เครื่องอ่านแถบแม่เหล็ก
READ MAGNETIC CARD MACHINE

โดย

นาย เชวง วงษ์ฉิม
นาย บุญยิ่ง นบนอบ
นาย บุญฤกษ์ กาญจนเทพ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต
สาขาวิชาเทคโนโลยีอิเลคทรอนิคส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2537

หัวข้อปริญญานิพนธ์ เครื่องอ่านบัตรแม่เหล็ก
READ MAGNETIC CARD MACHINE

ชื่อนักศึกษา นายเชวง วงษ์ฉิม
นายบุญยิ่ง นบนอบ
นายบุญฤกษ์ กาญจนเทพ

อาจารย์ที่ปรึกษา อาจารย์ชวลิต เบญจางคประเสริฐ
อาจารย์ไพศาล สิทธิโยภาสกุล

ภาควิชา เทคนิคอุตสาหกรรม
ปีการศึกษา 2537

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมัติให้รับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ
( )

.....กรรมการ
( )

.....กรรมการ
( )

.....กรรมการ
( )

.....กรรมการ
( )

ดิษสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เครื่องอ่านบัตรแม่เหล็ก

โดย	นายเชวง วงษ์ฉิม	รหัส 36012006
	นายบุญยิ่ง นบนอบ	รหัส 36012015
	นายบุญฤกษ์ กาญจนเทพ	รหัส 36012016
อาจารย์ที่ปรึกษา	อาจารย์ ชวลิต เบญจางคประเสริฐ	
	อาจารย์ ไพศาล สิทธิโยภาสกุล	
ปีการศึกษา	2537	

บทคัดย่อ

ในปัจจุบันการใช้บัตรแถบแม่เหล็กกำลังได้รับความนิยมเป็นอย่างมากในการเบิกเงินสดจากตู้ ATM ทำให้ผู้ใช้ได้รับความสะดวกและรวดเร็ว

ดังนั้นในโปรเจ็กนี้ได้นำความคิดที่จะทำให้บัตรแถบแม่เหล็กสามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวางในชีวิตประจำวัน รวมทั้งบัตรนักศึกษาที่มีข้อมูลประจำตัวนักศึกษา,บัตรห้องสมุดที่เก็บบันทึกการใช้ยืมหนังสือจากห้องสมุดรวมอยู่ในบัตรเดียวกัน โดยสามารถใช้กับเครื่องอ่านบัตรแถบแม่เหล็กทั่วไป โดยทำงานร่วมกับเครื่องคอมพิวเตอร์แบบ PC ที่ใช้โปรแกรม VISUAL BASIC ช่วยในการอ่านและบันทึกข้อมูลต่าง ๆลงในบัตร โดยข้อกำหนดต่าง ๆเป็นไปตามมาตรฐาน ISO

READ MAGNETIC CARD MACHINE

BY MR.CHAWENG WONGCHIM NO. 36012006
MR.BOONYING NOPNOP NO. 36012015
MR.BOONRERK KANJANATAP NO. 36012016

ADVISER MR.CHAWALIT BENJANGKAPRASERT
MR.PAISAN SITTIYOPASAKUL

YEAR 1995

ABSTRACT

NOW , THE MAGNETIC CARD WILL MOSTTY POPULAR IN A CASH WITHDRAW FROM AUTOMATIC TRAILLEL MACHINE (A.T.M.) AND DETERMINED CONUIENCE TO USER.

THEREFORE IN THIS PROJECT PRESENT THE IDEAL TO APPLY A MAGNETIC CARD CAN VARIOUS OPERATION IN THE LIFE SUCH AS THE STUDENT I.D. CARD WHICH RECORDING THE PERSONAL FILES OF STUDENT , LIBRARY CARD WHICH RECORDING THE BORROW/RECIEVER LIST OF STUDENTS ARE INCLUDED IN A CARD. WHICH IT CAN USE TO GENERAL MAGNETIC CARD RECORDER AND CO-OPERATE A PERSONAL COMPUTER (PC) WHICH DEPEND ON VISUAL BASIC PROGRAM TO AIDS IN DATA STORING AND DATA TRANSFER FROM READING OR RECORDING OF VARIOUS DATA DOWN TO THE CARD AND THE STANDARD SPECIFICATION FOLLOWING TO ISO STANDARDS.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ได้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของอาจารย์ชวลิต เบญจางคประเสริฐและอาจารย์ ไพศาล สิทธิโยภาสกุล อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งได้คำแนะนำและข้อคิดเห็นต่าง ๆของการวิจัยด้วยดีตลอด ขอขอบคุณ คุณสุกฤต วิชัยดิษฐ์ และคุณ นิตินาถ คำพา สำหรับคำแนะนำซึ่งเป็นประโยชน์อย่างยิ่งในการทำปริญญานิพนธ์ และขอบคุณ เพื่อน ๆทุกคนสำหรับกำลังใจท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา-มารดา ซึ่งสนับสนุนทางด้านการเงินและกำลังใจแก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

เชวง วงษ์ฉิม
บุญยิ่ง นบนอบ
บุญฤกษ์ กาญจนเทพ

สารบัญ

บทที่ 1

บทนำ

1. วัตถุประสงค์ของโครงงาน

1.1 เพื่อศึกษามาตรฐานของบัตร ATM ISO 7811/2 - 1985 (E)

1.2 เพื่อออกแบบเครื่องอ่านแถบแม่เหล็กทั้ง HARD WARE และ SOFT WARE

1.3 เพื่อนำเครื่องอ่านแถบแม่เหล็กไปประยุกต์ใช้งาน

2. ความเป็นมาของโครงงาน

ในปัจจุบันเทคโนโลยีของโลกได้ถูกพัฒนาไปอย่างมาก การดำเนินชีวิตในปัจจุบันจึงต้องมีสิ่งอำนวยความสะดวกอยู่มากมายและหนึ่งในจำนวนนั้นจะมีบัตรแม่เหล็กรวมอยู่ด้วยในปัจจุบันถูกนำมาใช้อย่างกว้างขวางไม่ว่าจะเป็นบัตร ATM บัตรเครดิตต่างๆ บัตรเช็คเวลาเข้างานของคนงาน รวมทั้งบัตรสมาชิกต่างๆ แลอื่นๆอีกมาก

ดังนั้นโครงงานนี้จึงเล็งเห็นความสำคัญของบัตรแม่เหล็กซึ่งจะต้องมีบทบาทอย่างมากในอนาคตอย่างแน่นอนจึงได้คิดทำโครงงานนี้ขึ้น โดยเห็นว่าก่อนที่จะนำบัตรแม่เหล็กไปใช้งานได้นั้นจะต้องทำการบันทึกข้อมูลก่อนโดยรูปแบบการบันทึกขึ้นอยู่กับผู้สร้างและเมื่อบันทึก ลงบัตรแล้วจะต้องอ่านข้อมูลจากบัตรให้ได้ก่อนก่อนที่จะนำไปประยุกต์ใช้งานต่อไป จึงได้ทำโครงงานเครื่องอ่านแถบแม่เหล็กนี้ขึ้น โดยโครงงานนี้จะสร้างเครื่องอ่านและใช้การรูดบัตรด้วยมือซึ่งประหยัดกว่าการใช้มอเตอร์

3. ขอบเขตของโครงงาน

3.1 เครื่องอ่าน (MAGNETIC CARD READER)

3.1.1 ส่วนอ่านข้อมูลจากบัตรแม่เหล็กจะใช้หัวเทปแบบ MONO

3.1.2 ส่วนประมวลผลใช้ CPUเบอร์8085

3.1.3 ส่วนแสดงผลใช้ MONITOR ของ COMPUTER

3.1.4 สามารถต่อเครื่องอ่านกับ COMPUTER ได้โดยต่อพ่วงกับ KEYBOARD หรือเป็น Keyboard Simulated

4. ขั้นตอนและวิธีดำเนินงาน

ขั้นตอนการดำเนินงานแบ่งออกเป็น 2 ช่วง โดยในช่วงที่1 คือช่วงของ PROJECT 1 จะเป็นการดำเนินงานในส่วนของเครื่องอ่านบัตรแม่เหล็ก ดังรายละเอียดดังต่อไปนี้

- ศึกษาค้นคว้าข้อมูลในส่วนที่เกี่ยวข้องกับโครงงาน
- เช่น การใช้งาน CPU 8085
- มาตราฐานของบัตร ATM
- ออกแบบในส่วนของ HARD WARE ของเครื่องอ่าน
- ทดลองวงจรในส่วนต่าง ๆ
- ออกแบบในส่วนของ SOFT WARE ของเครื่องอ่าน

สำหรับในช่วงที่ 2 คือช่วงของ PROJECT 2 จะเป็นการดำเนินงานในส่วนของเครื่องอ่านแถบ แม่เหล็ก ดังรายละเอียดต่อไปนี้

- ศึกษาค้นคว้าข้อมูลในการนำเครื่องอ่านไปประยุกต์ใช้
- ออกแบบ SOFT WARE ของเครื่องอ่าน
- เขียนโปรแกรมรับข้อมูลจากบัตรและส่งข้อมูลไปยัง Computor
- เขียนโปรแกรมบน PC เพื่อรับข้อมูลมาเป็นฐานข้อมูลของผู้ถือบัตรโดยใช้ Visual Basic for Windows

บทที่ 2

ทฤษฎีและหลักการ

บัตรแม่เหล็กที่ใช้กันอยู่ในปัจจุบัน มีอยู่ด้วยกันมากมายหลายชนิดดังแสดงในรูปที่ 1 ในยุคแรก ๆบัตรแม่เหล็กจะมีเนื้อวัสดุเป็นพลาสติก PVC ชนิดแข็งและแบบที่มีเนื่อวัสดุเป็นกระดาษ แต่ในปัจจุบันได้พัฒนาเนื้อวัสดุของบัตรแม่เหล็กมาเป็น Polyethylene Therephthalate หรือที่เรียกย่อ ๆ ว่า PET

ตารางที่ 1 ชนิดและการใช้งานบัตรแม่เหล็ก

ชนิด	เนื้อวัสดุ	ความหนาของบัตร	การใช้งาน
บัตรพลาสติก (Plastic Card)	-PVC ชนิดแข็ง -PVCAชนิดแข็ง	0.76 ม.ม.	-บัตร ATM -บัตรเครดิต -บัตรประจำตัว (ID)
PET Card	- Polyethylene terephthalate (PET)	0.2-0.4 ม.ม.	-บัตรโทรศัพท์ -บัตรช็อปปิ้งต่าง ๆ (PREPAID CARD)
บัตรกระดาษ (Composite Paper Card)	- กระดาษอย่างดี - กระดาษเคลือบพลาสติก -กระดาษบันทึกข้อมูล	0.2-0.76 ม.ม.	-ตั๋วรถไฟ -ตั๋วทางด่วน -ตั๋วเครื่องบิน -บัตรโรงแรม

ในตารางที่ 2 จะเป็นตัวอย่างการใช้งานบัตรแม่เหล็ก

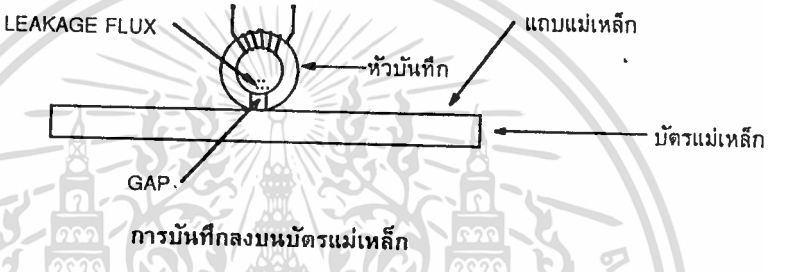
การใช้งาน	ชนิดของบัตร
การเงิน การธนาคาร	-บัตร ATM -บัตรเครดิต -บัตรเติมน้ำมัน -บัตรหลักทรัพย์ -บัตรช็อปปิ้ง
การคมนาคม	-ตั๋วรถไฟแม่เหล็ก -ตั๋วผ่านทางด่วน -ตั๋วเครื่องบิน -ตั๋วจอดรถ
การสื่อสาร	-บัตรโทรศัพท์
การสำนักงาน (OA,FA)	-บัตรประจำตัวพนักงาน -บัตรประจำตัวนักเรียน -บัตรเครื่องถ่ายเอกสาร - บัตรตรวจโรค - บัตรโรงแรม
การรักษาความปลอดภัย	- บัตรปิด - เปิดล็อค

คุณลักษณะของวงจรแม่เหล็ก

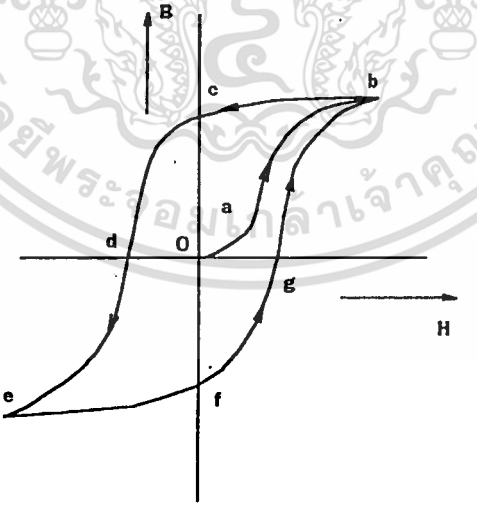
คุณลักษณะของวงจรแม่เหล็กสามารถแสดงได้ด้วยกราฟดังรูปที่2.2 โดยเป็นกา รพล็อทระหว่าง"ฟลักซ์แม่เหล็ก"(fluxdensity)"B"และ"สนามแม่เหล็ก"หรือMagnetizing force หรือ "H" ทั้งนี้ความสัมพันธ์ของ B-H เป็นความสัมพันธ์ระหว่างแรงดันกับกระแสฟลักซ์แม่เหล็กหมายถึงจำนวนเส้นแรงแม่เหล็กต่อตารางนิ้วที่ผ่านพื้นที่หน้าตัดของแกน ส่วน magnetizing force นั้นเท่ากับจำนวนแอมแปร์-รอบ(ampere-turn)ต่อนิ้วของความยาว แกนในการทำงานหากสนามแม่เหล็ก (H) มีค่าเพิ่มขึ้นทีละน้อยจาก

ศูนย์ฟลักซ์แม่เหล็ก (B)ก็ จะค่อยเพิ่มๆขึ้นเป็นสัดส่วนด้วย(0-a-b)จนกระทั่งถึงจุดอิ่มตัว ที่ b จากนั้นถ้าแรงดันตร่อมขดลวดตกลงเป็นศูนย์ฟลักซ์แม่เหล็ก B จะไม่กลับไปที่ศูนย์ แต่มันจะลดลงตามเส้นโค้งจาก b-c และถึงจุด c เมื่อ H กลายเป็นศูนย์โดยแมกนีจูด หรือขนาด 0-c

เราอาจกล่าวได้ว่า magnetic substance มี residual magnetism ภายหลังจาก ป้อน magnetizing force ให้กับมัน



รูปที่ 2.1 วงจรของหัวบันทึก



รูปที่ 2.2 B-H Curve

ถ้าหากลองป้อนกระแสหรือ H ในทิศทางตรงกันข้ามกับขดลวด ค่า B จะเป็นไปตามโค้งจาก c-d-e ที่จุด e เป็นจุดที่มีการอิ่มตัวตรงข้ามกันของวัสดุแม่เหล็กจะเห็นได้ว่าหากเราป้อน แรงดันไฟฟ้ากระแสสลับเข้าที่ขดลวดตามรูปที่ 2.1 แล้วค่า B จะเดินทางเกิดเป็นวงรอบ หรือลูป(loop)ขึ้นคือ จาก 0 -a-b-c-d-e-f-g-b เราเรียกลูปดังกล่าวว่า "ฮีสเตอร์รีซีส"

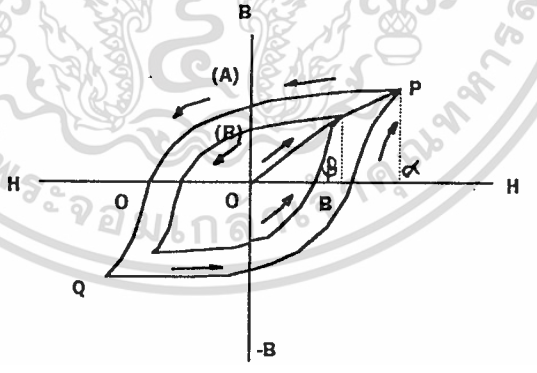
-ระยะระหว่างจุดตัดแกน (0) ถึง b เรียกว่า "magnetization curve ของวัสดุ"

-closed loop เรียกว่า "hysteresis loop"

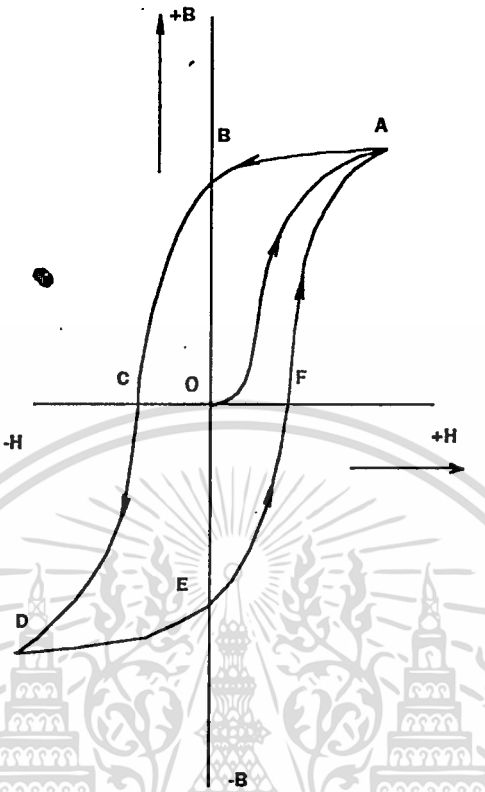
-ค่าของ flux density(B) จาก0 ถึง c เรียกว่า "retentivity ของ magnetic substance"

-ค่าของ magnetizing curve (H) จาก 0 ถึง d เรียกว่า "coereive force"

จากรูปที่ 2.2 นั้นหากปริมาณหรือขนาดของ H (ซึ่งป้อนเข้าไปครั้งแรก)หยุดที่จุดที่มีค่าต่ำ ๆ จุดหนึ่งแล้วกลับ จะมีผลทำ
ให้เกิดเป็นลูพเล็ก ๆ(เล็กกว่าครั้งแรก)ขึ้นดังแสดงในรูปที่ 2.3



รูปที่ 2.3 การเปลี่ยนแปลงเนื่องจากขนาดของ H



รูปที่ 2.4 ฮิสเทอริซิส ลูปของเหล็ก

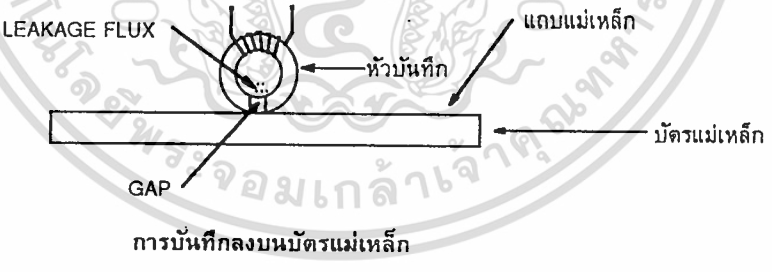
ในกรณีนี้หากป้อนแรงแม่เหล็กเข้าที่ตำแหน่งαแล้วแรงแม่เหล็ก (A)จะคงอยู่แม้เอาแรงแม่เหล็กออกแล้วก็ตาม และหากป้อนสนามแม่เหล็ก β เข้าไป(B)ก็จะคงอยู่ด้วยเช่นกัน อันนี้จึงเป็นการอธิบายถึงหลักการจำ(memory) ของการบันทึกข้อมูลด้วย hysteresis phenomenon "B-H characteristic"อีกนัยหนึ่งก็คือ ฟังก์ชั่นความจำ (memory function)ของการบันทึกนั้นก็คือเปลี่ยนแปลงรูปคลื่นของข้อมูลที่จะบันทึกให้อยู่ในรูป ของฟลักซ์แม่เหล็ก ในกรณีนี้จะกล่าวในเรื่องเครื่องบันทึกบัตรแม่เหล็กจากที่กล่าวมาข้างต้นเป็นทฤษฎีพื้นฐานซึ่งเป็นหลักการของเครื่องบันทึกเทปทั่ว ๆไปซึ่งนำมาเป็นแนวทางสำหรับเครื่องอ่านและบันทึกแถบแม่เหล็กเท่านั้นโดยลำดับต่อไปจะกล่าวถึงการบันทึกข้อมูลลงบนบัตรแม่เหล็ก

การบันทึกข้อมูลลงบนบัตรแม่เหล็ก

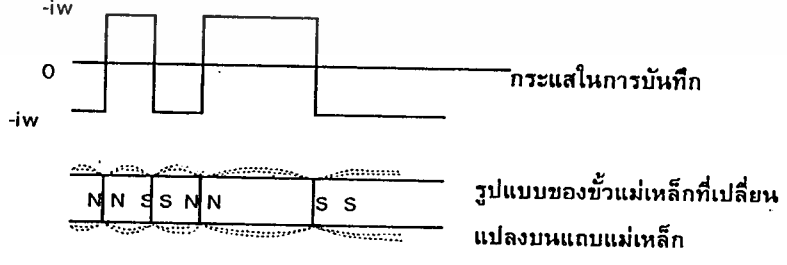
การบันทึกข้อมูลลงบนบัตรแม่เหล็กจะใช้วิธีการบันทึกแบบดิจิตอล. ในลักษณะเช่นเดียวกับที่ใช้ในแผ่นฟล็อปปี้ดิสก์ หรือ เทปแม่เหล็กสำหรับเครื่องคอมพิวเตอร์ทั่ว ๆ ไปนการบันทึกข้อมูลลงบนแถบแม่เหล็กนั้นจะต้องป้อนกระแสพัลซ์ซึ่งมีทั้งด้าน + และด้าน - พร้อมทั้งมีขนาดเพียงพอเข้าที่ขดลวดของหัวบันทึกซึ่งกดอยู่บนแถบแม่เหล็กที่เคลื่อนด้วยความเร็วคงที่ ดังในรูปที่ 2.5(a) แถบแม่เหล็กจะถูกเปลี่ยนให้มีรูปแบบของขั้วแหล็กตาม LEAKAGE FLUX จาก GAP ของหัวบันทึก ดังในรูปที่ 2.5(b)แถบแม่เหล็กจะเกิดเป็นแม่เหล็กถาวขนาดเล็กเรียงตัวกันตามขั้ว + หรือ - ของพัลซ์และความกว้างของพัลซ์สัญญาณที่บันทึกเนื่องจากกระแสพัลซ์ที่ใช้ในการบันทึกมีขนาดเพียงพอที่หัวบันทึก จะทำให้แถบแม่เหล็กมีสนามแม่เหล็กอิ่มตัวได้ดังนั้นเมื่อทำการบันทึกข้อมูล ตัวข้อมูลที่เคยมีอยู่จะถูกเขียนทับและหายไป เหลือเพียง ข้อมูลใหม่เท่านั้น

รูปแบบการบันทึกข้อมูล

รูปแบบการบันทึกข้อมูลลงบนบัตรแม่เหล็กส่วนใหญ่ จะใช้รูปแบบ F2F หมายถึง TWO FREQUENCY COHERENT ENCODING และ FM หมายถึง FREQUENCY MODULATION การบันทึกข้อมูลในลักษณะเช่นนี้จะบันทึกข้อมูล (DATA) และ CLOCK เข้าไว้ใน TRACK เดียวกันนอกเหนือจากนี้ยังมีการบันทึกที่ข้อมูลและ CLOCK แยกคนละ TRACK เช่นแบบ NRZI (NON ZERO INVERTED RECORDING)ซึ่งมีความจุในการบันทึกค่า



การบันทึกลงบนบัตรแม่เหล็ก



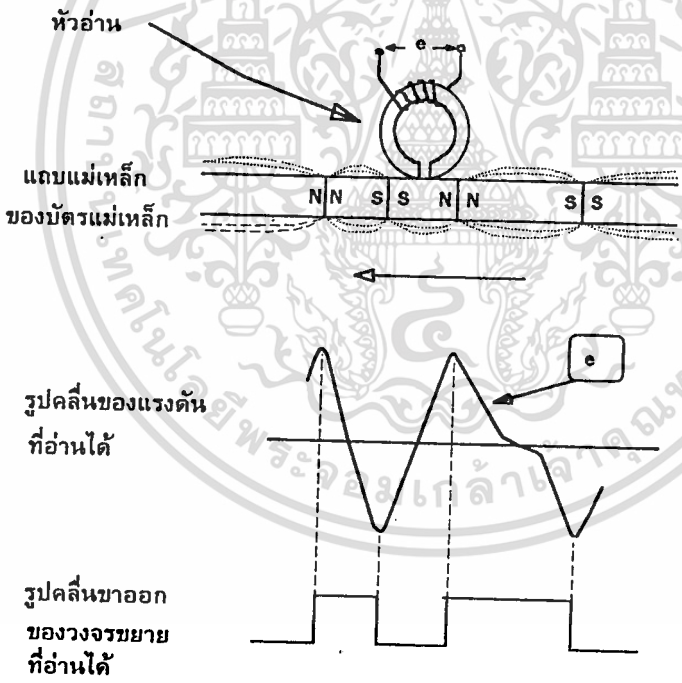
รูปแบบของขั้วแม่เหล็กที่บันทึก

รูปที่ 2.5 การบันทึกข้อมูล

การอ่านข้อมูลจากบัตรแม่เหล็ก

การอ่านข้อมูลจากบัตรแม่เหล็ก ทำได้โดยให้หัวอ่านสัมผัสกับแถบแม่เหล็กซึ่งเคลื่อนที่ด้วย ความเร็วคงที่ในรูปที่ 2.6 FLUX ที่เกิดจากแม่เหล็กถาวรขนาดเล็กบนแถบแม่เหล็กจะ ผ่านจาก GAP ของหัวอ่านไปยังแกน (CORE) การเปลี่ยนแปลงของ FLUX ตามข้อมูลนั้น

จุดสูงสุด (PEAK) ของแรงดันที่อ่านได้นั้นจะตรงกับจุดที่สนามแม่เหล็กบนแถบแม่เหล็กกลับ ทิศทางพอดี ดังนั้นถ้าขยายแรงดันนี้ขึ้นและตรวจหาจุดสูงสุด (PEAK) ด้วยวิธี DIFFERENTIAL แล้วเปลี่ยนเป็นสัญญาณพัลซ์ ก็จะได้ข้อมูลที่บันทึกอยู่ในบัตรแม่เหล็ก (รูปที่ 2.5)



รูปที่ 2.6 การอ่านข้อมูล

บทที่ 3

มาตรฐานบัตร ATM

มาตราฐาน ISO 7811 / 2-1985 (E)

เป็นมาตราฐานของบัตร ATM,บัตรเครดิต ที่ใช้กันอยู่ทั่วไปนั้นจะมีทั้งหมด 3 แทร็คโดยแทร็คที่ 2 จะเป็นแทร็คที่ใช้งานกันมากที่สุดซึ่งเป็นข้อมูลประจำตัวของผู้ถือบัตรจะมี ข้อมูลอยู่ได้สูงสุด 40 char/track มีรูปแบบดังนี้

sync	sync	start Bh	data 0-9h	sep Dh	data 0-9h	stop Fh	LRC	sync	sync

รูปที่ 3.1 รูปแบบของข้อมูลที่อยู่ในแทร็คที่ 2

Sync เป็นข้อมูลที่ใช้สำหรับเปรียบเทียบ ข้อมูลที่จะใช้งานและเป็นการแสดงการเริ่มต้น ข้อมูล

Start จะเป็นข้อมูล "Bh" โดยจะใช้ข้อมูล 4 ตัว Parity 1 ตัว

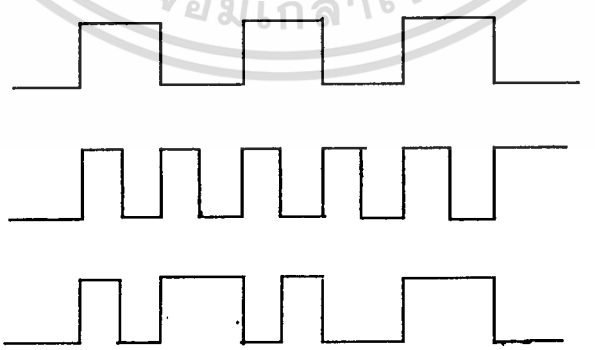
Data เป็นข้อมูลของบัตรโดยจะมีข้อมูล 4 ตัวParity 1 ตัวมีค่าตั้งแต่ 0-9h ความยาวจะแล้วแต่การใช้งานของแต่ละธนาคาร

Sep เป็นตัวเชื่อมหรือกั้นระหว่าง Data จะเป็นข้อมูล "Dh"

Stop เป็นตัวหยุดข้อมูลหรือให้รู้ว่าข้อมูลหมดแล้วจะเป็นข้อมูล "Fh"

LRC เป็นตัวตรวจสอบข้อมูลในแนวตั้ง

โดยในการบันทึกข้อมูลในบัตรแม่เหล็กนั้นจะถูกบันทึกแบบ FM mod โดยค่าที่เป็น "1 จะมีความถี่สูงเป็น 2เท่าของค่าที่เป็น "0" สามารถดูได้จาก Timming Diagram ดังรูปที่ 3.2



รูปที่ 3.2 timming diagram

ซึ่งบัตรแม่เหล็กที่ใช้ในโครงงานนี้เป็นไปตามมาตรฐานซึ่งบันทึกข้อมูลและสัญญาณนาฬิกา (Synchonous clock) อยู่ใน Track เดียวกัน หัวอ่านบัตรแม่เหล็กใช้เป็นหัวเทปแบบโมโน บัตรแม่เหล็กมีขนาด ความกว้าง 2.125 นิ้ว ความยาว 3.375 นิ้ว และความหนา 0.03 นิ้ว

โดยสัญญาณนาฬิกา ที่ได้นั้นจะไปทำการอ้างอิงเพื่ออ่านข้อมูล จากตารางที่ 1 จะแสดงให้เห็นถึงรหัสข้อมูล (Code Character) ซึ่งเป็น Format ของข้อมูลที่บันทึกลงบัตร จะเห็นว่ารหัสของข้อมูลจะเป็น Binary code ขนาด 4 bit และ Parity อีก 1 bit ซึ่งเป็น Parity แบบคี่ (Odd parity) ดังนั้นข้อมูล 1 ตัว (1Character) จึงประกอบด้วยรหัสเลขฐานสอง ขนาด 5 bit รหัสเหล่านี้ถูกนำมาเรียงต่อกันด้วย Start , Character , Separater , End sentinal และ Data character ประกอบกันเป็น Block ของข้อมูลที่ถูกบันทึกบนบัตรแม่เหล็ก ดังรูปที่ 3.3

ดังนั้นข้อมูลในแทร็คที่ 2 จะเป็นดังรูปที่ 3.3 ซึ่งมี sync เข้ามาก่อน 2 pulse แล้วก็ตามด้วย start ซึ่งมีข้อมูลเป็น Bh แล้วก็ตามด้วย data มีตั้งแต่ 0-9h และจะมีช่วงต่อระหว่าง data แต่ละ character คือข้อมูล Dh จนถึง data character สุดท้ายก็จะเป็น stop คือข้อมูล Fh แล้วก็จะมีการเช็คข้อมูลทางแนวตั้งแล้วเป็น sync เพื่อให้รู้ว่าข้อมูลที่ทำการอ่านนั้นหมดแล้ว

0B	01	02	13	04	15	16	07	08	19	1F

รูปที่ 3.3 ข้อมูลที่อยู่ในแทร็คที่ 2
แสดงลักษณะของข้อมูลที่ประกอบด้วย Start character **และ** End character

โดยในแทร็คที่ 2 นั้นข้อมูลที่จะบันทึกลงบนบัตรจริงๆแล้วจะมีเป็นตัวเลขตั้งแต่ 0-9h โดยข้อมูลจะมีทั้งหมด 4 bits และมี Parity รวมอยู่ด้วย 1 bits เมื่อรวมแล้วข้อมูล1 character จะมีทั้งหมด 5 bit ต่อ 1 ส่วนข้อมูลตั้งแต่ ah-fh จะเป็นข้อมูลที่ทำการเช็คซึ่งสามารถเขียนเป็นรายละเอียดได้ดังตารางที่ 1.

ซึ่งข้อมูลเหล่านี้จะถูก Process อีกครั้งด้วย Soft ware เพื่อจะแยกเอาข้อมูลที่ถูกต้องจริงอีกครั้งหนึ่ง ซึ่งจะได้กล่าวถึงวิธีการดึงข้อมูลซึ่งเป็นหมายเลขประจำตัวผู้ถือบัตรออกจาก Block ของข้อมูลอีกครั้งหนึ่งในส่วนของ Program และ Flowchart

ตารางรหัสสำหรับแทร็คที่ 2

bits						
p	b4	b3	b2	b1	row	char
1	0	0	0	0	0	0
0	0	0	0	1	1	1
0	0	0	1	0	2	2
1	0	0	1	1	3	3
0	0	1	0	0	4	4
1	0	1	0	1	5	5
1	0	1	1	0	6	6
0	0	1	1	1	7	7
0	1	0	0	0	8	8
1	1	0	0	1	9	9
1	1	0	1	0	10	A
0	1	0	1	1	11	B
1	1	1	0	0	12	C
0	1	1	0	1	13	D
0	1	1	1	0	14	E
1	1	1	1	1	15	F

จะเห็นว่ารหัสของข้อมูลจะประกอบด้วย รหัสเลขฐาน 2 ขนาด 5 bit ซึ่งเป็นรหัสของข้อมูลขนาด 4 bit และ Odd parity อีก 1 bit รวมเป็น 5 bit สำหรับเป็น Format 1 Character

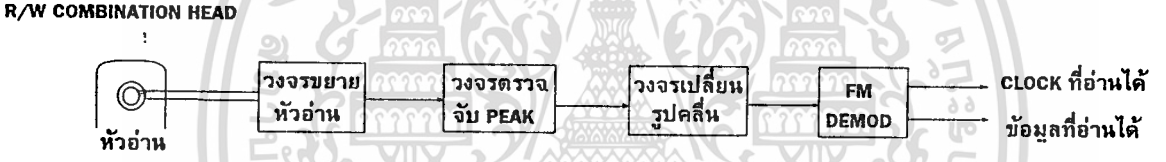
บทที่ 4
HARDWARE ของเครื่องอ่านบัตรแม่เหล็ก

โครงสร้างของระบบจะแบ่งเป็น 2 ส่วน คือ

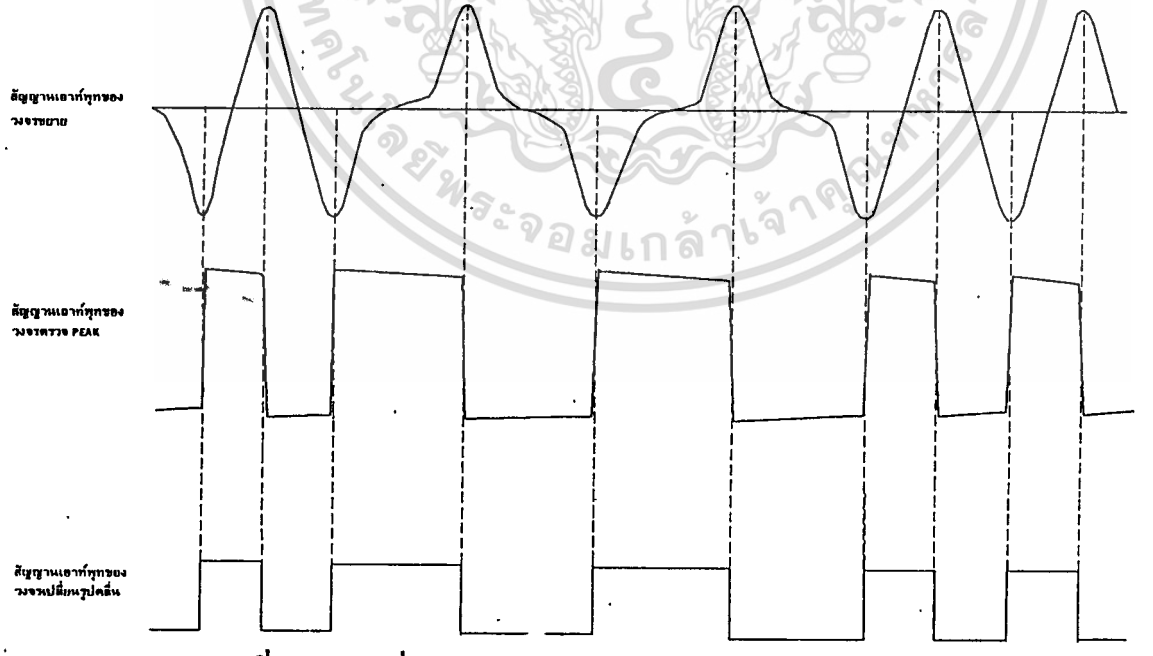
1. ส่วนของเครื่องอ่านบัตรแม่เหล็ก
2. ส่วนที่รับข้อมูลจากเครื่องอ่านบัตรแม่เหล็กแล้วมาProcess และ interface กับ computer

1. ส่วนของเครื่องอ่านบัตรแม่เหล็ก

ในส่วนของเครื่องอ่านบัตรแม่เหล็กจะมีบล็อกไดอะแกรมของเครื่องอ่านบัตรแม่เหล็ก จะเป็นดังรูปที่ 4.1 วงจรหลัก ของเครื่องประกอบด้วยหัวอ่านซึ่งจะจับสัญญานการอ่านที่วงจรตรวจจับ PEAK TO PEAK ที่ ตรวจพบ จะนำมาแปลงเป็นรูปพัลซ์ด้วยวงจรเปลี่ยนรูปคลื่นเพื่อให้ได้สัญญาณพัลซ์เหมือนที่บันทึกลงจะมีรูปดังรูปที่ 4.2



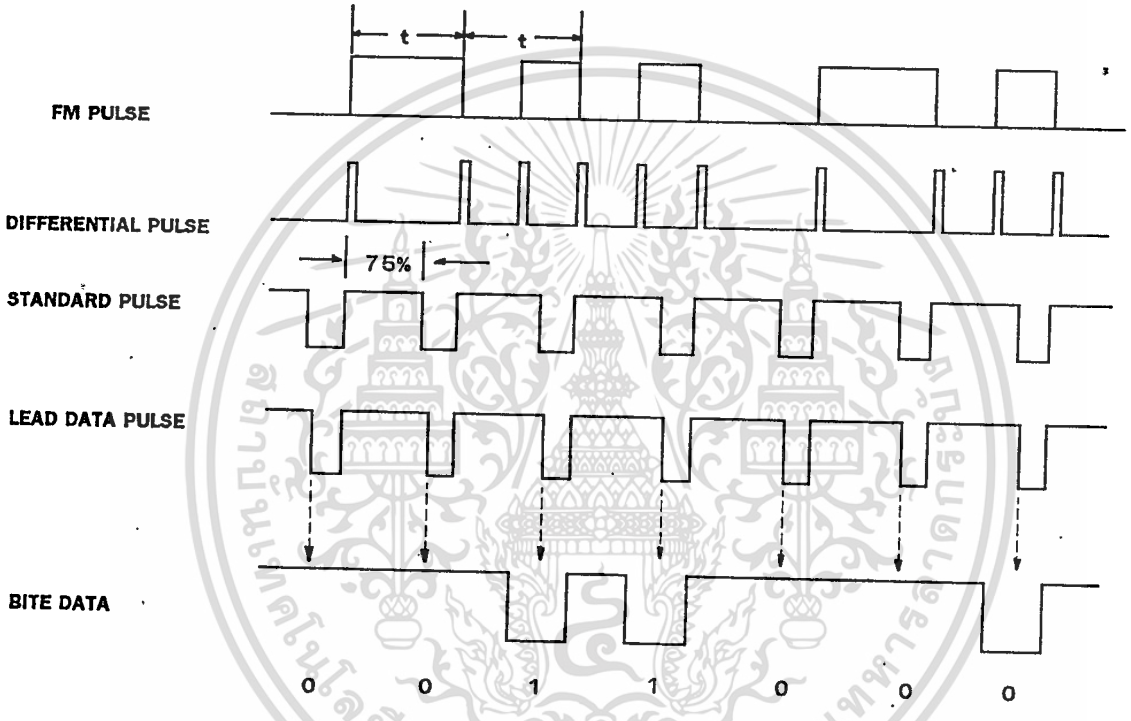
รูปที่ 4.1 BLOCK DIAGRAM ในส่วนของเครื่องอ่าน



รูปที่ 4.2 รูปคลื่นของการอ่านบัตรแม่เหล็ก

การถอดเฉพาะสัญญานข้อมูลออกมาจากสัญญานพัลซ์ที่ได้นี้จะต้องตรวจดูว่ามีการกลับขั้วของ pulse FM ในช่วง Standard Time ซึ่งมีค่าประมาณ 75%ของความยาว 1 bit (75% ของเวลา)หรือไม่ ถ้าไม่มีค่าของข้อมูลจะเป็น "0" แต่ถ้ามีค่าของข้อมูลจะเป็น "1"

การแยกสัญญานข้อมูลออกจากสัญญาน Clock จะใช้วงจร Demodulator โดยมี Time Chart ของ FM Demodulation ดังรูปที่ 4.3



รูปที่ 4.3 Time Chart ของ FM Demodulation

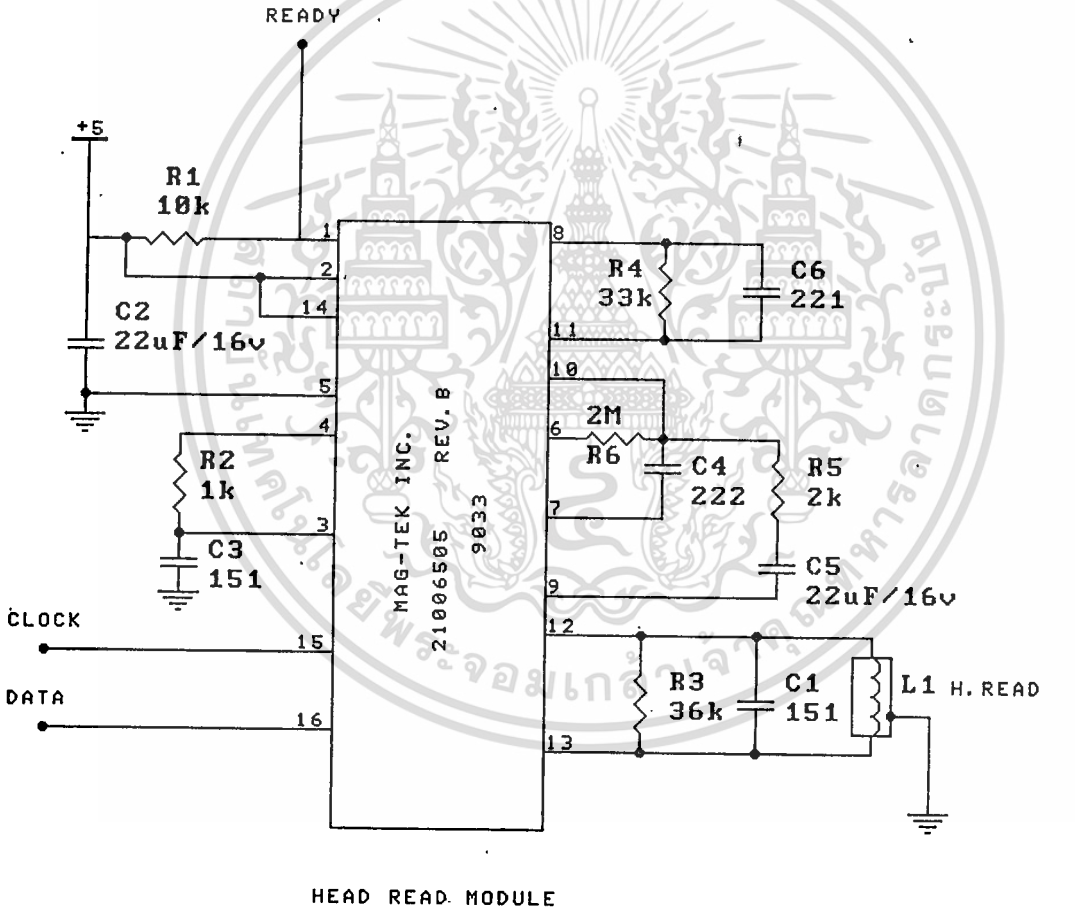
การสร้าง Standard Time เพื่อแยกข้อมูลออกมามี 2 แบบคือ แบบความยาวคงที่และแบบความยาวเปลี่ยนแปลงได้

แบบความยาวคงที่จะกำหนดStandardTimeจะให้ค่าคงที่และเปรียบกับทุกบิท ส่วนแบบที่ความยาวเปลี่ยนแปลงได้ จะวัดความยาวของบิทก่อนหน้าและสร้าง Standard time จากจุดนั้น แล้วนำไปเปรียบเทียบกับบิทถัดไป

สำหรับการสร้าง Standard time แบบเปลี่ยนแปลงความยาวได้นั้น ขณะที่กำลังอ่านบัตรอยู่ ถ้าความเร็วในการป้อนบัตรเปลี่ยนแปลง Standard time จะเปลี่ยน

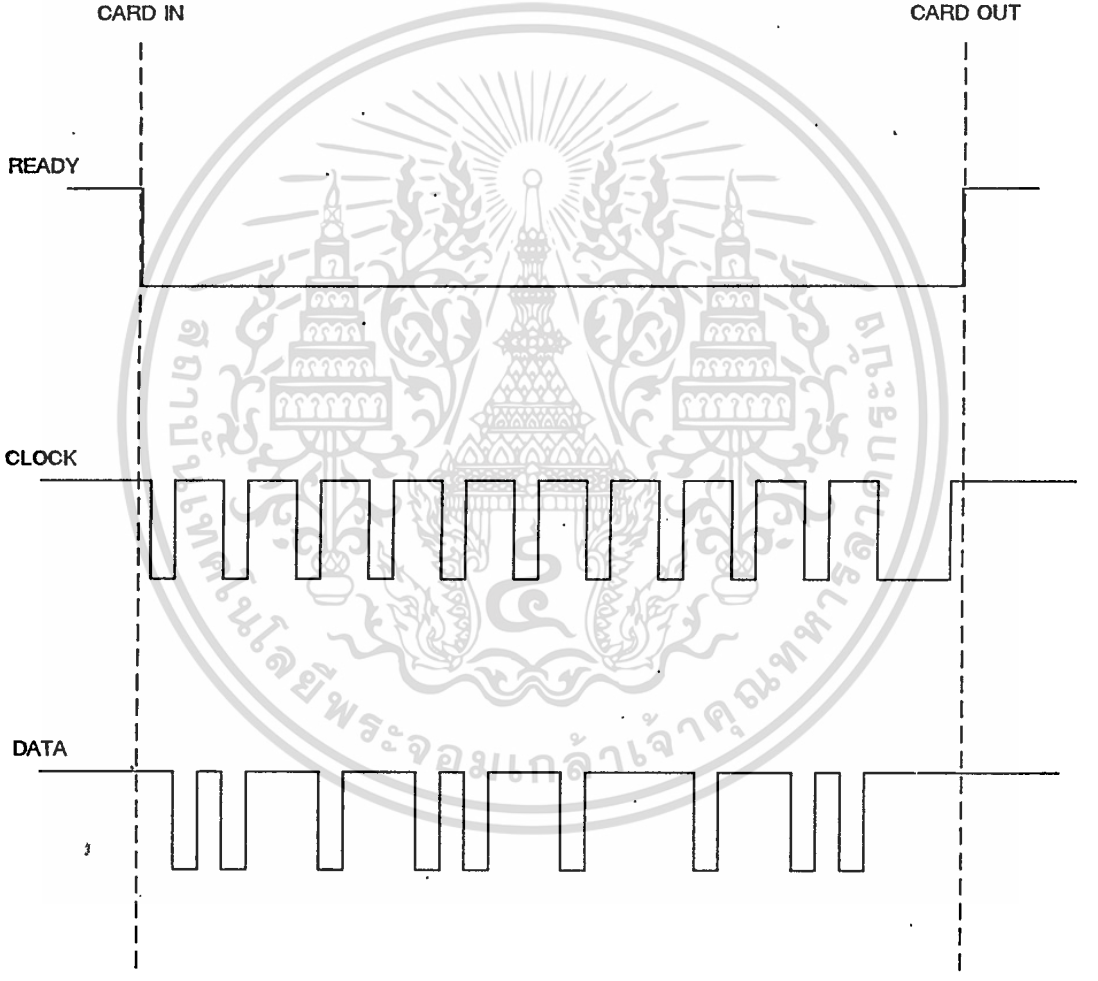
แปลงตามไปด้วยจึงใช้วิธีการนี้กับเครื่องอ่านบัตรที่ป้อนบัตรด้วยมือ เนื่องจากความเร็วในการรูดบัตรด้วยมือนั้นมีค่าไม่แน่นอน

จาก Block diagram คือวงจรขยายหัวอ่าน , วงจรตรวจจับ Peak , วงจรเปลี่ยนรูปคลื่น , FM Demodulation สามารถใช้ IC เบอร์ 9033 ในการสร้างสัญญาณ Data และ Clock ออกมาดังวงจรรูปข้างล่าง แล้วเอาสัญญาณนี้ไป Process เพื่อหาข้อมูลที่อยู่ในบัตรโดยใช้ CPU 8085 ในการ Process



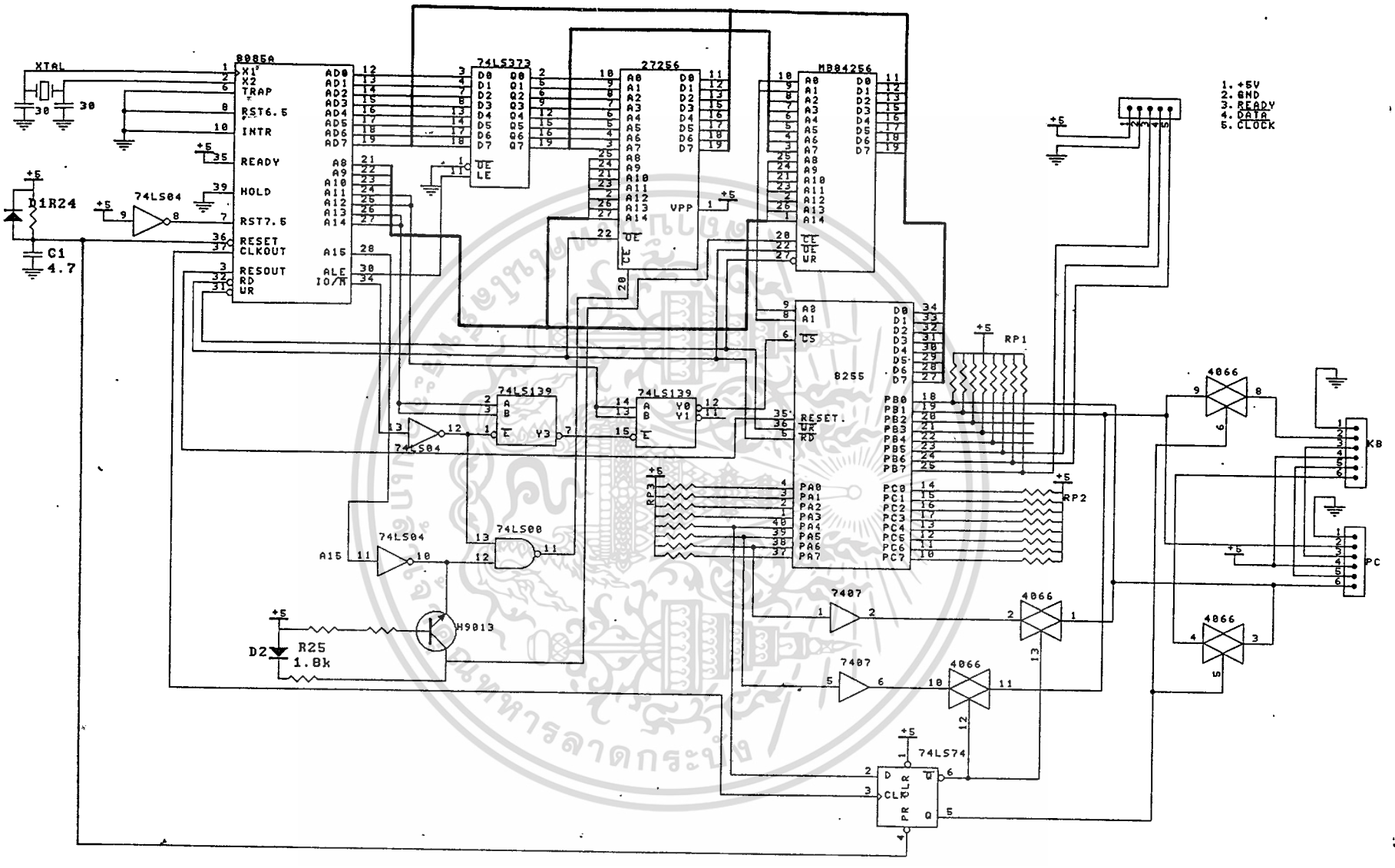
รูปที่ 4.4 วงจรสร้างสัญญาณ CLOCK & DATA

จากวงจรจะมี ขา16เป็น Data ขา15เป็นสัญญาณ Clock และขา 1เป็น Ready ซึ่งสัญญาณทั้งหมดนี้มี Data,Clock,Ready ซึ่งทั้งหมดจะถูกนำเข้าไปยังชุดประมวลผลต่อไป ดังนั้นสัญญาณที่จะนำไปประมวลผลมีอยู่สามสัญญาณคือ Ready, Clock, Data ดังรูปข้างล่างนี้ ซึ่ง Ready คือสัญญาณเพื่อบอกให้รู้ว่ามีบัตรเข้ามาทำการอ่าน ส่วน Clock ก็คือ Standard time ที่ใช้สำหรับเป็น Syncronous signal ในการอ่านข้อมูลจากบัตร ส่วน Data ก็คือข้อมูลที่อ่านได้จากบัตร



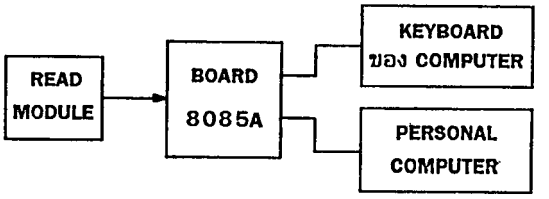
รูปที่ 4.5 รูปร่างของสัญญาณ READY,CLOCK,DATA

ส่วนที่รับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก แล้วนำมา Process และ Interface กับ Computer ในส่วนนี้ใช้ CPU 8085A ในการประมวลผล รายละเอียดภายในวงจรดังแสดงในรูป 4.6



รูปที่ 4.6 วงจรประมวลผลของเครื่องอ่าน

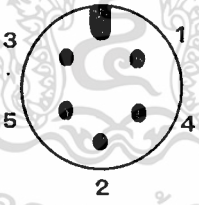
เมื่อเอาชุดเครื่องอ่านบัตรแม่เหล็กต่อร่วมกับชุดประมวลผล และต่อกับ Computer จะแสดงดัง Block Diagram รูปที่ 4.7



รูปที่ 4.7 Block diagram ของเครื่องอ่านเมื่อต่อกับชุด Control และ Computer

ในรูปที่ 4.6 ส่วน Input ที่รับมาจากเครื่องอ่านบัตรแม่เหล็กจะมาเข้าที่ Port B ของ IC 8255 ส่วนในการติดต่อกับ PC และ Keyboard ใช้ Port A ของ 8255 ในการควบคุมและส่งข้อมูลออกไปยัง PC โดยจะควบคุมผ่าน D F/F ไปควบคุม Buffer ซึ่งใช้ IC เบอร์ 14066 เพื่อส่งข้อมูลไปยัง PC

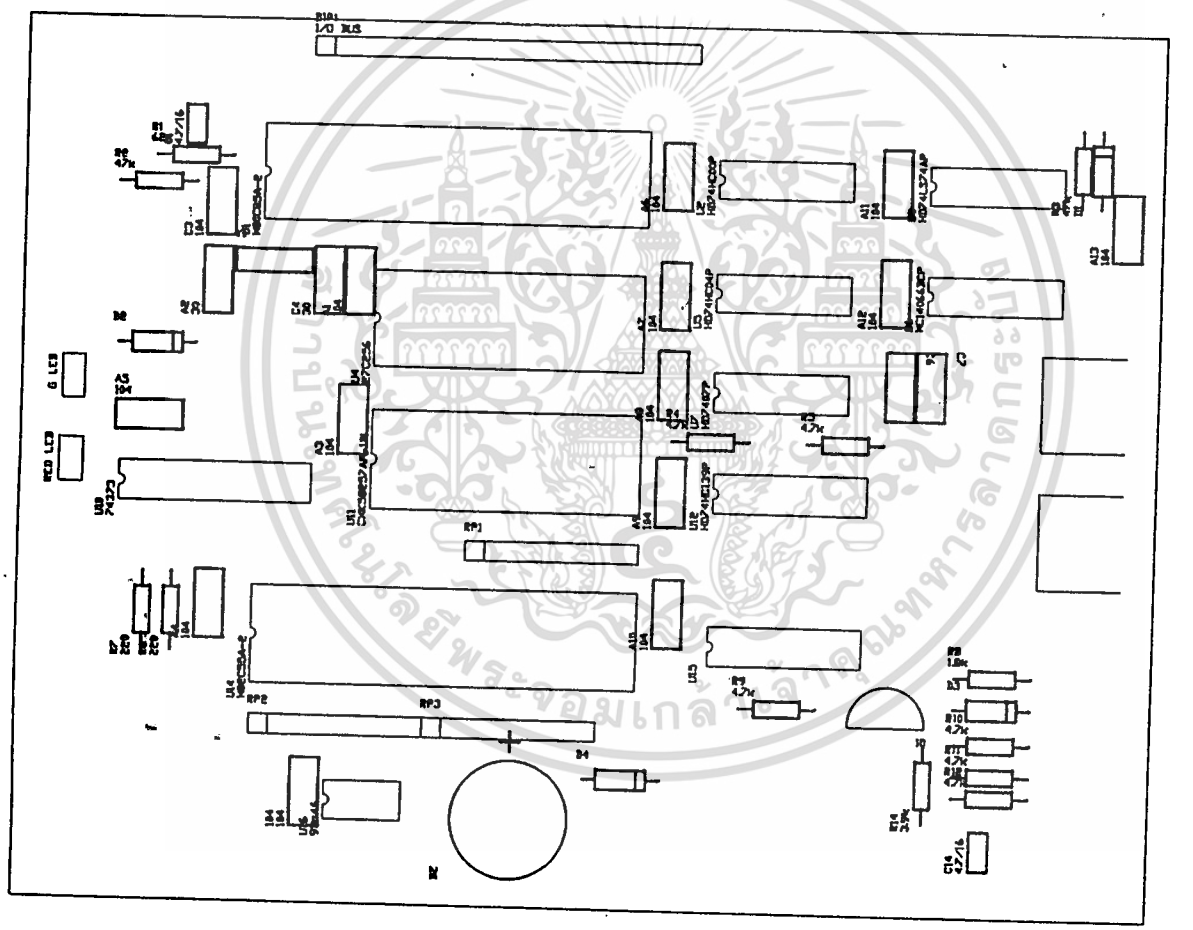
จากวงจรในส่วนของการต่อกับ PC และ Keyboard ก็คือเอาชุด Control ตัวนี้ต่อพ่วงกับ Keyboard เหมือนเป็นตัว Keyboard Simulated ซึ่งสายสัญญาณของ Keyboard ที่สำคัญมี ไฟบวก 5 โวลท์ , กราวด์ , สายสัญญาณข้อมูล , สายสัญญาณนาฬิกา ซึ่งจะมี Diagram ดังรูปที่ 4.8



CONNECTOR PIN	SIGNAL NAME
1	CLOCK
2	DATA
3	SPARE
4	GROUND
5	+5 Vdc

รูปที่ 4.8 ตำแหน่งของสัญญาณ

จากรูปที่ 4.8 ขา 3, 4, 5 ของ Key board จะต่อตรงไปยัง PC ซึ่ง Board control ก็ใช้ไฟ +5V และ Ground จาก PC ที่จ่ายมายัง Keyboard ด้วย ส่วนขา 1 กับ 2 ซึ่งเป็นสัญญาณ Clock และ Data ผ่าน Buffer ใช้ IC เบอร์ MC 14066 ซึ่งเราสามารถที่จะใช้ Keyboard ได้เป็นปกติหรือรับข้อมูลจากบัตรไปแสดงผลยัง PC ก็ได้ซึ่งรูปแบบการส่งข้อมูลที่อ่านมาจากเครื่องอ่านบัตรแม่เหล็กจะเหมือนกับการส่งข้อมูลจาก Keyboard ไปยัง PC ซึ่งจะแสดงรายละเอียดในส่วนของ Software ส่วนรายละเอียดของชุด Board Control จะมีรายละเอียดของตำแหน่งอุปกรณ์ดังรูปที่ 4.9



รูปที่ 4.9 ตำแหน่งอุปกรณ์บน Board

บทที่ 5

SOFT WARE ของเครื่องอ่านบัตรแม่เหล็ก

การออกแบบในส่วนของภาครับข้อมูลจากบัตรบันทึกแม่เหล็กเข้ามาเพื่อนำมา Process นั้นจะเริ่มต้นจากการออกแบบวงจรที่จะรับข้อมูลจากหัวอ่านบัตรบันทึกเสียก่อน เนื่องจากเราทราบว่าในการอ่านข้อมูลจากบัตรบันทึกแม่เหล็กนั้นเราจะได้สัญญาณออกมา 2 ตัวด้วยกันคือ ข้อมูลและสัญญาน นาฬิกาโดยเราต้องการนำข้อมูลทั้งหมดที่ ได้จากบัตรมาทำการเก็บลงในหน่วยความจำ ซึ่งหน่วยความจำที่ใช้ใน 1 Address จะมีขนาด 8 Bits และที่ตัวหัวอ่านบัตรจะมีขาสัญญาน Card Presend คือ สัญญาน Ready จะActive เมื่อเริ่มมีบัตรเข้ามา เมื่อเริ่มมีการออกแบบข้อมูลจากบัตรบันทึกแม่เหล็กจากสิ่งที่เราทราบเหล่านี้ทำให้เราต้องแบบวงจรที่จะรับข้อมูลที่ออกจากหัวอ่านบัตรบันทึกแม่เหล็กทีละ 1 Bit เข้ามาเก็บไว้จนครบ 8 Bit แล้วจึงนำข้อมูลทั้ง 8 Bit ที่ได้เข้ามายังหน่วยความจำโดยผ่าน Data Bus 8 Bits ซึ่งควบคุมการทำงานโดย CPU ในการทำงานตอนแรกนั้น จะต้องทำการตรวจสอบ Card Presend ก่อน

จากนั้นทำการศึกษาถึงลักษณะของการเก็บข้อมูลที่อยู่บน บัตรบันทึกแม่เหล็กเพื่อทราบถึงรูปแบบของข้อมูลที่จะนำไปใช้ ในการนำเอาข้อมูลที่จะใช้ในการ Process เข้ามาได้อย่างถูกต้อง หลังจากที่ได้ทำการทดลองใช้ Storage Scope มาทำการจับสัญญาณจากขา Strobe (clock สัญญาณนาฬิกา) และค่า Data (ข้อมูล) ก็จะได้ออกมาโดยมี Ch1 เป็นสัญญาณ Strobe และ Ch2 เป็นสัญญานของ Data ซึ่งจากข้อมูลที่ได้นั้นยังไม่สามารถเห็นได้อย่างชัดเจนว่าข้อมูลที่ได้มีลักษณะเป็นอย่างไรทั้งนี้เนื่องจากว่าข้อมูลที่จะได้ออกมานั้นจะเป็นจังหวะที่ Strobe Active คือเป็น Low จึงจะนำเอาสัญญาณที่ขา Data ออกมาโดย ข้อมูลที่ได้นั้นจะกลับกันด้วยเนื่องจากขา Data Active Low ก่อนนำมา Process จึงต้องมีการนำมา Invert เสียก่อน

จากสัญญานที่วัดได้จะมีลักษณะคือในตอนแรกจะเป็น “1” ซึ่งเมื่อ Invert แล้วก็คือ “0” มาก่อนซึ่งเราไม่สามารถควบคุมได้แต่เมื่อมีข้อมูลเป็น “0” ตัวแรกซึ่งเมื่อ Invert แล้วก็คือ “1” ตัวแรกนั้นถ้าทำการตรวจสอบ จะได้ว่า

	Code0	Code1	Code2	Code3	Code4	Code5	Code6	Code7
00000000	11010	10000	01000	11001	00100	10101	01101	11100

Code8	Code9	Code10	Code11	Code12	
00010	10011	00001	11111	10101	00000000

Code0 : 11010 เมื่อ Rotate กลับด้านกันจะได้ 01011 ซึ่งก็คือ 0BH
Code1 : 10000 เมื่อ Rotate กลับด้านกันจะได้ 00001 ซึ่งก็คือ 01H
Code2 : 01000 เมื่อ Rotate กลับด้านกันจะได้ 00010 ซึ่งก็คือ 02H
Code3 : 11001 เมื่อ Rotate กลับด้านกันจะได้ 10011 ซึ่งก็คือ 13H
Code4 : 00100 เมื่อ Rotate กลับด้านกันจะได้ 00100 ซึ่งก็คือ 04H
Code5 : 10101 เมื่อ Rotate กลับด้านกันจะได้ 10101 ซึ่งก็คือ 15H
Code6 : 01101 เมื่อ Rotate กลับด้านกันจะได้ 10110 ซึ่งก็คือ 16H
Code7 : 11100 เมื่อ Rotate กลับด้านกันจะได้ 00111 ซึ่งก็คือ 07H
Code8 : 00010 เมื่อ Rotate กลับด้านกันจะได้ 01000 ซึ่งก็คือ 08H
Code9 : 10011 เมื่อ Rotate กลับด้านกันจะได้ 11001 ซึ่งก็คือ 19H
Code10 : 00001 เมื่อ Rotate กลับด้านกันจะได้ 10000 ซึ่งก็คือ 10H
Code11 : 11111 เมื่อ Rotate กลับด้านกันจะได้ 11111 ซึ่งก็คือ 1FH
Code12 : 10101 เมื่อ Rotate กลับด้านกันจะได้ 10101 ซึ่งก็คือ 15H

ส่วน Data นอกนั้นจะเป็น "0" หมด ซึ่งเราจะไม่นำมาพิจารณาด้วยดังนั้นจึงพอจะเห็นได้ชัดขึ้นถึงรูปแบบของข้อมูลที่เก็บในบัตรบันทึกแม่เหล็กว่ามีข้อมูลที่จะใช้เริ่มต้นด้วย 0BH และจบท้ายด้วยข้อมูล 1FH ส่วน Code12 15H นั้นไม่ได้ใช้ในการทำงานเนื่องจากชุดของข้อมูลจะสิ้นสุดที่ 1FH ข้อมูลที่อยู่หลังจาก 1FH ไปแล้วไม่ถือว่าเป็นข้อมูลซึ่งเราจะไม่นำมา Process ซึ่งในทดลองนี้ได้ทำการทดลองหลายครั้งซึ่งจะได้ผลการทดลองคือสัญญาณ READY, CLOCK, DATA เหมือนรูปในบทที่ 4 คือรูปที่ 4.5

แนวความคิดในการออกแบบซอร์ฟแวร์

ซึ่งจะกล่าวถึงขั้นตอนการประมวลผล เริ่มต้นจากการต่อเครื่องเข้ากับคอมพิวเตอร์จนถึงการส่งข้อมูลไปยังคอมพิวเตอร์รวมถึงฐานข้อมูลบนคอมพิวเตอร์ โดยจะอธิบาย ในรูปของ Flowchart

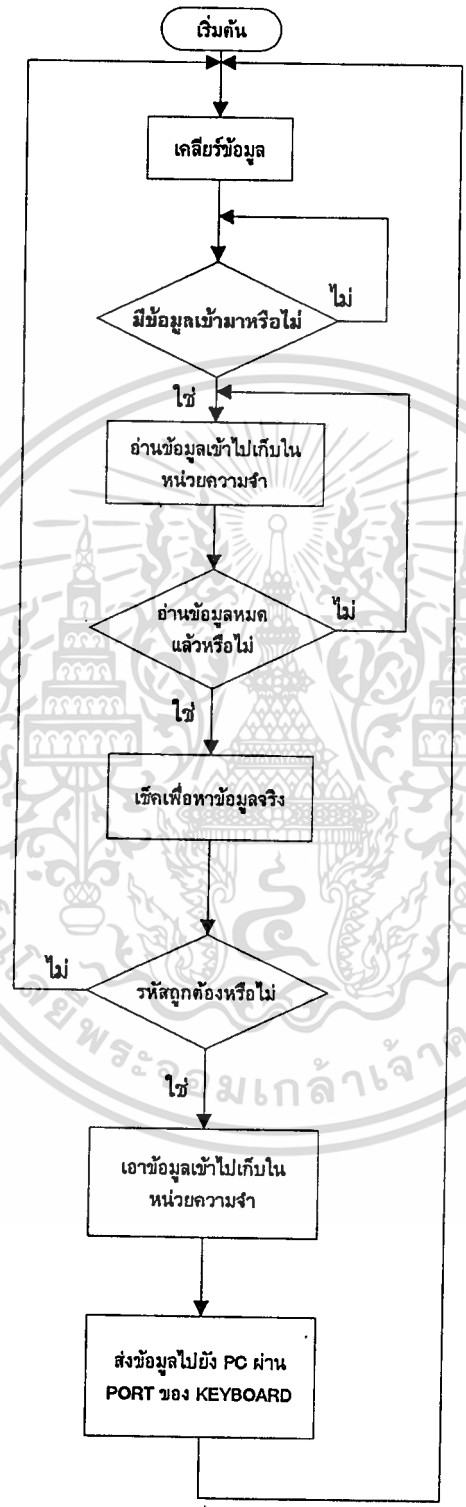
1. ช่วงรอรับบัตร

2. ค้นหา 0BH ซึ่งเป็นรหัสตัวแรกของข้อมูลบนบัตรซึ่งก็คือ Start Character

3. ส่วนของการเรียงข้อมูลที่อ่านเข้ามาได้และตรวจสอบ Error ของการอ่านข้อมูล โดยการเช็ค Parity

4. รูปแบบการส่งข้อมูลไปยัง PC (Personal Computer)

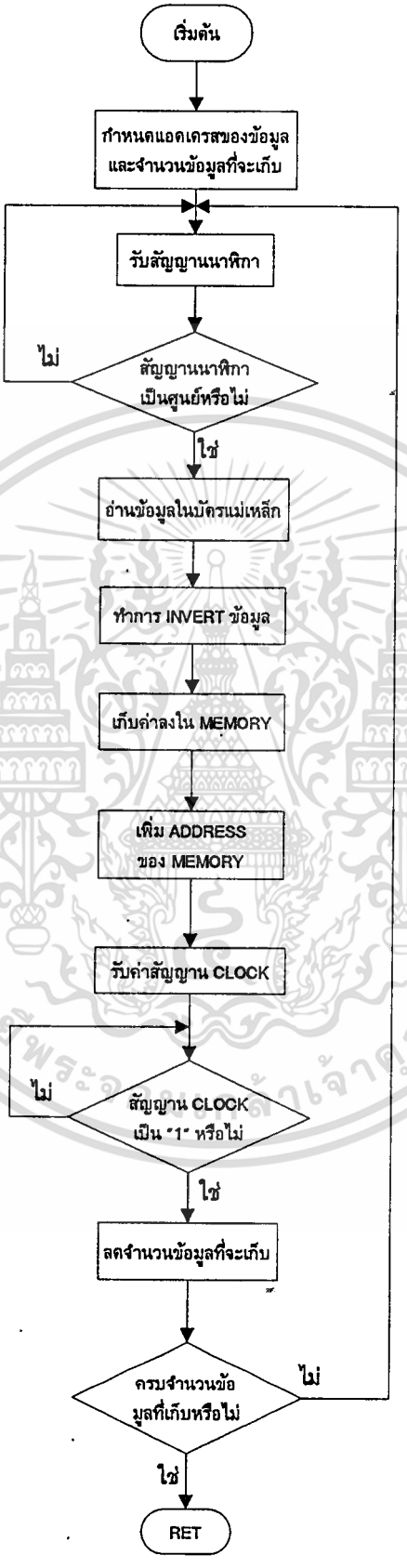
1. ช่วงการรอรับบัตร เมื่อมีการรูดบัตรแม่เหล็ก เราจะทำการเก็บข้อมูลที่อ่านได้โดยจะเก็บข้อมูลเป็นแบบตำแหน่ง ๆ ละ 1Bit ข้อมูลที่เราจะทำการเก็บมีทั้งหมด 60 ตำแหน่ง จากการทดลองอ่านข้อมูลจากสตอเรจ ออสซิลโลสโคป เราจะพบว่าเมื่อสัญญาณนาฬิกา มีระดับเป็น 0 เราจะได้ข้อมูล ที่ตำแหน่งนั้นออกมาจากนั้นเราก็ต้องทำการ กลับข้อมูลนั้นอีกทีเช่น ถ้าข้อมูลมีค่าเป็น 0 จะต้องกลับข้อมูลที่ได้ออกมานั้นให้มีค่าเป็น 1 เมื่อได้ข้อมูล เราก็จะเก็บค่าข้อมูลที่ได้มานั้นไว้ในตำแหน่งของหน่วยความจำตำแหน่งละ 1 ข้อมูล โดยตามตำแหน่งที่ชี้ไว้ในรีจิสเตอร์ HL ต่อมาก็ต้องมีการอ่านค่าสัญญาณอีกครั้งหนึ่งว่าสัญญาณนาฬิกามีการเปลี่ยนระดับเป็น 1 แล้วถ้าพบว่าสัญญาณนาฬิกามีค่าเป็น 1 ก็จะลดค่าใน ตัวนับลง และวนกลับไปอ่านข้อมูลเข้ามาใหม่จนกว่าจะเก็บข้อมูลได้หมดจากการทดลองเราพบว่า การรูดบัตรแม่เหล็ก 1 ครั้ง สัญญาณนาฬิกาที่ได้จะมีความถี่ประมาณ 200 Hz ดังนั้นจากการที่เราใช้ตัวประมวลผลเบอร์ 8085A ซึ่งความเร็วในการประมวลผลสูงจึงไม่มีปัญหาเรื่องเวลาในการรูดบัตรแม่เหล็ก ซึ่งจะมีส่วนของ โปรแกรมหลักดังรูปที่ 5.1



รูปที่ 5.1 FLOWCHART ของการทำงานหลัก

โปรแกรมหลักของระบบ

เริ่มต้นจาก Flowchart เราจะต้องมีการเคลียร์ข้อมูลจากหน่วยความจำเสียก่อนแล้วเช็คว่ามีข้อมูลเข้ามาหรือไม่ถ้าไม่ก็วนไปเช็คว่ามีข้อมูลเข้ามาหรือเปล่าถ้าไม่ก็จะวนอยู่อย่างนี้เรื่อย ๆ แต่ถ้ามีก็จะทำการอ่านข้อมูลจากจากบัตรมาเก็บไว้ในหน่วยความจำแล้วเช็คว่าอ่านข้อมูลหมดแล้วหรือยังถ้ายังก็วนอ่านไปเรื่อย ๆ ถ้าอ่านข้อมูลหมดแล้วก็จะทำการเช็คข้อมูลถ้าไม่ถูกต้องก็จะการกลับไปเคลียร์ข้อมูลหน่วยความจำใหม่ ถ้าถูกต้องก็จะนำข้อมูลไปเก็บยังหน่วยความจำก่อนที่จะทำการส่งข้อมูลไปยัง PC (Personal Computer)

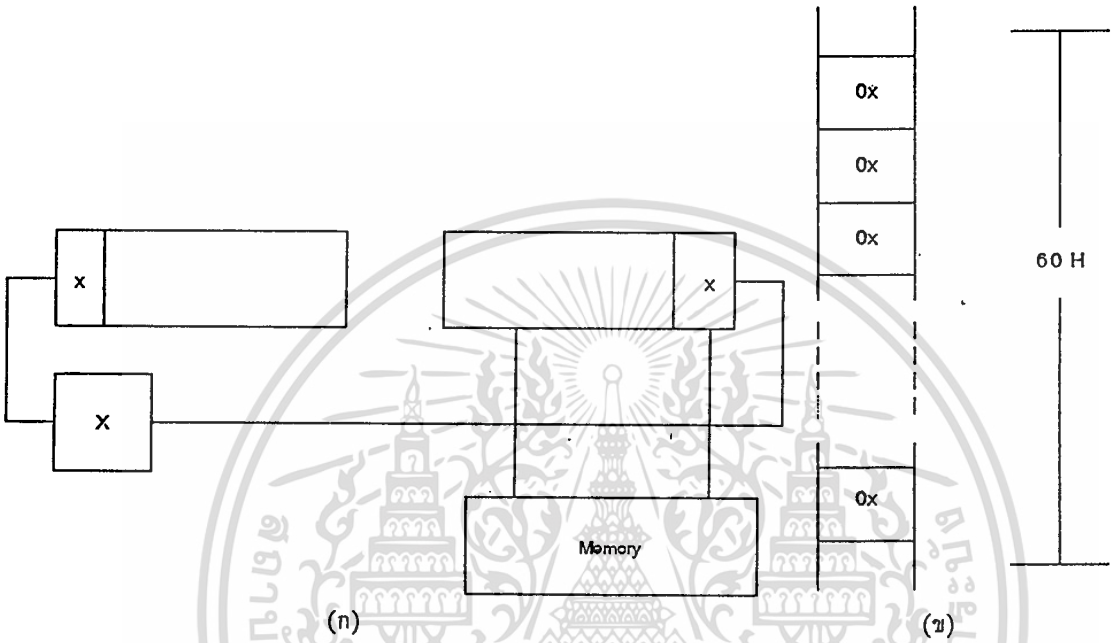


รูปที่ 5.2 FLOWCHARTของโปรแกรมรับข้อมูลจากบัตรแม่เหล็ก

โปรแกรมการรับข้อมูลจากบัตรแม่เหล็ก

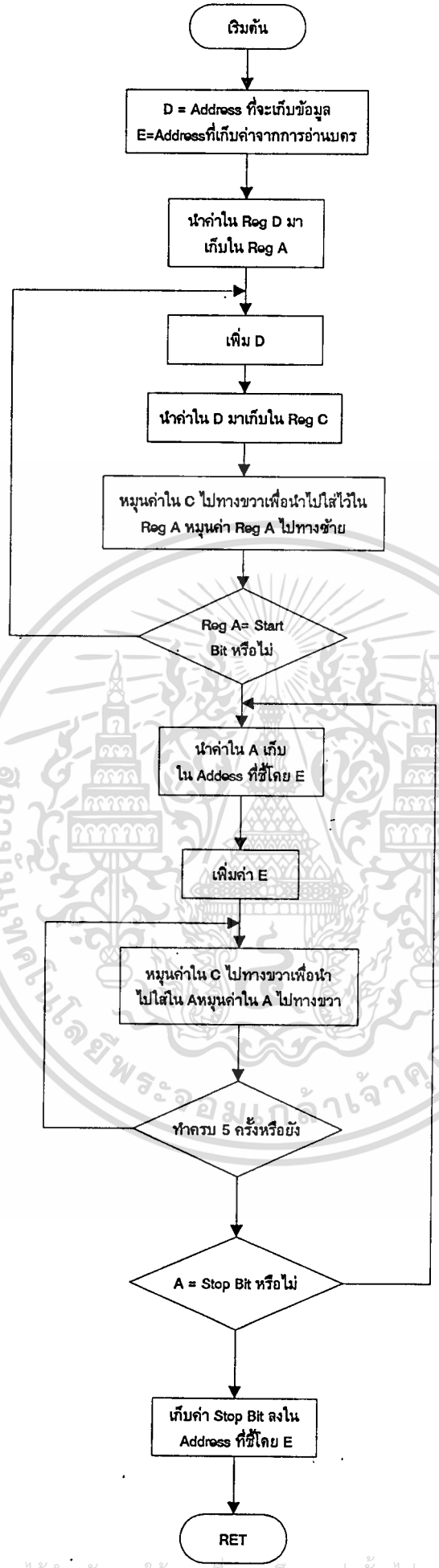
เริ่มต้นด้วยการกำหนด Address ที่จะเก็บข้อมูลและกำหนดจำนวนข้อมูลที่จะเก็บจากหัวอ่านบัตรแม่เหล็กเราจะรับข้อมูลเข้ามาเก็บไว้ทีละบิทโดยจำนวนข้อมูลกำหนดสูงสุด 60 บิท จากนั้นจะรอรับสัญญาณนาฬิกาซึ่ง Active ที่ Logic "0" ถ้าสัญญาณนาฬิกาเป็น 0 แสดงว่าเริ่มมีการรูดบัตรแล้วก็จะทำการอ่านข้อมูลจากขา Data ของหัวอ่านเข้ามาโดย Data นี้เราจะต้องทำการอินเวอร์ตค่าเสียก่อนและเก็บ Data ลงใน Memory สัญญาณนาฬิกานี้จะมาพร้อมกับ Data เสมอ เราจึงตรวจสอบว่าสัญญาณนาฬิกากลับเป็น Logic "1" จึงรับข้อมูลเข้ามา โปรแกรมจะทำงานตามนี้จนกว่าจะเก็บข้อมูลได้ครบ 60 บิท โดยข้อมูลแต่ละบิทจะเก็บลงในแต่ละแอดเดรส

2. การค้นหาบิทเริ่มต้น โดยการเอาตำแหน่งของข้อมูลที่รับข้อมูลเข้ามาเก็บไว้เป็นตัวอ้างอิงแล้วนำข้อมูลที่อยู่ใน ตำแหน่งข้อมูลนั้นออกมาแล้วทำการ หมุนข้อมูลเพื่อทำการหาข้อมูลที่เป็นตัวเริ่มต้น



รูป 5.3 ก) แสดงลักษณะการ Rotate ข้อมูล
ข) แสดงจำนวนตำแหน่งข้อมูลที่ใช้

ซึ่งที่เราต้องการหาก็คือ Start Character จะทำได้โดยการย้ายข้อมูลที่นำไปเก็บใหม่ทีละ Address มา Rotate Right นำข้อมูลทีละ Bit เข้ามาต่อกันใน Register แล้ว AND ด้วยค่า 11FH เพื่อให้เหลือข้อมูลเพียง 5Bit แล้วทำการเปรียบเทียบว่าคือ 1AH หรือไม่(ซึ่งก็คือ 0BH แต่ข้อมูลยังกลับกันอยู่จึงทำให้ต้องการตรวจสอบหา Start Character ด้วยค่า 1AH 11010 <---> 01011 ถ้ายังไม่พบจะทำการ Rotate อีก แล้วตรวจสอบใหม่อีกถ้าไม่พบก็จะทำเช่นนี้ซ้ำไปเรื่อย ๆ จนกว่าจะพบข้อมูลที่เราต้องการคือ 0BH แสดงได้ว่าถ้าทำซ้ำแบบเดิมต่อไปอีกทุก ๆ ถ Bit หมายถึง 1 Character ดังนั้นจึงทำเหมือนเดิมแต่นำค่าที่ได้ทุก ๆ 5 Bit ไปเก็บไว้ยัง หน่วยความจำ และจะทำการตรวจดูข้อมูลก่อนนำไปเก็บยังหน่วยความจำว่ามีค่าเป็น 1FH หรือยังถ้าใช่ก็จะพ้นจาก Program ในส่วนนี้ไปเนื่องจากข้อมูลที่เข้ามาหลังจาก 1FH นั้นไม่ใช่ข้อมูลที่เราจะนำมาใช้ในการ Process แล้วซึ่งจะเขียน Flowchart ได้ดังรูป



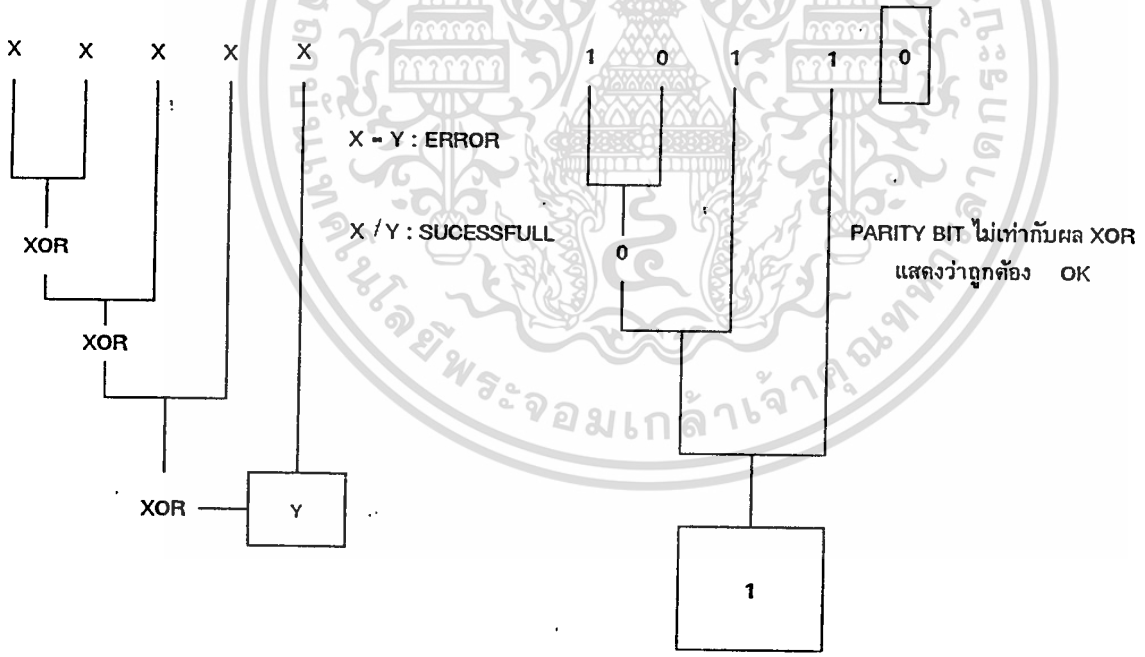
รูปที่ 5.4 Flowchart ของการค้นหาบิทเริ่มต้น

โปรแกรมย่อยของการค้นหาบิทเริ่มต้น

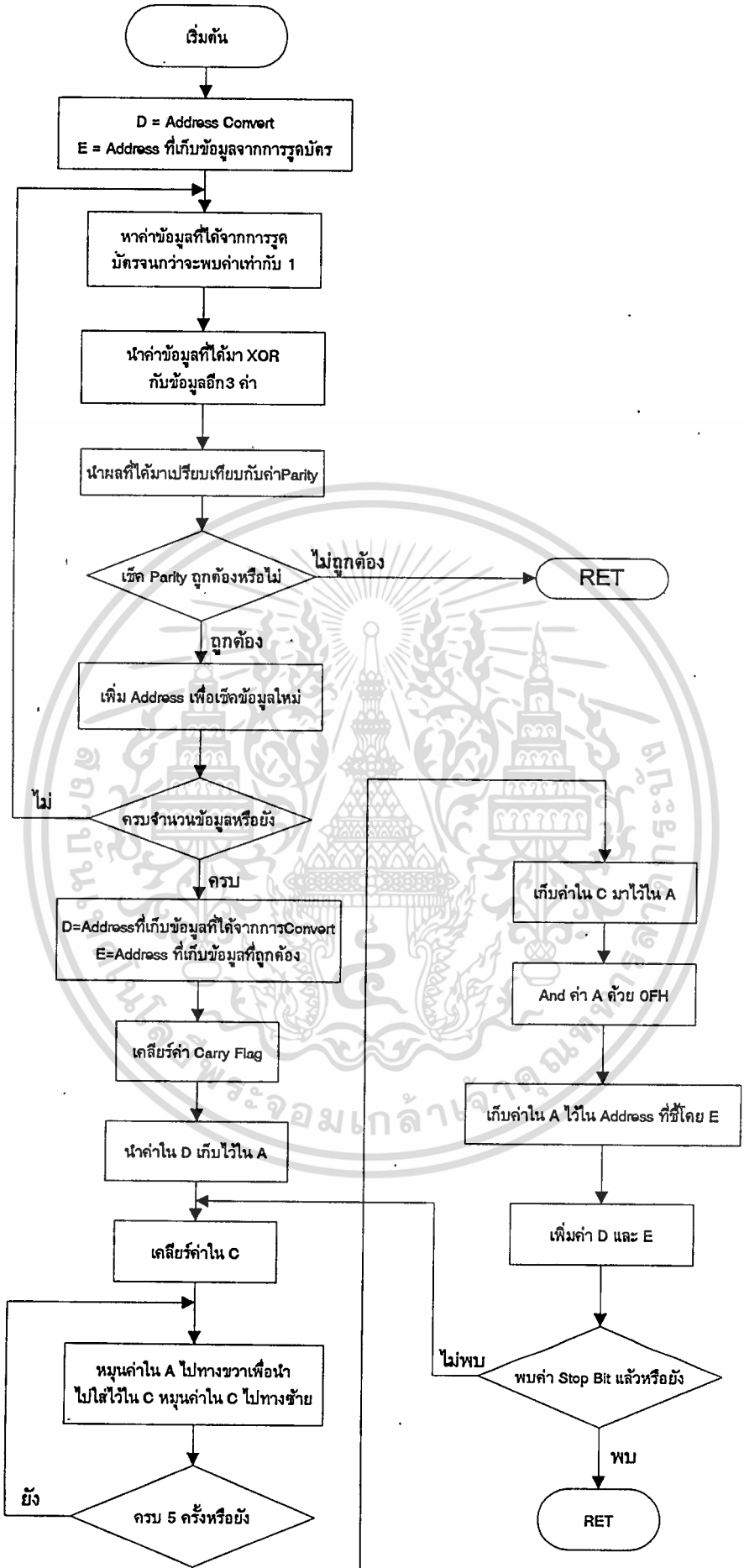
การหาบิทเริ่มต้นโดยการนำเอาตำแหน่งของข้อมูลที่รับข้อมูลเข้ามาเก็บไว้เป็นตัวอ้างอิงแล้วนำข้อมูลที่อยู่ใน ตำแหน่งข้อมูลนั้นออกมาแล้วทำการหมุนข้อมูลเพื่อทำการหาข้อมูลที่เป็นตัวเริ่มต้น

ซึ่งที่เราต้องการหาก็คือ Start Character จะทำได้โดยการย้ายข้อมูลที่นำไปเก็บใหม่ที่ละ Address มา Rotate Right นำข้อมูลที่ละบิทเข้ามาต่อกันใน Register แล้ว AND ด้วยค่า 1FH เพื่อให้เหลือข้อมูลเพียง 5 Bit แล้วทำการเปรียบเทียบว่าคือ 1AH หรือไม่ (ความจริงต้องเป็น 0BH แต่ข้อมูลยังกลับกันอยู่จึงทำให้ต้องการตรวจสอบ หา Start Character ด้วยค่า 1AH ; 11010 <—> 01011) ถ้ายังไม่พบก็จะทำการ Rotate อีก แล้วตรวจสอบใหม่อีกถ้ายังไม่พบก็จะทำเช่นนี้ซ้ำไปเรื่อย ๆ จนกว่าจะพบข้อมูลที่เราต้องการคือ 1AH แสดงว่าถ้าทำซ้ำ แบบเดิมต่อไปอีกทุก ๆ 5 Bit หมายถึง 1 Character ดังนั้นจึงทำเหมือนเดิมแต่นำค่าที่ได้ทุก ๆ 5 Bit ไปเก็บไว้ยังหน่วยความจำ และจะทำการตรวจดูข้อมูลก่อนนำไปเก็บยังหน่วยความจำว่ามีค่าเป็น 1FH หรือยังถ้าใช่แล้วก็จะพ้นจาก Program ในส่วนนี้ไปเนื่องจากข้อมูลที่เข้ามาหลังจาก 1FH นั้นไม่ใช่ข้อมูลที่เราจะนำมาใช้ในการ Process แล้ว

3. เป็นการเรียงข้อมูลและตรวจสอบว่ามีการผิดพลาดของข้อมูลจากการรูดบัตรหรือไม่โดยในการตรวจสอบข้อมูลว่ามีความถูกต้องหรือไม่โดยในส่วนสุดท้ายของการเตรียมข้อมูลนำหรับการ Process คือ ข้อมูลที่ถูกแบ่งออกเป็น 5 Bits นั้นในแต่ละ Address ยังมีค่ากลับกันอยู่เราจึงต้องทำการ Rotate และตรวจสอบ Parity หา Error ที่ได้จากการรูดบัตรบันทึกแม่เหล็กโดยการนำเอาข้อมูลที่ถูกตัดครั้งละ 5 Bits ต่อ 1 Address ณ. ตำแหน่งนั้นข้อมูลจะถูก Load ขึ้นมาแล้วจะทำการ Rotate Parity Bit ซึ่งอยู่ Bit ต่ำสุด ออกทาง Right ไปเก็บไว้ยัง Register อื่นซึ่งเป็นตัวตั้งสำหรับการเปรียบเทียบเพื่อหา Error ของ Data โดยผ่านทาง Carry Flag แล้วทำการ Rotate Right Bit ต่อไปผ่านทาง Carry Flag ไปยังอีก Register หนึ่งแล้วทำการ XOR ระหว่าง Register ที่เก็บข้อมูลนั้นกับข้อมูลBit ถัดมาโดยจะเก็บผลที่ได้จากการ XOR Data ทั้ง 4 Bit มาเปรียบเทียบกับค่าของ Parity Bit ที่อ่านเข้ามาได้ ถ้าถูกต้องก็ให้เก็บข้อมูลทุกตัวที่ Rotate ถูกต้องแล้วเอาไว้แต่ถ้ามีความผิดพลาดในการตรวจสอบพบจากการเปรียบเทียบกับ Parity ที่อ่านได้ก็จะ Error และจะไปรอการรูดบัตรเข้าใหม่ซึ่งจะมีลักษณะดังรูป



รูปที่ 5.5 แสดงการเปรียบเทียบข้อมูลที่ละ Bit (Parity Bit)

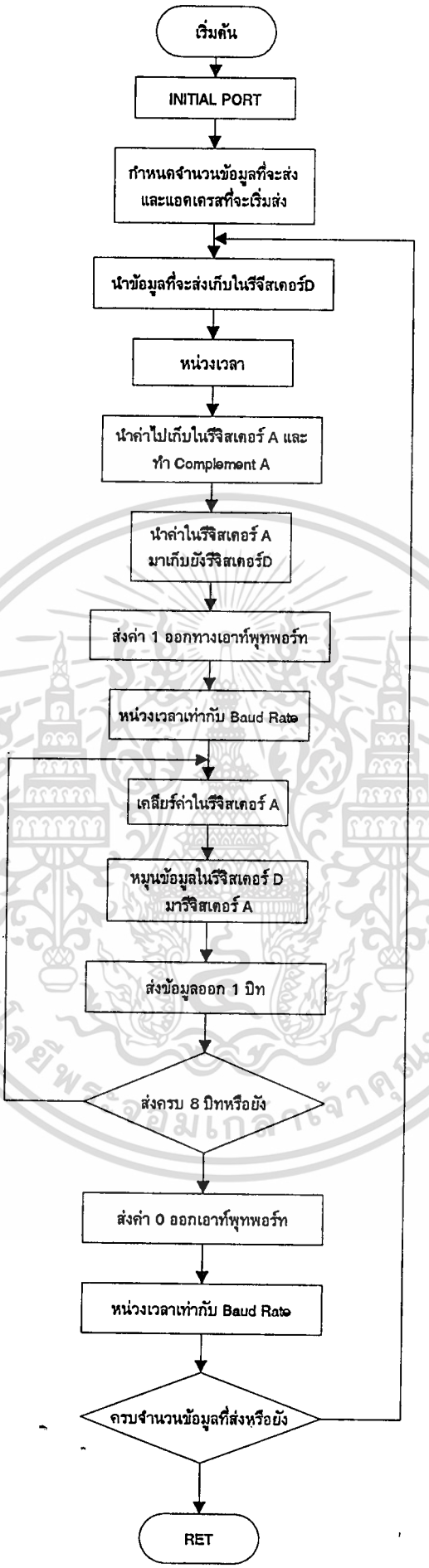


รูปที่ 5.6 Flowchartของการเรียงข้อมูลและตรวจสอบความผิดพลาดของข้อมูลที่รับเข้ามา

โปรแกรมของการเรียงข้อมูลและการตรวจสอบการผิดพลาด

โปรแกรมย่อยของการตรวจสอบความผิดพลาด จะอาศัยการตรวจสอบ พาริตี้ของข้อมูลที่รับเข้ามาโดยกำหนดแอดเดรส ตำแหน่งที่เก็บข้อมูลเป็นบิทจากการรูดบัตรและแอดเดรสตำแหน่งที่จะเก็บข้อมูลที่เรียงข้อมูลซึ่งข้อมูลที่อยู่บนบัตรแม่เหล็กจะมีข้อมูลเป็น 4 Bit และอีก 1 Bit เป็น Parity โดยข้อมูลจะมีประมาณ 60 Bit หลักการตรวจสอบก็คือ จะนำเอาข้อมูลทั้ง 4 Bit นั้นมา XOR กันและนำผลที่ได้ไปเปรียบเทียบกับพาริตี้บิท

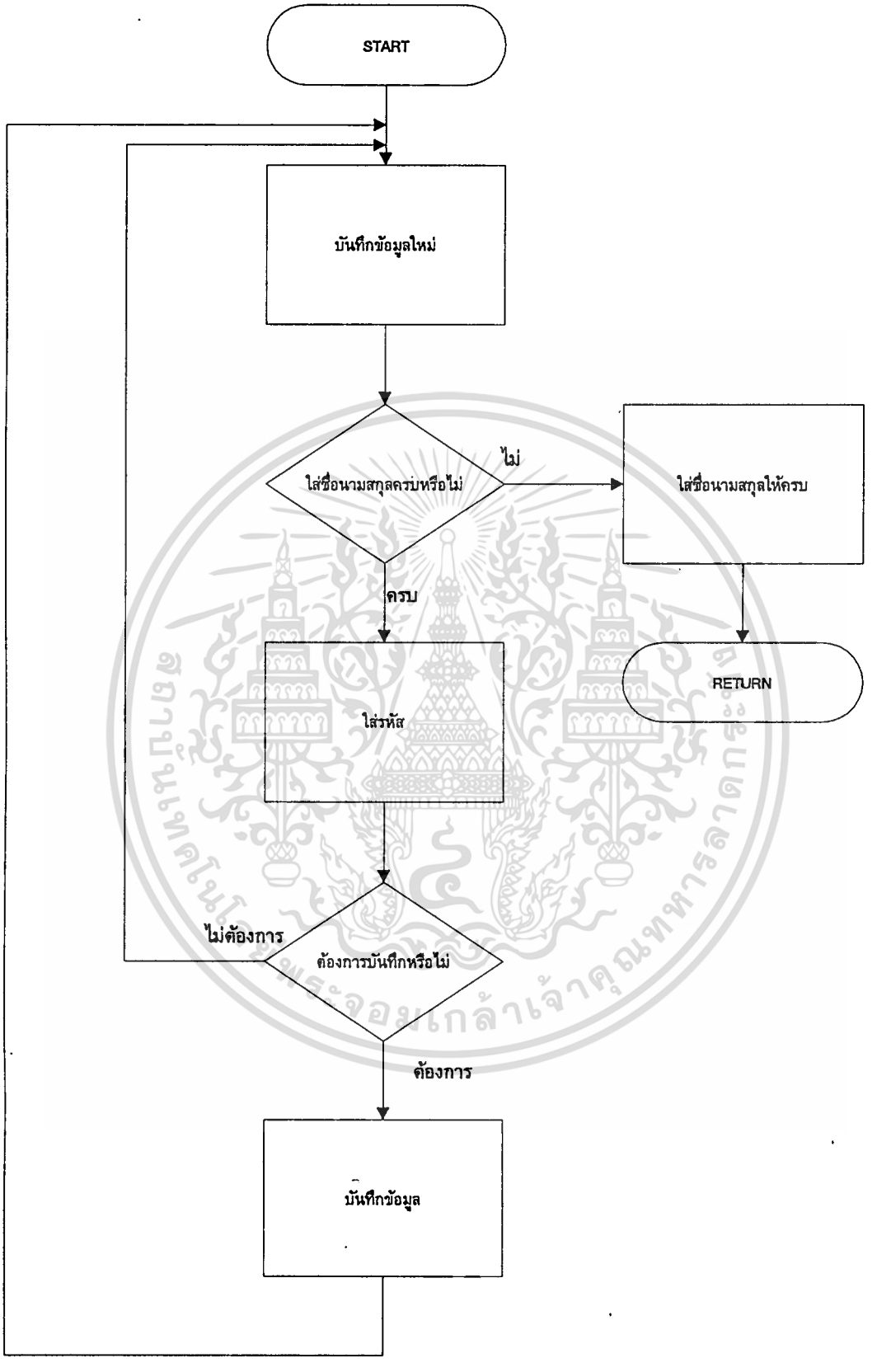
ตัวอย่าง สมมติข้อมูลในบัตร มี 1001 และพาริตี้เป็นคู่ จะเท่ากับ 1 เมื่อนำข้อมูล 1 XOR 0 XOR 0 XOR 1 จะเท่ากับ 0 ก็จะเปรียบเทียบโดยการที่ถ้าผลของการ XOR กันมีค่าไม่ตรงกับค่าพาริตี้ก็จะถือว่าถูกต้องและจะเพิ่ม Address ต่อไปเพื่อตรวจสอบจนครบ ถ้ามีการผิดพลาดก็จะไม่มีข้อมูลส่งออกไปยัง PC



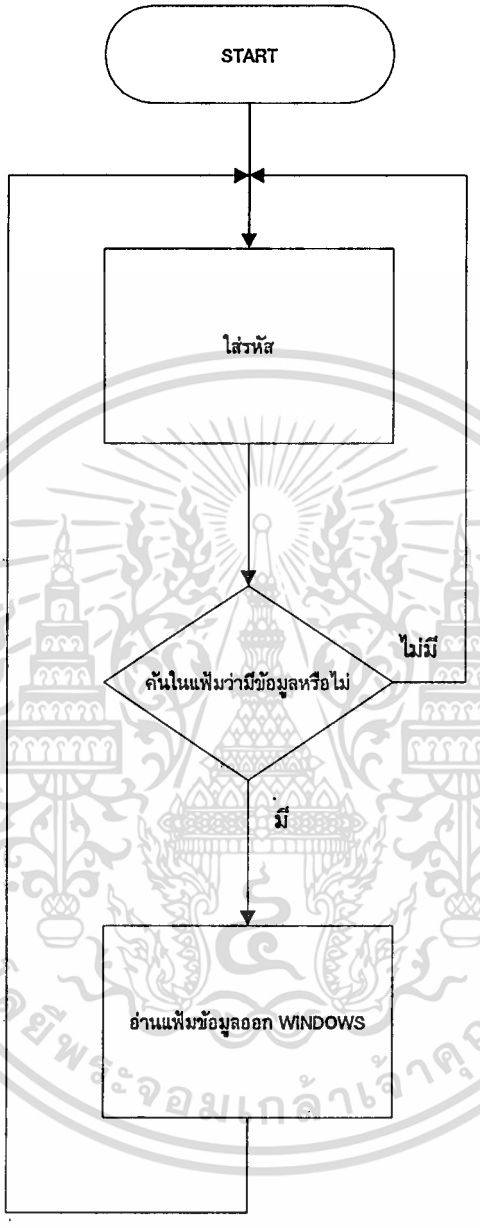
รูปที่5.7 Flowchart ของโปรแกรมการส่งข้อมูลแบบอนุกรม

โปรแกรมการส่งข้อมูลแบบอนุกรม

เมื่อจะส่งข้อมูลออกจากพอร์ทไปยัง PC จะต้องมีโปรแกรมที่ทำหน้าที่ส่งข้อมูลนี้ออกไปเริ่มด้วยการกำหนดพอร์ทเอาท์พุท กำหนดจำนวนข้อมูลที่จะส่งและกำหนดตำแหน่งหน่วยความจำการส่งข้อมูลนี้ต้องมีมาตรฐานของการส่งและต้องสัมพันธ์กับการรับของระบบด้วยซึ่งตามมาตรฐานการส่งข้อมูลของ Keyboard จะเป็นการส่งข้อมูลแบบ 8 Bit, 1 Start Bit, 1Stop Bit, และมีการตรวจสอบ Parity เป็น Odd Parity เป็น 0 หรือ 1 และอัตราความเร็วในการส่งข้อมูล Boad Rate เป็น 9600 BPS หลักการของโปรแกรมก็คือ นำค่าที่จะส่งมาทำ Complement เหตุที่ต้องทำก็เพื่อให้สัมพันธ์กับ ฮาร์ดแวร์ซึ่งจะเริ่มส่ง Start Bit ออกไปก่อนและหมุนข้อมูลที่จะส่งออกไปทีละบิทจนครบ 8 บิท และส่ง Stop Bit ออกมาการส่งข้อมูลแต่ละบิทจะต้องมีความสัมพันธ์กับค่า Boad Rate ด้วยโดยการใช้คำสั่งแต่ละคำสั่งใช้เวลาเท่าใด จะต้องนำมาใช้ในการคำนวณค่า Boad Rate ด้วย



Flowchart การเก็บข้อมูลของผู้ถือบัตร



Flowchart เปิดข้อมูลของผู้ถือบัตร

โปรแกรมฐานข้อมูล

โปรแกรมฐานข้อมูลซึ่งใช้ RUN บน Windows จาก Flowchart การเก็บข้อมูลของผู้ถือบัตรจะเริ่มที่บันทึกข้อมูลใหม่ เขียนชื่อและนามสกุลครบหรือไม่ ถ้าไม่ก็ใส่ชื่อและนามสกุลให้ครบ แล้วใส่รหัสและต้องการที่จะบันทึกหรือไม่ถ้าไม่ต้องการก็วนไปเริ่มใส่ข้อมูลใหม่ ถ้าต้องการก็บันทึกข้อมูลของผู้ถือบัตร

ส่วน Flowchart ของการเปิดข้อมูลของผู้ถือบัตรจะเริ่มที่ใส่รหัสแล้วไปค้นในแฟ้มว่ามีข้อมูลหรือไม่ถ้าไม่มีก็วนกลับให้ไปเริ่มต้น ถ้ามีก็จะไปค้นในแฟ้มข้อมูลแล้วโชว์ออกทางหน้าต่าง

สรุปการวิจัยและข้อเสนอแนะ

โครงงาน เครื่องอ่านบัตรแม่เหล็กนี้สามารถที่จะทำการอ่านบัตรแถบแม่เหล็กตามมาตรฐาน ISO ซึ่งสามารถอ่านข้อมูลในแถบแม่เหล็กของบัตร ATM ได้ซึ่งผู้ถือบัตรสามารถที่จะใช้บัตร ATM ร่วมกับบัตรอื่นก็ได้เช่นบัตรนักศึกษา หรือ บัตรสมาชิกซึ่งก็แล้วแต่ลักษณะงานที่ต้องการจะนำเอาไปใช้ ในโครงงานนี้จะใช้ RUN PROGRAM บน Windows โดยใช้ Visual Basic for Windows เป็นฐานข้อมูลในการเก็บข้อมูลของผู้ถือบัตรการใช้เครื่องอ่านบัตรแถบแม่เหล็กนี้จะไม่ยุ่งยากมากนักซึ่งจะมีรายละเอียดการใช้งานในภาคผนวก

การพัฒนาโครงงานนี้สามารถที่จะเพิ่มการใช้งานให้มากขึ้นโดยการใช้ร่วมกับบาร์โค้ดโดยเพิ่ม Hardware ขึ้นมาในส่วนของ Port ที่เหลือ

โดยการเปลี่ยนส่วนรับข้อมูลจากเครื่องอ่านโดยการเปลี่ยนแปลงฐานข้อมูลที่ RUN บน PC

เอกสารอ้างอิง

1. พันธ์ศักดิ์ ศรีทรัพย์,"เครื่องบันทึกเทปเล่ม 1",จัดพิมพ์โดย บริษัท อีเลคทรอนิคส์เวิลด์ จำกัด.

2. อ.นรินทร์ เนาวประทีป,"การใช้งาน Z80",จัดพิมพ์โดย หจก.สำนักพิมพ์ฟิสิกส์เซ็นเตอร์.

3. บริษัท ซีเอ็ดยูเคชั่น จำกัด,"คู่มือไอซีไมโครโปรเซสเซอร์และไอซีที่เกี่ยวข้อง"

4. CHARLES E.SPORCK,"LINEAR APPLICATIONS HANDBOOK",NATIONAL SEMICONDUCTOR COPORATION.

5. MOTOROLA INC.,1993"MOTOROLA LINEAR/INTERFACE ICS DEVICE DATA VOL.1"

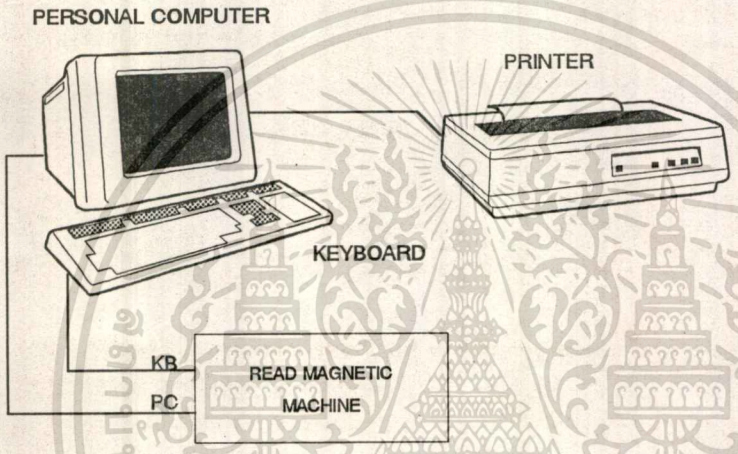
6. การเขียนโปรแกรมบนวินโดวส์ด้วย Microsoft Visual Basic โดย JOHN CLARK CRAIG

7. PC/AT Technical Reference

ภาคผนวก

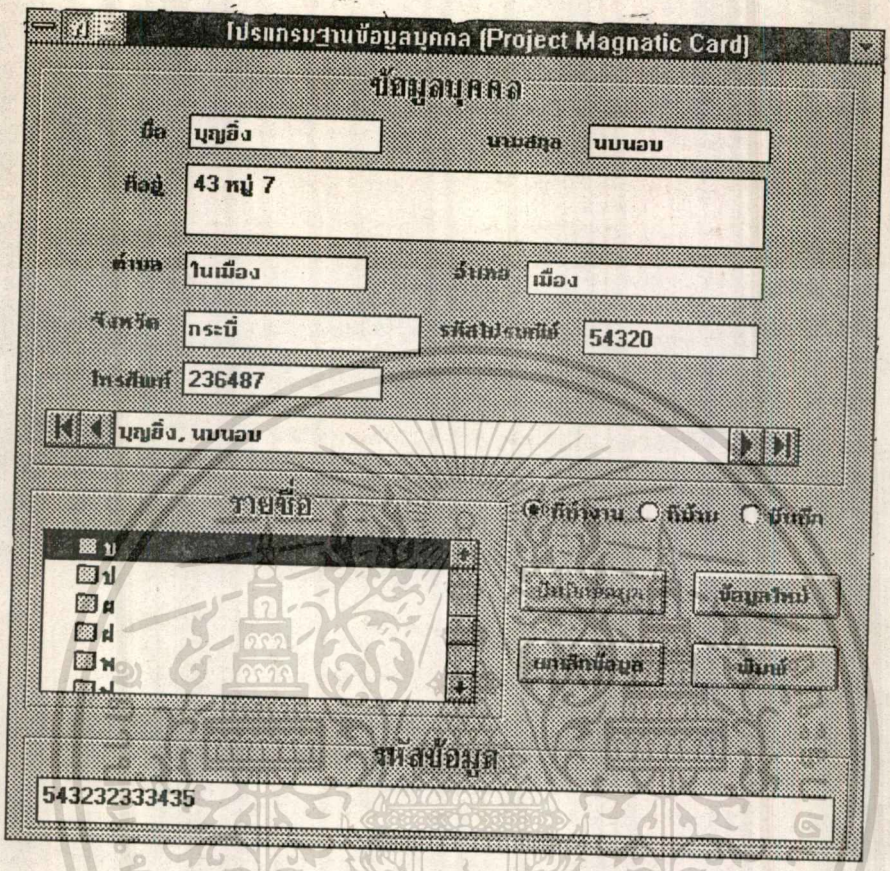
ภาคผนวก ก
การต่อใช้งาน,การติดตั้งฐานข้อมูล และการใช้งานฐานข้อมูล

การต่อเครื่องอ่านกับ Computer และ Printer



การต่อใช้งานเครื่องอ่านบัตรแม่เหล็กจะเป็นดังรูปข้างบน โดยเอา Keyboard มาต่อเข้ากับเครื่องอ่านบัตรแม่เหล็กที่ Connecter ที่เขียนว่า KB และ Connecter ที่เขียนว่า PC ก็ต่อไปเข้า PC ซึ่งสามารถใช้ร่วมกับ Printer ด้วยโดยการใช้ร่วมกับ Printer นั้น ก็โดยใช้Printer Manager ในโปรแกรมของ Windows สามารถพิมพ์ข้อมูลของผู้ถือบัตรออกทาง Printer ซึ่งจะมี Menu พิมพ์ ในโปรแกรมฐานข้อมูลที่เก็บข้อมูลของผู้ถือบัตร

การติดตั้งฐานข้อมูลลงบน Windows และการใช้งาน

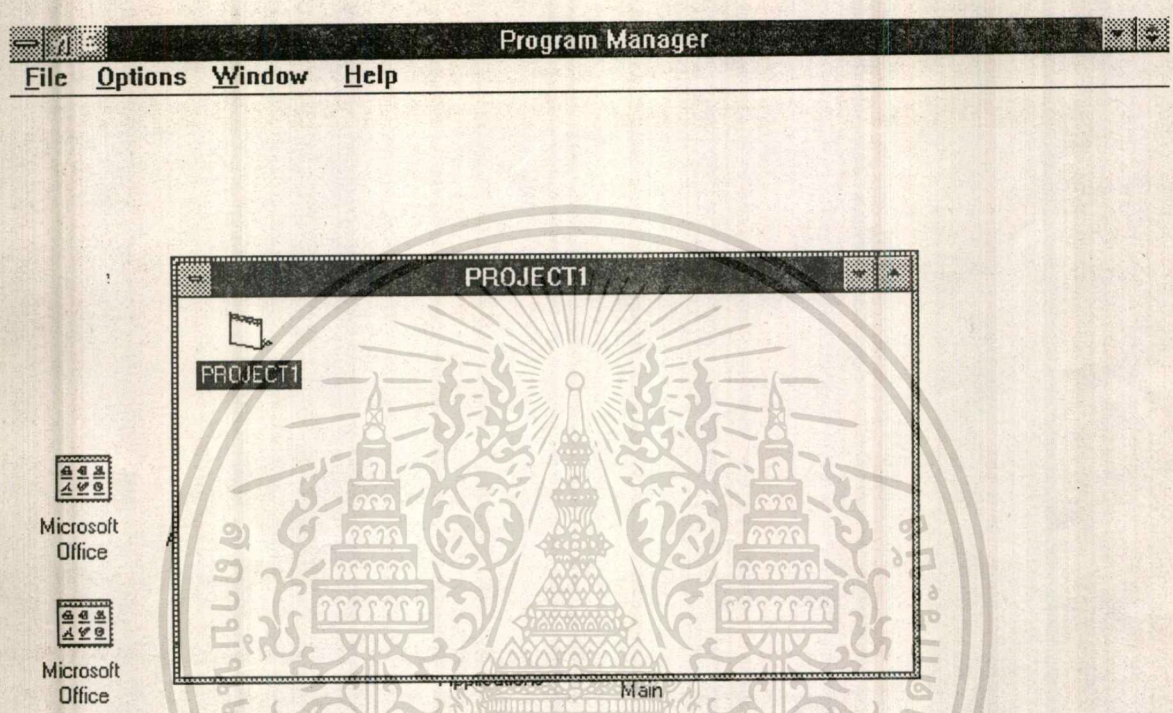


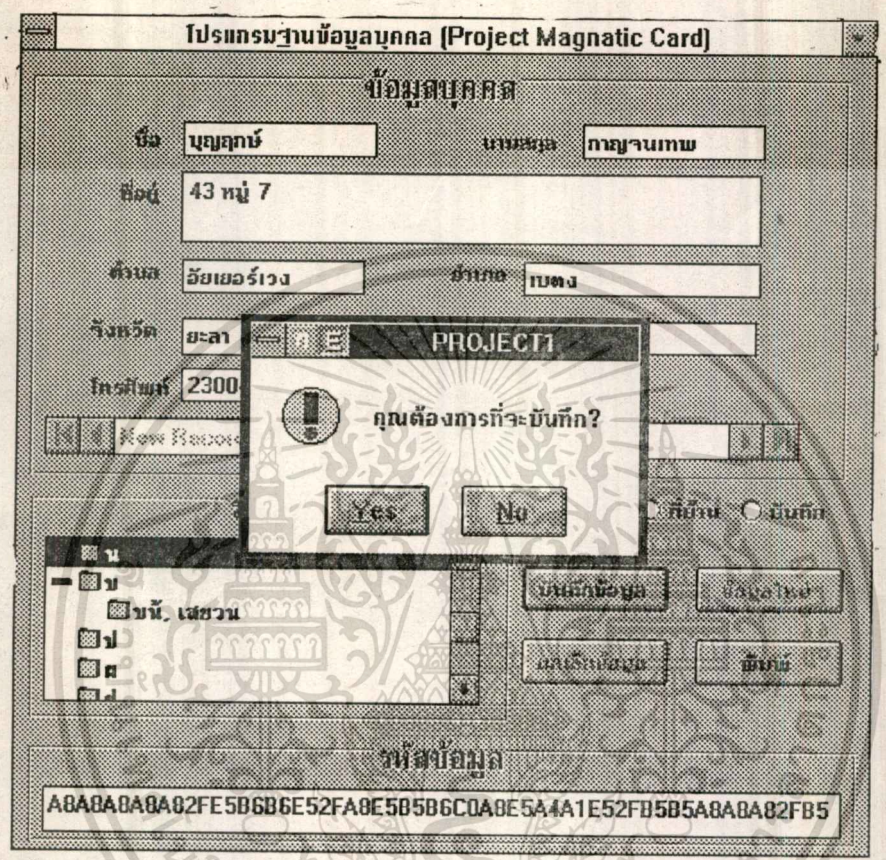
การ RUN โปรแกรมฐานข้อมูลบน Windows เมื่อRUN แล้วจะได้หน้าต่างออกมาดังรูป เมื่อต้องการที่จะเก็บข้อมูลของผู้ถือบัตร ก็ใช้เมาส์คลิกที่ - ข้อมูลใหม่- จะทำให้ Column ทุก Column ว่างใช้คีย์ Tab หรือใช้เมาส์คลิกในช่อง ชื่อ กับ นามสกุล ซึ่งจำเป็นต้องใส่แล้วมาคลิกที่ช่อง รหัสข้อมูล เสร็จแล้วทำการรูดบัตรผ่านเครื่อง รูดบัตรเสร็จแล้วจะมีหน้าต่างถามขึ้นว่า ต้องที่จะบันทึกหรือไม่ถ้าต้องการที่จะบันทึกก็คลิกในช่องที่ ต้องการฐานข้อมูลก็จะทำการบันทึกข้อมูลที่พิมพ์ลงไป ซึ่งข้อมูลที่บันทึกนี้สามารถที่จะพิมพ์ออกทาง Printer ได้

การติดตั้งโปรแกรมลงบน Windows

ในขั้นแรกใส่แผ่น Install ใน Drive A หรือ B เข้าไปยัง Drive C เข้าไปใน Windows ใช้เมาส์คลิก Main 2 ครั้ง แล้วคลิกที่ File Manager 2 ครั้ง แล้วคลิกไปที่ Drive ที่ใส่แผ่นดิสค์อยู่แล้วคลิกที่ File ชื่อว่า Setup.exe 2 ครั้ง แล้วทำตามลำดับขั้นตอนจนถึงติดตั้งเรียบร้อยก็จะมี Icon ชื่อว่า Project1 ขึ้นบน Windows เมื่อทำการติด

ตั้งเสร็จเรียบร้อยแล้วก็ทำการ Copy File ที่ชื่อว่า phone.mdb ไปยัง Directory Project1 เป็นการติดตั้งฐานข้อมูลที่จะเก็บข้อมูลของผู้ถือบัตรเรียบร้อยซึ่งการใช้งานได้อธิบายไว้แล้วในย่อหน้าก่อน





เมื่อใส่ชื่อนามสกุลและรหัสเรียบร้อยแล้วใช้เมาส์คลิกที่บันทึกข้อมูลที่หน้าต่างจะโชว์ -คุณต้องการที่จะบันทึกหรือไม่- ถ้าต้องการที่จะบันทึกข้อมูลก็ใช้เมาส์คลิกที่ Yes ถ้าไม่ต้องการที่จะบันทึกข้อมูลก็ใช้เมาส์คลิกที่ No

และถ้าต้องการที่พิมพ์ออกเครื่องพิมพ์ก็ใช้เมาส์คลิกที่ พิมพ์ ซึ่งข้อมูลที่บันทึกลงไปก็จะพิมพ์ออกมาหมดแต่มีข้อยกเว้นคือ รหัสของบัตรจะไม่พิมพ์ออกมาให้ซึ่งเมื่อพิมพ์ออกมาแล้วจะได้ดังรูปร่างดังในหน้าต่อไป

ประวัติส่วนบุคคล

ชื่อ *บุญฤกษ์* **นามสกุล** *กาญจนเทพ*

---------- **ที่อยู่ที่บ้าน** ----------

ที่อยู่ *43 หมู่ 7*

ตำบล *อัยเยอร์เวง* **อำเภอ** *เบตง*

จังหวัด *ยะลา* **ระหัสไปรษณีย์** *95110*

เบอร์โทรศัพท์ *230043*

---------- **ที่อยู่ที่ทำงาน** ----------

ที่อยู่

ตำบล **อำเภอ**

จังหวัด **ระหัสไปรษณีย์**

เบอร์โทรศัพท์

< ประวัติส่วนตัว >

****

ภาคผนวก ข
โปรแกรม

```
BUFF:           EQU     00F36H
ERR:            EQU     08081H
STR:            EQU     08082H
STR1:           EQU     08083H
STR2:           EQU     08084H
STR3:           EQU     08085H
CHA:            EQU     08087H
STR4:           EQU     0808FH
STR5:           EQU     08091H
STR6:           EQU     08095H
GET:            EQU     08097H
RED:            EQU     08099H
PAR:            EQU     0809CH
GET1:           EQU     0809DH
GET2:           EQU     0809EH
ERR1:           EQU     0809FH
ERR2:           EQU     080A0H
ERR3:           EQU     080A1H
STR7:           EQU     080A2H
PAR2:           EQU     080A3H
SEND:           EQU     080ACH
SEND1:          EQU     080ADH
CHEC:           EQU     080B7H
RIGHT:          EQU     080D6H
CLEAR:          EQU     084C0H
CLEA1:          EQU     084C1H
FAUL:           EQU     08743H
START:          EQU     087DCH
STAR:           EQU     087DDH
DATA:           EQU     087DEH
RIGHT1:         EQU     087E2H
RIGHT2:         EQU     087E3H
RIGHT3:         EQU     087E4H
RIGHT4:         EQU     087E5H
DAT:            EQU     087E9H
RIGHT5:         EQU     087EAH
DAT1:           EQU     087EBH
DAT2:           EQU     087ECH
;
        ORG     00000H
;
        DI
        MVI B,068H
;
DELAY:  DCR B
        JNZ DELAY
        LXI SP,0FFFFH
;
        MVI L,04BH
        MOV B,L
        DCR M
        MVI M,058H
        MVI L,021H
        DI
        ADD A
        SHLD  STR4
        LXI H,0F824H
        SHLD  STR5
        MVI A,08AH
        OUT 063H
        MVI A,0F8H
        OUT 060H
        MVI A,002H
```

```
        OUT 062H
        CALL  DELAY1
        LDA ERR
        CPI 082H
        JZ  RECV
        MVI A,082H
        STA ERR
        LHLD  STR4
        SHLD  STR3
        SHLD  CHA
        MVI A,001H
        STA STR2
        STA STR
        STA STR1
;
RECV:   IN  061H
        ANI 010H
        STA BUFF
        IN  062H
        ANI 080H
        STA BUFF
        MVI A,001H
        XRA A
        STA CHEC
        STA CLEAR
        STA SEND
        STA SEND1
        MVI A,07FH
        LXI H,08093H
        MOV M,A
        MVI C,030H
        LXI H,049E0H
        CPI 082H
        JZ  RECV1
        LXI H,04A0CH
        JMP INTE
;
;
RECV1:  LXI H,049F6H
;
INTE:   INX H
        MVI B,014H
;
REQ:    MVI A,01CH
        SIM
        EI
;
DATOU:  MVI A,0F0H
        OUT 060H
        IN  061H
        RLC
        MOV B,A
        IN  061H
        ANI 010H
        XRA B
        JNZ L3BAB
        LDA L8009
        MOV B,A
        IN  062H
        ANI 080H
        XRA B
        JNZ L3C23
        EI
        JMP DATOU
```

```
;
;
DELAY1: XRA A
        MOV C,A
        MOV D,A
        MVI E,008H
;
ER:     CALL  CHEC2
        LDA STAR
        CPI 0A0H
        JC  ER1
        INR D
;
ER1:    LDA START
        CPI 0A0H
        JC  ER2
        INR D
;
ER2:    DCR E
        JNZ ER
        MOV A,D
        CPI 010H
        JNZ SENDA
        CALL  DEL
        RET
;
SENDA:  CALL  CHEC
        RET
;
CHEC:   PUSH  B
        MVI C,000H
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        STA DAT
        LDA START
        STA RIGHT5
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        STA DAT1
        LDA START
        STA DAT2
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        STA RIGHT1
        LDA START
        STA RIGHT3
        CALL  CHEC2
        LDA STAR
        STA RIGHT2
        LDA START
```

```
        STA RIGHT4
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
        CALL  CHEC2
        LDA STAR
;
CHEC1:  MOV A,C
        CALL  SAMP3
        MOV A,C
        CALL  OUT4
        INR C
        RET
;
CHEC2:  MOV A,C
        CALL  CLR2
        INR C
        RET
;
OUTP:   MVI A,0F8H
        ORI 001H
        ORI 002H
        OUT 060H
        ANI 0FDH
        ORI 004H
        OUT 060H
        ORI 002H
        OUT 060H
        STA DATA
        RET
;
OUTP1:  PUSH  B
        MVI B,008H
        PUSH  PSW
;
ROTE:   POP PSW
        RLC
        PUSH  PSW
        JC  OUT2
        LDA DATA
        ANI 0FDH
        ANI 0FBH
        STA DATA
        OUT 060H
        JMP OUT3
;
OUT2:   LDA DATA
```

```
        ANI 0FDH
        ORI 004H
        STA DATA
        OUT 060H
;
OUT3:   ORI 002H
        STA DATA
        OUT 060H
        DCR B
        JNZ ROTE
        POP PSW
        POP B
        RET
;
        CALL  OUTP
        MVI A,010H
        CALL  OUTP1
        LDA STAR
        CALL  OUTP1
        LDA START
        CALL  OUTP1
        CALL  CONT
        RET
;
OUT4:   PUSH  B
        ANI 03FH
        MOV B,A
        MVI A,040H
        ORA B
        MOV B,A
        CALL  OUTP
        MOV A,B
        CALL  OUTP1
        LDA STAR
        CALL  OUTP1
        LDA START
        CALL  OUTP1
        CALL  CONT
        POP B
        RET
;
CLR1:   CALL  OUTP
        MVI A,030H
        CALL  OUTP1
        CALL  RECSYS
        RET
;
CLR1:   CALL  OUTP
        MVI A,000H
        CALL  OUTP1
        CALL  RECSYS
        RET
;
CLR2:   PUSH  B
        ANI 03FH
        MOV B,A
        MVI A,080H
        ORA B
        MOV B,A
        CALL  OUTP
        MOV A,B
        CALL  OUTP1
        CALL  RECSYS1
        CALL  OUT5
```

```
          STA START
          CALL  OUT5
          STA STAR
          CALL  RECSYS
          POP B
          RET
;
RECSYS:   LDA DATA
          ANI 0FEH
          STA DATA
          OUT 060H
          LDA DATA
          ORI 001H
          STA DATA
          OUT 060H
          MVI A,0F8H
          STA DATA
          OUT 060H
          RET
;
RECSYS1:LDA DATA
          ANI 0FDH
          ANI 0FBH
          STA DATA
          OUT 060H
          LDA DATA
          ORI 002H
          STA DATA
          OUT 060H
          RET
;
OUT5:     PUSH  B
          MVI B,008H
          XRA A
          PUSH  PSW
;
SAMP:     LDA DATA
          ANI 0FDH
          STA DATA
          OUT 060H
          IN  061H
          ANI 004H
          JZ  SAMP1
          POP PSW
          STC
          RAL
          PUSH  PSW
          JMP SAMP2
;
SAMP1:    POP PSW
          STC
          CMC
          RAL
          PUSH  PSW
;
SAMP2:    LDA DATA
          ORI 002H
          STA DATA
          OUT 060H
          DCR B
          JNZ SAMP
          POP PSW
          POP B
          RET
```

```
;
          CALL  OUTP
          MVI A,020H
          CALL  OUTP1
          CALL  CONT
          RET
;
SAMP3:    PUSH  B
          PUSH  PSW
          CALL  OUTP
          POP PSW
          ANI 03FH
          MOV B,A
          MVI A,0C0H
          ORA B
          CALL  OUTP1
          CALL  CONT
          POP B
          RET
;
CONT:     MVI A,0F8H
          OUT 060H
          STA DATA
          LDA DATA
          ORI 001H
          STA DATA
          OUT 060H
          IN  061H
          ANI 004H
          JNZ CONT1
          JMP CONT
;
CONT1:    MVI A,0F8H
          OUT 060H
          RET
;
COUNT:    CALL  STOP
          JNC COUNT1
          INX H
          INX H
          DCR C
          DCR C
          DCR B
          JNZ COUNT
;
COUNT1:   INX H
          INX H
          INX H
          INX H
          INX H
          INX H
;
STOP:     PUSH  B
          PUSH  D
          PUSH  H
          CALL  RORF
          CALL  SER
          LXI H,00A4AH
          DAD D
          MOV A,M
          CPI 00FH
          JZ  L07F5
          CPI 09AH
          CPI 09BH
```

```
        CPI 09CH
;
SER:    MVI B,000H
        MVI C,006H
        MOV A,D
        STC
        RAL
        MOV D,A
        INR B
        LDA ERR1
        CMP M
        MOV A,D
        STC
        RAL
        MOV D,A
        INR B
        LDA ERR2
        CMP M
;
STAT:   INX     H
        DCR     C
        MOV     A,B
        CPI     00AH
        JNZ     STAT1
        MOV     A,D
        ANI     001H
        MOV     D,A
        STC
        CMC
        RET
;
STAT1:  STC
        RET
;
RORF:   PUSH    D
        PUSH    H
        LXI     D,00000H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
        INX     H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
        INX     H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
        INX     H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
```

```
        INX     H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
        INX     H
        PUSH    H
        MOV     L,M
        MVI     H,000H
        DAD     D
        XCHG
        POP     H
        LXI     H,00000H
        DAD     D
        DAD     D
        DAD     D
        STA     GET1
        LHLD    GET
        STA     GET2
        MOV     E,A
        LDA     GET1
        ADD     E
        STA     ERR1
        LHLD    GET
        MOV     E,A
        LDA     GET1
        ADD     E
        STA     ERR2
        MOV     E,A
        LDA     GET2
        ADD     E
        STA     ERR3
        POP     H
        POP     D
        RET
;
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        RRC
        MOV     H,H
        RRC
        RRC
        MOV     H,E
        RRC
        RRC
        MOV     L,D
```

```
MOV     L,C
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
MOV     B,E
RRC
RRC
DCR     H
RRC
RRC
RRC
RRC
RRC
MOV     B,D
RRC
RRC
MOV     A,M
RRC
MOV     B,M
DFB     028H
RRC
MOV     B,L
RRC
RRC
MOV     C,H
RRC
RRC
RRC
MOV     C,A
RRC
MOV     L,A
RRC
RRC
RRC
RRC
RRC
MOV     H,D
RRC
RRC
INR     H
RRC
RRC
RRC
RRC
RRC
INX     H
RRC
RRC
MOV     M,D
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
MOV     H,C
RRC
```

```
RRC
RRC
RRC
RRC
MOV     M,C
RRC
RRC
MOV     L,B
DAA
RRC
MVI     H,00FH
RRC
MVI     L,067H
RRC
RRC
RRC
RRC
RRC
DCR     L
RRC
RRC
RRC
RRC
LXI     SP,00F0FH
SIM
RRC
RRC
MOV     M,L
MOV     M,H
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
RRC
MOV     B,C
RRC
RRC
MOV     A,L
RRC
RRC
RRC
RRC
RRC
RRC
MOV     H,B
RRC
MOV     E,A
RRC
RRC
RRC
RRC
RRC
```

```
MOV   M,B
RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
                RRC
        ;
                EI
                RET
        ;
                END
```

```
Dim CountSel%
Dim DBName As String
Dim gLIBDB As Database
Dim gDS As DynaSet
Dim gCode As String
Dim iCurrentRecord As Integer
Dim fAll As Integer
Dim CurrRec As Integer
Dim fStartUp As Integer
Dim iEditMode As Integer

Const EM_NOTHING = 0
Const EM_EDIT = 1
Const EM_ADDNEW = 2

Const YES = 6
Const MSGBOX_TYPE = 4 + 48

Dim Newdata As Integer
Dim ChexkData As Integer
Dim CheckNew As Integer
Dim textold$
Dim KeyHit$

Sub cDelete_Click ()
  If Outline1.Indent(Outline1.ListIndex) = 2 Then   'Expanded name.
    Outline1.RemoveItem Outline1.ListIndex
  Else
    Outline1.Expand(Outline1.ListIndex) = True
    For i = Outline1.ListIndex To Outline1.ListCount - 1
      If Outline1.List(i) = data1.Recordset!LastName + ", " + data1.Recordset!FirstName Then
        Outline1.RemoveItem i
```

```
        Exit For
      End If
    Next i
  End If

  data1.Recordset.Delete
  data1.Recordset.MoveNext
  If data1.Recordset.EOF Then data1.Recordset.MovePrevious
  Outline1.SetFocus

End Sub

Sub closedb ()
  On Error Resume Next
  gLIBDB.Close
End Sub

Sub cNew_Click ()
  On Error GoTo cNewErr
  textold = ""
  'text1.Text = ""
   CountSel = 0
  CheckNew = False
  CurrRec = data1.Recordset!ID
  data1.Recordset.AddNew
  data1.Caption = "New Record"
  data1.Enabled = False
  cNew.Enabled = False
  cDelete.Enabled = False
  cUpdate.Enabled = True
  Newdata = True
  tLName.SetFocus
```

```
    GoTo cNewEnd

cNewErr:
    If Err = 3021 Then Resume Next
    MsgBox Error$
    Resume cNewEnd

cNewEnd:


End Sub

Sub Command1_Click ()
    printer.Line (800, 0)-(7000, 700), &HC0E0FF, BF

    printer.FontName = "IrisUPC"
    printer.FontBold = True
    printer.FontUnderline = True
    printer.FontSize = 25
    printer.CurrentX = 2500
    printer.CurrentY = 10
    printer.Print "ประวัติส่วนบุคคล"
    printer.FontUnderline = False

    printer.FontSize = 15

    printer.CurrentY = 1000
    printer.CurrentX = 1000
    printer.Print "ชื่อ      "
    printer.CurrentY = 1000
    printer.CurrentX = 4000
```

```
printer.Print "นามสกุล  "
printer.CurrentY = 1600
printer.CurrentX = 2000
printer.Print " --------- ที่อยู่ที่บ้าน ---------"
printer.CurrentY = 2000
printer.CurrentX = 1000
printer.Print "ที่อยู่"
printer.CurrentY = 2800
printer.CurrentX = 1000
printer.Print "ตำบล"
printer.CurrentY = 2800
printer.CurrentX = 4000
printer.Print "อำเภอ"
printer.CurrentY = 3200
printer.CurrentX = 1000
printer.Print "จังหวัด"
printer.CurrentY = 3200
printer.CurrentX = 4000
printer.Print "ระหัสไปรษณีย์"
printer.CurrentY = 3600
printer.CurrentX = 1000
printer.Print "เบอร์โทรศัพท์"
printer.CurrentY = 4000
printer.CurrentX = 2000
printer.Print " --------- ที่อยู่ที่ทำงาน ---------"
printer.CurrentY = 4400
printer.CurrentX = 1000
printer.Print "ที่อยู่"
printer.CurrentY = 5200
printer.CurrentX = 1000
printer.Print "ตำบล"
printer.CurrentY = 5200
```

```
printer.CurrentX = 4000
printer.Print "อำเภอ"
printer.CurrentY = 5600
printer.CurrentX = 1000
printer.Print "จังหวัด"
printer.CurrentY = 5600
printer.CurrentX = 4000
printer.Print "ระหัสไปรษณีย์"
printer.CurrentY = 6000
printer.CurrentX = 1000
printer.Print "เบอร์โทรศัพท์"
printer.CurrentY = 6800
printer.CurrentX = 1000
printer.Print "< ประวัติส่วนตัว >"

printer.FontBold = False
printer.FontItalic = True
printer.CurrentY = 1000
printer.CurrentX = 1500
printer.Print tLName.Text
printer.CurrentY = 1000
printer.CurrentX = 5000
printer.Print tFName.Text

printer.CurrentY = 2000
printer.CurrentX = 2000
printer.Print taddress(0).Text
printer.CurrentY = 2800
printer.CurrentX = 2000
printer.Print tcity(0).Text
printer.CurrentY = 2800
printer.CurrentX = 5000
```

```
printer.Print tregion(0).Text
printer.CurrentY = 3200
printer.CurrentX = 2000
printer.Print tcountry(0).Text
printer.CurrentY = 3200
printer.CurrentX = 5500
printer.Print tpost(0).Text
printer.CurrentY = 3600
printer.CurrentX = 2500
printer.Print tphone(0).Text
printer.CurrentY = 4400
printer.CurrentX = 2000
printer.Print taddress(1).Text
printer.CurrentY = 5200
printer.CurrentX = 2000
printer.Print tcity(1).Text
printer.CurrentY = 5200
printer.CurrentX = 5000
printer.Print tregion(1).Text
printer.CurrentY = 5600
printer.CurrentX = 2000
printer.Print tcountry(1).Text
printer.CurrentY = 5600
printer.CurrentX = 5500
printer.Print tpost(1).Text
printer.CurrentY = 6000
printer.CurrentX = 2500
printer.Print tphone(1).Text
printer.CurrentY = 6800
printer.CurrentX = 3000
printer.Print tNotes.Text
printer.CurrentX = 1000
```

```
        ' tFName.Text
        printer.Print
"*************************************************************************************"
        printer.EndDoc


    ' printer.Print "ประวัติส่วนบุคคล"

End Sub

Sub cUpdate_Click ()
  CountSel = 0
  CheckNew = True
  Newdata = False
  If tLName <> "" And tFName <> "" And text1 <> "" Then
    If data1.EditMode = EM_ADDNEW Then
      'text1.Text = textold
      data1.Recordset.Update
      If data1.EditMode = 0 Then
        data1.Recordset.MoveLast
        CurrRec = data1.Recordset!ID
        data1.Refresh
        FillList
        ProcessOutline
        data1.Recordset.FindFirst "ID = " + CStr(CurrRec)
      Else
        data1.Recordset.FindFirst "ID = " + CStr(CurrRec)
      End If
    Else
      data1.Recordset.Update
      If data1.EditMode = 0 Then
        CurrRec = data1.Recordset!ID
```

```
        data1.Refresh
        FillList
        ProcessOutline
        data1.Recordset.FindFirst "ID = " + CStr(CurrRec)
      Else
        data1.UpdateControls
      End If
    End If
    data1.Enabled = True
    cNew.Enabled = True
    cDelete.Enabled = True
    cUpdate.Enabled = False
    Outline1.SetFocus

  Else
    MsgBox "ต้องใส่ชื่อนามสกุลและรหัสข้อมูลให้ครบ"
  End If
End Sub

Sub Data1_RePosition ()
  gCode = tNotes.Text
' textold = ""
  If Not data1.Recordset.EOF Then
    If Not IsNull(data1.Recordset!LastName) And Not IsNull(data1.Recordset!FirstName) Then
      data1.Caption = data1.Recordset!LastName + ", " + data1.Recordset!FirstName
    Else
      data1.Caption = ""
    End If
    'text1.Text = textold
    For i% = 0 To Outline1.ListCount - 1
If Outline1.List(i%) = data1.Recordset!LastName + ", " + data1.Recordset!FirstName Then
      Outline1.ListIndex = i%
```

```
    End If

  Case 7

  Case 8
    Save = False

  Case 9

  Case 10
    If Save = True Then
      If MsgBox("เก็บข้อมูลก่อนเลิก?", MSGBOX_TYPE) <> YES Then Save = False
    End If

 End Select
End Sub

Sub FillList ()
On Error GoTo FillPhoneErr
Dim test%
 Set gDS = data1.Recordset.Clone()

 Outline1.Clear
 'Fill top level ก-ฮ

 For i = 0 To 45
    Outline1.AddItem Chr$(161 + i)
    Outline1.Indent(Outline1.ListCount - 1) = 1
    gDS.MoveFirst
 Do While Not gDS.EOF
          test% = True
```

```
        If Not Outline1.IsItemVisible(Outline1.ListIndex) Then
          stChar = Left(Outline1.FullPath(Outline1.ListIndex), 1)
          Do While stChar <> Outline1.List(Outline1.ListIndex)
            Outline1.ListIndex = Outline1.ListIndex - 1
          Loop
        End If
        Exit For
      End If
    Next i%
  Else
    data1.Caption = "ไม่พบข้อมูล !"
  End If
 ' textold = text1.Text
End Sub

Sub Data1_Validate (Action As Integer, Save As Integer)

  Select Case Action
    Case 1

    Case 2

    Case 3

    Case 4

    Case 5

    Case 6
      If Save = True Then
        If MsgBox("คุณต้องการที่จะบันทึก?", MSGBOX_TYPE) <> YES Then Action = 0:
Save = False
```

```
            If (Asc(Left(gDS!LastName, 1)) >= Asc(Chr$(224))) And (Asc(Left(gDS!
LastName, 1)) <= Asc(Chr$(228))) Then
                test% = False
            End If
            If test% = True Then
                If (Left(gDS!LastName, 1)) = (Chr$(161 + i)) Then
                  Outline1.AddItem gDS!LastName + ", " + gDS!FirstName
                  Outline1.Indent(Outline1.ListCount - 1) = 2
                End If
            End If
            If test% = False Then
                If (Mid(gDS!LastName, 2, 1)) = (Chr$(161 + i)) Then
                  Outline1.AddItem gDS!LastName + ", " + gDS!FirstName
                  Outline1.Indent(Outline1.ListCount - 1) = 2
                End If
            End If
            gDS.MoveNext
        Loop
    Next i
EndOfData:
  Exit Sub
FillPhoneErr:

  MsgBox Error(Err)
  Resume Next
  Exit Sub

End Sub

Sub Form_Load ()
  DBName = App.Path + "\phone.mdb"
  fStartUp = True
```

```
    data1.DatabaseName = DBName
    X% = OpenDB(DBName)

    Load tphone(1)
    Load taddress(1)
    Load tcity(1)
    Load tregion(1)
    Load tcountry(1)
    Load tpost(1)

  ' cUpdate.Enabled = False
    RefreshForm

End Sub

Sub Label1_Click ()
  ' Picture2.Picture = pTab1.Picture
  ' Picture1.Visible = True

    tphone(0).Visible = True
    taddress(0).Visible = True
    tcity(0).Visible = True
    tregion(0).Visible = True
    tcountry(0).Visible = True
    tpost(0).Visible = True

    tphone(1).Visible = False
    taddress(1).Visible = False
    tcity(1).Visible = False
    tregion(1).Visible = False
    tcountry(1).Visible = False
    tpost(1).Visible = False
```

```
    tNotes.Visible = False
End Sub

Sub Label2_Click ()
    'Picture2.Picture = pTab2.Picture
    'Picture1.Visible = True

    tphone(1).Visible = True
    taddress(1).Visible = True
    tcity(1).Visible = True
    tregion(1).Visible = True
    tcountry(1).Visible = True
    tpost(1).Visible = True

    tphone(0).Visible = False
    taddress(0).Visible = False
    tcity(0).Visible = False
    tregion(0).Visible = False
    tcountry(0).Visible = False
    tpost(0).Visible = False

    tNotes.Visible = False
End Sub

Sub Label3_Click ()
  ' Picture2.Picture = pTab3.Picture

    Picture1.Visible = False
    tNotes.Visible = True

End Sub
```

```
Function OpenDB (DBName As String) As Integer
  Dim Connect As String

  On Error GoTo OpenDBErr

  Set gLIBDB = OpenDatabase(DBName)

  'success
  OpenDB = True
  GoTo OpenDBEnd

OpenDBErr:
  OpenDB = False
  Resume OpenDBEnd

OpenDBEnd:

End Function

Sub Option3D1_Click (Value As Integer)
 ' Picture2.Picture = pTab1.Picture
  Picture1.Visible = True

  tphone(0).Visible = True
  taddress(0).Visible = True
  tcity(0).Visible = True
  tregion(0).Visible = True
  tcountry(0).Visible = True
  tpost(0).Visible = True

  tphone(1).Visible = False
```

```
    taddress(1).Visible = False
    tcity(1).Visible = False
    tregion(1).Visible = False
    tcountry(1).Visible = False
    tpost(1).Visible = False

    tNotes.Visible = False

End Sub

Sub Option3D2_Click (Value As Integer)
    'Picture2.Picture = pTab2.Picture
    Picture1.Visible = True

    tphone(1).Visible = True
    taddress(1).Visible = True
    tcity(1).Visible = True
    tregion(1).Visible = True
    tcountry(1).Visible = True
    tpost(1).Visible = True

    tphone(0).Visible = False
    taddress(0).Visible = False
    tcity(0).Visible = False
    tregion(0).Visible = False
    tcountry(0).Visible = False
    tpost(0).Visible = False

    tNotes.Visible = False

End Sub
```

```
Sub Option3D3_Click (Value As Integer)
 ' Picture2.Picture = pTab3.Picture

  Picture1.Visible = False
  tNotes.Visible = True

End Sub

Sub Outline1_Click ()
  Dim stLName As String
  Dim stFName As String

  If Outline1.Indent(Outline1.ListIndex) = 2 Then
    stTmp$ = Outline1.List(Outline1.ListIndex)
    stLName = stGetToken$(stTmp$, ",")
    stFName = Right(stTmp$, Len(stTmp$) - 1)

    data1.Recordset.FindFirst "LastName='" + stLName + "' and FirstNAME='" + stFName +
"'"
  End If

End Sub

Sub Outline1_Collapse (i As Integer)
  Outline1.ListIndex = i
End Sub

Sub Outline1_DblClick ()
  If Outline1.Expand(Outline1.ListIndex) Then
    Outline1.Expand(Outline1.ListIndex) = False
  Else
    Outline1.Expand(Outline1.ListIndex) = True
```

```
  End If
End Sub

Sub Outline1_Expand (i As Integer)
  Outline1.ListIndex = i
End Sub

Sub Outline1_KeyPress (KeyAscii As Integer)
  If KeyAscii = 13 Then
    Outline1_DblClick
  End If
End Sub

Sub ProcessOutline ()
  For i% = 0 To Outline1.ListCount - 1
    If Outline1.HasSubItems(i%) Then
      Outline1.Expand(i%) = False
    End If
  Next i%
End Sub

Sub RefreshForm ()
  data1.RecordSource = "select * from PhoneList order by LastName,FirstName"
  data1.Refresh

  'Set DataField properties for control array
  tphone(0).DataField = "WorkPhone"
  taddress(0).DataField = "WorkAddress"
  tcity(0).DataField = "WorkCity"
  tregion(0).DataField = "WorkRegion"
  tcountry(0).DataField = "WorkCountry"
  tpost(0).DataField = "WorkPostalCode"
```

```
    tphone(1).DataField = "HomePhone"
    taddress(1).DataField = "HomeAddress"
    tcity(1).DataField = "HomeCity"
    tregion(1).DataField = "HomeRegion"
    tcountry(1).DataField = "HomeCountry"
    tpost(1).DataField = "HomePostalCode"

    FillList
    ProcessOutline

    If fStartUp Then
     Label1_Click
     SendKeys "{Home}"   'Move selection to top of Outline.
     fStartUp = False
    End If

End Sub

Function stGetID (ctrl As Control)
  stTxt$ = ctrl.Text
  i% = InStr(stTxt$, " ")
  stTmp$ = stTxt$
  Do While i% <> 0
    stTmp$ = Right$(stTmp$, Len(stTmp$) - i%)
    i% = InStr(stTmp$, " ")
  Loop
  stGetID = stTmp$
End Function

Function stGetToken (stLn$, stDelim$) As String
    On Error GoTo GetTokenError
```

```
    iOpenQuote% = InStr(1, stLn$, """")
    iDelim% = InStr(1, stLn$, stDelim$)

    If (iOpenQuote% > 0) And (iOpenQuote% < iDelim%) Then
        iCloseQuote% = InStr(iOpenQuote% + 1, stLn$, """")
        iDelim% = InStr(iCloseQuote% + 1, stLn$, stDelim$)
    End If

    If (iDelim% <> 0) Then
        stToken$ = LTrim$(RTrim$(Mid$(stLn$, 1, iDelim% - 1)))
        stLn$ = Mid$(stLn$, iDelim% + 1)
    Else
        stToken$ = LTrim$(RTrim$(Mid$(stLn$, 1)))
        stLn$ = ""
    End If

    If (Len(stToken$) > 0) Then
        If (Mid$(stToken$, 1, 1) = """") Then
            stToken$ = Mid$(stToken$, 2)
        End If
        If (Mid$(stToken$, Len(stToken$), 1) = """") Then
            stToken$ = Mid$(stToken$, 1, Len(stToken$) - 1)
        End If
    End If
    stGetToken = stToken$

GetTokenExit:
    Exit Function

GetTokenError:
    Resume GetTokenExit
```

```
End Function

Sub tAddress_KeyPress (Index As Integer, KeyAscii As Integer)
' data1.Enabled = False
' cNew.Enabled = False
' cDelete.Enabled = False
' cUpdate.Enabled = True

End Sub

Sub Text1_Click ()
    textold = ""
    text1.Text = ""
    CountSel = 0
If CheckNew Then

    tLName.Text = ""
    tFName.Text = ""
    taddress(0).Text = ""
    taddress(1).Text = ""
    tNotes.Text = ""
    tphone(0).Text = ""
    tphone(1).Text = ""
    tpost(0).Text = ""
    tpost(1).Text = ""
    tregion(0).Text = ""
    tregion(1).Text = ""
    tcountry(0).Text = ""
    tcountry(1).Text = ""
    taddress(0).Text = ""
    taddress(1).Text = ""
    tcity(0).Text = ""
```

```
    tcity(1).Text = ""
    textold = ""
  End If
 End Sub

 Sub Text1_KeyPress (KeyAscii As Integer)
  ' KeyHit = Hex(KeyAscii)
  ' text1.Text = text1.Text & KeyHit
  ' textold = textold & KeyHit
  ' text1.Text = textold
  ' keyascii = Asc(Chr(keyascii))

    text1.Text = Right(text1.Text, CountSel) & Hex(KeyAscii)
    CountSel = CountSel + 2
  If Not Newdata Then
   If Chr$(KeyAscii) = Chr$(13) Then
   CountSel = 0
   Bookmark = data1.Recordset.Bookmark
   Ake$ = "IDAress ='" + text1.Text + "'"
Agian:
   data1.Recordset.FindFirst Ake$
       If data1.Recordset.NoMatch Then
      data1.Recordset.Bookmark = Bookmark
   test = MsgBox("ไม่มีข้อมูลในโปรแกรม", 5)
   textold = ""
   If test = 4 Then GoTo Agian
       End If
'  CheckData = 0
'  MsgBox "OK"
    End If
 End If
End Sub
```

ภาคผนวก ค

DATA SHEET

7400, LS00, S00
Gates

Quad Two-Input NAND Gate
Product Specification

Logic Products

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
7400	9ns	8mA
74LS00	9.5ns	1.6mA
74S00	3ns	15mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE V_{CC} = 5V ±5%; T_A = 0°C to +70°C
Plastic DIP	N7400N, N74LS00N, N74S00N
Plastic SO	N74LS00D, N74S00D

NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

FUNCTION TABLE

INPUTS		OUTPUT
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

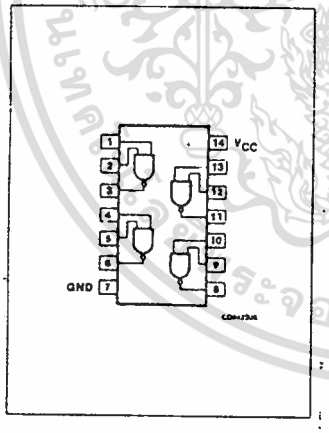
H = HIGH voltage level
L = LOW voltage level

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

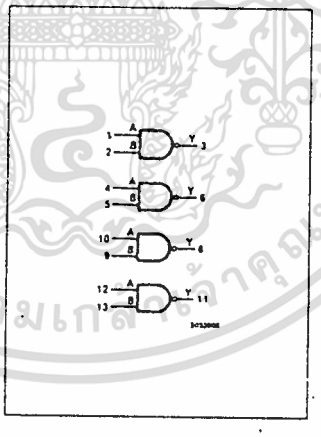
PINS	DESCRIPTION	74	74S	74LS
A, B	Inputs	1ul	1Sul	1LSul
Y	Output	10ul	10Sul	10LSul

NOTE:
Where a 74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, a 74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

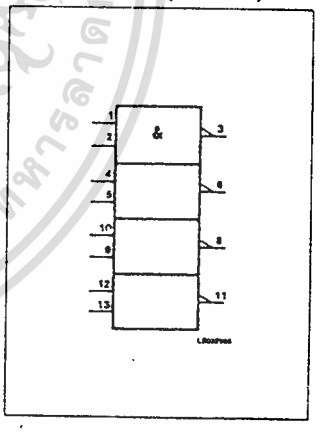
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



Logic Products

7404, LS04, S04
Inverters

Hex Inverter
Product Specification

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
7404	10ns	12mA
74LS04	9.5ns	2.4mA
74S04	3ns	22mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE V_{CC} = 5V ±5%; T_A = 0°C to +70°C
Plastic DIP	N7404N, N74LS04N, N74S04N
Plastic SO	N74LS04D, N74S04D

NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

FUNCTION TABLE

INPUT	OUTPUT
A	Y
L	H
H	L

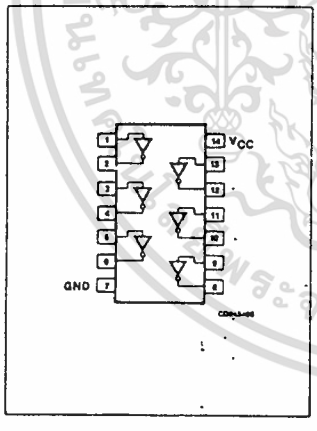
H = HIGH voltage level
L = LOW voltage level

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

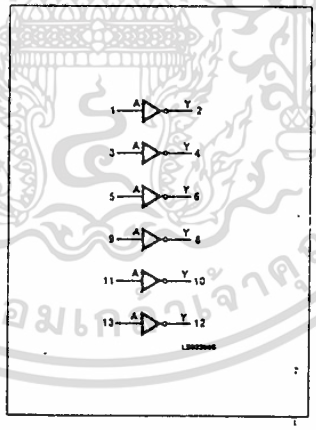
PINS	DESCRIPTION	74	74S	74LS
A	Input	1ul	1Sul	1LSul
Y	Output	10ul	10Sul	10LSul

NOTE:
Where a 74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, a 74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

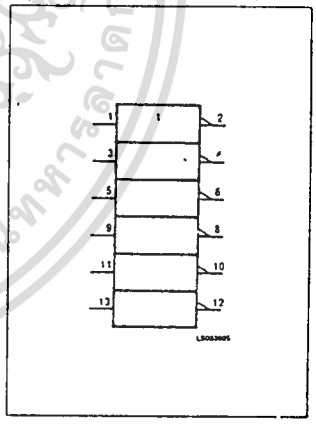
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



Logic Products

7406, 07
Inverter/Buffer/Drivers

'06 Hex Inverter Buffer/Driver (Open Collector)
'07 Hex Buffer/Driver (Open Collector)
Product Specification

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
7406	10ns (t_{PLH}) 15ns (t_{PHL})	31mA
7407	6ns (t_{PLH}) 20ns (t_{PHL})	25mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE V_{CC} = 5V ±5%; T_A = 0°C to +70°C
Plastic DIP	N7406N, N7407N
Plastic SO	N7406D, N7407D

NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

FUNCTION TABLE

'06		'07	
INPUT	OUTPUT	INPUT	OUTPUT
A	Y	A	Y
H	L	H	H
L	H	L	L

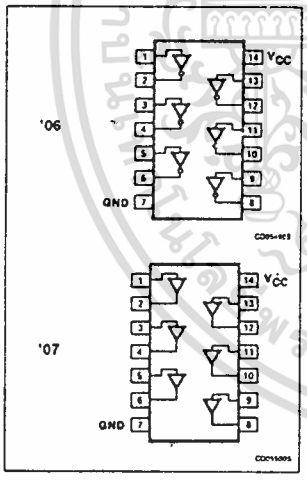
H = HIGH voltage level
L = LOW voltage level

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

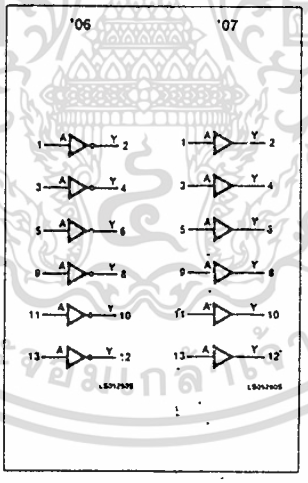
PINS	DESCRIPTION	74
A	Input	1ul
Y	Output	10ul

NOTE:
Where a 74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}.

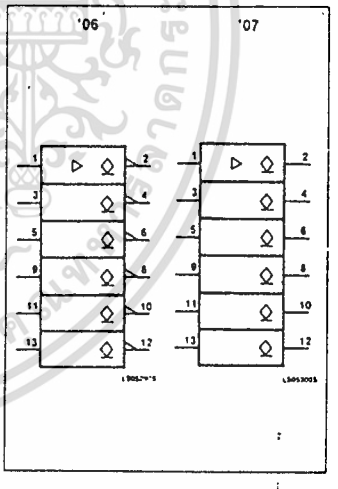
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



7474, LS74A, S74
Flip-Flops

Dual D-Type Flip-Flop
Product Specification

Logic Products

DESCRIPTION

The '74 is a dual positive edge-triggered D-type flip-flop featuring individual Data, Clock, Set and Reset inputs; also complementary Q and $\bar{Q}$ outputs.

Set ($\bar{S}_D$) and Reset ($\bar{R}_D$) are asynchronous active-LOW inputs and operate independently of the Clock input. Information on the Data (D) input is transferred to the Q output on the LOW-to-HIGH transition of the clock pulse. The D inputs must be stable one set-up time prior to the LOW-to-HIGH clock transition for predictable operation. Although the Clock input is level-sensitive, the positive transition of the clock pulse between the 0.8V and 2.0V levels should be equal to or less than the clock-to-output delay time for reliable operation.

TYPE	TYPICAL f_{MAX}	TYPICAL SUPPLY CURRENT (TOTAL)
7474	25MHz	17mA
74LS74A	33MHz	4mA
74S74	100MHz	30mA

NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

ORDERING CODE

PACKAGES	COMMERCIAL RANGE V_{CC} = 5V ±5%; T_A = 0°C to +70°C
Plastic DIP	,,474N, N74LS74AN, N74S74N
Plastic SO	N741S74A, N74S74D

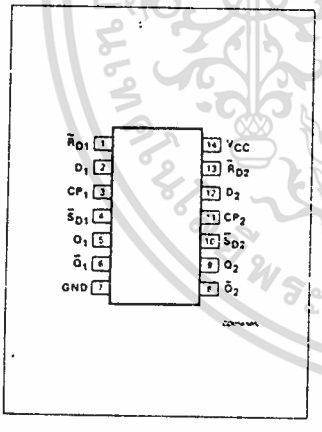
NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

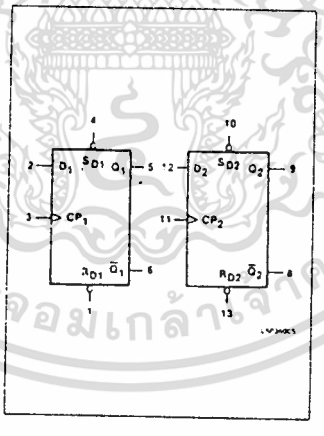
PINS	DESCRIPTION	74	74S	74LS
D	Input	1ul	1Sul	1LSul
$\bar{R}_D$	Input	2ul	3Sul	2LSul
$\bar{S}_D$	Input	1ul	2Sul	2LSul
CP	Input	2ul	2Sul	1LSul
Q,$\bar{Q}$	Outputs	10ul	10Sul	10LSul

NOTE:
Where a 74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, a 74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

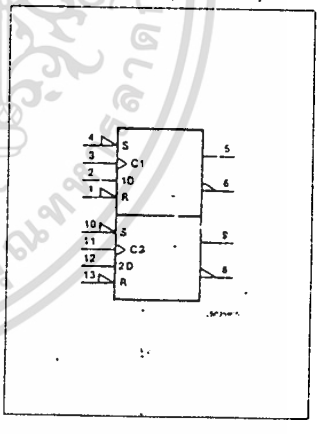
PIN CONFIGURATION



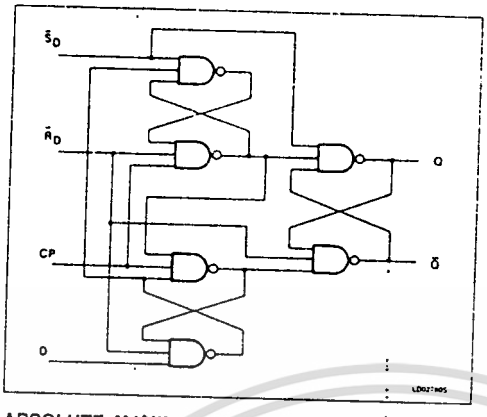
LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



LOGIC DIAGRAM



MODE SELECT — FUNCTION TABLE

OPERATING MODE	INPUTS				OUTPUTS	
	$\bar{S}_D$	$\bar{R}_D$	CP	D	Q	$\bar{Q}$
Asynchronous Set	L	H	X	X	H	L
Asynchronous Reset (Clear)	H	L	X	X	L	H
Undetermined[1]	L	L	X	X	H	H
Load "1" (Set)	H	H	↑	h	H	L
Load "0" (Reset)	H	H	↑	l	L	H

H = HIGH voltage level steady state.
h = HIGH voltage level one set-up time prior to the LOW-to-HIGH clock transition.
L = LOW voltage level steady state.
l = LOW voltage level one set-up time prior to the LOW-to-HIGH clock transition.
X = Don't care.
↑ = LOW-to-HIGH clock transition.

NOTE:
(1) Both outputs will be HIGH while both $\bar{S}_D$ and $\bar{R}_D$ are LOW, but the output states are unpredictable if $\bar{S}_D$ and $\bar{R}_D$ go HIGH simultaneously.

ABSOLUTE MAXIMUM RATINGS (Over operating free-air temperature range unless otherwise noted.)

	PARAMETER	74	74LS	74S	UNIT
V_{CC}	Supply voltage	7.0	7.0	7.0	V
V_{IN}	Input voltage	−0.5 to +5.5	−0.5 to +7.0	−0.5 to +5.5	V
I_{IN}	Input current	−30 to +5	−30 to +1	−30 to +5	mA
V_{OUT}	Voltage applied to output in HIGH output state	−0.5 to $+V_{CC}$	−0.5 to $+V_{CC}$	−0.5 to $+V_{CC}$	V
T_A	Operating free-air temperature range	0 to 70			°C

RECOMMENDED OPERATING CONDITIONS

	PARAMETER	74			74LS			74S			UNIT
		Min	Nom	Max	Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply voltage	4.75	5.0	5.25	4.75	5.0	5.25	4.75	5.0	5.25	V
V_{IH}	HIGH-level input voltage	2.0			2.0			2.0			V
V_{IL}	LOW-level input voltage			+0.8			+0.8			+0.8	V
I_{IK}	Input clamp current			−12			−18			−18	mA
I_{OH}	HIGH-level output current			−400			−400			−1000	μA
I_{OL}	LOW-level output current			16			8			20	mA
T_A	Operating free-air temperature	0		70	0		70	0		70	°C

Decoders/Demu

Logic Products

74LS139, S139
Decoders/Demultiplexers

Dual 1-of-4 Decoder/Demultiplexer
Product Specification

FEATURES

- **Demultiplexing capability**
- **Two independent 1-of-4 decoders**
- **Multifunction capability**
- **Replaces 9321 and 93L21 for higher performance**

DESCRIPTION

The '139 is a high-speed, dual 1-of-4 decoder/demultiplexer. This device has two independent decoders, each accepting two binary weighted inputs (A_0, A_1) and providing four mutually exclusive active LOW outputs ($\overline{0} - \overline{3}$). Each decoder has an active LOW Enable ($\overline{E}$). When $\overline{E}$ is HIGH, every output is forced HIGH. The Enable can be used as the Data input for a 1-of-4 demultiplexer application.

TYPE	TYPICAL PROPAGATION DELAY (ENABLE AT 2 LOGIC LEVELS)	TYPICAL SUPPLY CURRENT (TOTAL)
74LS139	19ns	6.8mA
74S139	6ns	60mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE $V_{CC} = 5V \pm 5\%$; $T_A = 0°C$ to $+70°C$
Plastic DIP	N74S139N, N74LS139N
Plastic SO	N74LS139D, N74S139D

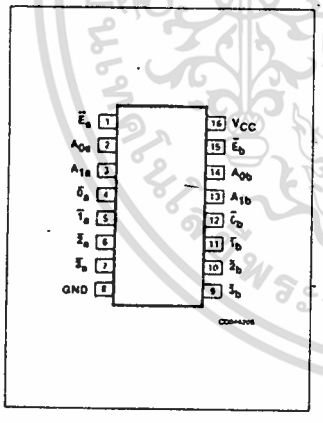
NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

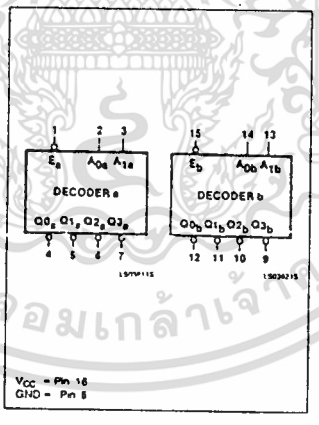
PINS	DESCRIPTION	74S	74LS
All	Inputs	1Sul	1LSul
All	Outputs	10Sul	10LSul

NOTE:
A 74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and a 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

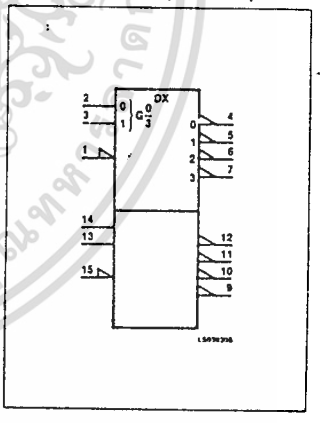
PIN CONFIGURATION



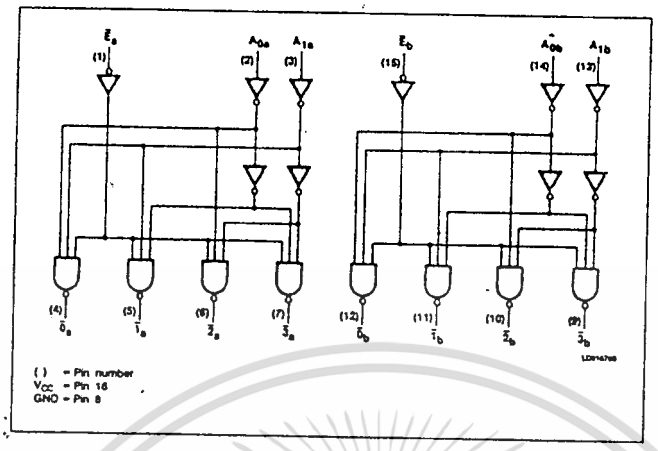
LOGIC SYMBOL



LOGIC SYMBOL (EEE/IEC)



LOGIC DIAGRAM



FUNCTION TABLE

INPUTS			OUTPUTS			
$\overline{E}$	A_0	A_1	$\overline{O}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH voltage level
L = LOW voltage level

ABSOLUTE MAXIMUM RATINGS (Over operating free-air temperature range unless otherwise noted.)

PARAMETER		74LS	74S	UNIT
V_{CC}	Supply voltage	7.0	7.0	V
V_{IN}	Input voltage	−0.5 to +7.0	−0.5 to +5.5	V
I_{IN}	Input current	−30 to +1	−30 to +5	mA
V_{OUT}	Voltage applied to output in HIGH output state	−0.5 to +V_{CC}	−0.5 to +V_{CC}	V
T_A	Operating free-air temperature range	0 to 70		°C

RECOMMENDED OPERATING CONDITIONS

PARAMETER		74LS			74S			UNIT
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply voltage	4.75	5.0	5.25	4.75	5.0	5.25	V
V_{IH}	HIGH-level input voltage	2.0			2.0			V
V_{IL}	LOW-level input voltage			+0.8			+0.8	V
I_{IK}	Input clamp current			−18			−18	mA
I_{OH}	HIGH-level output current			−400			−1000	µA
I_{OL}	LOW-level output current			8			20	mA
T_A	Operating free-air temperature	0		70	0		70	°C

74LS373, 74LS374, S373, S374
Latches/Flip-Flops

'373 Octal Transparent Latch With 3-State Outputs
'374 Octal D Flip-Flop With 3-State Outputs
Product Specification

Logic Products

FEATURES
- 8-bit transparent latch — '373
- 8-bit positive, edge-triggered register — '374
- 3-State output buffers
- Common 3-State Output Enable
- Independent register and 3-State buffer operation

DESCRIPTION
The '373 is an octal transparent latch coupled to eight 3-State output buffers. The two sections of the device are controlled independently by Latch Enable (E) and Output Enable ($\overline{OE}$) control gates.

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
74LS373	19ns	24mA
74S373	10ns	105mA
74LS374	19ns	27mA
74S374	8ns	116mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGE V_{CC} = 5V ±5%; T_A = 0°C to +70°C
Plastic DIP	N74LS373N, N74S373N, N74LS374N, N74S374N
Plastic SOL-20	N74LS373D, N74S373D, N74LS374D, N74S374D

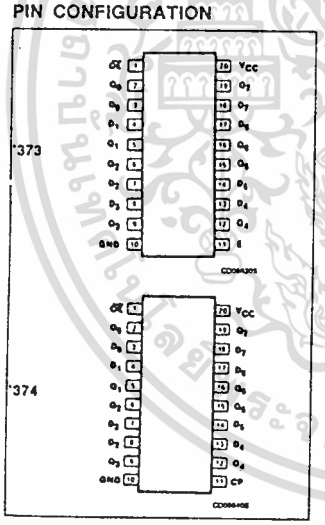
NOTE:
For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

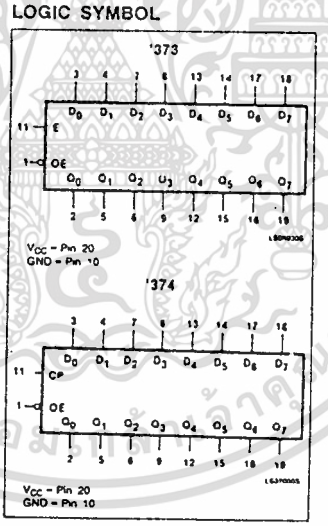
PINS	DESCRIPTION	74S	74LS
All	Inputs	1Sul	1LSul
All	Outputs	10Sul	30LSul

NOTE:
Where a 74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and a 74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

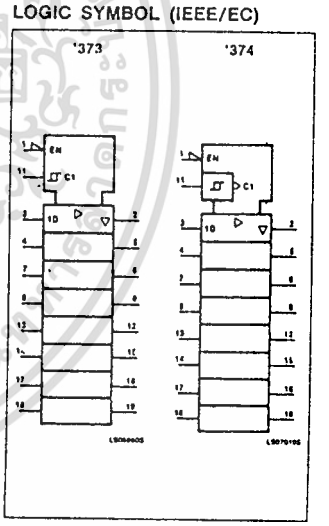
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/EC)



Latches/Flip-Flops

74LS373, 74LS374, S373, S374

The data on the D inputs are transferred to the latch outputs when the Latch Enable (E) input is HIGH. The latch remains transparent to the data inputs while E is HIGH, and stores the data present one set-up time before the HIGH-to-LOW enable transition. The enable gate has hysteresis built in to help minimize problems that signal and ground noise can cause on the latching operation.

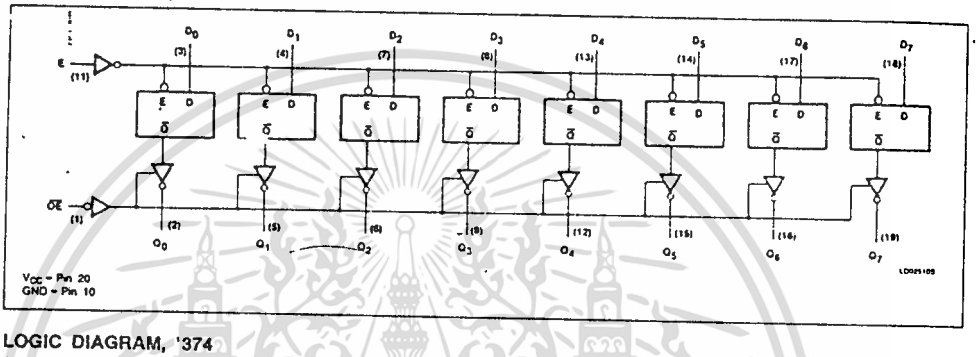
The 3-State output buffers are designed to drive heavily loaded 3-State buses, MOS memories, or MOS microprocessors. The active LOW Output Enable ($\overline{OE}$) controls all eight 3-State buffers independent of the latch operation. When $\overline{OE}$ is LOW, the latched or transparent data appears at the outputs. When $\overline{OE}$ is HIGH, the outputs are in the HIGH impedance "off" state, which means they will neither drive nor load the bus.

The '374 is an 8-bit, edge-triggered register coupled to eight 3-State output buffers. The two sections of the device are controlled independently by the Clock (CP) and Output Enable ($\overline{OE}$) control gates.

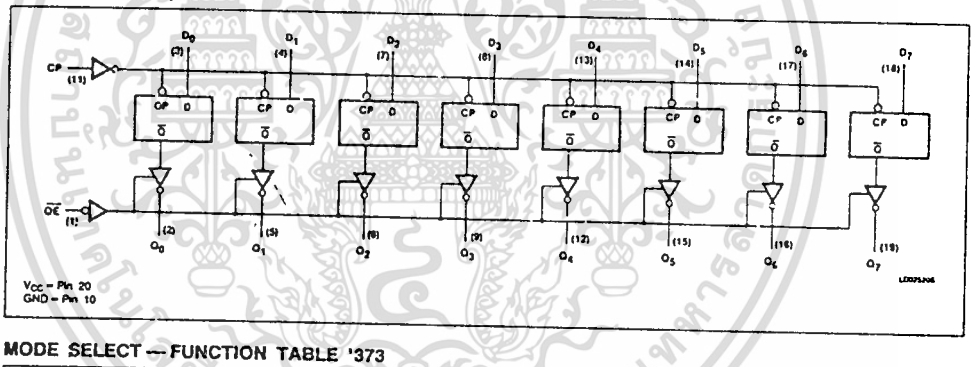
The register is fully edge triggered. The state of each D input, one set-up time before the LOW-to-HIGH clock transition, is transferred to the corresponding flip-flop's Q output. The clock buffer has hysteresis built in to help minimize problems that signal and ground noise can cause on the clocking operation.

The 3-State output buffers are designed to drive heavily loaded 3-State buses, MOS memories, or MOS microprocessors. The active LOW Output Enable ($\overline{OE}$) controls all eight 3-State buffers independent of the register operation. When $\overline{OE}$ is LOW, the data in the register appears at the outputs. When $\overline{OE}$ is HIGH, the outputs are in the HIGH impedance "off" state, which means they will neither drive nor load the bus.

LOGIC DIAGRAM, '373



LOGIC DIAGRAM, '374



MODE SELECT — FUNCTION TABLE '373

OPERATING MODES	INPUTS			INTERNAL REGISTER	OUTPUTS
	$\overline{OE}$	E	D_n		$Q_0 - Q_7$
Enable and read register	L	H	L	L	L
	L	H	H	H	H
Latch and read register	L	L	l	L	L
	L	L	h	H	H
Latch register and disable outputs	H	L	l	L	(Z)
	H	L	h	H	(Z)

intel®

8085AH/8085AH-2/8085AH-1
8-BIT HMOS MICROPROCESSORS

- Single +5V Power Supply with 10% Voltage Margins
- 3 MHz, 5 MHz and 6 MHz Selections Available
- 20% Lower Power Consumption than 8085A for 3 MHz and 5 MHz
- 1.3 μs Instruction Cycle (8085AH); 0.8 μs (8085AH-2); 0.67 μs (8085AH-1)
- 100% Compatible with 8085A
- 100% Software Compatible with 8080A
- On-Chip Clock Generator (with External Crystal, LC or RC Network)
- On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control
- Four Vectored Interrupt Inputs (One is Non-Maskable) Plus an 8080A-Compatible Interrupt
- Serial In/Serial Out Port
- Decimal, Binary and Double Precision Arithmetic
- Direct Addressing Capability to 64K Bytes of Memory
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8085AH is a complete 8 bit parallel Central Processing Unit (CPU) implemented in N-channel, depletion load, silicon gate technology (HMOS). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085AH (CPU), 8156H (RAM/IO) and 8355/8755A (ROM/PROM/IO)] while maintaining total system expandability. The 8085AH-2 and 8085AH-1 are faster versions of the 8085AH.

The 8085AH incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a high level of system integration.

The 8085AH uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8155H/8156H/8355/8755A memory products allow a direct interface with the 8085AH

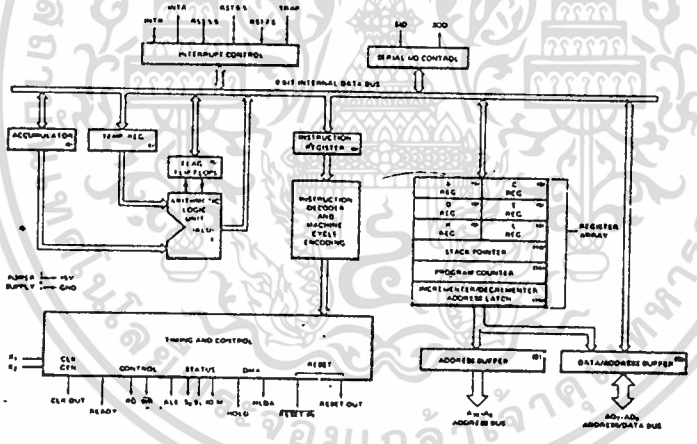


Figure 1. 8085AH CPU Functional Block Diagram

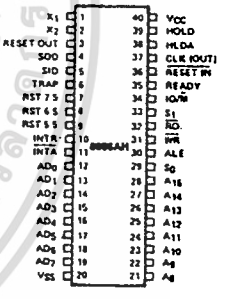


Figure 2. 8085AH Pin Configuration

Table 1. Pin Description

Symbol	Type	Name and Function
A_8–A_{15}	O	**Address Bus:** The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET
AD_0—7	I/O	**Multiplexed Address/Data Bus:** Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles
ALE	O	**Address Latch Enable:** It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1, and IO/$\overline{M}$	O	**Machine Cycle Status:** IO/$\overline{M}$ S_1 S_0 Status 0 0 1 Memory write 0 1 0 Memory read 1 0 1 I/O write 1 1 0 I/O read 0 1 1 Opcode fetch 1 1 1 Opcode fetch 1 1 1 Interrupt Acknowledge * 0 0 Halt * X X Hold * X X Reset * = 3-state (high impedance) X = unspecified S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines
$\overline{RD}$	O	**Read Control:** A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer. 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$	O	**Write Control:** A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET

Symbol	Type	Name and Function
READY	I	**Ready:** If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times
HOLD	I	**Hold:** Indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA	O	**Hold Acknowledge:** Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low
INTR	I	**Interrupt Request:** Is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.
$\overline{INTA}$	O	**Interrupt Acknowledge:** Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate an 8259A Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5	I	**Restart Interrupts:** These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 2. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction

Table 1. Pin Description (Continued)

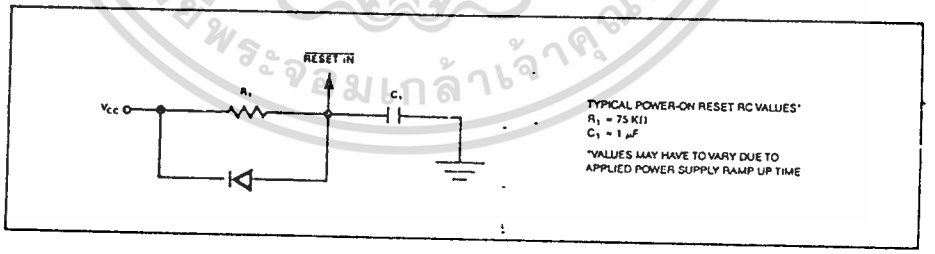
Symbol	Type	Name and Function
TRAP	I	**Trap:** Trap interrupt is a non-maskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 2.)
$\overline{\text{RESET IN}}$	I	**Reset In:** Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{\text{RESET IN}}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay (see Figure 3). Upon power-up, $\overline{\text{RESET IN}}$ must remain low for at least 10 ms after minimum V_{CC} has been reached. For proper reset operation after the power-up duration, $\overline{\text{RESET IN}}$ should be kept low a minimum of three clock periods. The CPU is held in the reset condition as long as $\overline{\text{RESET IN}}$ is applied.

Symbol	Type	Name and Function
RESET OUT	O	**Reset Out:** Reset Out indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2	I	X_1 and X_2: Are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK	O	**Clock:** Clock output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID	I	**Serial Input Data Line:** The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD	O	**Serial Output Data Line:** The output SOD is set or reset as specified by the SIM instruction.
V_{CC}		**Power:** +5 volt supply.
V_{SS}		**Ground:** Reference.

Table 2. Interrupt Priority, Restart Address, and Sensitivity

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2).	High level until sampled.

NOTES:
1. The processor pushes the PC on the stack before branching to the indicated address.
2. The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.



FUNCTIONAL DESCRIPTION

The 8085AH is a complete 8-bit parallel central processor. It is designed with N-channel, depletion load, silicon gate technology (HMOS), and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085AH), 5 MHz (8085AH-2), or 6 MHz (8085AH-1), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU (8085AH), a RAM/IO (8156H), and a ROM or EPROM/IO chip (8355 or 8755A).

The 8085AH has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085AH register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bits
PC	Program Counter	16-bit address
BC,DE,HL	General-Purpose Registers; data pointer (HL)	8 bits x 6 or 16 bits x 3
SP	Stack Pointer	16-bit address
Flags or F	Flag Register	5 flags (8-bit space)

The 8085AH uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085AH provides $\overline{RD}$, $\overline{WR}$, S_0, S_1, and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD and all interrupts are synchronized with the processor's internal clock. The 8085AH also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085AH has three maskable, vector interrupt pins, one nonmaskable TRAP interrupt, and a bus vectored interrupt, INTR.

INTERRUPT AND SERIAL I/O

The 8085AH has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 2.)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive.*

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request (a normally high level signal with a low going pulse is recommended for highest system noise immunity). The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a $\overline{RESET\ IN}$ to the 8085AH. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{RESET\ IN}$. (See SIM, Chapter 5 of the MCS-80/85 User's Manual.)

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP—highest priority, RST 7.5, RST 6.5, RST 5.5, INTR—lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive*. The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 4 illustrates the TRAP interrupt request circuitry within the 8085AH. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

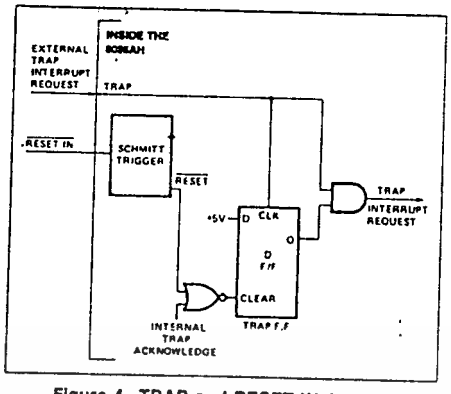


Figure 4. TRAP and RESET IN Circuit

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that interrupts are disabled. See the description of the RIM instruction in the MCS-80/85 Family User's Manual.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085AH, 8085AH-2, or 8085AH-1 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The crystal frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085AH is operated with a 6 MHz crystal (for 3 MHz clock), the 8085AH-2 operated with a 10 MHz crystal (for 5 MHz clock), and the 8085AH-1 can be operated with a 12 MHz crystal (for 6 MHz clock). If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired
C_L (load capacitance) ≤ 30 pF
C_S (shunt capacitance) ≤ 7 pF
R_S (equivalent shunt resistance) ≤ 75 Ohms
Drive level: 10 mW
Frequency tolerance: ±.005% (suggested)

Note the use of the 20 pF capacitor between X_2 and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the 8085AH, providing that its frequency tolerance of approximately ±10% is acceptable. The components are chosen from the formula:

$$f = \frac{1}{2\pi\sqrt{L(C_{ext} + C_{int})}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

An RC circuit may be used as the frequency-determining network for the 8085AH if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 5 shows the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4V and maximum low level voltage of 0.8V.

For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figure 5D). If the driving frequency is from 6 MHz to 12 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 5E). To prevent self-oscillation of the 8085AH, be sure that X_2 is not coupled back to X_1 through the driving circuit.

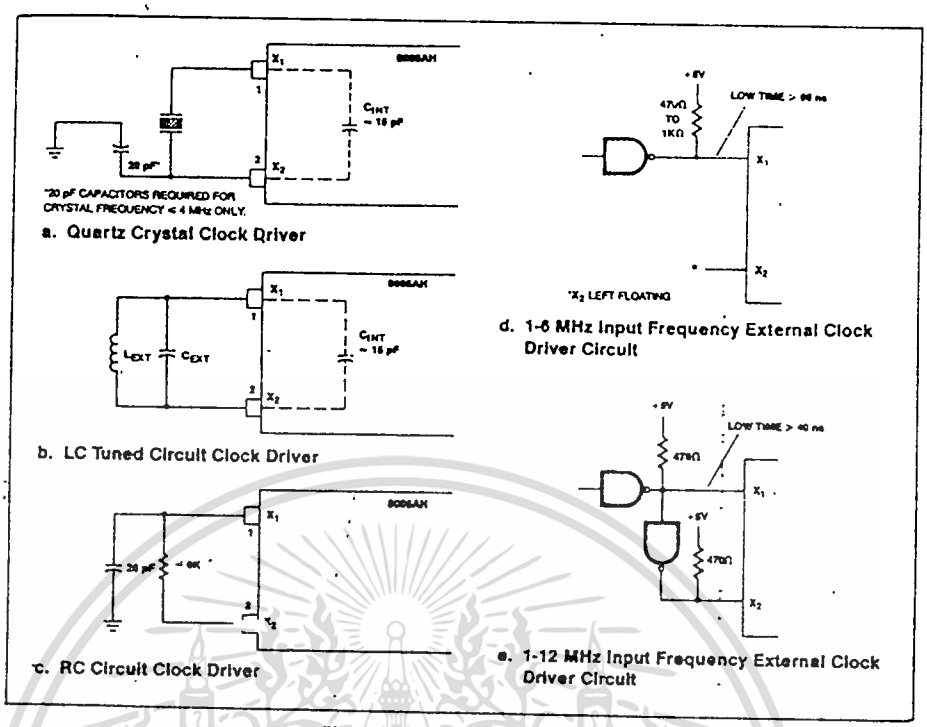


Figure 5, Clock Driver Circuits

GENERATING AN 8085AH WAIT STATE

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in Figure 6 may be used to insert one WAIT state in each 8085AH machine cycle.

The D flip-flops should be chosen so that
- CLK is rising edge-triggered
- CLEAR is low-level active.

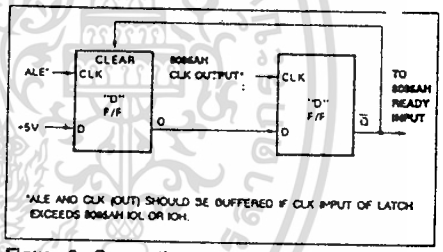


Figure 6. Generation of a Wait State for 8085AH CPU

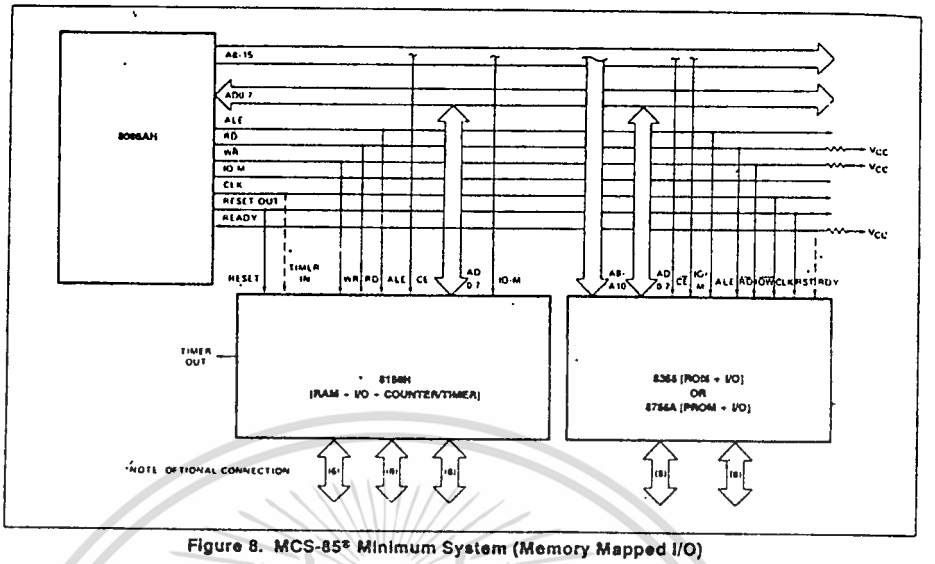


Figure 8. MCS-85® Minimum System (Memory Mapped I/O)

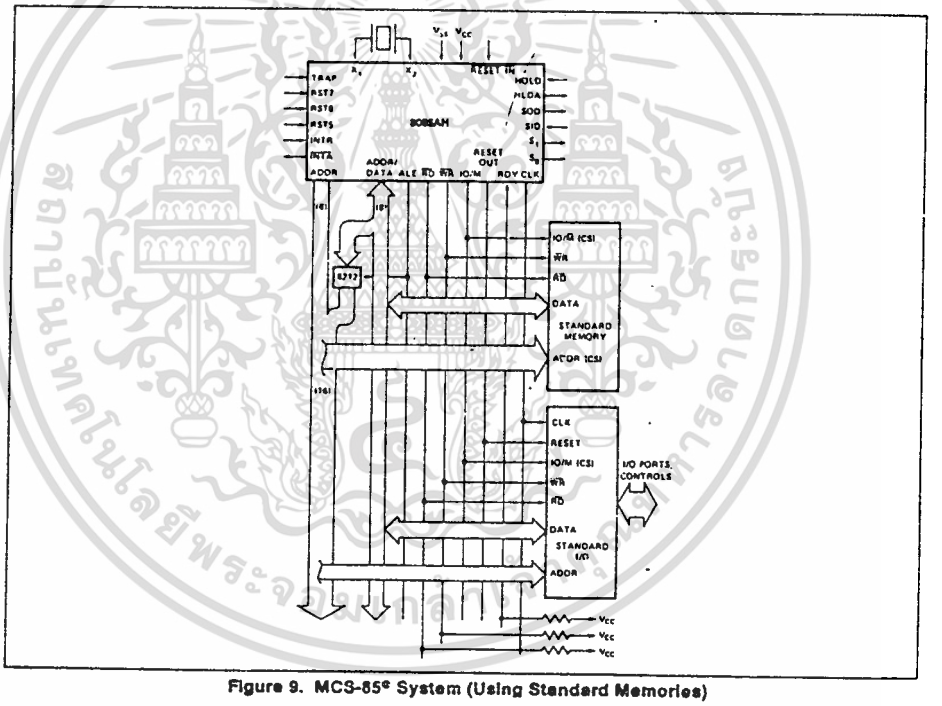


Figure 9. MCS-85® System (Using Standard Memories)

As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085AH can be used with slow memory. HOLD causes the CPU to relinquish the bus when it is through with it by floating the Address and Data Buses.

SYSTEM INTERFACE

The 8085AH family includes memory components, which are directly compatible to the 8085AH CPU. For example, a system consisting of the three chips, 8085AH, 8156H, and 8355 will have the following features:

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

This minimum system, using the standard I/O technique is as shown in Figure 7.

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. Figure 8 shows the system configuration of Memory Mapped I/O using 8085AH.

The 8085AH CPU can also interface with the standard memory that does *not* have the multiplexed address/data bus. It will require a simple 8212 (8-bit latch) as shown in Figure 9.

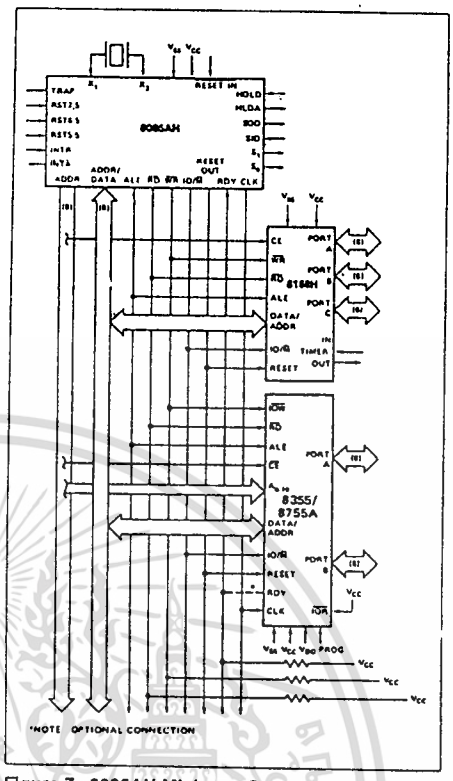


Figure 7. 8085AH Minimum System (Standard I/O Technique)

BASIC SYSTEM TIMING

The 8085AH has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 10 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ($IO/\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$). (See Table 3.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each machine cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of READY or HOLD inputs). Any T state must be one of ten possible states, shown in Table 4.

Table 3. 8085AH Machine Cycle Chart

MACHINE CYCLE		STATUS			CONTROL		
		IO/M	S1	S0	RD	WR	INTA
OPCODE FETCH	(OF)	0	1	1	0	1	1
MEMORY READ	(MR)	0	1	0	0	1	1
MEMORY WRITE	(MW)	0	0	1	1	0	1
I/O READ	(IOR)	1	1	0	0	1	1
I/O WRITE	(IOW)	1	0	1	1	0	1
ACKNOWLEDGE OF INTR	(INA)	1	1	1	1	1	0
BUS IDLE	(BI): DAD	0	1	0	1	1	1
	ACK. OF RST,TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

Table 4. 8085AH Machine State Chart

Machine State	Status & Buses				Control		
	S1,S0	IO/M	A_8-A_{15}	AD_0-AD_7	RD,WR	INTA	ALE
T_1	X	X	X	X	1	1	1*
T_2	X	X	X	X	X	X	0
T_{WAIT}	X	X	X	X	X	X	0
T_3	X	X	X	X	X	X	0
T_4	1	0†	X	TS	1	1	0
T_5	1	0†	X	TS	1	1	0
T_6	1	0†	X	TS	1	1	0
T_{RESET}	X	TS	TS	TS	TS	1	0
T_{HALT}	0	TS	TS	TS	TS	1	0
T_{HOLD}	X	TS	TS	TS	TS	1	0

0 = Logic "0"
1 = Logic "1"
TS = High Impedance
X = Unspecified

* ALE not generated during 2nd and 3rd machine cycles of DAD instruction
† IO/M = 1 during T_4-T_6 of INA machine cycle.

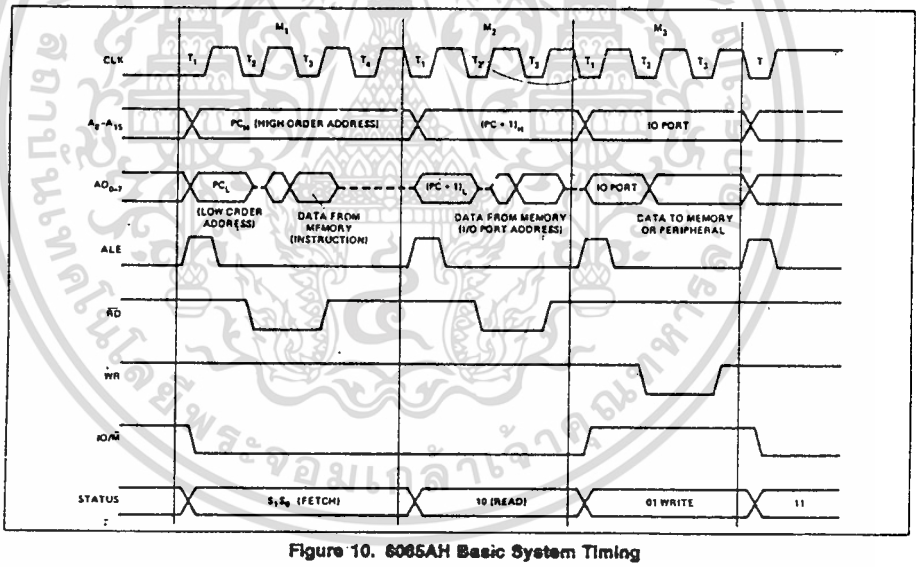


Figure 10. 8085AH Basic System Timing

Table 6. Instruction Set Summary

Mnemonic	Instruction Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Operations Description
MOVE, LOAD, AND STORE									
MOV r1 r2	0	1	D	D	D	S	S	S	Move register to register
MOV M,r	0	1	1	1	0	S	S	S	Move register to memory
MOV r,M	0	1	D	D	D	1	1	0	Move memory to register
MVI r	0	0	D	D	D	1	1	0	Move immediate register
MVI M	0	0	1	1	0	1	1	0	Move immediate memory
LXI B	0	0	0	0	0	0	0	1	Load immediate register Pair B & C
LXI D	0	0	0	1	0	0	0	1	Load immediate register Pair D & E
LXI H	0	0	1	0	0	0	0	1	Load immediate register Pair H & L
STAX B	0	0	0	0	0	0	1	0	Store A indirect
STAX D	0	0	0	1	0	0	1	0	Store A indirect
LDAX B	0	0	0	0	1	0	1	0	Load A indirect
LDAX D	0	0	0	1	1	0	1	0	Load A indirect
STA	0	0	1	1	0	0	1	0	Store A direct
LDA	0	0	1	1	1	0	1	0	Load A direct
SHLD	0	0	1	0	0	0	1	0	Store H & L direct
LHLD	0	0	1	0	1	0	1	0	Load H & L direct
XCHG	1	1	1	0	1	0	1	1	Exchange D & E, H & L Registers
STACK OPS									
PUSH B	1	1	0	0	0	1	0	1	Push register Pair B & C on stack
PUSH D	1	1	0	1	0	1	0	1	Push register Pair D & E on stack
PUSH H	1	1	1	0	0	1	0	1	Push register Pair H & L on stack
PUSH PSW	1	1	1	1	0	1	0	1	Push A and Flags on stack
POP B	1	1	0	0	0	0	0	1	Pop register Pair B & C off stack
POP D	1	1	0	1	0	0	0	1	Pop register Pair D & E off stack
POP H	1	1	1	0	0	0	0	1	Pop register Pair H & L off stack
POP PSW	1	1	1	1	0	0	0	1	Pop A and Flags off stack
XTHL	1	1	1	0	0	0	1	1	Exchange top of stack, H & L
SPHL	1	1	1	1	1	0	0	1	H & L to stack pointer
LXI SP	0	0	1	1	0	0	0	1	Load immediate stack pointer
INX SP	0	0	1	1	0	0	1	1	Increment stack pointer
DCX SP	0	0	1	1	1	0	1	1	Decrement stack pointer
JUMP									
JMP	1	1	0	0	0	0	1	1	Jump unconditional
JC	1	1	0	1	1	0	1	0	Jump on carry
JNC	1	1	0	1	0	0	1	0	Jump on no carry
JZ	1	1	0	0	1	0	1	0	Jump on zero
JNZ	1	1	0	0	0	0	1	0	Jump on no zero
JP	1	1	1	1	0	0	1	0	Jump on positive
JM	1	1	1	1	1	0	1	0	Jump on minus
JPE	1	1	1	0	1	0	1	0	Jump on parity even
JPO	1	1	1	0	0	0	1	0	Jump on parity odd
PCHL	1	1	1	0	1	0	0	1	H & L to program counter
CALL									
CALL	1	1	0	0	1	1	0	1	Call unconditional
CC	1	1	0	1	1	1	0	0	Call on carry
CNC	1	1	0	1	0	1	0	0	Call on no carry
CZ	1	1	0	0	1	1	0	0	Call on zero
CNZ	1	1	0	0	0	1	0	0	Call on no zero
CP	1	1	1	1	0	1	0	0	Call on positive
CM	1	1	1	1	1	1	0	0	Call on minus
CPE	1	1	1	0	1	1	0	0	Call on parity even
CPO	1	1	1	0	0	1	0	0	Call on parity odd
RETURN									
RET	1	1	0	0	1	0	0	1	Return
RC	1	1	0	1	1	0	0	0	Return on carry
RNC	1	1	0	1	0	0	0	0	Return on no carry
RZ	1	1	0	0	1	0	0	0	Return on zero
RNZ	1	1	0	0	0	0	0	0	Return on no zero
RP	1	1	1	1	0	0	0	0	Return on positive
RM	1	1	1	1	1	0	0	0	Return on minus
RPE	1	1	1	0	1	0	0	0	Return on parity even
RPO	1	1	1	0	0	0	0	0	Return on parity odd
RESTART									
RST	1	1	A	A	A	1	1	1	Restart
INPUT/OUTPUT									
IN	1	1	0	1	1	0	1	1	Input
OUT	1	1	0	1	0	0	1	1	Output
INCREMENT AND DECREMENT									
INR r	0	0	D	D	D	1	0	0	Increment register
DCR r	0	0	D	D	D	1	0	1	Decrement register
INR M	0	0	1	1	0	1	0	0	Increment memory
DCR M	0	0	1	1	0	1	0	1	Decrement memory
INX B	0	0	0	0	0	0	1	1	Increment B & C registers
INX D	0	0	0	1	0	0	1	1	Increment D & E registers
INX H	0	0	1	0	0	0	1	1	Increment H & L registers
DCX B	0	0	0	0	1	0	1	1	Decrement B & C
DCX D	0	0	0	1	1	0	1	1	Decrement D & E
DCX H	0	0	1	0	1	0	1	1	Decrement H & L
ADD									
ADD r	1	0	0	0	0	S	S	S	Add register to A
ADC r	1	0	0	0	1	S	S	S	Add register to A with carry
ADD M	1	0	0	0	0	1	1	0	Add memory to A
ADC M	1	0	0	0	1	1	1	0	Add memory to A with carry
ADI	1	1	0	0	0	1	1	0	Add immediate to A
ACI	1	1	0	0	1	1	1	0	Add immediate to A with carry
DAD B	0	0	0	0	1	0	0	1	Add B & C to H & L
DAD D	0	0	0	1	1	0	0	1	Add D & E to H & L
DAD H	0	0	1	0	1	0	0	1	Add H & L to H & L
DAD SP	0	0	1	1	1	0	0	1	Add stack pointer to H & L
SUBTRACT									
SUB r	1	0	0	1	0	S	S	S	Subtract register from A
SBB r	1	0	0	1	1	S	S	S	Subtract register from A with borrow
SUB M	1	0	0	1	0	1	1	0	Subtract memory from A
SBB M	1	0	0	1	1	1	1	0	Subtract memory from A with borrow
SUI	1	1	0	1	0	1	1	0	Subtract immediate from A
SBI	1	1	0	1	1	1	1	0	Subtract immediate from A with borrow

Table 6. Instruction Set Summary (Continued)

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
LOGICAL		
ANA r	1 0 1 0 0 S S S	And register with A
XRA r	1 0 1 0 1 S S S	Exclusive OR register with A
ORA r	1 0 1 1 0 S S S	OR register with A
CMP r	1 0 1 1 1 S S S	Compare register with A
ANA M	1 0 1 0 0 1 1 0	And memory with A
XRA M	1 0 1 0 1 1 1 0	Exclusive OR memory with A
ORA M	1 0 1 1 0 1 1 0	OR memory with A
CMP M	1 0 1 1 1 1 1 0	Compare memory with A
ANI	1 1 1 0 0 1 1 0	And immediate with A
XRI	1 1 1 0 1 1 1 0	Exclusive OR immediate with A
ORI	1 1 1 1 0 1 1 0	OR immediate with A
CPI	1 1 1 1 1 1 1 0	Compare immediate with A
ROTATE		
RLC	0 0 0 0 0 1 1 1	Rotate A left
RRC	0 0 0 0 1 1 1 1	Rotate A right
RAL	0 0 0 1 0 1 1 1	Rotate A left through carry
RAR	0 0 0 1 1 1 1 1	Rotate A right through carry

Mnemonic	Instruction Code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Operations Description
SPECIALS		
CMA	0 0 1 0 1 1 1 1	Complement A
STC	0 0 1 1 0 1 1 1	Set carry
CMC	0 0 1 1 1 1 1 1	Complement carry
DAA	0 0 1 0 0 1 1 1	Decimal adjust A
CONTROL		
EI	1 1 1 1 1 0 1 1	Enable Interrupts
DI	1 1 1 1 0 0 1 1	Disable Interrupt
NOP	0 0 0 0 0 0 0 0	No-operation
HLT	0 1 1 1 0 1 1 0	Halt
NEW 8085A INSTRUCTIONS		
RIM	0 0 1 0 0 0 0 0	Read Interrupt Mask
SIM	0 0 1 1 0 0 0 0	Set Interrupt Mask

NOTES:
1. DDS or SSS: B 000, C 001, D 010, E011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times (6/12) indicate instruction cycles dependent on condition flags.

Unspecified 8085A Op Codes

NEW 8085 INSTRUCTIONS

NEW CONDITION CODES V = bit 1, X5 = bit 5

Condition code format

S	Z	X5	AC	0	P	V	C

2's complement overflow
Underflow (DCX) or overflow (INX)
X5 = 01·02 + 01·R + 02·R, where
01 = sign of operand 1, 02 = sign of operand 2,
R = sign of result. For subtraction and comparisons, replace 02 with $\overline{02}$

DSUB (double subtraction)

(H) (L) = (H) (L) - (B) (C)

The contents of register pair B and C are subtracted from the contents of register pair H and L. The result is placed in register pair H and L. All condition flags are affected.

0 0 0 0 1 0 0 0	(08)

cycles	3
states	10
addressing	register
flags	Z, S, P, CY, AC, X5, V

ARHL (arithmetic shift of H and L to the right)

(H7=H7), (Hn-1) = (Hn)
(L7=Ho), (Ln-1) = (Ln), (CY) = (Lo)

The contents of register pair H and L are shifted right one bit. The uppermost bit is duplicated and the lowest bit is shifted into the carry bit. The result is placed in register pair H and L. Note: only the CY flag is affected.

0 0 0 1 0 0 0 0	(10)

cycles	2
states	7
addressing	register
flags	CY

RDEL (rotate D and E left through carry)

(Dn+1) = (Dn); (Do) = (E7)
(CY) = (D7), (En+1) = (En), (Eo) = (CY)

The contents of register pair D and E are rotated left one position through the carry flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. Only the CY and the V flags are affected.

0 0 0 1 1 0 0 0	(18)

cycles	3
states	10
addressing	register
flags	CY, V

LDHI (load D and E with H and L plus immediate byte)

(D) (E) = (H) (L) + (byte 2)

The contents of register pair H and L are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected.

0 0 1 0 1 0 0 0
data

(28)

cycles	3
states	10
addressing	immediate register
flags	none

LDSI (load D and E with SP plus immediate byte)

(D) (E) = (SPH)(SPL) + (byte 2)

The contents of register pair SP are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected.

0 0 1 1 1 0 0 0
data

(38)

cycles	3
states	10
addressing	immediate register
flags	none

RSTV (restart on overflow)

If (V)
((SP)-1) = (PCH)
((SP)-2) = (PCL)
(SP) = (SP)-2
(PC) = 40 hex

If the overflow flag V is set, the actions specified above are performed, otherwise control continues sequentially.

1 1 0 0 1 0 1 1	(CB)

cycles	1 or 3
states	6 or 12
addressing	register indirect
flags	none

SHLX (store H and L indirect through D and E)

((D)(E)) = (L)
((D)(E)+1) = (H)

The contents of register L are moved to the memory location whose address is in register pair D and E. The contents of register H are moved to the succeeding memory location.

1 1 0 1 1 0 0 1	(D9)

cycles	3
states	10
addressing	register indirect
flags	none

JnX5 (jump on not X5)

If (not X5)
(PC) = (byte 3) (byte 2)

If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction, otherwise control continues sequentially.

1 1 0 1 1 1 0 1	(DD)
low-order address	
high order address	

cycles	2 or 3
states	7 or 10
addressing	immediate
flags	none

LHLX (load H and L indirect through D and E)

(L) = ((D)(E))
(H) = ((D)(E)+1)

The content of the memory location whose address is in D and E, are moved to register L. The contents of the succeeding memory location are moved to register H.

1 1 1 0 1 1 0 1	(ED)

cycles	3
states	10
addressing	register indirect
flags	none

JX5 (jump on X5)

If (X5)
(PC) = (byte 3) (byte 2)

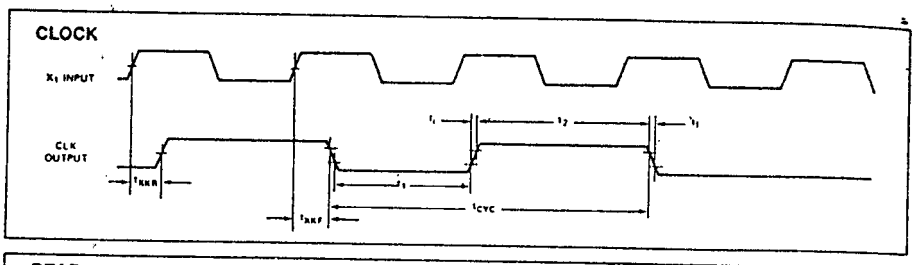
If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction, otherwise control continues sequentially.

1 1 1 1 1 1 0 1	(FD)
low-order address	
high-order address	

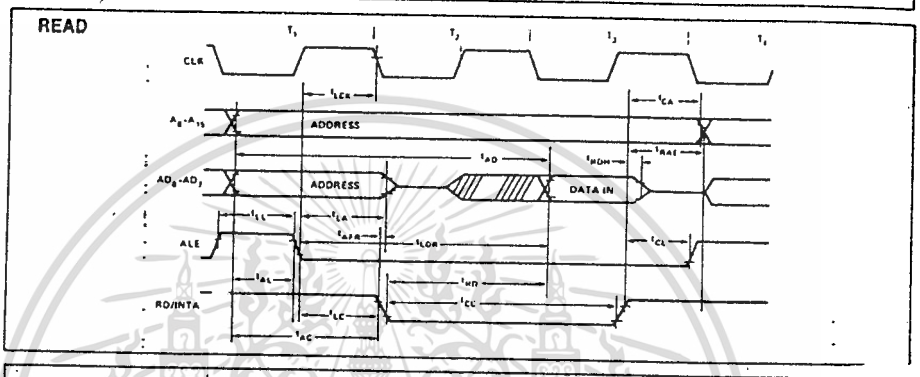
cycles	2 or 3
states	7 or 10
addressing	immediate
flags	none

WAVEFORMS

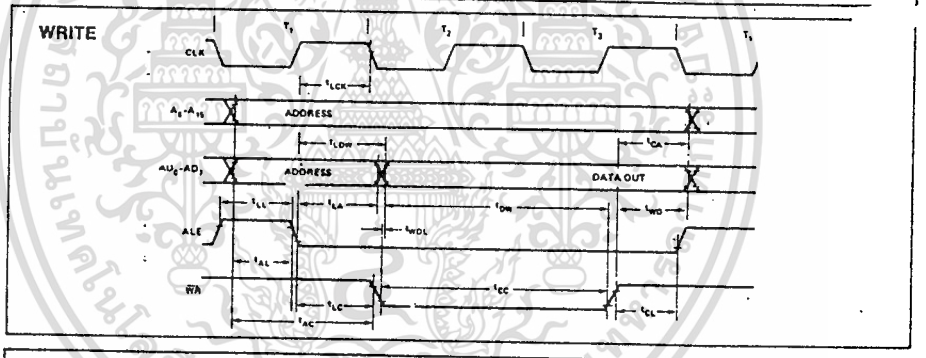
CLOCK



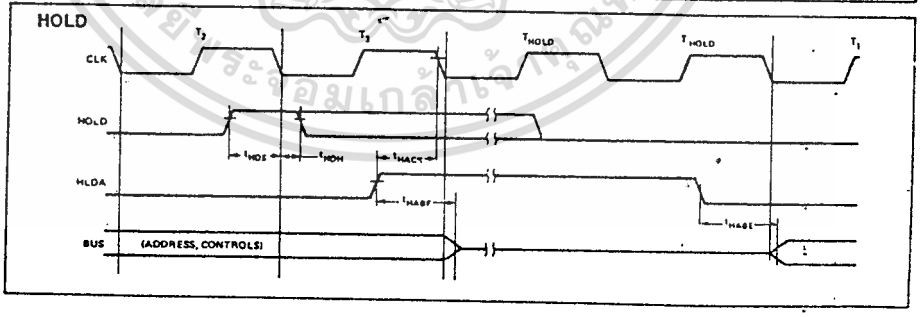
READ



WRITE



HOLD



8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- Reduces System Package Count
- Improved DC Driving Capability
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range
- 40 Pin DIP Package or 44 Lead PLCC
  (See Intel Packaging, Order Number: 231369)

The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

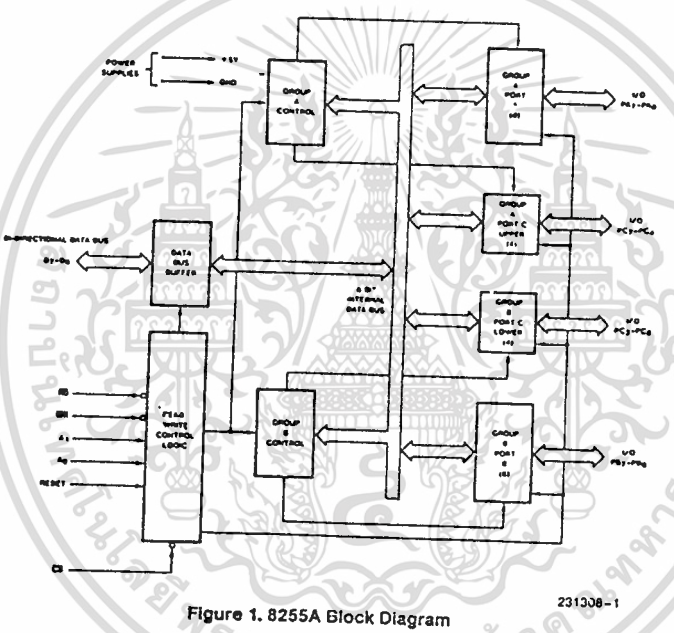


Figure 1. 8255A Block Diagram

231308-2

Figure 2. Pin Configuration

September 1987
Order Number: 231308-002

8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communication between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

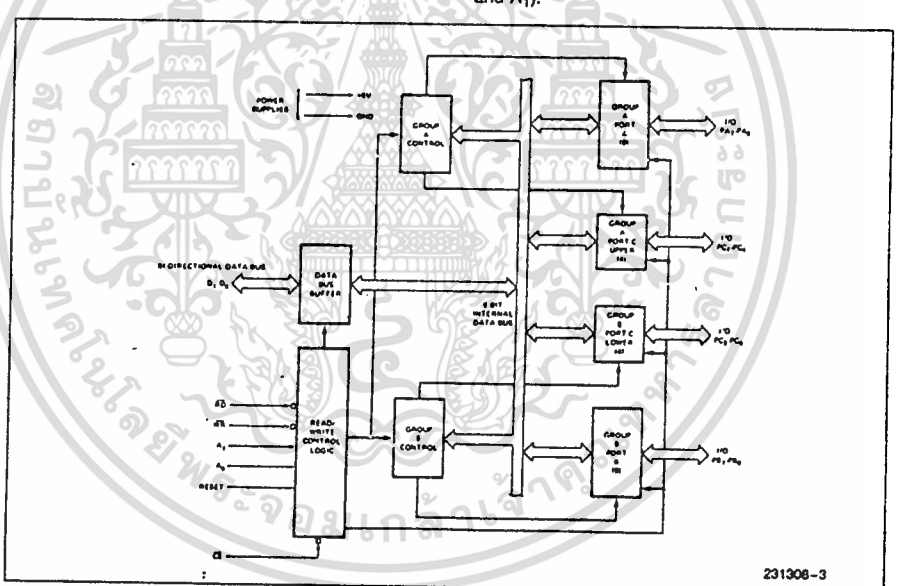


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

(RESET)

Reset. A "high" on this input clears the control register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A—Port A and Port C upper (C7–C4)
Control Group B—Port B and Port C lower (C3–C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

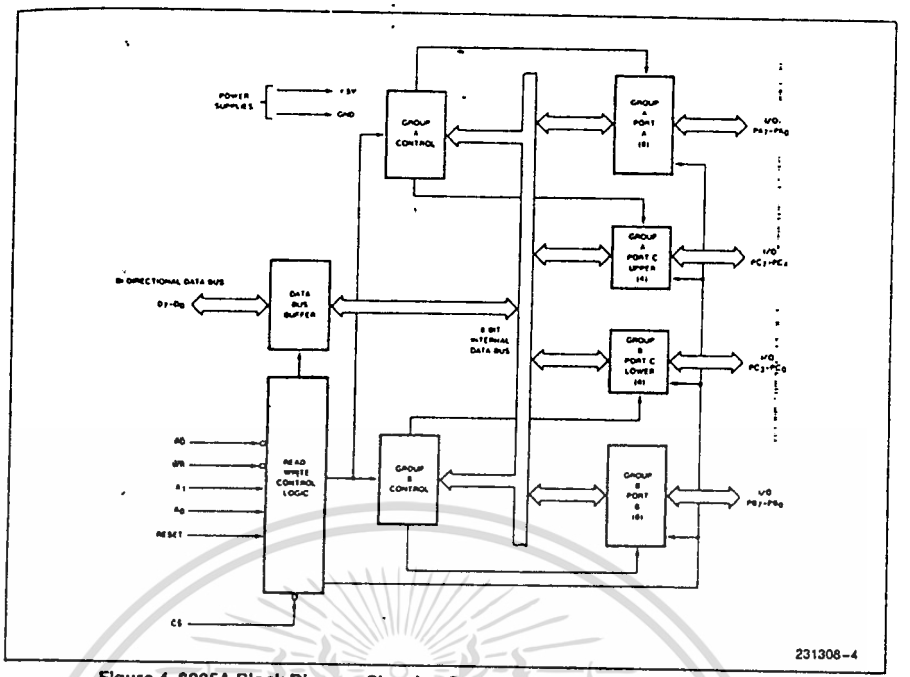
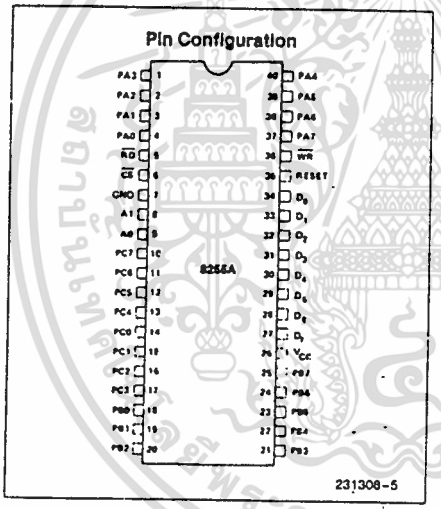


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions

Pin Configuration



Pin Names

D_7–D_0	Data Bus (Bi-Directional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0, A1	Port Address
PA7–PA0	Port A (BIT)
PB7–PB0	Port B (BIT)
PC7–PC0	Port C (BIT)
V_{CC}	+5 Volts
GND	0 Volts

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0—Basic Input/Output

Mode 1—Strobed Input/Output

Mode 2—Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

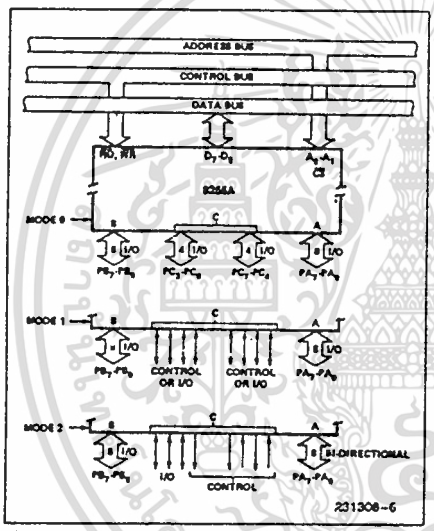


Figure 5. Basic Mode Definitions and Bus Interface

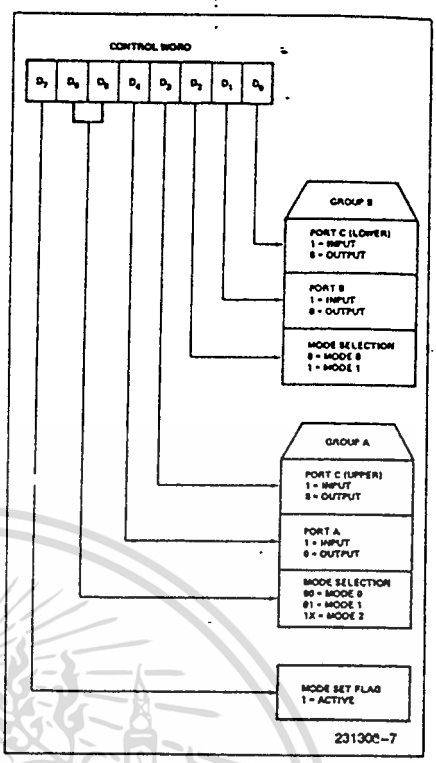


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

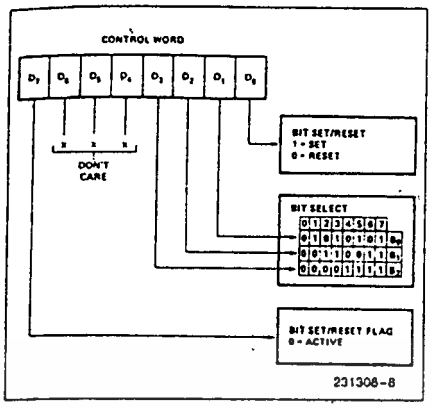


Figure 7. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is set—Interrupt enable

(BIT-RESET)—INTE is RESET—Interrupt disable

NOTE:
All Mask flip-flops are automatically reset during mode selection and device Reset.

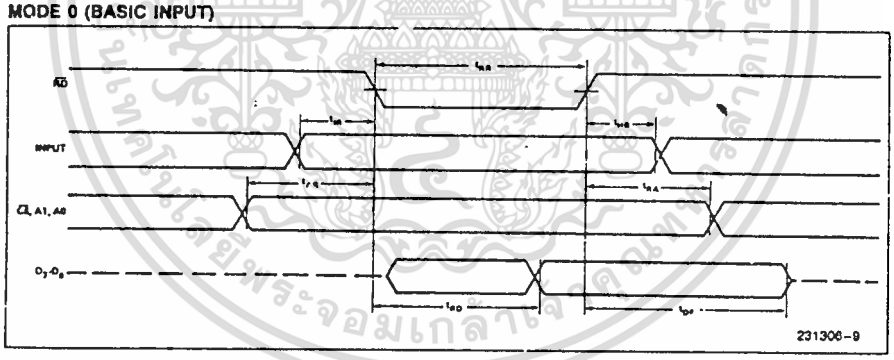
Operating Modes

MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

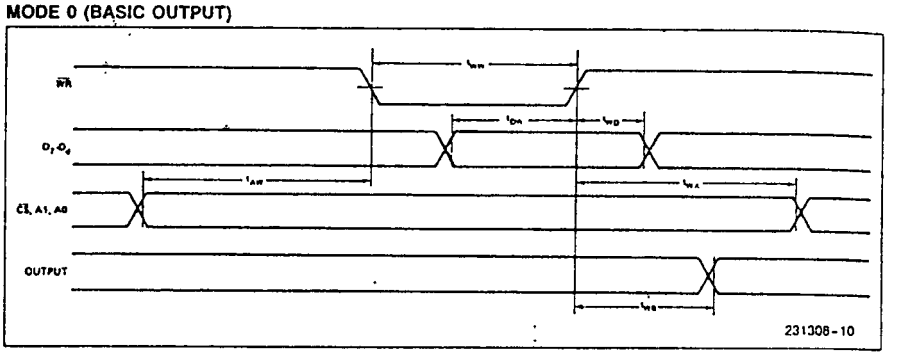
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



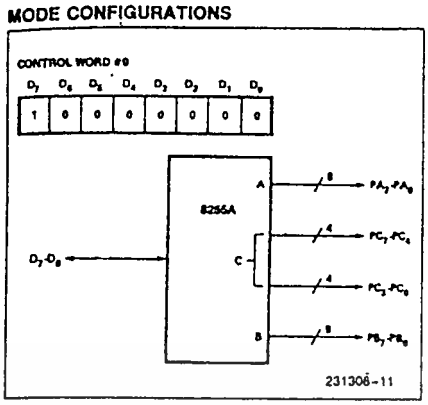
MODE 0 (BASIC OUTPUT)

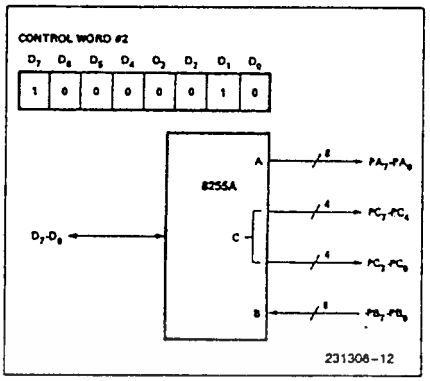


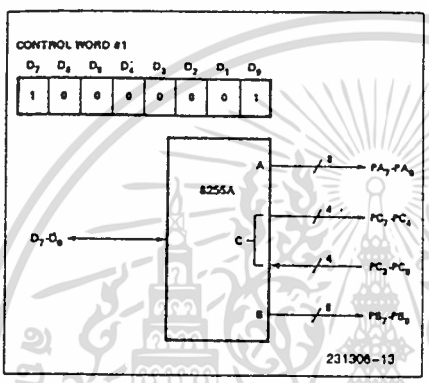
MODE 0 PORT DEFINITION

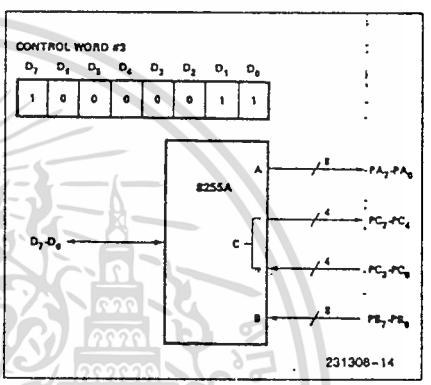
A		B		Group A			Group B	
D_4	D_3	D_1	D_0	Port A	Port C (Upper)	#	Port B	Port C (Lower)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

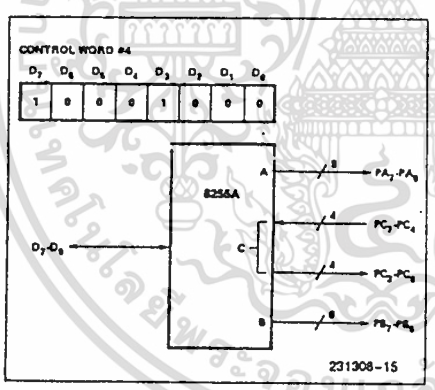
MODE CONFIGURATIONS

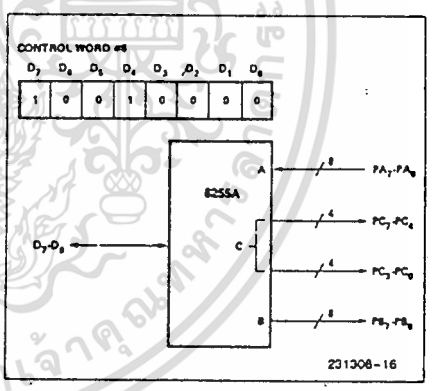


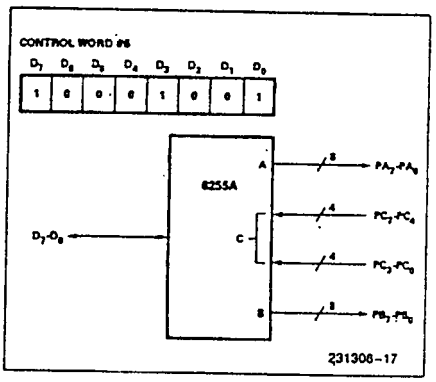


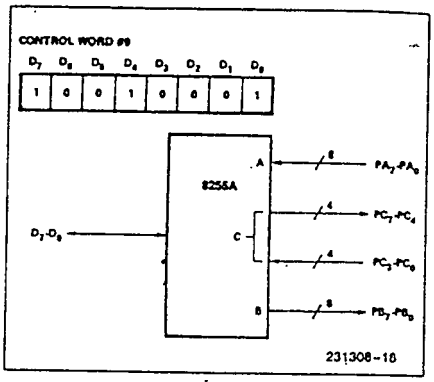


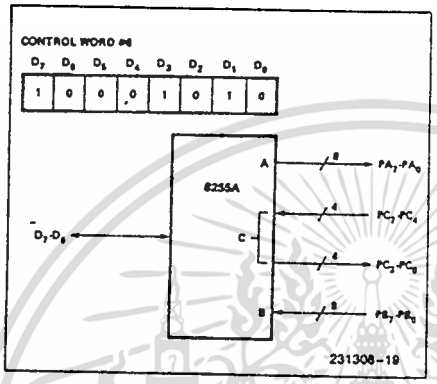


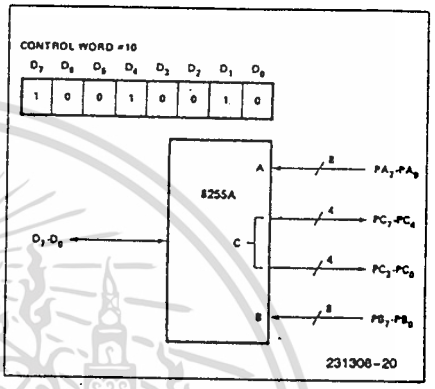


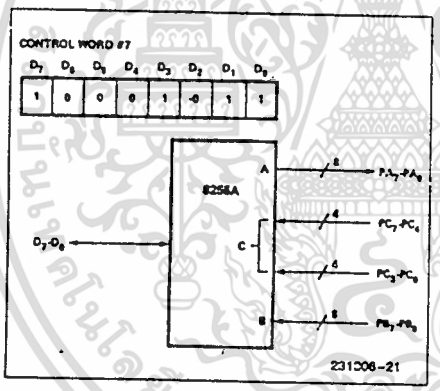


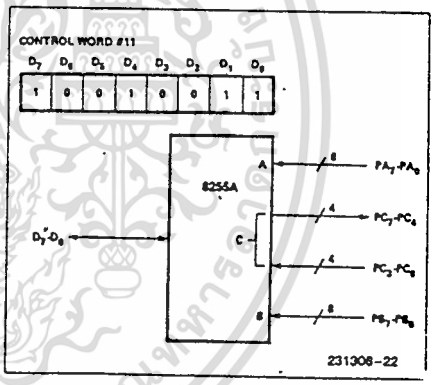


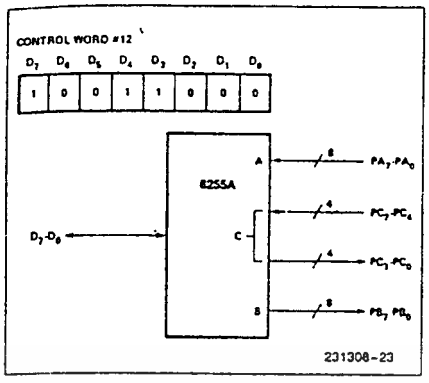


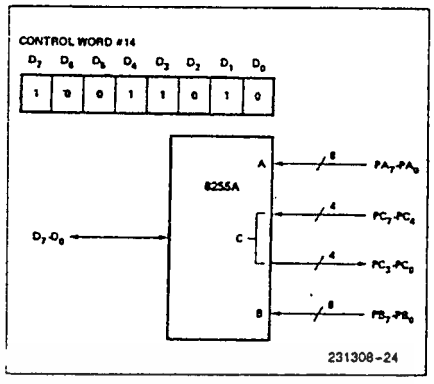


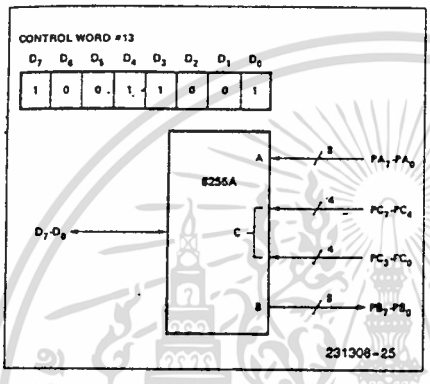


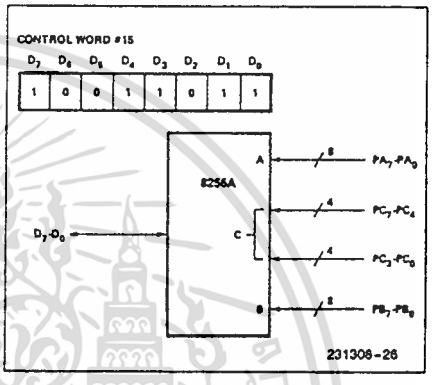












Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

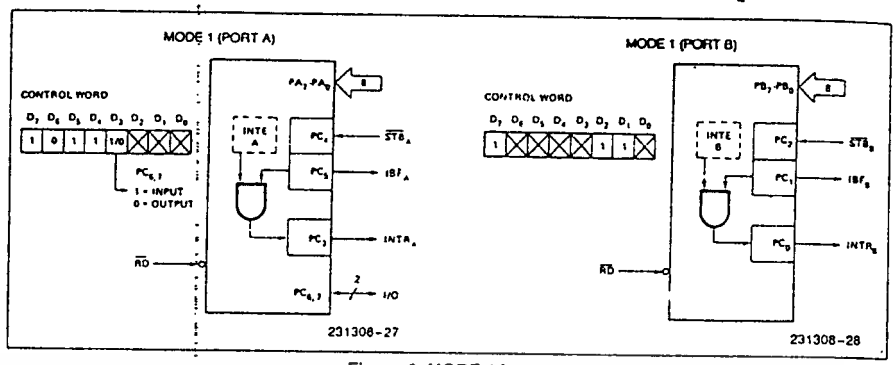


Figure 8. MODE 1 Input

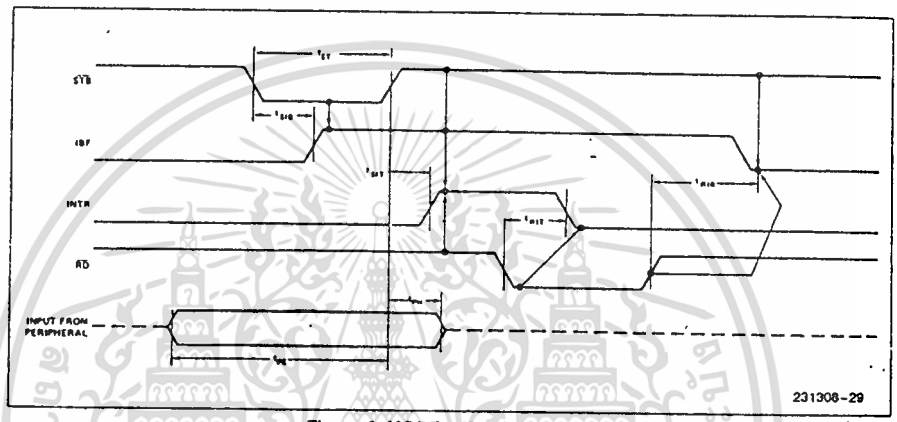


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one", and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

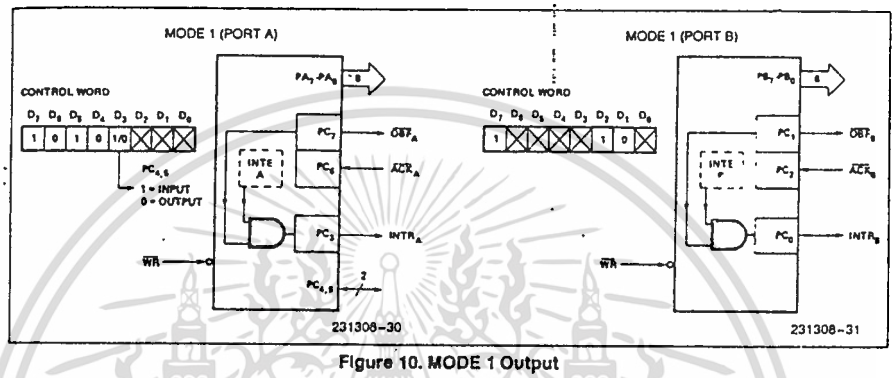


Figure 10. MODE 1 Output

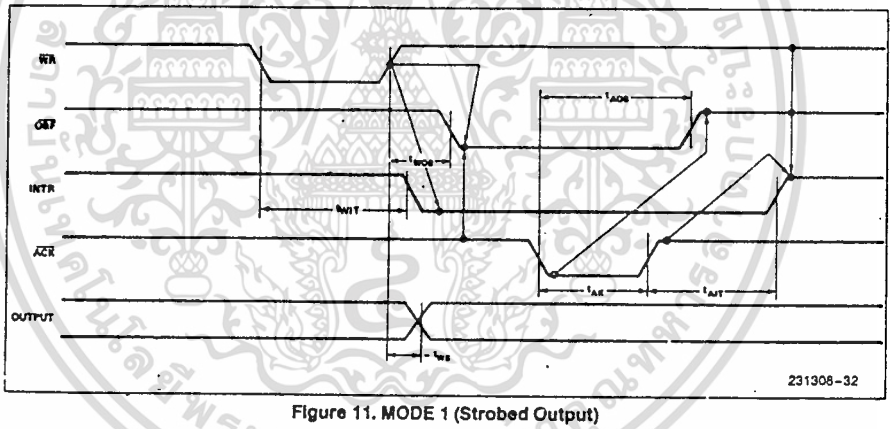


Figure 11. MODE 1 (Strobed Output)

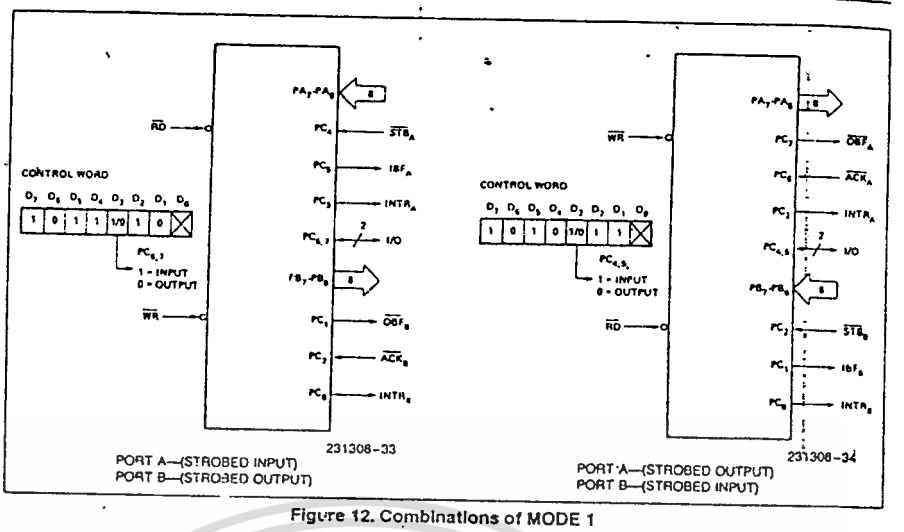


Figure 12. Combinations of MODE 1

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in MODE 1 to support a wide variety of strobed I/O applications.

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

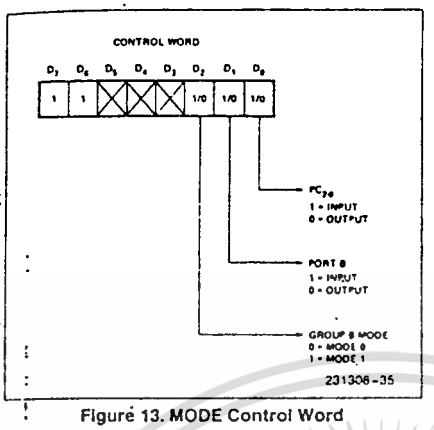


Figure 13. MODE Control Word

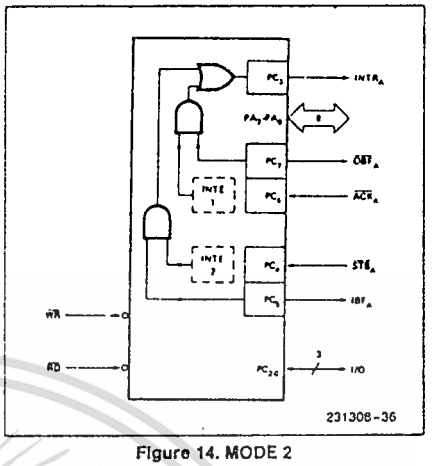
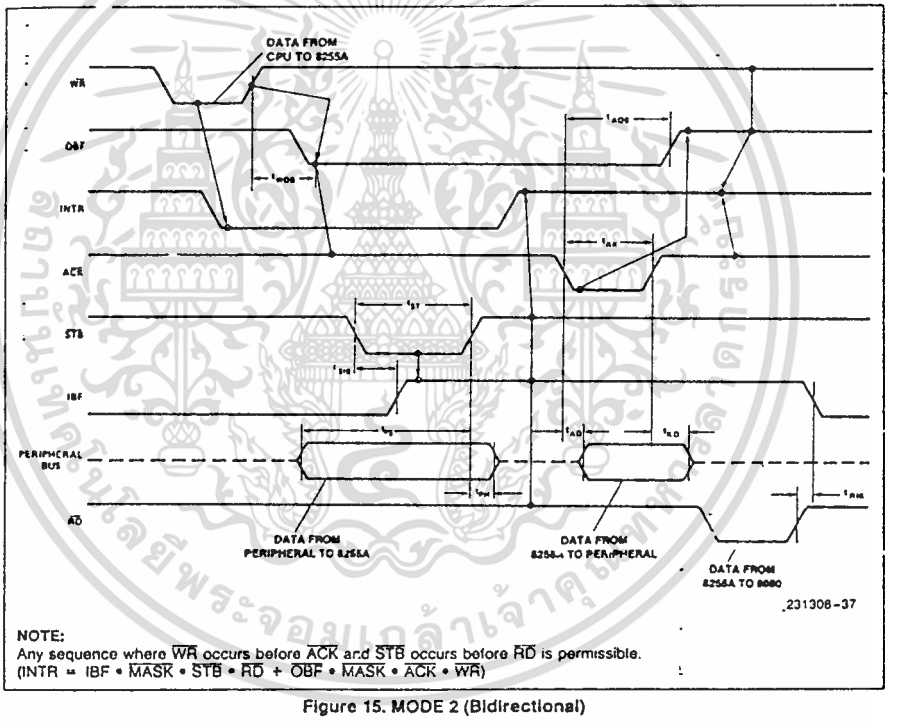


Figure 14. MODE 2



NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • MASK • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • MASK • $\overline{ACK}$ • $\overline{WR}$)

Figure 15. MODE 2 (Bidirectional)

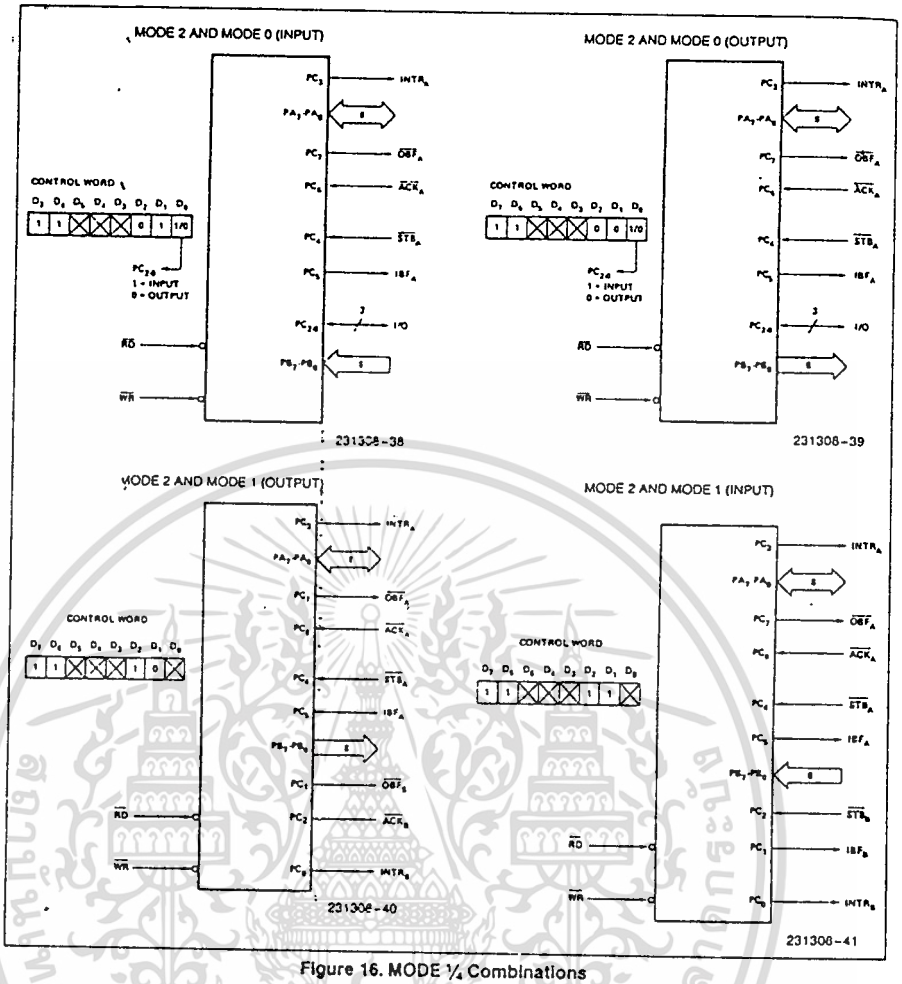


Figure 16. MODE 1/2 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	⟷	
PA_1	IN	OUT	IN	OUT	⟷	
PA_2	IN	OUT	IN	OUT	⟷	
PA_3	IN	OUT	IN	OUT	⟷	
PA_4	IN	OUT	IN	OUT	⟷	
PA_5	IN	OUT	IN	OUT	⟷	
PA_6	IN	OUT	IN	OUT	⟷	
PA_7	IN	OUT	IN	OUT	⟷	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs—

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs—

Bits in C upper (PC_7–PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3–PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of eight output buffers, selected randomly from Ports B and C can source 1 mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

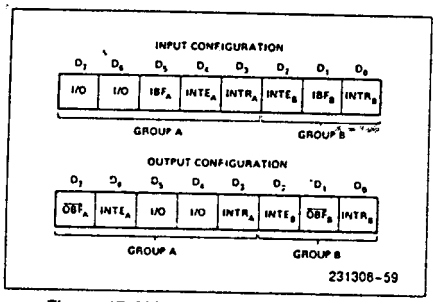


Figure 17. MODE 1 Status Word Format

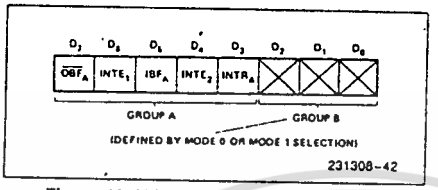


Figure 18. MODE 2 Status Word Format

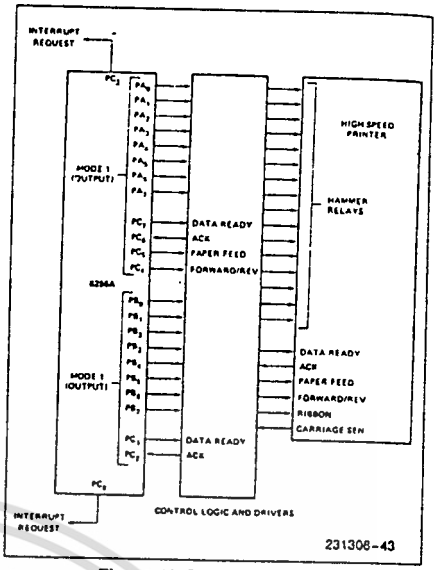


Figure 19. Printer Interface

APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 19 through 25 represent a few examples of typical applications of the 8255A.

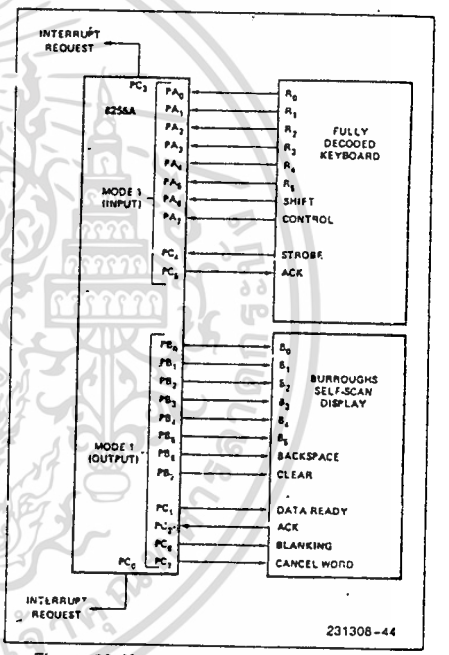


Figure 20. Keyboard and Display Interface

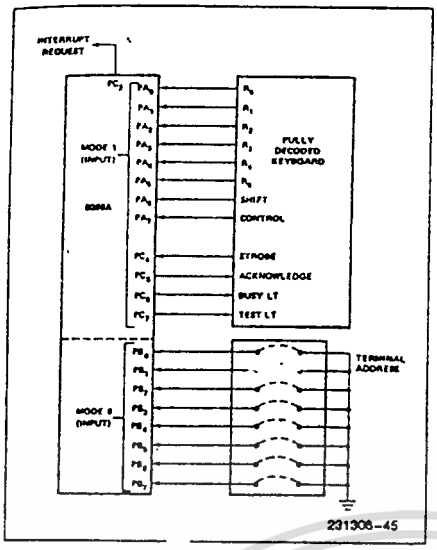


Figure 21. Keyboard and Terminal Address Interface

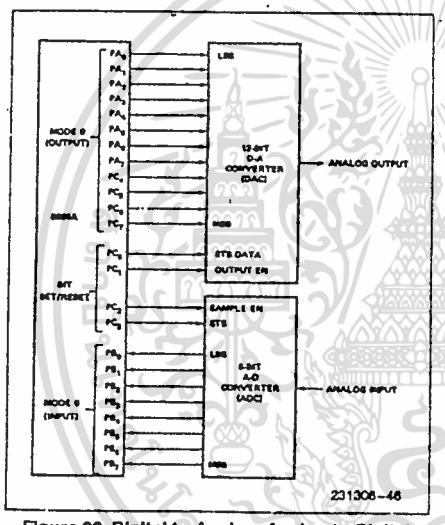


Figure 22. Digital to Analog, Analog to Digital

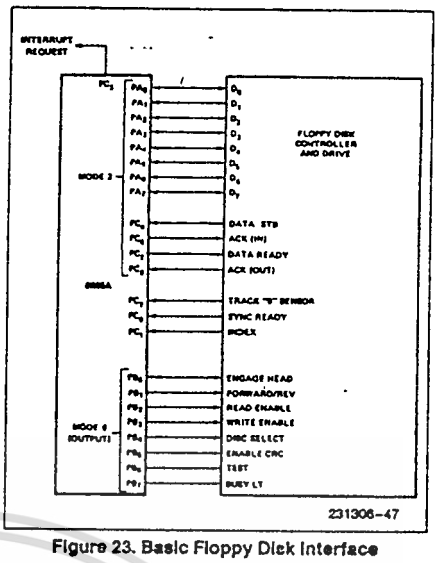


Figure 23. Basic Floppy Disk Interface

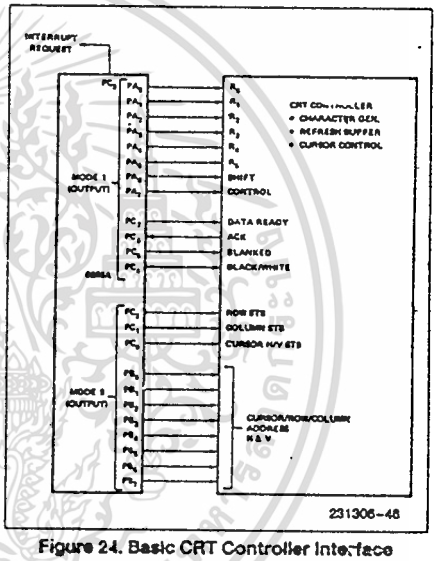


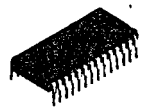
Figure 24. Basic CRT Controller Interface

256K-BIT (32,768 x 8) CMOS STATIC RANDOM ACCESS MEMORY WITH DATA RETENTION AND LOW POWER

The Fujitsu MB 84256 is a 32,768-word by 8-bit static random access memory fabricated with a CMOS silicon gate process. The memory utilizes asynchronouse circuitry and may be maintained in any state for an indefinite period of time. All pins are TTL compatible, and a single +5 volts power supply is required.

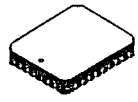
The MB 84256 is ideally suited for use in microprocesser systems and other applications where fast access time and ease of use are required. All devices offer the advantages of low power dissipation, low cost and high performance.

- Organization: 32,768 x 8 bits
- Fast access time: 100 ns max. (MB 84256-10/10L/10LL)
  120 ns max. (MB 84256-12/12L/12LL)
  150 ns max. (MB 84256-15/15L/15LL)
- Completely static operation: No clock required
- TTL compatible inputs/outputs
- Three-state outputs
- Single +5V power supply, ±10% tolerance
- Low power standby:
  CMOS level: 5.5 mW max. (MB 84256-10/12/15)
  0.55 mW max. (MB 84256-10L/10LL/12L/12LL/15L/15LL)
  TTL level: 16.5 mW max. (MB 84256-10/10L/10LL/12/12L/12LL/15/15L/15LL)
- Data retention: 2.0V
- Standard 28-pin DIP (600 mil) (Suffix: -P)
- Standard 28-pin Bend-type Plastic Flat Package (450 mil) (Suffix: -PF)
- Standard 32-pad LCC (Suffix: -CV)



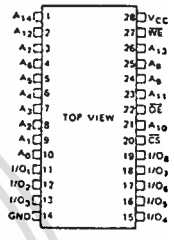
PLASTIC PACKAGE
DIP-28P-M02

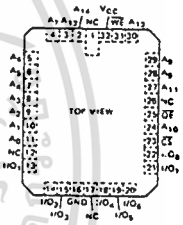
PLASTIC PACKAGE
FPT-28P-M02



CERAMIC PACKAGE
LCC-32C-A02

PIN ASSIGNMENT





ABSOLUTE MAXIMUM RATINGS (see NOTE)

Rating		Symbol	Value	Unit
Supply Voltage		V_{CC}	−0.5 to +7.0	V
Input Voltage		V_{IN}	−0.5 to V_{CC}+0.5	V
Output Voltage		V_{OUT}	-0.5 to V_{CC}+0.5	V
Temperature Under Bias		T_{BIAS}	−10 to +85	°C
Storage Temperature Range	CERAMIC	T_{STG}	−65 to +150	°C
	PLASTIC		−40 to +125	

intel®

27256
256K (32K x 8) PRODUCTION AND UV ERASABLE PROMS

- **New Quick-Pulse Programming™ Algorithm for Plastic P27256**
  - **— 4 Second Programming**
  - **— Int$_e$ligent Programming™ Algorithm Compatible**
- **Fast Access Time**
  - **— 170 ns D27256-1**
  - **— 200 ns P27256-2**
- **Int$_e$ligent Identifier™ Mode**
- **Plastic Production P27256 is Compatible with Auto-Insertion Equipment**
- **Moisture Resistant**
- **Industry Standard Pinout ... JEDEC Approved ... 28 Lead Cerdip and Plastic Package**

(See Packaging Spec, Order #231369)

The Intel 27256 is a 5V only, 262,144-bit Ultraviolet Erasable (Cerdip)/plastic production (P27256) electrically programmable read-only memory (EPROM). Organized as 32K words by 8 bits, individual bytes can be accessed in less than 170 ns (27256-1). This is compatible with high performance microprocessors, such as the Intel iAPX 186, allowing full speed operation without the addition of performance-degrading WAIT states. The 27256 is also directly compatible with Intel's 8051 family of microcontrollers.

The Plastic P27256 is ideal for high volume production environments where code flexibility is crucial. Plastic packaging is also well-suited to auto-insertion equipment in cost-effective automated assembly lines. Intel's new Quick-Pulse Programming Algorithm enables the P27256 to be programmed within four seconds (plus programmer overhead). Programming equipment which takes advantage of this innovation will electronically identify the EPROM with the help of the Int$_e$ligent Identifier and rapidly program it using a superior programming method. The Int$_e$ligent Programming Algorithm may be utilized in the absence of such equipment.

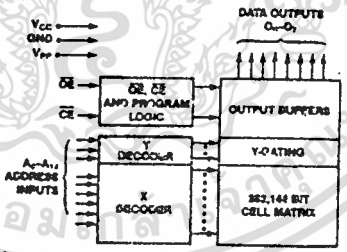
The 27256 enables implementation of new, advanced systems with firmware-intensive architectures. The combination of the 27256's high-density, cost-effective EPROM storage, and new advanced microprocessors having megabit addressing capability provides designers with opportunities to engineer user-friendly, high reliability, high-performance systems.

The 27256's large storage capability of 32 K-bytes enables it to function as a high-density software carrier. Entire operating systems, diagnostics, high-level language programs and specialized application software can reside in a 27256 EPROM directly on a system's memory bus. This permits immediate microprocessor access and execution of software and eliminates the need for time-consuming disk accesses and downloads.

Two-line control and JEDEC-approved, 28-pin packaging are standard features of all Intel high-density EPROMs. This assures easy microprocessor interfacing and minimum design efforts when upgrading, adding, or choosing between nonvolatile memory alternatives.

The 27256 is manufactured using Intel's advanced HMOS*II-E technology.

*HMOS is a patented process of Intel Corporation.



290097-1

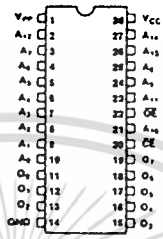
Figure 1. Block Diagram

Pin Names

A_0–A_{14}	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
O_0–O_7	Outputs
D.U.	Don't Use
$\overline{WE}$	Write Enable

27916	27513	27512	27128A	2764A 27C64	2732A	2716
V_{PP}	D.U.	A_{15}	V_{PP}	V_{PP}		
A_{12}	A_{12}	A_{12}	A_{12}	A_{12}		
A_7	A_7	A_7	A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0	A_0	A_0	A_0
O_0	D_0/O_0	O_0	O_0	O_0	O_0	O_0
O_1	D_1/O_1	O_1	O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2	O_2	O_2	O_2
Gnd	Gnd	Gnd	Gnd	Gnd	Gnd	Gnd

27256
P27256



290097-2

2716	2732A	2764A 27C64	27128A	27512	27513	27916
		V_{CC} PGM	V_{CC} PGM	V_{CC} A_{14}	V_{CC} $\overline{WE}$	V_{CC} PGM/$\overline{WE}$
V_{CC}	V_{CC}	N.C.	A_{13}	A_{13}	A_{13}	A_{13}
A_8	A_8	A_8	A_8	A_8	A_8	A_8
A_9	A_9	A_9	A_9	A_9	A_9	A_9
V_{PP}	A_{11}	A_{11}	A_{11}	A_{11}	A_{11}	A_{11}
$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}/V_{PP}$	$\overline{OE}$
A_{10}	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
O_7	O_7	O_7	O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3	O_3	O_3	O_3

NOTE:
Intel "Universal Site" Compatible EPROM pin configurations are shown in the blocks adjacent to the P27256 pins.

Figure 2. Cerdip/Plastic DIP Pin Configuration

ประวัติผู้เขียน

ชื่อ : นายเชวง วงษ์ฉิม

วันเดือนปีเกิด : วันที่ 15 มกราคม พ.ศ. 2515

ภูมิลำเนา : 189/3 ซ.ร่วมสันติ ถ.ประดิพัทธ์ ต.สามเสนใน อ.พญาไท จ. กรุงเทพมหานคร 10400

ประวัติการศึกษา : ม.3 จาก รร.วัดราชาธิวาส จ.กรุงเทพมหานคร พ.ศ.2527-2530

ปวช.สาขาช่างอิเล็กทรอนิกส์ จากสถาบันเทคโนโลยีราชมงคล วิทยาเขตนนทบุรี พ.ศ.2530-2533

ปวส.สาขาเทคโนโลยีอิเล็กทรอนิกส์จากมหาวิทยาลัยเอเชียอาคเนย์ พ.ศ. 2533-2535

ประวัติการทำงาน : บ.ฟิลลิปส์ เซมิคอนดัคเตอร์ (ไทยแลนด์) จำกัด ตั้งแต่ พ.ศ.2538

ชื่อ : นายบุญยิ่ง นบนอบ

วันเดือนปีเกิด : 14 ธันวาคม 2515

ภูมิลำเนา : เลขที่ 17 หมู่ 2 ต.คลองขนาน กิ่งอำเภอเหนือคลอง จ.กระบี่

ประวัติการศึกษา : ม.3 จาก รร.อำมาตย์พานิชนุกูล จ.กระบี่ พ.ศ. 2528-2531

ปวช.สาขาช่างอิเล็กทรอนิกส์ จากวิทยาลัยเทคนิคกระบี่ พ.ศ.2531-2534

ปวส.สาขาเทคโนโลยีอิเล็กทรอนิกส์ จากมหาวิทยาลัยเอเชียอาคเนย์ พ.ศ. 2534-2536

ชื่อ : นายบุญฤกษ์ กาญจนเทพ

วันเดือนปีเกิด : 4 เมษายน 2510

ภูมิลำเนา : 43 หมู่ 7 ต.อัยเยอร์เวง อ.เบตง จ.ยะลา

ประวัติการศึกษา : ม.3 จาก รร.วุฒิชัยวิทยา จ.ปัตตานี

ปวช. สาขาอิเล็กทรอนิกส์ จากวิทยาลัยเทคนิคยะลา จ.ยะลา

ปวส.สาขาเทคโนโลยีอิเล็กทรอนิกส์จากมหาวิทยาลัยเอเชียอาคเนย์ พ.ศ. 2533-2535

ประวัติการทำงาน : การสื่อสารแห่งประเทศไทย ตั้งแต่ 2536-ปัจจุบัน