนั้นจึงจำเป็นที่จะต้องใช้อุปกรณ์ที่มีความสามารถที่จะเป็นได้ทั้งอุปกรณ์รับและส่งข้อมูลในตัวเดียวกัน สำหรับในบทนี้จะกล่าวถึงลักษณะและวิธีการใช้งานไอซี 8251 USART (UNIVERSAL SYNCHRONOUS/ASYNCRONOUS RECEIVER/TRANSMITTER) ซึ่งเป็นพอร์ทที่ใช้ในการรับส่งข้อมูลแบบอนุกรมที่มีประสิทธิภาพมากตัวหนึ่ง



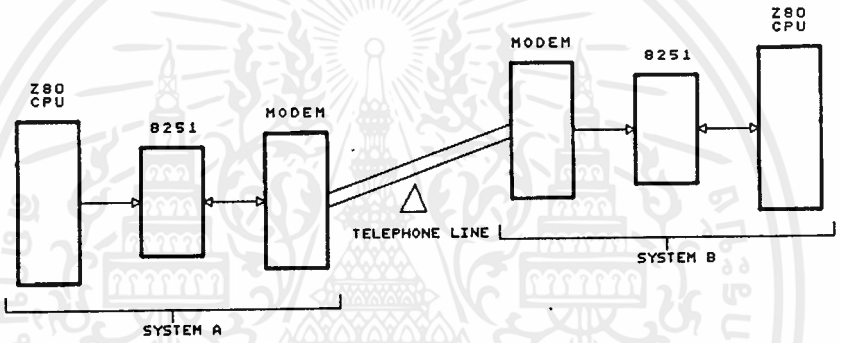
รูปที่ 6 การใช้งาน 8251 ร่วมกับ CHIP SUPPORT ของ Z80



รูปที่ 7 บล็อกไดอะแกรมของ 8251

จากรูปที่ 6 จะเห็นว่าเราสามารถที่จะนำ 8251 ไปเชื่อมต่อกับ Z80 ได้ในลักษณะเดียวกับ CHIP SUPPORT เบอร์อื่นๆ ของ Z80 เช่น PIO หรือ CTC และในรูปที่ 7 จะแสดงถึงบล็อกไดอะแกรมของ 8251 สำหรับส่วนแรกที่จะกล่าวถึงก็คือ DATA BUS BUFFER ซึ่งส่วนนี้ 8251 จะใช้ในการเชื่อมต่อระหว่าง 8251 กับบัสข้อมูลของ Z80 ส่วนต่อไปก็คือ READ/WRITE CONREOL LOGIC ซึ่งจะทำหน้าที่ในการควบคุมการรับส่งข้อมูลภายในของ 8251 ให้เป็นไปอย่างถูกต้อง

สำหรับ MODIM CONTROL นั้น จะใช้ในการติดต่อระหว่าง 8251 กับ MODIM (อุปกรณ์ที่ใช้ในการแปลงสัญญาณเพื่อส่งไปตามสายโทรศัพท์) รูปที่ 8 จะแสดงถึงบล็อกไดอะแกรม ของการใช้งานของ 8251 ร่วมกับ MODEM



รูปที่ 8 บล็อกไดอะแกรมของการใช้งาน 8251 ร่วมกับ MODEM

ส่วนที่จะกล่าวถึงต่อไปก็คือ TRANSMIT BUFFER (P-S) และ TRANSMIT CONTROL (P-S;PARALLEL TO SERIAL CONVERSION) ซึ่งใช้ในการส่งและควบคุมการส่งข้อมูลไปตามสายส่ง

สำหรับส่วนสุดท้ายที่จะกล่าวถึงก็คือ RECIEVER BUFFER (S-P;SERIAL TO PARALLEL CONVERSION) ซึ่งทำหน้าที่ในการรับ และควบคุมการรับข้อมูลของ 8251

การจัดเรียงขา และหน้าที่

8251 เป็นไอซีขนาด 28 ขา ซึ่งได้แสดงไว้ในรูปที่ 9 และเราสามารถที่จะแบ่งขาของ 8251 ออกเป็นกลุ่มๆ ได้ดังนี้คือ

1. กลุ่มที่ใช้ในการติดต่อกับ CPU

1.1) D0-D7 : ใช้ในการติดต่อกับ DATA BUS ของ CPU โดยตรง ซึ่งจะทำหน้าที่ในการรับส่งข้อมูลและคำสั่งต่างๆ ระหว่าง 8251 กับ CPU

1.2) RESET : 8251 จะถูกรีเซทเมื่อขานี้ได้รับลอจิก "1" ซึ่งเราอาจจะต่อมาจากขา RESET ของ Z80 โดยผ่าน INVERTER ก่อนก็ได้

1.3) CLK (CLOCK) : ใช้ในการควบคุมช่วงเวลาการทำงานภายในของ 8251 สำหรับการใช้งานนั้นจะต่อเข้าโดยตรงกับระบบ อย่างไรก็ตามสัญญาณที่ขา CLK นี้ไม่เกี่ยวข้องกับอัตราการรับส่งข้อมูลหรือ BAUD RATE แต่อย่างใด

1.4) RD : เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการส่งข้อมูลแบบขนานออกมาที่ DATA BUS เพื่อส่งให้กับ CPU

1.5) WR : เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการรับข้อมูลแบบขนานจาก DATA BUS ของระบบ

1.6) C/D (CONTROL/DATA) : ขา C/D นี้จะใช้ในการทำให้ 8251 ทราบว่า CPU ต้องการที่จะติดต่อกับ CONTROL ERGISTER หรือ DATA REGISTER โดยที่ถ้าขานี้ได้รับลอจิก "1" ก็แสดงว่า CPU ต้องการที่จะติดต่อกับ CONTROL REGISTER แต่ถ้าได้รับลอจิก "0" ก็แสดงว่า CPU ต้องการที่จะติดต่อกับ DATA REGISTER

1.7) CS (CHIP SELECT) : ในกรณีที่ขานี้ได้รับลอจิก "0" ก็จะเป็นการ ENABLE 8251 โดยทั่วไปแล้วสัญญาณที่ขานี้จะได้มาจากการถอดรหัสพอร์ทแอดเดรส ดังที่ใช้กับ CHIP SUPPORT อื่นๆ

2. กลุ่มที่ใช้ในการติดต่อกับ MODEM

2.1) DSR (DATA SET READY) : ขานี้เป็นขาที่ใช้ในการรับสัญญาณจากอุปกรณ์ภายนอก ซึ่ง CPU สามารถที่จะตรวจสอบสัญญาณที่ขานี้ได้ โดยการอ่านค่าในรีจีสเตอร์สถานะ (ซึ่งจะกล่าวถึงต่อไป) และระดับของสัญญาณที่ขานี้จะใช้ในการแสดงว่าอุปกรณ์ภายนอกพร้อมที่จะทำการติดต่อด้วยหรือยัง

2.2) DTR (DATA TERMINAL READY) : ขานี้เป็นเอาท์พุทที่ใช้ในการบอกให้อุปกรณ์ภายนอกทราบว่าCPU พร้อมที่จะทำการติดต่อด้วย 2.3) CTS (CLEAR TO SEND) : ขานี้เป็นขาอินพุทที่ใช้ในการทำให้ 8251 เริ่มทำการส่งข้อมูลได้ สิ่งที่ต้องระวังในการใช้งานขานี้ก็คือ เมื่อไม่ได้ใช้งานขานี้จะต้องถูกต่อเข้ากับลอจิก "0" ถ้าไม่เช่นนั้น 8251 จะทำการส่งข้อมูลไม่ได้

2.4) RTS (READY TO SEND) : ขานี้เป็นเอาท์พุทที่ CPU จะเป็นผู้ควบคุมสัญญาณที่ขานี้เอง (ขา DTR ก็ถูกควบคุมโดย CPU เช่นกัน)

3. กลุ่มที่ใช้ในการส่งข้อมูล

3.1) TxD (TRANSMIT DATA OUTPUT) : เป็นขาที่ใช้ในการส่งข้อมูลไปตามสายส่ง

3.2) TxC (TRANSMIT BAUD RATE CLOCK) : ขานี้เป็นขาที่ใช้ในการส่งสัญญาณคล็อกที่ใช้ในการส่งข้อมูล ซึ่งก็คือความถี่ที่ใช้ในการกำหนด BAUD RATE นั่นเอง โดยปกติแล้วจะต้องช้ากว่าสัญญาณคล็อกของระบบไม่น้อยกว่า 30 เท่า

3.3) TxRDY : ขานี้จะใช้ในการทำให้ CPU ทราบว่า 8251 พร้อมที่จะรับข้อมูลจาก CPU เพื่อที่จะทำการส่งต่อไปแล้วหรือยัง และขานี้อาจจะนำไปใช้ในการขออินเทอร์รัพท์ก็ได้

3.4) TxEMOTY : ขานี้จะใช้ในการแสดงว่าข้อมูลที่ CPU ส่งให้กับ 8251 นั้นได้ถูกส่งออกไปให้กับอุปกรณ์อื่นๆ หมดแล้ว โดยที่ 8251 จะทำให้ขานี้เป็น "1" และเมื่อCPU ทำการส่งข้อมูลชุดต่อไปให้กับ 8251 ขา TxEMPTY ก็จะเป็น "0" จนกว่า 8251 จะทำการส่งข้อมูลนี้ออกไปหมด 8251 ก็จะทำให้ขานี้กลับเป็น "0" อีกครั้ง

4. กลุ่มที่ใช้ในการรับส่งข้อมูล

4.1) RxD : ใช้ในการรับข้อมูลแบบอนุกรมจากสายส่ง

4.2) RxC : เป็นขาที่ใช้ในการรับสัญญาณคล็อกที่ใช้ในการรับข้อมูลโดยปกติแล้วจะทำการต่อเข้ากับ TxC โดยตรง

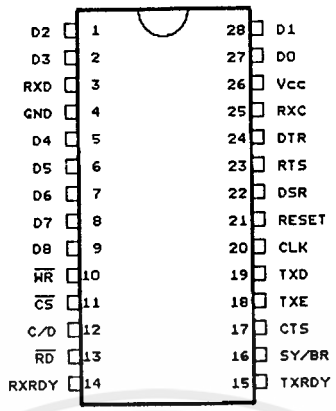
4.3) RxRDY : จะใช้ในการแสดงว่า 8251 พร้อมที่จะส่งข้อมูลให้กับ CPU และขานี้อาจจะใช้ในการขออินเทอร์รัพท์ได้เช่นเดียวกับขา TxRDY

4.4) SYNDET : ขานี้จะใช้ในการรับข้อมูลแบบ SUNCHRONOUS เท่านั้น (8251สามารถที่จะทำการรับส่งข้อมูลได้ทั้งแบบ SYNCHRONOUS และแบบ ASUNCGRONOUS ซึ่งได้กล่าวถึงต่อไป) โดยที่เราสามารถที่จะโปรแกรมให้ขานี้เป็นอินพุทหรือเอาท์พุทก็ได้ โดยที่เมื่อขา SYNDET นี้ถูกโปรแกรมเป็นเอาท์พุทนั้นขา SYNDET จะให้ลอจิก "1" เมื่อ 8251 สามารถที่จะตรวจจับ SYNDET CHAEACTER ได้และจะให้ลอจิก "0" เมื่อ CPU ทำการอ่านรีจีสเตอร์สถานะสำหรับขา SYNDET นี้จะให้ลอจิก "1" ในอีกกรณีหนึ่งคือ เมื่อ 8251 ได้รับข้อมูลจากสายส่งเป็น "0" หมดตั้งแต่ START BIT จนถึงSTOP BIT

ในกรณีที่ขา SYNDET ถูกโปรแกรมให้เป็นอินพุทนั้น ถ้าขานี้ได้รับสัญญาณขอบขาขึ้น (สัญญาณเปลี่ยนจากลอจิก "0" เป็น "1") 8251 ก็จะถือว่าข้อมูลที่ขา RxD เป็นข้อมูลทันที และเราสามารถที่จะทำให้ลอจิกที่ขานี้กลับเป็น "0" ได้ในสัญญาณ RxC ลูกต่อไป

5. กลุ่มไฟเลี้ยงของ 8251

8251 ใช้ไฟเลี้ยงเพียงชุดเดียว คือ +5V กับ GND เท่านั้น ดังนั้นขาไฟเลี้ยงของ 8251 จึงมีเพียง 2 ขา คือ Vcc กับGND



รูปที่ 9 การจัดเรียงขาบน 8251

การเชื่อมต่อระหว่าง 8251 กับ Z80

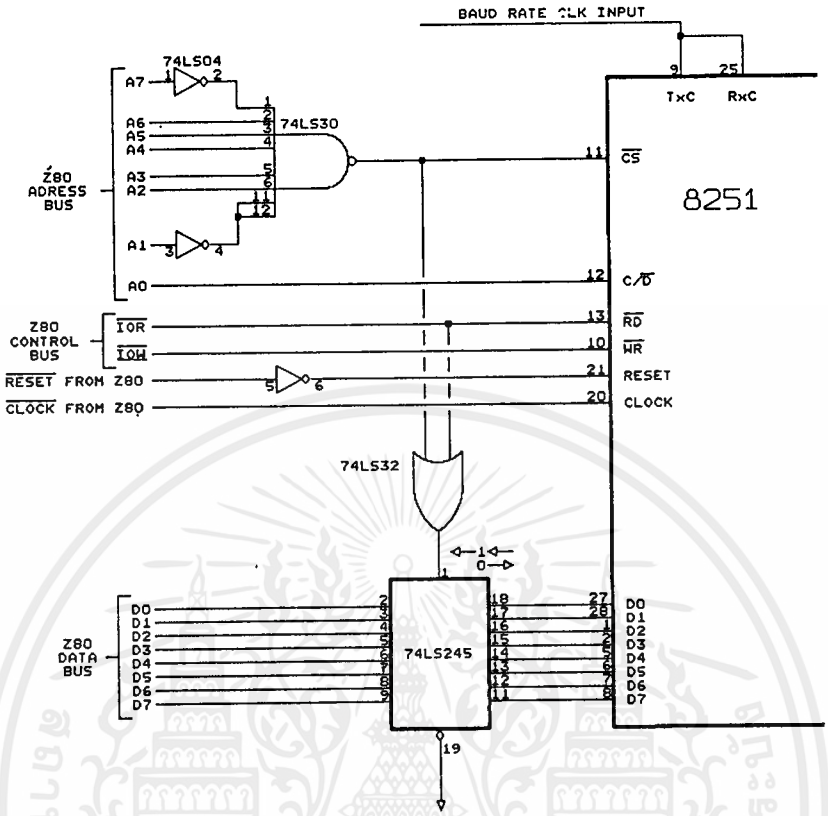
ในหัวข้อนี้จะกล่าวถึงวิธีการนำเอา 8251 ไปใช้งานร่วมกับ Z80 รูปที่ 10 จะแสดงถึงวิธีในการเชื่อมต่อ 8251 กับ Z80

จากรูปที่ 10 จะเห็นว่าระหว่างบัสข้อมูลของ 8251 และ Z80 จะมี DATA BUFFER ต่ออยู่ (74LS245) เพื่อช่วยในการขับกระแสของ Z80

ขา CS ของ 8251 ได้จากการถอดรหัสแอดเดรส A1-A7 ของ Z80 ส่วนขา A0 ของ Z80 จะต่อเข้ากับขา C/D ของ8251 โดยตรง และจากรูปเราจะได้ค่า PORT ADDRESS ของ 8251 เป็น 7DH สำหรับรีจีสเตอร์ควบคุม และ 7CH สำหรับ DATA REGISTER

สำหรับขา RD และ WR ของ 8251 นั้นจะต่อเข้ากับ IORD และ IOWR ของระบบตามลำดับ ขา RESET ของ Z80 ต่อผ่าน INVERTER เข้ากับขา RESET ของ 8251 ส่วนขา CLOCK ของ 8251 ถูกต่อเข้ากับสัญญาณคล็อกของระบบโดยตรง

สำหรับคล็อกที่ใช้ในการกำหนด BAUD RATE นั้นจะถูกนำไปป้อนให้กับขา TxC และ RxC ในกรณีนี้แสดงว่าการรับส่งข้อมูลใช้ BAUD RATE เท่ากัน) อย่างไรก็ตามคล็อกที่นำมาป้อนให้กับ TxC และ RxC นี้อาจจะเป็นคนละชุดกันก็ได้แต่ที่สำคัญก็คือ จะต้องให้สัมพันธ์กับอุปกรณ์ปลายทางที่ต้องการจะทำการติดต่อด้วยและดังที่ได้กล่าวไว้แล้วว่า คล็อกที่ใช้ในการกำหนด BAUD RATE นั้นจะต้องน้อยกว่า 30 เท่าของความถี่ของคล็อกที่ใช้ในระบบ นั่นคือที่ BAUD RATE เท่ากับ 2400 BAUD ความถี่ที่ใช้เป็นคล็อกของระบบอย่างน้อยจะต้องเท่ากับ 72 KHz

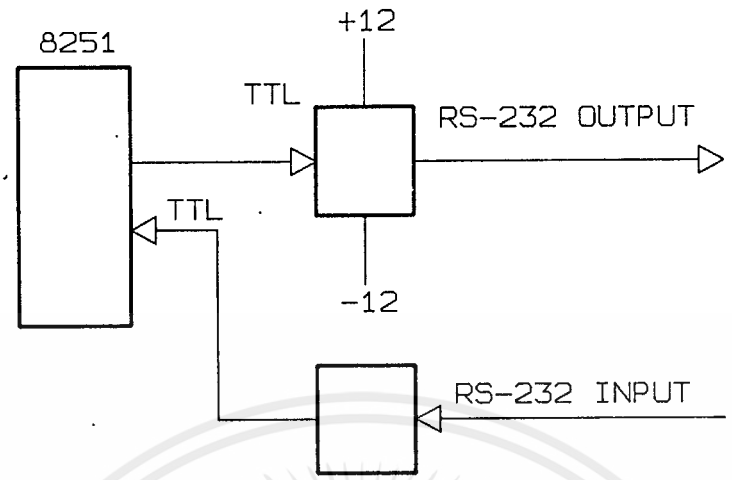


รูปที่ 10 ตัวอย่างแสดงการต่อใช้งาน 8251 ร่วมกับ Z80

การเชื่อมต่อกับสายส่งข้อมูล

การรับส่งข้อมูลแบบอนุกรมนั้น นิยมนำไปใช้ในการรับส่งข้อมูลในระยะทางไกล เพราะสามารถที่จะลดค่าใช้จ่ายในการวางสายลงได้มาก แต่อย่างไรก็ตามระดับสัญญาณที่ใช้ในวงจร (+5V) นั้นไม่สามารถที่จะส่งไปได้ไกลนัก ดังนั้นก่อนที่จะส่งข้อมูลไปในสายส่งจะต้องทำการเปลี่ยนระดับแรงดันของสัญญาณใหม่ เพื่อให้สามารถที่จะส่งสัญญาณได้ไกลขึ้น แต่การที่จะเปลี่ยนระดับของสัญญาณไปเป็นเท่าใดนั้น ขึ้นอยู่กับมาตรฐานที่ใช้ในการส่งในที่นี้จะสมมติว่า การรับส่งข้อมูลใช้มาตรฐาน RS-232 ซึ่งนิยมใช้กันมากในปัจจุบัน

สำหรับมาตรฐาน RS-232 นี้จะใช้ระดับแรงดันในสายส่งประมาณ +/- 12 VOLTS รูปที่ 11 จะแสดงบล็อกไดอะแกรมของการเปลี่ยนระดับแรงดันของสัญญาณจากระดับสัญญาณที่ใช้อุปกรณ์พวก TTL ไปเป็นระดับสัญญาณที่ใช้กับมาตรฐาน RS-232 ในการส่งข้อมูล และการเปลี่ยนจากระดับของ RS-232 ไปเป็นระดับสัญญาณ TTL ในการรับข้อมูลจากสายส่ง

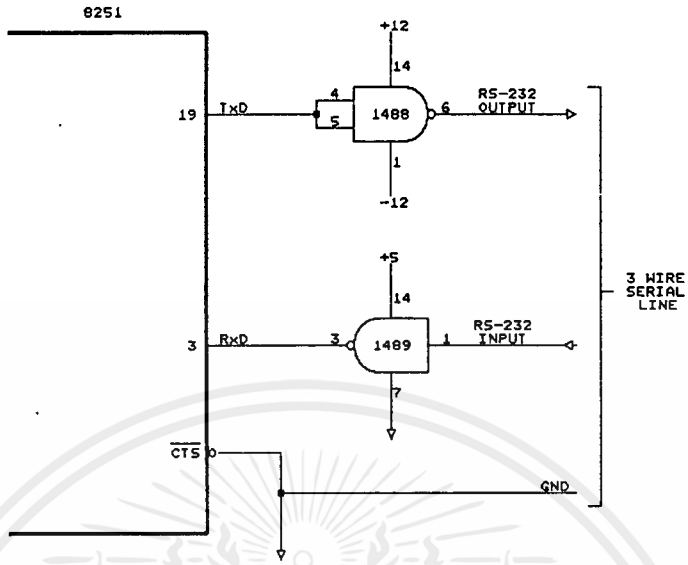


รูปที่ 11 บล็อกไดอะแกรมแสดงการเปลี่ยนระดับของสัญญาณ

อุปกรณ์ที่ใช้ในการทำหน้าที่ทั้งสองอย่างนี้ก็คือ IC เบอร์ MC1488 ซึ่งทำหน้าที่ในการเปลี่ยนระดับของสัญญาณจาก TTL ไปเป็น RS-232 และ MC1489 ซึ่งทำหน้าที่ในการเปลี่ยนระดับของสัญญาณจาก RS-232 ไปเป็น TTL

ในรูปที่ 12 จะแสดงวิธีในการนำเอา MC1488 และ MC1489 มาใช้งานร่วมกับ 8251 และจะเห็นว่าขา CTS จะถูกต่อกับลอจิก "0" ด้วย และจะเห็นว่าเราใช้สายส่งเพียง 3 เส้นคือ TxD, RxD และ GND เท่านั้น

อย่างไรก็ตามวิธีการเชื่อมต่อ 8251 กับสายส่งนี้เป็นเพียงวิธีหนึ่งในหลายวิธี ซึ่งก็ขึ้นอยู่กับจุดประสงค์ในการนำไปใช้งานในหัวข้อต่อไปจะกล่าวถึงการโปรแกรมสั่งงาน 8251 ให้ทำการรับส่งข้อมูลทั้ง 2 โหมดคือ ASYNCHRONOUS และ SYNCHRONOUS MODE



รูปที่ 12 การใช้งาน MC1488 และ MC1489

โปรแกรม 8251

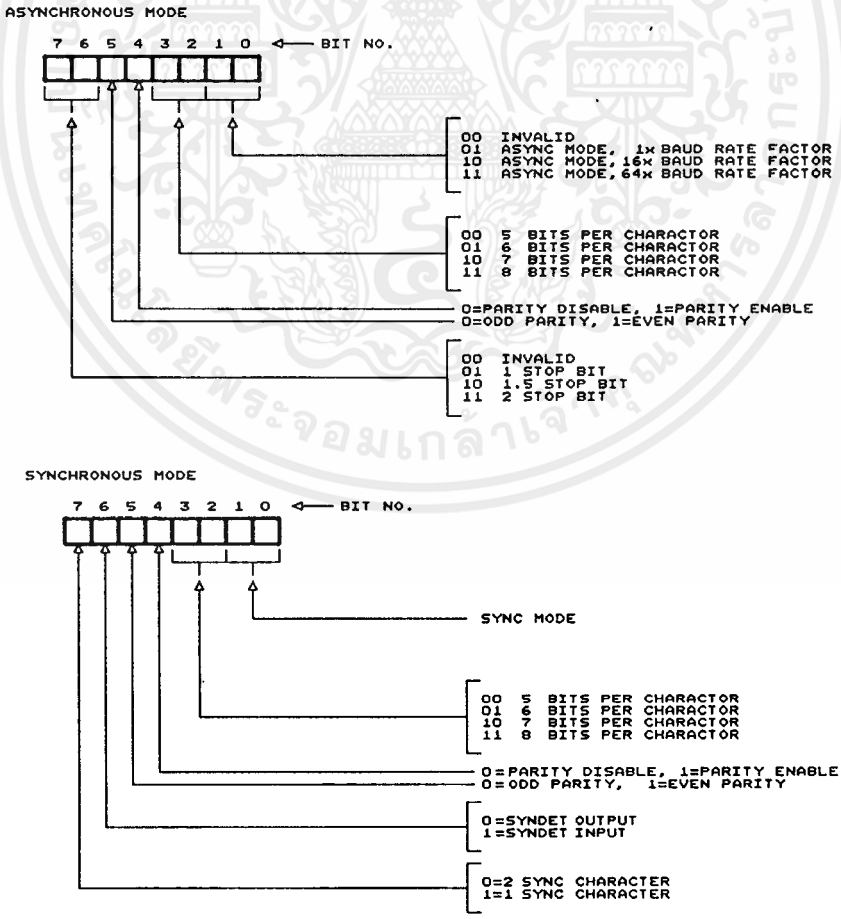
ดังที่ได้กล่าวไว้ในตอนต้นแล้วว่า การรับส่งข้อมูลบน 8251 นั้นสามารถที่จะทำได้ 2 แบบ คือ แบบ SYNCHRONOUS และแบบ ASYNCHRONOUS ซึ่งมีลักษณะความแตกต่างในการรับส่งข้อมูลดังนี้คือ

การรับส่งข้อมูลแบบ ASYNCHRONOUS นั้น จะทำการส่ง START และ STOP BIT ออกไปพร้อมกับข้อมูลด้วย ดังที่ได้กล่าวไว้ในตอนต้น ส่วนแบบ SYNCHRONOUS นั้นจะไม่มีการส่ง START และ STOP BIT แต่จะทำการส่ง SYNC CHARACTER แทน ซึ่ง 8251 จะทำการส่ง SYNC CHARACTER ออกไปเรื่อยๆ ในทันทีที่ CPU สั่งให้ 8251 ทำการส่งข้อมูลได้ จนกว่า CPU จะทำการส่งข้อมูลให้กับ 8251 เพื่อทำการส่งไปในสายส่ง

สำหรับการทำงานใน ASYNCHRONOUS MODE นั้น เมื่อ 8251 รับข้อมูลเข้ามาแล้วก็จะทำการตรวจสอบ START, STOP และ PARITY BIT ถ้าเกิดความผิดพลาดขึ้นก็จะไปแสดงไว้ในรีจีสเตอร์สถานะ ซึ่งจะกล่าวถึงอีกครั้งในภายหลังส่วนการรับข้อมูลใน SYNCHRONOUS MODE นั้น 8251 จะพยายาม SYNC กับ SYNC CHARACTER ที่เครื่องส่ง ส่งออกมาโดยการเลื่อนข้อมูลเข้ามาทีละบิท เพื่อนำมาเทียบกับ SYNC CHARACTER ที่ได้โปรแกรมไว้ว่าตรงกันหรือไม่ และเมื่อ 8251 ตรวจจับ SYNC CHARACTER ได้แล้ว (SYNC CHARACTER ที่ส่งมาตรงกับที่

ได้โปรแกรมไว้) ก็จะทำให้ขา SYNDET เป็น "1" เพื่อให้ CPU รับทราบ (ซึ่งอาจจะใช้วิธีการอินเทอร์รัพท์ก็ได้) การที่จะให้8251 ทำการตรวจจับ SYNC CHARACTER นั้นจะต้องทำการโปรแกรมให้ 8251 ทำงานใน HUNT MODE ซึ่งจะได้กล่าวถึงต่อไป และ 8251 จะออกจาก HUNT MODE เอง เมื่อทำการ SYNC ได้

รูปที่ 13 จะแสดงการจัดเรียงบิทใน CONTROL WORD (MODE WORD) ทั้งใน ASYNCHRONOUS และ SYNCHRONOUS MODE ซึ่ง MODE WORD นี้จะถูกส่งให้กับรีจีสเตอร์ควบคุมโดยการอ้างถึงพอร์ทแอดเดรส 7DH (จากวงจรรูปที่ 10) พร้อมกับการให้สัญญาณ WR กับ 8251 ซึ่งสามารถที่จะทำได้โดยการใช้คำสั่ง OUT (7DH), Aโดยที่ค่าใน รีจีสเตอร์ A เป็นค่าของ MODE WORD และ 8251 จะถือว่าข้อมูลที่ส่งให้กับรีจีสเตอร์ควบคุมเป็น MODE WORD ในกรณีที่ข้อมูลนี้เป็นข้อมูลไบท์แรกที่ถูกส่งให้กับ 8251หลังจากที่ถูกรีเซ็ท(อาจจะเป็นการรีเซ็ททางฮาร์ดแวร์หรือทำINTERNAL RESET ก็ได้ซึ่งจะได้กล่าวถึงต่อไป)



รูปที่ 13 การจัดเรียงบิทใน MODE WORD

ในที่นี้จะสมมติว่า 8251 ได้รับการรีเซ็ทแล้ว และจะทำการโปรแกรมให้ทำงานใน ASYNCHRONOUS MODE (บิท 0 และ 1 จะกำหนดว่าจะให้ 8251 ทำงานในโหมดใด ในกรณีที่บิททั้งสองนี้เป็น "0" ทั้งคู่เท่านั้น จึงจะเป็นการเลือกให้ 8251 ทำงานใน SYNCHRONOUS MODE) ดังนั้นจึงต้องทำการอ้างถึงพอร์ท 7DH และกำหนดการทำงานของ 8251 ดังนี้

1. กำหนดให้ใช้ค่า BAUD RATE เท่ากับ 2400 BAUD ในกรณีที่เราใช้ความถี่ของ คล็อก ที่ขา TxC และ RxC เท่ากับ BAUD RATE พอดี ก็เลือกโปรแกรมให้บิท 0 และ 1 เป็น "1" และ "0" ตามลำดับ แต่ในกรณีที่ความถี่ของคล็อก ที่ใช้มีค่ามากกว่า BAUD RATE ที่ใช้ 16 หรือ 64 เท่า ก็จะต้องทำการโปรแกรมให้ BAUD RATE FACTOR เป็น 16xหรือ 64x ตามลำดับ

2. กำหนดให้ข้อมูลที่จะส่งออกไปมีจำนวน 8 บิท (บิท 2 และ 1 เป็น "1") ซึ่งเราอาจจะโปรแกรมให้เป็น 5, 6 หรือ 7 บิทก็ได้ ในกรณีที่ข้อมูลที่เราต้องการที่จะส่งออกไปมีน้อยกว่า 8 บิท 8251 จะทำการตัดบิทสูงทิ้งไป เช่น ถ้าเลือกให้ส่งเพียง 5 บิท 8251 ก็จะทำการตัดบิท D7-D5 ทิ้ง

3. กำหนดให้ PARITY BIT เป็น PARITY คู่ (บิท 4 และ 5 เป็น "1")

4. เลือกใช้ STOP BIT จำนวน 2 บิท (บิท 7 และ 6 เป็น "1")

ดังนั้นข้อมูลที่จะทำการส่งให้กับพอร์ท 7DH จะเป็น 11111101B หรือ 0FDH สำหรับ ชุดคำสั่งของ Z80 ที่จะใช้ในการโปรแกรมสั่งงาน 8251 แสดงได้ดังนี้

```
LD A, 0FDH
OUT (7DH), A
```

สำหรับการโปรแกรม 8251 ให้ทำงานใน SYNCHRONOUS MODE นั้น ก็จะสมมติว่า 8251 ได้รับการรีเซทแล้วเช่นกัน และกำหนดการทำงานของ 8251 ดังนี้

1. เลือกให้ทำงานใน SYNCHRONOUS MODE (บิท 0 และ 1 เป็น "0"ทั้งคู่)

2. เลือกให้ส่งข้อมูลครบทั้ง 8 บิท (บิท 2 และ 3 เป็น "1" ทั้งคู่)

3. กำหนดให้ส่ง PARITY คี่ (บิท 4 และ 5 เป็น "1" และ "0" ตามลำดับ)

4. กำหนดให้ SYNDET เป็นเอาท์พุท (บิท 6 เป็น "0")

5. กำหนดให้ส่ง SYNC CHARACTER 2 ตัว (บิท 7 เป็น "0")

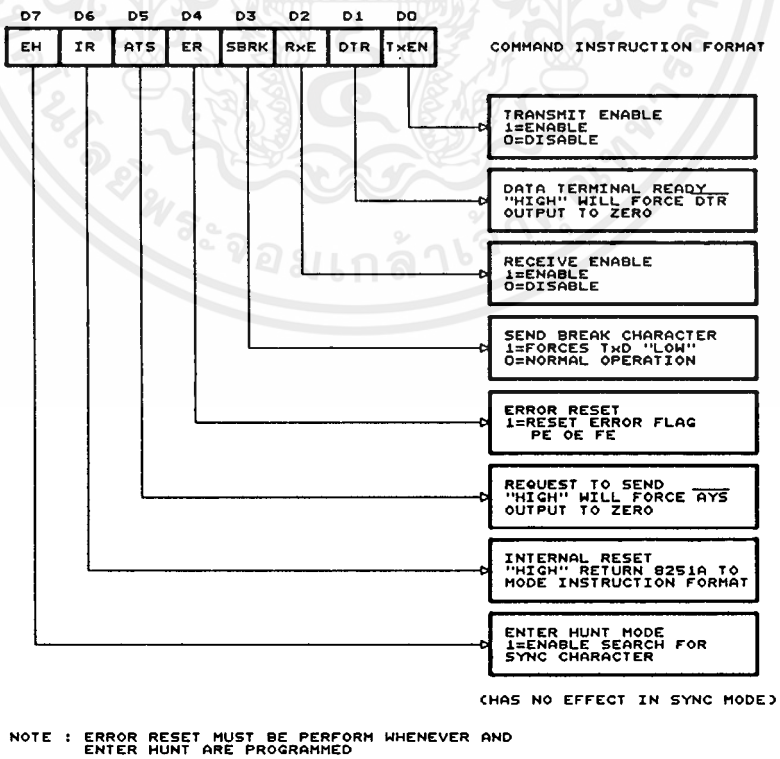
ดังนั้นเราจะได้ข้อมูลที่จะส่งให้กับรีจีสเตอร์ควบคุม คือ 00011100B หรือ 1CH สำหรับโปรแกรมที่จะใช้ในการสั่งงาน 8251 จะเป็นดังนี้

```
LD A, 1CH
OUT (7DH), A
```

สำหรับในโหมดนี้ หลังจากส่ง MODE WORD ให้กับรีจีสเตอร์ควบคุมแล้ว 8251 จะถือว่าไบท์ต่อไปที่ CPU ส่งให้นั้นเป็น SYNC CHARACTER ในกรณีนี้เราจะต้องทำการส่ง SYNC CHARACTER ให้กับ 8251 อีก 2 ไบท์ และข้อมูลไบท์ต่อไป 8251 จะถือเป็นการ COMMAND WORD

ส่วนในโหมด ASYNCHRONOUS นั้น หลังจากที่ได้ส่ง MODE WORD ให้กับ 8251 แล้ว ข้อมูลไบท์ต่อไปที่ถูกส่งให้กับรีจีสเตอร์ควบคุมจะถือว่าเป็น COMMAND WORD ทั้งสิ้น จนกว่า 8251 จะได้รับการรีเซ็ทอีก จึงจะถือว่าข้อมูลที่ส่งมาให้กับรีจีสเตอร์ควบคุมนั้นเป็น MODE WORD

ในที่นี้กำหนดให้ 8251 ทำการรับและส่งข้อมูลได้ (บิท 0 และ 2 เป็น "1") ทำการรีเซท ERROR FLAG ในรีจีสเตอร์สถานะคือ PE, FE และ OE สำหรับการทำ INTERNAL RESET (IR) นั้นสามารถที่จะทำได้โดยการทำให้บิท 6 (IR) ของ COMMAND WORD เป็น "1" แต่ในที่นี้จะยังไม่ทำ INTERNAL RESET ดังนั้น COMMAND WORD ที่จะส่งให้กับ 8251 ก็คือ 00010101B หรือ 15H สำหรับการส่ง COMMAND WORD นี้ให้กับรีจีสเตอร์ควบคุมของ 8251 นั้น สามารถที่จะทำได้โดยใช้วิธีเดียวกับการส่ง MODE WORD คือส่งออกไปที่ พอร์ท 7DH

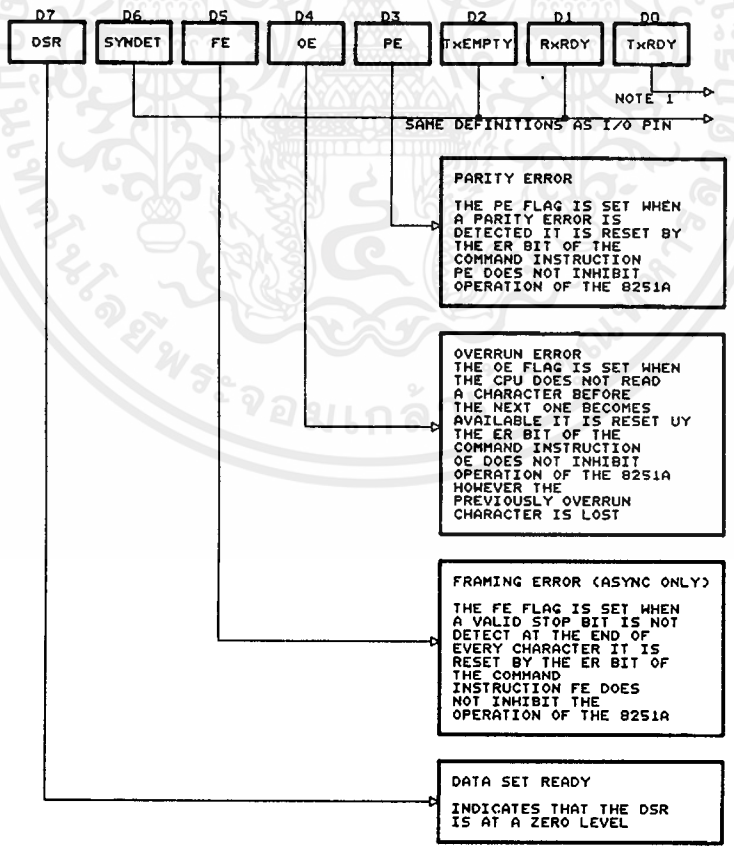


รูปที่ 14 การจัดเรียงบิทใน COMMAND WORD

ในกรณีที่เป็น SYNCHRONOUS MODE อาจจะกำหนดให้ 8251 เข้าสู่ HUNT MODE ได้โดยการทำให้บิท 7 เป็น "1" ในกรณีนี้ COMMAND WORD จะเป็น 10010101B หรือ 95H

สำหรับรีจีสเตอร์ตัวต่อไปที่จะกล่าวถึงก็คือ รีจีสเตอร์สถานะ ซึ่งจะทำหน้าที่ในการแสดงสภาวะของ 8251 ในขณะที่ทำงานอยู่ ว่ามีความผิดพลาดเกิดขึ้นหรือไม่ และยังมีหน้าที่ในการแสดงสถานะที่ขาต่างๆ ของ 8251 ที่เกี่ยวข้องกับการรับส่งข้อมูลด้วยว่าเป็นอย่างไร

สำหรับรีจีสเตอร์สถานะนี้ CPU สามารถที่จะทำการตรวจสอบได้โดยการอ่านข้อมูลจากพอร์ท 7DH ส่วนบิทต่างๆ ที่ใช้ในการแสดงสถานะของขาของ 8251 คือ DSR, SYNDET, TxEMPTY,RxRDY และ TxRDY นั้นจะช่วยให้ CPU ทราบถึงความพร้อมของอุปกรณ์รับส่งข้อมูล สำหรับอีก 3 บิทที่เหลือนั้น จะแสดงความผิดพลาดที่เกิดจากการรับส่งข้อมูลของ 8251 ซึ่งเราสามารถที่จะทำการรีเซทบิททั้งสามนี้ได้โดยการทำให้บิท 4 ใน COMMAND WORD เป็น "1"



รูปที่ 15 การจัดเรียงบิทบนรีจีสเตอร์สถานะ

สำหรับบิททั้ง 3 ที่แสดงความผิดพลาดของการรับส่งข้อมูลมีดังนี้คือ

1. PE (PARITY ERROR) จะเป็น "1" ในกรณีที่ 8251 พบว่ามีความผิดพลาดในการตรวจสอบ PARITY

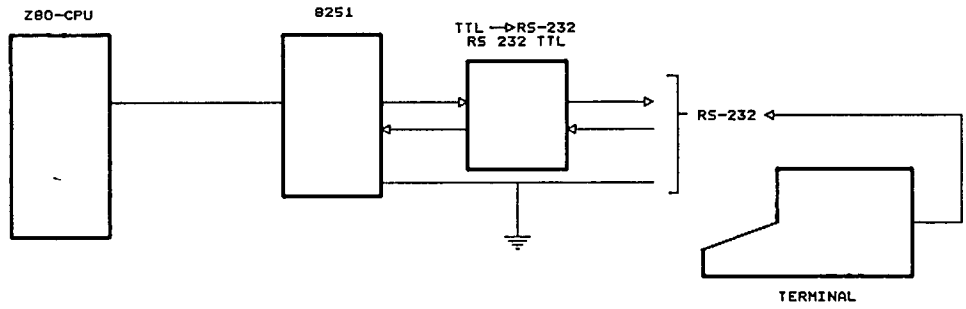
2. OE (OVERRUN ERROR) จะเป็น "1" เมื่อข้อมูลใหม่ถูกส่งเข้ามาทับข้อมูลเก่า โดยที่ข้อมูลเก่ายังไม่ถูก CPU อ่านออกไป

3. FE (FRAMING ERROR) จะเป็น "1" ในกรณีที่ 8251 ตรวจหา STOP BIT ไม่พบ

สำหรับรูปที่ 16 จะแสดงตัวอย่างโปรแกรมที่ใช้ในการสั่งงาน 8251 และการส่งข้อมูลออกไปตามสายส่งโดยผ่าน 8251

```
1800      3FDD          LD  A,0DDH      ;LOAD THE MODE WORD IN A REG
1802      D37D          OUT (7DH),A     ;OUTPUT TO THE 8251
;
; 2 STOP BITS , ODD PARITY , ENABLE PARITY , 8 BITR/CHAR.
; X 1 BAUD RATE MULTIPIER
;
1804      3E15          LD  A,15H       ;COMMAND WORD IN REG
1806      D37D          OUT (7DH),A     ;OUTPUT TO 8251
;
; TX ENABLE , RX ENABLE , RESET ERRORS
;
1808      3E49   LOOP1  LD  A,49H       ;ASCII "I"
180A      D37C          OUT (7CH),A     ;OUTPUT CHARACTER TO TX
180C      DB7D   LOOP2  IN  A,(7DH)     ;READ STATUS REGISTER
180E      CB47          BIT 0,A         ;TEST BIT 0 = 1
1810      CA0C18        JP  Z,LOOP2     ;BUFFER NOT EMPTY,KEEP POLLING
1813      C30818        JP  LOOP1       ;BUFFER EMPTY,NEXT CHA.
```

รูปที่ 16 ตัวอย่างโปรแกรมที่ใช้ในการโปรแกรม 8251



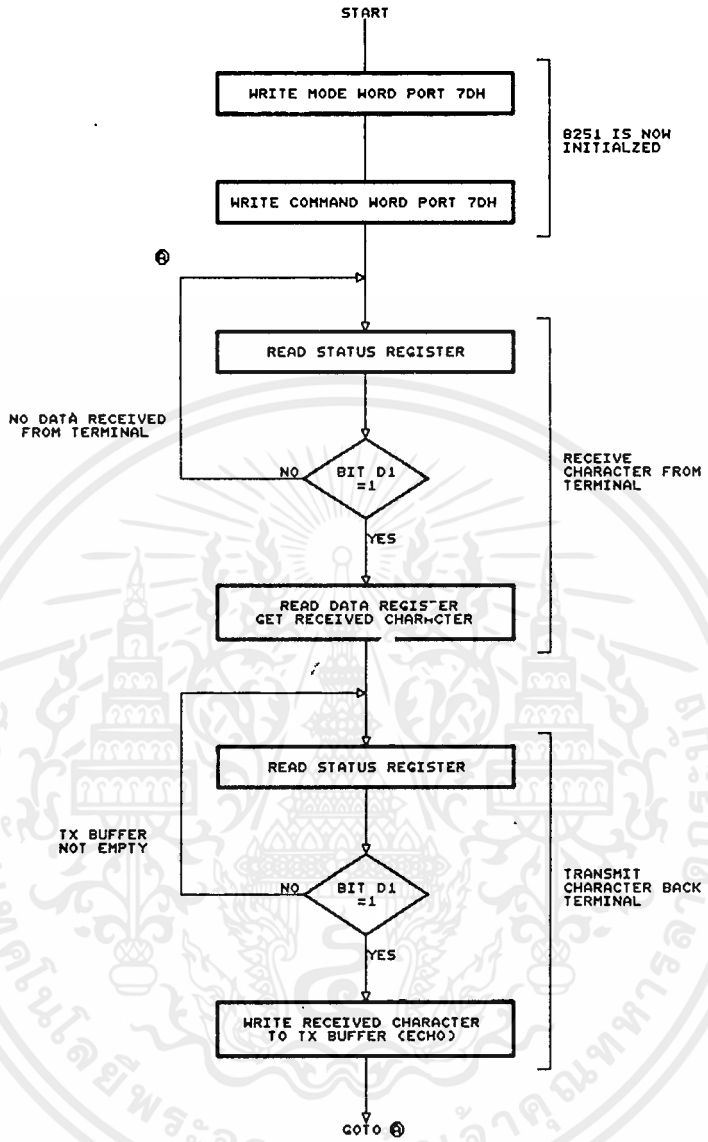
รูปที่ 17 บล็อกไดอะแกรมแสดงการติดต่อระหว่าง Z80 กับอุปกรณ์ภายนอก

รูปที่ 17 เป็นบล็อกไดอะแกรมแสดงการติดต่อระหว่าง Z80 กับ อุปกรณ์ปลายทาง (INTERNAL) โดยผ่าน 8251 ส่วนในรูปที่ 18 และ 19 นั้น จะแสดงตัวอย่างโฟลว์ชาร์ต และโปรแกรมที่ใช้ในการรับส่งข้อมูลโดยผ่าน 8251

```
1800      3EDD            LD  A,0DDH        ;MODE WORD
1802      D37D            OUT (7DH),A       ;WRITE TO 8251
1804      3E15            LD  A,15H         ;COMMAND WORD IN REG
1806      D37D            OUT (7DH),A       ;WRITE TO 8251
;
; 8251 IS NOW INITIALIZED
;
; FIRST THE DEVICE WILL WAIT FOR A CHARACTER TO BE
; SENT TO IT FROM THE TERMINAL
;
1808      DB7D   LOOP1    IN  A,(7DH)       ;READ THE STATUS REGISTER
180A      CB4F            BIT 1,A           ;TEST BIT D1 = 1
180C      CAD818          JP  Z,LOOP1       ;NOT READY KEEP POLLING
;
; WHEN WE REACH HERE A CHARACTER IS RECEIVED
;
```

```
180F      D07C            IN  A,(7CH)       ;READ THE CHARACTER
1811      47              LD  B,A
;
; WE WILL NOT ERROR CHECK THE DATA
;
; NOW TO TRANSMIT THE DATA
;
1812      DB7D   LOOP2    IN  A,(7DH)       ;READ STATUS REGISTER
1814      CB47            BIT 0,A           ;TEST BIT DO = 1
1816      CA1218          JP  Z,LOOP2       ;XMIT NOT READY,KEEP POLLING
;
; XMIT IS READY TO OUTPUT ANOTHER CHARACTER
;
1819      78              LD  A,B
181A      D37C            OUT (7CH),A       ;CHARACTER TO 8251
181C      C30818          JP  LOOP1         ;START OVER AGAIN
;
; END OF ECHO ROUTINE
;
```

รูปที่ 18 ตัวอย่างโปรแกรมที่ใช้ในการรับส่งข้อมูลระหว่าง Z80 กับอุปกรณ์ปลายทางโดยผ่าน 8251



รูปที่ 19 แสดงโฟลว์ชาร์ตของโปรแกรมในรูปที่ 18

บทที่ 4
การสร้างและการทำงาน

ภาค Distortion Meter

มาตรฐานของ CCITT ว่าด้วยการรับส่งข้อมูลแบบ ASYNCHONOUS จะเริ่มต้นด้วย START BIT ที่เป็น LOW ขนาด 1 BIT แล้วจึงตามด้วยข้อมูลขนาด 5,6,7 หรือ 8 BIT จึงปิดท้ายด้วย STOP BIT ที่เป็น HIGH โดยจะมีขนาด 1,1.5 หรือ 2 BIT ตามมาตรฐาน CCITT ซึ่ง TIME PERIOD ของแต่ละ BIT จะมีค่าคงที่เสมอ โดยมี BAUD RATE เป็นตัวกำหนด ดังนั้นในการวัดค่า DISTORTION เพียงแต่นำช่วงเวลาของข้อมูลที่ได้รับไปเปรียบเทียบกับช่วงเวลามาตรฐาน แล้วนำค่าความแตกต่างที่ได้แสดงออกมาเป็นเปอร์เซ็นต์ DISTORTION

จาก BLOCK DIAGRAM

- START BIT DETECTOR ทำหน้าที่ CHECK START BIT เมื่อสัญญาณ CLOCK ACTIVE LOW ก่อนที่จะการคำนวณหา DISTORTION เมื่อพบ START BIT แล้วจะมี OUTPUT ที่เป็น HIGH ส่งไปยัง CLOCK GENERATOR เพื่อทำการสร้าง CLOCK

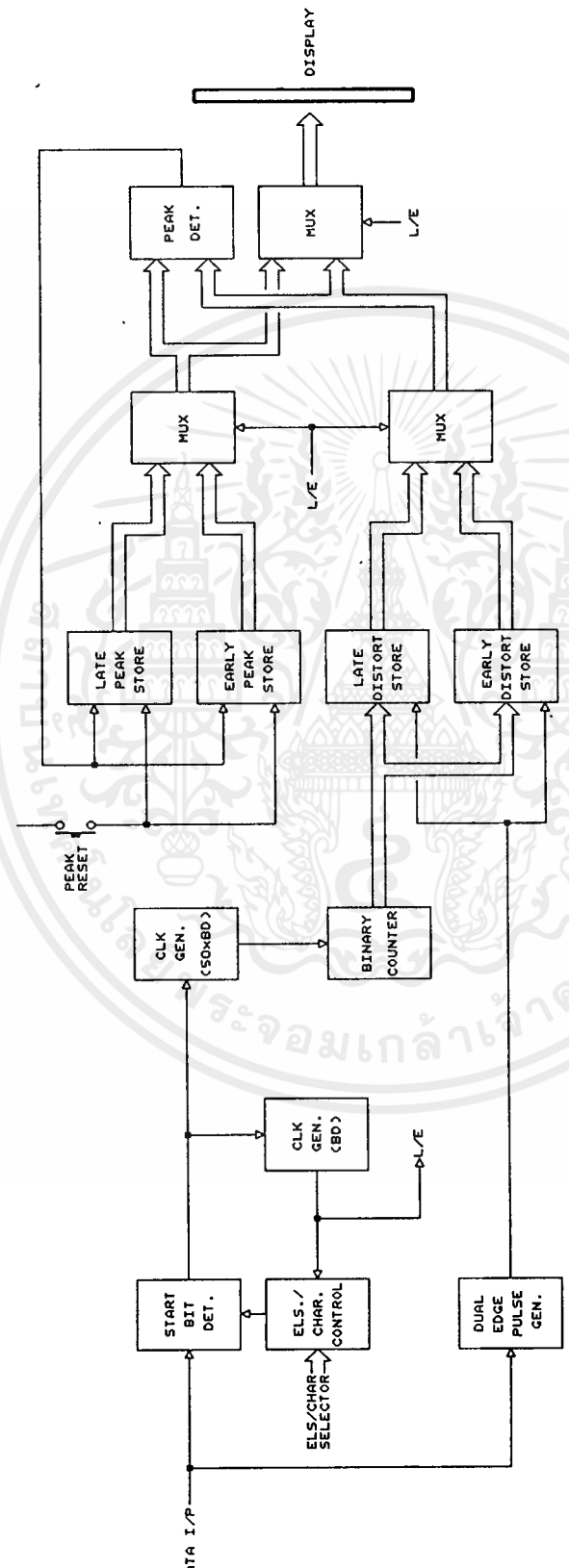
- CLOCK GENERATOR (BD) ทำหน้าที่สร้างความถี่โดยมีค่าเท่ากับ BIT RATE สำหรับ CLOCK ที่เป็น ELEMENT/CHARACTOR หรือ จำนวน BIT/CHARACTOR โดยจะสร้าง CLOCK จนกว่าจะครบจำนวน FORMAT 5,6,7 หรือ 8 BIT/CHARACTOR ที่ตั้งไว้ ถ้าครบแล้วจะสร้าง PULSE ไปทำการ RESET ให้แก่ START BIT DETECTOR โดยให้ OUTPUT เป็น LOW

- CLOCK GENERATOR (50xBD) จะผลิตความถี่ = 50 เท่าของ Bit Rate

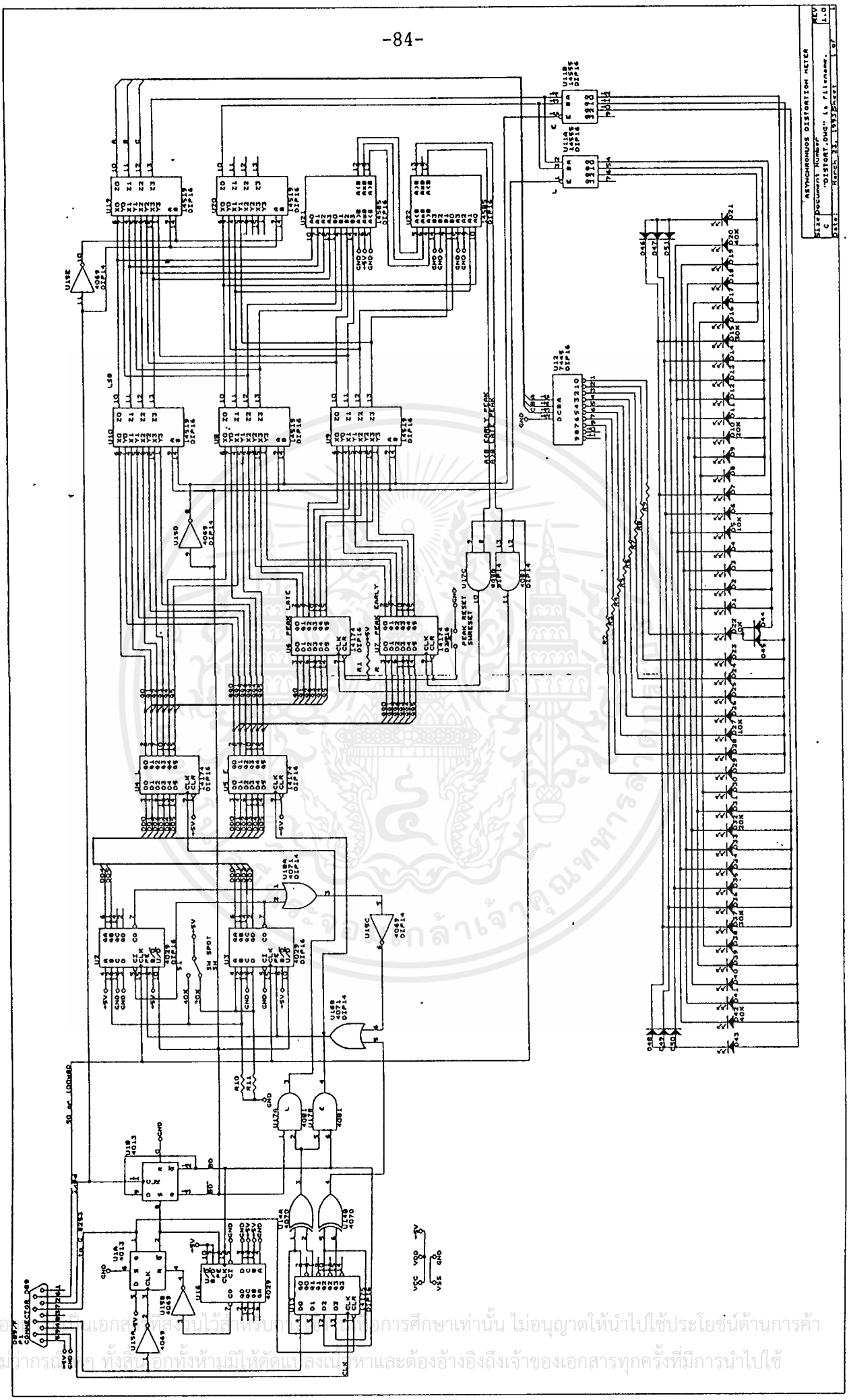
- DUAL EDGE PULSE GENERATOR ทำหน้าที่สร้างสัญญาณ PULSE แคบๆ จาก DATA INPUT เพื่อที่จะ LATCH ค่าที่ได้จาก BINARY COUNTER ไปเก็บไว้โดยแบ่งเป็น LATE และ EARLY แล้วนำค่าที่ได้ส่วนหนึ่งผ่าน MUX ไปแสดงผลอีกส่วนหนึ่ง จะนำไปเปรียบเทียบกับค่าที่เก็บไว้ใน PEAK STORE ซึ่งมีทั้ง LATE และ EARLY PEAK STORE

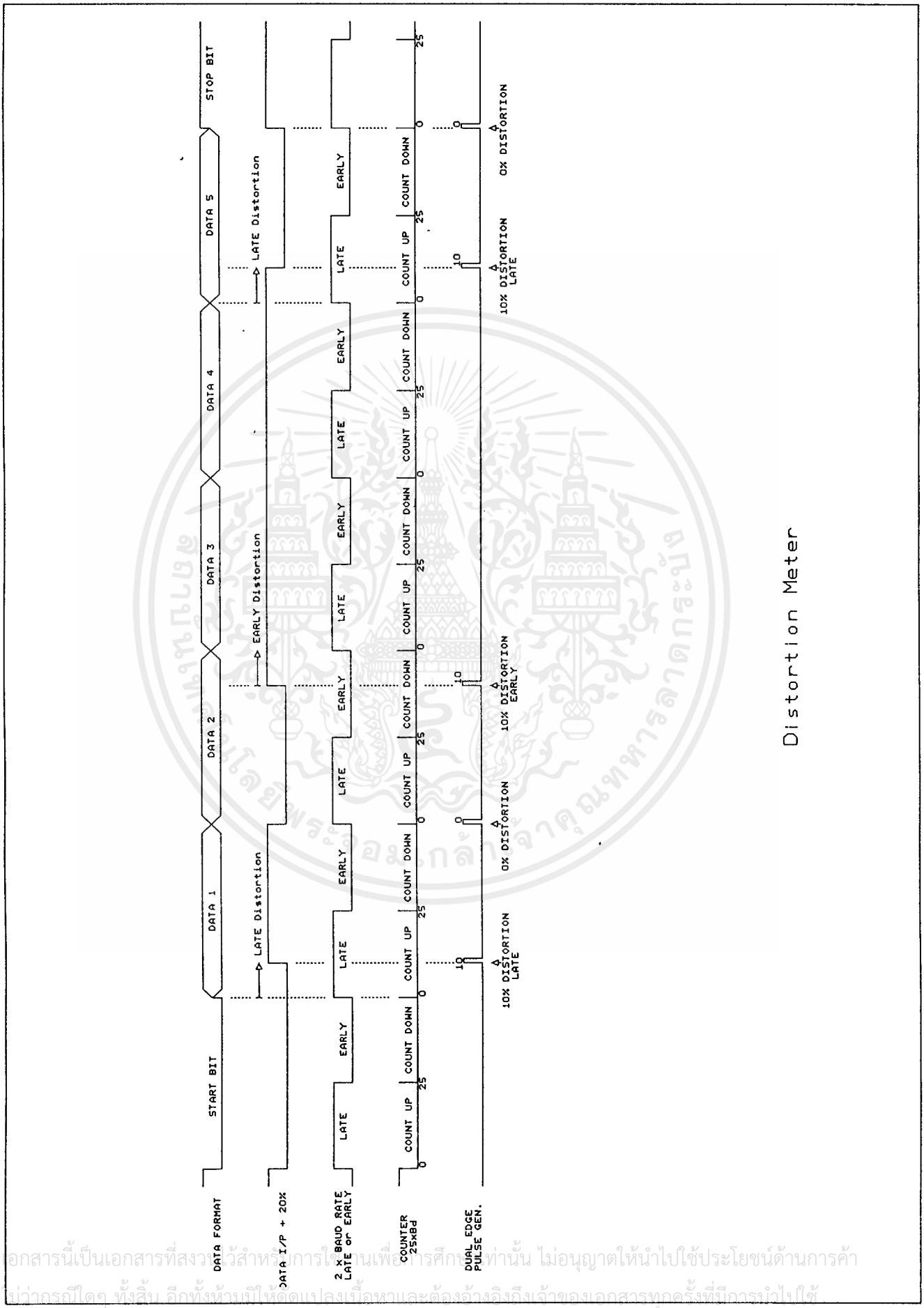
- DISPLAY ทำหน้าที่แสดงผลทั้งค่า PEAK STORE และ DISTORT STORE โดยวิธีการของ MUX

- ในขณะที่ CLOCK GEN. เริ่มทำงาน U_2 และ U_3 จะเริ่มนับ CLOCK จากส่วนที่เป็น 50 เท่าของ BIT RATE และ U_2, U_3 จะถูก SET ให้เป็นวงจรนับ 50



Distortion Meter Block Diagram





Distortion Meter

- U_{13} และ U_{14} จะทำหน้าที่เป็นวงจร DUAL EDGE PULSE GEN. โดยจะสร้าง PULSE ขึ้นมาขณะที่ INPUT มีการเปลี่ยนแปลง $U_{17A,B}$ จะเป็นตัวสร้างสัญญาณเพื่อ LATCH ค่า ซึ่งจะแบ่งเป็น 2 ส่วน คือส่วนที่เป็น LATE และ EARLY U_4 และ U_5 จะเป็นตัวเก็บค่า BINARY จากที่ COUNTER นับได้ ซึ่ง U_4 จะเป็นตัวเก็บค่า LATE และ U_5 เก็บค่า EARLY จาก TIMING DIAGRAM ในรูป ของ DISTORTION METER จะเห็นว่าเมื่อ DATA INPUT เริ่มตกจาก HI เป็น LOW COUNTER จะเริ่มทำงานโดย 2xBD จะเป็นตัวกำหนดในส่วนที่เป็น LATE หรือ EARLY และ 50 xBD จะเป็นตัวใช้สำหรับวัดค่า DISTORTION ที่เกิด ซึ่งจะ COUNT UP ขณะที่เป็น LATE และ COUNT DOWN ขณะที่เป็น EARLY

- U_8,U_9 และ U_{10} จะเป็นตัว MULTIPLEX เพื่อเลือกค่าที่เก็บไว้ใน U_4,U_5 และค่า PEAK ที่เก็บไว้ใน U_6,U_7 โดยจะเลือกเป็นคู่คือ ค่า LATE และ PEAK LATE หรือ EARLY และ PEAK EARLY ค่าที่เลือกออกมาจะแยกเป็น 2 ส่วนคือ ส่วนที่นำไปแสดงผลและส่วนที่เปรียบเทียบค่า PEAK

- U_{21},U_{22} จะทำหน้าที่เป็นส่วนเปรียบเทียบค่า PEAK โดยจะนำค่า LATE หรือ EARLY ที่ได้มาเปรียบเทียบกับค่า PEAK LATE หรือ PEAK EARLY เดิมเช่นในกรณีถ้าค่า LATE มากกว่าค่า PEAK LATE เดิม U_6,U_7 ก็จะส่งสัญญาณไปยัง U_6 เพื่อทำการเก็บค่า LATE นั้นไว้ในส่วน PEAK LATE (U_6) และถ้าค่า EARLY น้อยกว่าค่า PEAK EARLY (เนื่องจากขณะที่เป็น EARLY COUNTER จะเป็น COUNT DOWN) U_6,U_7 ก็จะส่งสัญญาณไปยัง U_7 เพื่อทำการเก็บค่า EARLY นั้นไว้ใน PEAK EARLY (U_7)

- U_{19},U_{20} จะเป็นตัว MULTIPLEX นำค่าทั้งหมด (LATE, PEAK LATE, EARLY และ PEAK EARLY) ซึ่งเป็นค่า BINARY ไปเปลี่ยนเป็นค่า DECIMAL โดย U_{12},U_{11} เพื่อแสดงผล

- ส่วนแสดงผล LED D_{22} จะแสดงผล DISTORTION ที่ 0% LED D_{21}, D_{43} จะแสดงค่า DISTORTION ที่มากกว่า 40% และ LED แต่ละตัวจะเป็นตัวละ 2%

ภาค GENERATOR

จะแบ่งเป็น 2 ส่วนใหญ่ๆ คือ ส่วน CPU และส่วน Distortion Generator

1) CPU

CPU จะอ่านคำต่างๆ จาก Function Key ซึ่งประกอบด้วย

1.1) Selector Keys เพื่อเป็นการเลือกข้อมูลที่ต้องการส่งออกไป ซึ่งมีอยู่ 5 แบบ

1.1.1 แบบ A ส่งตาม switch 1 ถ้วน ซึ่งสามารถกำหนดให้ Bit ใดเป็น Mark ("1") หรือ space ("0") ได้ตามต้องการแต่ทั้งนี้ก็ขึ้นอยู่กับการเลือกจำนวน ELS/Char.

1.1.2 แบบ B เป็นแบบ Binary Sequence โดยจะเริ่มจาก (00000000) ไปจนถึง (11111111) และกลับมาเริ่มต้นอีกครั้งจำนวน Bit ที่จะส่งขึ้นอยู่กับการเลือกจำนวน ELS/Char.

1.1.3 แบบ C เป็นแบบ RY โดยจะส่งข้อมูล (10101010) และ (01010101) สลับกัน ซึ่งจำนวน Bit ที่จะส่งก็ขึ้นอยู่กับการเลือกจำนวน ELS/Char. เช่นกัน

1.1.4 แบบ D เป็นข้อมูล THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

1.1.5 แบบ E เป็นข้อมูล "FACULTY OF ENGINEERING OF (KMITL)" เป็นข้อมูลแบบ 5 ELS/Char.

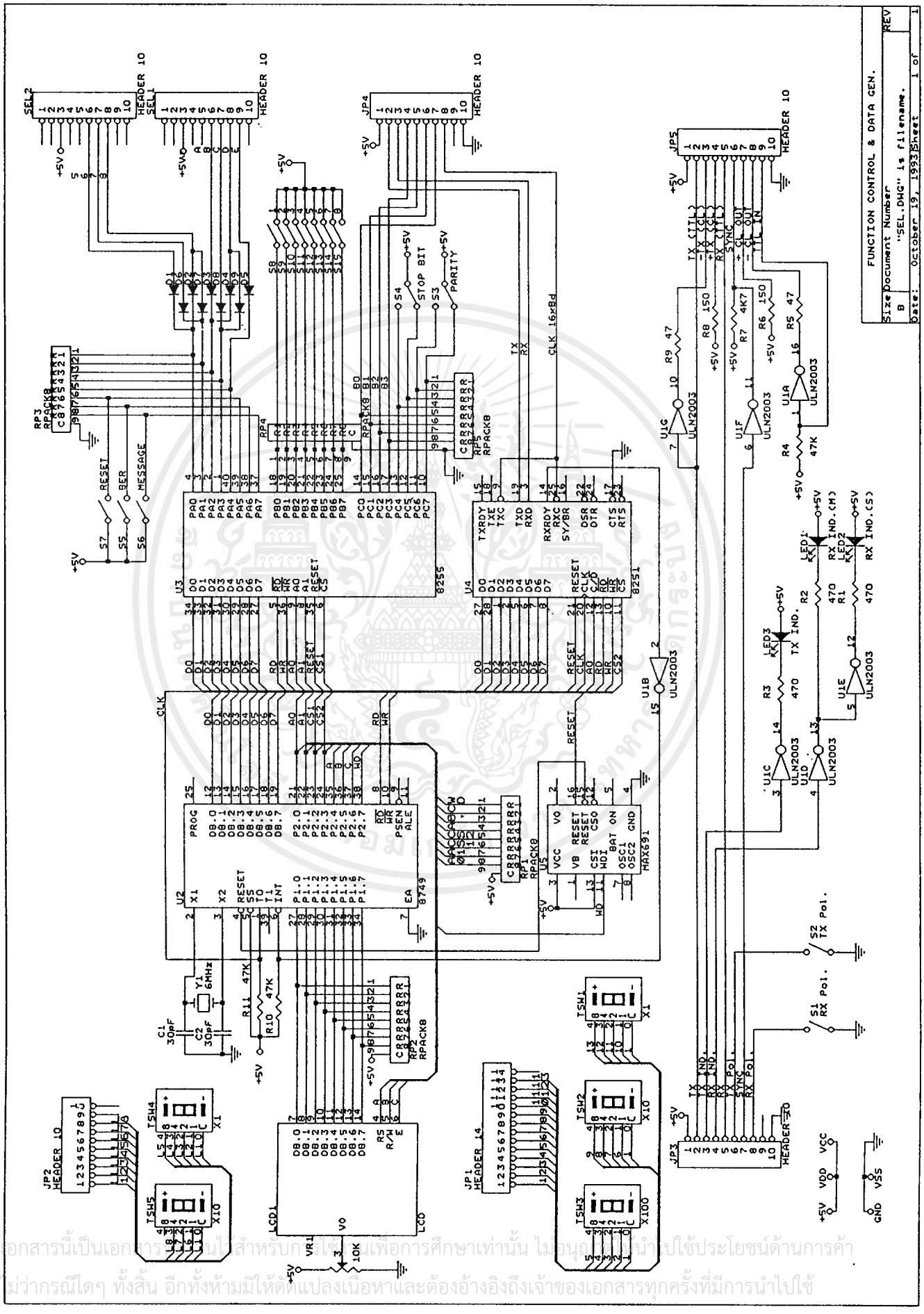
1.2) Selector Element/Charector ซึ่งเป็นจำนวนของข้อมูลในการส่งว่าต้องการส่งแบบ 5,6,7 หรือ 8 ELS (Bit)

1.3) Stop Pulse ซึ่งเป็น Bit สุดท้ายของการส่งใน char. นั้นๆ สามารถเลือกได้ 3 แบบ คือ 1,1.5 หรือ 2 ELS (Bit)

1.4) Parity เมื่อเลือกที่ตำแหน่ง ODD หรือ EVEN จำนวน ELS/Char. จะเพิ่มขึ้นจากที่เลือกไว้อีก 1 ELS (Bit) โดยจะเพิ่มขึ้นต่อจาก Data ELS (Bit) หลังสุด ซึ่งจะเป็น Mark ('1") หรือ space ("0") ก็ขึ้นอยู่กับการเลือก ODD,EVEn และข้อมูลที่จะส่งออกไป

1.5) BER ปกติเมื่ออยู่ตำแหน่ง "OFF" ที่จอ LCD Display จะแสดงข้อมูลที่รับเข้ามาทั้งสองบรรทัด แต่เมื่ออยู่ในตำแหน่ง"ON" ที่จอ LCD Display ในบรรทัดบนจะแสดงข้อมูลที่ได้รับตามปกติ แต่บรรทัดล่างจะมี Counter 6 หลัก อยู่ 2 ชุด ชุดแรกแสดงจำนวน Charector ที่ผิดพลาดในการส่งและ Loop Back เข้ามาเปรียบเทียบกับส่วนชุดที่สองจะแสดงจำนวน Charector ที่ส่งออกไปทั้งหมด

1.6) RESET เมื่อถูกกดจะทำการ Reset Counter ทั้ง 2 ชุด พร้อมทั้ง Restart ข้อมูลที่จะส่งออก



1.7) MESSAGE เมื่ออยู่ในตำแหน่ง STOP CPU จะหยุดทำการส่งข้อมูล และจะส่งข้อมูลเมื่ออยู่ในตำแหน่ง RUN

การทำงานของวงจร

- Port(U_3) ทำหน้าที่เป็นตัว Interface กับ function Switch ต่างๆ โดยที่ CPU(U_2) จะอ่านตำแหน่งต่างๆ ของ function ผ่าน Port(U_3)

- CPU ซึ่งเป็นแบบ Single cpmponent microprocessor (Single chip) ในกรณีที่ทำหน้าที่เป็นด้านส่งเมื่อทำการอ่านค่าต่างๆ จาก Port แล้ว จะทำการ Preset Communication Port U_4 เพื่อให้พร้อมที่จะส่ง เช่นการ Set ค่า STOP Bit , จำนวน Bit/Char., Parity, CLOCK Rate เป็นต้น จากนั้นก็จะทำการส่งข้อมูลที่ถูก Program ไว้ใน EPROM ของ CPU ออกไปทาง Communication Port ความเร็วที่ส่งออกทาง Com.Port ของ U_4 จะขึ้นอยู่กับ CLOCK Rate ที่ได้รับการ Set speed ที่ภาค Frequency Generator โดยจะเป็น 16 เท่าของ Bit Rate

ในกรณีที่ทำหน้าที่เป็นด้านรับ CPU จะอ่าน Data ที่ได้รับจาก Com.Port ทางด้านรับและนำ data ที่ได้ไปแสดงที่จอ LCD

- watch Dog circuit (U_5) จะตรวจสอบการทำงานของ CPU เพราะขณะที่ CPU ทำงานจะมี Pulse ปรากฏที่ขา WDI ตลอดเวลา ในบางครั้งเมื่อ CPU เกิดอาการ Hang U_5 จำทำการ Reset CPU ใหม่

2) Distortion Generator

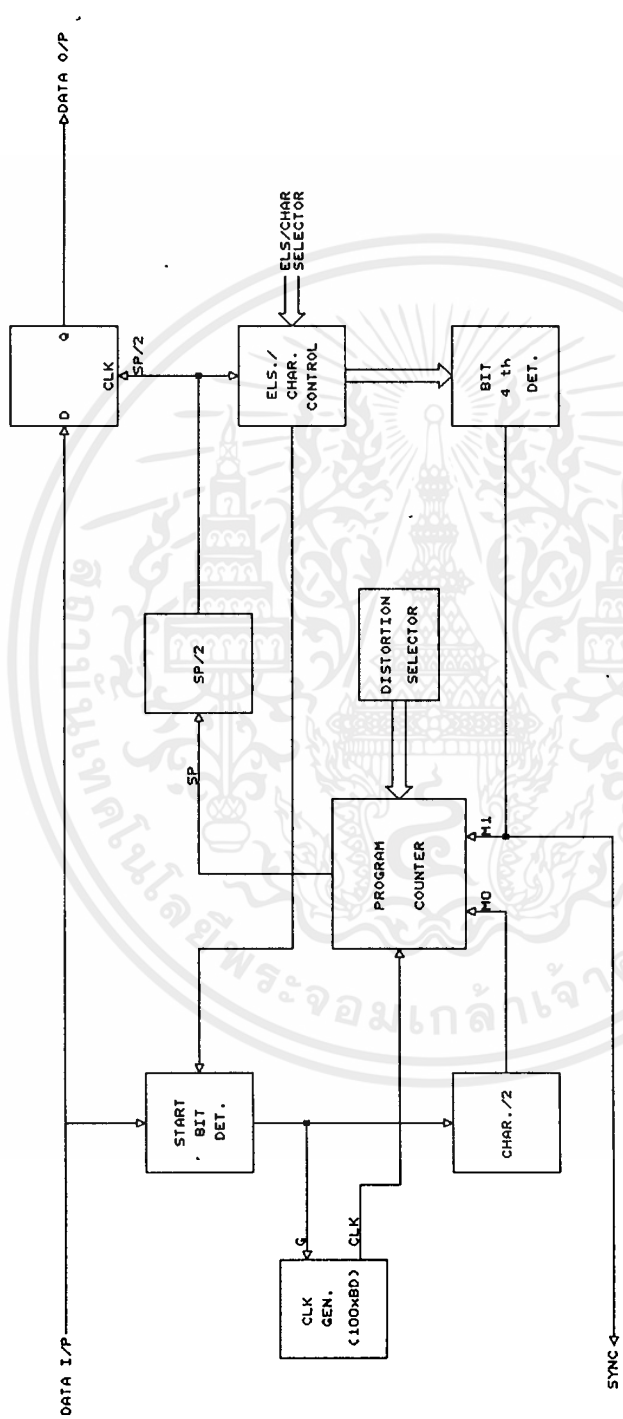
จาก BLOCK Diagram

- Start Bit Detector ทำหน้าที่ตรวจสอบ Start bit โดยปกติ OUTPUT จะเป็น LOW เมื่อตรวจพบ Start Bit จะให้ OUTPUT HIGH

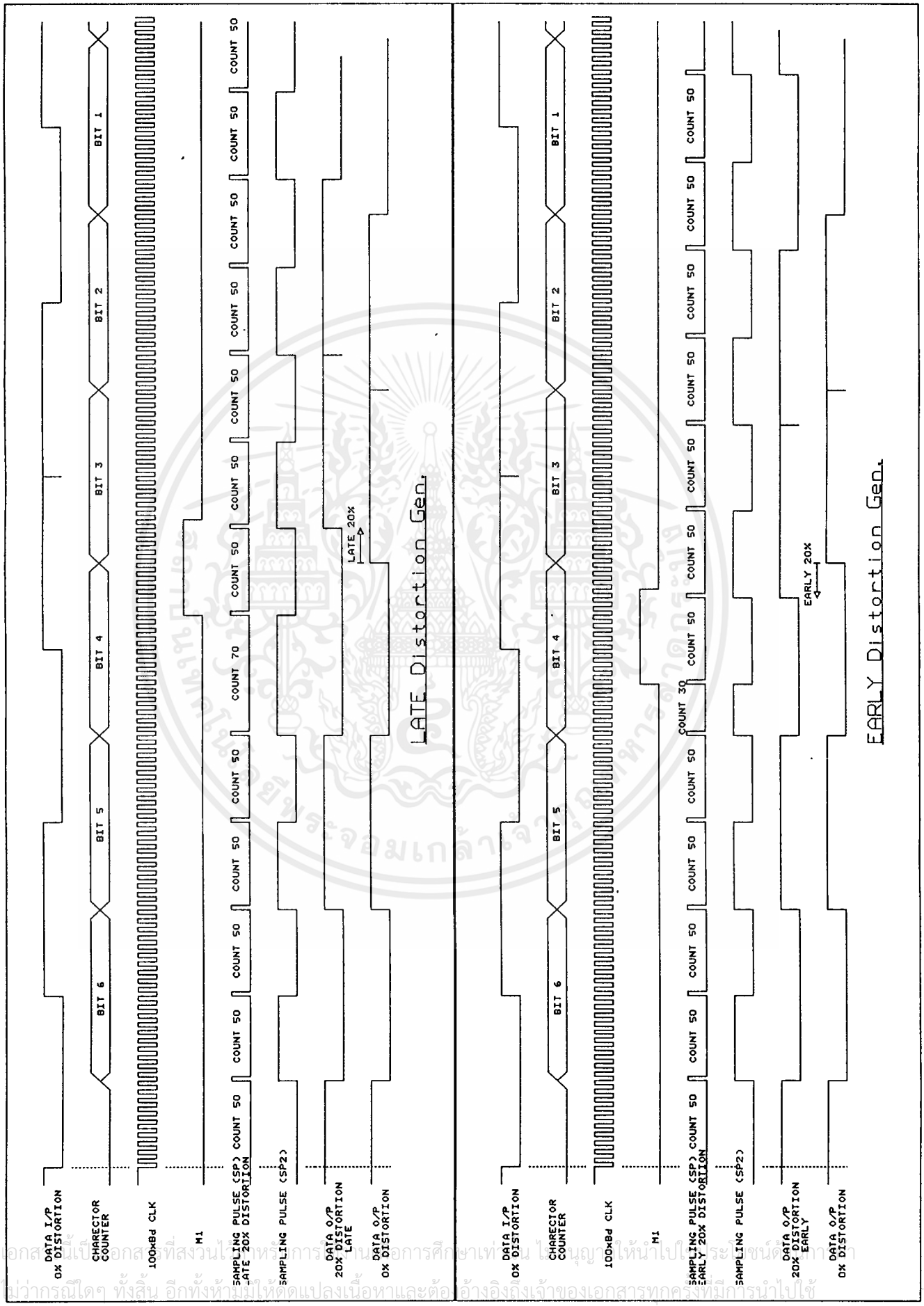
- CLK 100xBd มีหน้าที่สร้างความถี่มีค่าเป็น 100 เท่าของ Bit Rate โดยมีขา Enable ควบคุม

- ELS/Char.Control เป็นตัวกำหนดจำนวน Element ของการส่งในแต่ละ Charector

- Progran Counter ทำหน้าที่สร้างสัญญาณ Sampling เพื่อให้ได้ % Distortion ตามต้องการ โดยจะทำการนับ 50 สำหรับ 0% Distortion หรือขณะที่ M_1 เป็น "0" ขณะที่ M_1 เป็น "1" และ M_0 เป็น "0" จะนับ 50-Distortion sel. ถ้า M_0 เป็น "1" จะนับ 50+ Distortion sel. ตัวอย่างเช่น Distortion Sel. = 3% และเมื่อ M_1 = "1"



Distortion Generator Block Diagram



ขณะที่ M_0 = "0" Counter จะนับ (50-3) = 47

" = "1" " " (50+3) = 53

- Bit 4 th Det. จะตรวจหา Element (Bit) ที่ 4 โดยเริ่มนับจาก stop Bit นับย้อนกลับมา ซึ่งเป็นตำแหน่งที่สร้าง Distortion ขึ้นตามที่ตั้งค่าไว้

การทำงานของวงจร

- U_{16B} จะทำหน้าที่เป็น Start Bit Detector เมื่อพบ Start Bit จะทำการ Enable ให้กับวงจร CLOCK (100xBd), วงจร ELS/Char.Control (U_{11}) วงจร Counter (U_{19}) จะเริ่มนับเมื่อนับถึง 50 วงจร Program Counter (U_{10} และ U_{14}) จะให้ Sampling Pulse ออกมาผ่าน U_{17B} ไปที่วงจร Sampling Data (U_{16A}) เพื่อนำ Data Input ผ่านไปยัง output สัญญาณ Sampling ส่วนหนึ่งจะถูกนำไป Reset Counter (U_{19})

ให้เริ่มนับใหม่อีกครั้งจนถึง 50 ก็จะได้ Sampling อีก ในขณะเดียวกัน WLS/Char. Control ก็จะเริ่ม Count Down ตาม Sampling Pulse ที่ได้ในแต่ละครั้ง การนับที่ 50 จะนับไปเรื่อยๆ จนกระทั่งถึง Bit ที่ 4 การนับขึ้นอยู่กับค่า Distortion ที่เลือกไว้ โดยค่าที่นับจะได้จากการบวกหรือลบค่า Distortion ที่ตั้งไว้กับ 50 จึงจะให้ Sampling Pulse ออกมา และ Bit ต่อๆ ไป ก็จะนับ 50 อีกครั้ง ไปเรื่อยๆ จนกว่า วงจร ELS/Char. Control จะนับถึงศูนย์จึงจะทำการ Reset วงจรต่างๆ เพื่อเตรียมรับ Start Bit ของ charector ตัวต่อไป

- การบวกหรือลบ 50 กับค่า Distortion ที่ตั้งไว้ก็คือการทำให้ Data Output ที่ได้มี Distortion เกิดขึ้นในทาง Late และ Early ตามลำดับ ในที่นี้การทำ Late และ Early จำทำในลักษณะ Charector ที่เป็น Late และ Charector ที่เป็น Early สลับกัน โดยวงจร Char./2 (U_{17A}) ลักษณะการสร้าง Distortion ดูได้ดัง timing Diagram รูปที่

ภาค Frequency Generator

- มีหน้าที่สร้างความถี่ สำหรับภาคต่างๆ ซึ่งจะสัมพันธ์กับ Bit Rate ที่ต้องการ เนื่องจากในภาคต่างๆ ต้องการความถี่ที่แตกต่างกัน เช่น 2,16,50,100 เท่าของ Bit Rate เพื่อให้ประหยัดที่สุด จึงสร้างความถี่หลักขึ้นมาโดยมีค่าเป็น 400 เท่าของ Bit Rate และใช้วงจรหารเพื่อนำไปใช้กับภาคต่างๆ

- จาก BLOCK Diagram

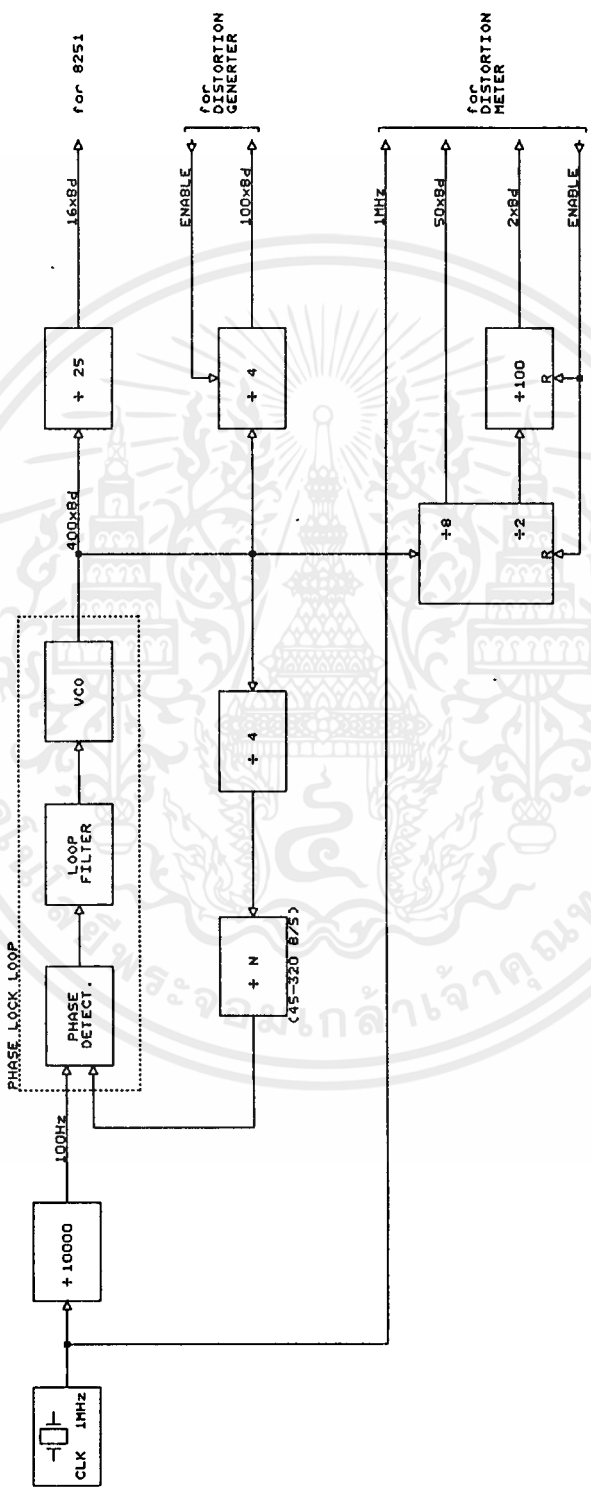
ความถี่ Reference 100 Hz สำกรับ Phase Detect ได้จากการนำ CLOCK 1 MHz มาหารด้วย 10,000 เพื่อเปรียบเทียบกับความถี่ที่ได้จากการหารในวงจร Feed Back นำผลต่างที่ได้ผ่านวงจร Loop Filter เพื่อทำให้เป็นแรงดัน DC. สำหรับควบคุมวงจรสร้างความถี่แรงดัน (Voltage Control Oscillator) หรือ VCO ความถี่ที่ได้จะเป็น 400 เท่าของ Bit Rate ส่วนหนึ่งจะผ่านวงจรหารเพื่อ Feed Back ไปยังวงจร Phase Detect วงจรหารส่วนนี้ประกอบด้วยหาร 4 และหารด้วยค่า Bit Rate ที่ต้องการ (45-320 B/S) และอีกส่วนหนึ่งก็จะนำไปผ่านวงจรหารค่าต่างๆ เพื่อนำไปใช้งาน

การทำงานของวงจร

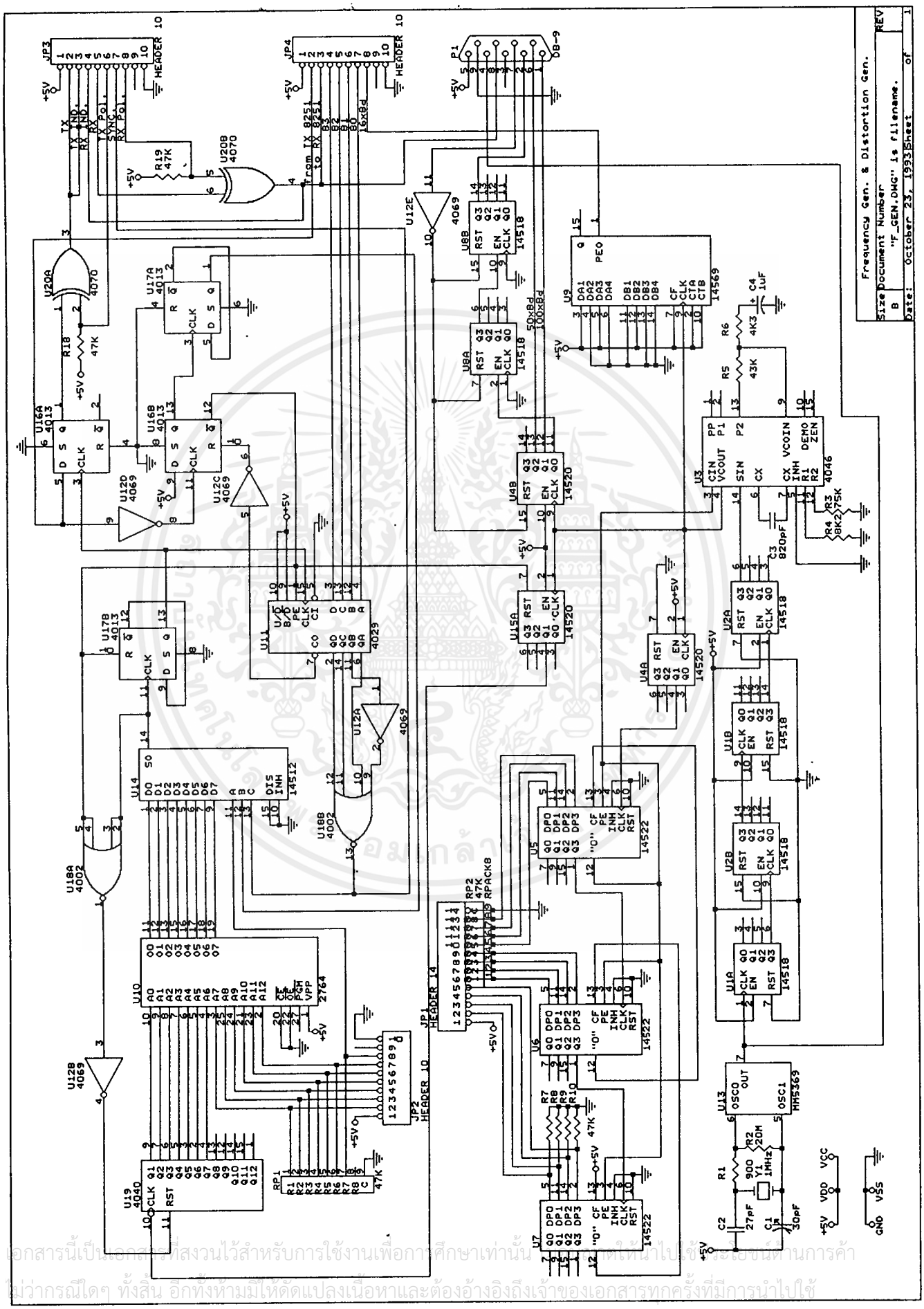
- Y_1 และ U_{13} ทำหน้าที่กำเนิด CLOCK 1 MHz นำมาหาร 10,000 โดย U_{1A},U_{2A} ,U_{2A},U_{2B} ได้ความถี่เท่ากับ 100 Hz เพื่อเป็นความถี่ Reference ให้กับ Phase Detector (U_3) ในการเปรียบเทียบกับความถี่จากวงจร Feed Back ผลต่างที่ได้ที่ P_2 ของ U_3 นำมาเปลี่ยนให้เป็นแรงดัน DC โดยผ่าน RC Loop Filter เพื่อควบคุมวงจร VCO ใน U_3

- วงจร Feed Back ประกอบด้วย U_{4A} ทำหน้าที่หาร 4 และ U_5,U_6,U_7 เป็น Programmable Divide By-N-BCD Counter ต่อ Cascade กันเป็นหลักหน่วย,สิบ,ร้อย ตามลำดับ สามารถเลือกค่าที่จะหารโดยผ่าน Trumb Wheel Switch ซึ่งค่าที่เลือกจะเป็นการเลือก speedl มีหน่วยเป็น Bit/Sec. ดังนั้นความถี่จาก output ของ VCO จะต้องมีค่าเป็น 400 เท่าของ Bit Rate และเมื่อผ่านวงจร Feed Back แล้วจึงจะเหลือ 100 Hz

- จากความถี่หลักที่ได้จาก VCO เป็น 400 เท่าของ Bit Rate เพื่อให้ได้ความถี่ตามที่ต้องการสำหรับภาคต่างๆ ดังนี้



FREQUENCY GENERATER



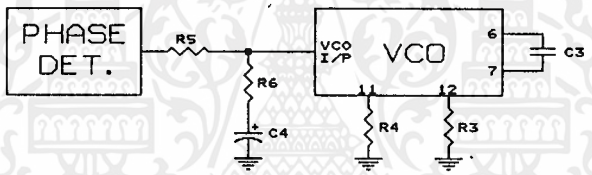
1) สำหรับ Communication Port ทำการหารด้วย 25 (U_9) จะได้ความถี่เป็น 16 เท่า ของ Bit Rate

2) Distortion Generator จะหารด้วย 4(U_{15A}) ได้ความถี่เป็น 100 เท่าของ Bit Rate

3) Distortion Meter หารด้วย 8 (U_{4B}) และหารด้วย 200 (U_{4B},U_{8A},U_{8B}) ได้ความถี่เป็น 50 และ 2 เท่า ของ Bit Rate ตามลำดับ

การคำนวณค่า RC ที่ต่อกับวงจร PLL (4046)

- จากที่กำหนดให้สามารถเลือก Bit Rate ได้ตั้งแต่ 45-320 B/S ดังนั้นความถี่ที่ VCO จะต้องผลิตเป็น 400 เท่าของ Bit Rate จะได้ = 10 KHz ถึง 128 KHz



กำหนด - VCO Range = 10KHz - 180 KHz

- C_3 = 820 pF

จากสูตร $f_{min} = \frac{1}{R_3 (C_3+32pF)}$

$$10\ KHz = \frac{1}{R_3 (820pF+32pF)}$$

$$R_3 = 117,370$$

$$\sim 120\ K$$

$$f_{max} = \frac{1}{R_4\ (820pF+32pF)} + f_{min}$$

$$180\ KHz-10\ KHz = \frac{1}{R_4\ (820pF+32pF)}$$

$$R_4 = 6904$$

$$\sim 6.8\ K$$

- RC LOOP Filter

กำหนด $C_4 = 1uF$, $f = 180\ KHz - 10\ KHz = 170\ KHz$

$N = 134$

$$R_6C_4 = \frac{6N}{f_{max}} - \frac{N}{2\ f}$$

$$= 4341$$

$$\sim 4.3\ K$$

$$R_5 = R_6 / 0.1$$

$$= 4.3k / 0.1$$

$$= 43\ K$$

ภาคผนวก

ข้อมูลทางเทคนิค

1) Distortion meter

1.1 วัด Distortion ได้ 0-40% ทั้งทางด้าน Late และ Early โดยแสดงความละเอียดจุดละ 20%

1.2 เลือกจำนวน Elements/Charaector ได้ 4 แบบ คือ 5,6,7 และ 8 ELS/Char

1.3 มี switch สำหรับสลับขั้วของสัญญาณที่รับเข้ามา

1.4 เลือก Speed ได้ ตั้งแต่ 45-320 Bit/second

2) Generator

2.1 เลือกข้อมูลที่ใช้ในการทดสอบได้ 5 แบบ

2.1.1 แบบ A จะส่งข้อมูลตาม switch หมายเลข 1-8 ทั้งนี้ขึ้นอยู่กับการเลือก ELS /Char. เช่นเลือก 5 ELS/Char. ก็จะส่งข้อมูลตาม switch หมายเลข 1-5 โดยเริ่มจากหมายเลข 1 ก่อนเสมอ

2.1.2 แบบ B ส่งข้อมูลแบบ Binary Sequence โดยเริ่มจาก (00000000) ไปจนถึง (11111111) แล้วกลับไปเริ่มต้นอีก ทั้งนี้จำนวน Bit ที่ส่งขึ้นอยู่กับการเลือกจำนวน ELS/Char. ด้วย

2.1.3 แบบ C เป็นข้อมูลแบบ RY คือจะส่งข้อมูล (10101010) และ (01010101) สลับกันไป จำนวน Bit ที่ส่งขึ้นอยู่กับการเลือกจำนวน ELS/Char.

2.1.4 แบบ D เป็นข้อมูล "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG" เป็นข้อมูลมาตรฐานใน ITA NO.2 ซึ่งตรงกับ CCITT REC. R52 โดยเป็นข้อมูลแบบ 5 ELS /Char.

2.1.5 แบบ E เป็นข้อมูล "FACULTY OF ENGINEERING OF (KMITL)" เป็นข้อมูลแบบ 5 ELS/Char.

2.2 เลือกจำนวน Elements/Charector ได้ 4 แบบ คือ 5,6,7 และ 8 ELS/Char.

2.3 เลือกขนาดของ STOP Pulse ได้ 3 แบบ คือ 1,1.5 และ 2 Bit

2.4 เลือก Parity ได้ 3 แบบ คือ ODD, EVEN และ MR/SP (non)

2.5 มี Switch สำหรับสลับขั้วสัญญาณที่จะส่งออกไป

2.6 เปลี่ยนค่า Distortion ของข้อมูลที่จะส่งออกไปได้ตั้งแต่ 0-50%

2.7 เลือก Speed ได้ตั้งแต่ 45-320 B/S

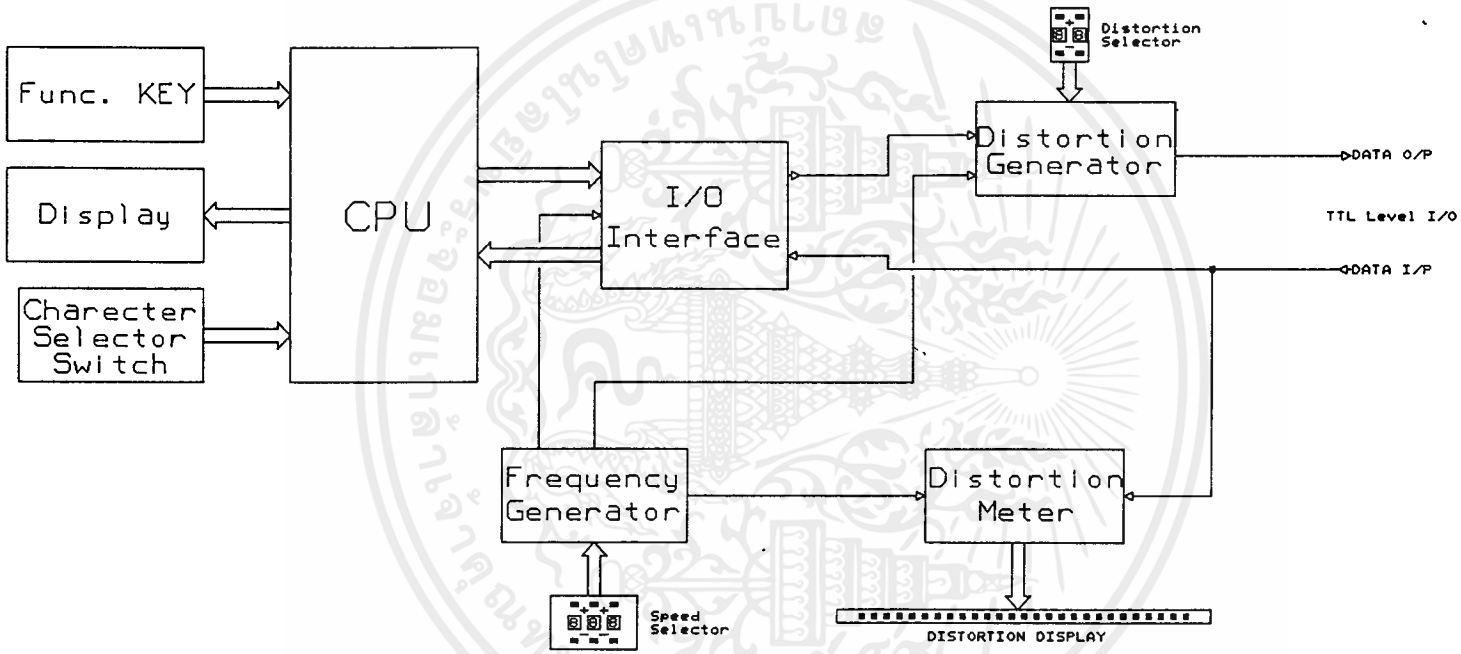
2.8 สามารถวัด Error Rate ได้ เมื่อ switch BER อยู่ตำแหน่ง ON โดยจะมี Counter นับบอกจำนวน char. ที่ส่งออกไปและบอกจำนวน char. ที่ผิดพลาด

3) Monitor

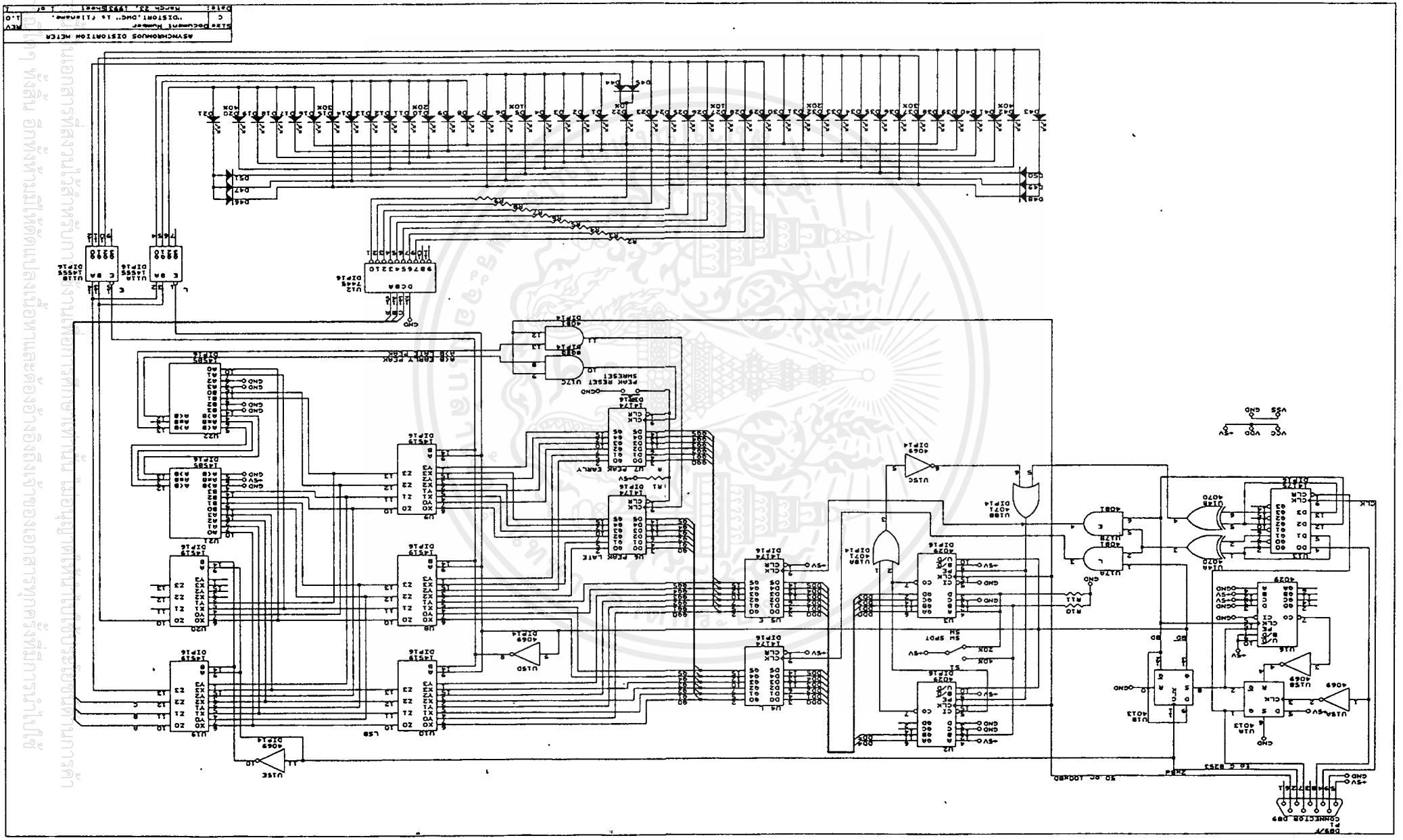
3.1 แสดง char. ที่รับได้ออกทางจอ LCD (เฉพาะแบบ 5 unit code (5 ELS/char.)

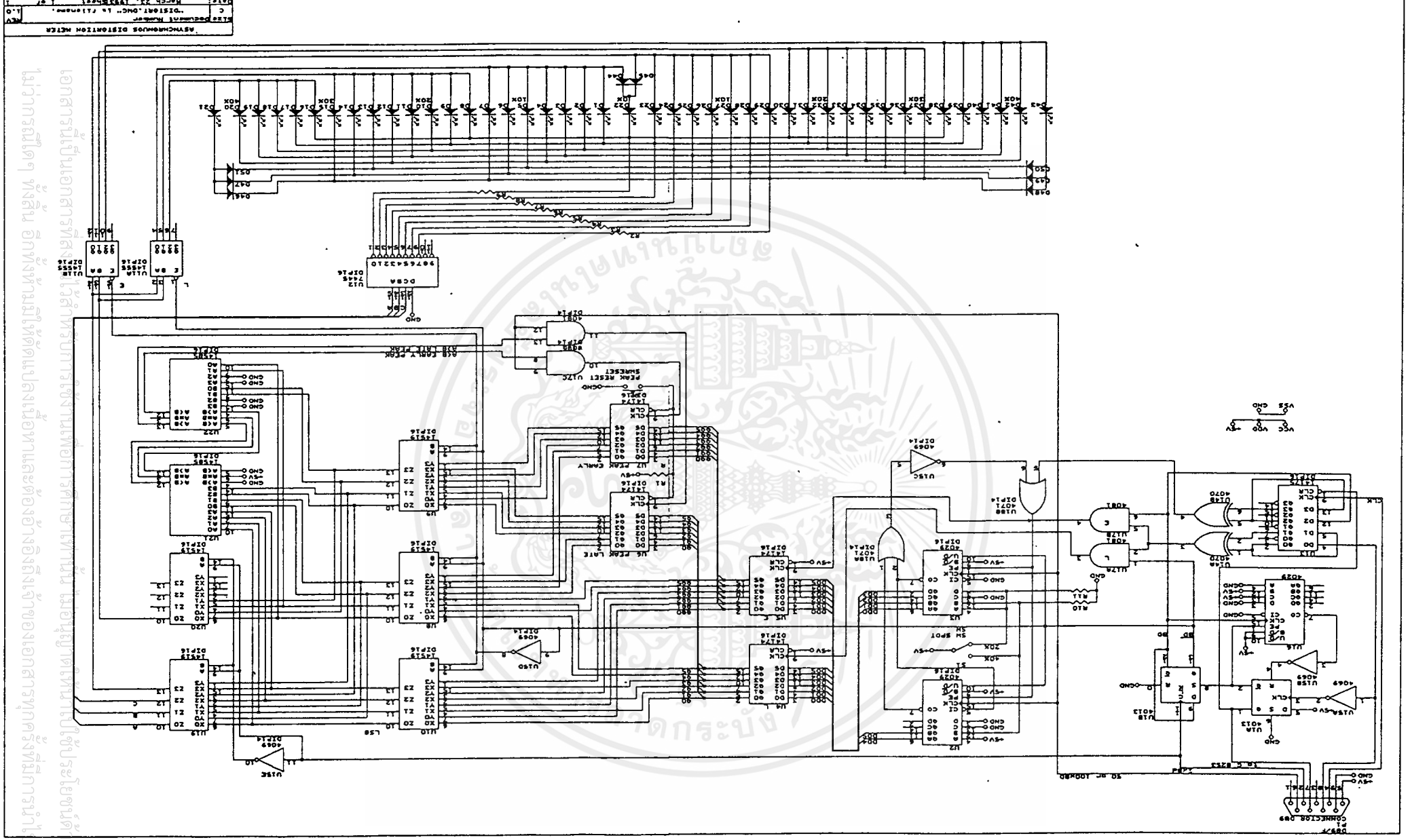
3.2 แสดง char. ที่รับได้พร้อมทั้งบรรทัดล่างเป็น Counter แสดงจำนวน Char. ที่ส่งออกไปและจำนวน Char. ที่ผิด

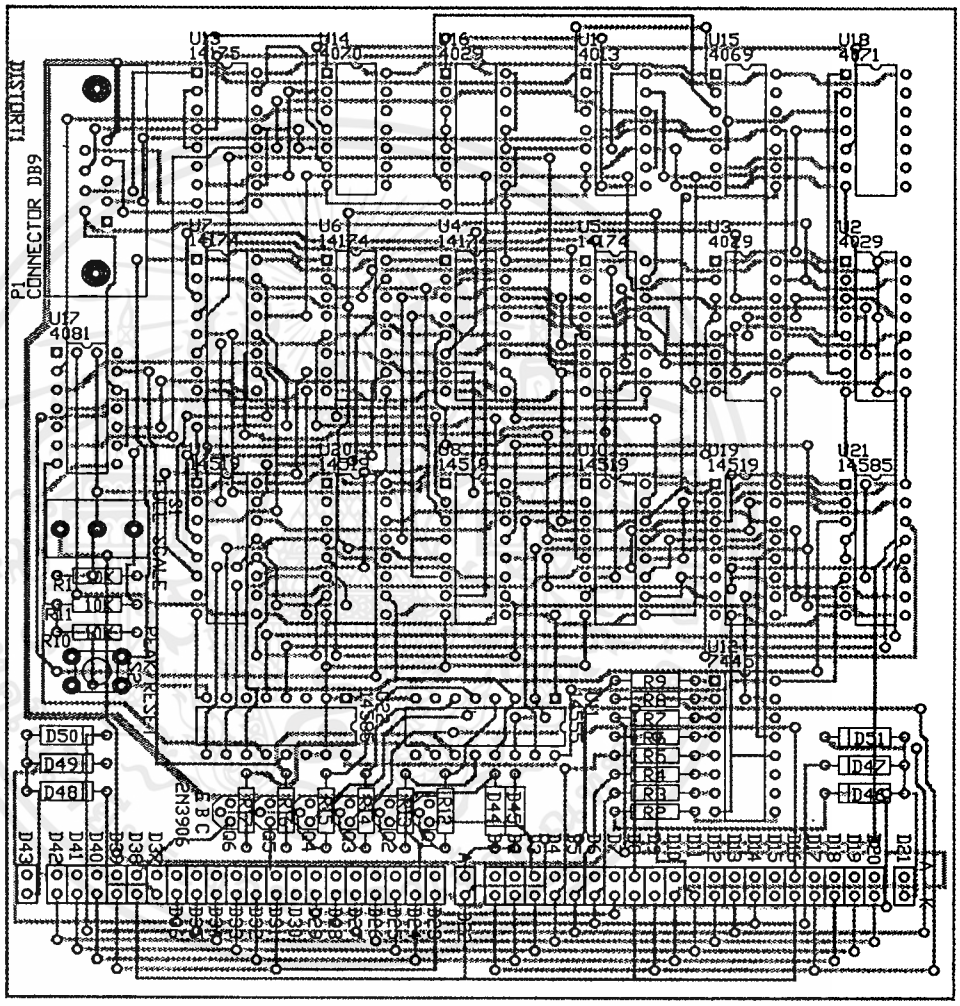
****

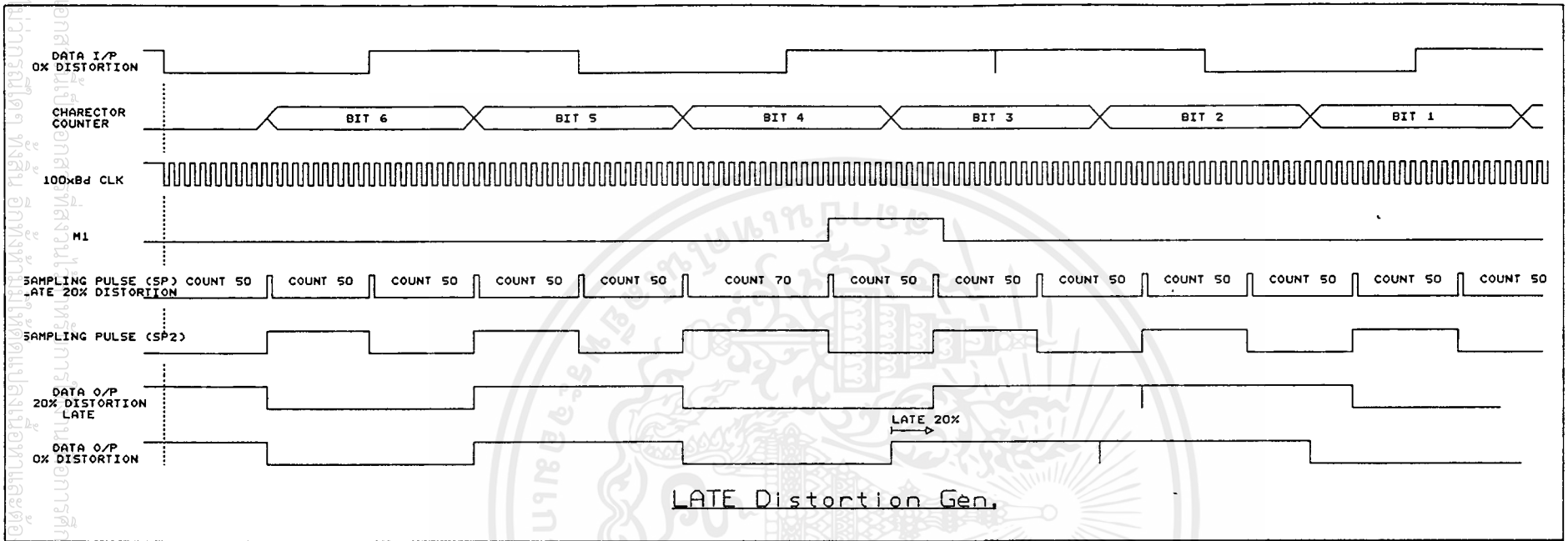


BER & Distortion Measuring for Asynchonous comm.

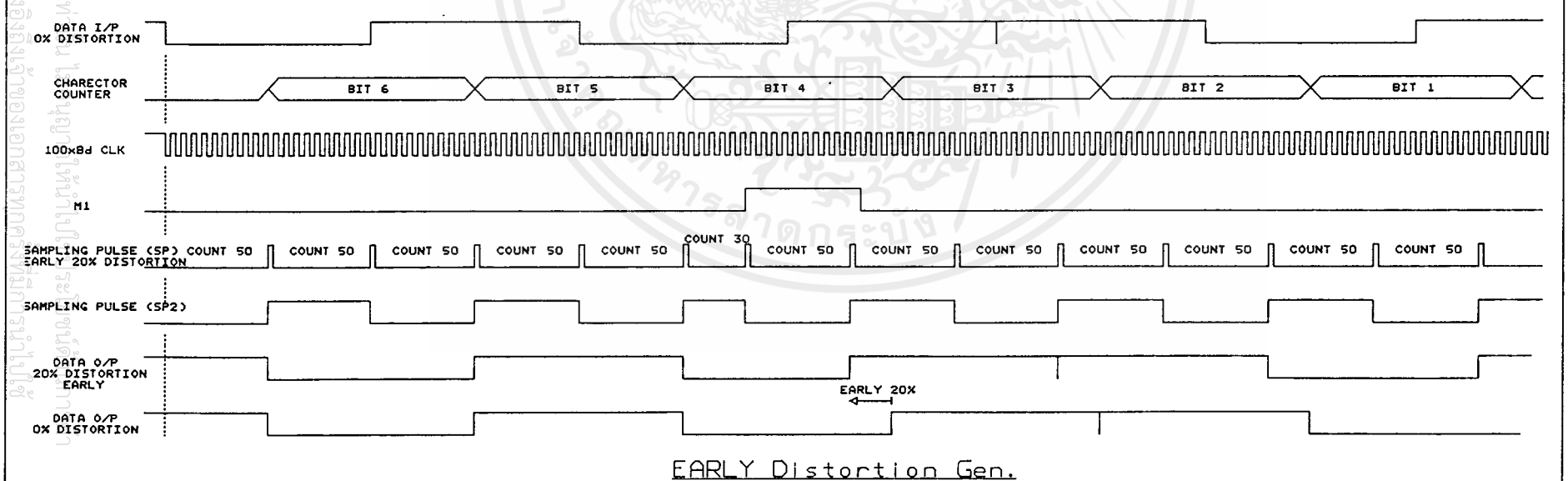

ASYNCHRONOUS DISTORTION METER
Size Document Number REV
C "DISTORT.DWG" 1.0
Date: March 25, 1993 Sheet 1 of 1
CONNECTOR DB9
SW SPDT
PEAK RESET
PEAK EARLY
PEAK LATE
GND
+5V



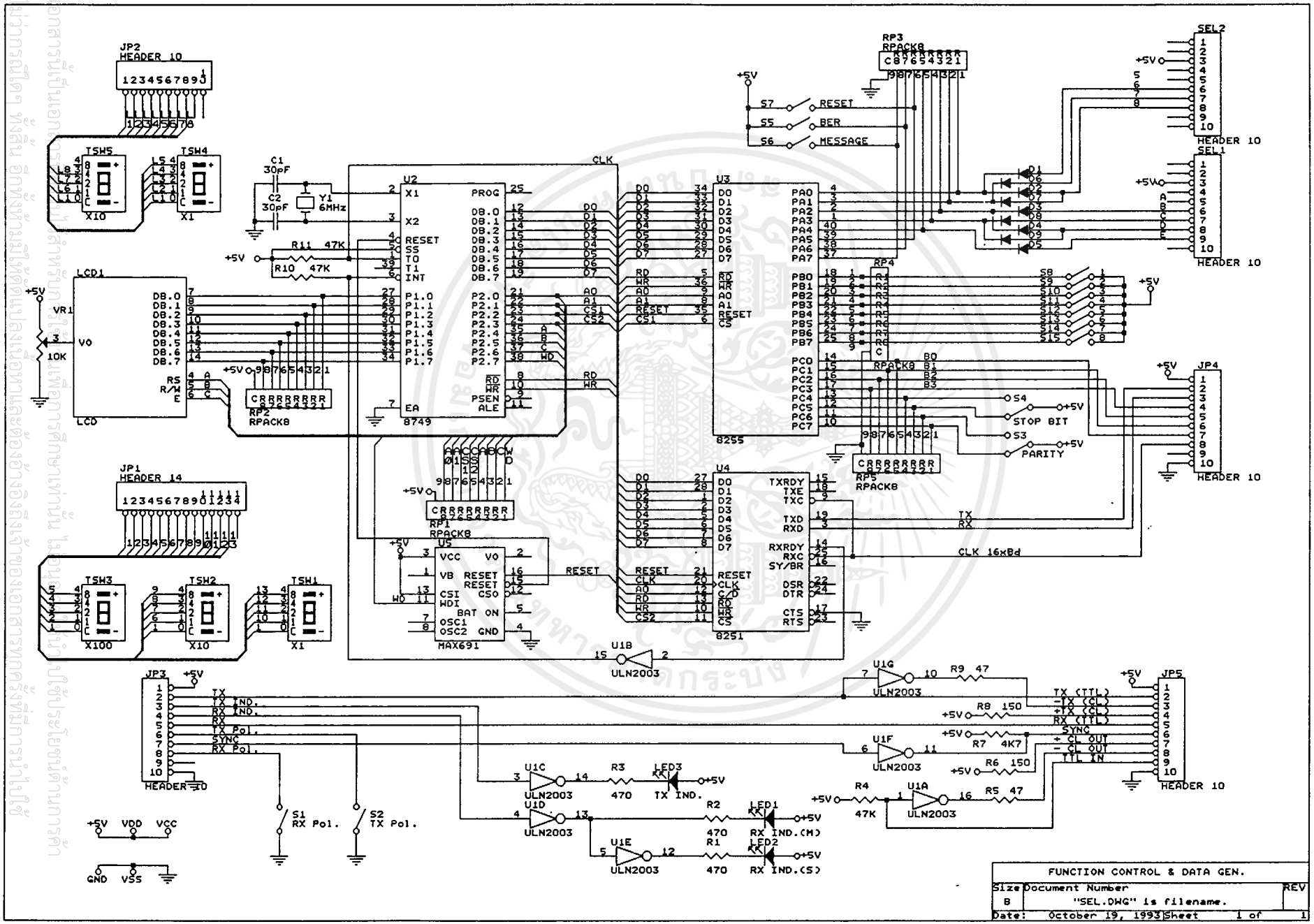


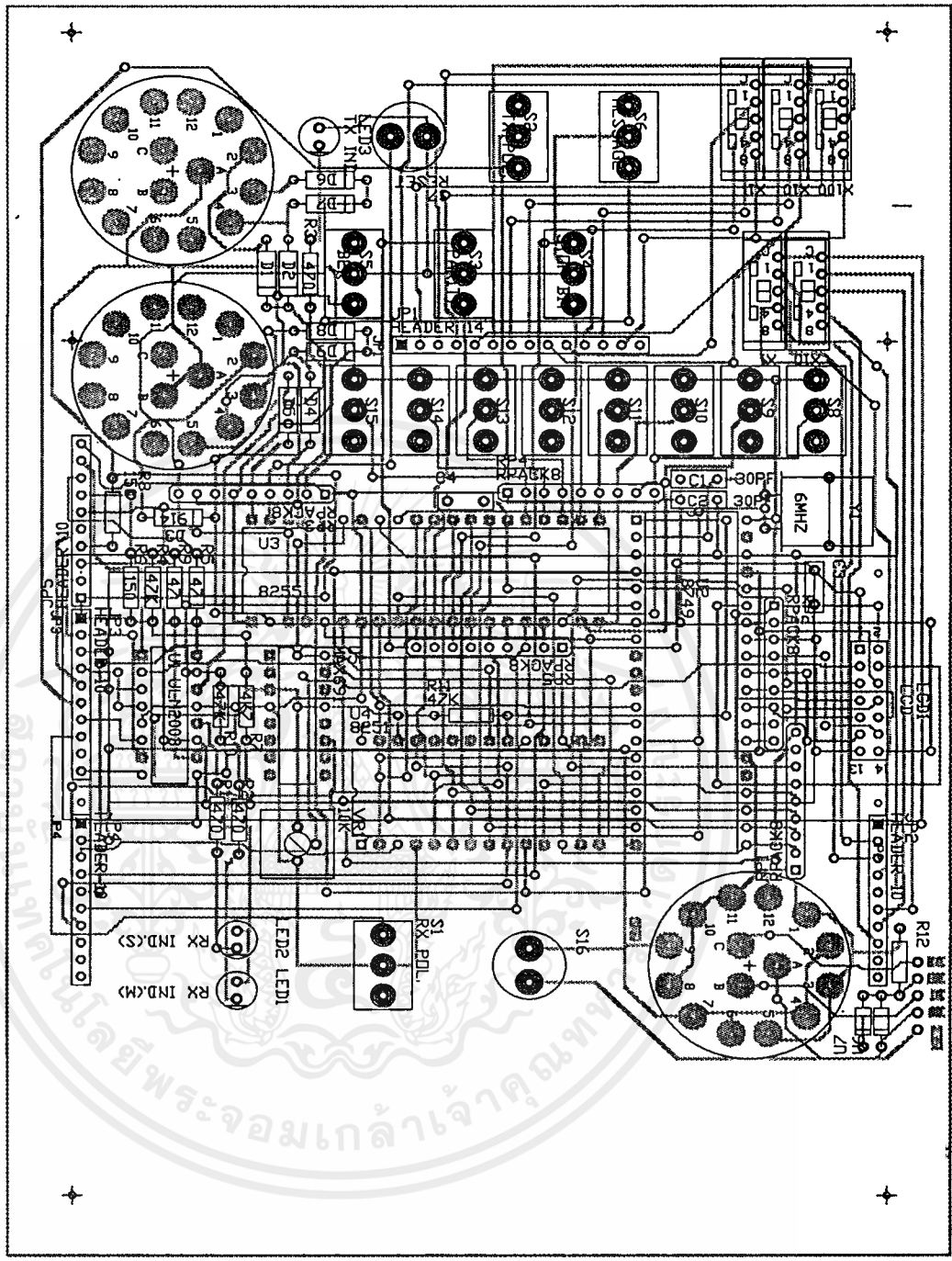


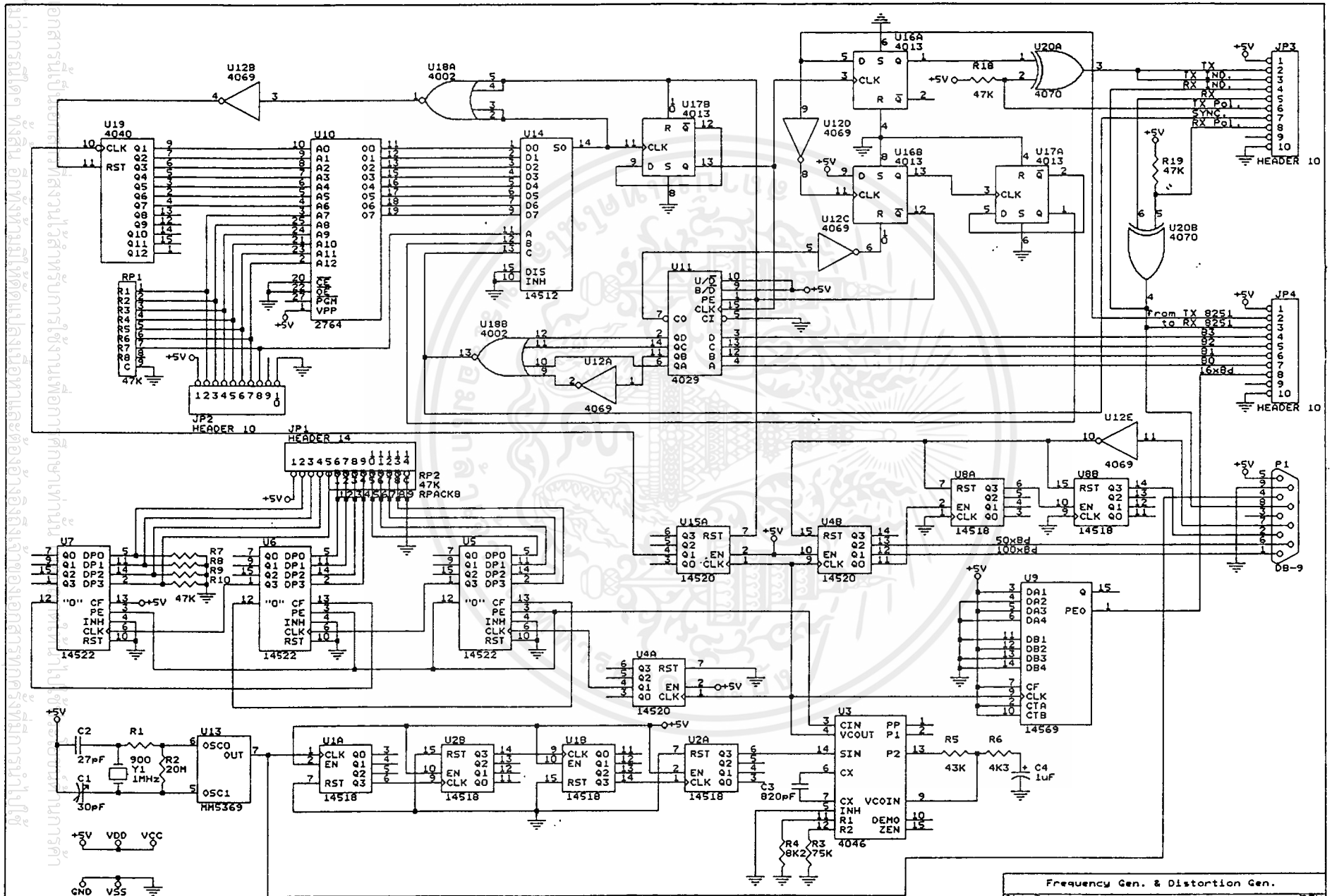
LATE Distortion Gen.

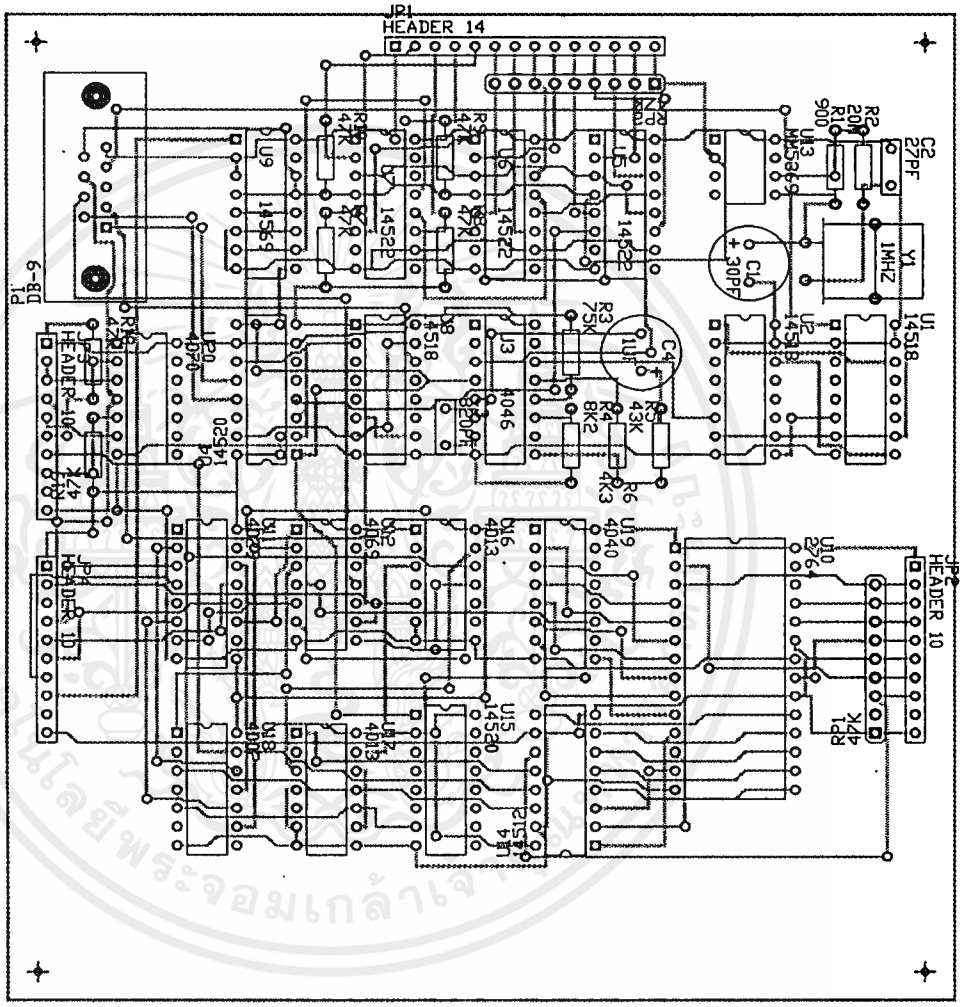


EARLY Distortion Gen.


JP2
HEADER 10
TSW5
TSW4
X10
X1
C1
30pF
C2
30pF
Y1
6MHz
R11 47K
R10 47K
+5V
LCD1
VR1
10K
LCD
RP2
RPACK8
U2
8749
CLK
RESET
U3
8255
RP3
RPACK8
SEL2
SEL1
HEADER 10
S7 RESET
S5 BER
S6 MESSAGE
RP4
S8
S4 STOP BIT
S3 PARITY
JP4
RP5
RPACK8
JP1
HEADER 14
TSW3
TSW2
TSW1
X100
X10
X1
RP1
RPACK8
U5
MAX691
U4
8251
TX
RX
CLK 16xBd
U1B
ULN2003
U1G
ULN2003
R9 47
JP3
HEADER 10
JP5
HEADER 10
TX IND.
RX IND.
TX Pol.
SYNC
RX Pol.
S1
RX Pol.
S2
TX Pol.
U1C
ULN2003
R3
470
LED3
TX IND.
U1D
ULN2003
R2
470
LED1
RX IND.(M)
U1E
ULN2003
R1
470
LED2
RX IND.(S)
R8 150
R7 4K7
R6 150
R4
47K
U1A
ULN2003
R5 47
U1F
ULN2003
TX (TTL)
-TX (CL)
+TX (CL)
RX (TTL)
SYNC
- CL OUT
+ CL OUT
TTL IN
VDD
VCC
GND
VSS
FUNCTION CONTROL & DATA GEN.
Size B
Document Number
"SEL.DWG" is filename.
REV
Date: October 19, 1993
Sheet 1 of 1




U12B 4069
U18A 4002
U19 4040
U10 2764
U14 14512
U17B 4013
U16A 4013
U20A 4070
U120 4069
U16B 4013
U17A 4013
U12C 4069
U20B 4070
R18 47K
R19 47K
JP3
HEADER 10
JP4
HEADER 10
TX
TX GND.
RX GND.
RX
TX Pol.
SYNC.
RX Pol.
from TX 8251
to RX 8251
B3
B2
B1
B0
16x8d
RP1
47K
JP2
HEADER 10
U11 4029
U18B 4002
U12A 4069
JP1
HEADER 14
RP2 47K RPACK8
U12E 4069
U8A 14518
U8B 14518
P1
DB-9
50x8d
100x8d
U15A 14520
U4B 14520
U7 14522
U6 14522
U5 14522
U4A 14520
U9 14569
U3 4046
U1A 14518
U2B 14518
U1B 14518
U2A 14518
U13 MM5369
C2 27pF
R1 900
R2 20M
Y1 1MHz
C1 30pF
R5 43K
R6 4K3
C4 1uF
C3 820pF
R4 8K2
R3 75K
+5V VDD VCC
GND VSS
Frequency Gen. & Distortion Gen.
Size B
Document Number "F_GEN.DHG" is filename.
REV
Date: October 23, 1993 Sheet of 1



ข้อเสนอแนะของ CCITT

Recommendation

R.51 Standardized text for Distortion Testing of the Code-Independent Element of a Complete Circuit

"(Letter Shift) S (Carriage Return)(Line-feed) Q (Figure Shift) (space) 9" เป็นแบบ 6 Unit Code

R.52 Standardized of International Text for The Measurement of the margin of start-stop equipment

"THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"

R.53 50 Baud 120 Hz. VFT channel Distortion ไม่เกิน 10%

R.54 50 Baud 5 unit Start-Stop ผ่าน least line, over land line และ Submamine Cable ผิดไม่เกิน 3 ตัวอักษรในการส่ง 100,000 ตัวอักษร

R.57 สัญญาณโทรพิมพ์ที่ผ่าน Switched Network สำหรับ 50 Baud

- เมื่อผ่าน VF1 1 Channel มี Distortion ไม่เกิน 10%
- " 2 " " " " 18%
- " 3 " " " " 24%
- " 4 " " " " 28%

R.120 75 Baud มี Distortion ไม่เกิน 28%

100 " " " 24%

200 " " " 32%

```
                ;       ************************************************************
                ;       *  BER & DISTORTION METER for ASYNCHRONOUS COMMUNICATION   *
                ;       *  FILE NAME   : BER.ASM                                   *
                ;       *  DATE        :  27/07/36                                 *
                ;       ************************************************************
                ;
0000                                    CPU   "8048.TBL"
0000                                    HOF   "BIN8"
                ;
000C =              CLR:                EQU   0CH           ; CHIP SELCT = "1"
00F8 =              MODE:               EQU   0F8H          ; CS1 = 0 ,A0,A0 = 0
0000 =              PA:                 EQU   00H           ; PORT A
0001 =              PB:                 EQU   01H           ; PORT B
0002 =              PC:                 EQU   02H           ; PORT C
0003 =              CONP:               EQU   03H           ; CON PORT
00F4 =              MOCOM:              EQU   0F4H          ; CS2 = 0
0001 =              COMCON:             EQU   01H           ; C = 1
0000 =              COMDATA:            EQU   00H           ; C = 0
0050 =              DEDATA:             EQU   50H           ; DATA DELAY
                ;
0040 =              DATAD:              EQU   40H           ; DATA TO DISPLAY 20*2
0060 =              DPORTA:             EQU   60H           ; DATA PORT A
0062 =              DPORTB:             EQU   62H           ; DATA PORT B
0064 =              DPORTC:             EQU   64H           ; DATA PORT C
0066 =              DCOUNT:             EQU   66H
0068 =              CUR:                EQU   68H
006A =              DFOX:               EQU   6AH
006C =              ECOU1:              EQU   6CH           ; ERROR COUNT
006D =              ECOU2:              EQU   6DH
006F =              ERCOU1:             EQU   6FH           ; ERROR COUNT
0070 =              ERCOU2:             EQU   70H
0074 =              DARS:               EQU   74H
0075 =              COUNT:              EQU   75H
0076 =              DATAB:              EQU   76H
0077 =              DATARY:             EQU   77H

                ;MAIN PROGRAM
0000                                    ORG    000H
                ;
0000 040A                               JMP   START
0002 00             NUMBERS:            NOP
0003 A483                               JMP   INTLINE
0005 00                                 NOP
0006 00                                 NOP
0007 A483                               JMP   INTTIMER
0009 00                                 NOP
                ;INIT 8255 PA-O,PB-O,PC0-3-O,PC4-7-I
000A 00             START:              NOP
000B 00                                 NOP
000C 75                                 ENTO CLK
000D C5                                 SEL   RB0
000E E5                                 SEL   MB0
000F 15                                 DIS   I
0010 27                                 CLR   A
0011 8A80                               ORL   P2,#80H
0013 00                                 NOP
0014 00                                 NOP
0015 9A7F                               ANL   P2,#7FH
                ;CLEAR MEMORY  ADDRESS   60H - 70H
```

```
0017 340E                       CALL CLRMEN

                ;
0019 2300                       MOV  A,#00H
001B 39                         OUTL P1,A
001C 230C                       MOV  A,#0CH
001E 3A                         OUTL P2,A
                ;
001F 9AF8                       ANL  P2,#MODE       ; DATA DIGIT  PORT
0021 8A03                       ORL  P2,#CONP
0023 239A                       MOV  A,#9AH         ; INPUT ALL
0025 02                         OUTL BUS,A
0026 00                         NOP
0027 8A0C                       ORL  P2,#CLR

0029 8A80                       ORL  P2,#80H
002B 00                         NOP
002C 00                         NOP
002D 9A7F                       ANL  P2,#7FH
                ;
002F 3470                       CALL INITLCD
                ;                CALL TESTMEM
                ;
                ;   MAIN
0031                            MAIN:
0031 9AF8                       ANL  P2,#MODE       ; DATA DIGIT  PORT
0033 8A02                       ORL  P2,#PC
0035 2355                       MOV  A,#55H         ; INPUT ALL
0037 02                         OUTL BUS,A
0038 00                         NOP
0039 8A0C                       ORL  P2,#CLR
003B 8A80               TE:     ORL  P2,#80H
003D 00                         NOP
003E 00                         NOP
003F 9A7F                       ANL  P2,#7FH
                ;
0041 14EC                       CALL INPORT
0043 9400                       CALL SETCOM
0045 B860               MODECOM: MOV  R0,#DPORTA
0047 F0                         MOV  A,@R0
0048 77                         RR   A
0049 77                         RR   A
004A 5307                       ANL  A,#07H
004C C667                       JZ   MODE1
004E 07                         DEC  A
004F C663                       JZ   MODE2
0051 07                         DEC  A
0052 C65F                       JZ   MODE3
0054 07                         DEC  A
0055 C65B                       JZ   MODE4
0057 B46E                       CALL MODEE
0059 0469                       JMP  MODECOE
005B B459               MODE4:  CALL MODED
005D 0469                       JMP  MODECOE
005F B43E               MODE3:  CALL MODEC
0061 0469                       JMP  MODECOE
0063 B424               MODE2:  CALL MODEB
0065 0469                       JMP  MODECOE
0067 B40B               MODE1:  CALL MODEA
```

```
0069 043B          MODECOE:       JMP  TE

                   ;    TE1:           JMP   TE
                   ;***********************************************
                   ;*   ROUTINE RECEIVER DATA 1 BYTE FROM COM      *
                   ;*   I/P REG : RXD R2                          *
                   ;*   O/P     : R2 (DATA)                       *
                   ;***********************************************
006B 9AF4          RXCOM1:        ANL  P2,#MOCOM      ; DATA DIGIT  PORT
006D 8A01                         ORL  P2,#COMCON
006F 00                           NOP
0070 08                           INS  A,BUS
0071 8A0C                         ORL  P2,#CLR
0073 727B                         JB3  RX_C1
0075 B27F                         JB5  RX_C12
0077 3293                         JB1  RX_C2
0079 04D9                         JMP  RX_C24
007B B86C          RX_C1:         MOV  R0,#ECOU1
007D 3431                         CALL INCDEC
007F 9AF4          RX_C12:        ANL  P2,#MOCOM      ; DATA DIGIT  PORT
0081 8A01                         ORL  P2,#COMCON
0083 2315                         MOV  A,#15H         ; TxEN,RxEN,ER
0085 02                           OUTL BUS,A
0086 00                           NOP
0087 8A0C                         ORL  P2,#CLR
0089 9AF4                         ANL  P2,#MOCOM      ; DATA DIGIT  PORT
008B 8A00                         ORL  P2,#COMDATA
008D 00                           NOP
008E 08                           INS  A,BUS
008F 8A0C                         ORL  P2,#CLR
0091 04EB          RX_C13:        JMP  RXCOME
0093 B875          RX_C2:         MOV  R0,#COUNT
0095 2330                         MOV  A,#30H
0097 A0                           MOV  @R0,A
0098 9AF4                         ANL  P2,#MOCOM      ; DATA DIGIT  PORT
009A 8A00                         ORL  P2,#COMDATA
009C 00                           NOP
009D 08                           INS  A,BUS
009E 8A0C                         ORL  P2,#CLR
00A0 AD                           MOV  R5,A
00A1 B874                         MOV  R0,#DARS
00A3 F0                           MOV  A,@R0
00A4 DD                           XRL  A,R5
00A5 C6AB                         JZ   RX_C20
00A7 B86C                         MOV  R0,#ECOU1
00A9 3431                         CALL INCDEC
00AB B868          RX_C20:        MOV  R0,#CUR
00AD B960                         MOV  R1,#DPORTA
00AF F1                           MOV  A,@R1
00B0 D2C8                         JB6  RX_C22          ; BER ON
00B2 F0                           MOV  A,@R0
00B3 D314                         XRL  A,#20
00B5 96BA                         JNZ  RX_C21
00B7 2340                         MOV  A,#40H
00B9 A0                           MOV  @R0,A
00BA F0            RX_C21:        MOV  A,@R0
00BB 34B4                         CALL GOTO
00BD 10                           INC  @R0
00BE F0                           MOV  A,@R0
```

```
00BF D355                       XRL  A,#40H+21
00C1 96D4                       JNZ  RX_C23
00C3 2300                       MOV  A,#00H
00C5 A0                         MOV  @R0,A
00C6 04D4                       JMP  RX_C23
00C8 F0             RX_C22:     MOV  A,@R0
00C9 34B4                       CALL GOTO
00CB 10                         INC  @R0
00CC F0                         MOV  A,@R0
00CD D315                       XRL  A,#21
00CF 96D4                       JNZ  RX_C23
00D1 2300                       MOV  A,#00H
00D3 A0                         MOV  @R0,A
00D4 FD             RX_C23:     MOV   A,R5
00D5 745A                       CALL  FOXCON
00D7 34BC                       CALL  WRBYTE
00D9 B960           RX_C24:     MOV  R1,#DPORTA
00DB F1                         MOV  A,@R1
00DC 37                         CPL  A
00DD D2EB                       JB6  RXCOME          ; BER OFF
00DF B86C                       MOV  R0,#ECOU1
00E1 BC42                       MOV  R4,#42H
00E3 94C8                       CALL DECASCII
00E5 B86F                       MOV  R0,#ERCOU1
00E7 BC4C                       MOV  R4,#4CH
00E9 94C8                       CALL DECASCII
00EB 83             RXCOME:     RET
                ;*************************************************
                ;*    INPUT PORT PA,PB,PC                        *
                ;*      OUTPUT     DPORTA  :  PORT A             *
                ;*                 DPORTB  :  PORT B             *
                ;*                 DPORTC  :  PORT C             *
                ;*************************************************
00EC B860           INPORT:     MOV  R0,#DPORTA
00EE 9AF8                       ANL  P2,#MODE        ; DATA DIGIT  PORT
00F0 00                         NOP
00F1 8A00                       ORL  P2,#PA
00F3 08                         INS  A,BUS
00F4 8A0C                       ORL  P2,#CLR
00F6 A0                         MOV  @R0,A           ; DATA PORT A
00F7 B862                       MOV  R0,#DPORTB
00F9 9AF8                       ANL  P2,#MODE        ; DATA DIGIT  PORT
00FB 00                         NOP
00FC 8A01                       ORL  P2,#PB
00FE 08                         INS  A,BUS
00FF 8A0C                       ORL  P2,#CLR
0101 A0                         MOV  @R0,A           ; DATA PORT B
0102 B864                       MOV  R0,#DPORTC
0104 9AF8                       ANL  P2,#MODE        ; DATA DIGIT  PORT
0106 00                         NOP
0107 8A02                       ORL  P2,#PC
0109 08                         INS  A,BUS
010A 8A0C                       ORL  P2,#CLR
010C A0                         MOV  @R0,A           ; DATA PORT C
010D 83                         RET
010E B860           CLRMEN:     MOV  R0,#60H         ;START ADDRESS TO CLEAR
0110 B920                       MOV  R1,#20H         ;COUNT TO CLEAR
0112 A0             CLRM:       MOV  @R0,A           ;CLEAR MEMORY
0113 8A80                       ORL  P2,#80H
```

```
0115 18                         INC  R0
0116 9A7F                       ANL  P2,#7FH
0118 E912                       DJNZ R1,CLRM

011A B866                       MOV  R0,#DCOUNT
011C 2300                       MOV  A,#00H
011E A0                         MOV  @R0,A
011F B868                       MOV  R0,#CUR
0121 A0                         MOV  @R0,A
0122 B86A                       MOV  R0,#DFOX
0124 A0                         MOV  @R0,A
0125 B875                       MOV  R0,#COUNT
0127 A0                         MOV  @R0,A
0128 B876                       MOV  R0,#DATAB
012A A0                         MOV  @R0,A
012B 2355                       MOV  A,#55H
012D B877                       MOV  R0,#DATARY
012F A0                         MOV  @R0,A
0130 83                         RET
                ;*************************************************
                ;*      INC DECIMAL                              *
                ;*      INPUT  R0,R0+1,R0+2                      *
                ;*************************************************
0131 10             INCDEC:     INC  @R0
0132 F0                         MOV  A,@R0
0133 530F                       ANL  A,#0FH
0135 D30A                       XRL  A,#0AH
0137 966F                       JNZ  INCEND
0139 F0                         MOV  A,@R0
013A 53F0                       ANL  A,#0F0H
013C 0310                       ADD  A,#10H
013E A0                         MOV  @R0,A
013F D3A0                       XRL  A,#0A0H
0141 966F                       JNZ  INCEND
0143 B000                       MOV  @R0,#00H
                ;
0145 18                         INC  R0
0146 10                         INC  @R0
0147 F0                         MOV  A,@R0
0148 530F                       ANL  A,#0FH
014A D30A                       XRL  A,#0AH
014C 966F                       JNZ  INCEND
014E F0                         MOV  A,@R0
014F 53F0                       ANL  A,#0F0H
0151 0310                       ADD  A,#10H
0153 A0                         MOV  @R0,A
0154 D3A0                       XRL  A,#0A0H
0156 966F                       JNZ  INCEND
0158 B000                       MOV  @R0,#00H
                ;
015A 18                         INC  R0
015B 10                         INC  @R0
015C F0                         MOV  A,@R0
015D 530F                       ANL  A,#0FH
015F D30A                       XRL  A,#0AH
0161 966F                       JNZ  INCEND
0163 F0                         MOV  A,@R0
0164 53F0                       ANL  A,#0F0H
0166 0310                       ADD  A,#10H
```

```
0168 A0                         MOV  @R0,A
0169 D3A0                       XRL  A,#0A0H
016B 966F                       JNZ  INCEND
016D B000                       MOV  @R0,#00H
016F 83             INCEND:     RET
                    ;*****************************************************
                    ;*           INITIAL   LCD DISPLAY                   *
                    ;*  P1.0-P1.7    : PIN D0-D7  (DATA READ/WRITE LCD) *
                    ;*  P2.4         : PIN RS     (REGITER SELECTION)   *
                    ;*  P2.5         : PIN R/W    (READ/WRITE)          *
                    ;*  P2.6         : PIN E      (ENABLE SIGNAL PULSE) *
                    ;*****************************************************
0170 9A8F           INITLCD:    ANL  P2,#8FH
0172 2338                       MOV  A,#38H         ;DL=1 8BIT,N=1 1/16 DUTY,F=0 5X
0174 39                         OUTL P1,A
0175 34AB                       CALL EPLUSE
0177 7491                       CALL DELAY
0179 230D                       MOV  A,#0DH         ;D=1 OFF,C=0 CURSOR ON,B=1 BLINK
017B 39                         OUTL P1,A
017C 34AB                       CALL EPLUSE
017E 2306                       MOV  A,#06H         ;I/D=1 INCREMENT,S=0 RIGHT
0180 39                         OUTL P1,A
0181 34AB                       CALL EPLUSE
0183 9A8F           CLRALL:     ANL  P2,#8FH
0185 2301                       MOV  A,#01H         ;CLEAR ALL DISPLAY
0187 39                         OUTL P1,A
0188 34AB                       CALL EPLUSE
018A 7491                       CALL DELAY
018C 83                         RET
                    ;*****************************************************
                    ;*          WRITE LINE 20 CHAR                       *
                    ;*     INPUT    (R0)    =    DATA                    *
                    ;*                A     =    LINE                    *
                    ;*****************************************************
018D AC             WRLINE:     MOV   R4,A
018E D301                       XRL   A,#01H
0190 C698                       JZ    WRL1
0192 FC                         MOV   A,R4
0193 D302                       XRL   A,#02H
0195 C69E                       JZ    WRL2
0197 83                         RET
                    ;
0198 2300           WRL1:       MOV   A,#00H
019A 34B4                       CALL  GOTO
019C 24A2                       JMP   WRLM
019E 2340           WRL2:       MOV   A,#40H
01A0 34B4                       CALL  GOTO
                    ;
01A2 BC14           WRLM:       MOV   R4,#20
01A4 F0             WRL:        MOV   A,@R0
01A5 34BC                       CALL  WRBYTE
01A7 18                         INC   R0
01A8 ECA4                       DJNZ  R4,WRL
01AA 83                         RET
                    ;*****************************************************
                    ;*              ENABLE PLUSE SUB.                    *
                    ;*****************************************************
01AB 8A40           EPLUSE:     ORL    P2,#40H      ;SET BNABLE = 1
01AD BB40                       MOV    R3,#40H
```

```
01AF EBAF          EP1:          DJNZ   R3,EP1
01B1 9ABF                        ANL    P2,#0BFH       ;SET ENABLE = 0
01B3 83                          RET
                ;*****************************************************
                ;*             GOTO   POSITION                       *
                ;*     INPUT   REG    A  =  DATA                     *
                ;*****************************************************
01B4 4380          GOTO:         ORL    A,#80H         ;SET DD RAM
01B6 39                          OUTL   P1,A
01B7 9A8F                        ANL    P2,#8FH        ;SET RS=0,R/W=0
01B9 34AB                        CALL   EPLUSE
01BB 83                          RET
                ;*****************************************************
                ;*             WRITE  DATA SUB.                      *
                ;*    INPUT    REG    A = DATA                       *
                ;*****************************************************
01BC 8A10          WRBYTE:       ORL    P2,#10H        ;SET DATA WRITE
01BE 39                          OUTL   P1,A           ;DATA BYTE
01BF 34AB                        CALL   EPLUSE
01C1 9AEF                        ANL    P2,#0EFH       ;OFF
01C3 83                          RET

                ;*************************************************
                ;*   ROUTINE SEND DATA 1 BYTE FROM COM           *
                ;*   I/P REG : R2 (DATA)                         *
                ;*   O/P     : TXD                               *
                ;*************************************************
01C4 B860          TXCOM:        MOV   R0,#DPORTA
01C6 F0                          MOV   A,@R0
01C7 5303                        ANL   A,#03H
                ;                 MOV   R3,A
01C9 96D1                        JNZ   TX_C1
01CB FA                          MOV   A,R2
01CC 531F                        ANL   A,#1FH
01CE AA                          MOV   R2,A
01CF 24E3                        JMP   TX_C5
01D1 07            TX_C1:        DEC   A
01D2 96DA                        JNZ   TX_C2
01D4 FA                          MOV   A,R2
01D5 533F                        ANL   A,#3FH
01D7 AA                          MOV   R2,A
01D8 24E3                        JMP   TX_C5
01DA 07            TX_C2:        DEC   A
01DB 96E3                        JNZ   TX_C5
01DD FA                          MOV   A,R2
01DE 537F                        ANL   A,#7FH
01E0 AA                          MOV   R2,A
01E1 24E3                        JMP   TX_C5
01E3 9AF4          TX_C5:        ANL   P2,#MOCOM       ; DATA DIGIT  PORT
01E5 8A01                        ORL   P2,#COMCON
01E7 00                          NOP
01E8 08                          INS   A,BUS
01E9 8A0C                        ORL   P2,#CLR
01EB 12EF                        JB0   TX_C6
01ED 24E3                        JMP   TX_C5
01EF 9AF4          TX_C6:        ANL   P2,#MOCOM       ; DATA DIGIT  PORT
01F1 8A00                        ORL   P2,#COMDATA
01F3 FA                          MOV   A,R2
01F4 02                          OUTL BUS,A
```

```
01F5 00                         NOP
01F6 8A0C                       ORL  P2,#CLR
01F8 B874                       MOV  R0,#DARS
01FA A0                         MOV  @R0,A
01FB B86F                       MOV  R0,#ERCOU1
01FD 3431                       CALL INCDEC
01FF 83                         RET
                ;*************************************************
                ;*            DATA                               *
                ;*************************************************
                                ;THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789
0200 2054484520     DATA0:      DFB  " THE "
0205 5155494348                 DFB  "QUICK"
020A 20464F5820                 DFB  " FOX "
020F 4A554D5053                 DFB  "JUMPS"
0214 204F564552                 DFB  " OVER"
0219 2054484520                 DFB  " THE "
021E 4C415A5920                 DFB  "LAZY "
0223 444F472020                 DFB  "DOG  "
0228 3031323334                 DFB  "01234"
022D 3536373839                 DFB  "56789"
0232 202020                     DFB  "   "
0235 1F10140104     DATA1:      DFB  1FH,10H,14H,01H,04H,17H,07H,06H,0EH,0FH
023F 04190A1813                 DFB  04H,19H,0AH,18H,13H,0CH,04H,0DH,18H,1DH
0249 040B071C16                 DFB  04H,0BH,07H,1CH,16H,05H,04H,18H,1EH,01H
0253 0A04101401                 DFB  0AH,04H,10H,14H,01H,04H,12H,03H,11H,15H
025D 0409181A04                 DFB  04H,09H,18H,1AH,04H,1BH,16H,17H,13H,01H
0267 0A10150706                 DFB  0AH,10H,15H,07H,06H,18H,08H,02H
                                ;FACULTY OF ENGINEERING (KMITL)
026F 1F0D030E07     DATA2:      DFB  1FH,0DH,03H,0EH,07H,12H,10H,15H,04H,18H
0279 0D04010C1A                 DFB  0DH,04H,01H,0CH,1AH,06H,0CH,01H,01H,0AH
0283 060C1A041B                 DFB  06H,0CH,1AH,04H,1BH,0FH,1FH,0FH,1CH,06H
028D 10121B1204                 DFB  10H,12H,1BH,12H,04H,08H,02H

                ;*************************************************
                ;*   ROUTINE SEND DATA 1 BYTE FROM COM           *
                ;*   I/P REG : DATA1                             *
                ;*   O/P     : TXD                               *
                ;*************************************************
0294 9AF4         TXCOM1:       ANL  P2,#MOCOM     ; DATA DIGIT  PORT
0296 8A01                       ORL  P2,#COMCON
0298 00                         NOP
0299 08                         INS  A,BUS
029A 8A0C                       ORL  P2,#CLR
029C 12A0                       JB0  TX_C11
029E 4494                       JMP  TXCOM1
02A0 B966         TX_C11:       MOV  R1,#DCOUNT
02A2 2335                       MOV  A,#35H
02A4 61                         ADD  A,@R1
02A5 AD                         MOV  R5,A
02A6 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
02A8 8A00                       ORL  P2,#COMDATA
02AA FD                         MOV  A,R5
02AB A3                         MOVP A,@A
02AC AD                         MOV  R5,A
02AD 02                         OUTL BUS,A
02AE 00                         NOP
02AF 8A0C                       ORL  P2,#CLR
02B1 B874                       MOV  R0,#DARS
```

```
02B3 A0                         MOV  @R0,A
                    ;
02B4 B966                       MOV  R1,#DCOUNT
02B6 11                         INC  @R1
02B7 F1                         MOV  A,@R1
02B8 D33A                       XRL  A,#58
02BA 96BF                       JNZ  TX_C12
02BC 2300                       MOV  A,#00H
02BE A1                         MOV  @R1,A
02BF B86F           TX_C12:     MOV  R0,#ERCOU1
02C1 3431                       CALL INCDEC
02C3 83                         RET
                    ;***********************************************
                    ;*   ROUTINE SEND DATA 1 BYTE FROM COM          *
                    ;*   I/P REG : DATA2                            *
                    ;*   O/P     : TXD                              *
                    ;***********************************************
02C4 9AF4           TXCOM2:     ANL  P2,#MOCOM      ; DATA DIGIT  PORT
02C6 8A01                       ORL  P2,#COMCON
02C8 00                         NOP
02C9 08                         INS  A,BUS
02CA 8A0C                       ORL  P2,#CLR
02CC 12D0                       JB0  TX_C21
02CE 44C4                       JMP  TXCOM2
02D0 B966           TX_C21:     MOV  R1,#DCOUNT
02D2 236F                       MOV  A,#6FH
02D4 61                         ADD  A,@R1
02D5 AD                         MOV  R5,A
02D6 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
02D8 8A00                       ORL  P2,#COMDATA
02DA FD                         MOV  A,R5
02DB A3                         MOVP A,@A
02DC AD                         MOV  R5,A
02DD 02                         OUTL BUS,A
02DE 00                         NOP
02DF 8A0C                       ORL  P2,#CLR
02E1 B874                       MOV  R0,#DARS
02E3 A0                         MOV  @R0,A
                    ;
02E4 B966                       MOV  R1,#DCOUNT
02E6 11                         INC  @R1
02E7 F1                         MOV  A,@R1
02E8 D325                       XRL  A,#37
02EA 96EF                       JNZ  TX_C22
02EC 2300                       MOV  A,#00H
02EE A1                         MOV  @R1,A
02EF B86F           TX_C22:     MOV  R0,#ERCOU1
02F1 3431                       CALL INCDEC
02F3 83                         RET
02F4 00                         NOP
02F5 00                         NOP
02F6 00                         NOP
02F7 00                         NOP
02F8 00                         NOP
02F9 00                         NOP
02FA 00                         NOP
02FB 00                         NOP
02FC 00                         NOP
02FD 00                         NOP
```

```
02FE 00                         NOP
02FF 00                         NOP
0300 03190E0901     DATA3:      DFB  03H,19H,0EH,09H,01H,0DH,1AH,14H,06H,0BH
030A 0F121C0C18                 DFB  0FH,12H,1CH,0CH,18H,16H,17H,0AH,05H,10H
0314 071E131D15                 DFB  07H,1EH,13H,1DH,15H,11H,08H,02H,04H,00H
031E 4142434445     DATA4:      DFB  "ABCDE"
0323 464748494A                 DFB  "FGHIJ"
0328 4B4C4D4E4F                 DFB  "KLMNO"
032D 5051525354                 DFB  "PQRST"
0332 5556575859                 DFB  "UVWXY"
0337 5A20202020                 DFB  "Z    "
033C 2D3F3A2033     DATA5:      DFB  "-?: 3"
0341 2020203820                 DFB  "   8 "
0346 28292E2C39                 DFB  "().,9"
034B 3031342735                 DFB  "014'5"
0350 373D322F36                 DFB  "7=2/6"
0355 2B20202020                 DFB  "+    "
                ;*****************************************************
                ;*        QUICK BROWN FOX TO ASCII                   *
                ;*     INPUT    REG    A                             *
                ;*     OUTPUT   REG    A                             *
                ;*****************************************************
035A BA1E           FOXCON:     MOV  R2,#30
035C AB                         MOV  R3,A
035D B900                       MOV  R1,#00H
035F B86A                       MOV  R0,#DFOX
0361 D31F                       XRL  A,#1FH
0363 966C                       JNZ  FOXC1
0365 2300                       MOV  A,#00H
0367 A0                         MOV  @R0,A
0368 2320                       MOV  A,#20H
036A 648F                       JMP  FOXCE
036C FB             FOXC1:      MOV  A,R3
036D D31B                       XRL  A,#1BH
036F 9678                       JNZ  FOXC2
0371 2301                       MOV  A,#01H
0373 A0                         MOV  @R0,A
0374 2320                       MOV  A,#20H
0376 648F                       JMP  FOXCE
0378 F9             FOXC2:      MOV  A,R1
0379 A3                         MOVP A,@A
037A DB                         XRL  A,R3
037B C684                       JZ   FOXC3
037D 19                         INC  R1
037E EA78                       DJNZ R2,FOXC2
0380 2320                       MOV  A,#20H
0382 648F                       JMP  FOXCE
                ;
0384 F0             FOXC3:      MOV  A,@R0
0385 C68B                       JZ   FOXC4
0387 233C                       MOV  A,#3CH         ;FIGS
0389 648D                       JMP  FOXC5
038B 231E           FOXC4:      MOV  A,#1EH         ;LTRS
038D 69             FOXC5:      ADD  A,R1
038E A3                         MOVP A,@A
038F 83             FOXCE:      RET
0390 00                         NOP
                ;*****************************************
                ;*          DELAY                        *
```

```
                ;*****************************************
0391 BF40       DELAY:      MOV  R7,#40H
0393 00         LOOP1:      NOP
0394 EF93                   DJNZ R7,LOOP1
0396 83                     RET
0397 BF10       DELAY1:     MOV  R7,#10H
0399 00         LOOP2:      NOP
039A EF99                   DJNZ R7,LOOP2
039C 83                     RET
039D 00                     NOP
039E 00                     NOP
039F 00                     NOP
03A0 00                     NOP
03A1 00                     NOP
03A2 00                     NOP
03A3 00                     NOP
03A4 00                     NOP
03A5 00                     NOP
03A6 00                     NOP
03A7 00                     NOP
03A8 00                     NOP
03A9 00                     NOP
03AA 00                     NOP
03AB 00                     NOP
03AC 00                     NOP
03AD 00                     NOP
03AE 00                     NOP
03AF 00                     NOP
03B0 00                     NOP
03B1 00                     NOP
03B2 00                     NOP
03B3 00                     NOP
03B4 00                     NOP
03B5 00                     NOP
03B6 00                     NOP
03B7 00                     NOP
03B8 00                     NOP
03B9 00                     NOP
03BA 00                     NOP
03BB 00                     NOP
03BC 00                     NOP
03BD 00                     NOP
03BE 00                     NOP
03BF 00                     NOP
03C0 00                     NOP
03C1 00                     NOP
03C2 00                     NOP
03C3 00                     NOP
03C4 00                     NOP
03C5 00                     NOP
03C6 00                     NOP
03C7 00                     NOP
03C8 00                     NOP
03C9 00                     NOP
03CA 00                     NOP
03CB 00                     NOP
03CC 00                     NOP
03CD 00                     NOP
03CE 00                     NOP
```

```
03CF 00                         NOP
03D0 00                         NOP
03D1 00                         NOP
03D2 00                         NOP
03D3 00                         NOP
03D4 00                         NOP
03D5 00                         NOP
03D6 00                         NOP
03D7 00                         NOP
03D8 00                         NOP
03D9 00                         NOP
03DA 00                         NOP
03DB 00                         NOP
03DC 00                         NOP
03DD 00                         NOP
03DE 00                         NOP
03DF 00                         NOP
03E0 00                         NOP
03E1 00                         NOP
03E2 00                         NOP
03E3 00                         NOP
03E4 00                         NOP
03E5 00                         NOP
03E6 00                         NOP
03E7 00                         NOP
03E8 00                         NOP
03E9 00                         NOP
03EA 00                         NOP
03EB 00                         NOP
03EC 00                         NOP
03ED 00                         NOP
03EE 00                         NOP
03EF 00                         NOP
03F0 00                         NOP
03F1 00                         NOP
03F2 00                         NOP
03F3 00                         NOP
03F4 00                         NOP
03F5 00                         NOP
03F6 00                         NOP
03F7 00                         NOP
03F8 00                         NOP
03F9 00                         NOP
03FA 00                         NOP
03FB 00                         NOP
03FC 00                         NOP
03FD 00                         NOP
03FE 00                         NOP
03FF 00                         NOP
                ;*****************************************************
                ;*          SET       8251                           *
                ;*****************************************************
0400 B864       SETCOM:         MOV   R0,#DPORTC
0402 F0                         MOV   A,@R0
0403 AB                         MOV   R3,A
0404 B860                       MOV   R0,#DPORTA
0406 F0                         MOV   A,@R0
0407 18                         INC   R0
0408 AA                         MOV   R2,A
```

```
0409 D0                         XRL   A,@R0
040A 9614                       JNZ   SETCOM2

040C B864                       MOV   R0,#DPORTC
040E F0                         MOV   A,@R0
040F 18                         INC   R0
0410 AB                         MOV   R3,A
0411 D0                         XRL   A,@R0
0412 C6C7                       JZ    SETCOME
0414 B866           SETCOM2:    MOV   R0,#DCOUNT
0416 27                         CLR   A
0417 A0                         MOV   @R0,A
0418 B86A                       MOV   R0,#DFOX
041A A0                         MOV   @R0,A
041B FA                         MOV   A,R2
041C 5340                       ANL   A,#40H
041E AC                         MOV   R4,A
041F B861                       MOV   R0,#DPORTA+1
0421 F0                         MOV   A,@R0
0422 5340                       ANL   A,#40H
0424 DC                         XRL   A,R4
0425 C62D                       JZ    SETCOM21
0427 B868                       MOV   R0,#CUR
0429 27                         CLR   A
042A A0                         MOV   @R0,A
042B 3483                       CALL  CLRALL
042D B861           SETCOM21:   MOV   R0,#DPORTA+1
042F F0                         MOV   A,@R0
0430 B241                       JB5   SETCOM23
0432 FA                         MOV   A,R2
0433 37                         CPL   A
0434 B241                       JB5   SETCOM23
0436 27                         CLR   A
0437 B86C                       MOV   R0,#ECOU1
0439 BC06                       MOV   R4,#06H
043B A0             SETCOM22:   MOV   @R0,A
043C 18                         INC   R0
043D EC3B                       DJNZ  R4,SETCOM22
043F 84C7                       JMP   SETCOME
0441 B861           SETCOM23:   MOV   R0,#DPORTA+1
0443 FA                         MOV   A,R2
0444 A0                         MOV   @R0,A
0445 B865                       MOV   R0,#DPORTC+1
0447 FB                         MOV   A,R3
0448 A0                         MOV   @R0,A
0449 B864                       MOV   R0,#DPORTC
044B BC07                       MOV   R4,#07H
044D F0                         MOV   A,@R0
044E 53C0                       ANL   A,#0C0H
0450 AA                         MOV   R2,A
0451 D380                       XRL   A,#80H
0453 965A                       JNZ   SETCOM3
0455 BB10                       MOV   R3,#10H         ;ODD PARITY
0457 1C                         INC   R4
0458 8466                       JMP   SETCOM5
045A FA             SETCOM3:    MOV   A,R2
045B D340                       XRL   A,#40H
045D 9664                       JNZ   SETCOM4
045F BB30                       MOV   R3,#30H         ;EVEN PARITY
```

```
0461 1C                         INC  R4
0462 8466                       JMP  SETCOM5
0464 BB00          SETCOM4:     MOV  R3,#00H        ;PARITY DISABLE
0466 F0            SETCOM5:     MOV  A,@R0
0467 5330                       ANL  A,#30H
0469 AA                         MOV  R2,A
046A D320                       XRL  A,#20H
046C 9672                       JNZ  SETCOM6
046E 2340                       MOV  A,#40H         ;1 STOP BIT
0470 847D                       JMP  SETCOM8
0472 FA            SETCOM6:     MOV  A,R2
0473 B310                       XRL  A,#10H
0475 967B                       JNZ  SETCOM7
0477 23C0                       MOV  A,#0C0H        ;2 STOP BIT
0479 847D                       JMP  SETCOM8
047B 2380          SETCOM7:     MOV  A,#80H         ;1 1/2 STOP BIT
047D 4B            SETCOM8:     ORL  A,R3
047E AB                         MOV  R3,A
047F B860                       MOV  R0,#DPORTA
0481 F0                         MOV  A,@R0
0482 5303                       ANL  A,#03H
0484 6C                         ADD  A,R4
0485 AC                         MOV  R4,A
0486 F0                         MOV  A,@R0
0487 E7                         RL   A
0488 E7                         RL   A
0489 530C                       ANL  A,#0CH
048B 4302                       ORL  A,#02H
048D 4B                         ORL  A,R3
048E AB                         MOV  R3,A
048F 9AF8                       ANL  P2,#MODE       ; DATA DIGIT  PORT
0491 8A02                       ORL  P2,#PC
0493 FC                         MOV  A,R4           ; INPUT ALL
0494 02                         OUTL BUS,A
0495 00                         NOP
0496 8A0C                       ORL  P2,#CLR

0498 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
049A 8A01                       ORL  P2,#COMCON
049C 2311                       MOV  A,#11H         ; INTERNAL RESET 8251 COM1
049E 02                         OUTL BUS,A
049F 00                         NOP
04A0 8A0C                       ORL  P2,#CLR
                ;
04A2 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
04A4 8A01                       ORL  P2,#COMCON
04A6 2340                       MOV  A,#40H         ; IR(INTERNAL RESET)
04A8 02                         OUTL BUS,A          ; RESET COM1
04A9 00                         NOP
04AA 8A0C                       ORL  P2,#CLR
                ; SET MODE 8251 COM1
                ; 8 BITDATA,1 STOPBIT,NOPARITY,*1 BUAD
04AC 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
04AE 8A01                       ORL  P2,#COMCON
04B0 FB                         MOV  A,R3           ; MODE WORD(4FH)
04B1 02                         OUTL BUS,A
04B2 00                         NOP
04B3 8A0C                       ORL  P2,#CLR
                ; SEND CONTROL WORD (15h)
```

```
                    ; ER,TxEN,RxEN
04B5 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
04B7 8A01                       ORL  P2,#COMCON
04B9 2315                       MOV  A,#15H         ; TxEN,RxEN,ER
04BB 02                         OUTL BUS,A
04BC 00                         NOP
04BD 8A0C                       ORL  P2,#CLR
                    ;
04BF 9AF4                       ANL  P2,#MOCOM      ; DATA DIGIT  PORT
04C1 8A00                       ORL  P2,#COMDATA
04C3 08                         INS  A,BUS
04C4 00                         NOP
04C5 8A0C                       ORL  P2,#CLR
04C7 83             SETCOME:    RET
                    ;*************************************************
                    ;*       HEX TO ASCII                            *
                    ;*    INPUT      R0    = LOW                     *
                    ;*               R0+1  = HI                      *
                    ;*               R0+2  = HI 4BIT                 *
                    ;*               R4    = CUR                     *
                    ;*************************************************
04C8 18             DECASCII:   INC  R0
04C9 18                         INC  R0
04CA F0                         MOV  A,@R0
04CB 77                         RR   A
04CC 77                         RR   A
04CD 77                         RR   A
04CE 77                         RR   A
04CF 530F                       ANL  A,#0FH
04D1 4330                       ORL  A,#30H
04D3 AD                         MOV  R5,A
04D4 FC                         MOV  A,R4
04D5 34B4                       CALL GOTO
04D7 FD                         MOV  A,R5
04D8 34BC                       CALL WRBYTE
04DA F0                         MOV  A,@R0
04DB 530F                       ANL  A,#0FH
04DD 4330                       ORL  A,#30H
04DF 34BC                       CALL WRBYTE
04E1 C8                         DEC  R0
04E2 F0                         MOV  A,@R0
04E3 77                         RR   A
04E4 77                         RR   A
04E5 77                         RR   A
04E6 77                         RR   A
04E7 530F                       ANL  A,#0FH
04E9 4330                       ORL  A,#30H
04EB 34BC                       CALL WRBYTE
04ED F0                         MOV  A,@R0
04EE 530F                       ANL  A,#0FH
04F0 4330                       ORL  A,#30H
04F2 34BC                       CALL WRBYTE
04F4 C8                         DEC  R0
04F5 F0                         MOV  A,@R0
04F6 77                         RR   A
04F7 77                         RR   A
04F8 77                         RR   A
04F9 77                         RR   A
04FA 530F                       ANL  A,#0FH
```

```
04FC 4330                       ORL  A,#30H
04FE 34BC                       CALL WRBYTE
0500 F0                         MOV  A,@R0
0501 530F                       ANL  A,#0FH
0503 4330                       ORL  A,#30H
0505 34BC                       CALL WRBYTE
0507 83                         RET
0508 00                         NOP
0509 00                         NOP
050A 00                         NOP
                ;*************************************************
                ;*        MODE                                   *
                ;*************************************************
050B 146B       MODEA:          CALL RXCOM1
050D B860                       MOV  R0,#DPORTA
050F F0                         MOV  A,@R0
0510 37                         CPL  A
0511 F223                       JB7  MOAEND
0513 B875                       MOV  R0,#COUNT
0515 10                         INC  @R0
0516 F0                         MOV  A,@R0
0517 D331                       XRL  A,#31H
0519 9623                       JNZ  MOAEND
051B 27                         CLR  A
051C A0                         MOV  @R0,A
051D B862                       MOV  R0,#DPORTB
051F F0                         MOV  A,@R0
0520 AA                         MOV  R2,A
0521 34C4                       CALL TXCOM
0523 83         MOAEND:         RET
0524 146B       MODEB:          CALL RXCOM1
0526 B860                       MOV  R0,#DPORTA
0528 F0                         MOV  A,@R0
0529 37                         CPL  A
052A F23D                       JB7  MOBEND
052C B875                       MOV  R0,#COUNT
052E 10                         INC  @R0
052F F0                         MOV  A,@R0
0530 D331                       XRL  A,#31H
0532 963D                       JNZ  MOBEND
0534 27                         CLR  A
0535 A0                         MOV  @R0,A
0536 B876                       MOV  R0,#DATAB
0538 F0                         MOV  A,@R0
0539 AA                         MOV  R2,A
053A 10                         INC  @R0
053B 34C4                       CALL TXCOM
053D 83         MOBEND:         RET
053E 146B       MODEC:          CALL RXCOM1
0540 B860                       MOV  R0,#DPORTA
0542 F0                         MOV  A,@R0
0543 37                         CPL  A
0544 F258                       JB7  MOCEND
0546 B875                       MOV  R0,#COUNT
0548 10                         INC  @R0
0549 F0                         MOV  A,@R0
054A D331                       XRL  A,#31H
054C 9658                       JNZ  MOCEND
054E 27                         CLR  A
```

```
054F A0                         MOV   @R0,A
0550 B877                       MOV   R0,#DATARY
0552 F0                         MOV   A,@R0
0553 37                         CPL   A
0554 A0                         MOV   @R0,A
0555 AA                         MOV   R2,A
0556 34C4                       CALL  TXCOM
0558 83             MOCEND:     RET
0559 146B           MODED:      CALL  RXCOM1
055B B860                       MOV   R0,#DPORTA
055D F0                         MOV   A,@R0
055E 37                         CPL   A
055F F26D                       JB7   MODEND
0561 B875                       MOV   R0,#COUNT
0563 10                         INC   @R0
0564 F0                         MOV   A,@R0
0565 D331                       XRL   A,#31H
0567 966D                       JNZ   MODEND
0569 27                         CLR   A
056A A0                         MOV   @R0,A
056B 5494                       CALL  TXCOM1
056D 83             MODEND:     RET
056E 146B           MODEE:      CALL  RXCOM1
0570 B860                       MOV   R0,#DPORTA
0572 F0                         MOV   A,@R0
0573 37                         CPL   A
0574 F282                       JB7   MOEEND
0576 B875                       MOV   R0,#COUNT
0578 10                         INC   @R0
0579 F0                         MOV   A,@R0
057A D331                       XRL   A,#31H
057C 9682                       JNZ   MOEEND
057E 27                         CLR   A
057F A0                         MOV   @R0,A
0580 54C4                       CALL  TXCOM2
0582 83             MOEEND:     RET

0583                INTLINE:
0583                INTTIMER:
0583 83                         RET
0000                            END
```

000C	CLR	0183	CLRALL	0112	CLRM
010E	CLRMEN	0001	COMCON	0000	COMDATA
0003	CONP	0075	COUNT	0068	CUR
0074	DARS	0200	DATA0	0235	DATA1
026F	DATA2	0300	DATA3	031E	DATA4
033C	DATA5	0076	DATAB	0040	DATAD
0077	DATARY	0066	DCOUNT	04C8	DECASCII
0050	DEDATA	0391	DELAY	0397	DELAY1
006A	DFOX	0060	DPORTA	0062	DPORTB
0064	DPORTC	006C	ECOU1	006D	ECOU2
01AF	EP1	01AB	EPLUSE	006F	ERCOU1
0070	ERCOU2	036C	FOXC1	0378	FOXC2
0384	FOXC3	038B	FOXC4	038D	FOXC5
038F	FOXCE	035A	FOXCON	01B4	GOTO
0131	INCDEC	016F	INCEND	0170	INITLCD
00EC	INPORT	0583	INTLINE	0583	INTTIMER
0393	LOOP1	0399	LOOP2	0031	MAIN
0523	MOAEND	053D	MOBEND	0558	MOCEND
00F4	MOCOM	00F8	MODE	0067	MODE1
0063	MODE2	005F	MODE3	005B	MODE4
050B	MODEA	0524	MODEB	053E	MODEC
0069	MODECOE	0045	MODECOM	0559	MODED
056E	MODEE	056D	MODEND	0582	MOEEND
0002	NUMBERS	0000	PA	0001	PB
0002	PC	006B	RXCOM1	00EB	RXCOME
007B	RX_C1	007F	RX_C12	0091	RX_C13
0093	RX_C2	00AB	RX_C20	00BA	RX_C21
00C8	RX_C22	00D4	RX_C23	00D9	RX_C24
0400	SETCOM	0414	SETCOM2	042D	SETCOM21
043B	SETCOM22	0441	SETCOM23	045A	SETCOM3
0464	SETCOM4	0466	SETCOM5	0472	SETCOM6
047B	SETCOM7	047D	SETCOM8	04C7	SETCOME
000A	START	003B	TE	01C4	TXCOM
0294	TXCOM1	02C4	TXCOM2	01D1	TX_C1
02A0	TX_C11	02BF	TX_C12	01DA	TX_C2
02D0	TX_C21	02EF	TX_C22	01E3	TX_C5
01EF	TX_C6	01BC	WRBYTE	01A4	WRL
0198	WRL1	019E	WRL2	018D	WRLINE
01A2	WRLM				

```
                ;       ****************************************
                ;       *  SOFTWARE FOR DISTORTION GENERATOR   *
                ;       *  FILE NAME    : DISGEN.ASM           *
                ;       *  DATE         :  20/07/36            *
                ;       ****************************************
                ;
0000                            CPU  "Z80.TBL"
0000                            HOF  "BIN8"
                ;
0000                            ORG  0000h

0000 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h
0010 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0020 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0030 00005F0000                 DFB   0h,0h,5fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0040 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0050 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h
0060 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0070 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0080 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h
0090 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
00A0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
00B0 00100F4000                 DFB   0h,10h,0fh,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
00C0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
00D0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h
00E0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
00F0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0100 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h
0110 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0120 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0130 10000F0040                 DFB   10h,0h,0fh,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0140 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0150 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h
0160 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0170 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0180 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h
0190 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
01A0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
01B0 00000F0000                 DFB   0h,0h,0fh,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
01C0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
01D0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
01E0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
01F0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0200 0000000000                 DFB   0h,0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0210 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0220 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h
0230 00000F0000                 DFB   0h,0h,0fh,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0240 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0250 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h
0260 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0270 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0280 0000000000                 DFB   0h,0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0290 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
02A0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h
02B0 00000F0000                 DFB   0h,0h,0fh,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h
02C0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
02D0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h
02E0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
02F0 0000000000                 DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
0300 0000000020          DFB   0h,0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0310 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0320 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h
0330 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h
0340 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0350 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0360 8000000000          DFB   80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0370 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0380 0000002000          DFB   0h,0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0390 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
03A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
03B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
03C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
03D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
03E0 0080000000          DFB   0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
03F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0400 0000200000          DFB   0h,0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0410 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0420 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h
0430 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h
0440 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0450 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0460 0000800000          DFB   0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0470 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0480 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0490 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
04A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
04B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
04C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
04D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
04E0 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
04F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0500 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0510 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0520 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
0530 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
0540 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0550 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0560 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0570 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0580 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0590 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
05A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
05B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
05C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
05D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
05E0 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
05F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0600 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0610 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0620 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
0630 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
0640 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0650 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0660 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0670 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0680 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0690 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
06A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
```

```
06B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
06C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
06D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
06E0 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
06F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0700 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0710 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0720 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
0730 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
0740 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0750 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0760 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0770 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0780 0020000000          DFB   0h,20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0790 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
07A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
07B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
07C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
07D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
07E0 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
07F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0800 2000000000          DFB   20h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0810 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0820 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h
0830 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h
0840 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0850 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0860 0000000080          DFB   0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0870 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0880 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0890 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
08A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h
08B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h
08C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
08D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
08E0 0000000000          DFB   0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
08F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0900 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0910 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0920 0000000000          DFB   0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0930 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h
0940 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0950 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0960 0000000000          DFB   0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0970 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0980 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0990 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
09A0 0000000000          DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
09B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
09C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
09D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
09E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
09F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A20 0000000010          DFB   0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A40 4000000000          DFB   40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
0A60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h
0A70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0A90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0AA0 0000001000          DFB   0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0AB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0AC0 0040000000          DFB   0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0AD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0AE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h
0AF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B20 0000100000          DFB   0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B40 0000400000          DFB   0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h
0B70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0B90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0BA0 0010000000          DFB   0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0BB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0BC0 0000004000          DFB   0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0BD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0BE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h
0BF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C20 1000000000          DFB   10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C40 0000000040          DFB   0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h
0C70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0C90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0CA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0CB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0CC0 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0CD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0CE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0CF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0D20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D40 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0D70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0D90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0DA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0DB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0DC0 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0DD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0DE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0DF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
0E10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0E20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E40 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0E70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0E90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0EA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0EB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0EC0 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0ED0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0EE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0EF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0F20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F40 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0F70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0F90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
0FA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0FB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0FC0 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0FD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
0FE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
0FF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1000 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1010 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h
1020 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1030 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1040 0000000000          DFB   0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1050 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1060 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h
1070 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1080 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1090 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h
10A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
10B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
10C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h
10D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
10E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h
10F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1100 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1110 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h
1120 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1130 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1140 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h
1150 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1160 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1170 8000000000          DFB   80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1180 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1190 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
11A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
11B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
11C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
11D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
11E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
11F0 0080000000          DFB   0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1200 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1210 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h
1220 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1230 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1240 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h
1250 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1260 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1270 0000800000          DFB   0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1280 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1290 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h
12A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
12B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
12C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h
12D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
12E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
12F0 0000008000          DFB   0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1300 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1310 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h
1320 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1330 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1340 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h
1350 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1360 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1370 0000000080          DFB   0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1380 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1390 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h
13A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
13B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
13C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h
13D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
13E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
13F0 0000000000          DFB   0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1400 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1410 0000000000          DFB   0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1420 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1430 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1440 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h
1450 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1460 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1470 0000000000          DFB   0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1480 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1490 0000000000          DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
14A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
14B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
14C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
14D0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
14E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
14F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1500 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1510 0000000000          DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1520 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1530 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1540 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
1550 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1560 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
1570 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1580 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1590 0000000000                DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
15A0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
15B0 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
15C0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
15D0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
15E0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
15F0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1600 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1610 0000000000                DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1620 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1630 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1640 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
1650 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1660 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1670 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1680 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1690 0000000000                DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
16A0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
16B0 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
16C0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
16D0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
16E0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
16F0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1700 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1710 0000000000                DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1720 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1730 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1740 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
1750 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1760 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1770 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1780 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1790 0000000000                DFB   0h,0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
17A0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
17B0 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
17C0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,40h
17D0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
17E0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
17F0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h,0h
1800 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1810 0000000010                DFB   0h,0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1820 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1830 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1840 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1850 4000000000                DFB   40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1860 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1870 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h,0h
1880 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1890 0000001000                DFB   0h,0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18A0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18B0 00000F0000                DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18C0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18D0 0040000000                DFB   0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18E0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
18F0 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h,0h
1900 0000000000                DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1910 0000100000                DFB   0h,0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
1920 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1930 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1940 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1950 0000400000          DFB   0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1960 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1970 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h,0h
1980 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1990 0010000000          DFB   0h,10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19A0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19B0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19C0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19D0 0000004000          DFB   0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19E0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
19F0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h,0h
1A00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A10 1000000000          DFB   10h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A50 0000000040          DFB   0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1A70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h,0h
1A80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h
1A90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AD0 0000000000          DFB   0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1AF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h,0h
1B00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h
1B10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B50 0000000000          DFB   0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1B70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h,0h
1B80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h
1B90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1BA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1BB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1BC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1BD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h,0h
1BE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1BF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,80h
1C00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h
1C10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h,0h
1C60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1C80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1C90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1CA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1CB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1CC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
```

```
1CD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1CE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1CF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1D10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1D60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1D80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1D90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1DA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1DB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1DC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1DD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1DE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1DF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1E10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1E60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1E80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1E90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1EA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1EB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1EC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1ED0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1EE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1EF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F00 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1F10 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F20 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F30 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F40 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F50 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1F60 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F70 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1F80 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,10h,0h,0h,0h,0h
1F90 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1FA0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1FB0 00000F0000          DFB   0h,0h,0fh,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1FC0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1FD0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,40h,0h,0h,0h,0h,0h,0h
1FE0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
1FF0 0000000000          DFB   0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h

END
```

0007 A	0000 B	0001 C
0002 D	0003 E	0004 H
0005 L