เครื่องแสดงผลอักษรเบรลล์

BRAILLE INDICATOR

โดย

นายชัยยุทธ ชีชะนะ

นายศรัณย์ สาตารมณ์

นายอรรถพล โมลิโต

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์ (บ่าย) ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์ เครื่องแสดงผลอักษรเบรลล์
(BRAILLE INDICATOR)

โดย นายชัยยุทธ ชีชะนะ 40013289
นายศรัณย์ สาตารมณ์ 40013306
นายอรรถพล โมลิโต 40013318

ภาควิชา เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา อ.อรรถสิทธิ์ หล่าสกุล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... อ.ที่ปรึกษา
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

หัวข้อปริญญานิพนธ์	เครื่องแสดงผลอักษรเบรลล์		
นักศึกษา	นายชัยยุทธ	ชีชะนะ	40013289
	นายศรัณย์	สาตารมณ์	40013306
	นายอรรถพล	โมลิโต	40013318
อาจารย์ที่ปรึกษา	อ.อรรถสิทธิ์	หล่าสกุล	
ระดับการศึกษา	ปริญญาตรีอุตสาหกรรมศาสตรบัณฑิต		
ปีการศึกษา	2542		

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ ได้เสนอวิธีการออกแบบแผงแสดงผลอักษรเบรลล์รวมถึงวงจรขับแผงแสดงผลอักษรเบรลล์ โดยการอินเทอร์เฟสระหว่าง เมนบอร์ดไอบีเอ็มพีซีกับ การ์ดอีทีพีซี8255 และแผงแสดงผลอักษรเบรลล์ ตัวเครื่องประกอบด้วย 2 ฟังก์ชันซึ่งมีประโยชน์มากในการช่วยเหลือบุคคลซึ่งมีปัญหาทางด้านสายตา

PROJECT	BRAILLE INDICATOR		
NAME	MR.CHAIYUT	CHEECHANA	40013289
	MR. SRAN	SATAROM	40013306
	MR.ATTHAPON	MOLITO	40013318
ADVISOR	MR.ATTHASIT HLASKUN		
LEVEL OF STUDY	BACHALOR'S DEGREE IN INDUSTRIAL (ELECTRONIC TECHNOLOGY)		
ACADEMIC	YEAR 1999		

ABSTRACT

This project describes the method of design braille indicator and braille indicator driver circuit together with interfacing between mainboard of IBM/PC , ET-PC8255 CARD and braille indicator.

The machine consist of two functions that are very useful for helping people who have a problem with his sight.

กิตติกรรมประกาศ

โครงงานและปริญญานิพนธ์ สำเร็จลุล่วงไปได้ด้วยดี เกิดจากความร่วมมือร่วมใจของคณะผู้จัดทำ โดยได้รับความช่วยเหลือทางด้านคำแนะนำต่างๆ เป็นอย่างดีจากอาจารย์ที่ปรึกษา และอาจารย์อีกหลายท่านในภาคเทคนิคอุตสาหกรรม และเพื่อนๆ ภาคอื่น ที่ได้ให้ความช่วยเหลือทางด้านคำแนะนำหลายๆ ด้าน

สุดท้ายนี้ขอขอบคุณอาจารย์ทุกท่านที่ช่วยประสาทวิชาให้ความรู้ต่างๆ จนสามารถทำโครงงานนี้ได้

นายชัยยุทธ ชีชะนะ

นายศรัณย์ สาตารมณ์

นายอรรถพล โมลิโต

สารบัญ

สารบัญรูปภาพ

บทที่1
บทนำ

1.1 แนวความคิดและที่มาในการทำปริญญานิพนธ์

ระบบคอมพิวเตอร์ได้เข้ามามีส่วนเกี่ยวข้องกับชีวิตมนุษย์มากขึ้นทำให้ก่อให้เกิดการศึกษาในรูปแบบเรียนด้วยตนเองมากขึ้นด้วยคอมพิวเตอร์และอินเทอร์เนต การที่ต้องใช้งานคีย์บอร์ด และอ่านไฟล์ข้อมูล(Text File)เป็นสิ่งจำเป็นที่ใช้ในการติอต่อกับคอมพิวเตอร์สำหรับคนปกติ แต่เป็นอุปสรรคที่สำคัญสำหรับคนที่พิการทางสายตา ซึ่งในการแสวงหาความรู้ของคนที่พิการทางสายตานอกจากจะอาศัยการฟังอย่างคนปกติแล้วการอ่านหนังสือต้องอาศัยการอ่านอักษรเบรลล์ ซึ่งต่างจากคนปกติ

ดังนั้นเพื่อเป็นการบุกเบิกเพื่อให้คนพิการดังกล่าวได้มีโอกาสได้ใช้งานเครื่องคอมพิวเตอร์ได้อย่างไม่ลำบากจนเกินไป จึงได้มีโครงงานนี้เกิดขึ้นเพื่อเป็นเครื่องมือช่วยเหลือดังกล่าว โดยในขั้นต้นของเครื่องต้นแบบได้เน้นในสองลักษณะคือ

1. เป็นเครื่องใช้งานคู่กับ คีย์บอร์ด เพื่อให้ผู้ใช้สามารถพิมพ์ข้อความได้อย่างถูกต้อง
2. เป็นเครื่องที่ช่วยในการอ่านไฟล์อักษรทั่วไปจากแผ่น Floppy Disk ให้แสดงผลออกเป็นอักษรเบรลล์และตัวเครื่องจะเป็นลักษณะทำงานด้วยตนเอง (Stand alone)

จากที่อวัยวะของคนเรามีอยู่ 32 ประการ ซึ่งดวงตาเป็นสิ่งสำคัญที่สุดก็ว่าได้ เพราะว่าเมื่อเราขาดตาไปเราไม่สามารถมองอะไรได้อีกเลย โลกทั้งโลกมืด ไม่สามารถรับรู้สิ่งที่สวยงามได้อีก รวมทั้งไม่สามารถรู้ข่าวสารต่างๆด้วยตัวเองได้ และการที่คนพิการคนตาบอด ไม่สามารถรับรู้เกี่ยวกับข่าวสารข้อมูลจำพวกข้อความ(Text File) จึงมีแนวความคิดที่จะสร้างเครื่องช่วยคนตาบอดขึ้นมาเพื่อให้คนตาบอดฝึกอ่านข้อความจากเครื่องนี้โดยอัตโนมัติ คือ จะให้คนตาบอดสามารถอ่าน Text File จาก disk drive ต่างๆ เช่น Floppy Disk และ พิมพ์ข้อความหรือตัวอักษรจาก keyboard แล้วให้คนตาบอดไปจับที่เครื่องแสดงผลอักษรเบรลล์ (Braille indicator) เพื่อให้รู้ว่าข้อความหรือตัวอักษรที่เขาพิมพ์ถูกต้องตามที่ตนต้องการ โดยจะลองทำเป็นเครื่องต้นแบบก่อนเพื่อที่จะพัฒนาเครื่อง (HARDWARE) และ (SOFTWARE)นี้ต่อไป

นอกจากนี้ยังมีส่วนของ SOFTWARE ซึ่งเป็นส่วนของโปรแกรมที่ใช้ในการควบคุม HARDWARE (Braille indicator) ให้ได้ตามผู้ใช้ต้องการ ในที่นี้จะใช้ TURBO C ในการเขียนโปรแกรมที่ใช้ในการควบคุม HARDWARE (Braille indicator)

1.2 วัตถุประสงค์และจุดมุ่งหมายในการทำปริญญานิพนธ์

1) เพื่อสร้างเครื่องต้นแบบระบบช่วยการอ่านข่าวสารอัตโนมัติจากคนพิการสำหรับคนตาบอด

2) เพื่อจะทำให้คนพิการสำหรับคนตาบอดฝึกใช้ keyboard จากเครื่อง computer ได้ และพิมพ์ ข่าวสารได้ เพราะว่าจะมี hardware เช็คว่าคนตาบอด พิมพ์ข้อความหรือตัวอักษรถูกต้องตามที่เขาพิมพ์หรือเปล่า

3) แปลง/อ่าน IBM Text files ออกแสดงผลที่ Panel อักษรเบรลล์

4) เพื่อศึกษาถึงระบบ INPUT ,OUTPUT และระบบ BUS ของ COMPUTER และการ INTERFACE กับอุปกรณ์ภายนอก

5) เพื่อศึกษาถึง การเขียนโปรแกรม TURBO C

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ช่วยให้คนพิการสำหรับคนตาบอดฝึกใช้ keyboard จากเครื่อง computer ได้ และ พิมพ์ข่าวสารได้ เพราะว่าจะมี hardware เช็คว่าคนตาบอด พิมพ์ข้อความหรือตัวอักษรถูกต้องตามที่เขาพิมพ์หรือเปล่า

2) ช่วยให้คนพิการสำหรับคนตาบอดสามารถรับรู้ข่าวสารจำพวก Text File ได้โดยแปลง/หอ่าน IBM Text files ออกแสดงผลที่ Panel อักษรเบรลล์

1.4 เนื้อหาของปริญญานิพนธ์

บทที่1 กล่าวถึง แนวความคิดและที่มา วัตถุประสงค์ และประโยชน์

บทที่2 กล่าวถึงทฤษฎีที่จำเป็นในโครงงานนี้ เช่น ระบบคอมพิวเตอร์พื้นฐาน, หลักการเบื้องต้นระบบBUS INPUT และ OUTPUT , พอร์ตอินพุต/เอาต์พุต

บทที่3 กล่าวถึงหลักการทำงานของ HARDWARE (BRAILLE INDICATOR), ET-PC8255 CARD และการทดสอบวงจรขับ Braille Indicator

บทที่4 กล่าวถึงการทดสอบการทำงาน Hardware ร่วมกับ Software

บทที่5 กล่าวถึงบทสรุปและวิจารณ์

บทที่2
ทฤษฎี

2.1 ระบบคอมพิวเตอร์พื้นฐาน (Basic computer system)

มีบล็อกไดอะแกรมการทำงาน ดังรูปที่2.1 หัวใจหลักคือ หน่วยประมวลผลกลาง หรือ CPU (Central processing unit) จะทำหน้าที่ประมวลผลข้อมูลต่างๆ เพื่อนำผลลัพธ์ที่ได้ไปทำการควบคุมระบบต่อไป ภายใน CPU ประกอบด้วย

2.1.1 หน่วยจัดการทางคณิตศาสร์ (Arithmatic Logic Unit :ALU) ทำหน้าที่จัดการด้านการคำนวนทางคณิตศาสร์และลอจิก แล้วนำผลลัพธ์หรือข้อมูลที่ได้ส่งให้แก่ CPU เพื่อประมวลผลและควบคุมการทำงานต่อไป

2.1.2 หน่วยเก็บข้อมูลชั่วคราว หรือ รีจีสเตอร์ (register) ทำหน้าที่เก็บข้อมูล เพื่อเตรียมเข้าสู่กระบวนการประมวลผล, เก็บข้อมูลที่ได้จากการประมวลผลโดย ALU ,เก็บข้อมูลเพื่อเตรียมส่งออกไปยังอุปกรณ์ภายนอก

2.1.3 แฟลก (Flag) ทำหน้าที่แสดงสภาวะการทำงานของ CPU ว่าขณะนี้เกิดอะไรขึ้นมีสภาวะลอจิกเป็นอย่างไร ซึ่งสามารถใช้ประโยชน์จากแฟลกนี้ในการตั้งเงื่อนไขในการทำงานของระบบคอมพิวเตอร์

2.1.4 สแต็กและโปรแกรมเคาน์เตอร์ (stack & program counter) สแตกทำหน้าที่เป็นที่พักข้อมูล เมื่อ CPU กระโดดไปทำโปรแกรมย่อย ก็จะเก็บข้อมูลที่ประมวลผลค้างไว้มาไว้ที่สแตก ส่วนโปรแกรมเคาน์เตอร์เป็นรีจีสเตอร์เก็บค่าการทำงานของ CPU ว่าขณะนี้ CPU ทำงานไปถึงแอดเดรสใด หรือเป็นเสมือนมอนิเตอร์แสดงตำแหน่งการทำงานของ CPU นั่นเอง

รูปที่2.1 ระบบคอมพิวเตอร์เบื้องต้น

ส่วนที่สองคือ หน่วยความจำ ซึ่งก็มีทั้งแบบรอม (Read Only Memory :ROM) และแรม (Random Access Memory :RAM) โดยรอมจะทำหน้าที่เก็บโปรแกรมควบคุมการทำงานของระบบและควบคุม Hardware ทั้งหมดหรือที่เรียกว่า โปรแกรมมอนิเตอร์ (monitor program) และหน่วยความจำแรมไว้เป็นที่พักข้อมูลระหว่างรอการประมวลผล หรือเก็บข้อมูลเพื่อเตรียมส่งออกไปยังอุปกรณ์ภายนอก

ส่วนพอร์ตอินพุตจะใช้เชื่อมต่อกับอุปกรณ์อินพุตที่จะส่งข้อมูลมายังคอมพิวเตอร์อันได้แก่ คีย์บอร์ด, ดิสก์ไดร์ฟ เป็นต้น

ส่วนพอร์ตเอาต์พุตจะใช้เชื่อมต่อกับอุปกรณ์เอาต์พุตเพื่อแสดงผลการทำงานได้แก่จอแสดงผล ,พริ้นเตอร์,พล็อตเตอร์ เป็นต้น

นอกเหนือไปจากนั้น คอมพิวเตอร์ยังมีพอร์ตสำหรับการสื่อสารข้อมูลอีกด้วย ที่นิยมใช้กันมากคือ พอร์ตสื่อสารอนุกรม RS-232C,RS422 และอื่นๆ

2.2 หลักการเบื้องต้นของพอร์ท อินพุต/เอาต์พุต

ระบบคอมพิวเตอร์จะมีระบบ Bus อินพุต/เอาต์พุต ที่ใช้สำหรับ Interface กับอุปกรณ์ภายนอกอยู่ 2 แบบคือ PCI BUS, ISA BUS ที่ใช้สำหรับขยายระบบเพื่อรับและส่งข้อมูล

เนื่องจาก ET-PC8255 CARD สำหรับ 16 Bit ISA Slot for PC ตัวนี้ ประกอบด้วย IC # 8255 เป็นหลัก ดังนั้นเรามาเรียนรู้เกี่ยวกับ IC # 8255 ซึ่งเป็น พอร์ตอินพุต/เอาต์พุต กันก่อน

8255

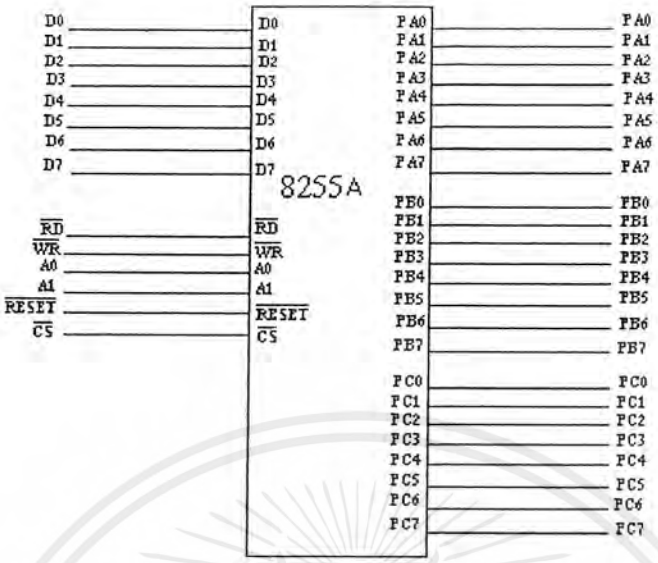
เป็นอุปกรณ์เฟอริเฟอรอลอินเตอร์เฟสที่สามารถโปรแกรมได้

เฟอริเฟอรอลอินเตอร์เฟสที่สามารถโปรแกรมได้นั้นมี ไอซีเบอร์ 8255A ของอินเทลและ Z80 PIO ของบริษัทไซลอก สำหรับ 8255A นั้น สามารถกำหนดพอร์ตอินพุต/เอาท์พุต ได้ทั้งหมด 3 พอร์ต (24บิต) และสามารถซิงโครนัสกับอุปกรณ์ภายนอกได้

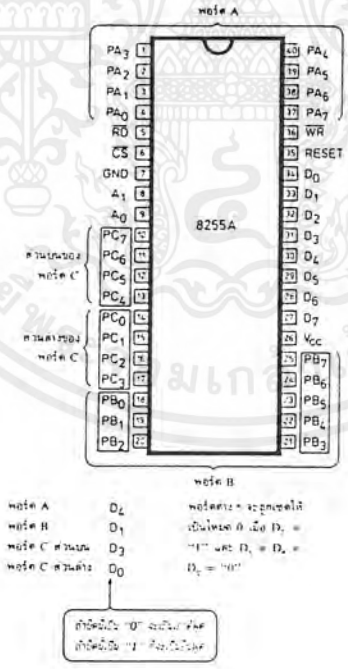
สำหรับ Z80 PIO สามารถกำหนดพอร์ตอินพุต/เอาต์พุต ได้ทั้งหมด 2 พอร์ต(16 บิต) ถึงแม้ว่าจะพอร์ตได้น้อยกว่า 8255A แต่ถ้ามีการอินเตอร์รัพต์ละก็ Z80 PIO จะใช้งานได้ดีกว่าเพราะที่โหมด 2 ของ Z80 PIO จะมีอินเตอร์รัพต์เวกเตอร์ ดังนั้นการจะเลือกใช้ตัวไหนก็ขึ้นอยู่กับการใช้งาน

8255A

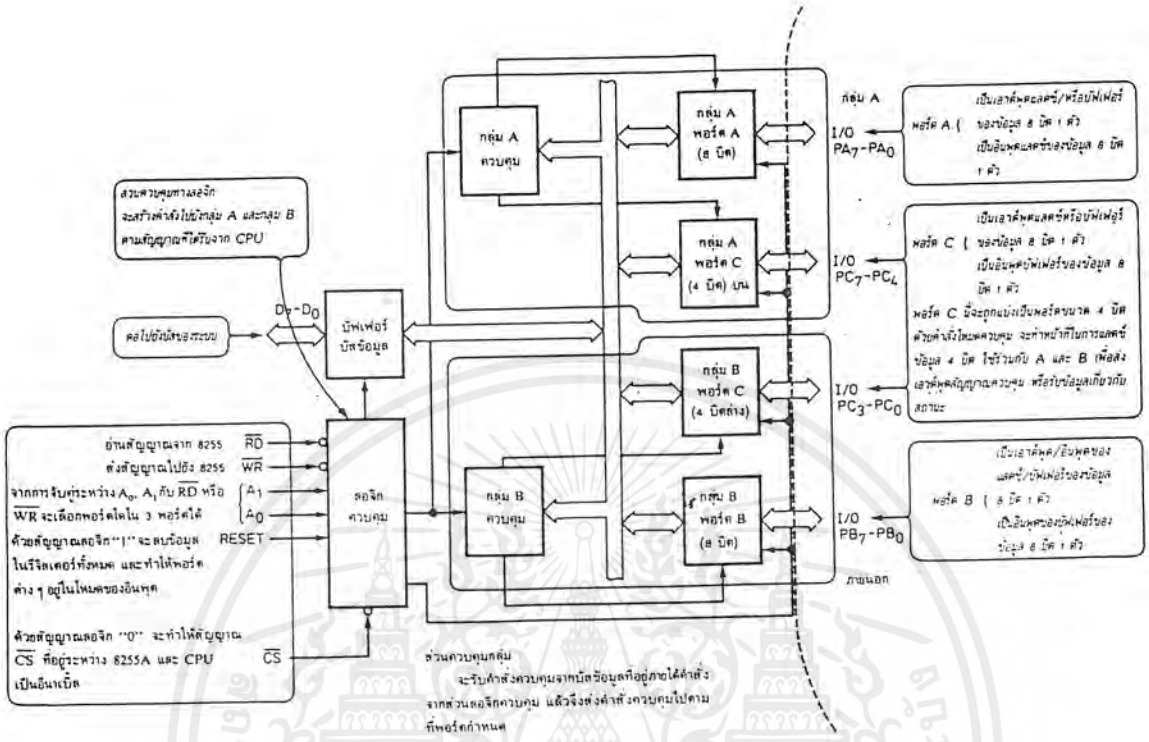
8255A เป็น I/O แบบขนาน ที่สามารถโปรแกรมได้ สามารถใช้งานได้หลายอย่างตามโปรแกรมที่ป้อนไว้ดังรูปที่2.2



รูปที่2.2 ลักษณะของ 8255A



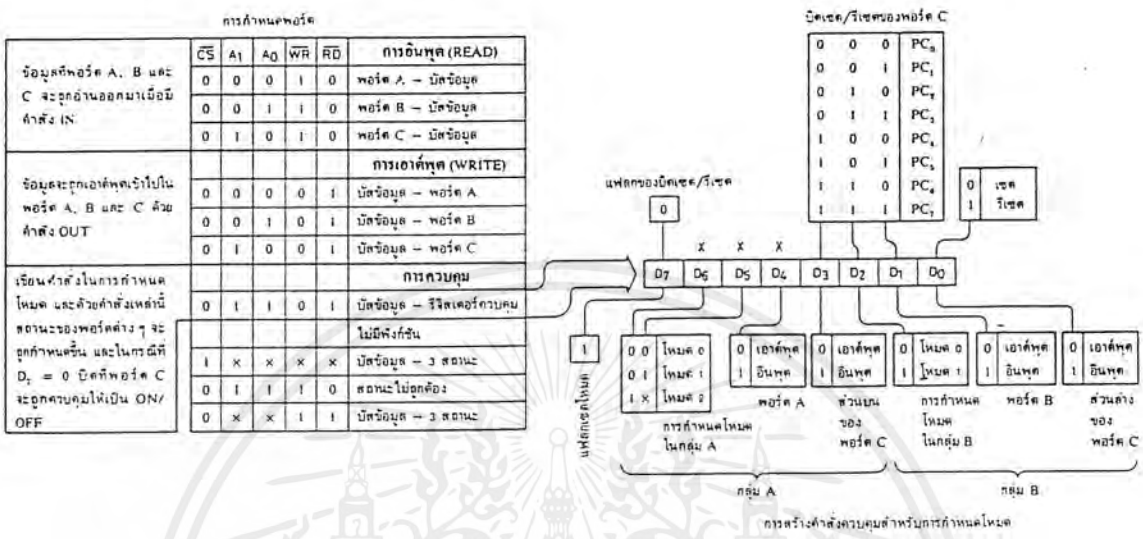
รูปที่2.3 แสดงตำแหน่งการวางขาของ 8255A



รูปที่2.4 แสดงการทำงานของส่วนต่างๆ ของ 8255A

โครงสร้างภายในของ 8255A ได้แสดงไว้ในรูปที่2.4 และโครงสร้างภายนอกได้แสดงไว้ในรูปที่ 2.3 พอร์ตI/O จะมี 3 พอร์ต คือ พอร์ต A,B และ Cและจัดแบ่งเป็น 2 กลุ่ม ดังแสดงในรูปที่ 2.4 โดยที่พอร์ต Aและ พอร์ต Bจะทำงานแตกต่างกันและไม่ขึ้นต่อกัน ส่วนพอร์ต C นั้น จะแบ่งสัญญาณเป็น 2 ส่วน ส่วนละ 4บิต ทำหน้าที่เป็นสัญญาณควบคุมให้กับพอร์ต A และ พอร์ต B

การทำงานอย่างง่ายที่สุดก็คือในโหมด 0 จะทำหน้าที่เป็นพอร์ตอินพุตและเอาต์พุตอย่างเดียว(รูปที่ 2.4 และรูปที่ 2.3) เมื่อมีข้อมูลเข้ามาที่พอร์ต A, B และ C ข้อมูลจะถูกอ่านไปที่ CPU ในกรณีการส่งข้อมูลข้อมูลที่อ่านไปที่พอร์ตต่าง ๆเมื่อถูกส่งไปแล้วก็จะถูกแลตช์ไว้เพื่อทำการส่งซ้ำอีก



รูปที่2.5 การกำหนดคำสั่งควบคุม (control word) ใน 8255A

<u>พอร์ตอินพุต/เอาต์พุต</u>

อุปกรณ์ภายนอกที่ต้องการส่งถ่ายข้อมูลกับซีพียู จะต้องใช้พอร์ตเป็นตัวกลางของการสั่งถ่ายข้อมูลเหล่านั้น โดยที่พอร์ตจะทำหน้าที่เป็นช่องทางผ่านของข้อมูลที่จะเข้ามายังซีพียู หรือข้อมูลที่จะออกไปจากซีพียูเพื่อส่งต่อไปยังอุปกรณ์ต่างๆ ซึ่งลักษณะของพอร์ตจะมี 2 ชนิด คือ พอร์ตอินพุต และ พอร์ตเอาต์พุต

พอร์ตอินพุต หมายถึง ช่องทางสำหรับนำข้อมูลจากอุปกรณ์ภายนอกเข้ามายัง ซีพียู

พอร์ตเอาต์พุต หมายถึง ช่องทางสำหรับนำข้อมูลออกจากซีพียู เพื่อออกไปสู่อุปกรณ์ภายนอก

ส่วนการส่งถ่ายข้อมูลระหว่างพอร์ตกับอุปกรณ์ภายนอก อาจเป็นแบบขนาน หรือแบบอนุกรมก็แล้วแต่ชนิดของอุปกรณ์ภายนอกที่ต้องการจะเชื่อมต่อด้วย

ซึ่งในเบื้องต้น จะกล่าวถึงพอร์ตที่เป็นแบบขนาน ซึ่งเป็นลักษณะของพอร์ตอินพุต และพอร์ตเอาต์พุตแบบขนานที่ต่ออยู่กับซีพียูที่ใช้เป็นพอร์ตอินพุตพื้นฐานคือ วงจรลอจิก 3 สถานะ

การทำงานของพอร์ตอินพุต คือเมื่อพอร์ตไม่ถูกเลือกให้ทำงาน (สัญญาณการเลือกพอร์ตเป็นระดับ 1) จะทำให้สภาวะของลอจิกแบบ 3 สถานะนี้ เป็นอิมพีแดนซ์สูง (high impedance) ซึ่งเหมือนกับปลดสายออกจากบัสข้อมูล แต่ถ้าสัญญาณการเลือกพอร์ตเป็นระดับ 0 ซึ่งหมายถึงพอร์ตถูกเลือกให้ทำงาน เมื่อเป็นเช่นนั้น จะทำให้ข้อมูลที่อินพุตของพอร์ตอินพุตต่อเข้ากับบัสข้อมูลของระบบ และซีพียูจะอ่านข้อมูลจากบัสข้อมูลนี้เข้าไปในตัวซีพียู เพื่อเข้าใช้งานต่อไป ส่วนอุปกรณ์ที่ใช้เป็นพอร์ตเอาต์พุตพื้นฐานคือ ฟลิปฟลอบแบบ D ซึ่งลักษณะของพอร์ตเอาต์พุตจะมีการทำงานดังนี้ คือ เมื่อพอร์ตเอาต์พุตนี้ถูกเลือก สัญญาณการเลือกพอร์ตเป็นระดับ 0 จะทำให้ข้อมูลบนบัสข้อมูลค้างสถานะไว้ที่เอาต์พุตของฟลิปฟลอป ซึ่งข้อมูลนี้เป็นข้อมูลที่ส่งไปยังอุปกรณ์ภายนอกเพื่อใช้งานต่อไป แต่ถ้าพอร์ตเอาต์พุตไม่ถูกเลือก สัญญาณที่เอาต์พุตจะคงค้างสถานะ คือไม่มีการเปลี่ยนแปลงแต่อย่างใด

เมื่อเป็น INPUT จะรับข้อมูลจากอุปกรณ์ภายนอกโดยตัวมันจะต่อกับ DATA BUS ของ CPU และจะเป็น TRI-STATE คือ จะไม่ส่งสัญญาณหรือข้อมูลใด ๆ เข้าไปยัง DATA BUS จนกว่า CPU มีความต้องการจะติดต่อด้วย

การต่อ PORT สัญญาณที่สำคัญมี

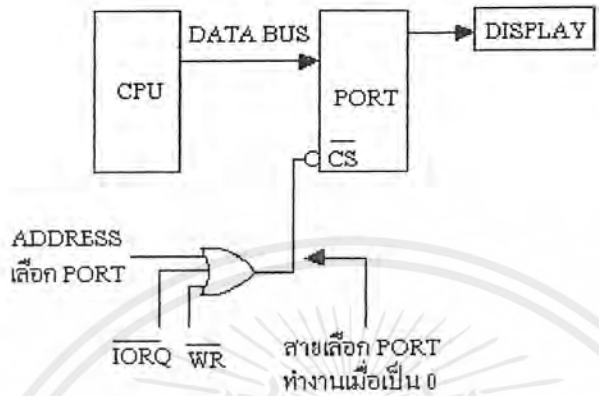
DATA BUS

ADDRESS BUS

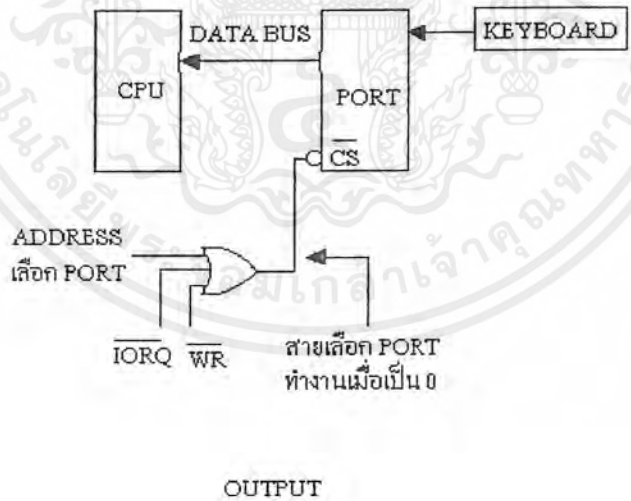
CONTROL BUS

- DATA BUS จะเป็นการติดต่อข้อมูลระหว่าง CPU กับอุปกรณ์ภายนอก
- ADDRESS BUS จะใช้กำหนดตำแหน่งของหน่วยความ หรือตำแหน่งพอร์ต I/O
- CONTROL BUS ใช้ในการแยกว่าตอนนี้ทำการติดต่อกับ PORT หรือ MEMORY จะให้อ่านหรือเขียน PORT นั่นก็คือ IORQ, RD และ WR

รูปแสดงการต่อ PORT I/O พื้นฐาน



รูปที่2.6 ลักษณะการต่อพอร์ตเอาท์พุท



รูปที่2.7 ลักษณะการต่อพอร์ตอินพุต

ใน PROJECT นี้ เราจะใช้ ET-PC8255 CARD ที่ใช้กับ 16 Bit ISA Slot for PC ซึ่งเบอร์PORT จะสามารถ DECODE ได้สะดวกโดย DECODE ที่ DIF SWITCH ของ PC-8255 CARD เพื่อหาที่ว่างของเบอร์ PORT ที่ไม่ได้ใช้งาน ก็จะได้ PORT INPUT และ PORT OUTPUT ตามต้องการ สามารถส่งข้อมูลออกหรือรับข้อมูลได้เลย

2.3 สัญญาณต่างๆ บนสล็อตของ IBM/PC

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเตอร์เฟสเข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล็อตบนเมนบอร์ดนี้จะมีจำนวน 5 สล็อต (สำหรับ IBM PC/XT จะมี 8 สล็อต) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขาซึ่งแบ่งออกเป็น 2 ข้างๆละ 31 ขา ส่วนการเรียกตำแหน่งของขาสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อต โดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา เช่น ขา B16 ก็คือขาทางด้านซ้ายของสล็อต ขาที่ 16 (นับจากทางด้านท้ายของเครื่อง) ส่วนขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของสล็อตขาที่ 24 (นับจากทางด้านท้ายของเครื่อง)

แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่างๆ บนเมนบอร์ด ทำให้การสร้างวงจรอินเทอร์เฟสกับ IBM/PC สามารถทำได้สะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอสเดรส (Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมการอ่าน/เขียนข้อมูลจากหน่วยความจำ หรืออพอร์ตอินพุท/เอาท์พุท, เส้นสัญญาณสำหรับการขออินเตอร์รัพท์ของวงจรอินเตอร์เฟส, เส้นสัญญาณสำหรับการขอ DMA,สัญญาณฐานเวลา (Timing Signal) ต่างๆที่ใช้ในระบบ, เส้นสัญญาณแสดงการรีเฟสหน่วยความจำและสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O Check)

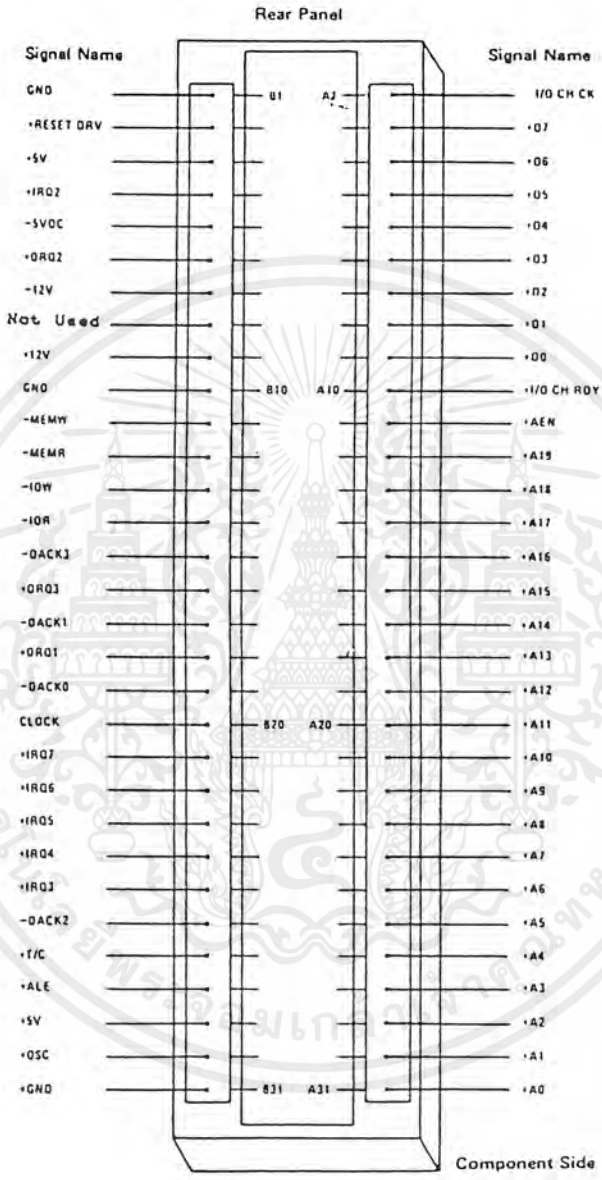
นอกจากเส้นสัญญานเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆ ที่ใช้ในระบบอีกด้วย คือ +5 Vdc, -5 Vdc, +12 Vdc และ -12 Vdc

2.4 รายระเอียดเกี่ยวกับสัญญาณต่างๆที่สำคัญที่จำเป็นในการอินเตอร์เฟสกับ ET-PC8255 CARD

OSC (Oscillator ; ขา B30)

ขานี้เป็นเอาท์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีค่าความถี่สูงสุดบนเมนบอร์ด คือ 14.31818 MHz ซึ่งมีคาบเวลาประมาณ 70 nS และมี Duty Cycle ประมาณ 50 % สัญญาณคล็อกอื่นๆของระบบเช่น คล็อกที่ป้อนให้กับ 8088 หรือชิปซัพพอร์ตต่างๆ นั้นจะถูกสร้างขึ้นโดยการหารสัญญาณคล็อกนี้ อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงในการใช้งาน OSC ก็คือสัญญาณนี้จะไม่

Synchronize กับสัญญาณอื่นๆ ของระบบ ดังนั้นจึงไม่ควรที่จะนำสัญญาณจากขา OSC นี้ไปใช้เป็นสัญญาณคล็อกสำหรับวงจรภายนอกอื่นๆ ที่ทำงานร่วมกับระบบ



รูปที่2.8 16 Bit slot ISA

CLK (Clock ; ขา B20)

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหารสัญญาณ OSC ด้วย 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz หรือมีช่วงเวลาใน 1 คาบ (ช่วงเวลาของคล็อก 1 ลูก) เท่ากับ 210 nS สำหรับค่า Duty Cycle ของสัญญาณนี้ค่าประมาณ 1/3 คือ ใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิก "1" เท่ากับ 1/3 ของคาบเวลาทั้งหมด หรือมีค่าประมาณ 70 nS และช่วงเวลาที่เป็นลอจิก "0" เท่ากับ 2/3 ของคาบเวลาทั้งหมด หรือประมาณ 140 nS สัญญาณนี้เป็นสัญญาณที่ใช้เป็นคล็อกของระบบ

A0 – A19 (Address Bus ; ขา A31 – A12)

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำหรืออุปกรณ์ I/O ที่ 8088(CPU) ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0– A19 นี้ จะถูกกำหนดโดย 8088 ในระหว่างขบวนการ DMA นั้น DMA – Controller จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสจะมีอยู่ 20 เส้น ซึ่งสามารถจะอ้างแอดเดรสของหน่วยความจำได้ถึง 64 Kbyte (สำหรับ IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำ ROM อีก 48 Kbyte ซึ่งถูกจัดในช่วงบนสุดใน 1 Mbyte คือ 0FC00H ถึง 0FFFFH (สำหรับ IBM PC/XT จะเป็น 64 Kbyte)

สำหรับการอ้างแอดเดรสของพอร์ท I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A0 – A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64 K พอร์ท โดยผ่านชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16 – A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสพอร์ทเพียง 10 เส้น คือจาก A0 – A9 และค่าแอดแดรดที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH เท่านั้น

Do-Do (Data Bus : ขา A9-A2)

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิท Do จะนัยสำคัญต่ำสุดและบิท D7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูล ก่อนที่สัญญาณ $\overline{IOW}$ (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ MEMW (ในกรณี

กรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ IOW หรือ MEMW นี้ จะถูกใช้เพื่อสั่งให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้น พอร์ท I/O หรือหน่วยความจำที่ถูกอ้างถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOR (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท) หรือ MEMR (ในกรณีที่ต้องการอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น)

RESET DRV (ขา B2)

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะเป็นแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และยังคงแอคทีฟไปจนกว่าระบบต่างๆ ภายใน IBM/PC จะพร้อมที่จะทำงานได้จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับแรงดันของแหล่งจ่ายตกลง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกันโดย ทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซทวงจรอินเทอร์เฟสหรืออุปกรณ์ I/O ต่างๆ ในช่วงที่เริ่มจ่ายไฟให้กับระบบ ซึ่งจะเป็นการทำให้วงจรหรืออุปกรณ์เหล่านั้นถูกปรับให้อยู่ในสภาวะที่แน่นอนก่อนที่จะเริ่มต้นการทำงานในระบบ(สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซต)

IOR (I/P Read ; ขา B14)

ขาสัญญาณนี้เป็นเอาท์พุตแอคทีฟที่ลอจิก "0" ที่สร้างโดย 8288 Bus Controler เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ $\overline{IOR}$ ประมาณ 30 nanosec เพื่อให้มั่นใจว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5 DMA Controler จะทำการสร้างสัญญาณ $\overline{IOR}$ เองโดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอด
เดรสของหน่วยความจำ(แทนที่จะเป็นแอดเดรสของพอร์ท I/O) ที่พอร์ท I/O ของ DMA ที่ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ทใดจะส่งข้อมูลมาบนพอร์ทนั้น จะอาศัยสัญญาณ DACK จาก DMA Controler เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอกทีพก็จะแสดงว่าพอร์ท

I/O ที่จะส่งข้อมูลออกมาบนบัสข้อมูลก็คือ พอร์ท I/O ที่ขอ DMA ผ่านทางเชนเนลที่ 1(DRQ1) เป็นต้น

IOW (I/O Write ; ขา B13)

ขาสัญญาณนี้เป็นเอาท์พุตแอคทีฟที่ลอจิก "0" ที่ถูกสร้างขึ้นโดย 8288 Bus Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ $\overline{IOW}$ นี้แอกทีพนั้นข้อมูลบนบัสอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ $\overline{IOW}$ แทนขอบขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อนสำหรับในขบวนการ DMAนั้น DMA- Controller จะทำการสร้างสัญญาณ $\overline{IOW}$ เองโดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

AEN (Address Enable ; ขา A11)

สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอกทีฟ(ลอจิก "1")นั้นเป็นบัสไซเคิลของขบวนการ DMA

สำหรับบนแมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการ ดิสเอเบิล(Disable) 8288 Bus Controller จะใช้ดิสเอเบิลพอร์ท I/O ต่างๆที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้น ที่จำเป็นต้องทำเช่นนี้ก็เพราะว่าในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรสและจะทำให้ $\overline{IOR}$ หรือ $\overline{IOW}$ แอกทีฟด้วยดังนั้นถ้าไม่ทำการ DISABLE พอร์ท I/O ที่ไม่เกี่ยวข้องไว้ก็อาจจะทำให้พอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดขึ้นได้

2.5 การอ้างแอดเดรสของพอร์ท I/O

ในการควบคุมและตรวจสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นซิฟซัพพอร์ทหรือการ์ดต่างๆ ที่ใช้ในระบบของ IBM/PC นั้นจะกระทำโดยผ่านทางพอร์ท I/O ของระบบดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้จึงจำเป็นที่จะต้องศึกษาถึงวิธีการควบคุม พอร์ท I/O ต่างๆของระบบด้วย และเนื่องจากการควบคุมหรือการติดต่อกับ

พอร์ทเหล่านี้ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ท I/O เหล่านี้โดยตรงเราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ท I/O ต่างๆนั้นจะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088(CPU) ซึ่งแอดเดรสเหล่านี้จะเป็นแอดเดรสที่ถูกจัดไว้สำหรับพอร์ท I/O โดยเฉพาะ คือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนของการส่งข้อมูลให้กับพอร์ทเหล่านี้จะกระทำโดยใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยังแอดเดรสของพอร์ทที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ทก็สามารถกระทำโดยใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการเช่นกัน

ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ท I/O อยู่ทั้งสิ้น 65,536 หรือ 64 K แอดเดรส ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้นดังนั้นในการอ้างถึงแอดเดรสของพอร์ทของอุปกรณ์หรือซิฟซัพพอร์ทใดๆที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นโดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเรส A10-A15 จะไม่ถูกนำไปใช้งานแต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT/IN อยู่ด้วยเพียงแต่ไม่ได้ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่างเช่นการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0010H นั้นจะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0410H,0810,0C10H ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งานจึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A10-A15 นั้นไม่ทำให้เกิดการเปลี่ยนแปลงใดๆทั้งสิ้น

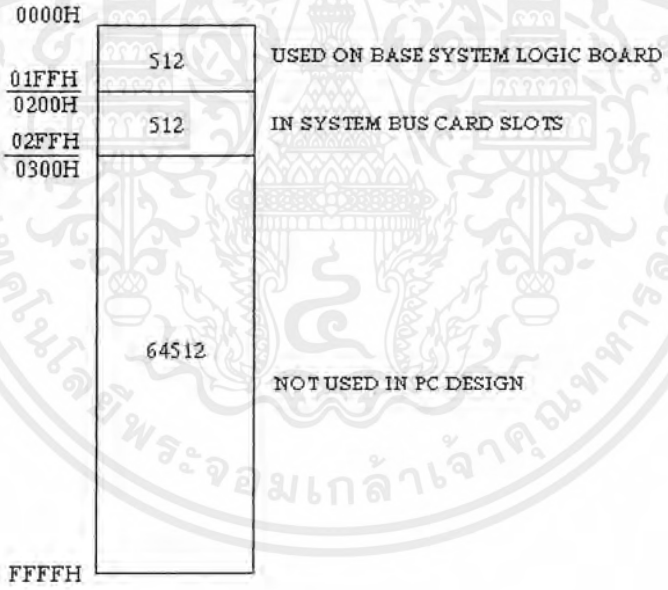
เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้นดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1,024 พอร์ทเท่านั้นนอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิทA9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1024 พอร์ทออกเป็น 2 กลุ่มโดยที่กลุ่มแรกเป็นกลุ่มของพอร์ทที่อยู่บนเมนบอร์ด และกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ทที่อยู่บนการ์ดต่างๆ

สำหรับในกรณีของการส่งข้อมูลให้กับพอร์ทบน IBM/PC ทั้ง 1024 พอร์ท เราสามารถที่จะเลือกส่งไปยังพอร์ทใดๆ ใน IBM/PC ได้ ดังนั้นการเลือกแอดเดรสสำหรับพอร์ทที่อยู่บนการ์ดจึงสามารถทำได้โดยสะดวก แต่อย่างไรก็ตามสิ่งหนึ่งที่ต้องคำนึงถึงคือ ถ้าแอดเดรสที่เลือกให้กับพอร์ทนี้ตรงค่ากับแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่ใน

ตำแหน่งแอดเดรสนี้ก็จะเท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ทที่อยู่บนเมนบอร์ดและพอร์ทที่อยู่บนการ์ดด้วย ซึ่งในกรณีนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน จึงควรใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น "1" คือแอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น (แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวก จึงควรให้กำหนดมีค่าเป็น "1" ในฐานสองให้หมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น "1" หรือ "0" ก็ได้)

2.6 การใช้งานแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

พอร์ท I/O ที่ผ่านมาทั้ง 1024 พอร์ท ใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่มๆ ละ 512 พอร์ทดังรูป2.9

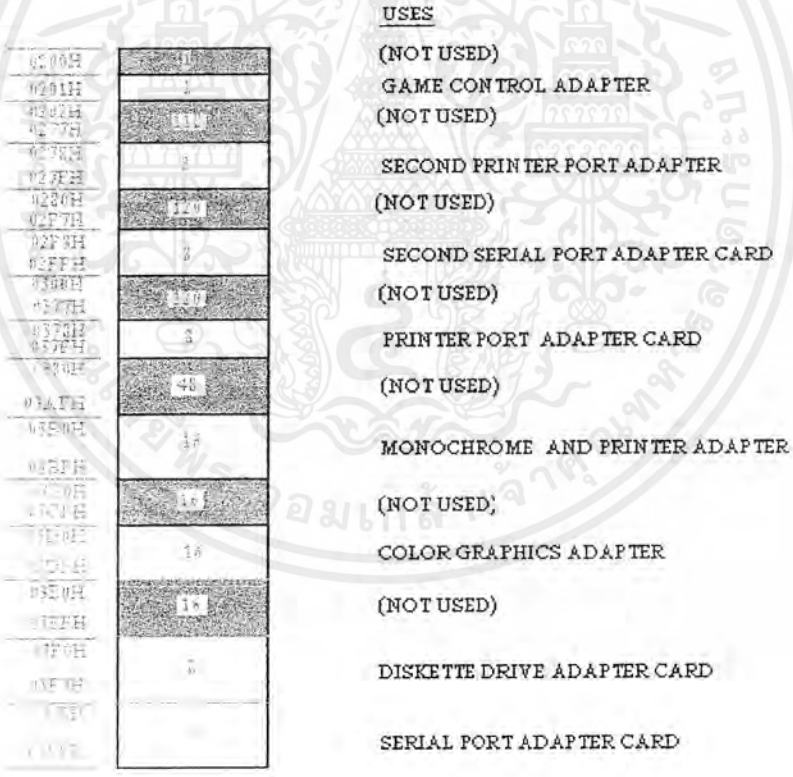


รูปที่2.9 การใช้งานแอดเดรสของพอร์ทบน IBM/PC

1. ในกลุ่มแรกนี้เป็นกลุ่มของพอร์ท I/O ที่อยู่บนเมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (ขอให้ระลึกอยู่เสมอว่า A0-A15 นั้นไม่ถูกใช้งาน)หรือแอดเดรสที่มีบิท A9 เป็น "0" นั่นเอง

สำหรับแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสของชิพชัพพอร์ทและอุปกณ์เป็น I/O ต่างๆ บนเมนบอร์ดของ IBM/PC เช่น แ อดเดรส 0000H จนถึง 000FH จะถูกใช้เป็นแอดเดรสสำหรับ 8237-5 DMA Controller เป็นต้น รูปที่2.9 แสดงถึงการใช้งานแอดเดรสต่างๆ ตั้งแต่ 0000H จนถึง 01FFH ในการอ้างแอดเดรสของชิพชัพพอร์ทและอุปกรณ์ต่างๆ ที่ทำหน้าที่เป็น I/O บนเมนบอร์ดของ IBM/PC

2. ในกลุ่มที่สองนี้ จะเป็นกลุ่มของพอร์ท I/O ที่ถูกใช้งานอยู่บนการ์ดที่ใช้เสียบบนสล็อตต่างๆ สำหรับแอดเดรสของพอร์ทเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือ แอดเดรสที่มีบิทที่ A9 เป็น "1" นั้นเอง สำหรับการใช้งานแอดเดรสของพอร์ท I/O ในกลุ่มนี้แสดงดังรูปที่2.10



รูปที่2.10 การใช้งานแอดเดรสสำหรับพอร์ท I/O บนการ์ดต่างๆ

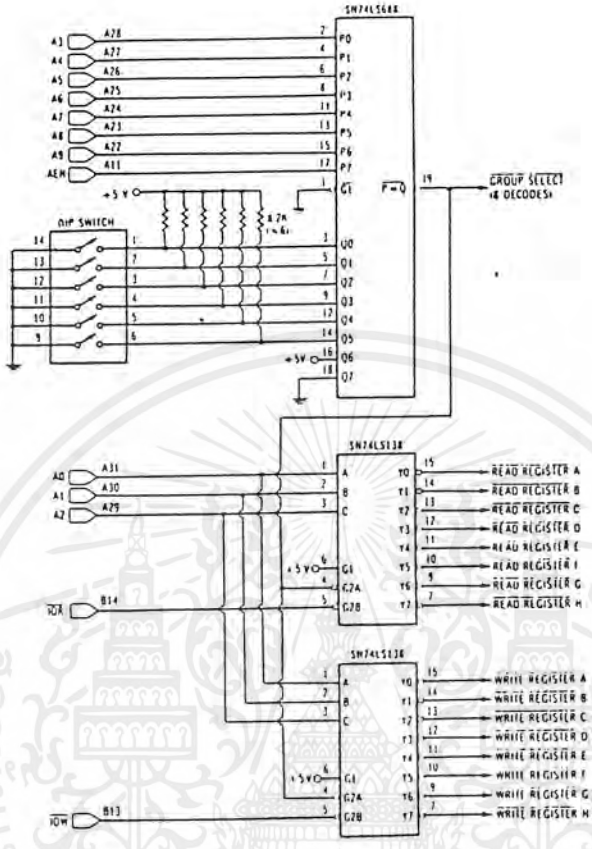
อย่างไรก็ตามการใช้งานแอดเดรสในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ ทั้งนี้ขึ้นอยู่กับการใช้งานการ์ดต่างๆ โดยที่การ์ดถูกออกแบบผลิตภัณฑ์ใหม่นั้น อาจจะใช้ค่าแอดเดรสต่างๆ ที่เหลืออยู่นี้ได้ ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเทอร์เฟสที่จำเป็นต้องใช้ค่าแอดเดรสสำหรับพอร์ท I/O จึงควรตรวจสอบก่อนดูว่าการ์ดต่างๆ ที่ใช้อยู่ในระบบของ IBM/PC ที่เราใช้งานอยู่นั้นมีการ์ดใดบ้าง และการ์ดเหล่านั้นใช้งานแอดเดรสใดบ้าง จากนั้นจึงทำการออกแบบวงจรอินเทอร์เฟส โดยเลือกใช้เฉพาะแอดเดรสที่ยังไม่ถูกใช้งาน

2.7 เทคนิคการดีโค้ดแอดเดรสสำหรับพอร์ท I/O

ในหัวข้อต่างๆ ที่ผ่านมาข้างต้นนั้น ได้กล่าวถึงการอ้างแอดเดรสและการใช้งานแอดเดรสต่างๆ ของพอร์ท I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการต่างๆ ที่ใช้ในการดีโค้ดแอด
เดรสต่างๆ ให้เป็นไปตามที่เราต้องการ ซึ่งการดีโค้ดมีหลายวิธี แต่จะอธิบายถึงการดีโค้ดโดยการใช้สวิตช์เลือก ซึ่ง Project นี้เราจะใช้วิธีนี้

การดีโค้ดโดยการใช้สวิตช์เลือก

จากการดีโค้ดในแบบ Fixed มีข้อเสียอยู่บางประการคือ แอดเดรสที่เราเลือกใช้งานไว้นั้นอาจจะซ้ำกับแอดเดรสของการ์ดอื่นที่เรานำมาเพิ่มเข้าไปในระบบในภายหลังก็ได้ ซึ่งกรณีเช่นนี้เราต้องแก้ไขวงจรเพื่อหลีกเลี่ยงไปใช้แอดเดรสอื่นที่ยังว่างอยู่ และไม่ถูกใช้งานโดยการ์ดที่จะเพิ่มเข้าไปใหม่ ซึ่งยุ่งยากและต้องเสียเวลามากขึ้น ปัญหาเช่นนี้เราสามารถแก้ไขได้โดยใช้วงจรดีโค้ดที่สามารถเปลี่ยนแปลงค่าแอดเดรสได้ โดยเพียงแต่เปลี่ยนตำแหน่งของสวิตช์(ในที่นี้คือ DIP Switch) ที่เซ็ตไว้ในวงจรเท่านั้น ดังรูปที่ 2.11



รูปที่2.11 ตัวอย่างวงจรดีโค้ดโดยใช้สวิตช์เลือก

จากรูปเป็นวงจรที่ทำการดีโค้ดกลุ่มแอดเดรสขนาด 8 แอดเดรส ซึ่งการเลือกกลุ่มแอดเดรสที่จะทำการดีโค้ดนี้จะทำโดยการเซ็ท DIP Switch ที่ขา Q0-Q5 ของ74LS688

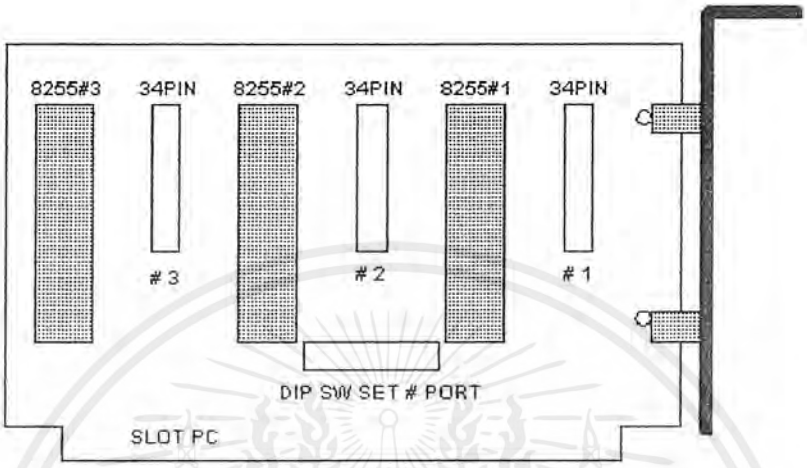
สำหรับหน้าที่ของ 74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุท 2 ชุดที่ถูกส่งเข้ามาทางขา P0-P7 และขา Q0-Q7 ถ้าอินพุททั้ง 2 ชุดนี้เท่ากันแล้ว เอาท์พุท P = Q จะให้เอาท์พุทเป็นลอจิก “0” จากในวงจรขา P0-P6 ของ 74LS688 ต่อกับแอดเดรสนิทรา A3-A9 ในขณะที่ขา Q0-Q5 ต่อกับความต้านทานที่ทำหน้าที่เป็น Pull Up (รักษาระดับแรงดันให้เป็นลอจิก “1” ไว้ในกรณีที่ไม่มีอินพุทใดๆ เข้ามา) และขา Q0-Q5 นี้จะต่อกับปลายข้างหนึ่งของDIP Switch ด้วย ส่วนปลายอีกข้างหนึ่งของ DIP Switch นั้นจะต่อลง Ground (ลอจิก “0”) ไว้ ดังนั้นถ้าเราทำการ “ON” DIP Switch ที่ต่อกับขาใดขานั้นก็จะได้รับลอจิก “0” ในขณะที่ถ้า DIP Switch ที่ต่อกับขาใดถูก “OFF” ขานั้นก็จะได้รับลอจิก “1” และเนื่องจากอินพุทที่ขา P0-P5 (แอดเดรส A3-A9) ต้องเท่ากับอินพุทที่ขา Q0-Q5

ดังนั้นถ้าเราเปลี่ยนแปลงการเซสเซอร์ DIP Switch เหล่านี้ก็จะทำให้แอดเดรสนิทรา A3-A5 ซึ่งต่อกับขา P0-P5 นั้นต้องเปลี่ยนแปลงตามไปด้วยจึงจะทำให้เอาท์พุทของ 74LS688 แอคทีฟได้ ทำให้เราสามารถเปลี่ยนค่าแอดเดรสที่ต้องต้องการจะดีโค้ดได้ง่ายกว่าวิธีการดีโค้ดแบบ Fixed สำหรับขา Q6 นั้นจะต่อกับลอจิก "1" (+5V) และขา P6 ต่อกับแอดเดรสนิทรา A9 ในกรณีเช่นนี้จึงเท่ากับเป็นการบังคับให้แอดเดรสที่จะทำการดีโค้ดนั้น จะต้องทีแอดเดรสนิทรา A9 เป็น "1" เท่านั้นส่วนขา P7 จะต่อกับสัญญาณ AEN โดยขา Q7 ต่อกับลอจิก "0" การต่อในลักษณะนี้ก็เพื่อป้องกันไม่ให้ 74LS688 ทำการดีโค้ดในระหว่างขบวนการ DMA นั่นเอง เอาท์พุทจากขา P = Q ของ 74LS688 นี้จะถูกนำไปใช้ในการอีนาเบิ่ง 74LS688 ซึ่งทำหน้าที่ในการดีโค้ดแอดเดรส ค่าแอดเดรสของกลุ่มแอดเดรสที่เราเลือก (โดยใช้ DIP Switch ดังที่ได้กล่าวในตอนต้น)

วงจรในลักษณะนี้เราสามารถจะนำไปใช้เป็นการดีโค้ดในแบบ Fixed ได้โดยการนำเอา DIP Switch ออก จากนั้นถ้าอินพุทใดต้องการลอจิก "0" จึงจะใช้ตัวนำเชื่อมต่อระหว่างขั้วทั้งสองแทนการเซ็ท DIP Switch ให้ "ON" แต่ถ้าอินพุทใดต้องการลอจิก "1" ก็ปล่อยขั้วทั้งสองนั้นไว้

2.8 ข้อมูลเกี่ยวกับ ET-PC8255 การ์ด

ลักษณะของ ET-PC 8255



รูปที่2.12 ลักษณะของ ET-PC 8255 Card

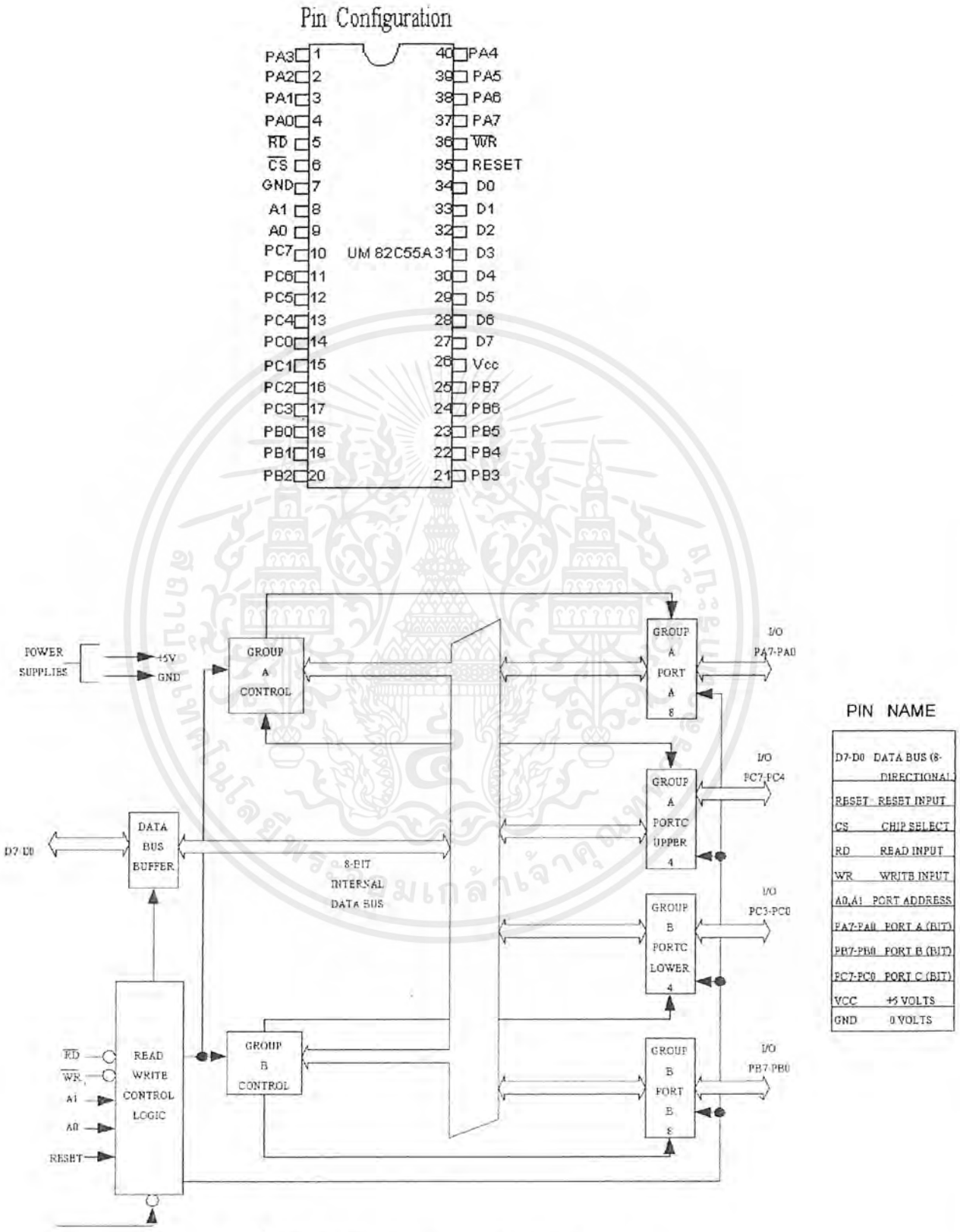
ET-PC 8255 จะเป็นการ์ดต่อขยายระบบเครื่อง PC ให้มีส่วนของอินพุท เอาท์พุทพอร์ท ใช้งานมากขึ้นโดยจะมีพอร์ทใช้งานเป็น อินพุทหรือ เอาท์พุทจำนวน 9 พอร์ทหรือ 72 บิท I/O(TTL 0-5)

การทำงานของ ET-PC 8255

CARD ET-PC 8255 จะประกอบไปด้วย 2 ส่วนใหญ่ๆ ก็คือ ส่วน IC 8255 ซึ่งเป็น IC ทำหน้าที่เป็น INPUT,OUTPUT PORT และส่วนของวงจร IC DECODE(เลือกตำแหน่งของ พอร์ท 8255) คือ IC 74LS688,74LS139 และ DIP SW

การใช้งาน IC 8255

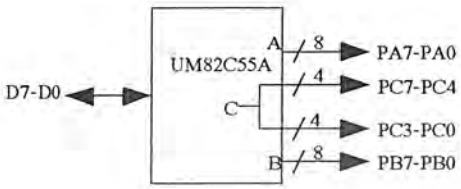
IC 8255 นี้จะเป็น IC ซึ่งประกอบด้วย PORT ใช้งาน 3 PORT และอีก 1 PORT ควบคุมก่อนที่เราจะใช้งาน 8255 เราจะต้องส่งข้อมูลไปยัง PORT ควบคุมก่อนว่าจะให้ PORT ทั้ง 3 PORT ของ 8255 นั้นทำอะไร เป็น INPUT หรือ OUTPUT PORT เราจะต้องเป็นผู้กำหนด CONTROL CODE PORT ควบคุมดังรูปที่2.14



รูปที่2.13 Pin Configuration and Block Diagram

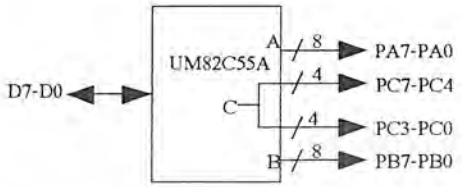
CONTROL WORD #0

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



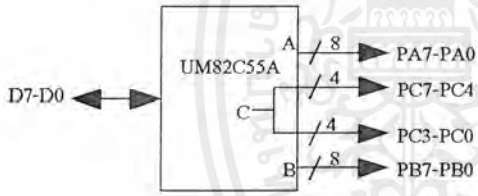
CONTROL WORD #4

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



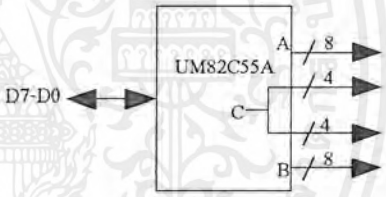
CONTROL WORD #1

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



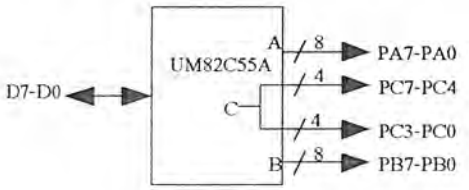
CONTROL WORD #5

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



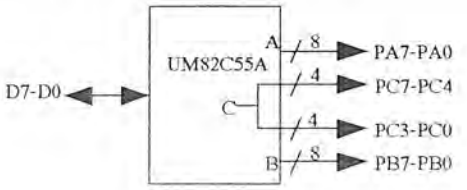
CONTROL WORD #2

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



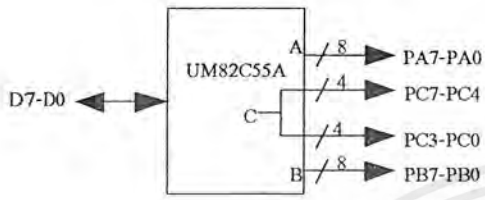
CONTROL WORD #6

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



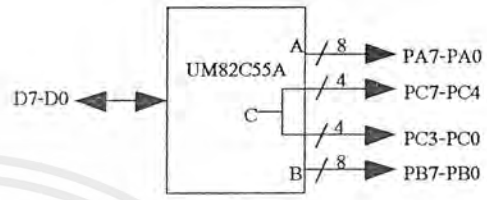
CONTROL WORD #3

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



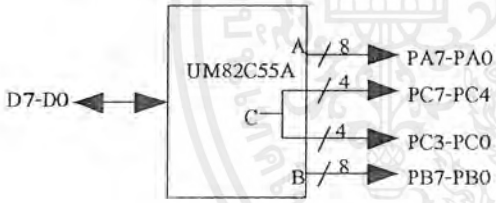
CONTROL WORD #7

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



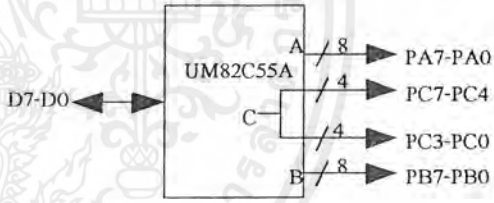
CONTROL WORD #8

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



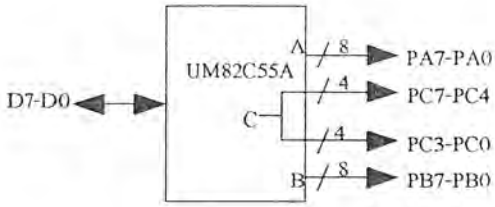
CONTROL WORD #12

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



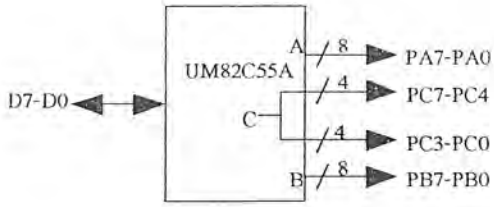
CONTROL WORD #9

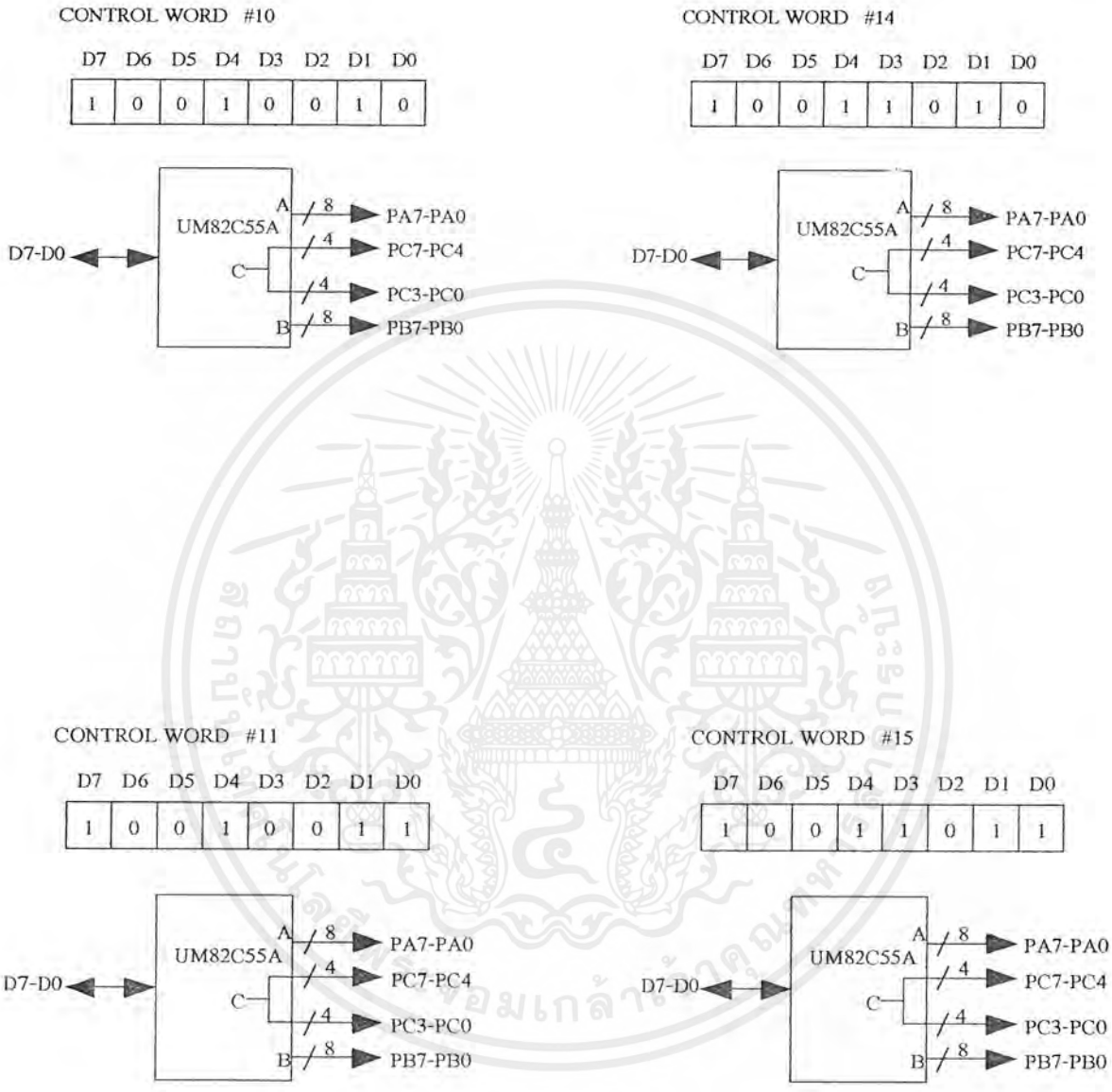
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	0	1



CONTROL WORD #13

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





รูปที่2.14 CONTROL CODE PORT

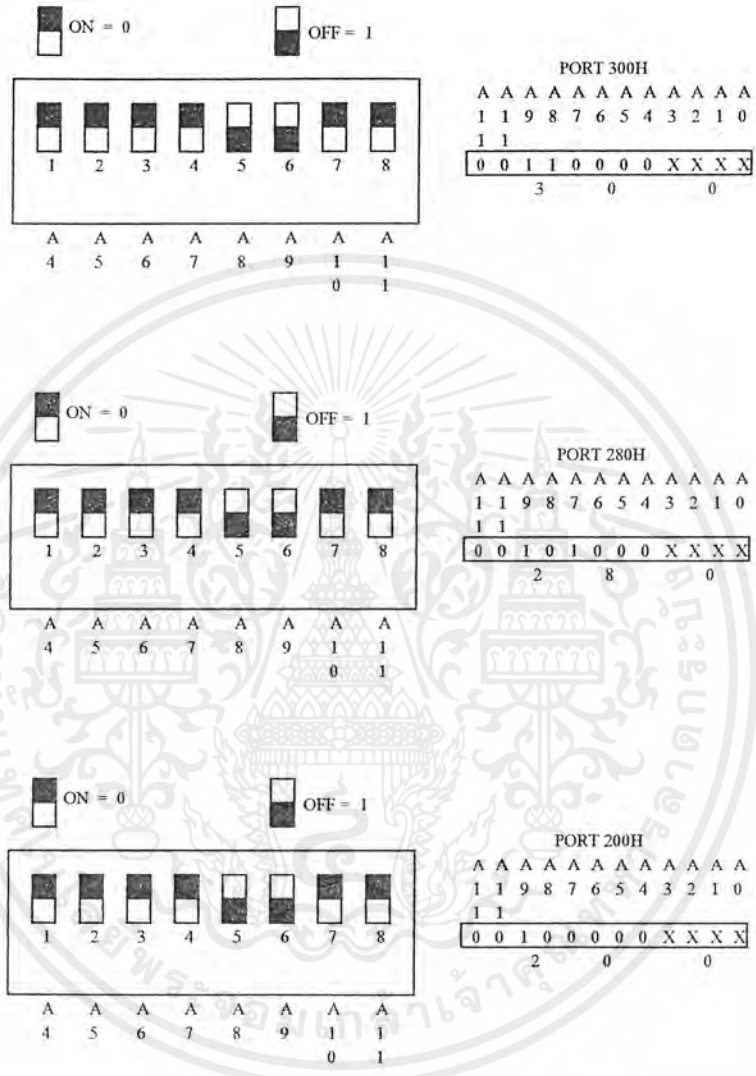
การ DECODE PORT

DECODE PORT 8255 บนการ์ดเราจะใช้ IC TTL 74LS688,IC TTL 74LS139 และ DIP SW 8 PIN เป็นวงจร DECODE เพื่อให้สามารถปรับ SET DIP SW ตั้งตำแหน่งเบอร์ PORT ของการ์ด ได้โดยในการปรับ DIP SW นั้นจะต้องไม่ปรับไปตรงตำแหน่ง PORT ของเครื่องคอมพิวเตอร์ด้วยดังรูปที่2.15

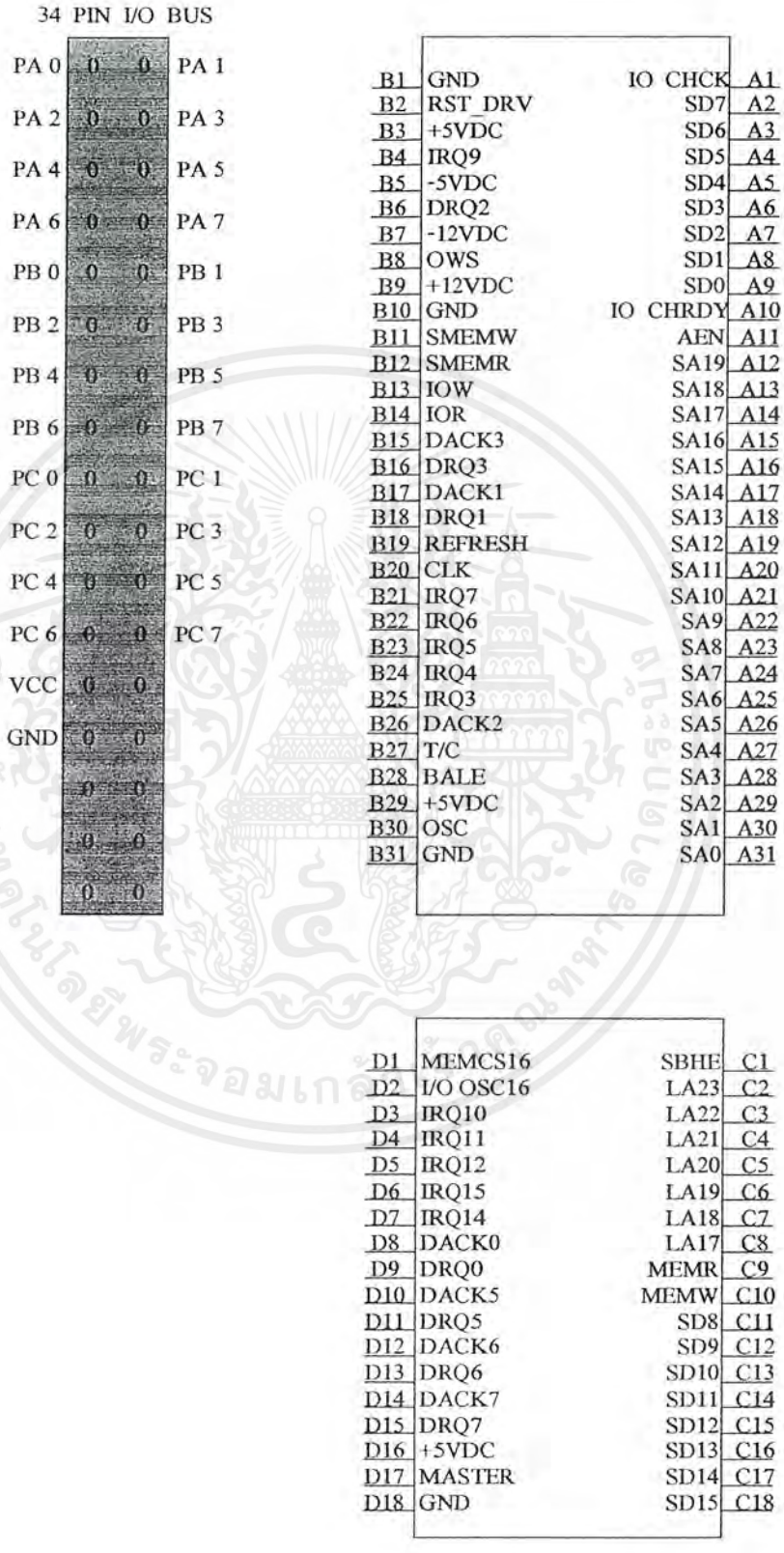
โดย CARD ET-PC8255 จะใช้ตำแหน่ง PORT 12 PORT ต่อ CARD DECODE PORT

XX0H	PORT A (#1)
XX1H	PORT B (#1)
XX2H	PORT C (#1)
XX3H	CONTROL PORT (#1)
XX4H	PORT A (#2)
XX5H	PORT B (#2)
XX6H	PORT C (#2)
XX7H	CONTROL PORT (#2)
XX8H	PORT A (#3)
XX9H	PORT B (#3)
XXAH	PORT C (#3)
XXBH	CONTROL PORT (#3)

เราตั้งเบอร์ DECODE PORT ได้โดยการปรับ DIP SW ซึ่งก็มีค่าเท่ากับค่า ADDRESS นั้นๆ เช่นเราตั้งตำแหน่ง 300H จะเซ็ต DIP SW ดังนี้



รูปที่2.15 การ DECODE PORT



รูปที่2.16 CONNECTOR 34 PIN AND ISA SLOT

2.9 คีย์บอร์ด (Keyboard)

คีย์บอร์ดนี้เป็นอุปกรณ์อินพุทที่เชื่อมต่อกับผู้ใช้โดยตรง คีย์บอร์ดของเครื่องไมโครคอมพิวเตอร์จะเชื่อมผ่านหัวต่อแบบ DIN โดยมีไดอะแกรมของหัวต่อดังรูปที่ 2.1 ต่อกับด้านหลังของเครื่องไมโครคอมพิวเตอร์ สายเชื่อมโยงของคีย์บอร์ดนี้มี 5 เส้น ประกอบด้วย

1. สัญญาณนาฬิกาคีย์บอร์ด (Keyboard Clock)
2. กราวนด์ (Ground)
3. ข้อมูลคีย์บอร์ด (Keyboard Data)
4. +5 โวลท์ (Power Supply +5V)
5. ไม่มีการเชื่อมต่อ (No Connection)

รูปที่2.17 แสดงหัวต่อ DIN ของคีย์บอร์ด

การทำงานของคีย์บอร์ด

การทำงานของคีย์บอร์ดคือ เมื่อเรากดคีย์ใดๆ บนคีย์บอร์ดแล้ว จะมิได้หมายถึงการส่งตัวอักษรบนคีย์ตัวนั้นเข้าสู่เครื่องคอมพิวเตอร์โดยทันที เช่นเมื่อเรากดปุ่ม A ก็มิได้หมายถึงจะส่งตัวอักษร A ในทันที ซึ่งเมื่อกดคีย์แล้ว ซีพียูที่อยู่คีย์บอร์ดนี้จะตรวจสอบการดกคีย์นั้นๆ และทำการบันทึกรหัสสแกนไว้

เหตุผลที่แป้นพิมพ์ไม่ส่งอักษรของปุ่มที่กดเข้าสู่เครื่องโดยตรงก็เพราะถึงแม้ว่าฮาร์ดแวร์ของคอมพิวเตอร์มีประสิทธิภาพสูงแต่มันไม่อยู่ในสถานะที่จะทำงานต่างๆเองได้ มันไม่มีหน้าที่ในการให้ความหมายสิ่งต่างๆ ที่เราทำ ซอฟต์แวร์จะเป็นตัวทำให้ฮาร์ดแวร์ทำงาน ดังนั้นในกรณีที่กล่าวมาแล้วว่าแป้นพิมพ์เป็นตัวให้ความหมายว่า เมื่อกดปุ่ม A จะหมายถึงตัวอักษร A ย่อมไม่เป็นความจริง เพราะแป้นพิมพ์ฮาร์ดแวร์ย่อมไม่สามารถทำหน้าที่ของซอฟต์แวร์ได้ เมื่อมีการกดปุ่มใดปุ่มหนึ่งบนแป้นพิมพ์ จะเกิดขั้นตอนการทำงานดังนี้คือ แป้นพิมพ์จะรับรู้ว่ามีการกดปุ่มขึ้น และทำการบันทึกรหัสสแกน (scan code) ไว้ หลังจากนั้นแป้นพิมพ์จะส่งอินเทอร์รัพต์หมายเลข 9 ให้แก่โปรเซสเซอร์(CPU)เพื่อบอกกับคอมพิวเตอร์ว่าขณะนี้ได้มีการกดปุ่มเกิดขึ้น ซึ่งก็จะทำให้ไมโคร

โปรเซสเซอร์หยุดค้างการทำงานใดๆ ที่กำลังทำอยู่ไว้ชั่วคราวเพื่อกระโดดไปยังโปรแกรมที่ดูแลจัดการการอินเทอร์รัพต์เสียก่อน อันได้แก่รอมไบออสนั่นเอง ในจุดนี้รอมไบออสในส่วนที่ดูแลการอินเทอร์รัพต์จากแป้นพิมพ์ก็จะเข้ามาทำงาน ด้วยการดูว่าเกิดอะไรขึ้นที่แป้นพิมพ์ โดยส่งคำสั่งไปยังแป้นพิมพ์เพื่อตรวจสอบว่าเกิดอะไรขึ้นแป้นพิมพ์จะตอบกลับโดยบอกกับไบออสว่าปุ่มใดถูกกด ซึ่งการโต้ตอบเหล่านี้จะดำเนินผ่านพอร์ตที่แป้นพิมพ์ใช้อยู่ จากนั้นแป้นพิมพ์จะโต้ตอบโดยส่งรหัสสแกนของปุ่มที่กดไปยังตำแหน่งพอร์ตอีกตำแหน่งซึ่งเป็นตำแหน่งที่รอมไบออสใช้ในการอ่าน ดังที่ทราบมาแล้วว่าแป้นพิมพ์จะเก็บค่ารหัสสแกนของปุ่มที่ถูกกดและรอจนกระทั่งไบออสเรียกใช้ ซึ่งเวลาที่รอนี้จะสั้นมากประมาณ 1/1000 วินาที ดังนั้นแป้นพิมพ์จะมีหน่วยความจำขนาดเล็กอยู่ด้วยเพื่อใช้ในการบันทึกค่ารหัสสแกนที่ยังไม่ถูกส่งไปให้กับไบออสเอาไว้ ในบางครั้งเราก็เรียกเนื้อที่ส่วนนี้ว่าเป็นบัฟเฟอร์ของแป้นพิมพ์ (keyboard buffer) ซึ่งโดยทั่วๆ ไปหน่วยความจำที่ว่านี้จะมีขนาดพอที่จะเก็บบันทึกรหัสสแกนไว้ได้จำนวนหนึ่ง อย่างไรก็ดีซึ่งแป้นพิมพ์จะเก็บรหัสสแกนทั้งตอนที่เรากดปุ่มและตอนที่ปล่อยปุ่ม โดยค่ารหัสสแกนของการกดปุ่มและปล่อยปุ่มนี้จะแตกต่างกัน กล่าวคือรหัสของการปล่อยปุ่มพิมพ์จะมีค่าเท่ากับรหัสการกดปุ่มบวกด้วย 128 หรือ 80 ในฐานสิบหกนั่นคือเมื่อมีการกดปุ่ม 1 ครั้งที่แป้นพิมพ์ ซึ่งหมายถึงต้องมีการกดปุ่มและปล่อยปุ่มนั่นเองจะทำให้ไบออสถูกอินเทอร์รัพต์ 2 ครั้ง ดังนั้นไบออสจึงสามารถรับรู้ได้ว่าขณะนี้ปุ่มกำลังถูกกดอยู่ หรือถูกปล่อยแล้ว ตัวอย่างเช่น ไบออสจะสามารถรับรู้ได้ว่าคุณกำลังกดปุ่มอักษรตัวใหญ่เพราะปุ่ม shift จะถูกกดค้างไว้โดยไม่ปล่อย

ในกรณีที่เรากดปุ่มค้างไว้โดยไม่ปล่อยจะเป็นหน้าที่ของส่วนฮาร์ดแวร์ของแป้นพิมพ์ที่จะรับรู้ เพราะการกดค้างนั้นนานเกินกว่าที่กำหนดไว้ ประมาณครึ่งวินาที และแป้นพิมพ์ก็จะทำการส่งรหัสสแกนพิเศษอันเป็นรหัสที่หมายถึงการกดปุ่มค้าง (repeat key)

กลับมาดูการทำงานของไบออสต่อเมื่อตัวจัดการอินเทอร์รัพต์ที่ดูแลแป้นพิมพ์ของไบออสเริ่มทำงาน มันจะรับค่ารหัสสแกนจากแป้นพิมพ์ โดยในคีย์บอร์ดต่างชนิดจะมีรหัสสแกนที่ไม่เหมือนกัน และจะจัดการให้ความหมายแก่มัน โดยผ่านการวิเคราะห์อย่างรวดเร็วหลายขั้นตอน เริ่มแรกมันจะตรวจสอบว่ามีการกดปุ่มประเภทพิเศษหรือไม่ บรรดาปุ่มพิเศษประเภท shift ได้แก่ปุ่ม shift,Alt ,Ctrl ทั้งทางด้านซ้ายและขวา ซึ่งถ้ามีการกด มันจะบันทึกสถานะ shift (shift state) ต่อจากนั้นไบออสจะตรวจสอบว่ามีการกดปุ่มประเภท Toggle หรือไม่ ปุ่มประเภทนี้ได้แก่ปุ่ม Caps Lock,NumLock,ScrollLock และประเภท Toggle สักนิดลักษณะการใช้งานของปุ่มประเภท Shift นั้นจะต้องกดค้างไว้และกดปุ่มอื่นตาม ซึ่งจะให้ความหมายที่แตกต่างจากการกดปุ่มตัวอักษรดังกล่าวเฉยๆ เช่น การกดปุ่ม Shift ตามด้วยปุ่ม a ก็จะทำให้หมายถึง A เป็นต้น ในขณะที่การกดปุ่ม

ประเภท Toggle จะกดเพื่อเปลี่ยนสถานะเช่น การกด Caps Lock จะเป็นการเปลี่ยนสถานะให้การกดปุ่มตัวอักษรในเวลาต่อมา
กลายเป็นตัวใหญ่หมด และถ้าเรากด Caps Lock อีกทีก็จะเปลี่ยนสถานะกลับมาสู่ปกติดังเดิม โดยสรุปแล้วปุ่มทั้งสองประเภทนี้จะสามารถส่งผลต่อการแปลงรหัสสแกนได้ โดยปุ่มประเภท Shift จะส่งผลขณะที่มันถูกกดค้างไว้เท่านั้นในขณะที่ปุ่มประเภท Toggle จะส่งผลเมื่อมันอยู่ในสถานะ on สถานะทั้ง 2 นี้จะถูกเก็บไว้ในเนื้อที่หน่วยความจำส่วนล่าง อันได้แก่ตำแหน่งแอดเดรส 417 และ 418 (ฐานสิบหก) โดยแต่ละนิทราในแต่ละไบต์จะเก็บสถานะภาพของการกดปุ่มเอาไว้ กล่าวคือในไบต์แรกจะเก็บสถานะของการที่มีการกดปุ่มค้างและไบต์ที่สองจะเก็บสถานะของปุ่ม Toggle ต่างๆว่าเป็น on หรือ off นอกเหนือจากข้อมูลที่เก็บในตำแหน่งที่ 417 และ 418 แล้วยังมีข้อมูลที่เก็บ ณ ตำแหน่ง 496 และ 497 ซึ่งจะไม่ใช้ในคีย์บอร์ดรุ่นเก่าๆ ในเนื้อที่ส่วนนี้จะบรรจุสภานะเอาไว้ว่า ปุ่ม Alt หรือ Ctrl ด้านขวามือถูกกดค้างไว้หรือไม่

หลังจากที่รอมไบออสได้ตรวจสอบว่าได้มีการกดปุ่มประเภท Shift หรือ Toggle หรือไม่และอยู่ในสถานะอะไรเป็นที่เรียบร้อยแล้ว มันก็จะตรวจต่อไปว่าได้มีการกดปุ่มประเภทที่จะทำให้เกิดความหมายพิเศษหรือไม่ เช่น กดปุ่ม Ctrl-Alt-Del พร้อมๆ กันจะทำให้เกิดการบูตดอสขึ้นมาใหม่ หรือการกดปุ่ม Pause ก็จะเกิดการหยุดการทำงานชั่วคราว ในที่สุดหลังการตรวจสอบดูแล้วพบว่ามีการกดปุ่ม 2 ประเภทนี้ คือปุ่มประเภท Shift หรือ Toggle พร้อมๆ กับการกดปุ่มอื่นๆ อีกปุ่มหนึ่ง ไบออสจะจัดการให้ความหมายแก่มัน เช่นตรวจสอบดูแล้วพบว่ามีการกดปุ่ม Shift และปุ่ม a พร้อมกันก็จะหมายถึง A ความหมายที่ได้นี้ แบ่งได้เป็น 2 ประเภทคือประเภทแรกเป็นรหัสแอสกี้ ที่เกิดจากการกดปุ่มแอสกี้ใดๆ เช่น A , control-A ส่วนประเภทที่ 2 ได้แก่รหัสพิเศษที่เกิดจากการกดปุ่มพิเศษ เช่นปุ่ม function ต่างๆ หรือปุ่มที่ใช้เลื่อนเคอร์เซอร์แทนเป็นต้น

ในการเก็บบันทึกรหัสแอสกี้และรหัสพิเศษของพีซีนั้น ไบออสจะใช้เนื้อที่ 2 ไบต์ในการเก็บ โดยถ้าเป็นตัวรหัสแอสกี้ ก็จะเก็บรหัสแอสกี้ไว้ไบต์แรก และเก็บรหัสสแกนของปุ่มนั้นไว้ในไบต์ที่ 2 แต่ถ้าเป็นตัวอักษรพิเศษซึ่งไม่มีรหัสแอสกี้ ก็จะเก็บ 0 ไว้ที่ไบต์แรกและเก็บรหัสของปุ่มคีย์พิเศษนี้ไว้ในไบต์ที่ 2

หลังจากที่เรากดปุ่มคีย์ ซึ่งจะเก็บรหัสสแกนไว้ในบัฟเฟอร์ของคีย์บอร์ด และไบออสจะเข้ามาแปลงปุ่มดังกล่าวเป็นข้อมูล 2 ไบต์แล้วนั้น ข้อมูลทั้ง 2 ไบต์นี้ก็จะถูกเก็บไว้ในบัฟเฟอร์ของไบออสในส่วนที่กันไว้สำหรับคีย์บอร์ด โดยที่ไบออสได้กันเนื้อที่ไว้สำหรับเก็บข้อมูล 32 อักขระ และถ้าเกิดการล้นของข้อมูล (overflow) มันจะส่งเสียง beep และตัดข้อมูลที่เข้ามาที่หลังทิ้ง หลังจากที่ไบออสแปลความหมายได้แล้ว โปรแกรมของเราก็สามารถไปดึงความหมายนี้มาได้โดยตรงจากรอมไบออส หรืออาจจะผ่านดอส ซึ่งดอสก็จะไปดึงมาจากรอมไบออสอีกที

บทที่3

หลักการทำงานของ HARDWARE (BRAILLE INDICATOR) และการทดสอบวงจร

3.1 หลักการทำงานของ HARDWARE

จากอักษรเบรลล์ จะเป็นจุด(dot) ดังนั้นทำอย่างไรจึงจะให้คนตาบอดรับรู้อักษรต่างๆหรือข้อความต่างๆได้

อักษรหนังสือนูนสำหรับคนจักษุพิการ (The Braille Alphabet)

หนังสือนูนสำหรับคนจักษุพิการ เป็นอักษรสำหรับคนตาบอด เป็นระบบของตัวอักษรทำให้เป็นรูปนูนขึ้นมาหรือเป็นการแสดงให้อักษรนูนขึ้นมา โดยรูปแบบจะใช้การรวมของ จุด 6 จุด ซึ่งประกอบด้วย แนวตั้ง 2 columns ซึ่งแต่ละ columns มี 3 จุด นั้นคือ 1 ช่องประกอบด้วย 6 ตำแหน่ง ดังรูปที่3.1

รูปที่3.1 หนังสือนูนสำหรับคนจักษุพิการ(The braille alphabet)

จะเห็นได้ว่าหนังสือนูนสำหรับคนจักษุพิการจะแบ่งเป็นเซล์ล(cell)ตัวอักษรนี้สามารถใช้สัญญลักษณ์ทางคณิตศาสตร์หรือวิทยาศาสตร์แทนคำพูดได้ ในการแสดงหนังสือนูนสำหรับคนจักษุพิการต้องมีเครื่องแสดงออกมาเป็นหนังสือนูน เพราะฉนั้นหลักการคือ เราจะใช้เซล์ล ดังรูปที่ 3.1 ทำกี่เซล์ลก็ได้ตามต้องการ แต่ละจุดจะใช้เหล็กแหลมเป็นตัวแสดงอักษรที่นูนขึ้นมา

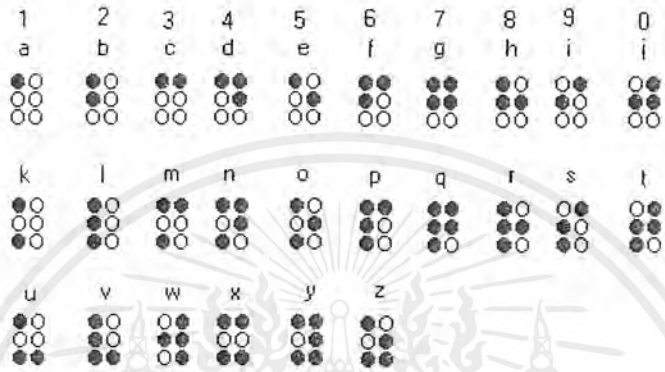
Note: ● = embossed dot;
○ = unused dot in that character

embossed dot แทนจุดที่ใช้ในการแสดงอักษรนูน

unused dot in that character แทนจุดที่ไม่ใช้ในการแสดงอักษรนูน

ดังนั้นเราจะมีอักษรเบรลล์ที่เป็นมาตรฐานของโลกดังนี้ เพื่อแสดงให้คนตาบอดใช้มือแตะที่แผงอักษรเบรลล์

ALPHABET AND NUMBERS



รูปที่3.2 แสดงอักษร(character) ในรูปอักษรเบรลล์

PUNCTUATION AND COMPOSITION SIGNS

1. Colon : แทนด้วย

2. Period . แทนด้วย

3. Question mark ? แทนด้วย

4. Semicolon ; แทนด้วย

5. Comma , แทนด้วย

6. Exclamation point ! แทนด้วย

7. Parenthesis, opening (แทนด้วย

8. Parenthesis, closing) แทนด้วย

9. Apostrophe ' แทนด้วย

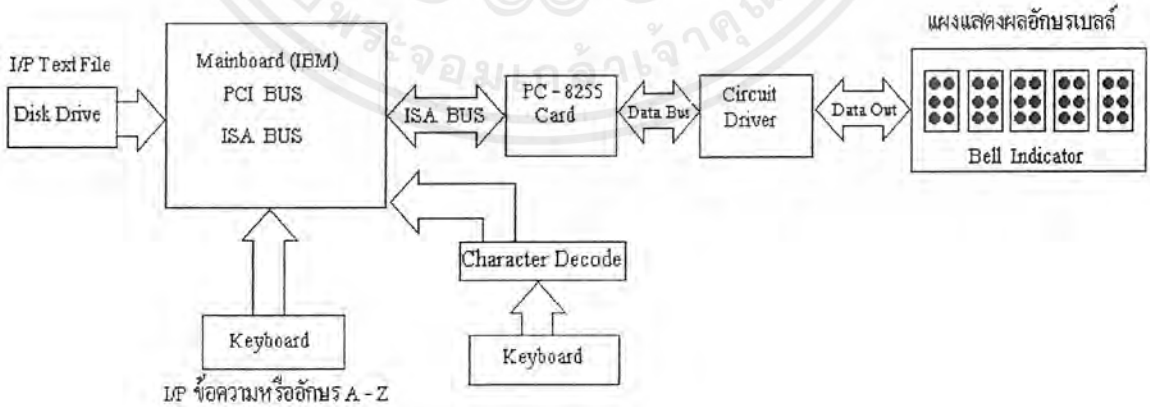
10. Capital sign,double " แทนด้วย

11. Number sign # แทนด้วย

รูปที่3.3 แสดงเครื่องหมายและสัญลักษณ์ต่างๆ ในรูปอักษรเบรลล์

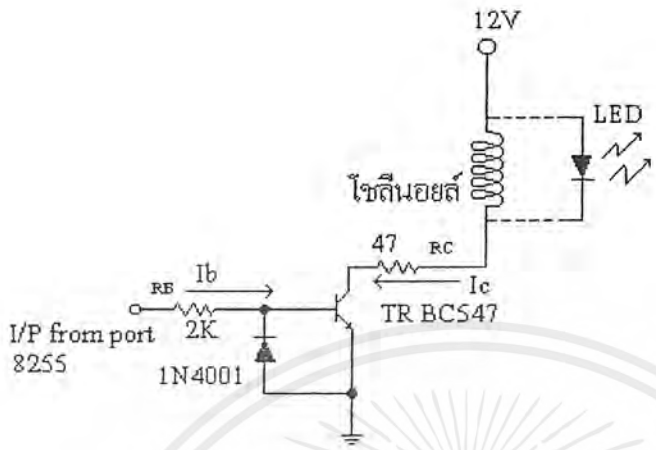
เราก็จะใช้อักษรเบรลล์ที่เป็นมาตรฐานของโลกนี้ในการแสดงให้คนตาบอดรับรู้ คืออักษร a-z หรือ A-Z และเครื่องหมายและสัญลักษณ์ต่างๆ

3.2 หลักการของแผงแสดงผลอักษรเบรลล์(Braille indicator)



รูปที่3.4 Block Diagram แผงแสดงอักษรเบรลล์

3.3 หลักการออกแบบวงจร Braille indicator driver circuit



รูปที่3.5 วงจรขับโซลีนอยร์

จากการออกแบบวงจรดังรูปที่ 3.5 ซึ่งใช้ในการขับโซลีนอยร์เพื่อให้แสดงผลออกมาเป็นอักษรเบรลล์ และจากอักษรเบรลล์นั้น 1 เซล์ล จะมี 6 จุด หรือ 6 ตำแหน่งตามที่ได้กล่าวมาข้างต้น โดย Project นี้จะทำขึ้นมาจำนวน 5 เซล์ลดังรูป3.4 เพื่อเป็นต้นแบบก่อน (และสามารถพัฒนาให้มีจำนวนเซล์ลเพิ่มขึ้นเพื่อเพิ่มประสิทธิ์ภาพ คือสามารถอ่านข้อความหรือประโยคได้ทีละมากๆ) ซึ่งจาก 5 เซล์ล จะได้ 30 จุด หรือ 30 ตำแหน่งนั่นเอง เพราะฉะนั้นเราจะต้องใช้ วงจรขับโซลีนอยร์จำนวน 30 ชุดเช่นกัน

หลักการทำงานของวงจร ก็คือ TR BC547 ทำหน้าที่เป็น Switch on/off เพื่อทำให้โซลีนอยร์นำกระแส และหยุดนำกระแส โดยบังคับให้โซลีนอยร์ และ TR BC547 นำกระแสและหยุดนำกระแสได้โดยป้อน Input ที่เป็น logic "1"(5V)และlogic "0"(0V)ตามลำดับ

<u>การทดลองและผลการทดลองจะเป็นดังนี้</u>

การคำนวนค่า Resister ในวงจรทำได้ดังนี้

- จาก TR BC547 มีเบต้า(B)หรืออัตราการขยายเท่ากับ 200
- โซลีนอยร์ ต้องการกระแสต่ำสุดในการทำงานประมาณ 80mA
- กำหนด ให้ RB = 2K
- ในสภาวะที่ Input เป็น 5V จะได้ IB = 5V-0.7V/2K=2.15mA ซึ่งได้

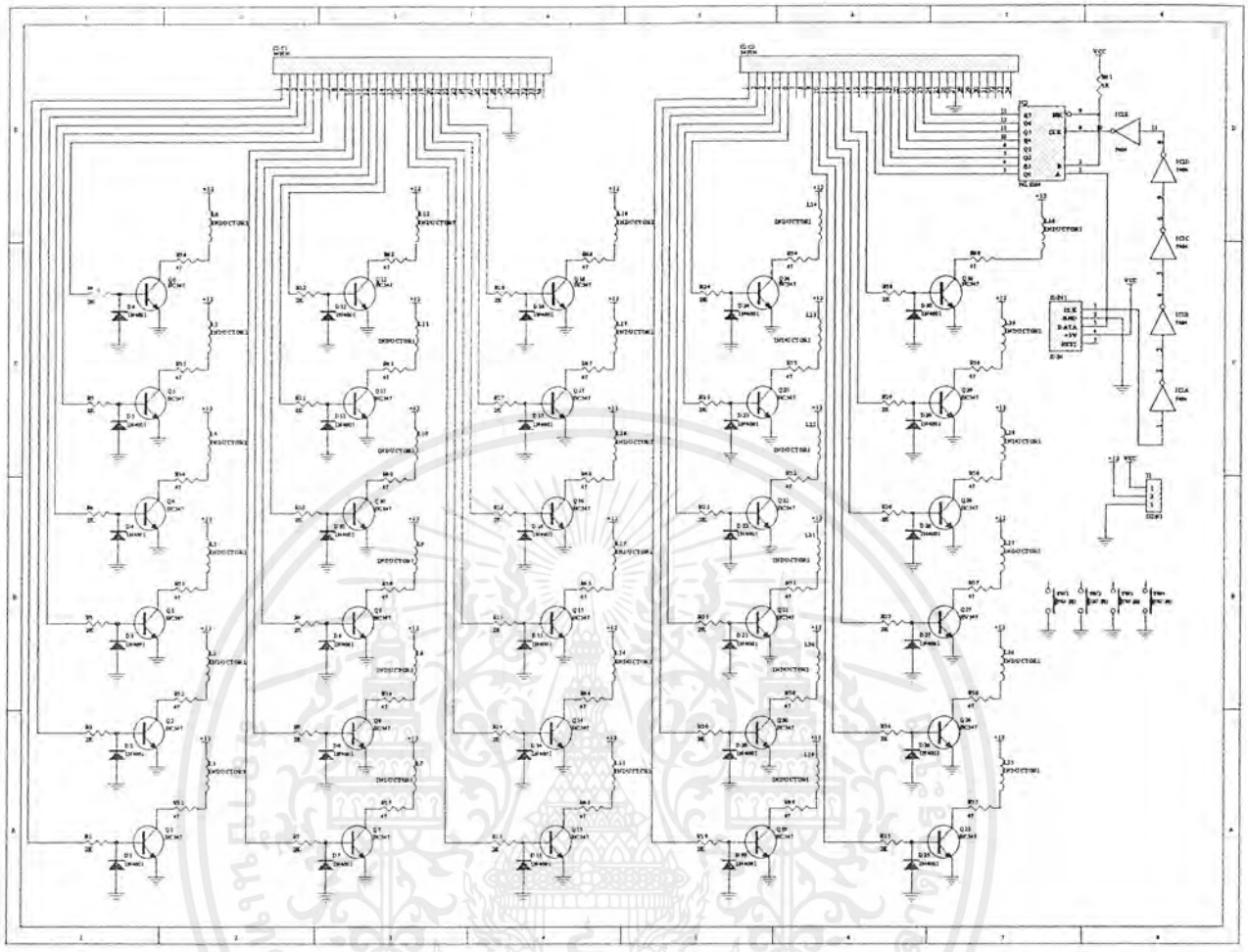
IC max = BxIB

=200x2.15mA

=430mA

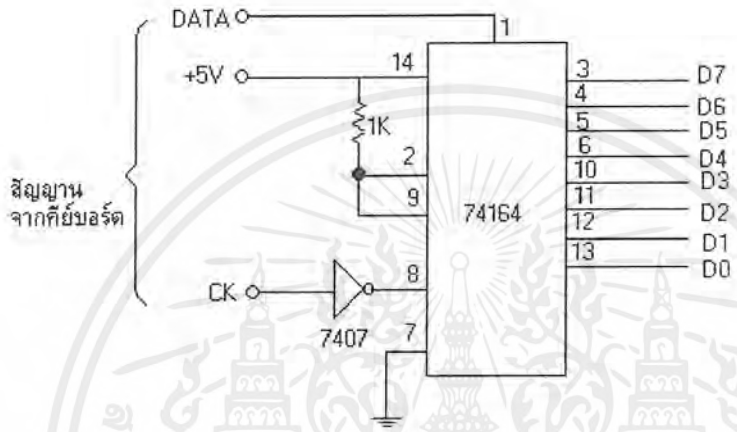
เพราะฉะนั้นเราต้องคำนวน Resister จำกัดกระแส RC(Resister ที่ขา C ของ Transister) โดยค่า RCนี้ต้องจำกัดกระแสไม่ให้กระแส IC เกินค่า IC max คือ 430 mA เพื่อไม่ให้Transister พัง และควรให้กระแสไหลผ่านโซลีนอยร์น้อยที่สุดที่ยังทำให้โซลีนอยร์นำกระแส เพื่อไม่ให้เกิดความร้อนที่ตัวโซลีนอยร์มากจนเกินไปจนทำให้โซลีนอยร์พังได้ ซึ่งจากการทดลอง เมื่อใช้ RC = 47 โอห์ม ให้โซลีนอยร์นำกระแสประมาณ 20 นาที ทำให้โซลีนอยร์ไม่ร้อนจนเกินไปซึ่งเป็นค่าที่เหมาะที่สุด และจากการทดลองเมื่อใช้ RC = 47 โอห์ม ทำให้กระแส IC ไหลประมาณ 80 mA

ดังนั้นเราจะได้วงจรขับโซลีนอยร์เพื่อแสดงผลเป็นอักษรเบรลล์ทั้งหมด 30 ชุด และ วงจรอ่านค่าคีย์บอร์ดของไอบีเอ็ม(IBM)ดังนี้



Braille Indicator Circuit

จากวงจรขับโซลีนอยร์เพื่อแสดงผลเป็นอักษรเบรลล์ข้างต้นได้ออกแบบในส่วนของ วงจรอ่านค่าคีย์บอร์ดของไอบีเอ็ม(IBM)ดังรูปที่ เพื่อประยุคให้ใช้งานได้อย่างกว้างขวางและมีประสิทธิภาพมากยิ่งขึ้น



รูปที่3.6 วงจรอ่านค่าคีย์บอร์ดของไอบีเอ็ม(IBM)

เราสามารถนำคีย์บอร์ดของไอบีเอ็มมาประยุคใช้กับโปรเจ็คนี้ได้ โดยต่อฮาร์ดแวร์เพิ่มอีกนิดตามรูปที่3.6

หลักการทำงานของวงจร วงจรนี้จะใช้ไอซีเบอร์ 74164 แปลงระหัสคีย์บอร์ดจากอนุกรมเป็นขนาน แล้วเอาสัญญาณเอาต์พุตของไอซีนี้ไปให้วงจรที่ต้องการอ่านค่าจากคีบอร์ดได้ สำหรับโปรเจ็คนี้วงจรอ่านค่าจากคีบอร์ดก็คือ **ET-PC8255 CARD** นั้นเอง

ขณะที่คีย์ถูกกด เอาต์พุตที่ D0-D7 จะเป็นระหัสสแกนคีย์ และเมื่อปล่อยคีย์ ก็จะได้รหัสคีย์ที่ตรงกับปุ่มที่เรากด

<u>การทดลองและผลการทดลองจะเป็นดังนี้</u>

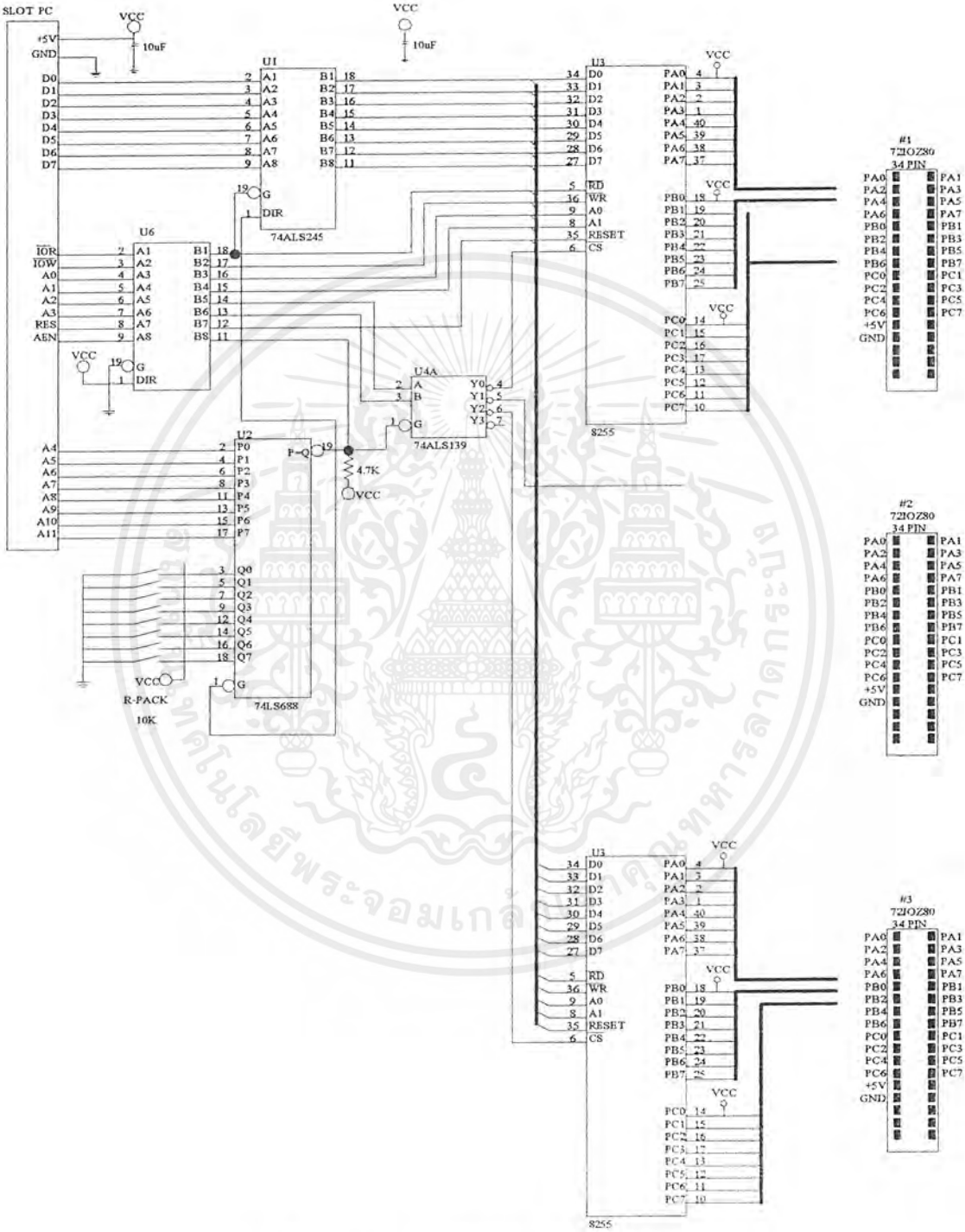
ต่อคีย์บอร์ดของไอบีเอ็ม กับวงจรอ่านค่าจากคีบอร์ดดังรูปที่3.6

<u>เมื่อกดคีย์อักษร</u>	<u>Output D7-D0 (ฐานสิบหก)</u>
a	0X87
b	0X8C
c	0XC8

เมื่อกดคีย์อักษร	Output D7-D0 (ฐานสิบหก)
d	0X88
e	0XC9
f	0XCA
g	0X8D
h	0XCC
i	0X90
j	0X8E
k	0XD0
l	0XD2
m	0XCE
n	0X8C
o	0XD1
p	0XD3
q	0X85
r	0XCB
s	0XC6
t	0X8B
u	0XCF
v	0X8A
w	0XC7
x	0XC8
y	0XCD
z	0X86

หมายเหตุ อักษรตัวเล็กกับอักษรตัวใหญ่ ได้ Output D7-D0 เหมือนกัน(a----> z = A ----> Z) และได้ทำการทดลองป้อนอักษรผ่านทางคีย์บอร์ดโดยไม่ผ่านวงจรอ่านค่าจากคีบอร์ดด้วย(คือต่อกับ Main Board โดยตรงเลย) ซึ่งต้องเขียนโปรแกรมควบคุมคีย์บอร์ดอีกทีหนึ่ง ซึ่งจะได้กล่าวถึงต่อไปในบทที่4

3.4 หลักการของ ET-PC8255 CARD



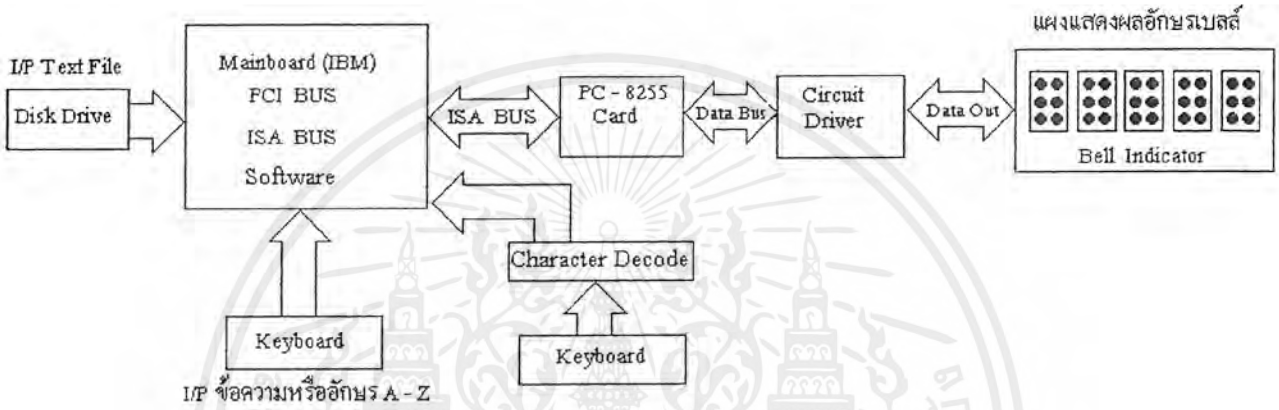
รูปที่3.7 แสดงวงจรดีโค้ดโดยใช้สวิตช์เลือก

จากรูปที่3.7 หลักการทำงานคือ จะมีสวิตช์เพื่อทำการเลือกพอร์ทที่เราต้องการที่ไม่ตรงกับพอร์ทที่กำลังใช้งานอยู่ โดยใช้ไอซี74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุท 2 ชุดที่ถูกส่งเข้ามาทางขา P0-P7(A0-A11) และขา Q0-Q7 ถ้าอินพุททั้ง 2 ชุดนี้เท่ากันแล้ว เอาท์พุท P = Q จะให้เอาท์พุทเป็นลอจิก "0" เพื่อไป Trig หรือจ่ายให้แก่ไอซี 74LS139 เพื่อทำการถอดรหัสว่าจะเลือกให้ไอซี 8255 ตัวใดทำงานซึ่งขึ้นอยู่กับแอดเดรส A2 และ A3

หลักการทำงานของเครื่องทั้งหมด คือ เราจะเขียนโปรแกรมเพื่อควบคุม PC-8255 CARD ส่งข้อมูลไปยังแผงอักษรเบรลล์ตามที่คนตาบอดพิมพ์อักษรหรือข้อความใดๆก็จะไปแสดง(display)ที่แผงอักษรเบรลล์ตามมาตรฐานของอักษรเบรลล์และแผงอักษรเบรลล์สามารถอ่านหรือแสดงText File จากFloppy Disk ได้ ซึ่งจะกล่าวถึงต่อไปในบทที่4

บทที่4
ผลการทดลอง

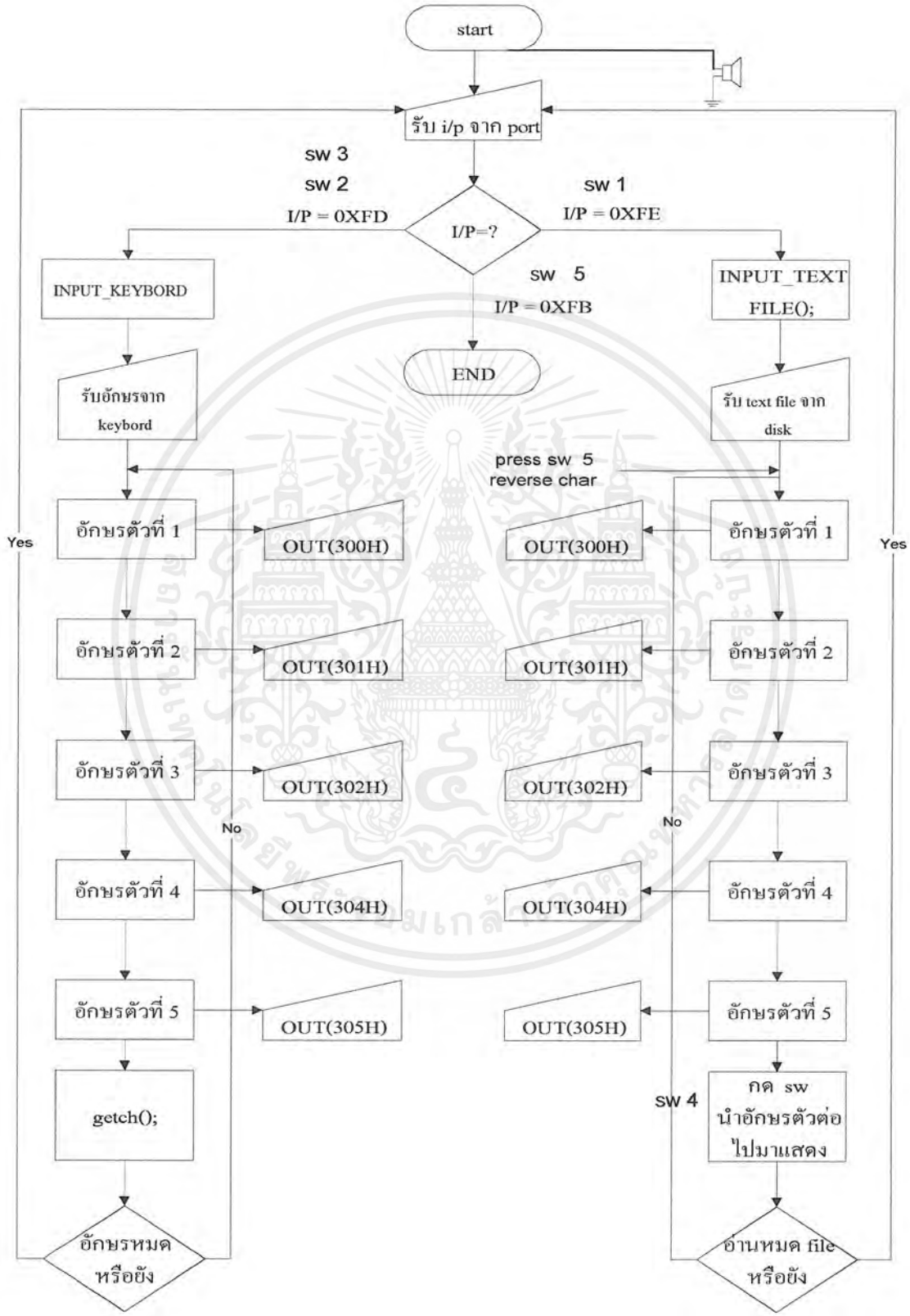
4.1 ผลการทดสอบการทำงาน Hardware ร่วมกับ Software



รูปที่4.1 Block Diagram หลักการทำงานทั้งหมดของโครงงาน

ต่อวงจรตาม Block Diagram เพื่อทดสอบการทำงานของ Hardware (ในส่วนของ PC-8255 Card และวงจรขับ Braille Indicator) ร่วมกับ Software ว่าทำงานถูกต้องหรือไม่ โดยใช้โปรแกรมต่อไปนี้ทดสอบ

FLOWCHART แสดงการทำงาน Braille Indicator



จาก Flowchart แสดงหลักการทำงานของเครื่อง Braille Indicator ดังนี้

เมื่อคนตาบอดเริ่มเปิดเครื่อง(Switch On) และรอจนกว่ามีเสียงจากลำโพงดังขึ้นซึ่งเป็นการเตือนว่าขณะนี้ผู้ใช้สามารถกดสวิทช์เพื่อเลือก Option ที่ต้องการOption มีอยู่ 2 Option คือ Input_Text File();หรือ Input_Keyboard) โดยในการกดสวิทช์นั้นก็คือการรับ Input จากพอร์ตนั่นเองและขณะนี้เสียงของลำโพงจะดังไปเรื่อยๆ จนกว่าจะมีการกดสวิทช์เพื่อเป็นการรับ Input จากพอร์ต(เ,อร์ต 8255)

หากผู้ใช้กด SW1 (I/P = 0XFE) หมายความว่าจะเป็น Option ของการอ่านไฟล์ข้อมูลหรืออักษรจาก Floppy Disk ซึ่งในขั้นตอนนี้ผู้ใช้จะต้องใส่แผ่นข้อมูลให้กับเครื่อง (ใส่แผ่น Floppy Disk ใน Drive A) ซึ่งข้อมูลหรืออักษรทั้งหมดที่อยู่ใน Floppy Disk จะไปแสดงเป็นอักษรเบรลล์ที่ชุด Braille Indicator ซึ่งจะแสดงออกมาทีละ 5 ตัวอักษรและหากต้องการดูข้อมูลตัวต่อไปหรือชุดต่อไปก็ให้ผู้ใช้กด SW4 เพื่อนำข้อมูลหรือตัวอักษรอีก 5 ตัวมาแสดงต่อ และกด SW4 ไปเรื่อยๆก็จะแสดงข้อมูลทีละ5 ตัวจนกระทั่งหมดข้อมูลก็จะออกจาก Optionการรับ Input_Text File(); โดยจะมีเสียงเตือนดังขึ้นมาอีกครั้งเพื่อให้คนตาบอดรับทราบว่าหมดแฟ้มข้อมูลแล้วซึ่งวนกลับมาที่เดิมเพื่อให้ผู้ใช้กดสวิทช์รับInput ว่าจะเลือก Option ใดต่อไป

หากกด Sw2 คราวนี้ก็คือให้ผู้ใช้ฝึกใช้คีย์บอร์ดนั่นเองคือให้ผู้ใช้ป้อนอักษรจนกระทั่งผู้ใช้กด Enter ข้อมูลก็จะมาแสดงครั้งละ5 ตัวเหมือนเดิม ซึ่งในขั้นตอนนี้ทำให้ผู้ใช้สามารถ Check ได้ว่าเขาพิมพ์ข้อความอะไรลงไปและพิมพ์ได้ถูกต้องตามที่พิมพ์หรือยัง

ขั้นตอนการทดสอบ Software ร่วม Hardware

1. เขียนโปรแกรมภาษาC เพื่อควบคุมการทำงานของ Hardwareโดย Safe เป็นชื่อBraille.c ทำการ Compile ได้เป็น Braille.obj แล้วทำการ Run Linker เป็น Braille.EXE ซึ่งเราก็นำ Braill.EXE มาใช้งาน และทำการ Copy Braille.EXE ลง Harddisk

2.เขียน File Autoexec.bat ให้ทำการ Run Braille.EXE โดยอัตโนมัติ ซึ่งนั่นก็คือเป็นการทำงานด้วยตัวมันเอง (Stand Alone) ดังนี้ C:\Braill.EXE

3.ทำการต่อวงจรตาม Block Diagram ดังรูปที่4.1

4.เปิดสวิทช์เครื่อง Braille Indicator (IBM PC) เครื่องก็จะทำการ Set Up ตัวมันเองโดยอัตโนมัติ

วิธีบูตเครื่อง

เมื่อเริ่มกด Power On PC จะมีกระบวนการ Setup ต่างๆจนขั้นสุดท้าย PC จะอ่านไฟล์ระบบ จาก Harddisk หรือ Floppy disk เพื่อเข้าสู่ระบบ DOS กระบวนการนี้เรียกว่า "บูตเครื่อง"," บูตระบบ","บูต"

เกิดอะไรขึ้นเมื่อเข้าสู่ระบบ

เมื่อเข้าสู่ระบบ DOS แล้ว ระบบปฏิบัติการ DOS จะควบคุมการทำงานทั้งหมดของ PC DOS จะทำการโหลดไฟล์ COMMAND.COM ซึ่งเป็นโปรแกรมหัวใจของ DOS เข้าสู่หน่วยความจำ และจะทำการ Excute (แปร, อ่าน) ไฟล์พิเศษ 2ไฟล์คือ CONFIG.SYS และ AUTOEXEC.BAT ซึ่งไฟล์ทั้ง2 นี้ เป็นตัวกำหนดควบคุมสภาพแวดล้อมการทำงานของ DOS
และไฟล์ Autoexec.bat ก็จะทำการ Run Braill.EXE โดยอัตโนมัติเพื่อควบคุมการทำงานของ Hardware ทั้งหมด ในขั้นตอนนี้ก็จะมีเสียงเตือนจากลำโพงดังขึ้นมาเพื่อบอกให้คนตาบอดทราบว่าต้องการเลือก Option ใดซึ่งมี2 Optionดังนี้

1. เมื่อกด SW1-→ ก็เป็น Input_Text File();
2. เมื่อกด SW2--> ก็เป็น Input_Keyboard
5. นำแฟ้มข้อมูลซึ่งมีข้อมูลอยู่ใน Floppy disk ดังนี้

Atthapon Molito

ทำการ Safe ข้อมูลดังกล่าวชื่อ Braille.txt
<u>หมายเหตุ</u> ต้องตั้งชื่อแฟ้มข้อมูล เป็น Braille.txt
6. จากขั้นตอนที่ 4 เมื่อมีเสียงเตือนดังจากลำโพงอยู่ ให้ใส่แผ่น Floppydisk ใน Drive A กด SW1 เครื่อง Braille Indicatorก็จะทำการอ่านข้อมูลในแผ่น Floppydisk ออกไปแสดงผลบนแผงแสดงผลอักษร Braille ครั้งละ5 ตัวอักษรโดยที่ในการแสดงผลชุดต่อไปให้กด SW4 จนกระทั่งหมดแฟ้มข้อมูลก็จะมีเสียงเตือนขึ้นมาอีกครั้ง

ผลการทดลองจากการอ่านFile ข้อมูลชื่อ Braille.txt ไปแสดงผลที่แผงอักษรเบรลล์ เป็นดังนี้

อ่านว่า Atthapon Molito (อรรถพล โมลิโต)

โดย

หมายถึง ไม่มีตัวอักษร เป็นช่องว่างเฉยๆ

7. กด SW2 คราวนี้ก็คือให้ผู้ใช้ฝึกใช้คีย์บอร์ดนั่นเองคือให้ผู้ใช้ป้อนอักษรจนกระทั่งผู้ใช้กด Enter ข้อมูลก็จะมาแสดงครั้งละ5 ตัวเหมือนเดิม ซึ่งในขั้นตอนนี้ทำให้ผู้ใช้สามารถ Check ได้ว่าเขาพิมพ์ข้อความอะไรลงไปและพิมพ์ได้ถูกต้องตามที่พิมพ์หรือยัง

ผลการทดลองในการป้อนอักษรทาง Keyboard เป็นดังนี้

เช่น ป้อนข้อความดังนี้

Atthapon Molito แล้วกด Enter ผลที่แผงอักษรเบรลล์ เป็นดังนี้

โปรแกรมที่ใช้ทดลอง เพื่อทดสอบ Braille Indicator(แผงแสดงผลอักษรเบรลล์)

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<string.h>

#define max 2000
#define PORT1 0x300
#define PORT2 0x304
#define PORT3 0x308

void Input_text();
void Input_keyboard();
void Input_keycode();

void main(void)
{
 int ch;
 clrscr();

 /* CONTROL WORD */
 outportb(PORT1+3,0x80);
 outportb(PORT2+3,0x89);
 outportb(PORT3+3,0x9B);
 /* CONTROL WORD */

 /* SET PORT */
 outportb(PORT1+0,0x00);
 outportb(PORT1+1,0x00);
```

```
 outportb(PORT1+2,0x00);
 outportb(PORT2+0,0x00);
 outportb(PORT2+1,0x00);
 outportb(PORT2+2,0x00);
 outportb(PORT3+0,0x00);
 /* SET PORT */

 do
 {
   ch=inportb(PORT3+0);  /* Selec SW option */
   printf("INPUT DATA SW ! = %X\n\n",ch);

          if(ch==0xFE)
           Input_text();

           else
              if(ch==0xFD)
               Input_keyboard();

                 else

                    if(ch ==0xFB)
                    Input_keydecode();

                          printf("\007");

 }while(ch!=0xF7);
 printf("\007");
}
```

```
 void Input_text()
{
 FILE *fp;
 char ch,intext[max];
 char data1[] = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O',
                 'P','Q','R','S','T','U','V','W','X','Y','Z',
                 'a','b','c','d','e','f','g','h','i','j','k','l','m','n',
                 'o','p','q','r','s','t','u','v','w','x','y','z'};

char output[] ={0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                 0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                 0x3D,0x1D,
                 0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                 0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                 0x3D,0x1D};

int str,inchk=0x00,i,j;
 i=0;
clrscr();

 if((fp=fopen("a:\\bell.txt ","r")) ==NULL)
{
 printf("Connot open file\n");
 printf("\007");
 exit(0);

}

   ch=getc(fp);
   intext[i]=ch;
```

```
        while(feof(fp)==0)
  {
        printf("%c",ch);
           ch=getc(fp);
           intext[i+1]=ch;
           i+=1;
  }

 fclose(fp);

str=strlen(intext);

printf("\n");
printf("Lenght of string = %d\n",str);


 for(i=0;i<=str;i++)
{
        do
  {  delay(500);
           inchk=inportb(PORT3+0);
           printf(" INPUT CHECK = %X\n",inchk);
           for(j=0;j<=51;j++)
        {
           if( (intext[i]-data1[j])==0)
           outportb(PORT1+0,output[j]);

        }
           for(j=0;j<=51;j++)
```

```
        {
         if( (intext[i+1]-data1[j])==0)
         outportb(PORT1+1,output[j]);
        }
         for(j=0;j<=51;j++)
        {
         if( (intext[i+2]-data1[j])==0)
         outportb(PORT1+2,output[j]);
        }
         for(j=0;j<=51;j++)
        {
         if( (intext[i+3]-data1[j])==0)
         outportb(PORT2+0,output[j]);
        }
         for(j=0;j<=51;j++)
        {
         if( (intext[i+4]-data1[j])==0)
         outportb(PORT2+1,output[j]);
        }

    }while(inchk!=0xFB);

        delay(10000);
       i=i+4;
         outportb(PORT1+0,0x00);
         outportb(PORT1+1,0x00);
         outportb(PORT1+2,0x00);
         outportb(PORT2+0,0x00);
         outportb(PORT2+1,0x00);
}
```

```
}

  void Input_keyboard()
{
 char data1[] = {'A','B','C','L','E','F','G','H','I','J','K','L','M','N','O',
                 'P','Q','R','S','T','U','V','W','X','Y','Z',
                 'a','b','c','d','e','f','g','h','i','j','k','l','m','n',
                 'o','p','q','r','s','t','u','v','w','x','y','z'};

char output[] ={0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                0x3D,0x1D,
                0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                0x3D,0x1D};

char in_keyboard[max];
int str,inchk,i,j;
          i=0;
          printf("Enter character:");
          gets(in_keyboard);
          printf("%s\n",in_keyboard);
          str = strlen(in_keyboard);
          printf("%d\n",str);


   for(i=0;i<=str-1;i++)
{
```

```
 for(j=0;j<=51;j++)
{
 if( (in_keyboard[i]-data1[j])==0)
 outportb(PORT1+0,output[j]);
 }
 for(j=0;j<=51;j++)
{
 if( (in_keyboard[i+1]-data1[j])==0)
 outportb(PORT1+1,output[j]);
}
 for(j=0;j<=51;j++)
{
 if( (in_keyboard[i+2]-data1[j])==0)
 outportb(PORT1+2,output[j]);
}
 for(j=0;j<=51;j++)
{
 if( (in_keyboard[i+3]-data1[j])==0)
 outportb(PORT2+0,output[j]);
}
 for(j=0;j<=51;j++)
{
 if( (in_keyboard[i+4]-data1[j])==0)
 outportb(PORT2+1,output[j]);
}

printf("!** PRESS ANY KEY TO SHOW CHARACTER **!\n");
 getch();
i=i+4;
```

```
            outportb(PORT1+0,0x00);
            outportb(PORT1+1,0x00);
            outportb(PORT1+2,0x00);
            outportb(PORT2+0,0x00);
            outportb(PORT2+1,0x00);
  }

}

  void Input_keydecode()
{
 int ch;
 clrscr();

 do
 {
   ch=inportb(PORT2+2);  /* Press SW with out from function Show_CHARACTER */
   printf("Input data = %X\n\n",ch);

   if(ch==0x87)
   {
          printf(" DISPLAY OF CHARACTER **** A ****\n");
          outportb(PORT1+0,0x04);

   }
   else
    if(ch==0x8C)
   {
          printf(" DISPLAY OF CHARACTER **** B *****\n");
         outportb(PORT1+0,0x06);
```

```
}
 else
  if(ch==0xC8)
{
        printf(" DISPLAY OF CHARACTER **** C ****\n");
       outportb(PORT1+0,0x024);
}
 else
  if(ch==0x88)
{
        printf(" DISPLAY OF CHARACTER **** D ****\n");
       outportb(PORT1+0,0x34);
}
 else
  if(ch==0xC9)
{
        printf(" DISPLAY OF CHARACTER **** E ****\n");
       outportb(PORT1+0,0x14);
}
 else
  if(ch==0xCA)
{
        printf(" DISPLAY OF CHARACTER **** F ****\n");
       outportb(PORT1+0,0x26);
}
 else
  if(ch==0x8D)
{
        printf(" DISPLAY OF CHARACTER **** G ****\n");
       outportb(PORT1+0,0x36);
```

```
}
 else
 if(ch==0xCC)
{
        printf(" DISPLAY OF CHARACTER **** H ****\n");
         outportb(PORT1+0,0x16);
}
 else
 if(ch==0x90)
{
        printf(" DISPLAY OF CHARACTER **** I ****\n");
       outportb(PORT1+0,0x22);
}
 else
 if(ch==0x8E)
{
        printf(" DISPLAY OF CHARACTER **** J ****\n");
       outportb(PORT1+0,0x32);

}
 else
 if(ch==0xD0)
{
        printf(" DISPLAY OF CHARACTER **** K ****\n");
       outportb(PORT1+0,0x05);
}
 else
 if(ch==0xD2)
{
        printf(" DISPLAY OF CHARACTER **** L ****\n");
```

```
        outportb(PORT1+0,0x07);
}
 else
  if(ch==0xCE)
{
        printf(" DISPLAY OF CHARACTER **** M ****\n");
       outportb(PORT1+0,0x25);
}
 else
  if(ch==0x8C)
{
        printf(" DISPLAY OF CHARACTER **** N ****\n");
       outportb(PORT1+0,0x35);
}
 else
 if(ch==0xD1)
{
        printf(" DISPLAY OF CHARACTER **** O ****\n");
       outportb(PORT1+0,0x015);
}
 else
  if(ch==0xD3)
{
        printf(" DISPLAY OF CHARACTER **** P ****\n");
       outportb(PORT1+0,0x27);
}
 else
  if(ch==0x85)
{
        printf(" DISPLAY OF CHARACTER **** Q ****\n");
```

```
        outportb(PORT1+0,0x37);
 }
 else
  if(ch==0xCB)
 {
        printf(" DISPLAY OF CHARACTER **** R ****\n");
       outportb(PORT1+0,0x17);
 }
 else
  if(ch==0xC6)
 {
        printf(" DISPLAY OF CHARACTER **** S ****\n");
       outportb(PORT1+0,0x23);
 }
 else
  if(ch==0x8B)
 {
        printf(" DISPLAY OF CHARACTER **** T ****\n");
       outportb(PORT1+0,0x33);
 }
 else
  if(ch==0xCF)
 {
        printf(" DISPLAY OF CHARACTER **** U ****\n");
      outportb(PORT1+0,0x0D);
 }
 else
  if(ch==0x8A)
 {
        printf(" DISPLAY OF CHARACTER **** V ****\n");
```

```
        outportb(PORT1+0,0x0F);
    }
    else
     if(ch==0xC7)
    {
         printf(" DISPLAY OF CHARACTER **** W ****\n");
        outportb(PORT1+0,0x3A);
    }
    else
    if(ch==0xC8)
    {
         printf(" DISPLAY OF CHARACTER **** X ****\n");
        outportb(PORT1+0,0x2D);
    }
    else
     if(ch==0xCD)
    {
         printf(" DISPLAY OF CHARACTER **** Y ****\n");
        outportb(PORT1+0,0x3D);
    }
    else
     if(ch==0x86)
    {
         printf(" DISPLAY OF CHARACTER **** Z ****\n");
        outportb(PORT1+0,0x1D);
    }
    else

{
    outportb(PORT1+0,0x00);
```

```
    printf(" You not Press character A ====Z\n");
    printf("   *** NO CHARACTER ! ****\n");
    printf("    NEW PRESS CHARACTER\n\n");
    delay(10000);
    }

  }while(ch!=0xF7);
}
```

บทที่5
บทสรุปและวิจารณ์

บทสรุป

โครงงานนี้เป็นการสร้างเครื่องแสดงผลออกมาเป็นอักษรเบรลล์สำหรับคนพิการทางสายตาแทนการอ่านหนังสือของคนปกติ

จุดมุ่งหมายเพื่อให้คนตาบอดได้รับรู้ข่าวสารต่างๆ จากเครื่องที่สร้างขึ้นมานี้ ซึ่งโครงงานนี้จะมีสอง option ที่ใช้ช่วยเหลือคนพิการสำหรับคนตาบอด คือให้คนตาบอดฝึกใช้ keyboard (IBM PC) จากเครื่อง Braille Indicator ได้ และ พิมพ์ ข่าวสารได้ เพราะว่าจะมี hardware เช็คว่าคนตาบอด พิมพ์ข้อความหรือตัวอักษรถูกต้องตามที่เขาพิมพ์หรือเปล่า และช่วยให้คนพิการสำหรับคนตาบอดสามารถรับรู้ข่าวสารจำพวก Text File ได้โดยแปลง/อ่าน IBM Text files ออกแสดงผลที่ Panel อักษรเบรลล์

ปัญหา

โครงงานนี้มีปัญหาที่หาวัสดุอุปกรณ์ในการทำตัวอักษรนูน (Braille Alphabet) ได้ยากมากที่จะให้เหมาะสม สะดวก สวยงาม และยืดหยุ่นต่อการใช้งาน ซึ่งในโครงงานนี้จะใช้ โซลีนอยร์ในการทำตัวอักษรนูน(Braille) จึงทำให้เซล์ลใหญ่เกินไป ดังนั้นจะทำให้คนพิการสำหรับคนตาบอดฝึกใช้เครื่องหรือทำความคุ้นเคยกับมันมากขึ้นเพื่อให้สามารถอ่านข่าวสารได้เร็วขึ้น

แนวทางการพัฒนาต่อไป

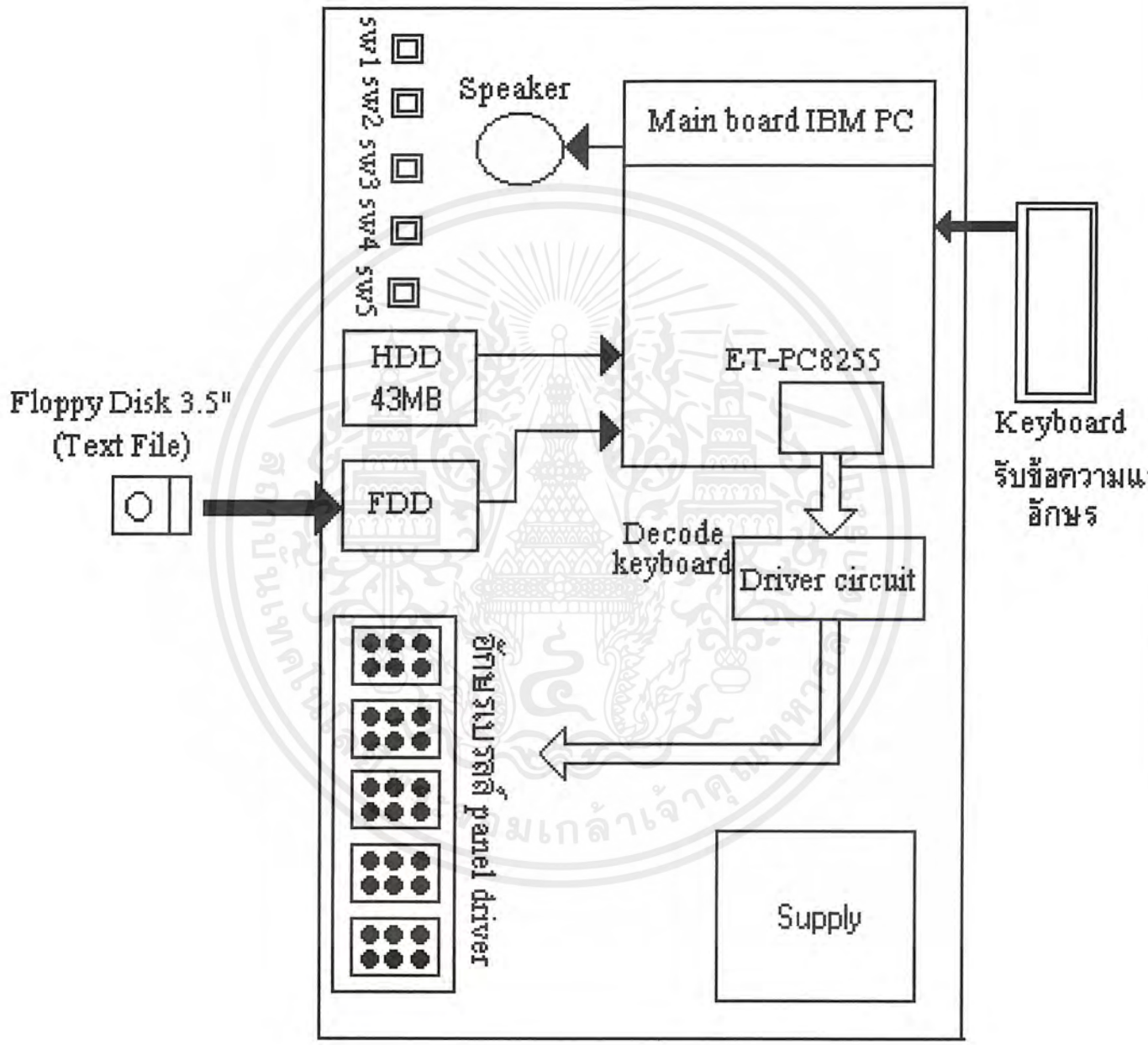
พัฒนาต่อไปเป็นเครื่องที่สามารถอ่านข้อมูลไฟล์อักษรให้ออกมาเป็นเสียงของตัวอักษรแต่ละตัวหรือออกมาเป็นเสียงพูดเป็นประโยคโดยตรง โดยไม่จำเป็นต้องใช้ส่วนของอักษรเบรลล์ งานส่วนนี้จะหนักในส่วนของ Soft ware และ Data-base

บรรณานุกรม

1. ธานินทร์ ถาวรศาสนวงศ์, ทินกร ดุ๊ก : การอินเทอร์เฟส IBM/PC
   B.Eng., (KMIT – Ladkrabang)
2. ดร.วิทยา เรืองพรวิสุทธิ์ : คู่มือโปรแกรมภาษา C สำหรับผู้เริ่มต้น
3. บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), กรุงเทพฯ, 2538
   เข้าใจ/สร้าง/เล่น ไมโครโปรเซสเซอร์ 1

ภาคผนวก

Braille Indicator



แสดงรูปเครื่องแสดงผลอักษรเบรลล์

คู่มือการใช้งานเครื่องแสดงผลอักษรเบรลล์(Braille Indicator)

1. เปิดเครื่องแล้วรอจนกว่ามีเสียงเตือนจาก Speaker ดังขึ้น
2. เลือก Switch ที่อยู่บนเครื่อง(SW1,SW2,SW3 เป็นอันดับแรก)
3. หากเลือก SW1 นั้นคือการอ่านข่าวสารหรือข้อมูลที่อยู่ใน Floppy Disk (ต้องใส่แผ่น Floppy Disk ด้วย) ออกมาเป็นอักษรเบรลล์

หมายเหตุ แฟ้มข้อมูลที่ต้องการจะอ่านออกมาเป็นอักษรเบรลล์ ให้ตั้งชื่อแฟ้มเป็น **braille.txt**

4. กด SW4 เพื่ออ่านข้อมูลตัวต่อไป หากต้องการกลับไปดูข้อมูลก่อนหน้านี้ กด SW5 แล้วกลับมากด SW4อีกเพื่ออ่านข้อมูลตัวต่อไปจนกระทั่งอ่านจนหมดแฟ้มก็จะมีเสียงเตือนเพื่อให้เลือก Switch(SW1,SW2,SW3)อีกครั้ง
5. หากเลือก SW2 นั้นคือการฝึกพิมพ์ข้อความหรือตัวอักษรออกมาเป็นอักษรเบรลล์ วิธีใช้ก็คือป้อนข้อความหรือตัวอักษรจำนวนตามต้องการ จนกระทั่งกด Enter มันก็จะอ่านออกมาเป็นอักษรเบรลล์ครั้งละ 5 ตัว ต้องการดูข้อมูลตัวต่อไปกด ที่ SW4 หากต้องการกลับไปดูอักษรที่เราพิมพ์ที่อยู่ก่อนหน้านี้ ให้กด SW5 แล้วกด SW4 อีกเมื่อต้องการอ่านข้อมูลตัวต่อไป จนกระทั่งหมดข้อความหรือตัวอักษรที่เราพิมพ์ไปก็จะมีเสียงเตือนเพื่อให้เลือกSwitch (SW1,SW2,SW3)อีกครั้ง

6. หากเลือก SW3 นั้นคือการฝึกเพื่อให้คนตาบอดรู้ว่าอักษรแต่ละตัวอยู่ที่ตำแหน่งใดของ Keyboard พิมพ์อักษรทีละตัว โดยกดอักษร 1 ตัว แสดงอักษรเบรลล์ 1ตัว ปรากฏในเซลล์ที่ 1 ของ panel driver ดังรูปข้างต้นหากต้องการออกจาก option นี้กด ESC ก็จะมีเสียงเตือนเพื่อให้เลือก Switch (SW1,SW2,SW3)อีกครั้ง

7. กด SW5 เมื่อต้องการเลิกจาก Project
8. ปิดเครื่อง

โปรแกรมที่ใช้งานจริงในการควบคุมเครื่องแสดงผลอักษรเบรลล์(Braille Indicator)

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<string.h>

#define MAX 5000
#define PORT1 0x300
#define PORT2 0x304
#define PORT3 0x308

void Input_text(FILE *fp,char data1[],char output[]);
void Input_keyboard(char data1[],char output[]);
void Input_decodekey();


/* ***** START FUNCTION MAIN */
void main(void)
{
 FILE *fp;
 int ch;
  char data1[] = {
                  'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O',
                  'P','Q','R','S','T','U','V','W','X','Y','Z',
                  'a','b','c','d','e','f','g','h','i','j','k','l','m','n',
                  'o','p','q','r','s','t','u','v','w','x','y','z',
                 '?',';','.','(',')',':',',','!','-','#','" ','1','2','3','4',
                 '5','6','7','8','9','0'};
```

```
  char output[] ={
                  0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                  0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                  0x3D,0x1D,
                  0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32,0x05,0x07,
                  0x25,0x35,0x15,0x27,0x37,0x17,0x23,0x33,0x0D,0x0F,0x3A,0x2D,
                  0x3D,0x1D,
                  0x0B,0x03,0x1A,0x1B,0x1B,0x12,0x02,0x13,0x09,0x39,0x08,
                  0x04,0x06,0x24,0x34,0x14,0x26,0x36,0x16,0x22,0x32};
  clrscr();

  /* CONTROL WORD */
  outportb(PORT1+3,0x80);
  outportb(PORT2+3,0x89);
  outportb(PORT3+3,0x9B);
  /* CONTROL WORD */

  /* SET PORT */
  outportb(PORT1+0,0x00);
  outportb(PORT1+1,0x00);
  outportb(PORT1+2,0x00);
  outportb(PORT2+0,0x00);
  outportb(PORT2+1,0x00);
  outportb(PORT2+2,0x00);
  outportb(PORT3+0,0x00);
  /* SET PORT */

loop:

   printf(" ===> PREASE SELECTER SWITCH OPTION <===\n");
```

```
printf(" ====>\n");
printf("     SW1(0xFE) : INPUT TEXT FILE FROM FLOPPY DISK\n");
printf("     SW2(0xFD) : INPUT CHARACTER FROM KEYBOARD\n");
printf("     SW3(0xFB) : INPUT 1 DISPLAY CHARACTER \n");
printf("     SW4(0xF7) : BRAILLE ALPHABET CONTINUOUS\n");
printf("           ESC : OUT OF Input_decodekey FUNCTION\n");
printf("     SW5(0xDF) : BACK CHARACTER  AND OUT OF PROJECT\n");
printf("\n");

do
{
  ch=inportb(PORT3+0);                  /* Selec SW option */
  printf("INPUT DATA FROM SW IS  = %X\n\n",ch);

  if(ch==0xFE)                          /* PRESS SW1 */
  {
       if((fp=fopen("a:\\braille.txt ","r")) ==NULL)
      {
       printf("Connot open file\n");
       printf("\007");
       clrscr();
       goto loop;
       }

        Input_text(fp,data1,output);
  }

        else
            if(ch==0xFD)                /* PRESS SW2 */
              Input_keyboard(data1,output);
```

```
                else
                    if(ch==0xFB)            /* PRESS SW3 */
                     Input_decodekey();

                else
                    printf("\007");     /* SOUND REMIND TO PRESS SW */

  }while(ch!=0xDF);                                    /* PRESS SW6 */

}
  /*  ***** END FUNCTION MAIN ***** */


 void Input_text(FILE *fp,char data1[],char output[])
{
char ch,intext[MAX];
int str=0,inchk,reinchk,i,j;
clrscr();

  i=0;
   ch=getc(fp);
   intext[i]=ch;

       while(feof(fp)==0)
   {
       printf("%c",ch);
          ch=getc(fp);
          intext[i+1]=ch;
          i+=1;
```

```
        str = str+1;
  }

 fclose(fp);   /*  close file  */

printf("\n");
printf("Lenght of string = %d\n",str);


 for(i=0;i<=str-1;i++)
{
        printf("\007");

      do
  {
        delay(500);
        inchk=inportb(PORT3+0);          /* PRESS SW4 */
        printf(" INPUT CHECK CHAR CONTINUOUS (press sw4) = %X\n",inchk);

        for(j=0;j<=72;j++)
       {
        if( (intext[i]-data1[j])==0)
        outportb(PORT1+0,output[j]);
       }
        for(j=0;j<=72;j++)
       {
        if( (intext[i+1]-data1[j])==0)
        outportb(PORT1+1,output[j]);
       }
        for(j=0;j<=72;j++)
```

```
    {
     if( (intext[i+2]-data1[j])==0)
     outportb(PORT1+2,output[j]);
    }
     for(j=0;j<=72;j++)
    {
     if( (intext[i+3]-data1[j])==0)
     outportb(PORT2+0,output[j]);
    }
     for(j=0;j<=72;j++)
    {
     if( (intext[i+4]-data1[j])==0)
     outportb(PORT2+1,output[j]);
    }

       reinchk=inportb(PORT3+0);  /* PRESS SW6 */
       printf("BACK CHARACTER (press sw6) = %X\n",reinchk);

   if(reinchk==0xDF)
{
     outportb(PORT1+0,0x00);
     outportb(PORT1+1,0x00);
     outportb(PORT1+2,0x00);
     outportb(PORT2+0,0x00);
     outportb(PORT2+1,0x00);

      do
      {
            if(i < 5 )
        {
```

```
outportb(PORT1+0,0x00);
        outportb(PORT1+1,0x00);
        outportb(PORT1+2,0x00);
        outportb(PORT2+0,0x00);
        outportb(PORT2+1,0x00);
  }
  printf("\007");
       for(j=0;j<=72;j++)
  {
       if( (intext[i-5]-data1[j])==0)
       outportb(PORT1+0,output[j]);
  }
      for(j=0;j<=72;j++)
  {
       if( (intext[(i+1)-5]-data1[j])==0)
       outportb(PORT1+1,output[j]);
  }
      for(j=0;j<=72;j++)
  {
       if( (intext[(i+2)-5]-data1[j])==0)
       outportb(PORT1+2,output[j]);
  }
  for(j=0;j<=72;j++)
 {
      if( (intext[(i+3)-5]-data1[j])==0)
      outportb(PORT2+0,output[j]);
 }
  for(j=0;j<=72;j++)
 {
      if( (intext[(i+4)-5]-data1[j])==0)
```

```
                outportb(PORT2+1,output[j]);
            }

          }while(reinchk!=0xDF);  /* CHECK PRESSION SW6 */
                delay(10000);

        }                   /* END OF IF */
    }while(inchk!=0xF7);        /* CHECK PRESSION SW4 */
        delay(10000);
         outportb(PORT1+0,0x00);
         outportb(PORT1+1,0x00);
         outportb(PORT1+2,0x00);
         outportb(PORT2+0,0x00);
         outportb(PORT2+1,0x00);
  }                 /* END OF FOR */

}                   /* END OF INPUT_TEXT FUNCTION */


  void Input_keyboard(char data1[],char output[])
{
 char in_keyboard[MAX];
 int str,st=0,inchk,reinchk,i,j;
 clrscr();

          i=0;
           printf("Enter character:");
           gets(in_keyboard);
           printf("%s\n",in_keyboard);
           str = strlen(in_keyboard);
```

```
            printf("%d\n",str);


printf("\007");

 for(i=0;i<=str-1;i++)
{

      do
 {
       delay(500);
       inchk=inportb(PORT3+0);                /* PRESS SW4 */
       printf("CHARACTER CONTINUOUS (press sw4) = %X\n",inchk);
       st++;
       for(j=0;j<=72;j++)
       {
        if( (in_keyboard[i]-data1[j])==0)
        outportb(PORT1+0,output[j]);
         if(st>str)
           outportb(PORT1+0,0x00);
       }
       st++;
       for(j=0;j<=72;j++)
      {
       if( (in_keyboard[i+1]-data1[j])==0)
       outportb(PORT1+1,output[j]);
         if(st>str)
           outportb(PORT1+1,0x00);
      }
       st++;
```

```
 for(j=0;j<=72;j++)
{
 if( (in_keyboard[i+2]-data1[j])==0)
 outportb(PORT1+2,output[j]);
  if(st>str)
    outportb(PORT1+2,0x00);
}
 st++;
 for(j=0;j<=72;j++)
{
 if( (in_keyboard[i+3]-data1[j])==0)
 outportb(PORT2+0,output[j]);
  if(st>str)
    outportb(PORT2+0,0x00);
}
 st++;
 for(j=0;j<=72;j++)
{
 if( (in_keyboard[i+4]-data1[j])==0)
 outportb(PORT2+1,output[j]);
  if(st>str)
    outportb(PORT2+1,0x00);
}

 st = st-5; /* VERY INPORTANT */

  reinchk=inportb(PORT3+0);                    /* PRESS SW6 */
 printf("BACK CHARACTER (press sw6) = %X\n",reinchk);

if(reinchk==0xDF)
```

```
{
    outportb(PORT1+0,0x00);
    outportb(PORT1+1,0x00);
    outportb(PORT1+2,0x00);
    outportb(PORT2+0,0x00);
    outportb(PORT2+1,0x00);

  do
  {
        if(i<5)
        {
   outportb(PORT1+0,0x00);
         outportb(PORT1+1,0x00);
         outportb(PORT1+2,0x00);
         outportb(PORT2+0,0x00);
         outportb(PORT2+1,0x00);
        }
       printf("\007");

        for(j=0;j<=72;j++)
   {
       if( (in_keyboard[i-5]-data1[j])==0)
       outportb(PORT1+0,output[j]);
   }
      for(j=0;j<=72;j++)
   {
       if( (in_keyboard[(i+1)-5]-data1[j])==0)
       outportb(PORT1+1,output[j]);
   }
      for(j=0;j<=72;j++)
```

```
        {
            if( (in_keyboard[(i+2)-5]-data1[j])==0)
            outportb(PORT1+2,output[j]);
        }
        for(j=0;j<=72;j++)
        {
            if( (in_keyboard[(i+3)-5]-data1[j])==0)
            outportb(PORT2+0,output[j]);
        }
        for(j=0;j<=72;j++)
        {
            if( (in_keyboard[(i+4)-5]-data1[j])==0)
            outportb(PORT2+1,output[j]);
        }

    }while(reinchk!=0xDF);          /* CHECK PRESSION SW6 */
            i=i-5;
            delay(100);

}                                   /* END OF IF */

} while(inchk!=0xF7);               /* CHECK PRESSION SW4 */
    printf("\007");
    delay(100);
    i=i+4;
    st=st+5;

  outportb(PORT1+0,0x00);
  outportb(PORT1+1,0x00);
  outportb(PORT1+2,0x00);
```

```
    outportb(PORT2+0,0x00);
    outportb(PORT2+1,0x00);
  }                                          /* EN OF FOR */
}

 void Input_decodekey()
 {
 int ch,outdekey;
 clrscr();


 do
 {
   ch = tolower(getch());


          if(ch==':')
   {
          printf(" DISPLAY OF CHARACTER **** : ****\n");
          outportb(PORT1+0,0x12);
          printf("\007");
   }
    else

         if(ch==',')
   {
          printf(" DISPLAY OF CHARACTER **** , ****\n");
          outportb(PORT1+0,0x02);
          printf("\007");
   }
```

```
else

 if(ch=='-')
{
        printf(" DISPLAY OF CHARACTER **** - ****\n");
        outportb(PORT1+0,0x09);
        printf("\007");
}
 else

 if(ch=='!')
{
        printf(" DISPLAY OF CHARACTER **** ! ****\n");
        outportb(PORT1+0,0x13);
        printf("\007");
}
 else

 if(ch=='#')
{
        printf(" DISPLAY OF CHARACTER **** # ****\n");
        outportb(PORT1+0,0x39);
        printf("\007");
}
 else

 if(ch=='(')
{
        printf(" DISPLAY OF CHARACTER **** ( ****\n");
        outportb(PORT1+0,0x1B);
```

```
        printf("\007");
}
 else

  if(ch==')')
{
        printf(" DISPLAY OF CHARACTER **** ) ****\n");
        outportb(PORT1+0,0x1B);
        printf("\007");
}
 else

 if(ch=='.')
{
        printf(" DISPLAY OF CHARACTER **** . ****\n");
        outportb(PORT1+0,0x1A);
        printf("\007");
}
 else

  if(ch=='?')
{
        printf(" DISPLAY OF CHARACTER **** ? ****\n");
        outportb(PORT1+0,0x0B);
        printf("\007");
}
 else

  if(ch==';')
{
```

```
        printf(" DISPLAY OF CHARACTER **** ; ****\n");
        outportb(PORT1+0,0x03);
        printf("\007");
}
 else

 if(ch=='a')
{
        printf(" DISPLAY OF CHARACTER **** A ****\n");
        outportb(PORT1+0,0x04);
        printf("\007");
}
 else

 if(ch=='b')
{
        printf(" DISPLAY OF CHARACTER **** B *****\n");
        outportb(PORT1+0,0x06);
    printf("\007");
}

 else
 if(ch=='c')
{
        printf(" DISPLAY OF CHARACTER **** C ****\n");
        outportb(PORT1+0,0x24);
    printf("\007");
}

 else
```

```
 if(ch=='d')
{
        printf(" DISPLAY OF CHARACTER **** D ****\n");
        outportb(PORT1+0,0x34);
    printf("\007");
}

 else
  if(ch=='e')
{
        printf(" DISPLAY OF CHARACTER **** E ****\n");
        outportb(PORT1+0,0x14);
    printf("\007");
}

 else
  if(ch=='f')
{
        printf(" DISPLAY OF CHARACTER **** F ****\n");
        outportb(PORT1+0,0x26);
    printf("\007");
}

 else
  if(ch=='g')
{
        printf(" DISPLAY OF CHARACTER **** G ****\n");
        outportb(PORT1+0,0x36);
   printf("\007");
}
```

```
else
 if(ch=='h')
{
        printf(" DISPLAY OF CHARACTER **** H ****\n");
        outportb(PORT1+0,0x16);
   printf("\007");
}

else
 if(ch=='i')
{
        printf(" DISPLAY OF CHARACTER **** I ****\n");
        outportb(PORT1+0,0x22);
   printf("\007");
}

else
 if(ch=='j')
{
        printf(" DISPLAY OF CHARACTER **** J ****\n");
        outportb(PORT1+0,0x32);
   printf("\007");
}

else
 if(ch=='k')
{
        printf(" DISPLAY OF CHARACTER **** K ****\n");
        outportb(PORT1+0,0x05);
```

```
    printf("\007");
}

 else
  if(ch=='l')
{
        printf(" DISPLAY OF CHARACTER **** L ****\n");
        outportb(PORT1+0,0x07);
    printf("\007");
}

 else
  if(ch=='m')
{
        printf(" DISPLAY OF CHARACTER **** M ****\n");
        outportb(PORT1+0,0x25);
        printf("\007");

}

 else
  if(ch=='n')
{
        printf(" DISPLAY OF CHARACTER **** N ****\n");
        outportb(PORT1+0,0x35);
    printf("\007");
}

 else
 if(ch=='o')
```

```
{
        printf(" DISPLAY OF CHARACTER **** O ****\n");
        outportb(PORT1+0,0x15);
    printf("\007");
}

 else
  if(ch=='p')
{
        printf(" DISPLAY OF CHARACTER **** P ****\n");
        outportb(PORT1+0,0x27);
    printf("\007");
}

 else
  if(ch=='q')
{
        printf(" DISPLAY OF CHARACTER **** Q ****\n");
        outportb(PORT1+0,0x37);
    printf("\007");
}

 else
  if(ch=='r')
{
        printf(" DISPLAY OF CHARACTER **** R ****\n");
        outportb(PORT1+0,0x17);
    printf("\007");
}
```

```
 else
  if(ch=='s')
 {
        printf(" DISPLAY OF CHARACTER **** S ****\n");
        outportb(PORT1+0,0x23);
    printf("\007");
 }

 else
  if(ch=='t')
 {
        printf(" DISPLAY OF CHARACTER **** T ****\n");
        outportb(PORT1+0,0x33);
    printf("\007");
 }

 else
  if(ch=='u')
 {
        printf(" DISPLAY OF CHARACTER **** U ****\n");
        outportb(PORT1+0,0x0D);
    printf("\007");
 }

 else
  if(ch=='v')
 {
        printf(" DISPLAY OF CHARACTER **** V ****\n");
        outportb(PORT1+0,0x0F);
    printf("\007");
```

```
}

 else
  if(ch=='w')
{
        printf(" DISPLAY OF CHARACTER **** W ****\n");
        outportb(PORT1+0,0x3A);
    printf("\007");
}

 else
 if(ch=='x')
{
        printf(" DISPLAY OF CHARACTER **** X ****\n");
        outportb(PORT1+0,0x2D);
   printf("\007");
}

 else
  if(ch=='y')
{
        printf(" DISPLAY OF CHARACTER **** Y ****\n");
        outportb(PORT1+0,0x3D);
        printf("\007");
}

 else
  if(ch=='z')
{
        printf(" DISPLAY OF CHARACTER **** Z ****\n");
```

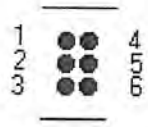
```
            outportb(PORT1+0,0x1D);
            printf("\007");
    }

     else
            outportb(PORT1+0,0x00);

  }while(ch!=27);

   outportb(PORT1+0,0x00);
}
```

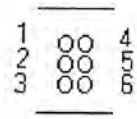
Hello!
Group

Name
Mr.Atthapon Molito
Mr.Saran Satarom
Mr.Chaiyut Cheechana

Project Braille Indicator
Advisor Mr.Atthasit Hlaskun

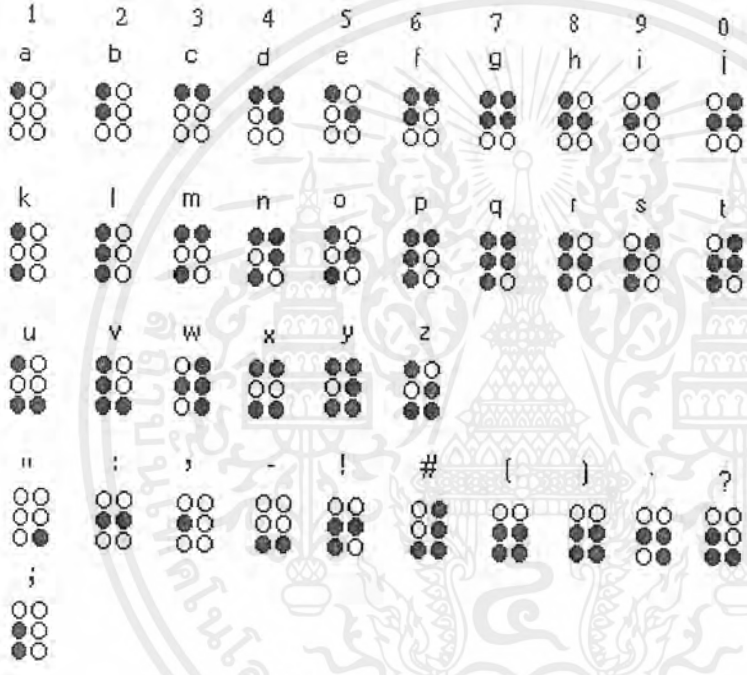
The Braille Alphabet

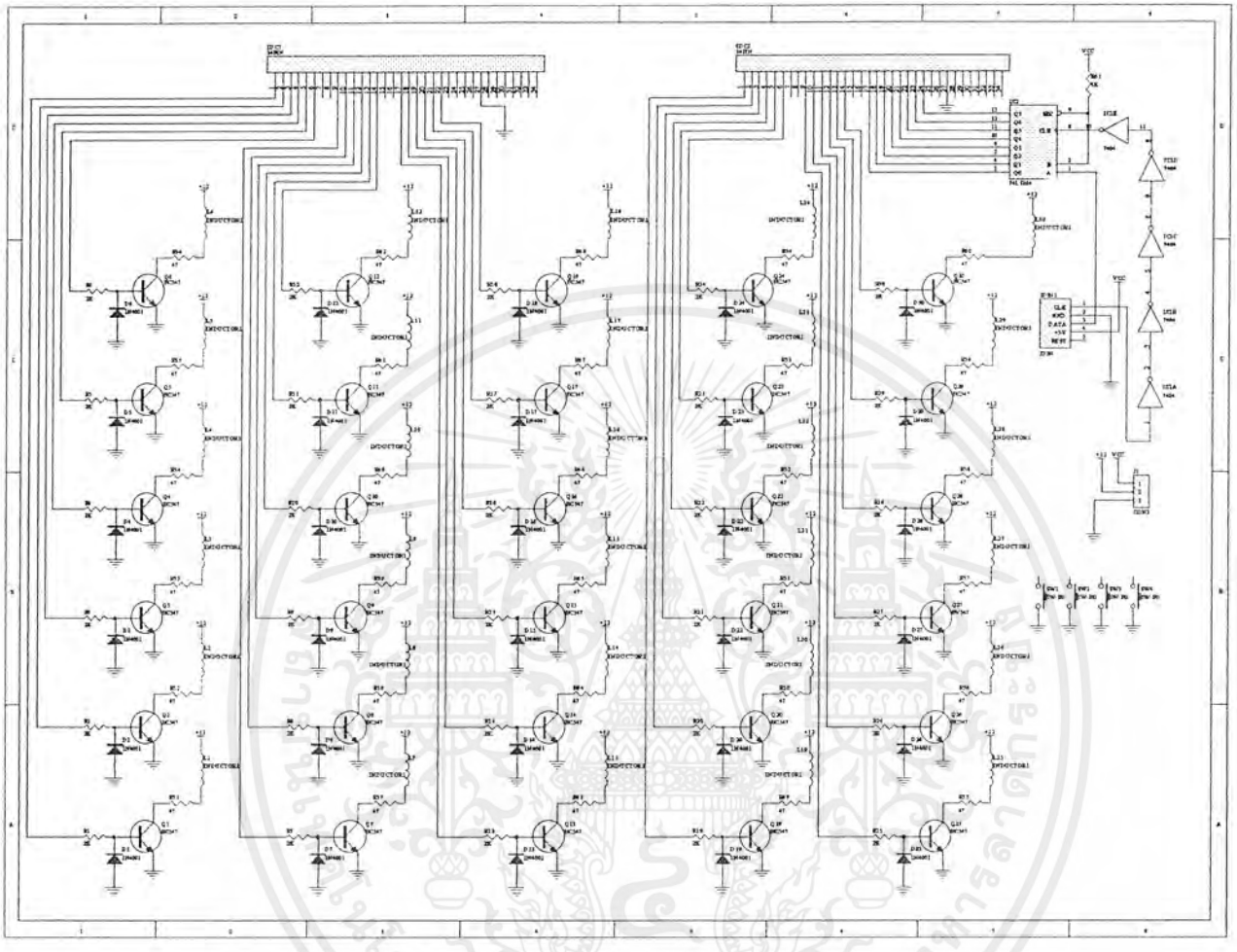




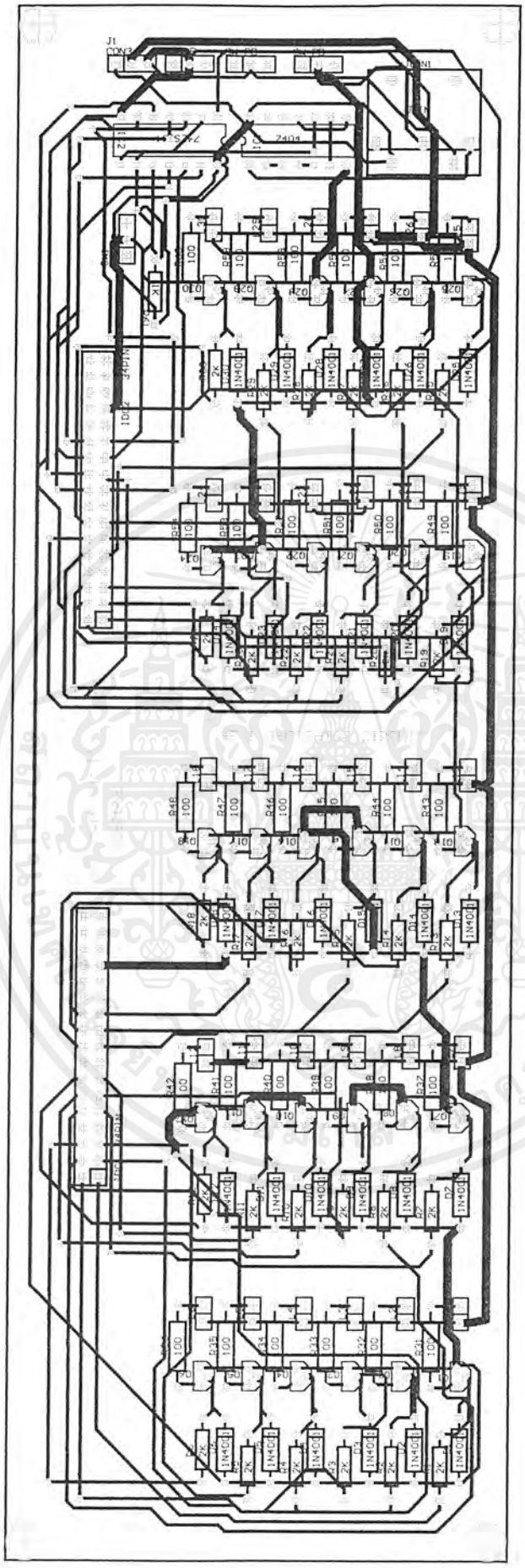
Note: ● = embossed dot;

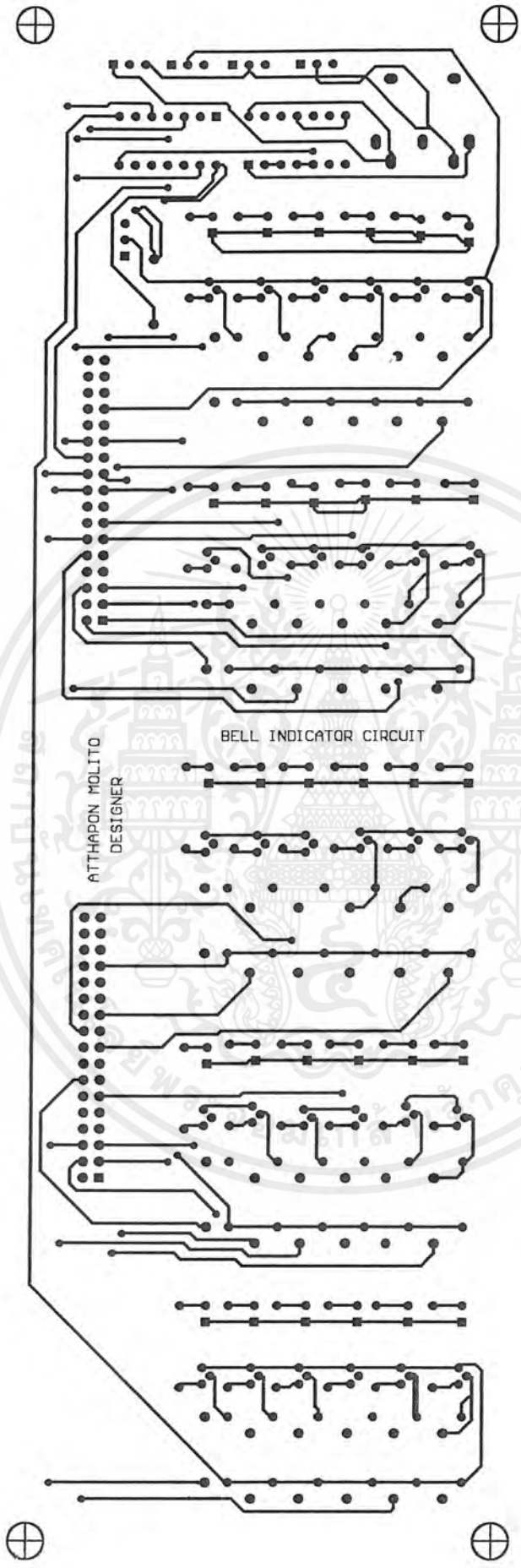
○ = unused dot in that character

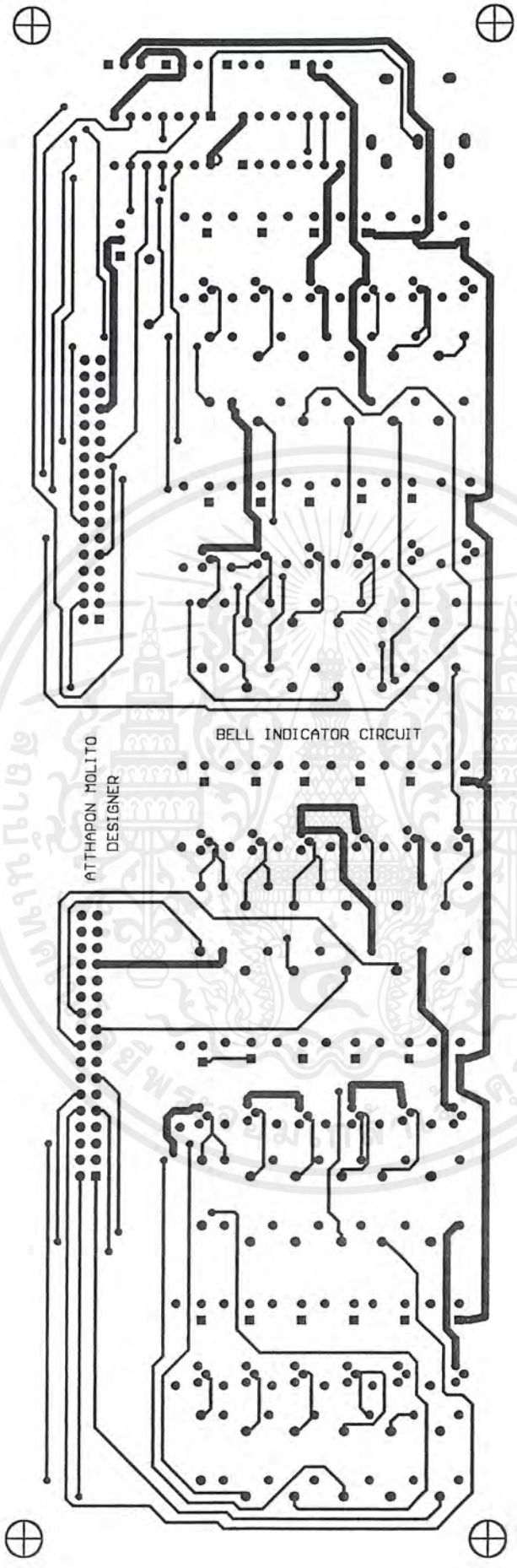


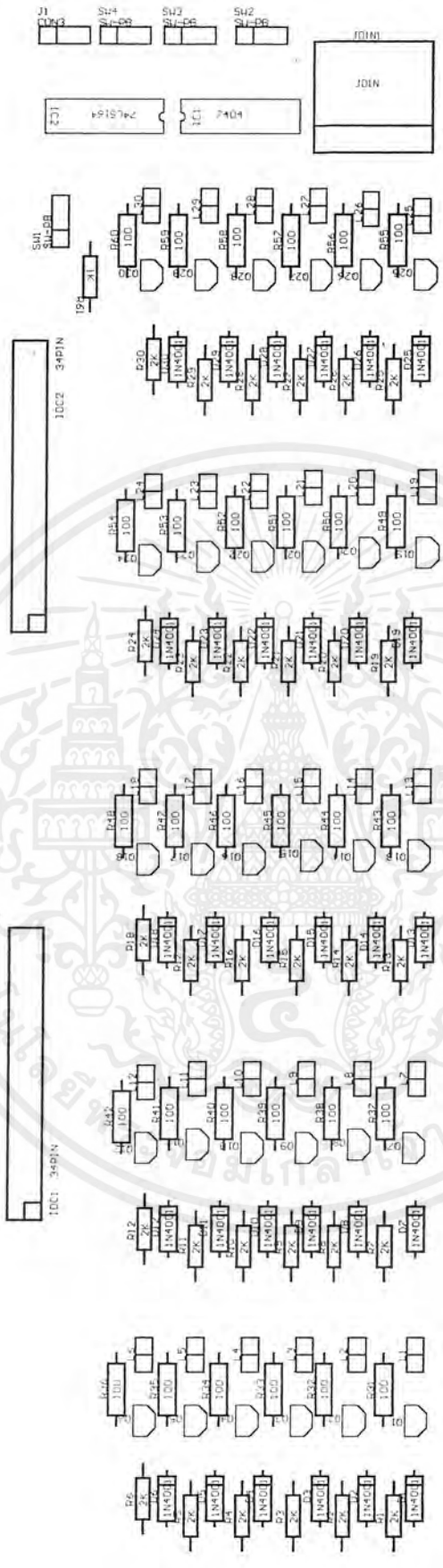


Braille Indicator Circuit









7404, LS04, S04
Inverters

Hex Inverter
Product Specification

Logic Products

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

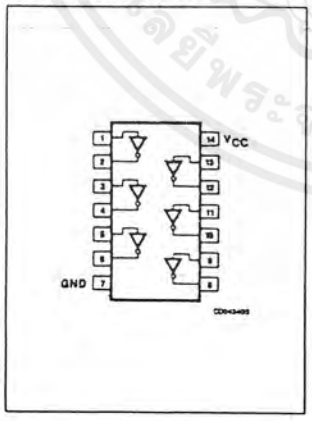
H = HIGH voltage level
L = LOW voltage level

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

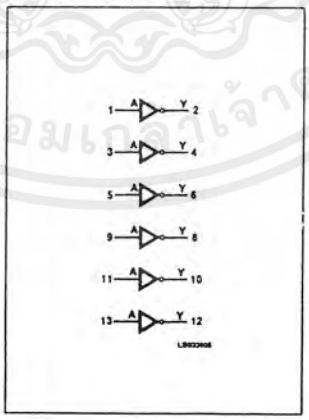
PINS	DESCRIPTION	74	74S	74LS
A	Input	1ul	1Sul	1LSul
Y	Output	10ul	10Sul	10LSul

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}.

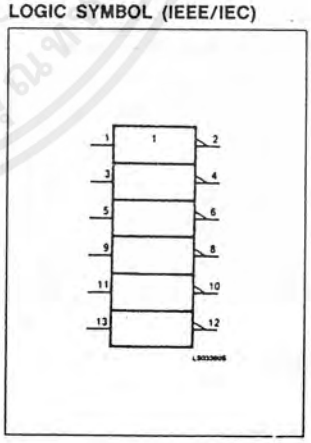
PIN CONFIGURATION



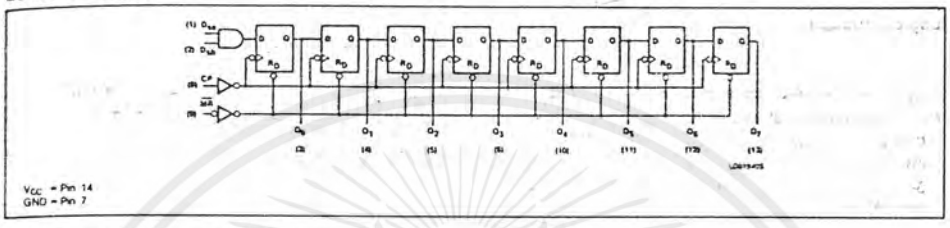
LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



LOGIC DIAGRAM



MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS				OUTPUTS	
	$\overline{MR}$	CP	D_{sa}	D_{sb}	Q_0	Q_1 — Q_7
Reset (clear)	L	X	X	X	L	L — L
Shift	H	↑	l	l	L	q_0 — q_6
	H	↑	l	h	L	q_0 — q_6
	H	↑	h	l	L	q_0 — q_6
	H	↑	h	h	H	q_0 — q_6

H = HIGH voltage level.
h = HIGH voltage level one set-up time prior to the LOW-to-HIGH Clock transition.
L = LOW voltage level.
l = LOW voltage level one set-up time prior to the LOW-to-HIGH Clock transition.
q = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW-to-HIGH Clock transition.
X = Don't care.
↑ = LOW-to-HIGH Clock transition.

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

	PARAMETER	74	74LS	UNIT
V_{CC}	Supply voltage	7.0	7.0	V
V_{IN}	Input voltage	−0.5 to +5.5	−0.5 to +7.0	V
I_{IN}	Input current	−30 to +5	−30 to +1	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	74			74LS			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			−12			−18	mA
I_{OH}	HIGH-level output current			−400			−400	µA
I_{OL}	LOW-level output current			8			8	mA
T_A	Operating free-air temperature	0		70	0		70	°C

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).

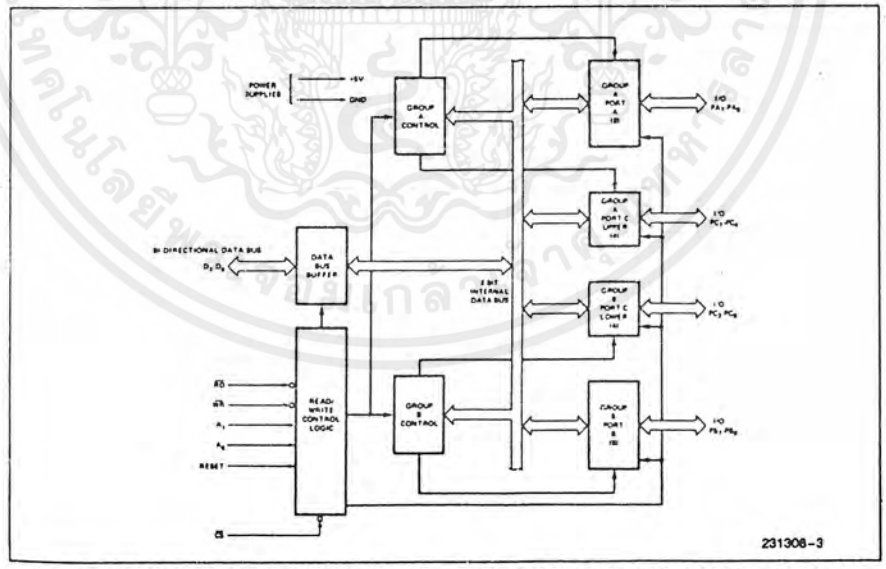


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

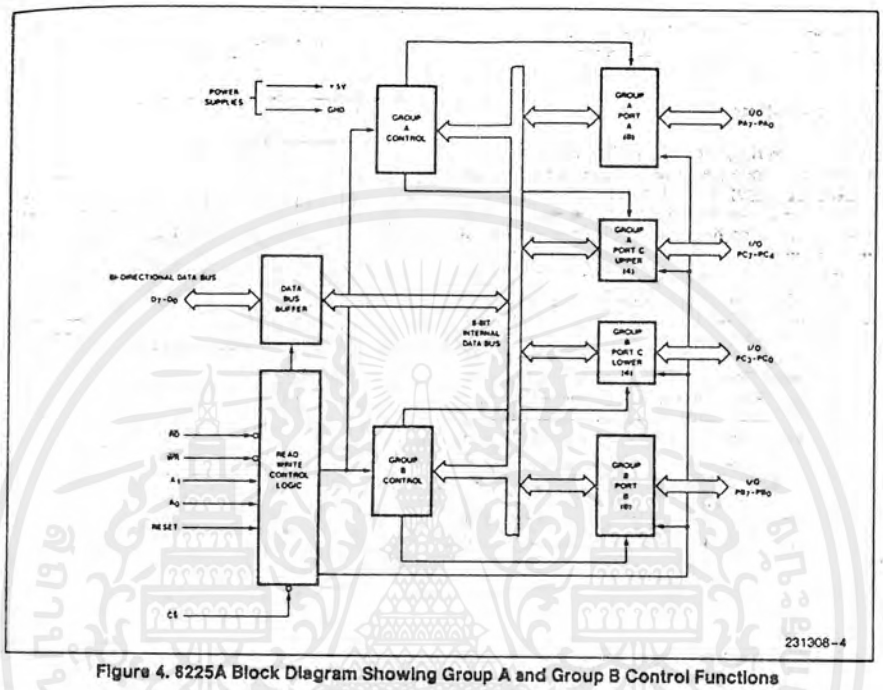
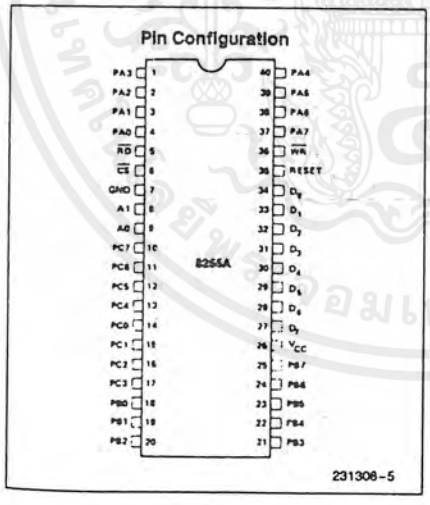


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions



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:

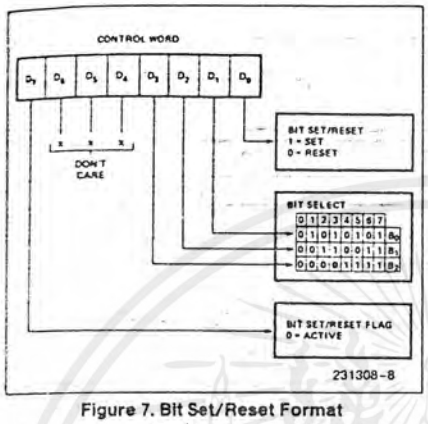


Figure 7. Bit Set/Reset Format

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.

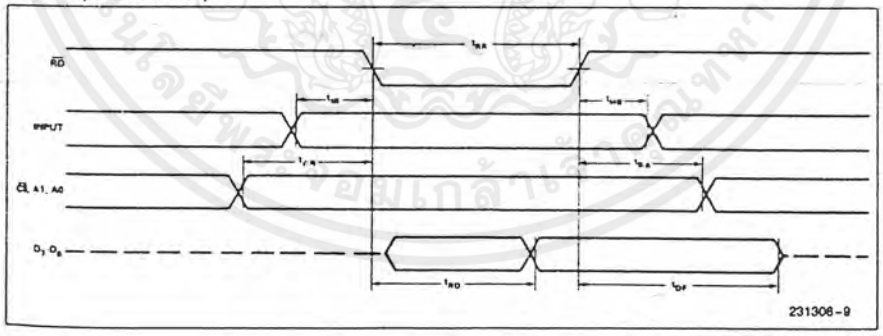
Operating Modes

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:

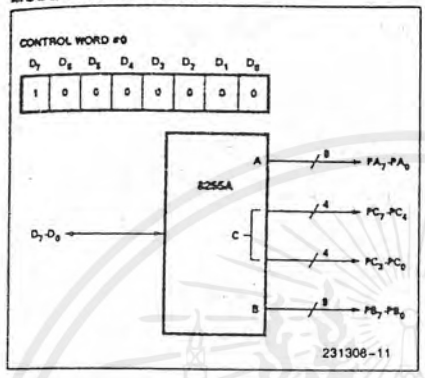
- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



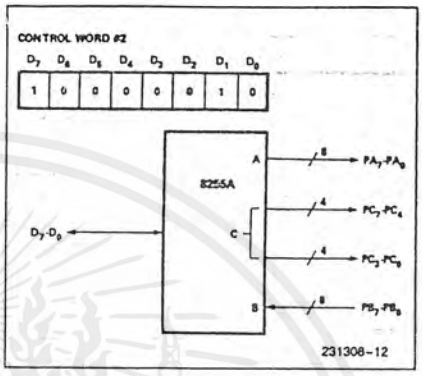
MODE CONFIGURATIONS

CONTROL WORD #0



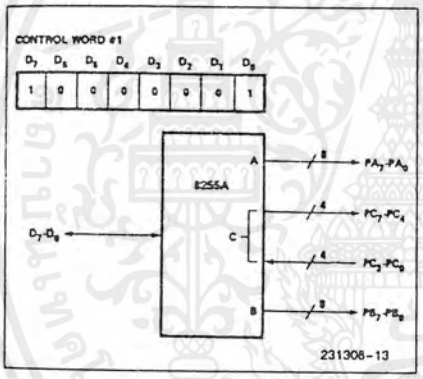
231308-11

CONTROL WORD #2



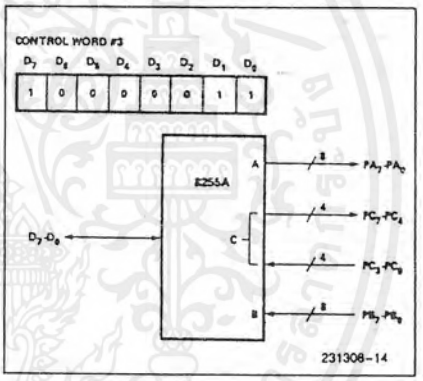
231308-12

CONTROL WORD #1



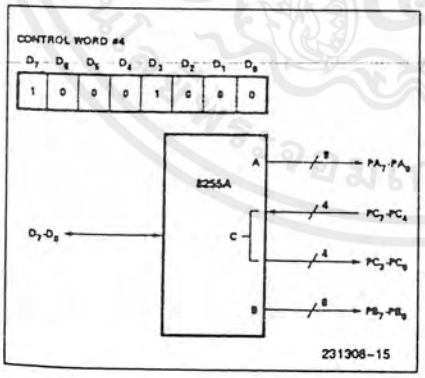
231308-13

CONTROL WORD #3



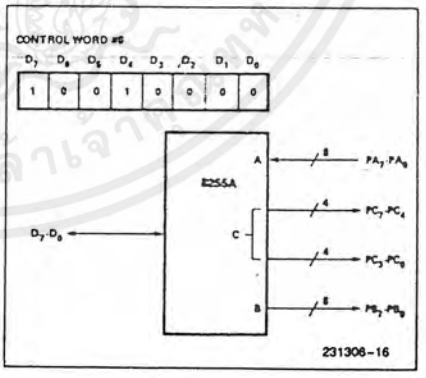
231308-14

CONTROL WORD #4

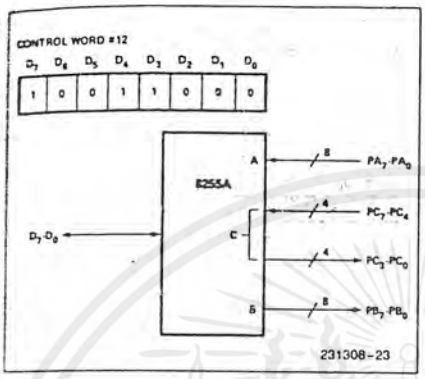


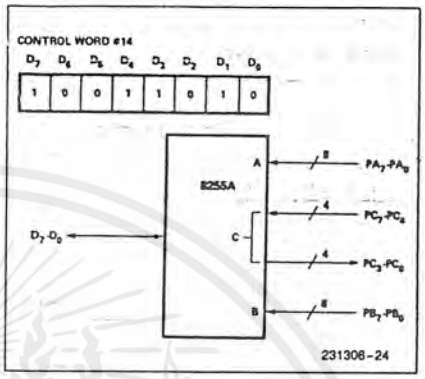
231308-15

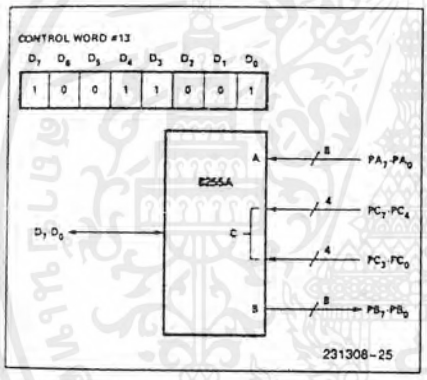
CONTROL WORD #5

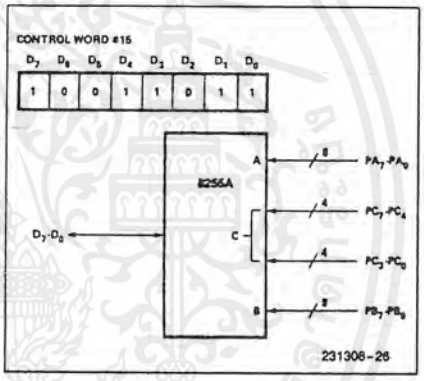


231308-16









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

$\overline{\text{STB}}$ (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 $\overline{\text{STB}}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{RD}}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

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.

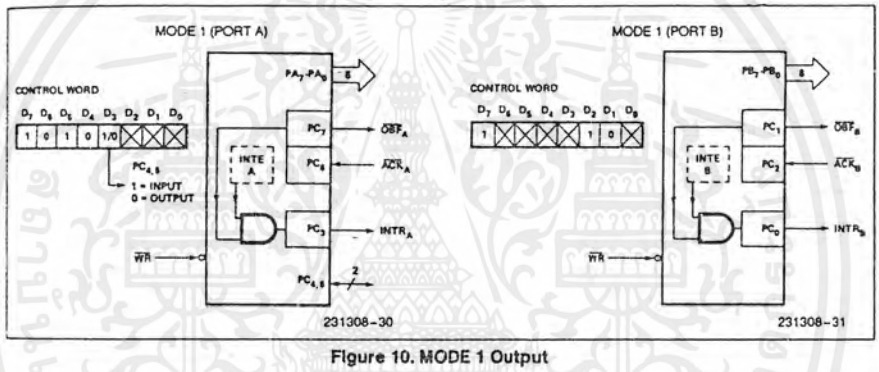


Figure 10. MODE 1 Output

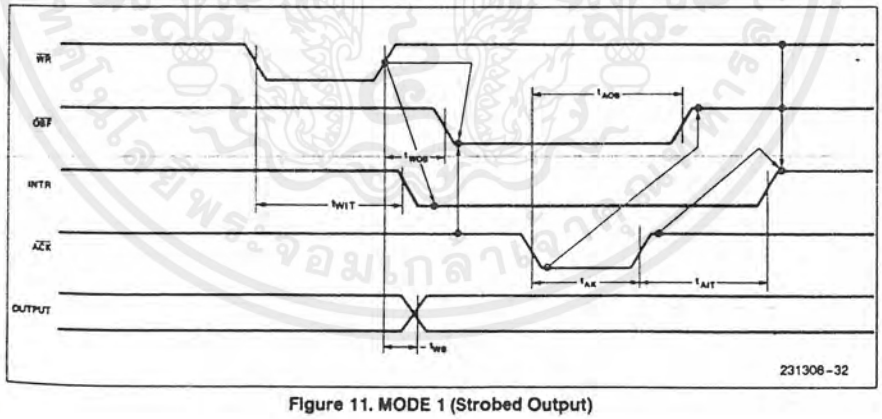


Figure 11. MODE 1 (Strobed Output)

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.

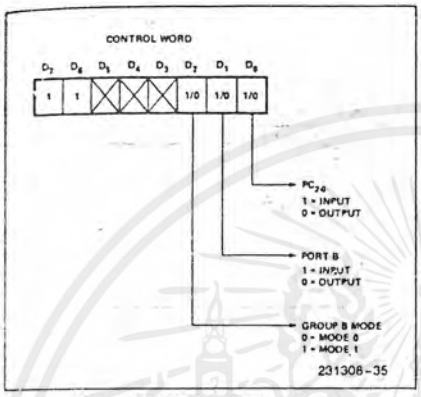


Figure 13. MODE Control Word

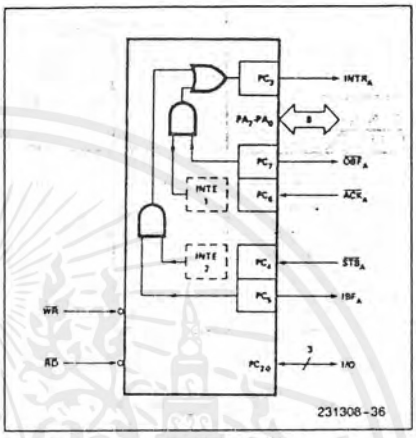
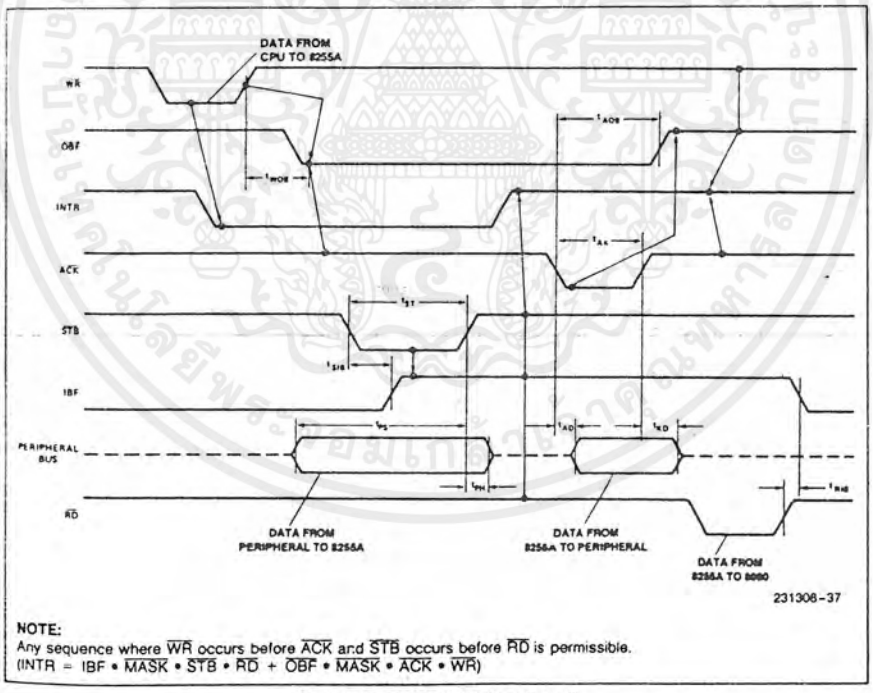


Figure 14. MODE 2



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

Figure 15. MODE 2 (Bidirectional)

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.

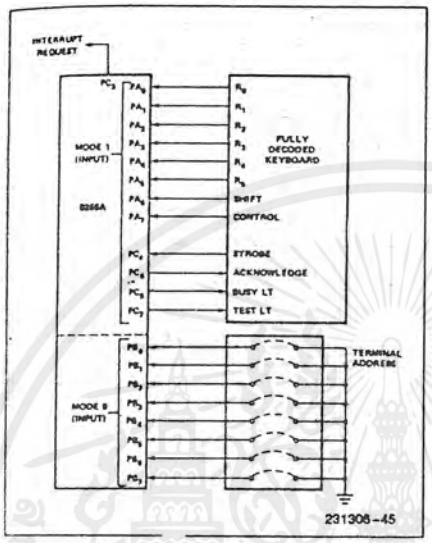


Figure 21. Keyboard and Terminal Address Interface

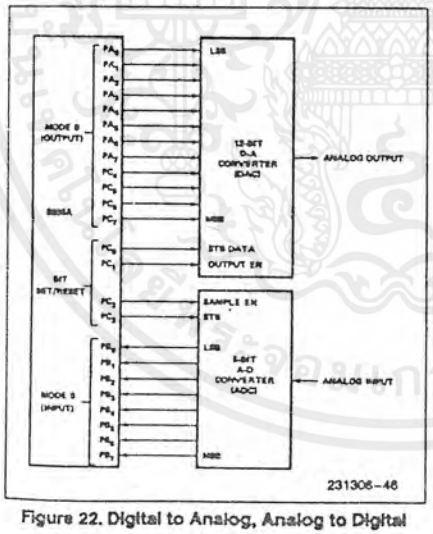


Figure 22. Digital to Analog, Analog to Digital

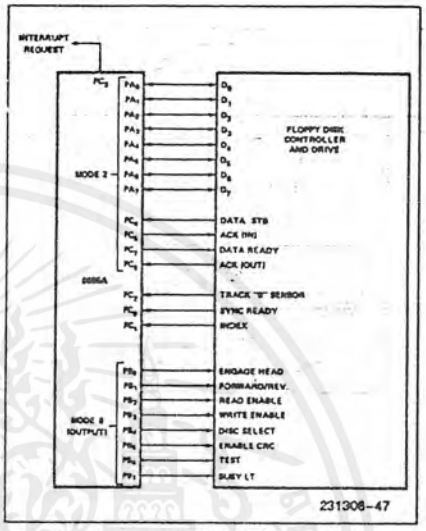


Figure 23. Basic Floppy Disk Interface

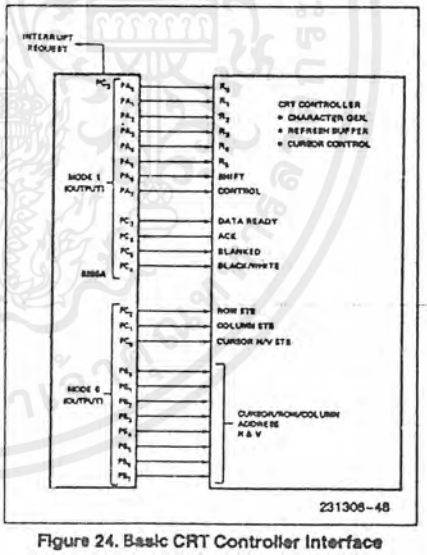


Figure 24. Basic CRT Controller Interface