เมนูอัตโนมัติ

AUTOMATIC MENU

โดย

นายอุทัย สกุลอุดมกาญจน์

นายเอกพจน์ บุณยรัตพันธ์

นายโอฬาร ศรีเสวต

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

เมนูอัตโนมัติ

AUTOMATIC MENU

โดย

นายอุทัย สกุลอุดมกาญจน์ รหัส 37013229

นายเอกพจน์ บุณยรัตพันธ์ รหัส 37013230

นายโอฬาร ศรีเสวต รหัส 37013232

อาจารย์ ที่ปรึกษา

รศ. ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ใบแนบที่ 3

รายงานปีการศึกษา 2539

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เมนูอัตโนมัติ

ผู้จัดทำ

1	นายอุทัย	สกุลอุดมกาญจน์	รหัส	37013229	3R/1
2.	นายเอกพจน์	บุณยวัตพันธ์	รหัส	37013230	3R/1
3.	นายไอพาร	ศรีเสวต	รหัส	37013232	3R/1

..... อาจารย์ที่ปรึกษา

(.....)

ใบแบบที่4 แบบฟอร์มรับรองความพร้อมในการสอบ

เมนูอัตโนมัติ

AUTOMATIC MENU

1 นายอุทัย สกุลอุดมกาญจน์ รหัส 37013229

2. นายเอกพจน์ บุณยรัตพันธ์ รหัส 37013230

3. นายโอฬาร ศรีเสาวต รหัส 37013232

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

....

(.....)

อาจารย์ที่ปรึกษา

เมนูอัตโนมัติ

อุทัย	สกุลอุดมกาญจน์
เอกพจน์	บุญยรัตพันธ์
โอพาร	ศรีเสวต
รศ.ดร.มนัส	สังวรศิลป์ อาจารย์ที่ปรึกษา
ปีการศึกษา	2539

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เป็นการศึกษาถึงทฤษฎีการอินเตอร์เฟสของระบบสื่อสารข้อมูลทางคอมพิวเตอร์ รวมถึงการนำไปประยุกต์ในการใช้งาน ซึ่งลักษณะปริญญานิพนธ์นี้ เป็นการศึกษาความรู้ทางด้านการติดต่อสื่อสารข้อมูลผ่านระบบคอมพิวเตอร์ โดยมีโครงสร้างและแนวคิด เริ่มจากการส่งข้อมูลจากส่วนหนึ่ง ซึ่งเป็นส่วนของไมโครโปรเซสเซอร์ Z80 ต่อรวมกับคีย์บอร์ดและจอแสดงผล ส่งผ่านไปยังการ์ด RS-232 จะทำหน้าที่ส่งข้อมูลไปการ์ด RS-232 ปลายทาง ซึ่งจะต่ออยู่กับเครื่องไมโครคอมพิวเตอร์ ทำหน้าที่รับข้อมูลและนำไปประมวลผลใช้งานต่อไป ปริญญานิพนธ์นี้มีจุดประสงค์เพื่อใช้กับงาน ที่มีลักษณะการสื่อสารทิศทางเดียว ซึ่งภายในขอบเขตงานดังกล่าวสามารถช่วยลดการสิ้นเปลืองได้มาก โดยเฉพาะระบบงานโครงข่ายใหญ่ ๆ

INTERFACE OF COMPUTER NETWORK

Uthai sakuludomkan
Akkapot Boonyarattapum
Oran Srisawel

Abstract

This thesis direct us to the computer network interfacing theorem including its application as well. The contex of this thesis quided us to the ability of the data communication via computer system. The framework and idea start on a unit of data is transmitted from the Z-80 set which consists of keyboard and monitor to a destination microprocessor for processing by using RS-232 end-to-end connection. As the above mentioned, this thesis confined us to haft duplex on unidirectional communication. Within this scope we can apply do the large network system and make it efficency with low cost.

สารบัญ

สารบัญรูปภาพ

หน้า

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

ในยุคข้อมูลและข่าวสารนี้สิ่งที่มีความสำคัญมากที่สุดสิ่งหนึ่งก็คือ ระบบการสื่อสารการใช้ระบบการสื่อสารนั้นจำเป็นต้องมีการกำหนดมาตรฐานขึ้น ถ้าหากไม่มีการกำหนดมาตรฐานแล้วการสื่อสารก็คงจะสับสนยุ่งเหยิงไร้ระเบียบ เกิดปัญหามากมายไม่รู้จบข้อมูลที่มีอยู่ไม่รู้ว่า จะมีคุณค่ามากเพียงใดก็ไม่สามารถนำไปใช้ให้เกิดประโยชน์อย่างเต็มที่ได้ในชีวิตประจำวัน การสื่อสารนับวันจะมีบทบาทมากขึ้นเริ่มตั้งแต่อดีตที่มีการคิดค้นรหัสมอส ได้มีการพัฒนาโทรเลขจากโทรเลขก็พัฒนามาถึงยุคสมัยของคอมพิวเตอร์ความจำเป็นในการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ก็มีมากขึ้น การพัฒนาการสื่อสารข้อมูลทำให้การใช้งานของคอมพิวเตอร์มีประโยชน์มหาศาลกิจการหลายอย่าง ได้อาศัยความสะดวกสบายของการสื่อสารข้อมูล เช่น งานด้านฐานข้อมูล สายการบิน การธนาคาร ในช่วงไม่กี่ปีที่ผ่านมาเทคโนโลยี ไมโครคอมพิวเตอร์ได้ก้าวหน้าขึ้นมาก มีขีดความสามารถทางสาร์ดแวร์สูงขึ้น ได้มีการพัฒนาการสื่อสารข้อมูลโดยนำไมโครคอมพิวเตอร์หลายๆเครื่องมาต่อรวมกันเป็นระบบ

1.1 การสื่อสารข้อมูล

ในการสื่อสารข้อมูลจะมีแบบอนุกรมและการสื่อสารแบบขนาน ในส่วนของการการสื่อสารข้อมูลแบบขนานก็คือข้อมูลทุกๆบิตในแต่ละเวิร์ด จะถูกส่งออกไปพร้อมๆกัน ขึ้นอยู่กับว่าเวิร์ดดังกล่าวมีขนาดเท่าไหร่ โดยทั่วไปก็คือ 8 บิตนั่นเอง การส่งข้อมูลแบบขนานนี้ จะมีข้อจำกัดทางด้านระยะทางระหว่างต้นทางและปลายทางโดยทั่วไปจะส่งได้ในระยะไม่เกิน 3-5 ฟุตเท่านั้นทั้งนี้ขึ้นอยู่กับอัตราที่ใช้ส่งข้อมูลถ้าหากใช้อัตราการส่งข้อมูลสูงก็จะส่งได้ระยะสั้นลง การส่งข้อมูลแบบขนานนั้นนิยมในระบบที่ต้องการความเร็วสูงมาก ๆแต่อุปกรณ์ไม่อยู่ห่างกันมากนัก

ส่วนการส่งข้อมูลแบบอนุกรมนั้นข้อมูลจะถูกทยอยส่งออกไปทีละบิตจนครบทั้งเวิร์ดในสายสัญญาณเพียงเส้นเดียวแต่ในการใช้งานจริงจะต้องมีสายสัญญาณอีกเส้นเป็นระดับกราวด์ ดังนั้นเมื่อเราส่งข้อมูลในแบบอนุกรมเราจะสามารถใช้สายสัญญาณอย่างน้อยที่สุดเพียง 2 เส้น ในขณะที่การส่งข้อมูลแบบขนาน จะต้องอย่างน้อยเท่ากับจำนวนบิต บวกกับสายสัญญาณ ระดับแรงดัน Grand อีก 1 เส้นนอกจากนี้การส่งข้อมูลแบบอนุกรมนั้นจะสามารถส่งข้อมูลได้ไกลกว่ามาก อย่างไรก็ดีการส่งข้อมูลแบบอนุกรมนั้นจะต้องมีส่วนที่ทำหน้าที่แปลงข้อมูลจาก ข้อมูลแบบขนานมาเป็นแบบอนุกรมซึ่งสามารถเป็นได้ทั้งฮาร์ดแวร์และซอร์ฟแวร์และในการส่ง ก็มีข้อจำกัดบางประการเพื่อให้การส่งข้อมูลมีความถูกต้องมากยิ่งขึ้นด้วย

1.2 Simplex และ Duplex

ในการสื่อสารไม่ว่าจะเป็นการสื่อสารข้อมูล หรือการสื่อสารทั่วไปนั้นย่อมจะต้องประกอบด้วยผู้รับและผู้ส่ง ผู้รับในขณะนี้สามารถเป็นผู้ส่งในอนาคตได้ แต่มีบางกรณีสำหรับ การสื่อสารข้อมูลที่ผู้รับและผู้ส่งจะตายตัวอยู่ตลอดเวลา เช่น การสื่อสารข้อมูลระหว่าง เครื่องคอมพิวเตอร์กับเครื่องพิมพ์ การสื่อสารของอุปกรณ์ที่มีผู้รับและผู้ส่งตายตัวนั้นเราเรียกว่าการสื่อสารแบบ Simplex กล่าวคือการสื่อสารเป็นไปในลักษณะทิศทางเดียวตลอดเวลาซึ่งจะมีที่ใช้น้อยมาก การสื่อสารโดยทั่วไปนั้นจะเป็นลักษณะ Duplex คือมีทิศทางการสื่อสาร 2 ทิศทางทั้งไปและกลับการสื่อสารในลักษณะ Duplex ยังแบ่งออกได้เป็น 2 ชนิด คือ Half Duplex ซึ่งจะมีทิศทางการสื่อสารในลักษณะที่ผลัดกันเป็นผู้ส่งและผู้รับ กับสื่อสารในลักษณะของ Full Duplex ซึ่งอุปกรณ์ที่ปลายทางจะเป็นผู้ส่งและผู้รับในเวลาเดียวกัน เราอาจเปรียบเทียบการสื่อสารแบบ HDX หรือ Half Duplex เป็นทางรถไฟ ซึ่งจะมีเพียงรถไฟขบวนเดียวเท่านั้นที่วิ่งอยู่บนรางในเวลาหนึ่งและเปรียบเทียบ FDX เป็นถนนที่รถสามารถวิ่งสวนกันได้ในเวลาเดียวกันการสื่อสารระหว่างคอมพิวเตอร์มักจะอยู่ในลักษณะของ Duplex แบบใดแบบหนึ่ง

1.3 หน้าที่ของพอร์ตอนุกรม

จุดประสงค์แรกที่มีการประดิษฐ์พอร์ตอนุกรมขึ้นมานั้น ก็ใช้เป็นพอร์ตสำหรับการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ด้วยกันดังจะเห็นได้จากที่บางครั้งจะมีคนเรียกพอร์ตนี้ว่า Communication Port แต่เนื่องจากในช่วงแรกๆ ที่เรารู้จักพอร์ตอนุกรมจากเครื่องคอมพิวเตอร์ใน ระดับไมโครคอมพิวเตอร์นั้นการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ไม่ใช้เรื่องที่น่าสนใจนัก จึงทำให้ความหมายของพอร์ตชนิดนี้เปลี่ยนแปลงไปจนกระทั่งมีผู้ประยุกต์นำเอาพอร์ตอนุกรมมาต่อกับเมาส์ จึงทำให้มีความสำคัญอีกครั้ง

1.4 โครงสร้างของพอร์ตอนุกรมบนไอบีเอ็มพีซี

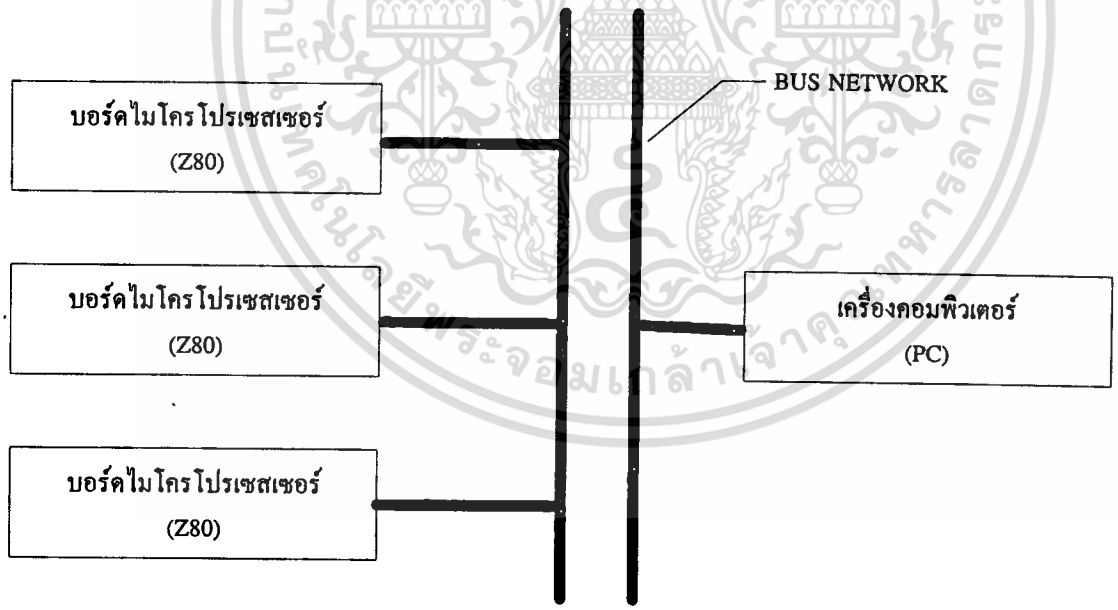
พอร์ตอนุกรมไอบีเอ็มมีชิป 8250 เป็นหัวใจสำคัญ ทุกๆส่วนที่เกี่ยวข้องกับพอร์ตอนุกรมจะต้องมีการติดต่อกับชิปนี้ ส่วนอื่นๆนอกเหนือจากนี้มีการติดต่อกับ CPU จะมีการ decode ของ Address Bus ตามโครงสร้างทั่วไปของระบบคอมพิวเตอร์ ส่วนที่เป็น Oscillator จะทำหน้าที่สร้างสัญญาณนาฬิกาเพื่อใช้เป็นเวลาอ้างอิงสำหรับชิป 8250 ในการทำงานต่างๆ เช่นในการรับส่งข้อมูลเป็นต้น ส่วน driver ทำหน้าที่ขับสัญญาณให้มีคุณสมบัติตามมาตราฐาน EIA-232 ทั้งที่การเชื่อมต่อระหว่างอุปกรณ์ผ่านซีเรียลพอร์ตของไอบีเอ็มพีซีจะใช้มาตราฐานของEIA-232 (RS-232) เป็นหลัก

1.5 บทนำของโครงงาน

ความก้าวหน้าทางเทคโนโลยีของวิทยาศาสตร์ในทางอิเลคทรอนิคส์ และคอมพิวเตอร์ในระยะเวลาหลายปีที่ผ่านมา มีการพัฒนาไปอย่างรวดเร็ว และมีประสิทธิภาพสูง ทำให้มีการใช้ระบบอิเล็คทรอนิกส์และคอมพิวเตอร์มาเป็นตัวควบคุมกระบวนการต่างๆ ที่ซับซ้อนและยุ่งยาก ส่งผลทำให้สามารถเพิ่มประสิทธิภาพของระบบงานต่างๆ ได้เป็นอย่างดี

ในโครงงานนี้ เป็นการศึกษาในหัวข้อเรื่อง การอินเตอร์เฟสของระบบสื่อสารคอมพิวเตอร์ (INTERFACE OF COMMUNICATION COMPUTER)

โครงสร้างของ PROJECT นี้ เป็นการสื่อสารข้อมูลระหว่างเครื่องไมโครคอมพิวเตอร์ กับบอร์ดไมโครโปรเซสเซอร์ (Z80) หลายๆบอร์ด ที่อยู่ในที่ต่างๆกัน ไปที่เครื่องไมโครคอมพิวเตอร์ (PC) เครื่องหนึ่ง โดยผ่านการ์ด RS-232 เพื่อนำไปเป็นข้อมูลที่ใช้ในการประมวลผลต่อไป



รูปที่ 1.1 แสดงระบบโครงข่ายตามแนวความคิด PROJECT

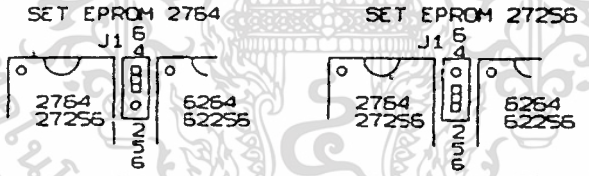
บทที่ 2

CP-Z80_V2

2.1 ข้อมูลของ BOARD CP - Z80 V2

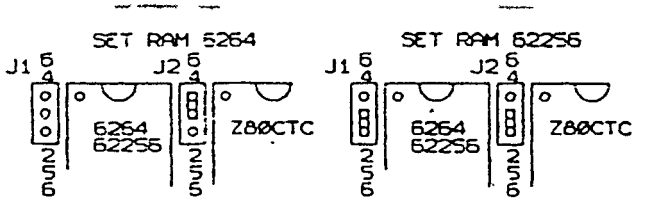
CPU ใช้CPU ยอดนิยม Z80 CPU เป็น CPU ประจำบอร์ดโดยใช้ Z84C00-6 (Z80 B แบบ CMOS) ซึ่งเป็น CPU Z80 แบบ CMOS กินกำลังงานต่ำ สามารถต่อใช้งานกับความถี่ได้สูงสุด 6 Mhz แต่ในบอร์ดนี้ เราจะใช้ความถี่ 4 MHz เพื่อไม่จำเป็นต้องใช้ ROM หรือ RAM ที่มี ACESS TIME ต่ำมากนักก็ได้ แต่ถ้าผู้ใช้จะเปลี่ยนเป็น RUN 6 MHz ก็ได้โดยการเปลี่ยนเป็น XTAL ใหม่จาก 4 MHz เป็น 6 MHz ก็ได้

ROM หรือ EPROM บอร์ด CP-Z80V2 จะต่อใช้ EPROM เป็น MONITOR PROGRAM ได้ 2 เบอร์ คือ เบอร์ 2764 และ 27256 โดยการเลือก JUMPER J2 หน่วยความจำนี้จะ DECODE อยู่ระหว่าง 0000H ถึง 7FFFH



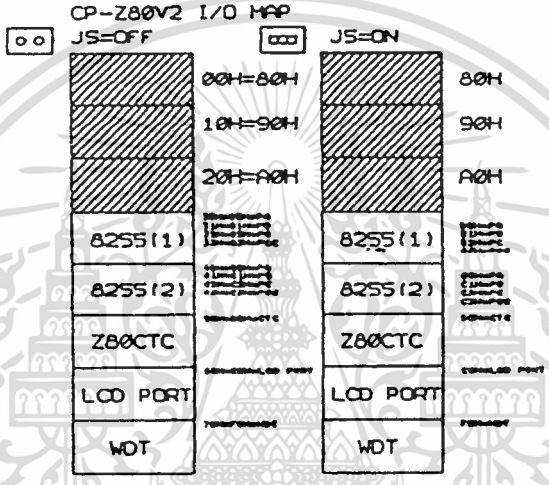
รูปที่ 2.1 ลักษณะของ ROM

RAM บอร์ด CP-Z80V2 จะใช้ RAM ขนาด 8K BYTE ขนาดเล็กพิเศษ โดยตำแหน่งขาจะเป็นแบบ RAM 6264 ธรรมดาทุกประการ หรือ เบอร์ 62256 ขนาด 32 KBYTE โดยการเลือกใช้ JUMPER J2 ดังรูป หน่วยความจำนี้จะ DECODE อยู่ในระหว่าง 8000H ถึง BFFFH (โดยตำแหน่ง 8000H-9FFFH และ A000H-BFFFH จะเป็นตำแหน่งเดียวกัน) ใน RAM นี้ ยังสามารถต่อ BATTERY ขนาดเล็ก 3V เพื่อใช้ BACK UP ข้อมูลได้ด้วย โดยใช้ J1 ในการ ON/OFF BATTERY



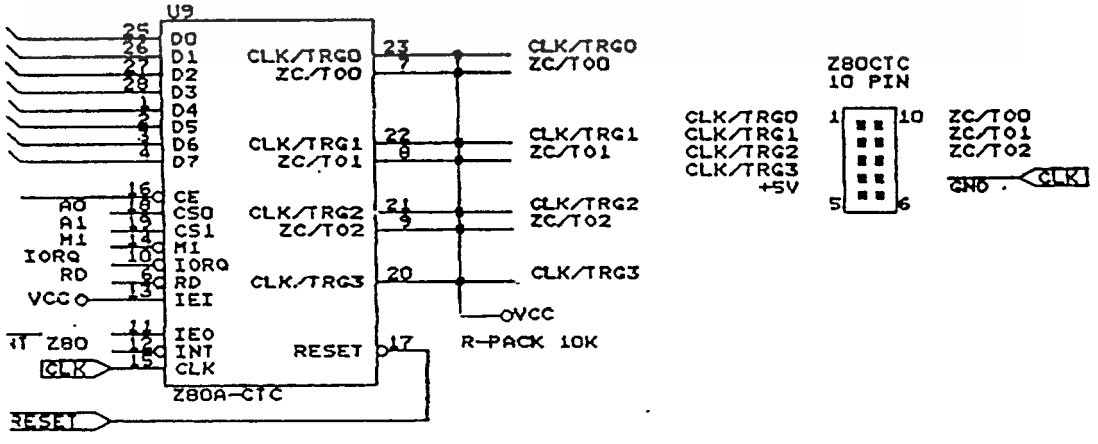
รูปที่ 2.2 ลักษณะของ RAM

PORT จะใช้ IC PORT เบอร์ 82C55 ขนาดเล็กพิเศษจำนวน 2 ตัวหรือ 6 Port แบบ PLCC TYPE 34 PIN คุณภาพสูงเป็น PORT I/O ประจำบอร์ด ทำให้กินกำลังงานต่ำ โดยจะต่อออก PORT ทางขั้ว 34 PIN มาตราฐาน อีทีที ทำให้เลือกใช้อุปกรณ์บอร์ดต่าง ๆ ของ อีทีที ได้ เช่น ET-SSRAC , ET-SMCC ฯลฯ นอกจากนี้ยังต่อขั้ว 10 PIN จาก Port C ของ 8255 ให้สามารถเลือกต่อ Keyboard ขนาด 4 * 4 (16 Key) ได้ด้วย ตำแหน่ง DECODE นั้นเราสามารถใช้ J6 ซึ่งต่อจาก Address (A7) DECODE ตำแหน่งบอร์ดได้ 2 แบบ ดังรูป



รูปที่ 2.3 ตำแหน่งของ PORT

CTC จะใช้ไอซีของบริษัท ZILOG เบอร์ Z8430 เป็น CTC ประจำบอร์ด ซึ่งเป็นไอซีแบบ 4 MHz บอร์ด CP - Z80 V2 จะต่อขาใช้งานของ CTC ออกมาที่ Connector 10 PIN ดังรูป ส่วนขา INT ของ CTC นั้นจะต่อเข้ากับ INT ของ Z80 โดยตรง



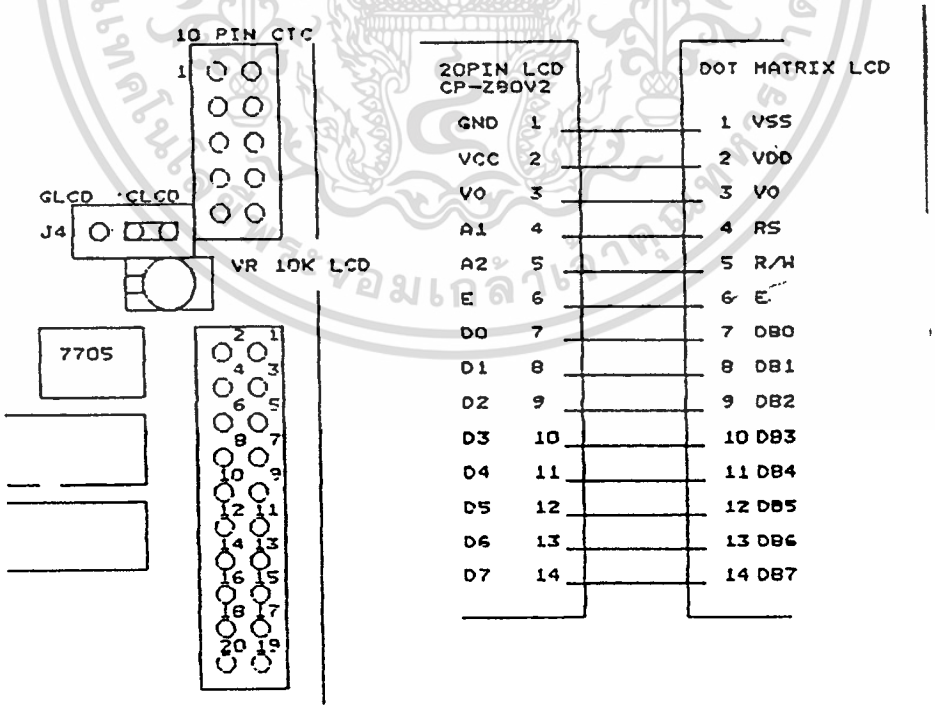
รูปที่ 2.4 ลักษณะของขาต่อ CTC

LCD_PORT บอร์ด CP-Z80V2 นี้ มีขั้ว CONNECTOR ขนาด 20 PIN มาตราฐาน อีทีที สามารถต่อ DOT MATRIX LED หรือ GRAPHIC LCD ได้โดยตรง เพียงต่อสายจาก LCD PORT เท่านั้น ไม่เสีย PORT 82C55 ใช้ JUMPER J4 ในการเลือกว่าเป็น LCD แบบใดปรับค่าความคมชัดได้ด้วย VR 10K บนบอร์ด

PORT LCD นี้ จะ DECODE อยู่ในตำแหน่ง 80H-BFH

LCD ADDESS PORT

CHANNEL PORT	ASSRESS PORT
	SET J5 = OFF
WRITE DATA INSTRUCTION	60 H
WRITE DATA TO OG OR DD RAM	62 H
READ BUSY FLAG AND ADDRESS	64 H



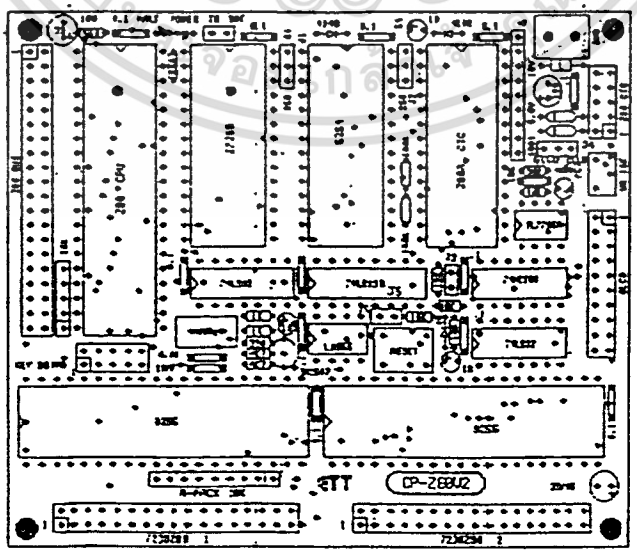
รูปที่ 2.5 ลักษณะของขาต่อ LED PORT

WDT (WATCH DOG TIMER) เป็นวงจรจาก ไอซี 555 ที่จะคอย reset CPU อยู่ตลอดเวลา ตามค่าเวลา ถ้าไม่มีการเรียกโปรแกรมที่จะมาทำการ Disable Watch dog ในวงจรนี้จะอยู่ที่ตำแหน่ง 70H หรือ F0H โดยเราอาจจะปรับเปลี่ยนค่า เวลาของวงจร WDT ได้ด้วย R3,R4, และ C4 และถ้าเราไม่ต้องการใช้วงจร WDT นี้ก็สามารถ SET JUMPER J3 โดยถอด J3 ออกก็ได้ค่าเวลา WDT จะ SET ไว้ประมาณ 1 วินาที

* (WATCH DOG เป็นลักษณะวงจรที่ทำการ RESET CPU อยู่เสมอตามเวลาที่เรากำหนด ซึ่งถ้าเราไม่ทำการ Disable Watch dog ภายในเวลากำหนด CPU นั้น จะถูก RESET เช่นในโปรแกรมทำงานปกติเราจะ CALL Disable dog อยู่เสมอ แต่ถ้า CPU กำลัง RUN อยู่นั้น เกิดมีสัญญาณรบกวนขึ้น ทำให้ไม่สามารถ RUN โปรแกรมตามปกติที่มีการเรียกใช้ CALL Disable Watch dog ได้ CPU ก็จะเกิดการ RESET ขึ้นทันที เพื่อให้ CPU กลับไปเริ่มต้น RUN โปรแกรมใหม่อีกครั้งหนึ่ง)

40 PIN Z80 BUS สามารถต่อขยายบอร์ดได้ทาง 40 PIN Z80 BUSโดย 40 PIN Z80BUS นี้จะมีขาต่อออกมาเช่นเดียวกับขา IC CPU Z80

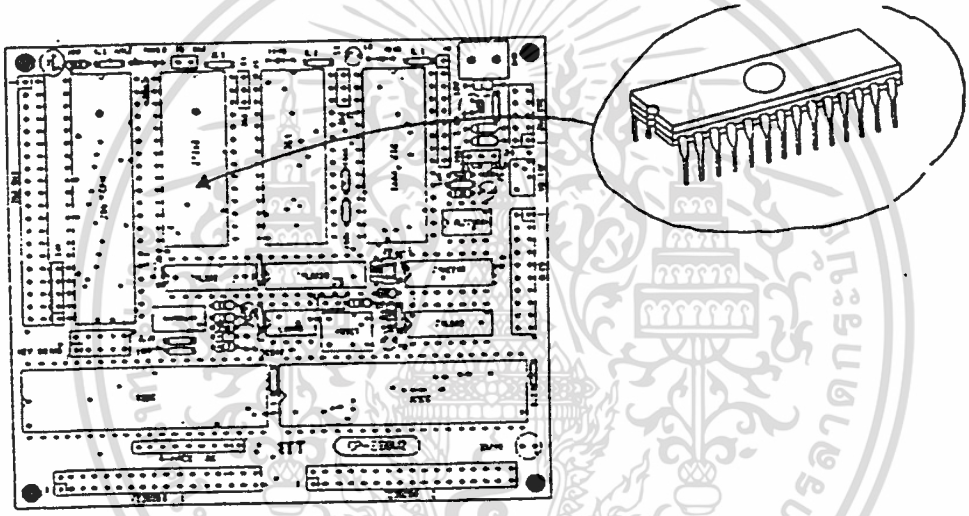
POWER SUPPLY ตัวบอร์ด CP-Z80V2 นี้จะต่อใช้ POWER SUPPLY +5V โดยใช้ไฟ +5V DC โดยต่อให้ถูกต้องด้วย และถ้าต่อกลับขั้ว ตัวบอร์ดจะมี DIODE 1N4001 ต่อกลับขั้วไว้ พร้อมทั้งยังมี ZENER DIODE 5.6V 1W ต่อกันในกรณี POWER SUPPLY เกิน 5V อีกด้วย



รูปที่ 2.6 แสดงการวางอุปกรณ์บนบอร์ด CP-Z80 V2

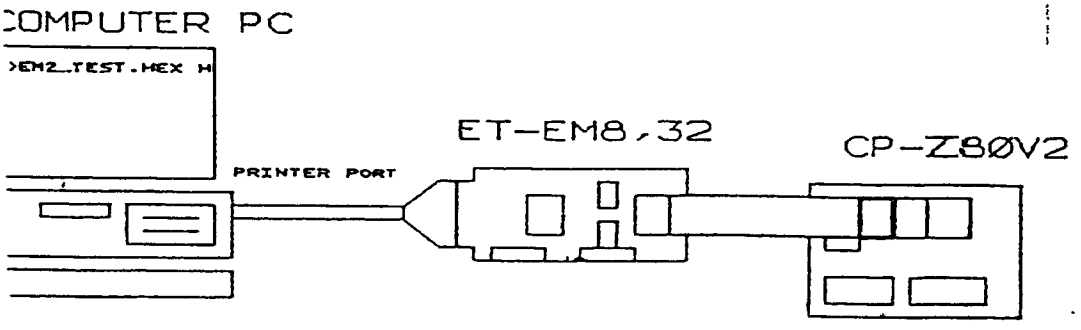
2.2 การต่อพัฒนาเขียนโปรแกรม

ตัวบอร์ด CP-Z80V1 นี้จะให้ผู้ใช้เขียนโปรแกรมสั่งงานขึ้นมาใช้เอง โดยอาจจะใช้การเขียนข้อมูลเข้า EPROM และนำ EPROM นั้น ๆ มาใส่ยัง SOCKET ROM บนบอร์ดแล้วเปิดไฟเข้าตัวบอร์ดเพื่อ TEST โปรแกรม ซึ่งเป็นวิธีหนึ่ง แต่เรามีวิธีที่ดีกว่านั้นมากถ้านำมาต่อร่วมกับอุปกรณ์ของทาง อีทีที ในการพัฒนาระบบ



รูปที่ 2.7 แสดงตำแหน่ง EPOM

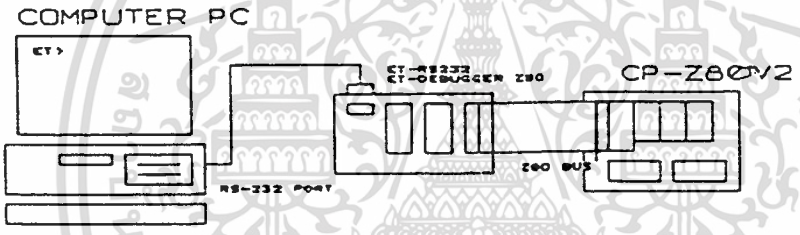
1) ต่อใช้กับ EPROM EMULATOR (ET-EM) เราสามารถเลือกต่อใช้กับชุด ET-EM8 หรือ ET-EM32 ในการพัฒนาเขียนโปรแกรมได้ ดังรูป



รูปที่ 2.8 แสดงการต่อกับ EPOM EMULATOR

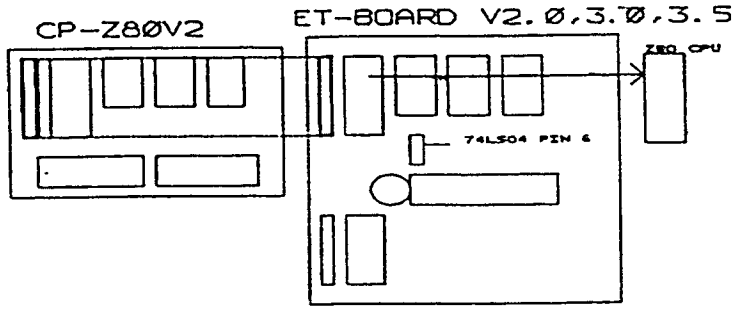
โดยเราสามารถเขียนโปรแกรมเป็นภาษา ASSEMBLER Z80 บนเครื่องคอมพิวเตอร์ PC แล้วให้เครื่องทำการแปลงเป็นภาษาเครื่อง จากนั้นใช้ ET-EM รับข้อมูลจากภาษาเครื่อง จากคอมพิวเตอร์ PC ส่งต่อมายังบอร์ด CP-Z80V1Viเป็น EPROM MONITOR PROGRAM ใช้ TEST การทำงาน จากนั้นเมื่อ TEST จนเป็นที่พอใจแล้วก็มาทำการ COPY เป็น EPROM ใช้งานจริง

2) ต่อใช้กับ ET-RS232 และ ET-DEBUGGER Z80 ต่อใช้ CP-Z80V1 กับ ET-RS232 และ ET-DEBUGGER Z80โดยใช้เครื่องคอมพิวเตอร์ PC ในการเขียนและพัฒนา-ระบบโดยเราจะสามารถใช้ PC คีย์ข้อมูลเข้าไปยัง PC-Z80V1 ผ่านทาง RS232 PORT ได้โดยตรงหรือจะทำการสั่ง RUN ได้จากเครื่อง PC ก็ได้ สามารถทำให้เราเหมือน กับมีชุด ซิงเกิ้ลบอร์ด PC-Z80V1 โดยใช้คีย์ และ จอของเครื่อง PC แทน



รูปที่ 2.9 แสดงการต่อกับ ET-RS 232

3) ต่อใช้กับ ET-BOARD เราสามารถต่อ CP-Z80V1 กับ ET-BOARD เข้าด้วยกันได้โดยต่อทาง Z80 BUS 40 PIN โดยใช้ชุด ET-BOARD นั้นเข้าครอบครองระบบ BUS ของ CP-Z80V1 ด้วยวิธีการถอดตัว CPU Z80 บนบอร์ด CP-Z80V1 ออกและถอน JUMPER J4(CLK) ออกและถอด IC หน่วยความจำที่อยู่ระหว่างบอร์ดทั้ง 2 ที่มีตำแหน่ง ADDRESS ตรงกันออกเสียก่อนด้วยจากนั้น ET-VOARD ก็สามารถครองระบบ BUS ของ CP-Z80V1 ได้

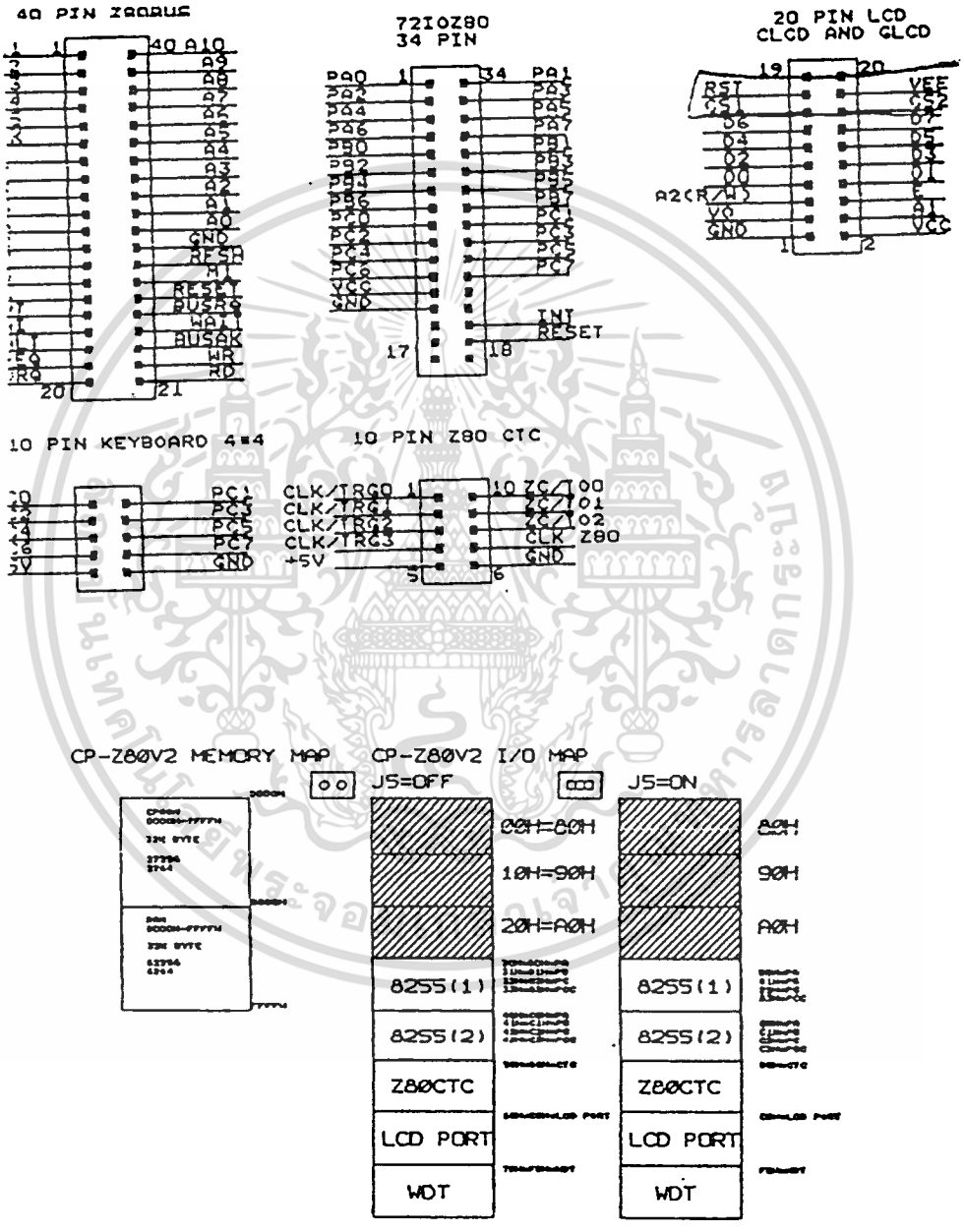


รูปที่ 2.10 แสดงการต่อกับ ET-BOARD

2.3_SPECIFICATION

CPU	: ZILOG Z84C00-6 (Z80 B CPU CMOS)
MEMORY	: ROM 2764,27256 (32K)
	: RAM 6264 (8K)
PORT	: 82C55 * 2 (PLCC 34 PIN)
	: Z80 CTC
LCD PORT	: 1 LCD MODVLE (DOT OR GRAGHE)
CLOCK RATE	: 4 MHZ (MAX 6MHZ)
POWER SUPPLY	: COMSUMPTION 5V DC & TERMINAL 5V DC
CONNECTOR	: 1 40 PIN EXPANSION HEADER-STRIP (Z80 BUS)
	: 2 34 PIN EXPANSION HEADER-STRIP (72 IO Z80 ETT)
	: 1 20 PIN EXPANSION HEADER-STRIP (LCD PORT ETT)
	: 1 10 PIN EXPANSION HEADER-STRIP (Z80 CTC)
	: 1 10 PIN EXPANSION HEADER-STRIP (KEY BOARD)
	: 1 3 PIN JUMPER (DOT/GRAPHIC LCD)
	: 1 3 PIN JUMPER (RAM 6264 / 62256)
	: 1 3 PIN JUMPER (EPROM 64/256)
	: 1 2 PIN JUMPET (ON/OFF BAT)
	: 1 2 PIN JUMPER (ON/OFF WDT)
	: 1 2 PIN JUMPER (DECODE PORT)
LED	: 1 POWER RED LED
	: 1 HALT GREEN LED
	: 1 PC 7 RED LED
PCB SIZE	: 11 CM X 9.5 CM

PIN_CONNECTOR



รูปที่ 2.11 แสดง PIN CONNECTOR

รายละเอียดอุปกรณ์ CP-Z80

IC + SEMI

1. Z80CPU (Z84C006)	1
2. 82C55 (PLCC 34 PIN)	2
3. 74 LS 04	1
4. 74HCT138	1
5. RAM 6264 (SK)	1
6. LM 555C	1
7. TTL 74HCT00	1
8. 7705	1
9. Z80 CTC	1
10. TTL 74LS32	1

CONNECTOR + อื่น ๆ

1. 40 PIN	1
2. 34 PIN	2
3. 20 PIN	1
4. 10 PIN	2
5. 3 PIN	3
6. 2 PIN TERMINAL	3

SOCKET

1. 14 PIN	3
2. 16 PIN	1
3. 28 PIN	1
4. 40 PIN	1
5. 8 PIN	1

R 1/8 W + C

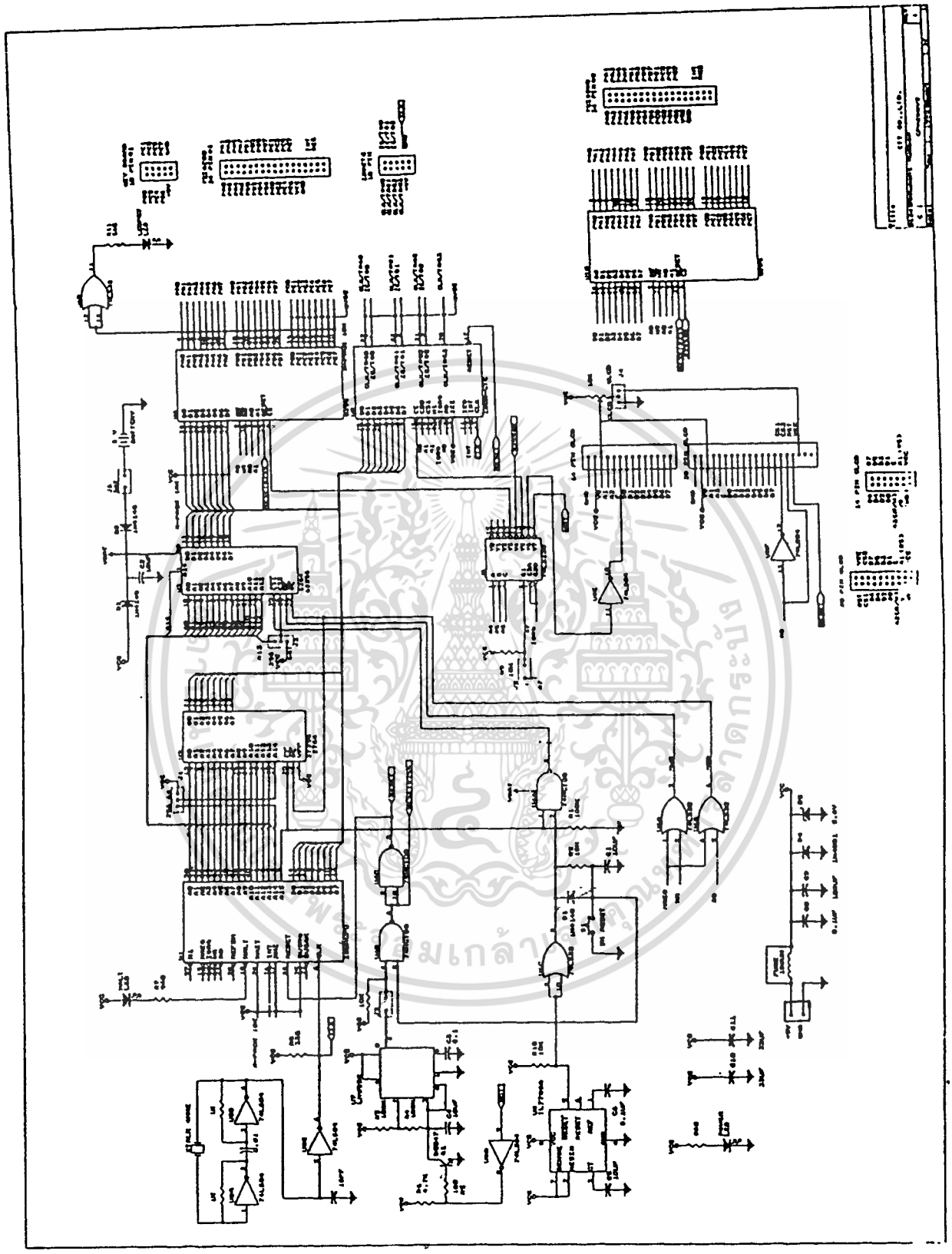
1. 330	1
2. 560	3
3. 1 K	2
4. 4.7	1
5. 10 K	5
6. 100 K	2
7. 100	1

C

1. C 0.1 uF MUT	13
2. 10 PF	1
3. C 0.01 uF	1
4. 10 uF 16V ELE	4
5. 100 uF 16V ELE	1
6. 33 uF 16V ELE	2

ทั่วไป

1. R PACK 10 K 9 ขา	2
2. R PACK 10 K 5 ขา	1
3. VR 10 K	1
4. LED 3 mm สีแดง	2
5. LED 3 mm สีเขียว	1
6. 1N4148	3
7. 1N4001	1
8. ZENER DIODE 5.6 V 1 W	1
9. X TAL 4 MHZ	1
7. MINI JUMPER	5
9. ลังถ่าน 3 V	1
10. TR BC 547	1
11. PCB CP-Z80 V2	1



รูปที่ 2.12 วงจรรวมของบอร์ด CP-Z80 V2

2.4 DOT MATRIX LCD MODULE

อุปกรณ์ในปัจจุบันนี้ในส่วนแสดงผลนั้นจะใช้ LCD เสียเป็นส่วนใหญ่ไม่ว่าจะเป็นเครื่องเล่น VEDIO , เครื่องถ่ายเอกสาร, เครื่องมือวัดคุมต่างๆ, เครื่องคอมพิวเตอร์

เราพอจะแบ่ง DOT MATRIX LCD MODULE นี้ออกได้เป็นพวกๆดังนี้

1. CHARACTER LCD MODULE
2. GRAPHIC LCD MODULE
3. SEGMENT DISPLAY TYPE LCD MODULE

โดยในแต่ละแบบนี้ก็จะมีส่วนประกอบใหญ่ๆแบ่งได้เป็น

1. DOT MATRIX LCD เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิดและเปิดตัวเองกับแสงก็คือส่วนของตัวกระจกบรรจุผลึก
2. DRIVER เป็นตัวรับสัญญาาณจากตัวควบคุมมาขับผลึก LCD อีกที่หนึ่งโดยมีเบอร์ที่นิยมใช้ใน LCD MODULE เช่น HD44100H, MSM5259
3. CONTROLLER เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาและจัดการควบคุม LCD MODULE ให้ทำงานแสดงผลต่างๆเช่น การลบจอภาพ, การเกิดตัวอักษร, เป็นต้น โดยมีเบอร์ IC ที่เป็นที่นิยมใช้กัน คือ HD4478 ซึ่งจะใช้ในแบบ CHARACTER LCD MODULE เป็นส่วนใหญ่และเบอร์ IC HD61830 จะใช้ในแบบ GRAPHIC LCD MODULE

ในการศึกษาการทำงานและใช้งาน LCD MODULE นั้นไม่ใช้เรื่องยากเลยถ้าเราสามารถทำความเข้าใจในส่วนของ CONTROLER ได้ก็เพียงพอแล้วและโดยมาก LCD MODULE .ในแต่ละบริษัทแล้วจะใช่ตัวอักษรหรือจำนวนบรรทัดก็มีการทำงานแบบเดียวกันทั้งหมด IC ที่นิยมมากที่สุดตัวหนึ่งที่เป็น CONTROLLER ก็คือเบอร์ HD44780 โดยรูปแบบการทำงานของมันได้เป็นมาตรฐานให้กับ CONTROLLER LCD ตัวอื่นๆด้วย

HD 44780 เป็นไอซี LSI ตัวหนึ่งใช้ควบคุม LCD โดยแสดงผลในรูปตัวอักษรหรือสัญลักษณ์ต่าง ๆ ตัวม้านเองสามารถต่อใช้งานแบบ 4 BIT หรือ 8 BIT จะต่อใช้งานที่ DB7-DB4 เท่านั้นโดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 BIT บน และข้อมูล 4 BIT ล่าง

จากวงจรเป็นการต่อ 8255 ให้เข้ากับ LCD โดยเราจะจำลองสัญญาณต่างๆ ขึ้นมา โดยการใช้ PORT A และ PORT B โดย PORT A นั้น เราให้เป็น DATA PORT และ PORT B นั้นให้เป็นสัญญาณควบคุมไปใช้เมื่อเราเริ่มเปิดไฟป้อนให้ HD44780 นั้นก็จะทำการ RESET ตัวมันเองโดยจะใช้เวลาประมาณ 10 ms หลังจากไฟ VDD ถึง 4.5 VOLT แล้ว โดยจะ SET ตัวเองดังนี้

1.DISPLAY CLEAR จะทำการลบข้อมูลจอภาพ LCD

2.FUNCTION SET โดยจะ SET ค่าภายใน

DL=1: เป็นการติดต่อแบบ 8 BIT

N =0: SET เป็น 1บรรทัดต่อการแสดงผล

F =0: 5X7 DOT ต่อหนึ่งตัวอักษร

3.DISPLAY ON/OFF D=0:DISPLAY OFF

C =0: CURSOR OFF

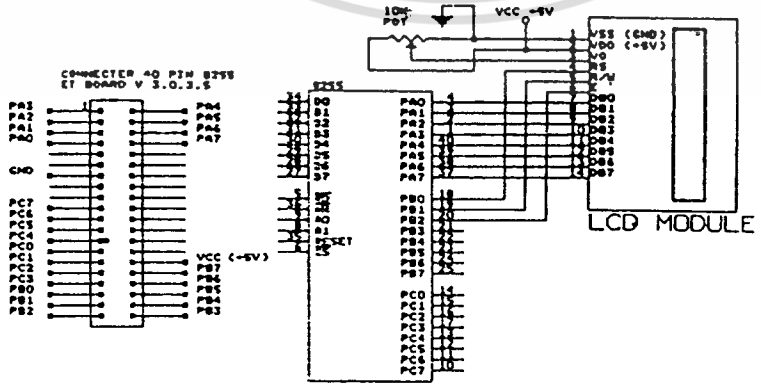
B =0:BLINK OFF

4.ENTRY MODE SET I/D=1:+1(เพิ่มค่า COUNTER ขึ้น 1)

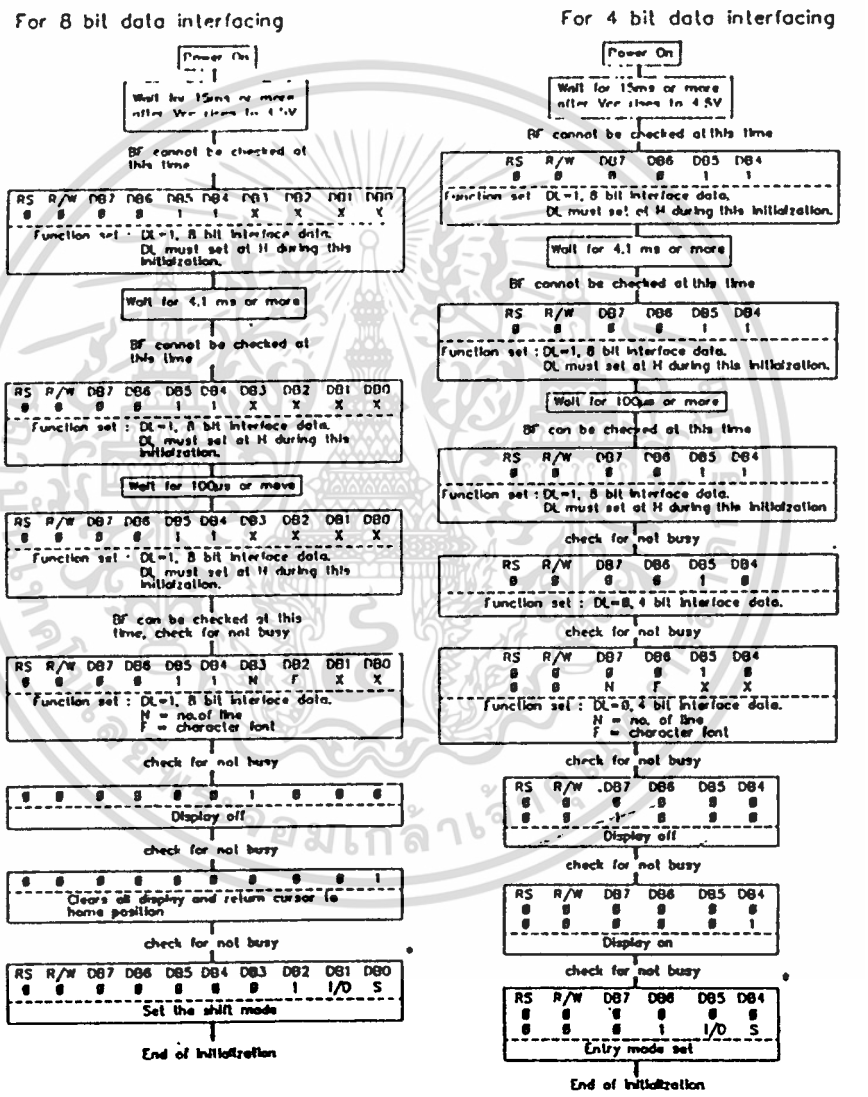
S=0:NO SHIFT

เมื่อเราเริ่มเปิดเครื่องทำงานแล้วก็จะต้องส่งค่าสั่งควบคุมให้มันเริ่มทำงานดังตาราง

ตัวอย่างการต่อใช้งานกับ ET-BOARD V2



รูปที่ 2.13 การต่อ LED กับ ET-BOARD



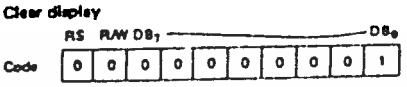
รูปที่ 2.14 แสดงลำดับขั้นตอนการ SET LCD

ตารางที่ 2.1 ตารางคำสั่ง HD 44780

Instruction	Code										Description	Execution Time (max) (when fcp or fosc is 250 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	1.64 ms
Return Home	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display being shifted to original position. DD RAM contents remain unchanged.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	40 μs
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and shifts display without changing DD RAM contents.	40 μs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (L) and character font (F).	40 μs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40 μs
Read Busy Flag & Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5×10 dots, F = 0: 5×7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM ACG: CG RAM address ADD: DD RAM Address: Corresponds to cursor address AC: Address counter used for both DD and CG RAM address.	Execution time changes when frequency changes Example: When fcp or fosc is 270 kHz: $40\mu s \times \frac{250}{270} = 37\mu s$

2.5 รายละเอียดของคำสั่ง HD44780

1). CLEAR DISPLAY



คำสั่งนี้จะเป็นการเขียนช่องว่างหรือ SPACE (ASCII 20H)เข้าไปใน DD RAM ทั้งหมดและทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ ตำแหน่งบนสุดซ้ายมือของภาพ SET I/D=1,S wไม่มีการเปลี่ยน

2). RETURN HOME

คำสั่งนี้จะทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพข้อมูลในจอภาพไม่เปลี่ยน

3). ENTRY HOME

Address	Function	Read/Write?
8000H	Write Command to LCD	Write Only
8001H	Write Data to LCD	Write Only
8002H	Read Status from LCD	Read Only
8003H	Read Data from LCD	Read Only
8004H to FFFFH	No Access	

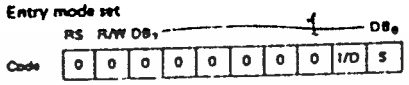
BIT I/D: โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียนหรืออ่านข้อมูลแล้วจะทำให้ DD RAM ADDRESS เพิ่มขึ้นหนึ่งหรือลดลงหนึ่งโดย

1= เพิ่ม

0=ลดลงหนึ่ง

BIT S: เป็นกำหนดแสดงผลโดยถ้า S=1 จะเป็นการใส่ข้อมูลแล้วตัว CURSOR อยู่กับที่ข้อมูลจะถูกดันไปทางซ้าย ถ้า S=0 ข้อมูลจะ อยู่กับที่ตัว CURSOR จะถูกดันไปทางขวามือ

4). DISPLAY ON/OFF CONTROL



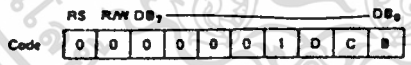
BIT D: เป็น BIT ให้เปิดปิดหน้าจอภาพโดยถ้า

D=1 จะON และ

D=0 จะ OFF

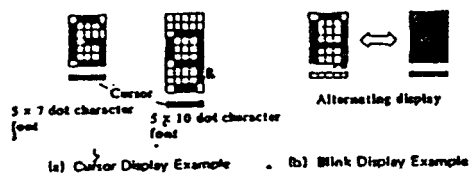
BIT C: จะให้แสดง CURSOR ให้ BIT C=1 และถ้าไม่ต้องการแสดงCURSOR BIT C=0 โดยตัว CORSOR จะอยู่ที่LINE ที่ 8 ในแบบ 5X7 DOTและจะอยู่ที่11ในแบบ5X10 DOT

BIT B: เป็น BIT SET การกระพริบของ CORSOR โดยB=1 การกระพริบ B=0 ไม่มีการกระพริบโดยมีระยะเวลาการกระพริบประมาณ 379.2ms



รูปที่ 2.15 ลักษณะ CURSOR และ ตัวอักษร

5). CURSOR OR DISPLAY SHIFT

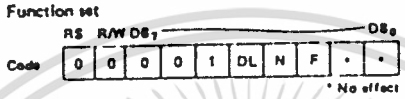


เป็นคำสั่งกำหนดให้ตำแหน่ง CURSOR หรือข้อมูลไปเกิดทางซ้ายหรือขวาโดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน โดย

S/C	R/L	
0	0	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปซ้ายมือ 1 ตำแหน่ง
0	1	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปขวามือ 1 ตำแหน่ง
1	0	เป็นการดันตัวอักษรที่เกิดไปทางซ้าย
1	1	เป็นการดันตัวอักษรที่เกิดไปทางขวามือ

ตารางที่ 2.2 ตารางการ Set S/C ,R/L

6). FUNCTION SET



BIT DL: เป็นการ SET การติดต่อว่าจะให้เป็นแบบ 8 BIT หรือ 4 BIT

โดยถ้าต้องการติดต่อ 4 BIT DL =0 และ 8 BIT DL =1

N : เป็นการ SET บรรทัดการแสดงผล N = 0 แสดง 1 บรรทัด

N = 1แสดง 2 บรรทัด ในกรณีมากกว่า 2 บรรทัด ก็ให้ SET N =1

F : เป็นการ SET ขนาด DOT การแสดงผล 5X7 หรือ 5X10 โดย

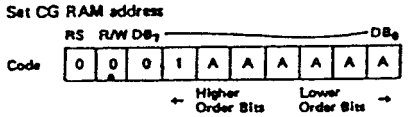
F = 0 เป็นแบบ 5X7 และ F = 1 เป็นแบบ 5X10

N F	No. of display lines	Character font	Duty factor	Remarks
0 0	1	5 x 7 dots	1/8	
0 1	1	5 x 10 dots	1/11	
1 •	2	5 x 7 dots	1/16	Cannot display 2 lines with 5 x 10 dot character font.

* No effect

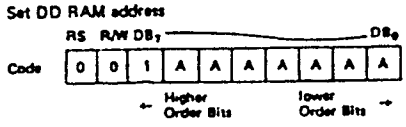
ตารางที่ 2.3 ตารางสภาวะ-NF

7). SET CG RAM ADDRESS



ใน HD44780 นั้นจะมีหน่วยความจำอยู่ 2 ชุด คือ DISPLAY DATA RAM (DD RAM) จำนวน 80 X8 BIT และ CHARACTER GENTERATOR ROM CG RAM จำนวน 512 BIT และ 7200BIT คำนี้จะเป็นการ SET ADDRESS ใน CG RAM โดยต้องทำการ SET ADDRESS ก่อนเขียนหรืออ่านข้อมูลจาก CG RAM ด้วย

8). SET DD RAM ADRESS



เป็นคำสั่ง SET ค่า ADDRESS ใน DD RAM ในการเขียนหรืออ่านค่าจาก DD RAM (DD RAM คือส่วนที่แสดงผลหหน้าจอ LCD) โดยจำนวน ADDRESS ที่จะเกิดขึ้นบนจอ LCD จะอยู่กับ SET ค่า N ด้วย

N = 0 (1 บรรทัด) ADDRESS จะอยู่ 00H-4FH

N = 1 (2 บรรทัด) ADDRESS จะอยู่ 00H-27H สำหรับบรรทัดที่ 1 และ 40H-67H สำหรับ บรรทัดที่ 2

แบบการจัด ADDRESS ของ DD RAM หน้าจอLCD แบบ 16 ตัวอักษร 1 บรรทัด, 16 ตัวอักษร 2 บรรทัด, 16 ตัวอักษร 4 บรรทัด, 20 ตัวอักษร 1 บรรทัด, 20 ตัวอักษร 2 บรรทัด, 40 ตัวอักษร 2 บรรทัด

16 ตัวอักษร 1 บรรทัด

00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47

16 ตัวอักษร 2 บรรทัด

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

16 ตัวอักษร 4 บรรทัด

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

20 ตัวอักษร 1 บรรทัด

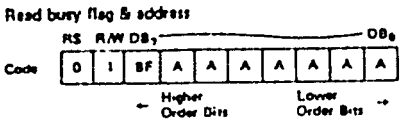
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13

20 ตัวอักษร 2 บรรทัด

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53

รูปที่ 2.16 การจัด ADDRESS ของ DD RAM

9). READ BUST FLAG AND ADDRESS



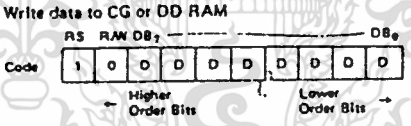
เป็นคำสั่งอ่านค่า BUSY FLAG ซึ่งจะเป็นตัวบอกว่าตัวHD44780 นี้อยู่ในขบวนการทำงานภายในอยู่หรืออยู่ในสภาพพร้อมจะรับข้อมูล โดย

BF=1 อยู่ในขบวนการทำงานภายในไม่พร้อมจะรับข้อมูลหรือคำสั่ง

BF=0 พร้อมจะรับข้อมูลหรือคำสั่งได้

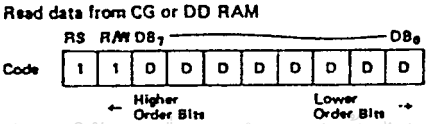
และนอกจากนี้ยังเป็นคำสั่งอ่ายข้อมูล ADDRESS ของ CG RAM หรือ DD RAM หรือ DD RAM ด้วย

10). WRITE DATA TO CG หรือ DD RAM



เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลและ ADDRESS จะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งที่ SET ใน ENTRY MODE ข้อกำหนดที่จะรู้ว่าเป็นการเขียนข้อมูลของ CG RAM หรือ DD RAM ทำได้โดยการ์ SET ADDRESS ของ CG RAM หรือ DDR RAM ขึ้นมาก่อนจะเขียนข้อมูล

11). READ DATA FROM CG OR DD RAM



เป็นคำสั่งอ่านข้อมูลจาก CG RAM หรือ DD RAM โดยก่อนอ่านจาก DD RAM หรือ CG RAM นี้ควรจะใช้คำสั่ง SET ADDRESS ก่อนเพื่อให้รู้ว่าข้อมูลที่อ่านได้นั้นเป็น DD หรือ CG RAM จากตารางการทำงานจะเห็นว่าการใช้งาน LCD MODULE นั้นง่ายเพียงแต่เราส่งคำสั่งเริ่มแรกและ SET ความต้องการในขนาดตัวอักษร, CURSOR หลังจากนั้นเราก็สามารถเขียนตัวอักษรเข้าไปใน DD RAM ตามตารางตัวอักษรที่ให้มานั้นก็จะเกิดอักษรในจอภาพ LCD เรายังสามารถกำหนดตำแหน่งตัวอักษรที่จะให้เกิดบนจอได้โดยการ SET DD RAM ADDRESS ตามตารางที่ให้มาในหัวข้อ SET DD RAM ADDRESS ขอให้ทดสอบทำความเข้าใจกับโปรแกรมที่ใช้กับ ET-BOARD V3.0, V3.5 นี้ที่ให้มาจะเห็นว่าจะมีส่วนเริ่มต้นก็คือ ส่วนการ INITIAL LCD เพื่อกำหนดหน้าที่การทำงานต่างๆ

2.6 ส่วนประกอบของโปรแกรม

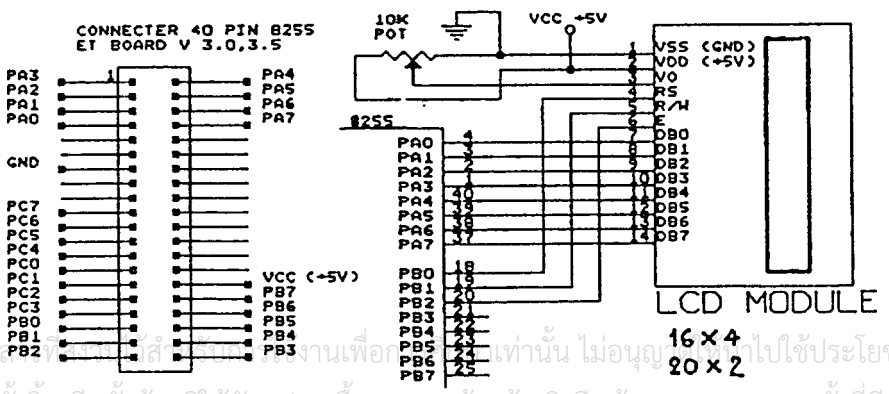
EPLUSE จะเป็นส่วนกำเนิดสัญญาณ ENABLE SIGNAL โดยการใช้ PORT B BIT ที่ 2 กำเนิด PLUSE สัญญาณ ENABLE ขึ้น

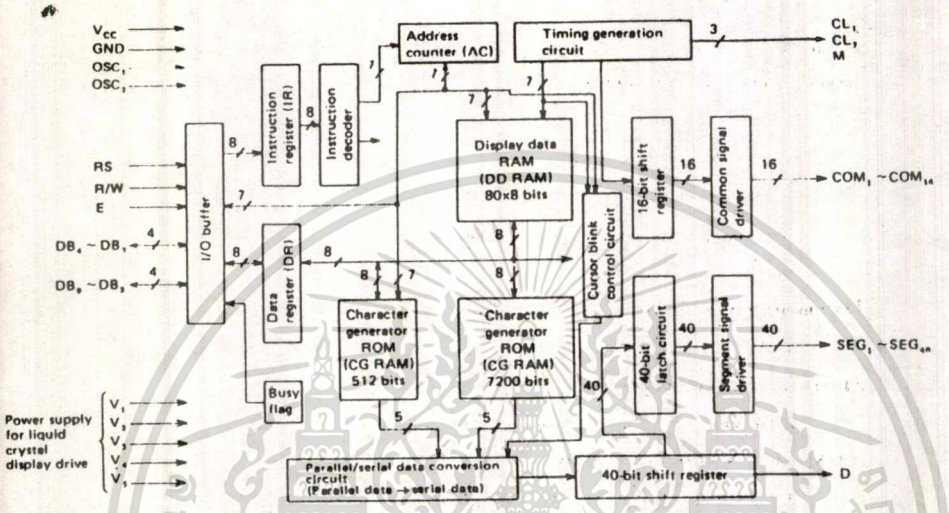
GOTO จะเป็นส่วนกำหนดตำแหน่งของส่วน DD RAM ADDRESS ที่จะเขียนข้อมูล โดยจากโปรแกรม INITIAL ที่เรา SET ไว้ เมื่อเขียน ข้อมูลเข้าไปใน DD RAM แล้ว ADDRESS ของ DD RAM จะเพิ่มขึ้น 1 โดยทันที

WRBYTE เป็นส่วนเขียนข้อมูล 1 BYTE เข้าไปในตำแหน่ง ADDRESS ของ DD RAMJ ขณะนั้นๆ

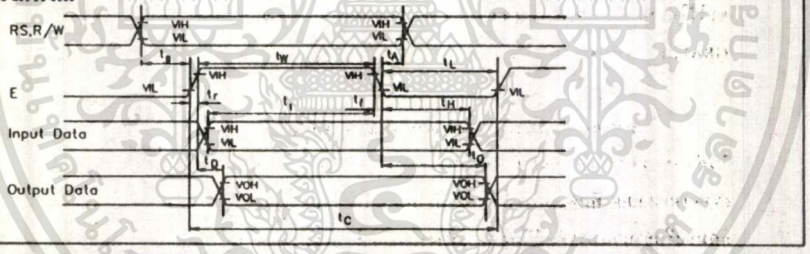
WRLINE เป็นส่วนในการคขียนข้อมูลทีละ 1 LINE เพระาตำแหน่ง DD RAM ที่เกิดบนจอภาพ LCD นั้นแต่ละตำแหน่งจะไม่ต่อกันไปในแต่ละบรรทัด

จากตัวอย่างที่ให้จะมี LCD แบบ 20 ตัวอักษร 2 บรรทัด และ 16 ตัวอักษร 4 บรรทัด





TIMING DIAGRAM



TIMING CHARACTERISTICS FOR ALL COMPATIBLE CONTROLLER CHIPS.

PARAMETERS \ CONTROLLERS CHIPS		SAMSUNG KS0066	HITACHI HD44780	SANYO LC7985NA	EPSON SED1278	OKI MSM6222	RECOMMENDED TIMING	UNIT
Enable Cycle Time	tC (min)	1000	1000	1000	500	667	1000	nS
Enable Pulse Width High Level	tW (min)	450	450	450	220	280	450	nS
Enable Pulse Width Low Level	tL (min)	450	450	450	220	280	450	nS
E Raise Time	tr (max)	25	25	25	25	25	25	nS
E Fall Time	tf (max)	25	25	25	25	25	25	nS
Set-up Time	tB (min)	140	140	140	40	140	140	nS
Data Set-up Time	tH (min)	195	195	195	60	180	195	nS
Data Delay Time	tD (max)	320	320	320	120	220	320	nS
Address Hold Time	tA (max)	10	10	10	10	10	10	nS
Hold Time Input Data	tH (min)	10	10	10	10	10	10	nS
Hold Time Output Data	tO (min)	20	20	20	20	20	20	nS

NOTE

1. [redacted]
2. MODULE INITIALIZATON DOES NOT AFFECT BY USING HD44100, OR KS0065, OR LC7930, OR MSM5259, OR MSM5839, OR MSM5260 DRIVER CHIPS.

รูป บล็อคไดอะแกรมและไทม์มิ้งไดอะแกรมของ HD44780

นอกจากนี้ LCD MODULE (HD44780) นี้จะยังมีส่วนหนึ่งของ CHARACTER GENRATOR ที่เราสามารถเขียนข้อมูลในการเกิดตัวอักษรขึ้นเองได้จากตารางตัวอักษร 5X7 DOT นั้นจะเห็นว่าคือตำแหน่งในตาราง 00H ถึง 07H ส่วนตำแหน่ง 08H-OFH จะเป็น ตำแหน่งเดียวกับ 00H-07H จะเห็นว่าจะมี CHARACTER GENERATOR 8 ตัวที่เราสามารถ เขียนข้อมูลกำหนดเองได้และถ้าเป็นแบบ 5X10 DOT จะเขียนได้ 4 ตัวอักษรซึ่งจากข้อพิเศษนี้ทำให้เราสามารถเขียนตัวอักษรและสัญลักษณ์หรืออักษรภาษาไทยได้

การเขียนข้อมูล CHARACTER GENERATOR

เราสามารถเขียนข้อมูลได้โดยกำหนด ADDRESS ของCG RAM ก่อนโดยเขียนได้ 64 ตำแหน่ง BIT 5-BIT 0 และเมื่อกำหนด ADDRESS แล้วก็ทำการเขียนข้อมูลลงใน CG RAM โดยเป็นลักษณะ BIT บนจอ 1 ตัวอักษร คือ 5X7 DOT นั้นจะใช้ข้อมูล BIT 4 ถึง BIT 0 ต่อ 1 BYTE เท่านั้น 1 ตัวอักษรจะใช้ข้อมูล 8 BYTE ด้วยกันให้ดูจากตาราง ประกอบไปด้วยและเมื่อเขียนข้อมูลลงใน CG RAM แล้วเวลาเราจะใช้งานก็ให้เขียนข้อมูลใน DD RAM คือ ข้อมูลตำแหน่งในตาราง CHARACTER ที่ตำแหน่ง OOH-07H ตัวอย่างโปรแกรมการเขียนข้อมูลตัวหนังสือภาษาไทยเป็นตัว(อ), (ท), และตัว ( ) เข้าไปใน CG RAM ตำแหน่งที่ 00H, และ 02H และนำมาแสดงผลทางจอ LCD โดยใช้ 2 บรรทัด ในการแสดงผล

สรุป การใช้งาน LCD MODULE นั้นที่สำคัญคือ ต้องเข้าใจในตัว CONTROLLER ของ LCD MODULE นั้น โดย CONTROLLER ทุกๆบริษัทจะมีการทำงานที่เหมือนกันเป็นส่วนใหญ่

2.7 ขาต่างๆในการต่อใช้งาน HD44780

1). RS (REGISTOR SELECTION) จะเป็นขาเลือก REGISTOR ภายในซึ่งมีอยู่ 2 ตัวคือ INSTRUCTION REGISTOR (IR) และ DATA REGISTOR (DR)

2). R/W(READ/WRITE) เป็นตัวเลือกว่าจะเขียนหรืออ่านข้อมูลจากตัว IC โดยอ่านข้อมูล =1, เขียนข้อมูล=0

3) .E (ENA BLE SIG NAL) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล

Register selection			
RS	R/W	E	Operation
0	0	↓	IR write as internal operation (Display clear, etc.)
0	1	↑	Read busy flag (DB_7) and address counter (DB_0 ~ DB_6)
1	0	↓	DR write as internal operation (DR to DD or CG RAM)
1	1	↑	DR read as internal operation (DD or CG RAM to DR)

ตารางที่ 2.4 กำหนด ENABLE SIGNAL

4). DBO-DB7 เป็นขารับส่งข้อมูลจากตัว IC

5). VDD ไฟเลี้ยงตัววงจร +5V

6).VSS เป็นขา GND

7).VO เป็นขารับ VOLTAGE ในการขับ LCD ให้สว่างหรือมืด

PIN CONNECTION

Pin No.	Symbol	Level	Function
1	V_{SS}	—	0V
2	V_{DD}	—	+5V
3	V_O	—	—
4	RS	H/L	L: Instruction code input H: Data input
5	R/W	H/L	H: Data read (LCD module → MPU) L: Data write (LCD module ← MPU)
6	E	H, H→L	Enable signal
7	DB0	H/L	Data bus line Note (1), (2)
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	

ตารางที่ 2.5 PIN CONNECTOR

เราสามารถต่อ VR ปรับค่าได้ 2 แบบ

1.ต่อ GND แบบธรรมดา

2.ต่อ ไฟลบ ในกรณี LCD บางรุ่นที่ต้องใช้ไฟลบ เช่น LM2017

*เราอาจประยุกต์ใช้ไฟลบได้จาก ขา 6 IC MAX232 ได้

บทที่ 3
พอร์ตสื่อสารอนุกรม

พอร์ตสื่อสารเป็นส่วนสำคัญส่วนหนึ่งที่มีอยู่บนไมโครคอมพิวเตอร์ 16 บิต พอร์ตสื่อสารนี้มีชื่อเรียกอีกอย่างหนึ่งว่า คอม-พอร์ต (com port) ผู้ออกแบบพอร์ตสื่อสารต้องการ ให้เป็นไปตามมาตรฐานการเชื่อมต่อแบบอนุกรมที่เรียกว่า RS232C พอร์ตนี้เป็นพอร์ตที่จำเป็น และผู้จัดซื้อไมโครคอมพิวเตอร์มักจะกำหนดอยู่ในสเปกด้วยเสมอ

พอร์ตสื่อสาร RS232C บนเครื่องไมโครคอมพิวเตอร์ 16 บิตมีโครงสร้างที่สามารถโปรแกรมด้วยการส่งรหัสคำสั่งให้กับชิปหลักได้ อย่างไรก็ตามผู้ออกแบบได้ให้ทางเลือกในการสื่อสารด้วยกระแสวนรอบ (current loop) หรือแบบแรงดันคือ RS232Cโดยใช้จัมเปอร์เพื่อเลือกระบบ สำหรับตัว RS232C นี้เป็นมาตราฐานแบบอะซิงโครนัสที่โปรแกรมสตาร์ตบิต สต๊อปบิตและพาริตี้บิต อัตราการส่งก็สามารถกำหนดได้ตั้งแต่ 50 บอดถึง 9600 บอต ลักษณะพิเศษของวงจรคือสามารถส่งสัญญาณมาอินเตอร์รัพต์ซีพียูตามเงื่อนไขได้ และยังมีโครงสร้างฮาร์ดแวร์ป้อนกลับเพื่อใช้ในการตรวจสอบระบบว่าทำงานปกติดีหรือไม่ได้อีกด้วย วงจรพอร์ตสื่อสารของไมโครคอมพิวเตอร์ 16 บิตนี้ใช้ไอซีหมายเลข 8250 เป็นตัวสำคัญของระบบ 8250เป็นไอซีขนาด 40 ขา มีขีดความสามารถพิเศษดังนี้

- มีบัฟเฟอร์ในตัวเพื่อทำให้ไม่จำเป็นต้องซิงโครนัสการรับส่ง
- ใช้สัญญาณนาฬิกาอิสระต่างหากไม่ขึ้นกับสัญญาณนาฬิกาของระบบ
- มีสัญญาณตอบโต้ควบคุมโมเดมทั้ง CTS (clear to send), RTS (request to send), DSR (data set ready), DTR (data terminal ready), RC (ring indicataor) และ สัญญาณดีเทคตัวพาหะ (carrier detect)

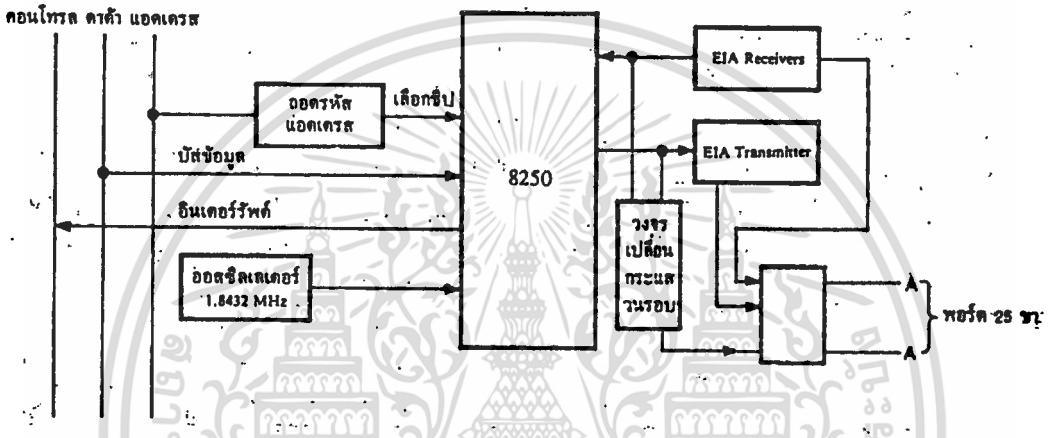
ตรวจสอบสตาร์ตบิตที่ผิดพลาด

ตรวจสอบและสร้างสัญญาณสายขาด (line break) เพื่อใช้ในการตรวจสอบการทำงานของระบบ

ซิป 8250 มีโครงสร้างการทำงานคล้ายกับ 8251 ที่นักฮาร์ดแวร์ทั่วไปรู้จักดี การทำงานของระบบ 8250 ต้องได้รับการโปรแกรมก่อน หลังจากนั้นจะทำงานตามรูปแบบที่ได้โปรแกรมไว้จนกว่าจะมีการโปรแกรมคำใหม่ อย่างไรก็ตาม มีขีดความสามารถในการทำงานได้สูงกว่า 8251 หลายอย่าง และด้วยเหตุนี้เอง 8250 จึงเป็นชิปประจำที่ใช้กับเครื่องไมโครคอมพิวเตอร์ 16 บิต

3.1 โครงสร้างของพอร์ตสื่อสาร

พอร์ตสื่อสารของเครื่องไมโครคอมพิวเตอร์ 16 บิต ที่จะกล่าวถึงนี้เป็นระบบมาตราฐานตามแบบเครื่องไอบีเอ็มพีซีเอ็กซ์ที โครงสร้างไดอะแกรมแสดงดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างพอร์ตสื่อสารข้อมูลที่ใช้ 8250

พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือสล็อตขนาด 31*2(62) ขาของระบบนั่นเอง ซีพียูติดต่อกับ 8250 ในลักษณะพอร์ตที่เป็นอินพุตเอาต์พุต การจัดพอร์ตนี้กำหนดหมายเลขพอร์ตอย่างเจาะจง ระบบไมโครคอมพิวเตอร์ 16 บิต มีพอร์ตสื่อสารสองพอร์ตคือ คอม1 (COM1) และคอม 2 (COM2) ทั้ง COM1 และ COM2 มีหมายเลขอินพุตเอาต์พุตพอร์ต ดังตารางที่ 3.1

การเลือกหมายเลขอินพุตเอาต์พุตพอร์ตแยกเป็นสองกลุ่ม กลุ่มหนึ่งคือ COM_1 จะกำหนดหมายเลขพอร์ตจาก 3F8 ถึง 3FE อีกกลุ่มหนึ่งถ้ากำหนดหมายเลขพอร์ตเป็น 2F2-2FE ในการเลือกหมายเลขรีจิสเตอร์ภายในกำหนดด้วยแอดเดรส 3 บิต คือ A_0 , A_1 และ A_2 สำหรับการเลือก COM_1 และ COM_2 เราใช้แอดเดรส A8 เป็นตัวเลือกการเลือกนี้ กำหนดเป็นตารางได้ดังตารางที่ 3.2

ตารางที่ 3.1 หลายเลขอินพุตพอร์ตของ COM1 และ COM2

อินพุตเอาต์พุตพอร์ต		เลือกรีจิสเตอร์	สถานะ DLAB
พอร์ต COM1	พอร์ต COM2		
3F8	2F8	บัฟเฟอร์ IX	DLAB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB = 0 (อ่าน)
3F8	2F8	แลดซ์ตัวหาร (LSB)	DLAB = 1
3F9	2F9	แลดซ์ตัวหาร (MSB)	DLAT = 1
3F9	2F9	อีนาเบิลอินเตอร์รัพต์	
3FA	2FA	กำหนดอินเตอร์รัพต์	
3FB	2FB	ควบคุมสายสื่อสาร	
3FC	2FC	ควบคุมโมเดม	
3FD	2FD	แสดงสถานะสายสื่อสาร	
3FC	2FE	แสดงสถานะโมเดม	

ตารางที่ 3.2 การกำหนดแอดเดรสสำหรับหมายเลขพอร์ตต่างๆ

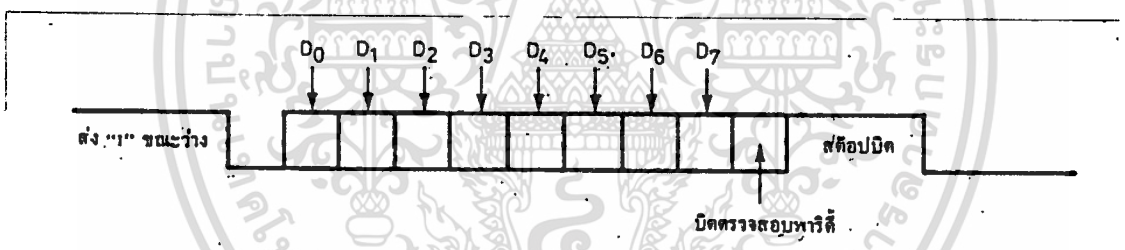
แอดเดรส 3F8-3FE และ 2F8-2FE											
A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	DLAB	rigister
1	1/0	1	1	1	1	1	X	X	X		
							0	0	0	0	บัฟเฟอร์สำหรับตัวรับข้อมูล (อ่าน) โฮลสดิงสำหรับตัวส่งข้อมูล (เขียน)
							0	0	1	0	อีนาเบิลอินเตอร์รัพต์
							0	1	0	X	กำหนดอีนาเบิลอินเตอร์รัพต์
							0	1	1	X	ควบคุมสายสื่อสาร
							1	0	0	X	ควบคุมโมเดม
							1	0	1	X	แสดงสถานะสายสื่อสาร
							1	1	0	X	แสดงสถานะโมเดม
							1	1	1	X	ไม่ใช้
							0	0	0	1	แลตช์ตัวหาร (LSB)
							0	0	1	1	แลตช์ตัวหาร (MSB)

3.2 การอินเตอร์รัพต์

จากที่เคยกล่าวถึงโครงสร้างการควบคุมอินเตอร์รัพต์ของชิป 8259 มาแล้ว 8259 ได้จัดลำดับการอินเตอร์รัพต์ไว้ 8 ระดับ สัญญาณรับอินเตอร์รัพต์ที่เข้าทางชิป 8259 มี 8 เส้น คือ IRQ0 - IRQ7 สำหรับกรณีของระบบสื่อสารอนุกรมได้ กำหนดสัญญาณการอินเตอร์รัพต์ไว้แล้ว คือให้ IRQ4 เป็นสัญญาณอินเตอรรัพต์ของระบบ COM1 และ IRQ3 เป็นของ COM2 ในการที่จะส่งสัญญาณอินเตอร์รัพต์ต้องให้บิต 3 ของรีจิสเตอร์ควบคุมโมเดมได้รับการเซตค่าเป็น "1" ก่อน จากนั้นข้อมูลอินเตอร์รัพต์ที่อยู่ในรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์จะเป็นตัวสั่งการอินเตอร์รัพต์

3.3 รูปแบบข้อมูลที่รับหรือส่ง

การสื่อสารข้อมูลของระบบนี้เป็นการสื่อสารแบบอะซิงโครนัส รูปแบบของข้อมูลจะมีสตาร์ตบิต บิตตรวจสอบพาริตี้ และสต็อปบิต โครงสร้างของข้อมูลแต่ละเฟรมเป็นดังรูปที่ 3.2

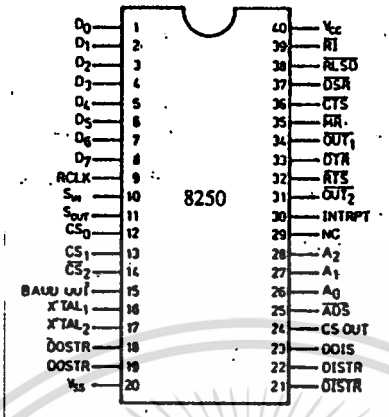


รูปที่ 3.2 รูปแบบของข้อมูล 1 เฟรม

ข้อมูลบิตแรกของการส่งเป็นสตาร์ตบิต ระบบจะส่งสตาร์ตบิตก่อน หลังจากนั้นจะตามด้วยข้อมูลและแทรกด้วยบิต ตรวจสอบพาริตี้ตามด้วยสต๊อปบิต ขนาดของข้อมูลมีค่าได้ตั้งแต่ 5-8บิต แต่ทั่วๆไปใช้ 8 บิตสต๊อปบิตมีได้ 1,1.5 หรือ 2 บิตค่าเหล่านี้กำหนดลงไปในรีจิสเตอร์ควบคุมสายสื่อสาร

3.4 ชิป 8250

8250 เป็นไอซีขนาด 40 ขา มีการทำงานเพื่อควบคุมสิ่งต่าง ๆ ที่เกี่ยวกับการสื่อสารแบบอนุกรมได้หมด โครงสร้างทางฮาร์ดแวร์ของอะแดปเตอร์การ์ดนี้จึงไม่ยุ่งยากมากนัก การจัดวางขาของไอซี 8250 มีรายละเอียดดังรูปที่ 3.3



รูปที่ 3.3 การจัดวางขาของไอซี 8250

ขาสัญญาณอินพุต ชิป CS0, CS1, CS2 (chip select) คือ ขา 12, ขา 13 และขา 14 ตามลำดับ เป็นสัญญาณเลือกชิป โดยที่เมื่อต้องการเลือกชิปจะให้ CS0 $\overline{CS1}$ เป็น "1" และ CS2 เป็น "0" สัญญาณเลือกชิปนี้จะได้รับการเลือกแลตช์ไว้ในขณะที่สัญญาณ $\overline{ADS}$ มีค่าเป็น "0" การเลือกชิปนี้จะมีไว้เพื่อให้ซีพียูติดต่อกับ 8250

ขาสโตรบข้อมูลอินพุต (data input strobe) คือ DISTR (ขา 22) $\overline{DISTR}$ (ขา21)เมื่อสัญญาณที่ DISTR เป็น "1" และ $\overline{DISTR}$ เป็น "0" ในขณะที่มีการเลือกชิปเป็นขณะที่ซีพียูจะอ้างข้อมูลจากรีจิสเตอร์ภายในที่ได้รับการกำหนดไว้แล้วมายังซีพียู สัญญาณนี้จึงเป็นสัญญาณอ่านข้อมูลหรือ read นั่นเอง

ขาสโตรบข้อมูลเอาต์พุต คือ DOSTR (ขา19) $\overline{DOSTR}$ (ขา18) เมื่อมีค่าเป็น "0" จะแอกทีฟเพื่อลดแลตช์ค่า A0-A1 เลือกรีจิสเตอร์ภายใน การเลือกรีจิสเตอร์จะทำขณะที่การเลือกชิปแอกทีฟอยู่

ขาเลือกรีจิสเตอร์ คือ A0(ขา26) , A1(ขา27) A2 (ขา 28) ตามลำดับเป็นตัวกำหนดแอดเดรสของรีจิสเตอร์ภายใน เพื่อให้ซีพียูทำการติดต่อกับ 8250 ตามค่ารีจิสเตอร์ที่กำหนดเพื่อการเขียนหรืออ่าน อย่างไรก็ตามการทำงานของ A0-A2 นี้ ก็ขึ้นกับสัญญาณเลือกตัวหาร LAB-divisor latch access bit ซึ่งกำหนดค่ารีจิสเตอร์ได้ดังตารางที่ 3.3

ตารางที่ 3.3 การกำหนดค่ารีจิสเตอร์

DLAB	A_2	A_1	A_0	register
0	0	0	0	บัฟเฟอร์สำหรับตัวรับข้อมูล (อ่าน)
0				โฮลดิงสำหรับตัวส่งข้อมูล (เขียน)
0	0	0	1	อีนาเบิลอินเตอร์รัพต์
X	0	1	0	กำหนดอินเตอร์รัพต์
X	0	1	1	ควบคุมสายสื่อสาร
X	1	0	0	ควบคุมโมเดม
X	1	0	1	แสดงสถานะสายสื่อสาร
X	1	1	0	แสดงสถานะโมเดม
X	1	1	1	ไม่ใช้
1	0	0	0	แลตช์ตัวหาร(LSB)
1	0	0	1	แลตช์ตัวหาร (MSB)

ขารีเซต (master reset) คือ MR (ขา35) เมื่อมีค่าเป็น "1" จะรีเซตการทำงานของชิป 8250 โดยทำให้ค่าต่าง ๆ ในรีจิสเตอร์ถูกเคลียร์ทั้งหมด (ยกเว้นบัฟเฟอร์ของตัวรับ ตัวส่งและตัวหาร) ขณะทำการรีเซตจะมีผลต่อสัญญาณเอาต์พุตด้วย ผลของการรีเซตแสดงไว้ในตารางที่ 3.3

ขาสัญญาณนาฬิกาตัวรับ (receiver clock) คือ RCLK (ขา9)เป็นขาที่ตัวรับสัญญาณนาฬิกาเพื่อกำหนดอัตราบอดสัญญาณนาฬิกานี้จะมีค่าเป็น 16 เท่าของที่นำมาใช้

ขาอินพุตข้อมูลอนุกรม (serial input) คือ SIN (ขา10) เป็นขารับข้อมูลอนุกรมจากสายส่งในการเชื่อมโยงการติดต่อสื่อสาร

ขาเคลียร์ทูเซนต์ (clear to send) คือ $\overline{CTS}$ (ขา36) เป็นสัญญาณที่ใช้ในการติดต่อกับโมเดม เงื่อนไขของสัญญาณนี้สามารถเก็บไว้ภายในชิป 8250 ที่จะให้ซีพียูอ่านไปตรวจสอบได้โดยเก็บไว้ที่บิต4 ของรีจิสเตอร์ แสดงสถานะโมเดมส่วนบิตของรีจิสเตอร์แสดงสถานะจะเป็นตัวบอกว่า CTS ได้เปลี่ยนสถานะไปหลังจากการอ่านครั้งก่อนแล้วหรือไม่

ขาดาต้าเซตรีดี้ (data set ready) คือ DSR (ขา37) เมื่อเป็น "0" จะแสดงว่าโมเดมหรือข้อมูลได้รับการเซตเตรียมพร้อมแล้วสำหรับการเชื่อมต่อกับสายสื่อสาร และส่งข้อมูลระหว่าง 8250 กับโมเดม สัญญาณ DSR สัญญาณอินพุตของ 8250 ที่ซีพียูสามารถอ่านไปดูได้ทางบิตที่ 5 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 1 ของรีจิสเตอร์แสดงสถานะเป็นตัวบอกว่าสัญญาณ DSR ได้เปลี่ยนสถานะไปหลังจากที่อ่านครั้งก่อนแล้วหรือไม่

หมายเหตุ ทั้ง $\overline{CTS}$ และ $\overline{DSR}$ เมื่อมีการเปลี่ยนสถานะและถ้าได้รับการอินาเบิลที่ modem status interrupt จะส่งผลในการสร้างสัญญาณอินเตอร์รัพต์

ขาตรวจสอบสายสื่อสาร(received line signal detect)คือ $\overline{RLSD}$ (ขา38) ถ้าเป็น "0" หมายถึงแอกทีฟ คือ 8250 รับสัญญาณเตรวจสอบสัญญาณพาหะจากโมเดมว่า โมเดมตรวจสอบได้แล้ว หรือข้อมูลได้รับการเซตแล้ว ซีพียูสามารถตรวจสอบสัญญาณนี้ทางบิต 7 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 3 จะเป็นบิตที่แสดงสถานะว่าสัญญาณนี้ได้รับการเปลี่ยนแปลงหลังจากอ่านไปแล้วหรือยัง

ขาแสดงวงจรเรียก (ring indicator) คือ RI (ขา39) สัญญาณนี้แอกทีฟด้วยลอจิก "0" เป็นสัญญาณที่ส่งมาจากโมเดม โมเดมตรวจสอบสัญญาณการเรียก (ringing) สัญญาณนี้ตรวจสอบได้ทางบิต 6 และดูสถานะการเปลี่ยนหลังจากอ่านแล้วจากบิต 2

ขาไฟเลี้ยง คือ VCC (ขา40), VSS (ขา20) เป็นสัญญาณจากแหล่งจ่ายไฟเลี้ยง 5 โวลต์ และกราวด์

ขารีเควสต์ทูเซน (request to sent) คือ $\overline{RTS}$ (ขา32) เริ่มขานี้มีลอจิกเป็น "0" หมายความว่า 8250 พร้อมที่จะส่งข้อมูลแล้ว สัญญาณขานี้จะได้รับการเซตให้แอกทีฟด้วยการโปรแกรมค่าลงไปในรีจิสเตอร์ควบคุมบิตที่1

ขาเอาต์พุต 1 คือ $\overline{QUTI}$ (ขา34) เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอกทีฟเป็น "0" ด้วยการโปรแกรมลงไปในบิตที่ 2 ของรีจิสเตอร์ควบคุมที่บิต1

ขาเอาต์พุต 2 คือ $\overline{QUT2}$ (ขา31) เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอกทีฟเป็น "0" ด้วยการโปรแกรมลงไปในบิต 2 ของรีจิสเตอร์ควบคุมโมเดมทางบิต3

ขาเลือกชิปเอาต์ (chip select out) คือ CSOUT (ขา24) เมื่อมีค่าเป็น "1" จะบอกว่าชิปนี้ได้รับการเลือกชิปโดยซีพียูทางขา CS0, CS1 และ CS2

ขาไดรฟ์เวอร์ดีสเอเบิล (driver disable) คือ DDIS (ขา23) เป็นลอจิก "0" เมื่อซีพียูกำลังอ่านข้อมูลจาก 8250 สัญญาณ DDIS เป็น "1" มีไว้สำหรับการดิสเอเบิลการรับส่งภายนอก ในกรณีที่ใช้ 8250 กับซีพียูผ่านทางบิต D0-D7 เพื่อบอกเวลาว่าซีพียูเป็น 8250 ติดต่อกันอย่างไร

ขาสัญญาณกำหนดบอด คือ BAUDOUT (ขา15) เป็นสัญญาณนาฬิกาที่มีความถี่เป็น 16 เท่าของสัญญาณนาฬิกาแล้วหารด้วยค่าที่โปรแกรมกำหนดในตัวหาร

ขาอินเตอร์รัพต์คือINTRPT(ขา30) เป็น"1"เป็นการส่งสัญญาณอินเตอร์รัพต์ออกไปจาก 8250

ขาข้อมูลเอาต์พุต คือ SOUT (ขา11) เป็นขาที่ใช้ส่งข้อมูลอนุกรมออกไปยังสายสื่อสาร

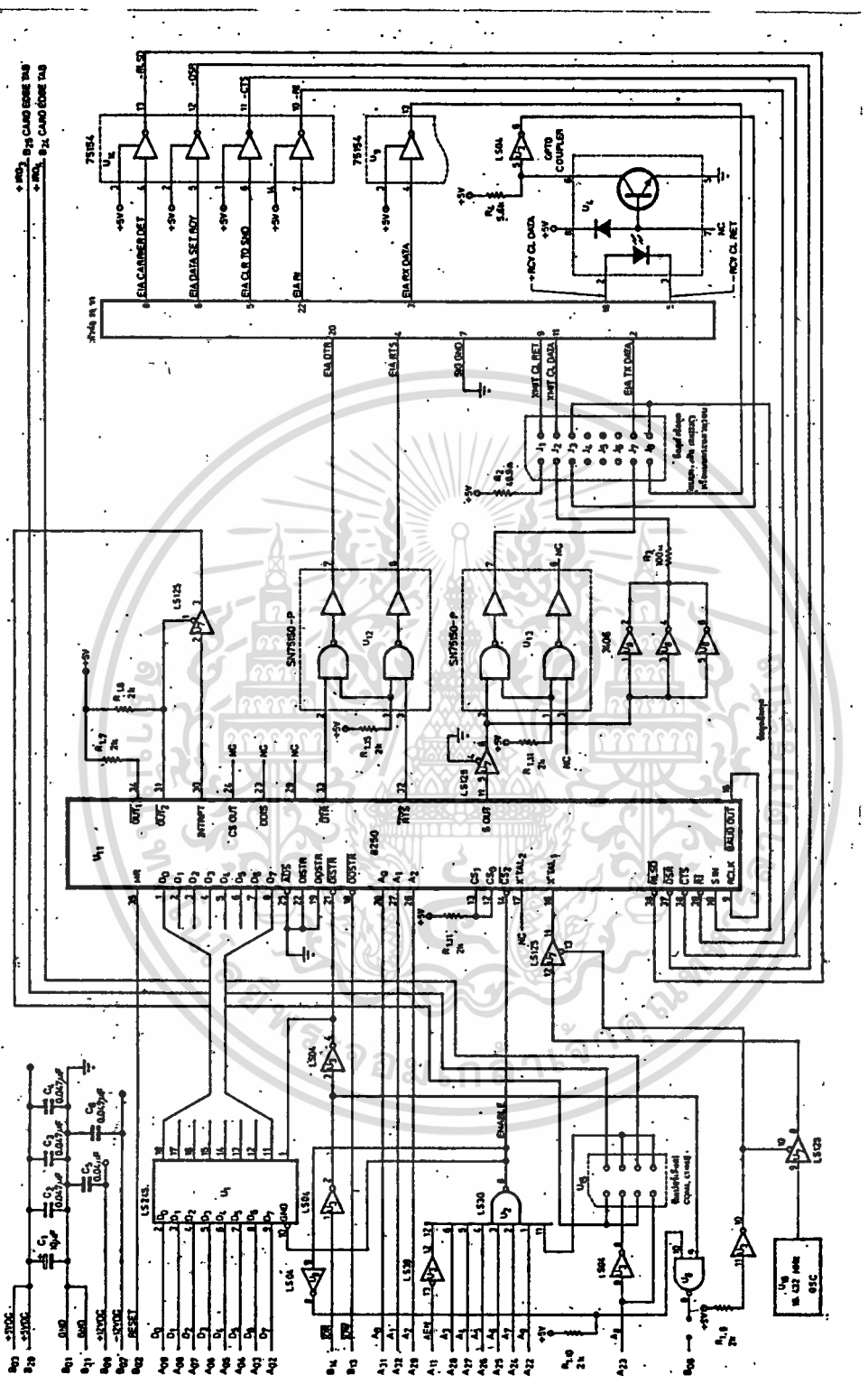
ขาสัญญาณอินพุต/เอาต์พุต ข้อมูล D6-D7 เป็นสัญญาณต่อเชื่อมกับบัสข้อมูลของระบบขาสัญญาณ XTAL1 , XTAL2 คือ ขา 16, ขา17 เป็นขาต่อกับคริสตอลเพื่อสร้างสัญญาณนาฬิกา

3.5 สถานะของ 8250 เมื่อเริ่มต้น

ก่อนการทำงานหรือโปรแกรมเราควรจะทราบว่าสถานะต่าง ๆของ 8250 เป็นอย่างไร โดยเฉพาะอย่างยิ่งเมื่อเริ่มเปิดเครื่อง จะมีสัญญาณรีเซตซึ่งเป็นสัญญาณเดียวกับซีพียูมาทำการรีเซต 8250 8250 ขณะนั้นจะมีสถานะเป็นอย่างไรเอาต์พุตของวงจรจะให้สัญญาณอะไรตารางที่จะให้รายละเอียด 8250 ขณะเริ่มต้นมีดังตารางที่ 3.4

ตารางที่ 3.4 ค่าเริ่มต้นและเอาต์พุตของ 8250

register / สถานะ	การควบคุม	สถานะเมื่อรีเซต
อินเตอร์รัพต์รีจิสเตอร์ภายใน	มาสเตอร์รีเซต	ทุกบิตเป็น "0" (0-3ถูกกำหนด 4-7 จะเป็นถาวร)
register กำหนดอินเตอร์รัพต์	มาสเตอร์รีเซต	บิต0 จะเป็น"1"บิต1-2เป็น "0"บิต 3-7 เป็น "0"ถาวร
register ควบคุมสายสื่อสาร	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
register ควบคุมโมเดม์	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
register แสดงสถานะสายสื่อสาร	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
register แสดงสถานะโมเดม	มาสเตอร์รีเซต	บิต 0-3 เป็น "0"บิต 4-7 เป็นสัญญาณอินพุต
SOUT	มาสเตอร์รีเซต	เป็น "1"
INTRPT (RCVR Errors)	Read LSR/มาสเตอร์รีเซต	เป็น "0"
INTRPT (RCVR Data Ready)	Read RBR/มาสเตอร์รีเซต	เป็น "0"
INTRPT (เปลี่ยนสถาะโมเดม)	Read MSR/Write THR/มาสเตอร์รีเซต	เป็น "0"
OUT2	มาสเตอร์รีเซต	เป็น "1"
RTS	มาสเตอร์รีเซต	เป็น "1"
DTR	มาสเตอร์รีเซต	เป็น "1"
OUT1	มาสเตอร์รีเซต	เป็น "1"



รูปที่ 3.4 วงจรของบอร์ดอะแดปเตอร์สื่อสาร

การต่อวงจรเข้ากับระบบ

บอร์ดอะแดปเตอร์สื่อสารนี้มีโครงสร้างการเชื่อมต่อตามพอร์ต 3F8-3FE และ 2F8-2FE ดังนั้นจะใช้บิต A8 เป็น ตัวเลือกบนบอร์ตจะมีจัมเปอร์เพื่อจะบอกว่าเป็นพอร์ตสื่อสารแบบ COM1 และ COM2 การเลือกแอดเดรสใช้ 74LS30 ซึ่งเป็น NAND เกตแบบ 8 อินพุตมาเป็นตัวเลือกโดยมี U15 ในรูปเป็นตัวเลือก A1 ดังได้กล่าวแล้ว

ส่วนของบัสข้อมูล D0-D7จะผ่านบัฟเฟอร์คือ 74LS245 ก่อนเข้าสู่ 8250 ส่วนสัญญาณควบคุมบน 8250 ทั้ง 10 มีดังนี้

ขามาสเตอร์รีเซต MR จะต่อโดยตรงกับสัญญาณรีเซตของระบบ

ขา ADS, DISTR, DOSTRต่อลงกราวด์ทั้งนี้เพราะแอดเดรสที่ต่อมาที่ชิป 8250 นี้เป็นสัญญาณแอดเดรสล้วนแล้วไม่ต้องสโตรบอีก

ขา DISTR ต่อกับ IOR ของระบบโดยผ่านอินเวอร์เตอร์ 2 ตัว

ขา DOSTR ต่อกับ IOW ของระบบ

ขา CS1 CS2 ต่อขึ้นเป็นลอจิก "1"

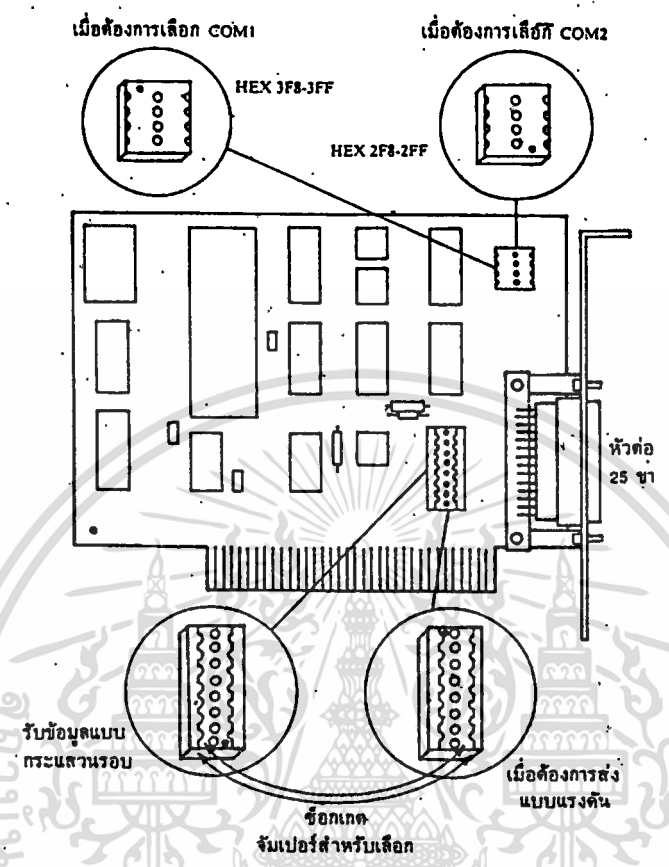
ส่วนขา CS2 มาจากการถอดรหัสให้เป็นพอร์ตของ COM1 หรือ COM2 ตามต้องการ

ขา XTAL2 ไม่ใช้ ส่วน XTAL1 ต่อมาจากออสซิลเลเตอร์ 18.432 MHz ของระบบวงจรของบอร์ดอะแดปเตอร์สื่อสารเป็นดังรูปที่ 3.4

ส่วนของสัญญาณเอาต์พุตประกอบด้วย สัญญาณควบคุมโมเดม RLSD, DSR, CTS, RI ต่อออกไปยังหัวต่อตามมาตราฐาน EIA RS232

ขา SIN, SOUT ต่อออกไปขาเอาต์พุตเช่นกันแต่มีการปรับให้เป็น สัญญาณแรงดันตามมาตราฐาน EIA หรือเลือกส่งเป็นกระแสวนรอบก็ได้ การเลือกใช้จัมเปอร์ I1-I8 เป็นตัวเลือกการแปลงกระแสเป็นแรงดันใช้ออปโตคัปเปลอร์ดังรูปที่ 3.4

การเลือกจัมเปอร์บนบอร์ดอะแดปเตอร์นั้น ผู้ออกแบบบอร์ดทำให้ง่ายต่อการใช้ ด้วยการให้จัมเปอร์มา เพียงผู้ใช้เลือกทิศทางก็จะได้พอร์ต COM1 หรือ COM2 หรือถ้าเลือกจัมเปอร์อีกตัวก็เลือกการส่งแบบกระแสหรือแรงดันได้ ลักษณะของหัวต่อจัมเปอร์แสดงดังรูปที่ 3.5

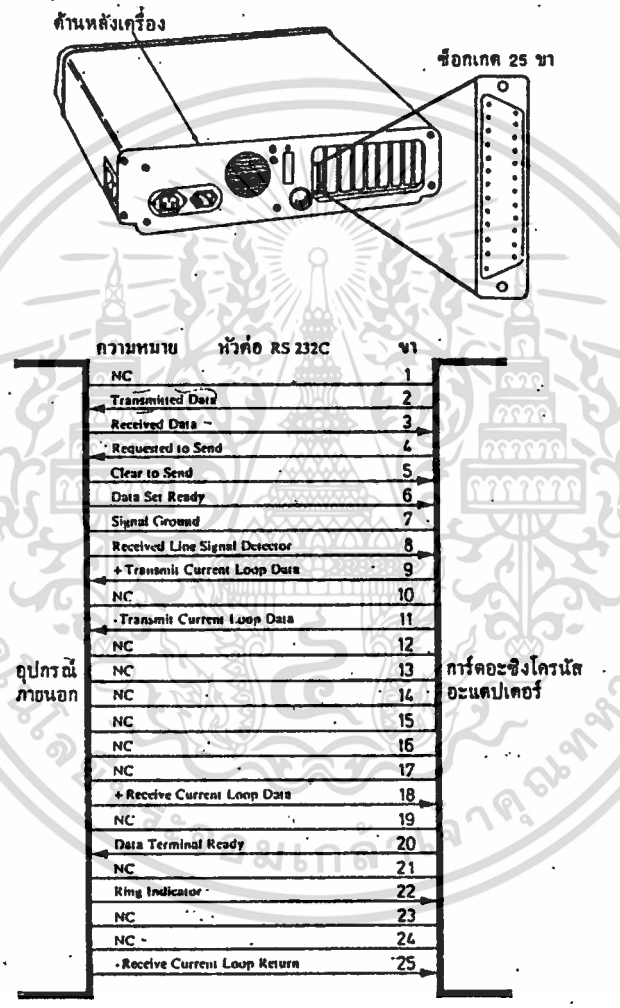


รูปที่ 3.5 การเลือกใส่จัมเปอร์เพื่อกำหนดวงจรตามต้องการ

ขาต่าง ๆของหัวต่อ 25 ขา RS232C ที่ใช้กันนั้นมีการจัดเรียงสัญญาณตามมาตรฐานสากล ขั้วต่อของสายแต่ละขาแสดงได้ดังรูปที่ 3.6

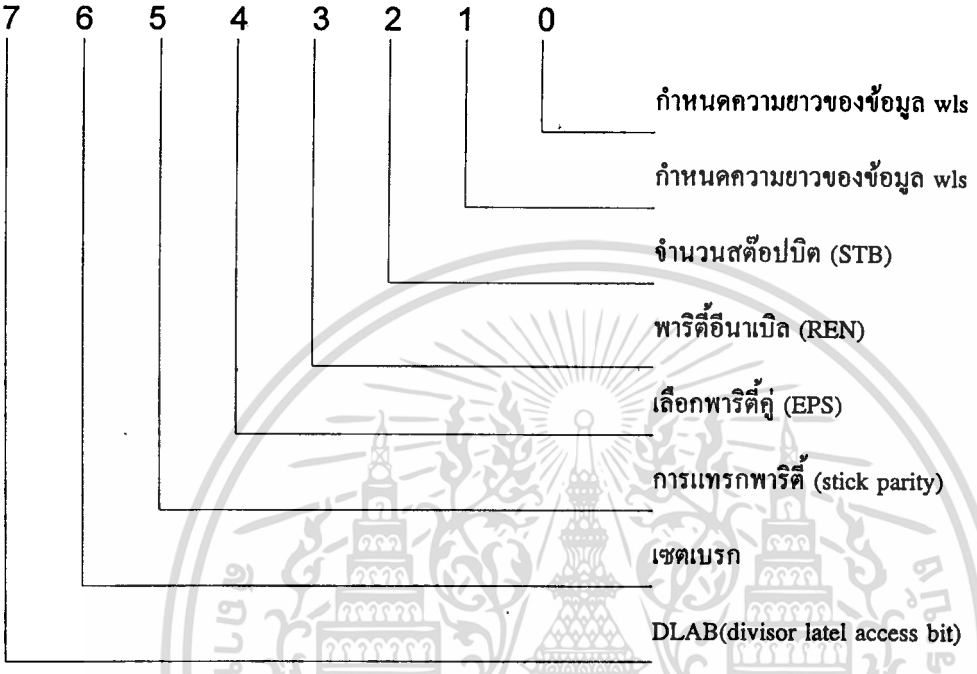
3.7 การใช้งานรีจิสเตอร์ต่าง ๆบน 8250

เพียงหลังจากการใช้งานบอร์ดอะแดปเตอร์สื่อสารที่จะต้องโปรแกรมค่าไมโครโค้ดเข้าไปใน 8250 ก่อนดังนั้นผู้ใช้งาน 8250 จำเป็นต้องเข้าใจว่ารีจิสเตอร์ของ 8250 มีความหมายอย่างไรอธิบายได้ดังนี้



รูปที่ 3.6 วงจรต่าง ๆ ของหัวต่อ D-Shell 25 ขา

รีจิสเตอร์ควบคุมสายสื่อสาร (line control register) ในการควบคุมรูปแบบของข้อมูลอะซิงโครนัส ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจิสเตอร์ควบคุม สายสื่อสารรีจิสเตอร์ตัวนี้โดยที่แต่ละบิตมีความหมายดังนี้



รูปที่ 3.7 ค่าของบิตในรีจิสเตอร์ควบคุมสายการสื่อสาร

บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	ความหมาย
0	0	หมายถึงข้อมูลขนาด 5 บิต
0	1	หมายถึงข้อมูลขนาด 6 บิต
1	0	หมายถึงข้อมูลขนาด 7 บิต
1	1	หมายถึงข้อมูลขนาด 8 บิต

บิต 2 เป็นบิตที่ใช้ในการกำหนดจำนวนสต๊อปบิต ถ้าเป็น "0" หมายถึงใช้สต๊อปบิต 1 บิตแต่ถ้าบิต 2 เป็น "1" ในกรณีส่งแบบ 5 บิตจะมีความยาวของสต๊อปบิตเป็น 1.5 บิต แต่ถ้าส่งแบบ 6,7 หรือ 8 บิตความยาวของสต๊อปบิตจะเป็น 2

บิต 3 บิตนี้เป็นบิตแสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้ โดยถ้าบิตนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้

บิต 4 มีค่าเป็น "0" และบิต 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะเป็นพาริตี้คู่

บิต 5 เมื่อบิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" และบิต 4 มีค่าเป็น 1" จะมีการแทรกหรือตรวจสอบพาริตี้ (stick parity) ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" จะมีการกำหนดบิตพาริตี้เป็น "1"

บิต 6 เป็นบิตที่ควบคุมการเบรก เมื่อบิต 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด

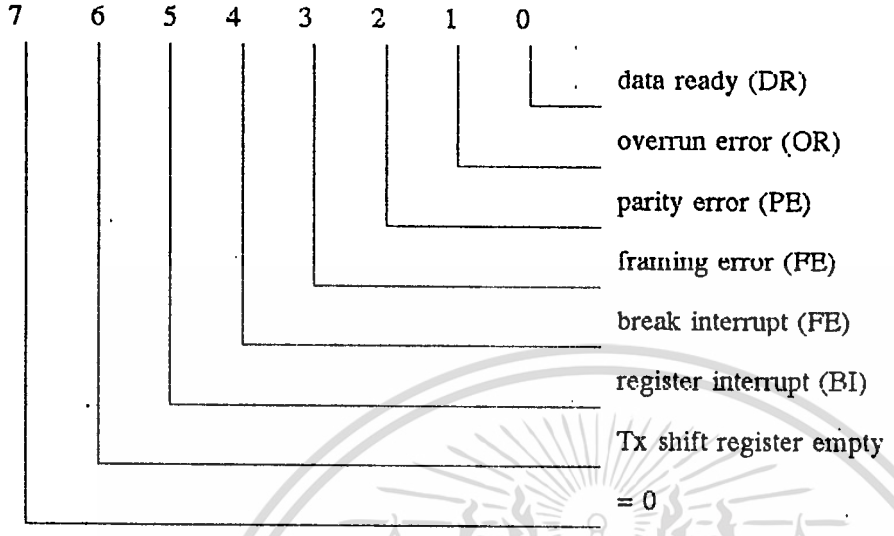
บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิต ที่จะมีผลต่อการแลตซ์ตัวหารดังที่กล่าวมาแล้วจากตารางที่ 3.1 และตารางที่ 3.2

การโปรแกรมอัตราบอด(boud rate generator)อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 Mhz และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1- (216-1) ค่าความถี่เอาต์พุตของตัวกำหนดอัตราบอดมีค่าเท่ากับ 16*อัตราบอด ดังนั้นตัวหาร = ความถี่สัญญาณนาฬิกา / (อัตราบอด*16) การกำหนดอัตราบอดด้วยการกำหนดตัวหารจึงเป็นค่าที่กำหนดในรีจิสเตอร์ 2 ตัว ตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ DLAB =1 แล้วให้ลดมาในรีจิสเตอร์ 3F8 ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F9 เมื่อ DLAB =1 จะเป็นค่าของตัวหาร MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 Mhz เป็นดังตารางที่ 3.5

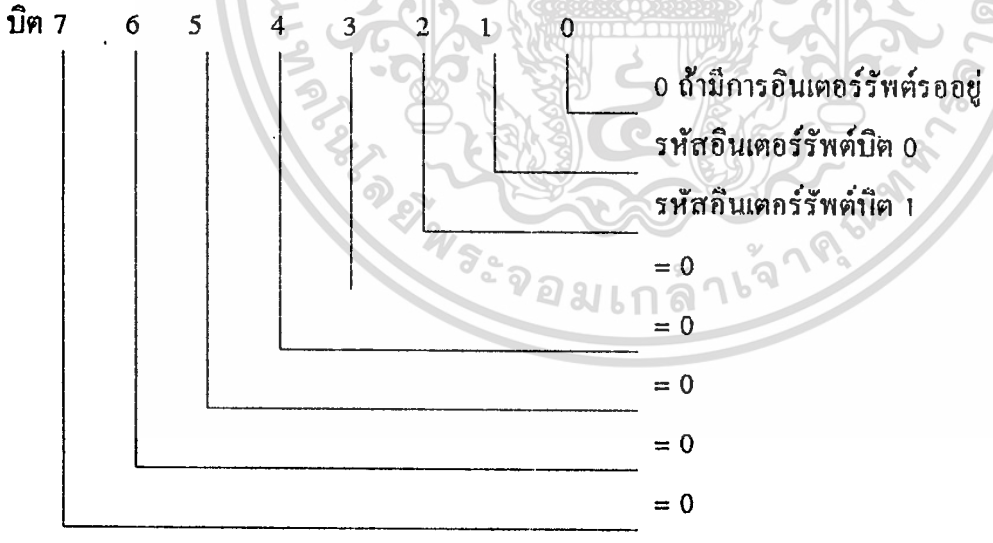
ตารางที่ 3.5 ค่าตัวหารสำหรับการกำหนดอัตราบอด

อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

รีจิสเตอร์แสดงสถานะสายสื่อสาร (line status register) รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบิตต่าง ๆในรีจิสเตอร์เป็นดังนี้



รูปที่ 3.8 ค่าของบิตแสดงสถานะสายสื่อสาร



รูปที่ 3.9 ค่าของบิตในรีจิสเตอร์กำหนดอินเตอร์รัพต์

บิต 0 บิตนี้เป็นบิตที่บอกสถานะการรับข้อมูล ถ้าบิตนี้เป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น"0"เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว หรือจะให้ซีพียูเขียนข้อมูลกลับมายังรีจิสเตอร์นี้ก็ได้

บิต 1 บิตนี้ถ้ามีค่าเป็น"1"แสดงว่าเกิด overrun error (OR)กล่าวคือขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไป ปรากฏว่ามีข้อมูลชุดใหม่มาเขียบทับบนบัฟเฟอร์นี้ บิตนี้จะรีเซต โดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 2 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด parity error (PE) กล่าวคือถ้ามีการตรวจสอบบิตพาริตี้แล้วไม่เป็นไปตามที่กำหนดไว้ บิตนี้จะได้รับการรีเซตโดยซีพียู เมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 3 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่น ตรวจสอบจำนวนบิตโดยดูที่พาริตี้และสต๊อปบิตไม่เป็นไปตามที่กำหนด

บิต 4 บิตนี้เรียกว่า break interrupt (BI) บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" ถ้าหากว่ารับข้อมูลอินพุตเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิต 5 บิตนี้เป็นบิตที่บอกว่า 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสาร บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" บิตนี้ยังคงสร้างสัญญาณ์อินเตอร์รัพต์เพื่อส่งไปบอกซีพียูด้วย บิตนี้จะมีสถานะเซตเมื่อมีการส่งถ่ายข้อมูลจากโฮลด์รีจิสเตอร์ไปยังชิปรีจิสเตอร์เพื่อพร้อมที่จะส่ง

บิต 6 เป็นบิตที่จะบอกว่า ชิปรีจิสเตอร์ว่างเปล่า บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิต 7 จะเป็น "0" ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพต์ (IIR-interrupt indentification register)

ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพต์ภายในชิป เพื่อว่าให้การทำงานระหว่าง 8250 กับซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และเพื่อให้ผู้เขียนซอฟต์แวร์สามารถเขียนซอฟต์แวร์ได้ง่ายและสั้นลงได้มาก 8250 กำหนดความสำคัญของอินเตอร์รัพต์ไว้ 4 ระดับคือระดับแรกสถานะการรับข้อมูลจากสายสื่อสาร ระดับที่สอง-การพร้อมรับข้อมูล ระดับที่สาม-ขณะรีจิสเตอร์โฮลดิงสำหรับส่งข่าว ระดับที่สี่-สัญญาณสถานะโมเดม

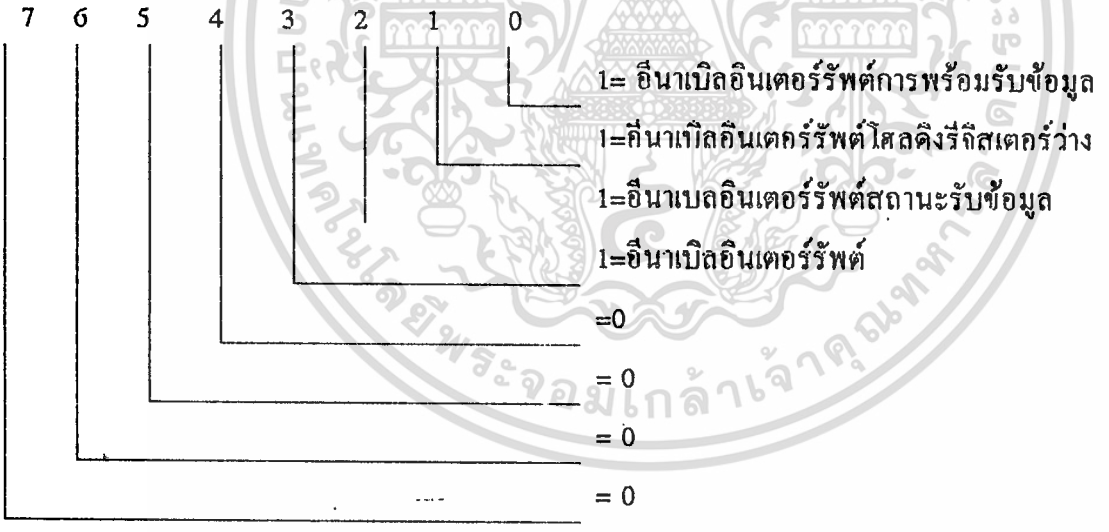
ในขณะที่มีความต้องการอินเตอร์รัพต์หลายระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่ารอไว้ก่อน โดยเก็บสถานะการอินเตอร์รัพต์นี้ไว้ในรีจิสเตอร์กำหนดอินเตอร์รัพต์ ความหมายของรีจิสเตอร์นี้มีดังนี้

บิต0 เป็นบิตที่ใช้แสดงว่ามีอินเตอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถให้ซีพียูตรวจสอบดูด้วยวิธีการ polling ได้ ถ้าบิตนี้เป็น "1" หมายถึง ไม่มีอินเตอร์รัพต์เกิดขึ้น

บิต 1-2 เป็นบิตที่แสดงความหมายบอกว่า การอินเตอร์รัพต์ที่เกิดขึ้นนั้น มาจากการอินเตอร์รัพต์ตามฟังก์ชันใด

บิต 3-7 มีค่าเป็น "0"

รีจิสเตอร์อีนาเบิลอินเตอร์รัพต์ (INTRPT -interrupt enable registe) ใน COM1 เมื่อให้ DLAB =0 พอร์ต 3F9 จะเป็นรีจีสเตอร์อีนาเบิลอินเตอร์รัพต์ผู้ใช้สามารถกำหนดให้เกิดอินเตอร์รัพต์หรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจิสเตอร์นี้ จากที่กล่าวแล้วว่าการอินเตอร์รัพต์ของ 8250 นี้มี 4 แบบ ดังนั้นจึงต้องกำหนดการอีนาเบิลได้ทั้ง 4 แบบ โดยการใช้ข้อมูลแต่ละบิตของรีจิสเตอร์นี้เพื่อการกำหนดอีนาเบิล ข้อมูลที่อยู่ในรีจิสเตอร์นี้มีความหมายดังนี้



รูปที่ 3.10 ค่าของบิตในรีจิสเตอร์อีนาเบิลอินเตอร์รัพต์

ตารางที่ 3.6 ฟังก์ชันการอินเตอร์รัพต์รหัสอินเตอร์รัพต์

บิต 2	บิต1	บิต0	ระดับความสำคัญ	ชนิดของอินเตอร์รัพต์	แหล่งเกิดอินเตอร์รัพต์	การรีเซตควบคุมอินเตอร์รัพต์
0	0	1	สูงสุด	ไม่เกิด สถานะการรับข้อมูลจากสายส่งสื่อสาร	ไม่เกิด overrun error parity error framming error break interrupt	อ่านข้อมูลจากRegister สถานะ สายสื่อสาร
1	0	0	ที่สอง	การพร้อมรับข้อมูล	มีข้อมูลที่ตัวรับ	การอ่านข้อมูลจากบัฟเฟอร์
0	1	0	ที่สาม	โฮลดิงรีจิสเตอร์สำหรับส่งข่าว	โฮลดิงรีจิสเตอร์สำหรับส่งข่าว	อ่านรีจิสเตอร์กำหนดอินเตอร์รัพต์ IIR หรือเขียนลงไปยังโฮลดิง Register สำหรับส่ง
0	0	0	ที่สี่	สถานะโมเดม	CIS DSR RI ตรวจสอบสายส่งโดยตรง	อ่านรีจิสเตอร์แสดงสถานะของโมเดม

บิต 0 บิตนี้ ได้รับการเซตเป็น “1” เมื่อต้องการอีนาเบิลอินเตอร์รัพต์การพร้อมรับข้อมูล

บิต “1” บิตนี้ได้รับการเซตเป็น “1” เมื่อต้องการอีนาเบิลอินเตอร์รัพต์โฮลดิงรีจิสเตอร์ว่าง

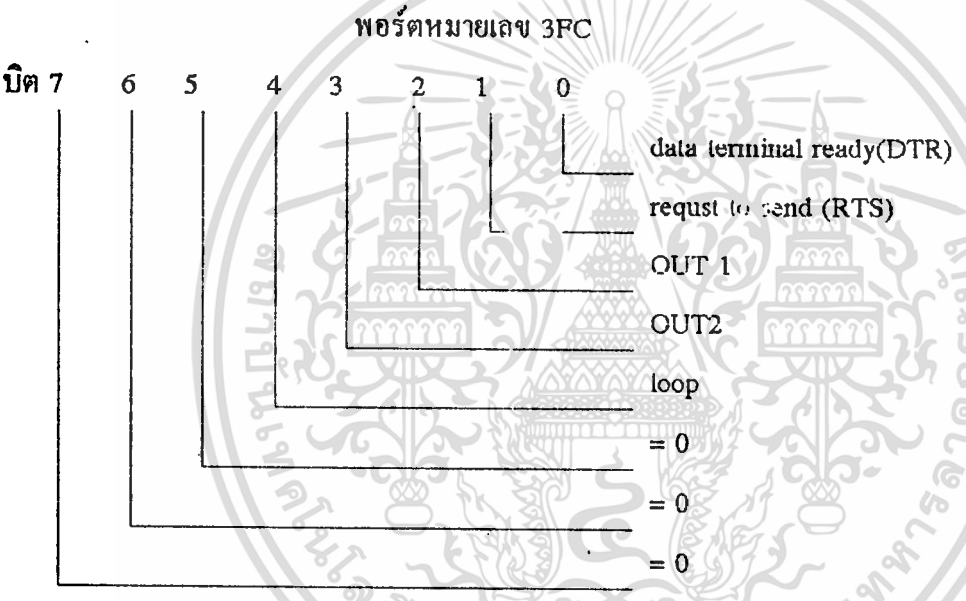
บิต 2 บิตนี้ได้รับการเซตเป็น “1” เมื่อต้องการอีนาเบิลอินเตอร์รัพต์จากสถานะการรับข้อมูลจากสายสื่อสาร

บิต 3 บิตนี้ได้รับการเซตเป็น “1” เมื่อต้องการอีนาเบิลอินเตอร์รัพต์จากสถานะโมเดม

บิต 4-7 ได้รับการกำหนดให้เป็น 0 เสมอ

รีจิสเตอร์ควบคุมโมเดม (modem control register) รีจิสเตอร์ตัวนี้มีไว้ให้ซีพียูส่งผ่านข้อมูลมาเก็บเพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเดม การกำหนดพอร์ตของรีจิสเตอร์ตัวนี้คือ 3FC ข้อมูลต่าง ๆที่มีในรีจิสเตอร์ตัวนี้มีความหมายดังนี้

บิต 0 บิตนี้มีความหมายถึงการควบคุมสัญญาณ DTR เมื่อบิตนี้มีค่าเป็น "1" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "0" และถ้าบิตนี้มีค่าเป็น "0" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "1"



รูปที่ 3.11 ค่าของบิตในรีจิสเตอร์ควบคุมโมเดม

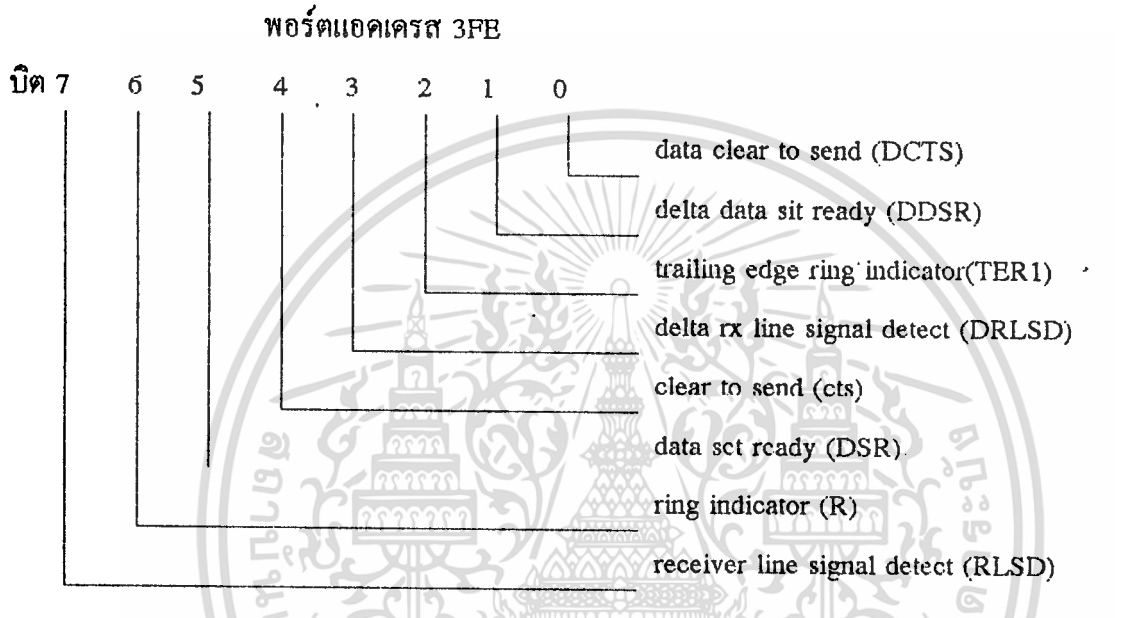
บิต 1 บิตนี้มีความหมายถึงสัญญาณ RTS ซึ่งจะมีผลเหมือนกับบิต 0 ในกรณีของ DTR

บิต 2 บิตนี้ใช้ควบคุมขาเอาต์พุต 1 (OUT 1) ซึ่งจะมีผลเหมือนบิต 0

บิต 3 บิตนี้ใช้ควบคุมเอาต์พุต 2 (OUT2) ซึ่งมีผลเหมือนบิต 0

บิต 4 บิตนี้จะใช้สำหรับการกำหนดวงรอบสำหรับการตรวจสอบ 8250 เมื่อบิต 4 บิตนี้ได้รับการเซตเป็น "1" สิ่งที่จะเกิดขึ้นเป็นดังนี้ ข้อมูลที่ SOUT จะได้รับการเซตให้เป็นลอจิก "1" ขาข้อมูลอินพุต SIN จะได้รับการแยกตัวออก ข้อมูลของเอาต์พุตชิปรีจิสเตอร์จะได้รับการป้อน กลับมายังรีจิสเตอร์ข้อมูลอินพุต ส่วนสัญญาณ CTS, DSR, RLSD และ RI จะได้รับการแยกออกจากระบบแต่สัญญาณควบคุมโมเดมคือ DTR, DSR, OUT 1 และ OUT 2 จะต่อเข้ากับสัญญาณทั้งสี่ที่เป็นอินพุต ดังนั้นจึงตรวจสอบระบบการทำงานได้

รีจิสเตอร์แสดงสถานะโมเดม รีจิสเตอร์ตัวนี้จะเป็นตัวที่รับสถานะจาก โมเดมมาเก็บไว้เพื่อให้ซีพียูสามารถอ่านครวจสอบดูได้ สถานะของข้อมูลจะแยกแอกทีฟเมื่อมีข้อมูลเป็น "1"และจะได้รับการรีเซตเมื่อซีพียูอ่านข้อมูลในรีจิสเตอร์นี้ไปพอร์ตที่ใช้กำหนดเป็นพอร์ตหมายเลข 3FEข้อมูลภายในรีจิสเตอร์นี้เป็นดังนี้



รูปที่ 3.12 ค่าของบิตในรีจิสเตอร์แสดงสถานะโมเดม

บิต 0 บิตนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ CTS กล่าวคือเมื่อขา CTSของ 8250 ได้เปลี่ยนสถานะหลังจากที่ซีพียูได้อ่านสถานะนี้ไปแล้ว บิตนี้ก็จะบอกด้วยเซตและเมื่อซีพียูอ่านก็จะได้รับการรีเซต "0" และจะได้รับการเซต "1" เมื่อมีการเปลี่ยนสถานะที่ขา CTS

บิต 1 เหมือน บิต0 แต่เป็นบิตที่แสดงสถานะการเปลี่ยนแปลงของ DSR

บิต 2 บิตนี้เป็นบิตแสดงว่าสัญญาณ RI ซึ่งเป็นอินพุตของ 8250 ได้รับการเปลี่ยนจากออฟ "1" มาเป็นออฟ "0"

บิต 3 บิตนี้เหมือนบิต 0 แต่เป็นบิตแสดงสถานะการเปลี่ยนแปลงของ line signak detector ซึ่งเป็นขาอินพุต RLSD

บิต 4 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา CTS

บิต 5 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา DSR

บิต 6 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RI

บิต 7 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RLSD

อนึ่งถ้าบิต 4 ของ MCR ได้รับการเซตหรือให้ทำลูป (loop) ตรวจสอบข้อมูลในบิต 4 จะเหมือนกับ RTS ใน MCR ข้อมูลในบิต 5 จะเหมือนกับ DTR ใน MCR ข้อมูลในบิต 6 จะเหมือนกับ OUT1 ใน MCR ข้อมูลในบิต 7 จะเหมือนกับ OUT2 ใน MCR

รีจิสเตอร์บัฟเฟอร์สำหรับตัวรับข้อมูล (receiver buffer register) เป็นรีจิสเตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนด คือ หมายเลขแอดเดรส 3F8 ขณะที่ DLAB = 0 หากซีพียูอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

รีจิสเตอร์โฮลดิงสำหรับตัวส่งข้อมูล (transmitter holding register) เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจิสเตอร์นี้จะรับข้อมูลจากซีพียู โดยที่กำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ DLAB = 0 ข้อมูลของซีพียูที่เอาต์พุตมาที่พอร์ตนี้ก็เพื่อจะส่งออกไปยังสายสารข้อมูล

ตัวอย่างการเขียนโปรแกรมรับส่งข้อมูลแบบ polling

เพื่อให้เข้าใจวิธีการใช้ วายการ์ดอะแดปเตอร์สื่อสารนี้ดียิ่งขึ้น จึงขอยกตัวอย่างการเขียนโปรแกรมเพื่อควบคุมพอร์ตสื่อสาร COM1 แบบง่าย ๆ โดยไม่มีการตรวจสอบสัญญาณโมเดม โปรแกรมตัวอย่างนี้เริ่มจากการกำหนดค่าเริ่มต้นให้กับ 8250 ใหม่โดยการกำหนดอัตราบอดให้ใหม่ ในที่นี้ใช้ตัวหารเป็น 384 คือ กำหนดเป็น 300 บอด สังเกตว่าโปรแกรมเซตพอร์ต 3FB ซึ่งเป็น register ควบคุมให้บิต 7 เป็น 1 ก่อนที่จะส่งตัวหารไป ส่วนพอร์ต 3FB ส่วนสุดท้ายที่ส่งรหัส 00000011B ไปก็คือกำหนดพอร์ตแมตของข้อมูลที่ส่งเป็นแบบ 8 บิต ไม่มีพาริตี้ ส่วนที่เหลืออีก 2 ส่วนของตัวอย่างเป็นการส่งตัวหนังสือ 1 ตัว และรับตัวหนังสือ 1 ตัวรีจิสเตอร์แสดงสถานะของสายสื่อสารคือ พอร์ต 3FDH มีข้อมูลแสดงสถานะของบัฟเฟอร์ตัวส่งและตัวรับเราจะส่งข้อมูลออกไปเมื่อบัฟเฟอร์ว่างเท่านั้น และทำนองเดียวกันก็คือ เราจะอ่านข้อมูลเมื่อข้อมูลมาพร้อมในบัฟเฟอร์แล้วเท่านั้น การส่งจะตรวจสอบบิต 5 การรับจะตรวจสอบบิต 0

3.8 ตัวอย่างการใช้อินเตอร์รัพต์ของ 8250

เนื่องจากการ์ดอะแดปเตอร์สื่อสารนี้มีอินเตอร์รัพต์ให้ผู้ใช้งานได้ หากดูวงจรทางฮาร์ดแวร์แล้วจะพบว่าเราได้ต่อ OUT2 ซึ่งเป็นขาเอาต์พุตของการ์ดเข้ากับสัญญาณอินเตอร์รัพต์ ที่จะมาจากการ์ดนี้ การโปรแกรมจะต้องโปรแกรมรีจิสเตอร์อินาเบิลอินเตอร์รัพต์ของ 8250 ให้ส่งอินเตอร์รัพต์ ซึ่งในที่นี้จะใช้ receive line status interrupt ซึ่งส่งออกไปเป็น IRQ3 ของ 8250 สังเกตว่าโปรแกรมนี้

เรามีวิธีการควบคุมกับ 8259 อย่างไร ในที่นี้เรารู้ว่าเมื่อ IRQ3 เกิดขึ้น 8250 จะเรียกอินเตอร์รัพต์ OB ซึ่งอยู่ที่ตำแหน่ง 58H ดังนั้นเราจะต้องมีการเตรียมตัวชี้มายังโปรแกรมบริการที่เราเตรียมไว้

ตอนแรกของโปรแกรมในกระบวนความ SET-INTERRUPT นี้เริ่มจากการกำหนดค่าที่พอยเตอร์ของอินเตอร์รัพต์เวกเตอร์ให้ชี้มายังโปรแกรมบริการคือ INT-HANDLER หลังจากนั้นก็อีนาเบิลรีจิสเตอร์อินเตอร์รัพต์โดยเซตบิต 2 หมายถึงการอีนาเบิลให้รับข้อมูล โปรแกรมส่ง OUT2 เมื่อส่งอินเตอร์รัพต์เข้าในระบบเอาต์พุตของ OUT2 จะเป็นตัวไปเปิดเกตเพื่อให้อินเตอร์รัพต์ส่งผ่านต่อไปได้

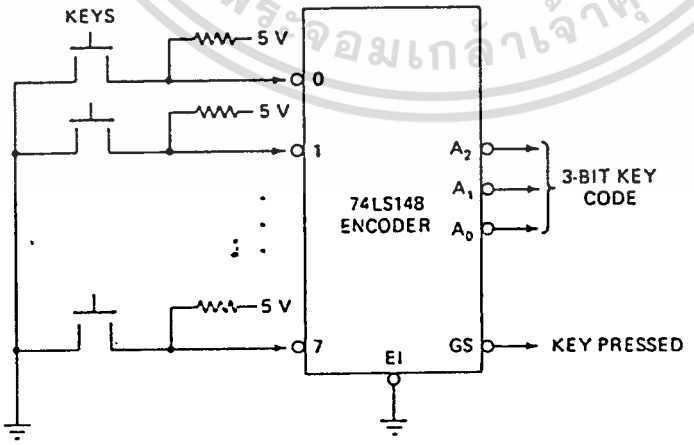
ประโยชน์ของพอร์ตสื่อสารบนไมโครคอมพิวเตอร์ 16 บิตมีได้มากมาย ปัจจุบันเราเริ่มใช้พอร์ตสื่อสารนี้มากขึ้น และเชื่อแน่ว่าในอนาคตบอร์ตสื่อสารจะมีบทบาทสำคัญมากยิ่งขึ้นการเรียนรู้และเข้าใจพอร์ตสื่อสารให้ถ่องแท้ จึงเป็นเรื่องดีที่จะทำให้เราใช้งานได้อย่างมีประสิทธิภาพ

บทที่ 4
คีย์บอร์ด

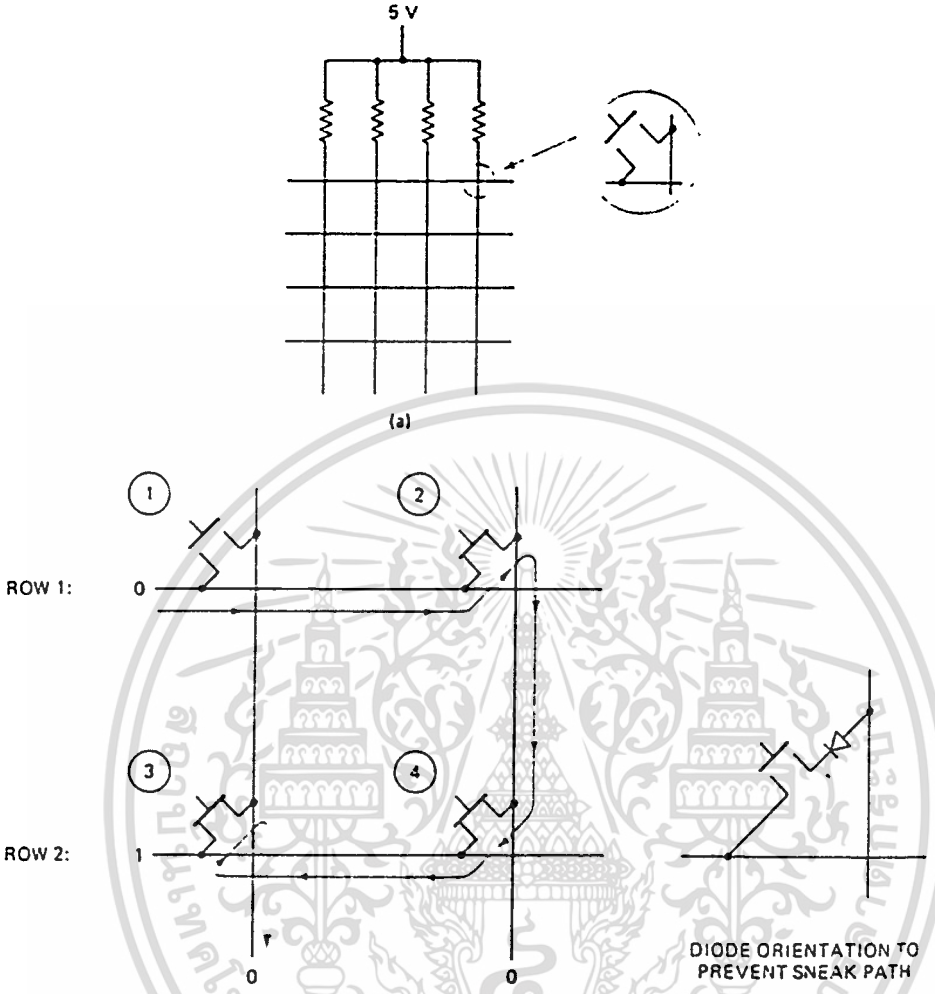
คีย์บอร์ด

เมื่อพิจารณาในคู่มือต่างๆโดยจะเห็นสัญลักษณ์ตัวใหญ่จะเป็นสัญลักษณ์พิเศษที่แสดงจะเป็นจุดสำคัญที่จะบอกให้เรารู้ว่าภายในคีย์บอร์ดนี้มีจำนวนคีย์กดเท่าไหร่ซึ่งในนี้จะใช้สูงสุดถึง 100 ปุ่มซึ่งเป็นสวิทช์กดนั่นเองคือมันจะต่อร่วมกันที่จุดๆหนึ่งในการใช้งานนั่นเอง กลุ่มของ KEYPAD เป็นการประยุกต์การใช้ KEYBOARDภายในซึ่งมีคีย์เล็กๆอยู่จำนวนมากใน KEYPAD และKEYBOARDอื่นๆจะสัมพันธ์กันตามรายละเอียดของสัญลักษณ์หรือค่า BINARYเมื่อมีการกดคีย์ มันจะทำการสร้างรหัส binary ที่แทนคีย์นั้นๆเมื่อตัวเลขของคีย์หรือจำนวนคีย์น้อยกว่าหรือเท่ากับ 16 ถ้ามีการกดคีย์จะทำให้มีการเข้าระหัสออกเป็นข้อมูลขนาน ซึ่งอาจจะนำผลที่ได้นี้ไปทำการ COMBINATION ในวงจรต่างๆ สำหรับในตัวอย่างนี้เป็นไอซีเข้าระหัสโดยมีอินพุท และจะทำให้เอาท์พุทนั้นACTIVE ที่อินพุทบางตัวจะเป็น LOW ถ้าหากใช้วงจรหรือซอฟแวร์ ในการ LATCHED และใช้ผลิตการขัดจังหวะโดยมันจะยอมให้ไมโครโปรเซสเซอร์ ทำการอ่านเอาท์พุทของการเข้าระหัสนั้นเอง

ไอซีเบอร์ 74LS148 เป็นไอซีเข้าระหัสแสดงดังรูป 4.1 มี 8 หน้าสัมผัสแล้วออกเอาท์พุทมี 3 บิท และจะต่อที่ขาหนึ่งร่วมกันของแต่ละหน้าสัมผัส



รูปที่4.1ใช้ไอซี 74LS148 PRIORITY ENCODER เป็นตัวเข้าระหัสใช้ EIGHT-KEY KEYPAD

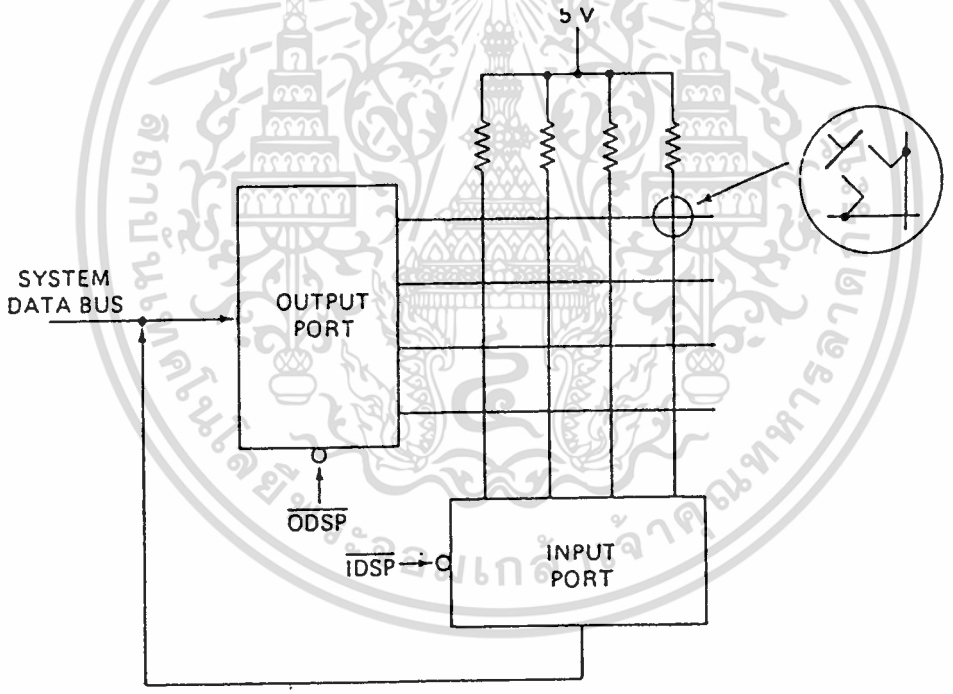


รูปที่ 4.2 (a) คีย์บอร์ดสวิทช์เมตริก

(b) การเกิด SNEAK PATH ในสวิทช์เมตริกซ์ ; เกิดการกดคีย์ 2 , 3 และ 4 จะเกิดดังรูป

จะทำการจัดรูปแบบของคีย์ในรูป 4.1 ให้การทำงานของคีย์ต่างๆ ทำงานสัมพันธ์กันสามารถทำได้จากการกำหนดจุดต่อของสายตัวนำของคีย์นั้นๆ จะทำให้เกิดรูปแบบของเมตริกซ์ดังแสดงในรูป 4.2a การให้คีย์ทำงานตามแบบที่กล่าวมานั้น จะมีประโยชน์อย่างยิ่ง คือ เมื่อพิจารณาที่คีย์ใหญ่ (แบบแรก) จะทำให้เข้าใจยากเพราะว่าการทำแบบนั้นจะจำกัดขุปกรณ์ ของ HARD WARE ในการเข้ารหัสการที่จะรู้ว่า KEY ที่ใช้นั้นมีการกดหรือไม่จะพิจารณาจากการที่ถ้ามีการกดคีย์มันจะรับรู้การกดเป็นทอดๆ ไปเพราะเป็นการต่อแบบเมตริกซ์ ซึ่งจะทำให้มีการสแกนคีย์เกิดขึ้นนั้นเอง ดังนั้นรหัสดิจิตอลก็คือการเข้ารหัสจากคีย์บอร์ดนั้นเอง

การสแกนคีย์เมตริกซ์จะใช้ทั้ง SOFT WARE และ HARD WARE โดยให้แถวของเมตริกซ์เป็น LOGIC 0 และทำการตรวจจับค่า LOGIC ของหลักถ้าหากมีจุดๆ หนึ่งในหลักเป็น LOGIC 0 จะเกิดมีการกดคีย์นั้นเกิดขึ้นการเข้ารหัสตัวอื่นๆก็จะหมุนวนไปตามแนวนอนของสายโดยมันจะทำงานที่ LOGIC 0 และในสายแนวนอนอื่นๆเป็น LOGIC 1



รูปที่ 4.3 โครงสร้าง HARD WARE สำหรับ การ SCAN SOFT WARE ของคีย์เมตริกซ์

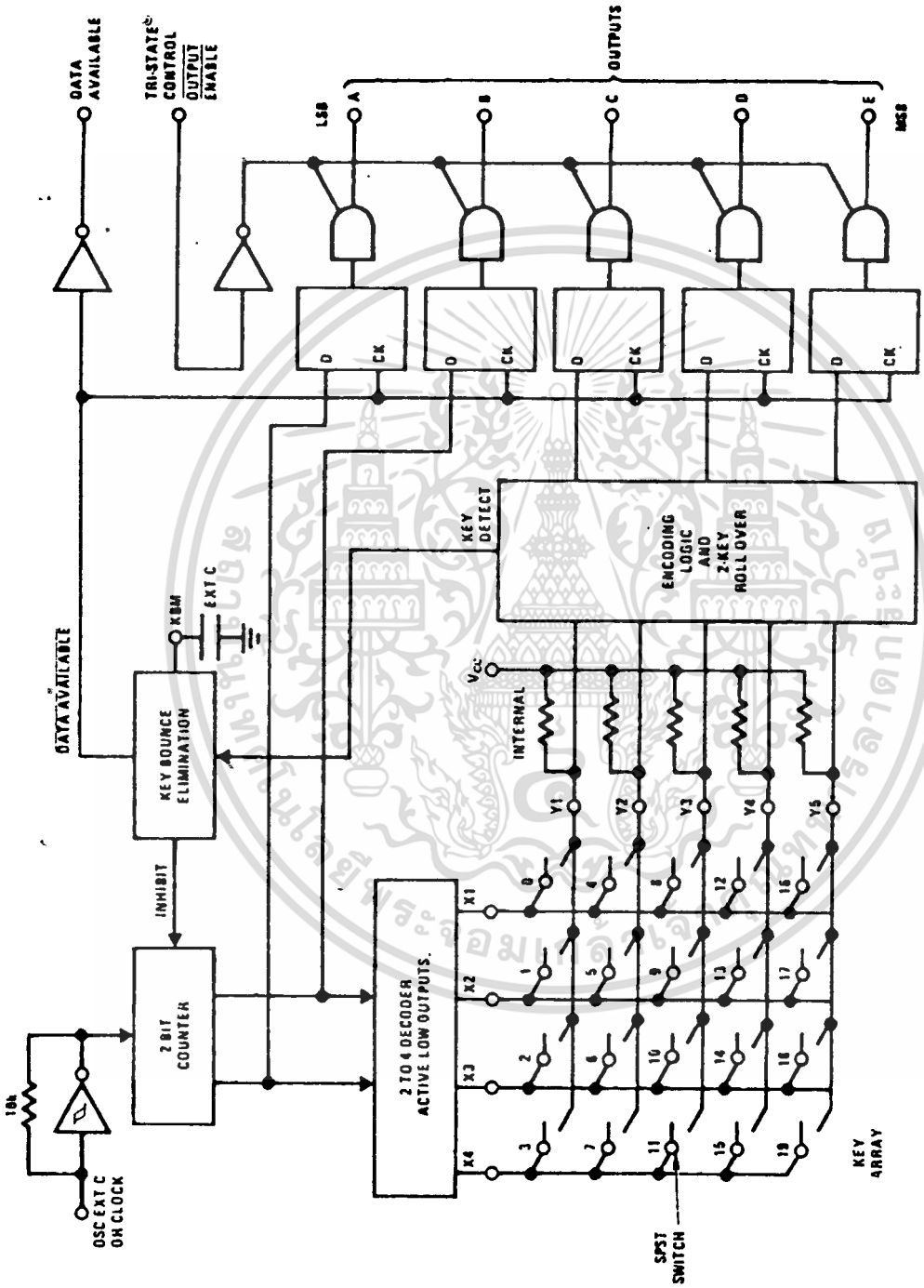
ในขณะที่สัญญาณในสายแนวนอนเป็น LOGIC 0 สายสัญญาณแนวตั้งใช้เป็นการตรวจจับ ซึ่งในขณะนั้นมันจะเป็น LOGIC 0 โดยเส้นนี้จะเป็นเส้นที่เราพิจารณาไปตัดกันที่คีย์นั้นๆ หรือที่จุดนั้นจะแสดงว่ามีการกดคีย์เกิดขึ้น

ตัวอย่างของ HARD WARE ที่ใช้ในการตรวจจับการกดคีย์ และการสแกนคีย์ด้วย SOFT WARE จะแสดงดังรูป 4. 3 แถวของคีย์เมตริกจะถูกควบคุมโดยเอาท์พุตที่พอร์ตนั้นๆ โดยที่หลักของเมทริกจะถูกตรวจโดยใช้อินพุตพอร์ตในการควบคุมเช่นกัน เมื่อเริ่มต้นการกดคีย์และปล่อยให้พ้นจากการตรวจจับ SOFT WARE จะ SCAN โดยใช้ SUBROTINE โดยมันยจะเรียกใช้ซ้ำๆ และจะทำการเพิ่มจำนวนรอบ ซึ่งเรียกว่าการ SCAN อัลกอลิทึม ซึ่งจะต้องเตรียมและแยกการกดคีย์ในแต่ละครั้ง โดยให้ช่วงระหว่างเวลาการกดคีย์โดยให้มีค่าคาบเวลาที่ยาว

ปัญหาที่เกิดขึ้น เมื่อมีการกดคีย์มากกว่า 2 คีย์ พร้อมๆ กัน เมื่อเกิดเหตุการแบบนี้เรียกว่าการเกิด ROLLOVER ก่นการ SCAN ในอัลกอลิทึม ถ้าเกิด ROLLOVER ผลที่เกิดจากการกดคีย์ที่ป้อนไปจะขึ้นอยู่กับจำนวนของการตรวจจับ (DETECTION) โดยใช้ SUBROTINE และมันเป็นตัวบอกว่าไม่มีการกดคีย์นั้นเอง

โดยทั่วไปการป้องกันปัญหาใน ROLLOVER จะเกิดปัญหาสองแบบเกิดจาก TWO-WAY และ N-KEY และ TWO-WAY นั้นคือในKEY BOARD จะมีการสแกน โดยถ้ามีการกดคีย์ก็คือต้องมีการรับข้อมูลตลอด ถ้ามีการกดคีย์ในเวลาเดียวกันเรียก TWO-KEY ROLLOVER เมื่อมีการกดคีย์โดยคีย์อื่นๆ ปล่อยทำให้คีย์ที่กดมีการรับข้อมูลและจะไม่รับรู้ เมื่อมีการปล่อยคีย์ แต่ถ้าหากมีการกดคีย์ในครั้งที่สอง ก่อนที่การกดคีย์ครั้งแรกจะปล่อยมันจะรับรู้ข้อมูลเพียงข้อเดียว ภายหลังจากที่มีการปล่อยคีย์ที่กดในครั้งแรกแล้ว

N-KEY ROLLVER เป็นขั้นตอนของการกด KEY ที่เป็นลำดับไปโดยจะตรวจการกดโดยการสแกน และมันจะไม่คำนึงถึงสภาวะของคีย์อื่นๆ วิธีนี้จะทำให้เกิดข้อเสียขึ้นคือเมื่อผ่านข้อมูลเข้าไปผู้ใช้ข้อมูลจะผ่านไปอย่างรวดเร็ว ทำให้ผู้ที่กดครั้งที่สอง และอาจจะมีผู้ที่กดครั้งที่สามด้วย โดยคนแรกยังไม่ได้ปล่อยคีย์เลย



รูปที่4.4 วงจรภายในของ IC 74C923

ส่วนที่เพิ่มเข้าไป เมื่อเกิด N-KEY ROLLOVER เพื่อต้องการเพิ่มส่วนประกอบในคีย์เมตริกซ์ ดังแสดงในรูป 4.2a จุดตัดที่จุดต่างๆของสายในเมตริกซ์จะทำให้ดูง่ายโดยต่อสวิทว์ระหว่างแถวและหลักของคีย์นั้น ดังนั้นมันจะเชื่อมต่อกันทางไฟฟ้าที่จุดทั้งสองเมื่อมีการกดสวิทช์ พิจารณาตำแหน่งในรูป 4.2b จะเห็นว่าในแถวหนึ่งมีการเลือกสแกนและสวิทช์ 2, 3 และ 4 จะถูกกดอยู่ส่วนนั้นจะเป็นทางเดินของกระแสไฟฟ้า เรียกว่า SNEAK PATH โดยมันจะมี สวิทช์ 2 , 4 , 3 จะทำให้หลักที่หนึ่งมีเอาท์พุตเป็นศูนย์เมื่อแถวหนึ่งถูกเลือกและหลักหนึ่งเป็นศูนย์จะทำให้เกิดสวิทช์หนึ่งเห็นเป็น การกดแต่ไดโอดจะถูกต่ออนุกรมกับสวิทช์เพื่อป้องกัน NEAK PATHS ในเมตริกซ์

สำหรับ HARD WARE ของการสแกนของคีย์เมตริกซ์ ใช้ LSI คีย์เข้ารหัสจะใช้งานได้ง่าย จะต้องมีส่วนของสแกนที่เล็กและพื้นที่ของคีย์สวิทช์จะใหญ่ และมันจะเป็นตัวสร้างเอาท์พุตของการเข้ารหัส ให้ตรงกันกับเมื่อมีการกดคีย์ ตัวอย่างของอุปกรณ์ที่ใช้งาน เช่น มีปุ่มคีย์สวิทช์ 16 , 20 , 64 , 78 และ 90 คีย์เมตริกซ์ เช่นใช้ไอซีคีย์บอร์ดเข้ารหัสเบอร์ MM74C923 แสดงได้ดังตาราง 4.1 อุปกรณ์ตัวนี้จะมีปุ่ม 20 คีย์เป็นแบบ SPST สวิทช์ โดยภายในมีวงจร COUNTER 2 BIT และ 2 TO 4 โดยเป็นการถอดรหัส จากนั้นก็สแกนหลักของสวิทช์เมตริกซ์โดยความถี่ในการสแกนจะกำหนดจากส่วนของการ OSCILLATOR ภายในเมื่อมีการเข้ารหัสของไลจิกจะทำให้เกิดที่แถวนั้นๆ โดยขณะนั้นจะมีการตรวจจับการกดคีย์ด้วยจะมีการเพิ่มวงจรเพื่อให้ CONTACT ตัดกับการ BOUNCE (ช่วงของสัญญาณ IIIGIIT) และ TWO-KEY ROLLOVER คาบเวลาของการ DEBOUNCE (ช่วงของสัญญาณ LOW) สามารถกำหนดได้จากค่า C ภายนอกเมื่อคีย์มีการกดเป็นการรับข้อมูล 3 BIT ที่ได้จากการเข้ารหัสไลจิกและ 2 BIT จากวงจร COUNTER

ทำให้เกิดการค้างสภาวะ z (DATENER) ออกมาเป็นเอาท์พุตให้ REGISTER 5 BIT โดยข้อมูลของสัญญาณ AVAILABLE นั้นจะทำให้เอาท์พุตเป็น HIGHT จะมีช่วงยาวเท่ากับ เมื่อคีย์มีการรับข้อมูลที่มีการกดค้างเมื่อปล่อยคีย์ ข้อมูลสัญญาณ AVAILABLE จะเป็น LOW ถ้าหากคีย์อื่นๆ ไม่ได้กด ข้อมูลจะเป็น HIGHT อีกครั้งหลังจาก เกิดการ DEBOUNCE PERIOD แสดงว่ามีข้อมูลสำหรับการกด 2 ครั้ง

เหตุที่ใช้ HARD WARE ของ ELIMINATES เพื่อต้องการให้ไมโครโปรเซสเซอร์ SCAN คีย์บอร์ดด้วยจำนวนคงที่ ภายในช่วงเวลาที่กำหนดข้อมูลของสัญญาณจะถูกขัดจังหวะ จากไมโครโปรเซสเซอร์มันจะยอมให้เกิดการ SCAN เมื่อมันมีตำแหน่ง ที่ตรงกันด้วยการใช้ ไมโครโปรเซสเซอร์ในลักษณะอื่นๆ

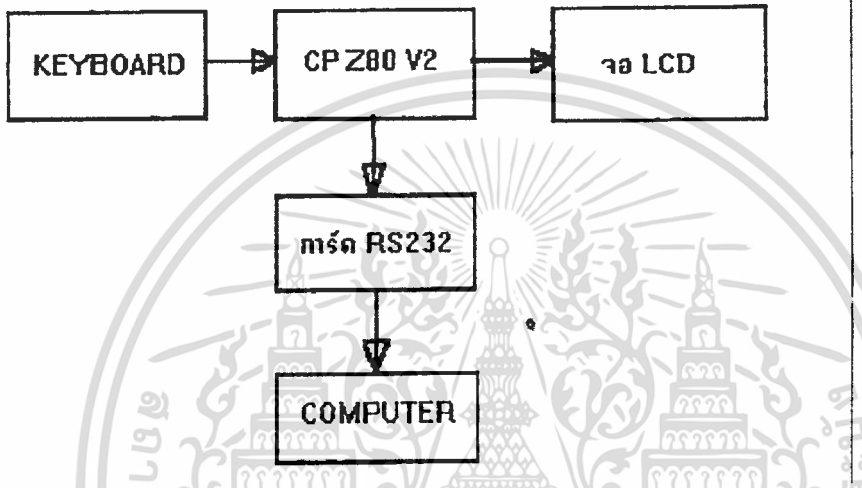
KEYBOARD SCANNER ส่วนใหญ่จะใช้เบอร์ MM 5740 เป็นคีย์บอร์ด ENCODER สามารถ SCAN ได้ถึง 90 คีย์ โดยจะถูกควบคุมด้วยสัญญาณ CLOCK จากภายนอกและภายใน มี RING COUTERS เป็นตัวเลือกแถวและหลัก ซึ่งเป็นพื้นที่สำหรับการ SCAN นั้นเอง

ในการพิจารณา COUNTERS ด้วย ADDRESS ภายใน ROM นั้นคือต้องมีรหัสตรงกับตำแหน่ง ที่ทำให้เกิดการ SCAN ถ้าหากคีย์มีการกดรหัสจะเกิดเป็นลักษณะตำแหน่งของ REGISTER และก็จะใช้เป็นข้อมูล AVAILABLE อีกส่วนหนึ่ง นั้นคือจะใช้เป็นสัญญาณขัดจังหวะไมโครโปรเซสเซอร์ ในคีย์บอร์ดของ ENCOTER มีเอาท์พุท 3 สภาวะ จะถูกควบคุมโดยเอาท์พุท ENABLE ที่เป็นสัญญาณอินพุทของ NOT GATE ดังในรูป นั้นคืออินพุท ENABEL เป็นBUFFER 3 สภาวะและจะเป็นไปตามตำแหน่งของ DATA BUS ที่ได้จากลักษณะของENCODER ในขณะที่คีย์บอร์ดเข้ารหัสในช่วง TEBOUNCEจะมีการกดคีย์ และจะต้องทำให้ค่าของตัวเลขอื่นๆ เป็นไปในลักษณะเพื่อนำไปใช้ประโยชน์อย่างอื่นโดยการออกแบบคีย์บอร์ดใหม่ในการเข้ารหัสที่มีอินพุทและเอาท์พุทมากๆจะใช้ไอซี TTL โดยใช้เบอร์ MM5740 นี้เป็นเบอร์ที่หาใช้ง่าย โดยภายในเป็น ROM ที่โปรแกรมด้วยรหัส ASCII และสามารถเลือกอุปกรณ์ที่จะโปรแกรมได้ด้วย ลักษณะของข้อมูลที่ใช้โปรแกรมลงไป

บทที่5

รายละเอียดของโครงงาน

5.1 BLOCK DIAGRAM ของโครงงาน



รูปที่5.1 BLOCK DIAGRAM ของโครงงาน

-ข้อมูลที่ได้จากคีย์บอร์ดจะถูกส่งผ่านไปยังบอร์ด Z80 V2 แล้วนำไปทำการประมวลผลโดยส่วนของโปรแกรม จากนั้นข้อมูลจะถูกส่งออกมา โดยมีขนาด 8 บิตต่อ 1 รหัส ซึ่งเป็นรหัส ASCII เพื่อนำข้อมูลนี้ส่งผ่านไปยังการ์ดRS 232 และข้อมูลที่ LCD PORT จะถูกส่งต่อไปยังจอLCD

-จากที่กล่าวมาแล้วจะสามารถทำงานได้ต้องมีทั้งส่วนของ HARD WARE และ ส่วน SOFT WARE ประกอบเข้าด้วยกันเช่น HARD WARE ประกอบด้วยส่วนของ วงจรคีย์บอร์ดและ Z80 V2 ในส่วนของSOFT WARE ประกอบด้วยโปรแกรมมอนิเตอร์จึงจะสามารถทำงานได้

5.2 ผลการทดลองของส่วนคีย์บอร์ด

จากวงจรคีย์บอร์ดถูกออกแบบให้ใช้ไอซีเข้ารหัสเบอร์ 74C923 มาใช้งาน ซึ่งจะต้องใช้สวิทช์ทั้งหมด 32 คีย์ ดังนั้นจึงต้องใช้ไอซีเข้ารหัสจำนวน 2 ตัว คือไอซี 1 และ ไอซี 2 เมื่อกดคีย์ใดๆบนคีย์บอร์ดจะทำให้เกิดสัญญาณเอ้าท์พุท ปรากฏที่ขาเอ้าท์พุทของไอซีเข้ารหัสดังนั้นเพื่อให้ไอซี 1 และ ไอซี 2 ทำงานร่วมกัน ได้จึงต้องใช้ไอซี 3 ซึ่งเป็นไอซีนอร์เกต มาเชื่อมต่อเพื่อรวมสัญญาณที่ได้จากการกดคีย์ในทุก ๆ คีย์เข้าด้วยกันออกมาได้เป็นเอ้าท์พุทขนาด 5 บิตเพื่อส่งข้อมูลเข้าพอร์ต Aของไอซี 8255 ตัวที่ 1 เป็นข้อมูลขนาด 8 บิตโดยจะเพิ่มบิตที่ 5-7 และกำหนดให้เป็น 0โปรแกรมมอนิเตอร์จะทำหน้าที่แปลงข้อมูลจากคีย์บอร์ดเป็นรหัสแอสกี้แล้ว ส่งผ่านไปแสดงผลที่จอ LCD และส่งต่อไปยังการ์ด RS 232

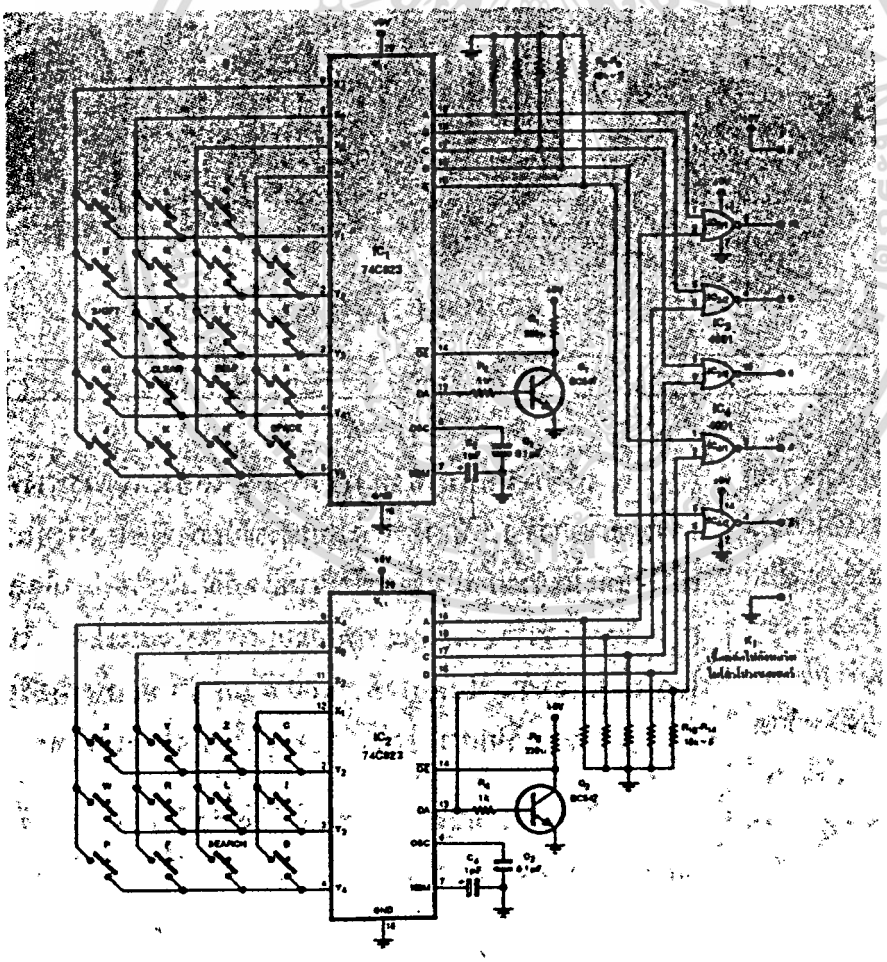
-จากการทดลองข้อมูลที่ได้จากคีย์บอร์ดจะเป็นข้อมูลรหัส BINARY ขนาด 5บิตของแต่ละคีย์

	11001 B	00011 B	11010 B	11000 B	11101 B	10000 B	11110 B
00111 B	01110 B	11100 B	00110 B	10110 B	01100 B	00001 B	00101 B
10011 B	11011 B	00000 B	01011 B	10100 B	10111 B	10101 B	01000 B
01111 B	01101 B	00100 B	00010 B	10010 B	01010 B	10001 B	01001 B

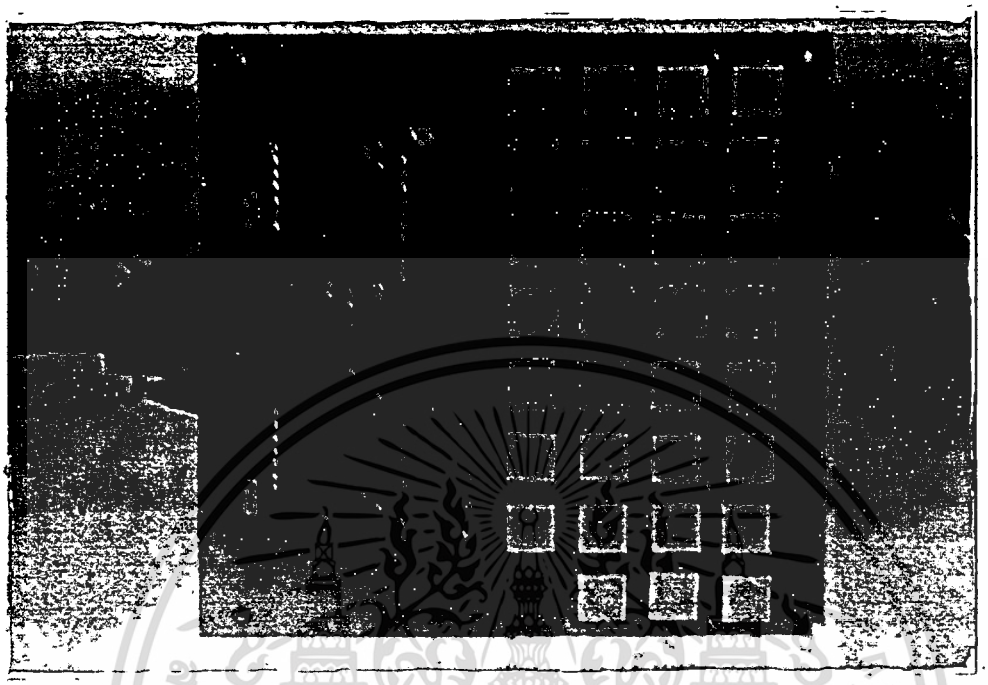
ตาราง5.1 ข้อมูลของแป้นคีย์บอร์ด

หลักการทำงานของส่วน KEYBOARD

วงจร Keyboard ถูกออกแบบให้ใช้ไอซีเข้ารหัสเบอร์ 74C923 มาใช้งานดังแสดงในรูปวงจรควบคุม Keyboard ซึ่งในโครงงานนี้ใช้สวิตช์ทั้งหมด 31 คีย์ ดังนั้นจึงต้องใช้ไอซีเข้ารหัสจำนวน 2 ตัวคือ IC1 และ IC2 เมื่อกดคีย์ใดๆ บนคีย์ จะทำให้เกิดสัญญาณเอาต์พุตปรากฏที่ขาเอาต์พุตของไอซีเข้ารหัส ดังนั้นเพื่อให้ทั้ง IC1 และ IC2 ทำงานร่วมกันได้จึงต้องใช้ IC3 ซึ่งเป็นนอร์เกตมาต่อเชื่อมเพื่อรวมสัญญาณที่ได้จากการกดคีย์ในทุกๆคีย์เข้าด้วยกันออกมาได้เป็นเอาต์พุต ที่มีขนาด 5 บิต เพื่อส่งข้อมูลไปยังไมโครโปรเซสเซอร์เพื่อทำการประมวลผลต่อไป



รูปที่ 5.2 แสดงวงจรสมบูรณ์ของส่วนคีย์บอร์ดรับข้อมูล



รูปที่ 5.3 ภาพถ่ายในส่วน KEYBOARD

รายการอุปกรณ์ส่วน KEYBOARD

ตัวต้านทาน			
1.	ตัวต้านทาน 330 Ω 1/4 W 5 %	2	ตัว
2.	ตัวต้านทาน 1 K Ω 1/4 W 5 %	2	ตัว
3.	ตัวต้านทาน 10 K Ω 1/4 W 5 %	10	ตัว
ตัวเก็บประจุ			
1.	โพลีเอสเตอร์ 0.1 μ F 50 V	2	ตัว
2.	อิเล็กทรอไลต์ 1 μ F 16 V	2	ตัว
ไอซี			
1.	BC547	2	ตัว
2.	74C923	2	ตัว
3.	4001	2	ตัว

5.3 ระบบสื่อสารข้อมูล RS - 232C

ได้จัดทำการ์ด RS 232C เพื่อใช้ในการส่งข้อมูลจากไมโครโปรเซสเซอร์ (Z80) ไปยังเครื่องไมโครคอมพิวเตอร์ โดยได้ใช้ IC เบอร์ 8250 เป็นตัวแปลงข้อมูลในลักษณะ PARALLEL เป็นแบบ SERIAL แล้วส่งผ่านไปตามสาย ไปยังเครื่องไมโครคอมพิวเตอร์ โดยอ้างถึง DATA LINK LAYER ซึ่งเป็น layer ที่ 2 ของมาตราฐาน OSI (open system interconnection) ซึ่งใช้ในการติดต่อสื่อสารของระบบโครงข่ายคอมพิวเตอร์

พอร์ตสื่อสาร RS 232 C บนเครื่องไมโครคอมพิวเตอร์เป็นมาตราฐาน แบบอะซิงโครนัส ที่สามารถโปรแกรมสตาร์ตบิต สต็อปบิตและพาริตี้บิต อัตราส่งสามารถกำหนดได้ตั้งแต่ 50 บอด ถึง 9600 บอด ลักษณะพิเศษของวงจรคือ สามารถส่งสัญญาณมาอินเตอร์รัพซีพียูตามเงื่อนไขได้ และ ยังมีโครงสร้างฮาร์ดแวร์ป้อนกลับเพื่อใช้ในการตรวจสอบระบบว่าทำงานปกติดีหรือไม่อีกด้วย วงจรพอร์ตสื่อสารของ ไมโครคอมพิวเตอร์ใช้ไอซีหมายเลข 8250 เป็นตัวสำคัญของระบบ มีขนาด 40 ขา มีขีดความสามารถคือ

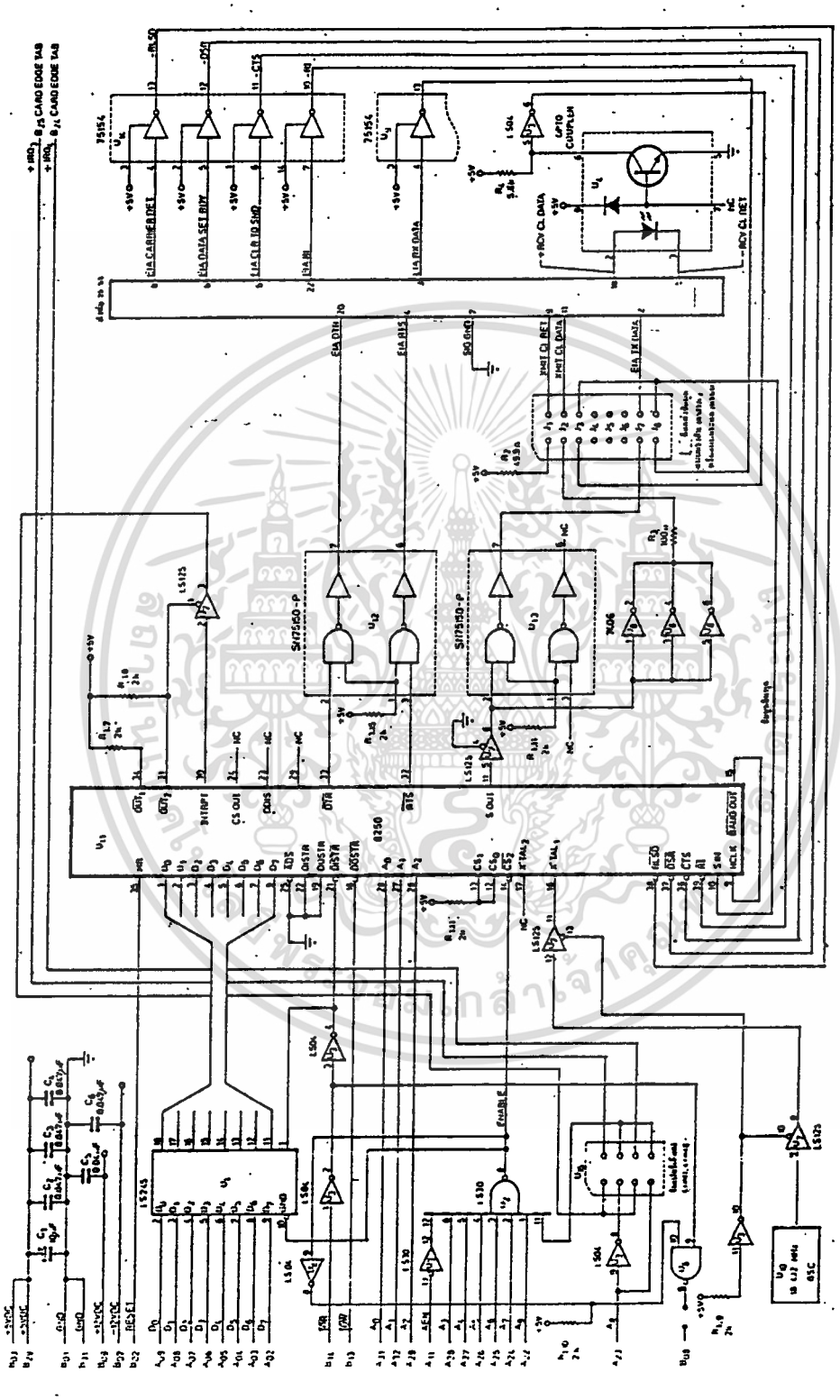
- มีบัฟเฟอร์ในตัววเพื่อทำให้ไม่จำเป็นต้องซิงโครนัสในการรรับส่ง
- ใช้สัญญาณนาฬิกาอิสระต่างหากไม่ขึ้นกับสัญญาณนาฬิกาของระบบ
- มีสัญญาณตอบโต้โมเดมทั้ง CTS (clear to send) , RTS (request to send) ,DSR (data terminal ready) , RC (ring indicator) และสัญญาณดีเทคตัวพาหะ (carrier detect)

CPU ติดต่อกับ 8250 ในลักษณะที่เป็นอินพุทเอาท์พุท การจัดพอร์ตจะกำหนดหมายเลขพอรร์ตอย่างเจาะจง โดยในไมโครคอมพิวเตอร์จะมีพอร์ต COM1 (3F8 - 3FE) และ COM2 (2F8 - 2FE)

5.3.1 การต่อวงจรของ IC 8250

บอร์ดอะแดปเตอร์สื่อสารนี้มีโครงสร้างการเชื่อมต่อตามพอร์ต 3F8-3FE และ 2F8-2FE ดังนั้นจะใช้บิต A8 เป็นตัวเลือกบนบอร์ดจะมีจัมเปอร์เพื่อจะบอกว่าเป็นพอร์ตสื่อสารแบบ COM1 หรือ COM2 การเลือกแอดเดรสใช้ 74LS30 วึ่งเป็น NAND เกตแบบ 8 อินพุตมาเป็นตัวเลือกโดยมี U15 ในรูปเป็นตัวเลือก A1 ดังได้กล่าวแแล้ว

ส่วนบัสของข้อมูล D0-D7 จะผ่านบัฟเฟอร์คือ 74LS245 ก่อนเข้าสู่ 8250 ส่วนสัญญาณควบคุมบน 8250 ทั้ง 10 มีดังนี้



รูปที่ 5.4 วงจรของบอร์ด RS - 232

ขามาสเตอร์รีเซต MR จะต่อโดยตรงกับสัญญาณรีเซตของระบบ

ขา ADS*,DISTR,DOSTR ต่อลงกราวนด์ ทั้งนี้เพราะแอดเดรสที่ต่อมาที่ชิป 8250 นี้เป็นส่วนสัญญาณแอดเดรสล้วนแล้วไม่ต้องสโตรบอีก

ขา DISTR* ต่อกับ IOR* ของระบบ โดยผ่านอินเวอร์เตอร์ 2 ตัว

ขา DOSTR* ต่อกับ IOW* ของระบบ

ขา CS1 CS2 ต่อขึ้นเป็นลอจิก "1"

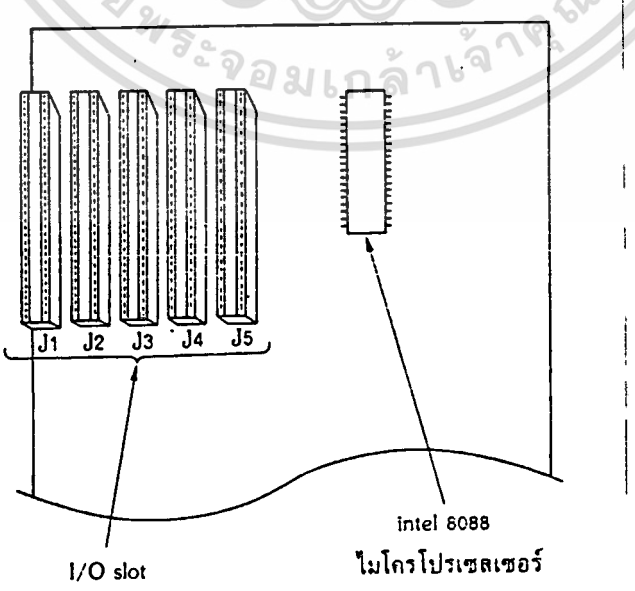
ส่วนขา CS2* มาจากการถอดระหัสให้เป็นพอร์ตของ COM1 หรือ COM2 ตามต้องการ ขา XTAL2 ไม่ใช้ ส่วน XTAL1 ต่อมาจากออสซิลเลเตอร์ 18.432 MHz ของระบบวงจรของบอร์ดอะแดปเตอร์สื่อสารเป็นดังรูปที่ 5.4

ส่วนของสัญญาณเอาต์พุตประกอบด้วยสัญญาณควบคุมโมเดม RLSD*,DSR*,CTS*,RI* ต่อออกไปยังหัวต่อตามมาตรรฐาน EIA RS232

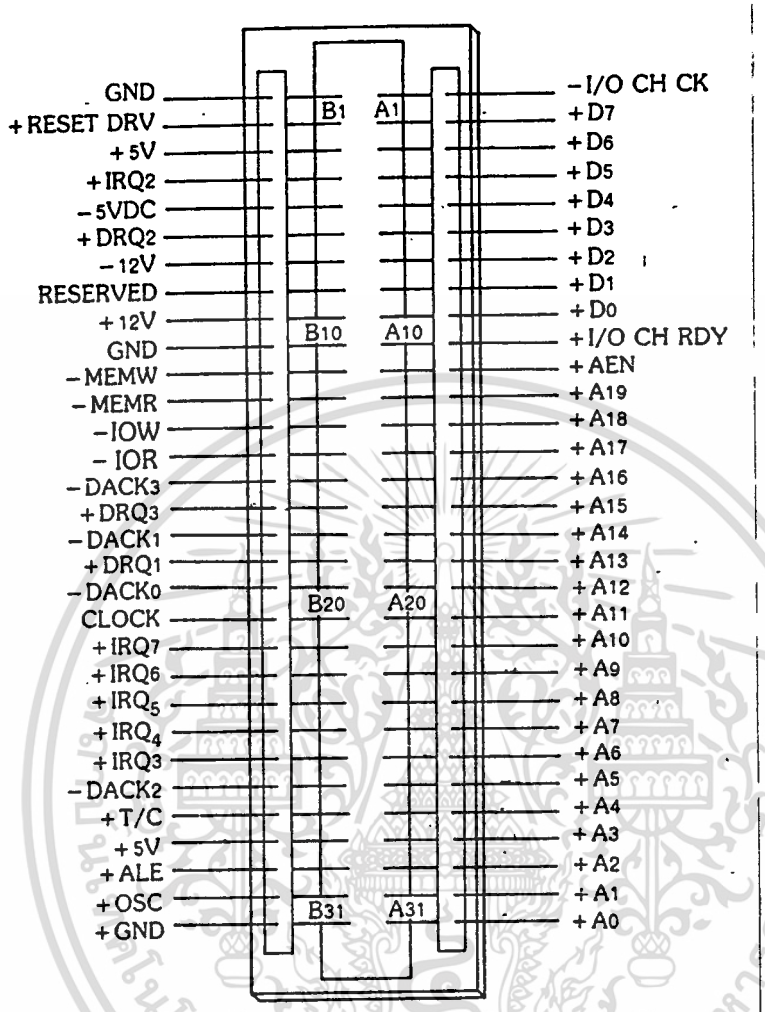
ขา SIN ,SOUT ต่อออกไปขาเอาต์พุตเช่นกันแต่มีการปรับให้เป็น สัญญาณแรงดันตามมาตรฐาน EIA หรือเลือกส่งเป็นกระแสวนรอบก็ได้ การเลือกใช้จัมเปอร์ I1-I8 เป็นตัวเลือกการแปลงกระแสเป็นแรงดันใช้ออปโตคัปเปลอรร์ดังรูปที่ 5.4

5.3.2 RS - 232 บนเครื่อง IBM PC

.ในการส่งข้อมูล (output data) และข้อมูลเข้า (input data) สำหรับ IBM PC นั้นจำเป็นต้องมีวงจรอิเล็กทรอนิกส์เชื่อมต่อเข้ากับ IBM PC ซึ่งมีระบบ I/O Slot แสดงลายละเอียดดังรูปที่ 5.5 และ รูปที่ 5.6 ตามลำดับ



รูปที่ 5.5 ระบบ I/O Slot บน IBM PC

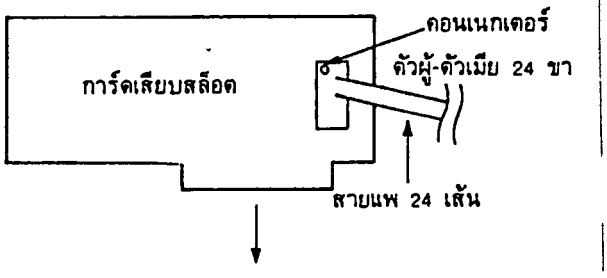


รูปที่ 5.6 ขาสัญญาณต่างๆ ใน I/O Slot บน IBM PC

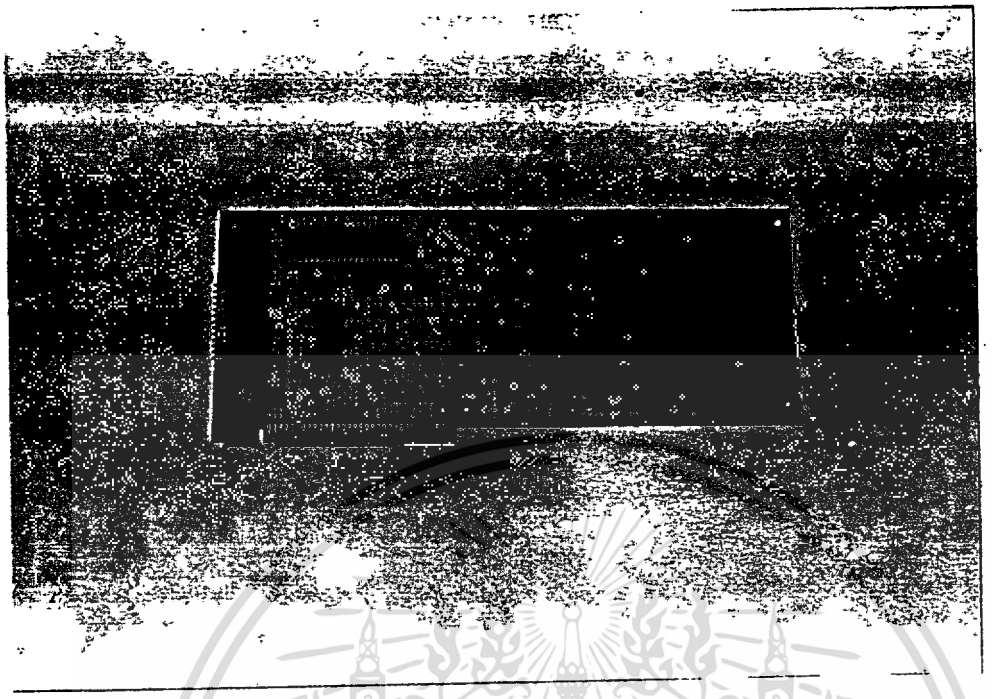
การส่งข้อมูลออกและนำข้อมูลเข้าจำเป็นต้องมีเส้นทาง ในที่นี้เราจะเรียกว่า พอร์ต (PORT) ซึ่งจะมีการจัดสรรที่ไม่ซ้ำซ้อนกัน มีแอดเดรส (ADDRESS) ที่แน่นนอน ตัวอย่างการจัดสรรแอดเดรสที่ใช้ติดต่อกับอุปกรณ์ ภายนอกบน IBM PC แสดงดังตารางที่ 5.2 และการสร้างการ์ด (CARD) เพื่อใช้เชื่อมต่อ I/O Slot แสดงดังรูปที่ 5.7 ตามลำดับ

ตารางที่ 5.2 การจัดสรรแอดเดรสที่ใช้ติดต่อกับอุปกรณ์ภายนอกบน IBM PC

พื้นที่แอดเดรสของอุปกรณ์ I/O										
เลขฐานสิบหก	9	8	7	6	5	4	3	2	1	0	อุปกรณ์ I/O
00-0F	0	0	0	0	0	Z	A3	A2	A1	A0	DMA CHIP B237-2
20-21	0	0	0	0	1	Z	Z	Z	Z	A0	INTERRUPT 8259A
40-43	0	0	0	1	0	Z	Z	Z	A1	A0	TIMER 8253-5
60-63	0	0	0	1	1	Z	Z	Z	A1	A0	PPI 8255A-5
80-83	0	0	1	0	0	Z	Z	Z	A1	A0	DMA PAGE REGS
AX	0	0	1	0	1						NMI MASK REG
CX	0	0	1	1	0						RESERVED
EX	0	0	1	1	1						RESERVED
3F8-3FF	1	1	1	1	1	1	1	A2	A1	A0	TP RS-232-C CD
3F0-3F7	1	1	1	1	1	1	0	A2	A1	A0	5¼" DRV ADAPTOR
2F8-2FF	1	0	1	1	1	1	1	Z	A1	A0	RESERVED
378-37F	1	1	0	1	1	1	1	Z	A1	A0	PARALLEL PRTR PRT
3D0-3DF	1	1	1	1	0	1	A3	A2	A1	A0	COLOR/GRAPHICS ADAPTER
278-27F	1	0	0	1	1	1	1	Z	A1	A0	RESERVED
200-20F	1	0	0	0	0	0	A3	A2	A1	A0	GAME I/O ADAPTER
3B0-3BF	1	1	1	0	1	1	A3	A2	A1	A0	IBM MONOCHROME DISPLAY
											PARALLEL PRINTER ADAPTER



รูปที่ 5.7 การ์ดที่ใช้เสียบใน I/O Slot (PC-card)



รูปที่ 5.8 รูปถ่ายบอร์ด RS - 232

รายการอุปกรณ์บอร์ด RS - 232

1.	ไอซี เบอร์ 8250	1	ตัว
2.	ไอซี เบอร์ 74 LS 245	1	ตัว
3.	ไอซี เบอร์ 74 LS 30	1	ตัว
4.	ไอซี เบอร์ 74 LS 04	1	ตัว
5.	ไอซี เบอร์ 74 LS 32	1	ตัว
6.	XTAL 1.8342 Mhz	1	ตัว
7.	ตัวต้านทาน 1 K Ω 1/4 W	1	ตัว
8.	ตัวต้านทาน 1.5 K Ω 1/4 W	1	ตัว
9.	ตัวต้านทาน 1 M Ω 1/4 W	1	ตัว
10.	คาปาซิเตอร์ 30 PF ไม่มีขั้ว	1	ตัว
11.	คาปาซิเตอร์ 56 PF ไม่มีขั้ว	1	ตัว
12.	connector ต่างๆ		

5.4 ส่วนของโปรแกรมฝ่ายส่ง

โปรแกรมการทำงานในส่วนนี้จะเขียนขึ้น โดยโปรแกรมภาษาแอสแซมบลี้ แล้วทำการแปลง FILE เป็น FILE . OBJ โดยใช้โปรแกรม X80.EXE และแปลงเป็น FILE .HEX โดยใช้โปรแกรม LINK.EXE ตามลำดับ จากนั้นนำโปรแกรมที่ได้ COPY ลง EPOM แล้วนำไปต่อร่วมกับบอร์ดไมโครโปรเซสเซอร์ ซึ่งโครงงานนี้ใช้ Z80 เป็น CPU

5.4.1 หลักการทำงานของโปรแกรม

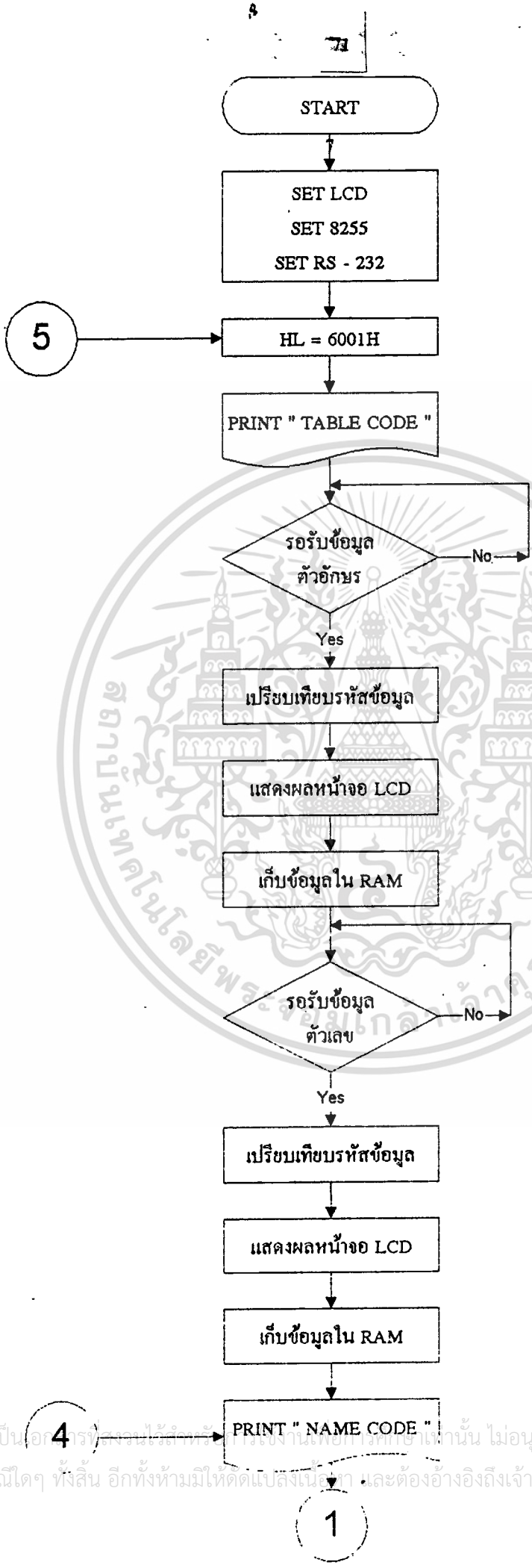
โดยในส่วนของโปรแกรมนี้จะรับ INPUT จาก KEYBOARD และส่ง OUTPUT ออก 2 ทาง คือ แสดงผลทางจอ LCD และส่งข้อมูล ไปยังการ์ด RS-232

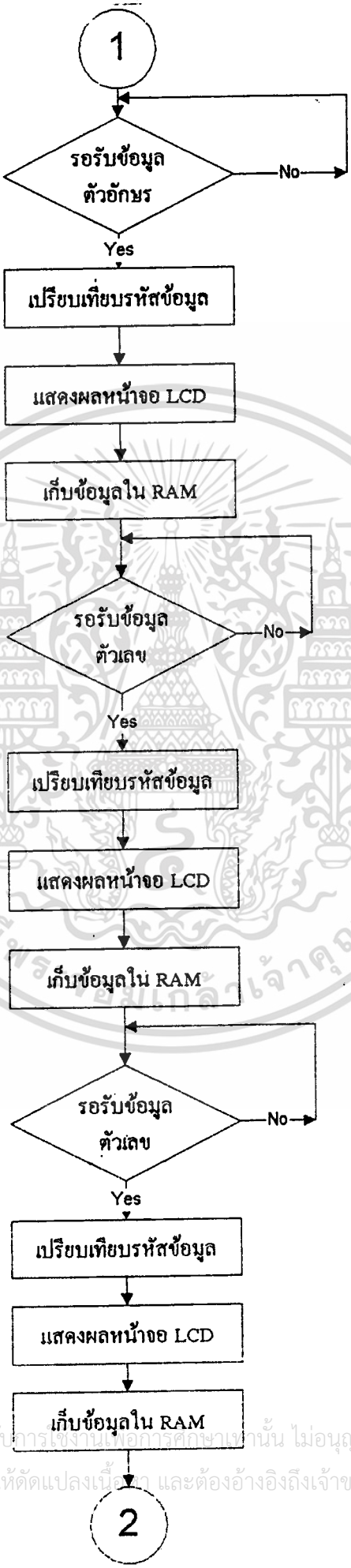
หลักการทำงานของโปรแกรมเริ่มจาก รับข้อมูลจาก KEYBOARD ขนาด 8 BIT (โดยในส่วนของ HARD WARE ของ KEYBOARD จริงๆแล้ว จะให้ข้อมูลขนาด 5 BIT ส่วนอีก 3 BIT จะทำให้คงสภาวะไว้ที่ 0 ผลทำให้ได้ข้อมูล 8 BIT) จากนั้นนำข้อมูลที่ได้มา ทำการเทียบกับตารางภายในโปรแกรม เพื่อแปลงรหัสที่ได้ให้เป็นรหัส ASCII ต่อจากนั้นนำรหัส ASCII ส่งออกไปยังส่วนของ LCD แสดงผลของข้อมูลออกทางจอภาพ ส่วน OUTPUT อีกด้านหนึ่งจะส่งออกไปยังการ์ด RS-232 เพื่อส่งข้อมูลไปยังปลายทางต่อไป

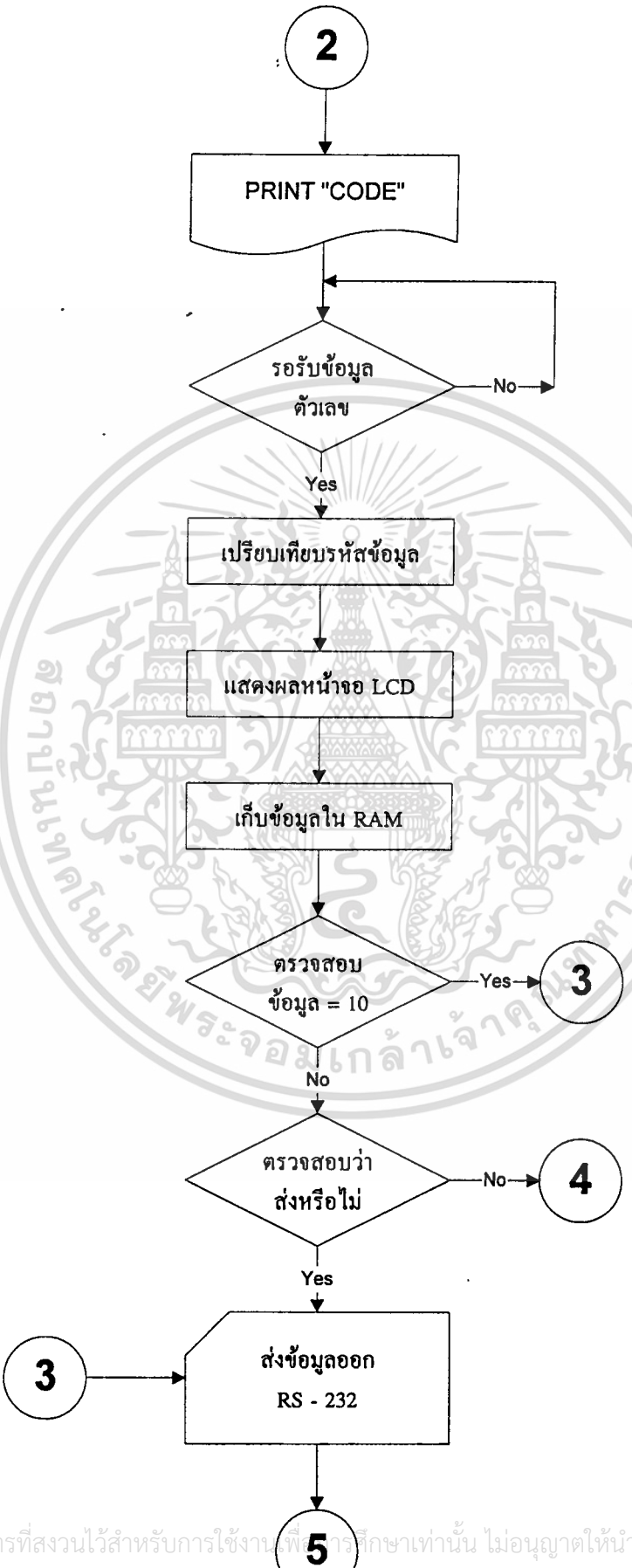
ลักษณะในการกำหนดรูปแบบข้อมูล ในโครงงานเป็นดังนี้

- รหัสเริ่มต้นข้อมูล ขนาด 1 BYTE
- รหัส TABLE ขนาด 2 BYTE
- รหัส NAME ขนาด 3 BYTE จำนวน 10 ชุด
- รหัสจำนวน ขนาด 1 BYTE จำนวน 10 ชุด
- รหัสสิ้นสุดข้อมูล ขนาด 1 BYTE

การส่งข้อมูลจะส่งเมื่อข้อมูลครบสมบูรณ์แล้ว กล่าวคือ รวมส่วนข้อมูลที่ต้องการส่งครบตามต้องการกับส่วนแสดงการเริ่มต้นและสิ้นสุดข้อมูลที่ส่งแล้วส่งไปพร้อมกัน โดยส่งผ่านการ์ด RS-232 ไปยังเครื่องไมโครคอมพิวเตอร์







รูป FLOWCHART ของโปรแกรมฝ่ายส่ง

5.5โปรแกรมแสดงผลฝ่ายรับ

ในส่วนของฝ่ายรับซึ่งเป็นส่วนของ PC จะรับข้อมูลจาก PORT โดยจะตรวจสอบสัญญาณว่ามี * หรือไม่ ถ้ามีจะนำข้อมูลที่ส่งมา นำมาอ่านค่าโต๊ะ,ระหัสอาหารและจำนวนอาหารจาก นั้นก็จะตรวจสอบ # ซึ่งเป็นการแสดงการสิ้นสุดรายการต่างๆที่ส่งมาแต่ถ้าไม่มี # จะกลับไปอ่านค่าระหัสอาหาร,จำนวนอาหารใหม่ เมื่อพบ # โปรแกรมจะทำการคำนวนค่าอาหารทั้งหมดที่โต๊ะนั้นสั่งมาและนำไปแสดงผลที่หน้าจอ

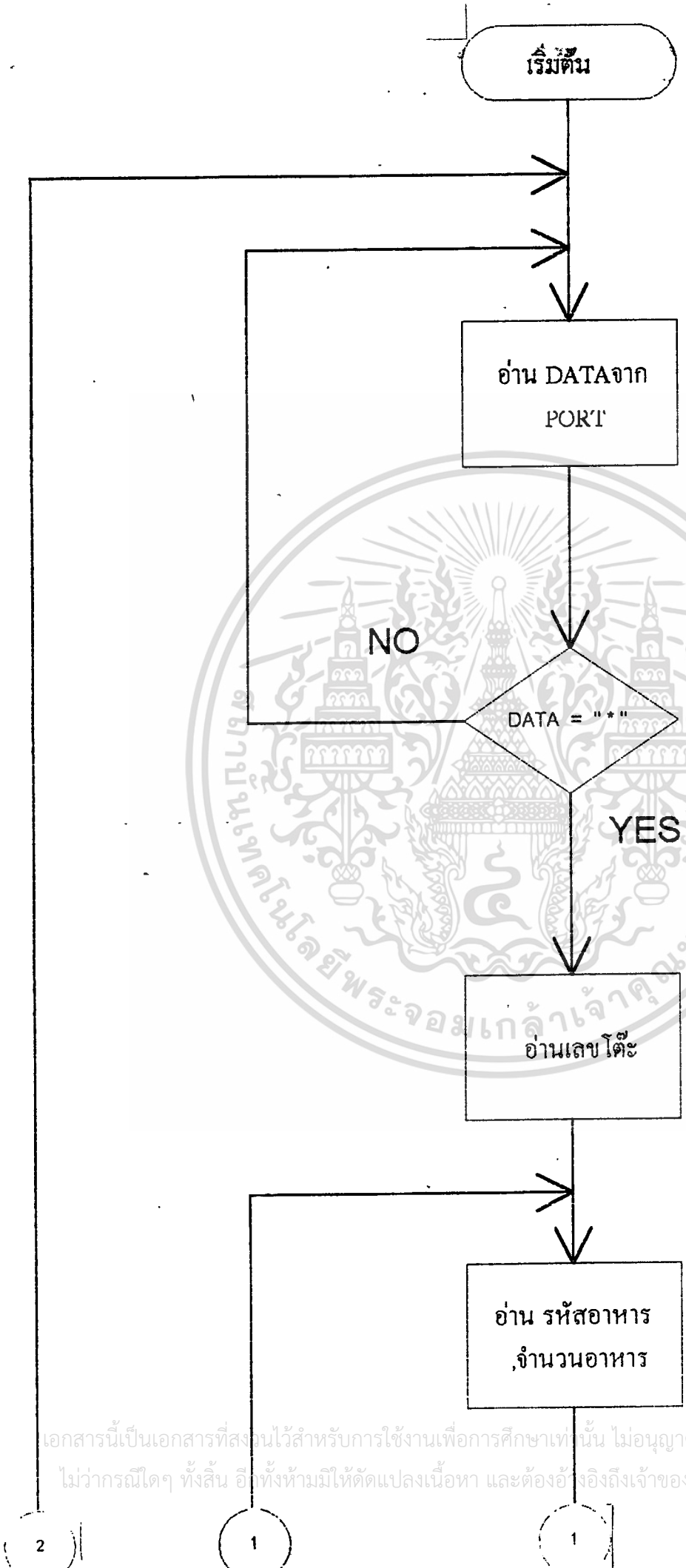
สัญญาณที่ใช้

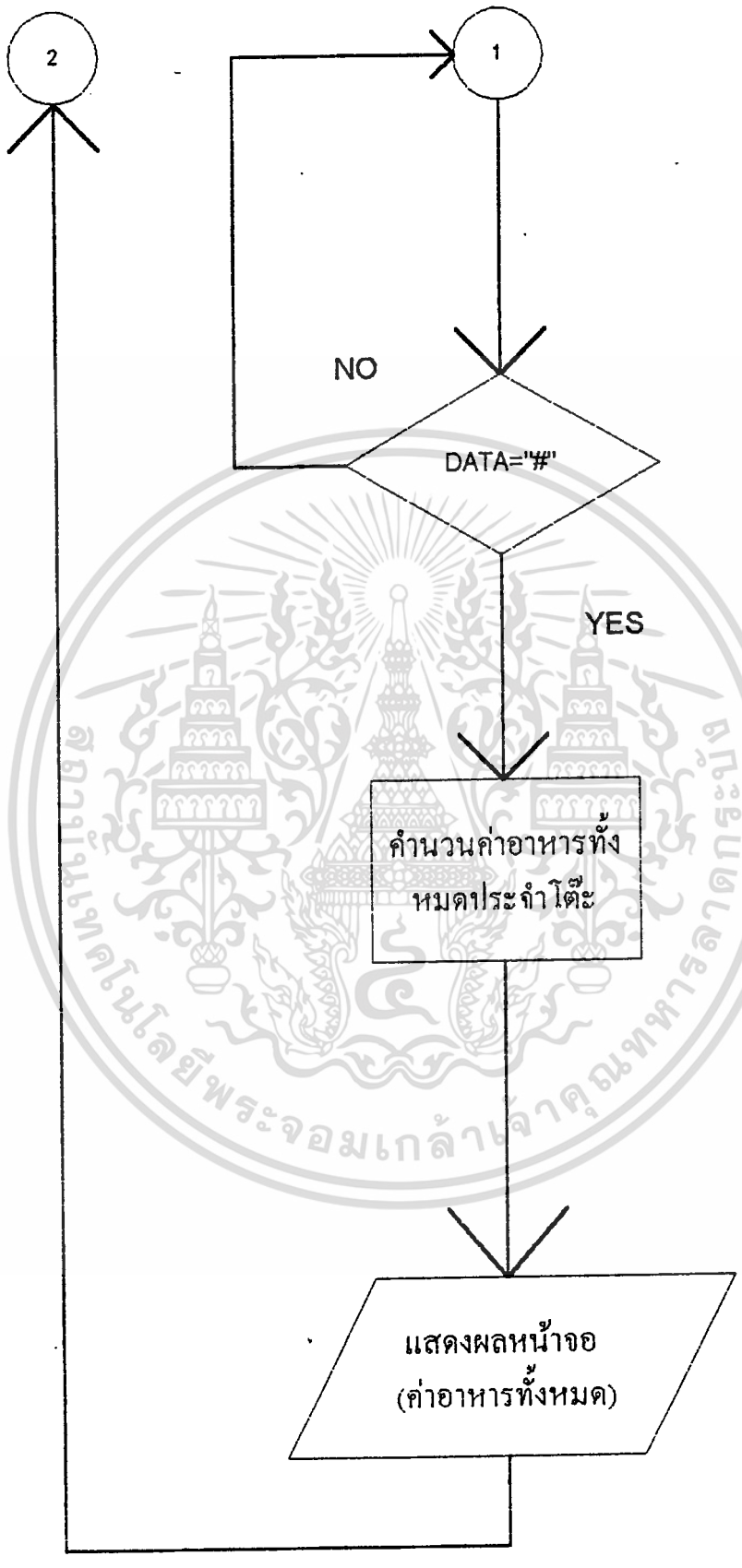
1. " * " เป็นสัญญาณเริ่มส่ง
2. " # " เป็นสัญญาณสิ้นสุดการส่ง

รูปแบบของข้อมูล

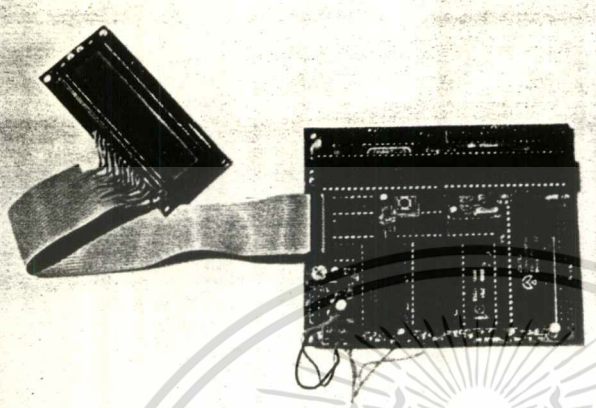
1. เลขโต๊ะ มีข้อมูล 2 บิต
2. ระหัสอาหาร มีข้อมูล 3 บิต
3. จำนวนอาหาร มีข้อมูล 1 บิต

เช่น ถ้ามีการสั่งอาหารจากโต๊ะ A1 โดยสั่งอาหารที่มีระหัส A01,A02,A03 จำนวนอย่างละ3ที่ จะมีข้อมูลดังนี้ "*A1A013A023A033#"

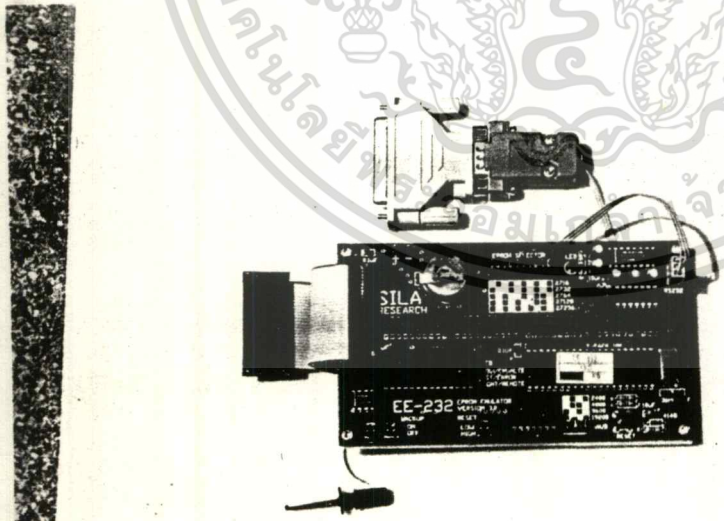




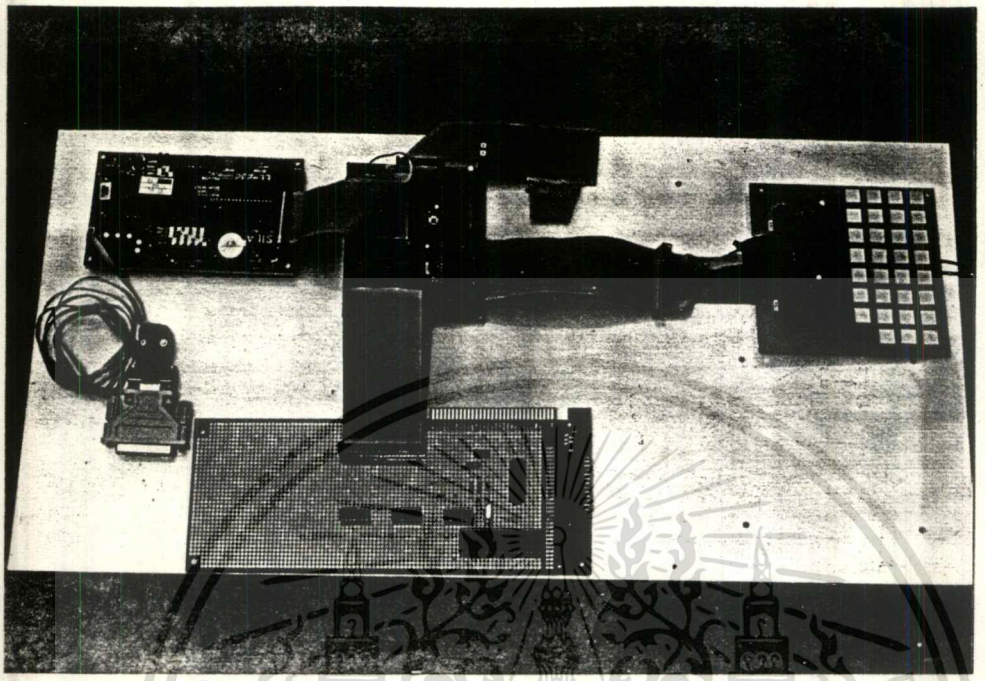
รูป FLOWCHART ของโปรแกรมแสดงผลของฝ่ายรับ



รูปที่ 5.9 รูปบอร์ด CP Z80-V2



รูปที่ 5.10 รูปบอร์ด EPROM EMULATOR



รูปที่ 5.11 รูปชิ้นงานทั้งหมดของโครงงาน

หนังสืออ้างอิง

1. วิบูลย์ ชื่นแขก " ไมโครโปรเซสเซอร์ " สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ , หน้าที่ 27 - 51 , 88 - 131 , 202 - 210 , 269 - 281
2. ทีมงาน ETT " DOT MATRIX LCD MODULE " บริษัท อีทีที จำกัด , หน้าที่ 1 - 27
3. ทีมงาน ETT " คู่มือการทดลอง ET-HARDWARE LAB " บริษัท อีทีที จำกัด , หน้าที่ 1 - 24 , 30 -44
4. กิตติ องค์คุณารักษ์ " แอดวานซ์แอสเซมบลี " บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , หน้าที่ 43 - 69
5. ทีมงาน ETT " ET-Z80 HARDWARE EXPERIMENT " บริษัท อีทีที จำกัด , หน้าที่ 9 - 27 , 60 - 65 , 75 - 91
6. คู่มือไอซี ทีทีแอล MOTOROLA
7. คู่มือ ECG Semiconductors Master Replacement Guide , หน้าที่ 1-41 (ECG 123AP) ,TO-92 , T16
8. ทีมงาน ETT " CP-Z80 V2 Smart Controllor " บริษัท อีทีที จำกัด , หน้าที่ 1 - 10
9. ยืนภู่ วรวรรณ " เทคโนโลยีฮาร์ดแวร์ " บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , หน้าที่ 126 -148
10. ปราโมทย์ วาดเขียน , ดร. วิวัฒน์ กิรานนท์ " พื้นฐานการสื่อสารข้อมูล " สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , หน้าที่ 202 - 214

ภาคผนวก ก

(โปรแกรมทางด้านส่ง)

```
    ORG 0000H
PA1: EQU 30H
PB1: EQU 31H
PC1: EQU 32H
PCC1: EQU 33H
PA2: EQU 40H
PB2: EQU 41H
PC2: EQU 42H
PCC2: EQU 43H
WR_COMM: EQU 60H
RD_COMM: EQU 64H
WR_DATA: EQU 62H
;   ****************************
START: LD SP,9FFFH
       LD B,080H
       LD L,080H
ST1: DEC L
     LD A,00H
     CP L
     JR NZ,ST1
     DJNZ ST1
     CALL RESET
     CALL DELAYL
     CALL T8255
     CALL DELAYL
WORK1: CALL CLSLCD
       LD HL,6001H
READ11: CALL CLSLCD
          PUSH HL
          LD HL,TAB3
          CALL WRP
          POP HL
          LD B,80H
          CALL TABLE1
READ12: LD B,81H
          CALL TABLE2
```

```
ENTER1: IN A,(PA1)
        OR 00H
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,ENTER1
        LD A,E
        CP 11H
        JP Z,WORK1
        LD A,E
        CP 09H
        JR NZ,ENTER1
READ21: CALL CLSLCD
        PUSH HL
        LD HL,TAB4
        CALL WRP
        POP HL
        LD B,80H
        CALL TABLE1
READ22: LD B,81H
        CALL TABLE2
READ23: LD B,82H
        CALL TABLE2
ENTER2: IN A,(PA1)
        OR 00H
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,ENTER2
        LD A,E
        CP 11H
        JP Z,WORK1
        LD A,E
        CP 09H
        JR NZ,ENTER2
```

```
READ31: CALL CLSLCD
        PUSH HL
        LD HL,TAB5
        CALL WRP
        POP HL
        LD B,80H
READ311: IN A,(PA1)
        OR 00H
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,READ311
        LD A,E
        CP 11H
        JP Z,WORK1
        CALL CLSLCD
        CALL TABLE22
ENTER3: IN A,(PA1)
        OR 00H
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,ENTER3
        LD A,E
        CP 11H
        JP Z,WORK1
        LD A,E
        CP 09H
        JR NZ,ENTER3
LOOP: LD A,L
    CP 35H
    JP Z,SEND
    CALL CLSLCD
    PUSH HL
    LD HL,TAB6
```

```
        CALL WRP
        POP HL
LOOP1: IN A,(PA1)
        OR 00H
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,LOOP1
        CALL CLSLCD
        LD A,E
        CP 08H
        JP Z,LOOP11
        LD A,E
        CP 15H
        JP Z,LOOP22
        LD A,E
        CP 11H
        JP Z,WORK1
        JR LOOP1
;       **************************
LOOP11: LD B,80H
            LD E,4EH
            CALL GOTO12
            CALL WRBYTE
LOOP12: IN A,(PA1)
            OR 00H
            CALL DELAYL
            LD E,A
            CP 1FH
            JP Z,LOOP12
            LD A,E
            CP 11H
            JP Z,WORK1
            LD A,E
            CP 09H
```

```
        JP Z,READ21
        JP LOOP12
;   ***************************
LOOP22: LD B,80H
        LD E,59H
        CALL GOTO12
        CALL WRBYTE
LOOP23: IN A,(PA1)
        OR 00H
        CALL DELAYL
        CALL DELAYL
        LD E,A
        CP 1FH
        JP Z,LOOP23
        LD A,E
        CP 11H
        JP Z,WORK1
        LD A,E
        CP 09H
        JP Z,SEND
        JP LOOP23
;   ************************
SEND: CALL CLSLCD
    PUSH HL
    LD HL,TAB7
    CALL WRP
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    CALL DELAYL
    POP HL
```

```
        LD A,2AH
        LD (6000H),A
        PUSH DE
        LD DE,6000H
SEND1: LD A,(DE)
        OUT (PB1),A
        INC DE
        LD A,L
        CP E
        JR NZ,SEND1
        POP DE
        JP WORK1
;       ***************************
TABLE1: IN A,(PA1)
        CALL DELAYL
        LD E,A
        CP 1FH
        JR Z,TABLE1
        CALL CLSLCD
        LD A,E
        CP 11H
        JP Z,WORK1
        LD A,E
        CP 07H
        JP Z,CPP11
        LD A,E
        CP 13H
        JP Z,CPP11
        LD A,E
        CP 0FH
        JP Z,CPP11
        LD A,E
        CP 19H
        JP Z,CPP11
```

```
        LD A,E
        CP 0EH
        JP Z,CPP11
        LD A,E
        CP 1BH
        JP Z,CPP11
        LD A,E
        CP 0DH
        JP Z,CPP11
        LD A,E
        CP 03H
        JP Z,CPP11
        LD A,E
        CP 1CH
        JP Z,CPP11
        LD A,E
        CP 00H
        JP Z,CPP11
        LD A,E
        CP 04H
        JP Z,CPP11
        JP TABLE1
ENDT1: CALL DELAYL
     RET
;    ***************************
CPP11: CALL CPP
     CALL GOTO12
     CALL WRBYTE
     CALL GET
     JP ENDT1
;    ***************************
TABLE2: IN A,(PA1)
        OR 00H
        CALL DELAYL
```

```
        LD E,A
        CP 1FH
        JR Z,TABLE2
        LD A,E
        CP 11H
        JP Z,WORK1
TABLE22: LD A,E
        CP 1AH
        JP Z,CPP22
        LD A,E
        CP 18H
        JP Z,CPP22
        LD A,E
        CP 1DH
        JP Z,CPP22
        LD A,E
        CP 06H
        JP Z,CPP22
        LD A,E
        CP 16H
        JP Z,CPP22
        LD A,E
        CP 0CH
        JP Z,CPP22
        LD A,E
        CP 0BH
        JP Z,CPP22
        LD A,E
        CP 14H
        JP Z,CPP22
        LD A,E
        CP 17H
        JP Z,CPP22
        LD A,E
        CP 12H
        JP Z,CPP22
```

```
ENDT2: CALL DELAYL
       RET
;      ***************************
CPP22: CALL CPP
       CALL GOTO12
       CALL WRBYTE
       CALL GET
       JP ENDT2
;      ***************************
CPP: LD A,E
     PUSH HL
     LD HL,DATA1
     ADD A,L
     LD L,A
     LD A,00
     ADC A,H
     LD A,(HL)
     LD E,A
     POP HL
     RET
;      ***************************
GOTO12: LD A,B
          SET 7,A
          OUT (WR_COMM),A
          CALL RD_BUSY
          RET
;      ***************************
GET: LD A,E
     LD (HL),A
     INC HL
     RET
;      ***************************
RESET: CALL INITLCD
       PUSH HL
       LD HL,TAB1
       CALL WRP
```

```
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        LD HL,TABRES
        CALL WRP
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        CALL DELAYL
        POP HL
        CALL DELAYL
        RET
;       ************************
INITLCD: LD A,38H
         OUT (WR_COMM),A
         CALL DELAYL
         LD A,0FH
         OUT (WR_COMM),A
         CALL DELAYL
         LD A,06H
         OUT (WR_COMM),A
         CALL DELAYL
CLSLCD: PUSH AF
        LD A,1
        OUT (WR_COMM),A
        POP AF
        RET
```

```
;   *************************
DELAYL: PUSH AF
        PUSH BC
        PUSH DE
        PUSH HL
        LD HL,9000H
DELAY1: DEC HL
        LD A,H
        OR L
        JR NZ,DELAY1
        POP HL
        POP DE
        POP BC
        POP AF
        RET
;   *************************
WRP: LD A,80H
    CALL GOTO
    CALL WRLINE
    LD A,C0H
    CALL GOTO
    CALL WRLINE
    RET
;   *************************
GOTO: SET 7,A
     OUT (WR_COMM),A
     CALL RD_BUSY
     RET
;   *************************
RD_BUSY: PUSH AF
RD_BUSY1: IN A,(RD_COMM)
        BIT 7,A
        JR NZ,RD_BUSY1
        POP AF
        RET
```

```
;    ************************
WRLINE: LD B,8
TEST1: LD A,(HL)
       LD E,A
       CALL WRBYTE
       INC HL
       DJNZ TEST1
       RET
;    ************************
WRBYTE: LD A,E
          OUT (WR_DATA),A
          CALL RD_BUSY
          RET
;    ************************
T8255: PUSH HL
       LD HL,TAB2
       CALL WRP
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       CALL DELAYL
       POP HL
       LD A,10010010B
       OUT (PCC1),A
       LD A,10010000B
       OUT (PCC2),A
       CALL DELAYL
       RET
;      ************************
TAB1: DB "[*KMITL ELEC*] "
;      END
```

```
;       ************************
TABRES: DB "[* CPU RESET *] "
;        END
;       ************************
TAB2: DB " [* SET 8255 *] "
;       END
;       ************************
TAB3: DB "[* TABLE CODE *]"
;       END
;       ************************
TAB4: DB "[* NAME CODE *] "
;       END
;       ************************
TAB5: DB " [ * AMOUNT * ] "
;       END
;       ************************
TAB6: DB " [* SEND Y/N *] "
;       END
;       ************************
TAB7: DB "[* SEND DATA *] "
;       END
;       ************************
DATA1: DB 4AH,00H,00H,48H,4BH
       DB 00H,34H,41H,00H,00H
       DB 00H,37H,36H,47H,45H
       DB 43H,00H,00H,30H,42H
       DB 38H,00H,35H,39H,32H
       DB 44H,31H,46H,49H,33H
       DB 00H
       END
;       ************************
```

ภาคผนวก ข

(โปรแกรมทางด้านรับ)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include<dos.h>
#include<time.h>
#include<graphics.h>

#define ESC1 27                    /* ESC button used for exit program */
#define RXbuffer 0x2f8             /* receiver buffer */
#define TXbuffer 0x2f8             /* transmitter buffer */
#define IER      0x2f9             /* Interupt Enable register */
#define IID      0x2fa             /* Interupt Identification register */
#define LCR      0x2fb             /* Line control register */

#define MDC      0x2fc             /* Modem control register */
#define LST      0x2fd             /* Line status register */
#define MST      0x2fe             /* Modem status register */
#define DLL    0x2f8               /* Divisor latch Lower byte */
#define DLM      0x2f9             /* Divisor latch Upper byte */

typedef unsigned char BYTE;        /* use BYTE for one byte data */
typedef unsigned int WORD;         /* use WORD for two bytes data */

void init_8250(void);              /* initial control word for 8250 */
void send_one_char(BYTE data);     /* used for senf one character to port */
BYTE receive_one_char(void);       /* used for receive one character from
port */
BYTE read_kb(void);                /* used for read data from KB */

void clearbox(int left,int up,int right,int low);
void blackbox(int left,int up,int right,int low);
void wall(void);
void screen(void);
void bye(void);



#define ESC 0x1b
#define UP 0x4800
#define DOWN 0x5000
#define ENTER 0x1c0d
#define Max 30000
#define Min 3000


#define TXT_BACKG 9
#define TXT_LOGO 14
```

```
#define TXT_MENU 12
#define TXT_SELECT 14
#define TXTB_SELECT 14
#define TXT_TABLE 15


char *menu_msg[]={
        "   [ CREAT NEW MENU    ] ",
        "   [ OPEN MENU FILE    ] ",
        "   [ SEARCH BY KEYBORD ] ",
        "   [ ORDER RECEIVER    ] ",
        "   [ EDIT RECORD       ] ",
        "   [ APPEND RECORD     ] ",
        "   [ INSERT RECORD     ] ",
        "   [ DELET RECORD      ] ",
        "   [****   EXIT    ****]",
      };


struct data{
        char number[5];
        char code[7];
        char name[40];
        int price1;
        }var,var1,var2;

time_t start,end;
int loop,check,v,di[6];
int nfile,speed=105;
unsigned int wait_key;
unsigned char title[Max];
char s_code[7];
int list,list_menu;
BYTE data_kb,table[3],code_food[20][7],amoung[20][2],data_port[100];
char code_f[7];

main()
{
        int n=0,b,out=0;
        int i,c,x=25,y=5 ;

        init_8250();
        screen();
        gotoxy(34,25);
        printf("please any key");
        getch();
        cur_off();
        textmode(3);
        textbackground(TXT_BACKG);
```

```
clrscr();
box(2,1,79,25,0);
name();

do
{

        window(3,7,78,24);
        clrscr();
        box(22,4,55,14,1);
        textcolor(13);
        gotoxy(21,2);
        cprintf("!!! WELL COME TO AUTOMETIC MENU !!!");
        textcolor(TXT_MENU);
        for(i=0;i<=8;++i)
        {
                gotoxy (x,y+i);
                cprintf(menu_msg[i]);
        }
        n=select_menu(x,y,n);
        clrscr();
        switch(n)
        {
        case 0:
                out=creatnf();
                break;
        case 1:
                out=readof();
                break;
        case 2:
                out=searchkey();
                break;
        case 3:

                out=search_port();

                break;
        case 4:
                window(1,1,80,25);

                out=editrec();
                break;
        case 5:
                out=appendrec();
                break;
        case 6:

                window(1,1,80,25);
```

```
                out=insrec();
                break;
        case 7:

                window(1,1,80,25);

                out=delrec();
                break;
        case 8:
                out=1;
                clrscr();
                break;
        }
    }while(out!=1);

    textmode(-1);
    window(1,1,80,25);
    clrscr();
    cur_on();
}


/*-------------------------------------------------------*/
creatnf()
{
    FILE *fp;
    char file_name[15],m_price[4];
    gotoxy(28,1);printf("# CREAT NEW FILE #");

    gotoxy(21,8);printf("Enter file name : ");

    bigcur();
    gets(file_name);
    cur_off();
    if((fp = fopen(file_name,"wb")) == NULL)
    {
        clrscr();
        printf("\007");
        gotoxy(28,8);cputs("Error in open file\n");
        window(1,1,80,25);
        gotoxy(32,24);cputs(" Pless any key ");
        getch();
        return;
    }
    do
    {
        clrscr();
        gotoxy(22,5);printf("NUMBER : ");
```

```
                bigcur();
                gets(var.number);
                cur_off();
                if(var.number[0] != '\0')
                {
                        gotoxy(22,7);printf("Code   : ");
                        bigcur();
                        gets(var.code);
                        gotoxy(22,9);printf("Name   : ");
                        gets(var.name);
                        gotoxy(22,11);printf("Price   : ");
                        gets(m_price);
                        cur_off();
                        var.price1=atoi(m_price);
                        fwrite(&var,sizeof var,1,fp);
                        if(ferror(fp))
                        {
                                clrscr();
                                gotoxy(28,8);printf("Error in writing\n");
                                window(1,1,80,25);
                                gotoxy(32,24);cputs(" Pless any key ");
                                fclose(fp);
                                getch();
                                var.number[0]='\0';
                        }
                }
        }while(var.number[0] != '\0');
        printf("\007");
        fclose(fp);
}
/*-------------------------------------------------------------------
*/
readof()
{
        FILE *fp;
        int i;
        char ans,file_name[15];
        gotoxy(28,1);printf("# READ OLD FILE #");
        gotoxy(21,8);printf("Enter file name : ");
        bigcur();
        gets(file_name);
        cur_off();
        if((fp=fopen(file_name,"rb")) ==NULL)
        {
                clrscr();
                printf("\007");
                gotoxy(28,8);printf("Error in open file\n");
                window(1,1,80,25);
                gotoxy(32,24);cputs(" Pless any key ");
```

```
                getch();
                return;
        }
        while(fread(&var,sizeof var,1,fp) ==1)
        {
                if(ferror(fp))
                {
                        clrscr();
                        gotoxy(28,8);printf("Error in file\n");
                        window(1,1,80,25);
                        gotoxy(32,24);cputs(" Pless any key ");
                        getch();
                        return;
                }
                window(3,7,74,23);
                clrscr();
                gotoxy(22,5);printf("Number : %s",var.number);
                gotoxy(22,7);printf("Code   : %s",var.code);
                gotoxy(22,9);printf("Name   : %s",var.name);
                gotoxy(22,11);printf("Price  : %d",var.price1);
                window(1,1,80,25);
                gotoxy(11,24);
                printf(" Q = Exit to MAIN MENU , Any other keys = Next to
RECORD ");
                ans=getch();
                if(toupper(ans)=='Q') return;
        }
        printf("\007");
        fclose(fp);
}
/*-------------------------------------------------------------
*/
searchkey()
{
        FILE *fp;
        int x;
        char k_code[7],file_name[15],ch;
        gotoxy(26,1);printf("# SEARCH BY KEYBORD #");
        window(3,7,74,23);
        gotoxy(21,8);printf("Enter file name : ");
        bigcur();
        gets(file_name);
        cur_off();
        if((fp = fopen(file_name,"rb")) == NULL)
        {
                printf("\007");
                clrscr();
                gotoxy(28,8);cputs("Error in open file\n");
                window(1,1,80,25);
```

```
                gotoxy(32,24);cputs(" Pless any key ");
                getch();
                return;
        }
        do
        {
                clrscr();
                gotoxy(20,8);printf("Enter code for seraching : ");
                bigcur();
                gets(k_code);
                cur_off();
                while((fread(&var,sizeof var,1,fp) == 1) &&
strcmp(k_code,var.code))
                {
                        if(ferror(fp))
                        {
                                clrscr();
                                printf("\007");
                                gotoxy(28,8);cputs("Error in file\n");
                                window(1,1,80,25);
                                gotoxy(32,24);cputs(" Pless any key ");
                                fclose(fp);
                                getch();
                                return;
                        }
                }
                if(!strcmp(k_code,var.code))
                {
                        clrscr();
                        gotoxy(22,5);printf("Number : %s",var.number);
                        gotoxy(22,7);printf("Code   : %s",var.code);
                        gotoxy(22,9);printf("Name   : %s",var.name);
                        gotoxy(22,11);printf("Price  : %d",var.price1);
                }
                else
                {
                        printf("\007");
                        clrscr();
                        gotoxy(20,8);printf("No such code (%s) in the file\n",k_code);
                }
                window(1,1,80,25);
                gotoxy(15,24);printf(" Q = Quit, Any other keys = Continue
Searching ");
                ch=getch();
                gotoxy(3,24);
                for(x=3;x<=74;x++) putch(205);
                window(3,7,74,23);
                rewind(fp);
        }while(toupper(ch) != 'Q');
```

```
        fclose(fp);
}
/*-------------------------------------------------------------------*/

search_port()
{
        FILE *fp;
        int x,l ,list1;
        char k_code[7],file_name[15],ch;
        int amoung1,sum,total,data;
        gotoxy(30,1);printf("# READ TABLE #");
        window(3,7,74,23);
        gotoxy(21,8);printf("Enter file name : ");
        bigcur();
        gets(file_name);
        cur_off();
        if((fp = fopen(file_name,"rb")) == NULL)
        {
                printf("\007");
                clrscr();
                box(2,1,79,25,0);
                gotoxy(28,8);cputs("Error in open file\n");
                window(1,1,80,25);
                gotoxy(32,24);cputs(" Pless any key ");
                getch();

                clrscr();
                return;
        }
                window(3,9,74,23);
                clrscr();
                window(3,7,74,23);

        do
        {
                gotoxy(24,17);printf(": Waiting receive MENU FOOD : ");
                bigcur();
                data_kb='0';
                decode_data();
                cur_off();
                total=0;
                for(list=0;list<list_menu;list++)
                {
                        if(list==0)
                        {
                                window(3,9,74,23);
                                clrscr();
                                window(3,7,74,23);
```

```
                        gotoxy(28,2);
                        printf(" TABLE NUMBER : %s ",table);
                        gotoxy(7,4);printf(" CODE ");
                        gotoxy(14,4);printf(" NAME ");
                        gotoxy(51,4);printf(" PRICE ");
                        gotoxy(58,4);printf(" AMOUNT ");
                        gotoxy(70,4);printf(" SUM ");
                        fprintf(stdprn," \t\t\t TABLE NUMBER:%s\n\n",table);

        fprintf(stdprn,"CODE\tNAME\t\t\t\t\tPRICE\tAMOUNT\tSUM\n\n");
                    }


                    while((fread(&var,sizeof var,1,fp) == 1) && strcmp
(code_food[list],var.code))
                    {
                        if(ferror(fp))
                        {
                            clrscr();
                            printf("\007");
                            gotoxy(28,8);cputs("Error in file\n");
                            window(1,1,80,25);
                            gotoxy(32,24);cputs(" Pless any key ");
                            fclose(fp);
                            getch();
                            data_kb='Q';
                        }
                    }
                    if(!strcmp(code_food[list],var.code))
                    {

                        amoung1=atoi(amoung[list]);
                        sum=amoung1*var.price1;
                        total=total+sum;

                        gotoxy(4,6+list);printf(" %d. %s ",list+1,var.code);
                        gotoxy(14,6+list);printf(" %s ",var.name);
                        gotoxy(53,6+list);printf(" %d ",var.price1);
                        gotoxy(61,6+list);printf(" %s ",amoung[list]);
                        gotoxy(70,6+list);printf(" %d ",sum);
                        fprintf(stdprn,"%d.%-5s\t%-
40s%d\t%s\t%d\n\n",list+1,var.code,var.name,var.price1,amoung[list],sum);

                    }
                    else
                    {
                        printf("\007");
                        gotoxy(8,6+list);printf("No such code (%s) in the
file\n",code_food[list]);
```



```
                    }
```

```
                }
                de_lay(500);
                rewind(fp);
            }
            if((kbhit()!=0))data_kb=getch();
            gotoxy(62,6+list+1);printf(" TOTAL : %d ",total);
            fprintf(stdprn,"\t\t\t\t\t\t\t TOTAL : %d\n\n ",total);
        }while(data_kb!=ESC1);
        fclose(fp);
}
/*-------------------------------------------------------------------------*/
de_lay(int i)
{
        int j;
        for(j=0;j<=i;j++);
}
/*-------------------------------------------------------------------------*/


select_menu(int x,int y,int n )
{
        int in;
        high(n,x,y,1);
        for(;;)
        {
            in=bioskey(0);
            switch (in)
            {
                case UP : n--;
                if(n<0)
                {
                    n=8;
                    high(n,x,y,1);
                    n=0;
                    high(n,x,y,0);
                    n=8;
                }
                else
                {
                    high(n,x,y,1);
                    n++;
                    high(n,x,y,0 );
                    n--;
                }
                break;
                case DOWN :n++;
                if(n>8)
                {
                    n=0;
```

```
                        high(n,x,y,1);
                        n=8;
                        high(n,x,y,0);
                        n=0;
                }
                else
                {
                        high(n,x,y,1);
                        n--;
                        high(n,x,y,0);
                        n++;
                }
                break;
                case ENTER:
                        return n;
        }
    }
}
/*-------------------------------------------------------------------*/
high(int n,int x,int y,int h1)
{
    if(h1==1)
    {
        textcolor(TXT_SELECT);
        textbackground(TXT_BACKG);

        gotoxy(x,y+n);
        cprintf(menu_msg[n]);
        textcolor(TXT_MENU);
        textbackground(TXT_BACKG);
    }
    else
    {
        textcolor(TXT_MENU);
        textbackground(TXT_BACKG);
        gotoxy(x,y+n);
        cprintf(menu_msg[n]);
    }
}
/*-------------------------------------------------------------------*/
cur_off()
{
    union REGS regs;
    regs.h.ch=0x20;
    regs.h.cl=0;
    regs.h.ah=1;
    int86(0x10,&regs,&regs);
}
/*-------------------------------------------------------------------*/
```

```
cur_on()
{
        union REGS regs;
        regs.h.ch=12;
        regs.h.cl=13;
        regs.h.ah=1;
        int86(0x10,&regs,&regs);
}
/*-------------------------------------------------------------------------*/
bigcur()
{
        union REGS regs;
        regs.h.ch=1;
        regs.h.cl=10;
        regs.h.ah=1;
        int86(0x10,&regs,&regs);
}
/*-------------------------------------------------------------------------*/
box(int xul,int yul,int xlr,int ylr,int j)
{
        char  hz,vt,ul,ur,lr,ll,lt,lm;
        textcolor(14);
        textbackground(TXT_BACKG);
        switch(j)
        {
                case 2:
                        hz = 205;
                        vt = 186;
                        ul = 201;
                        ur = 187;
                        lr = 188;
                        ll = 200;
                        lt = 185;
                        lm = 204;
                        subbox(hz,vt,ul,ur,lr,ll,lt,lm,xul,yul,xlr,ylr,0);
                        break;
                case 3:
                        hz = 205;
                        vt = 186;
                        ul = 201;
                        ur = 187;
                        lr = 188;
                        ll = 200;
                        lt = 185;
                        lm = 204;
                        subbox(hz,vt,ul,ur,lr,ll,lt,lm,xul,yul,xlr,ylr,1);
                        break;
                case 0:
                        hz = 205;
```

```
                vt = 179;
                ul = 213;
                ur = 184;
                lr = 190;
                ll = 212;
                lt = 181;
                lm = 198;
                subbox(hz,vt,ul,ur,lr,ll,lt,lm,xul,yul,xlr,ylr,0);
                break;
        case 1:
                hz = 196;
                vt = 179;
                ul =-218;
                ur = 191;
                lr = 217;
                ll = 192;
                lt = 180;
                lm = 195;
                subbox(hz,vt,ul,ur,lr,ll,lt,lm,xul,yul,xlr,ylr,1);
                break;
        case 4:
                hz = 205;
                vt = 186;
                ul = 201;
                ur = 187;
                lr = 188;
                ll = 200;
                lt = 185;
                lm = 204;
                subbox(hz,vt,ul,ur,lr,ll,lt,lm,xul,yul,xlr,ylr,1);
                break;
        }
}
/*-------------------------------------------------------------------*/
subbox(int hz,int vt,int ul,int ur,int lr,int ll,int lt,int lm,int xul,int yul,int xlr,int ylr,int
j)
{
        int i;
        gotoxy(xul,yul);
        for(i=xul;i<=xlr;i++)
        putch(hz);
        if(j==0)
        {
                gotoxy(xul,yul+5);
                for(i=xul;i<=xlr;i++)
                putch(hz);
        }
        gotoxy(xul,ylr);
        for(i=xul;i<=xlr;i++)
```

```
        putch(hz);
        for(i=yul;i<=ylr;i++)
        {
                gotoxy(xul,i);
                putch(vt);
                gotoxy(xlr,i);
                putch(vt);
        }
        gotoxy(xul,yul);putch(ul);
        gotoxy(xlr,yul);putch(ur);
        gotoxy(xlr,ylr);putch(lr);
        gotoxy(xul,ylr);putch(ll);
        if(j==0)
        {
                gotoxy(xul,yul+5);putch(lm);
                gotoxy(xlr,yul+5);putch(lt);
        }
}
/*-------------------------------------------------------------------*/
name()
{
        textcolor(TXT_LOGO);
        textbackground(TXT_BACKG);
        gotoxy(17,3);
        cprintf(" KING MONGKUT'S INSTITUTE OF TECHNOLOGY
LADKRABANG\n ");
        gotoxy(30,4);cprintf("FACULTY OF ENGINEERING\n");

}
/*-------------------------------------------------------------------*/
appendrec()
{
        FILE *fp;
        int n;
        char m_price[6],file_name[10];
        gotoxy(26,1);printf("# APPEND RECORD #");
        gotoxy(22,8);printf("Enter file name : ");
        bigcur();gets(file_name);cur_off();
        if((fp =fopen(file_name,"ab")) ==NULL)
        {
                printf("\007");
                clrscr();
                box(2,1,79,25,0);
                gotoxy(22,8);printf("Error in open file\n");
                getch();
                return;
        }
        do
        {
```

```
                clrscr();
                gotoxy(25,3);printf("NEW RECORD");
                gotoxy(22,7);printf("Number : ");
                gotoxy(22,9);printf("Code   : ");
                gotoxy(22,11);printf("Name   : ");
                gotoxy(22,13);printf("Price  : ");
                gotoxy(31,7);bigcur();gets(var.number);cur_off();
                if(var.number[0]!='\0')
                {
                        gotoxy(31,9);bigcur();gets(var.code);
                        gotoxy(31,11);gets(var.name);
                        gotoxy(31,13);gets(m_price);cur_off();
                        var.price1=atoi(m_price);
                        fwrite(&var,sizeof var,1,fp);
                        if(ferror(fp))
                        {
                                printf("\007");
                                box(2,1,79,25,0);
                                clrscr();
                                gotoxy(22,8);printf("Error in writing\n");
                                window(1,1,80,25);
                                getch();
                                return;
                        }
                }
        }while(var.number[0]!='\0');
        printf("\007");
        fclose(fp);
}
/*-----------------------------------------------------------------------*/
insrec()
{
        FILE *fp;
        char rec[4],m_price[6],file_name[10];
        long loc1;
        int n_rec;
        clrscr();
        box(2,1,79,25,0);
        name();
        gotoxy(24,14);printf("Enter file name : ");
        bigcur();gets(file_name);cur_off();
        ffile(file_name);
        if((fp =fopen(file_name,"r+b")) ==NULL)
        {
                clrscr();
                box(2,1,79,25,0);
                name();
                printf("\007");
                gotoxy(24,14);printf("Error in open file\n");
```

```
        getch();
        return;
}
do
{
        clrscr();
        box(2,1,79,25,0);
        name();
        gotoxy(20,14);
        printf("Enter the NUMBER to be inserted :");
        bigcur();gets(rec);cur_off();
        if((atoi(rec))<=nfile)
        {
        if(rec[0]!='\0')
        {
                clrscr();
                box(2,1,79,25,0);
                name();
                gotoxy(25,9);printf("Enter new data");
                gotoxy(25,11);printf("Number :");
                bigcur();gets(var1.number);
                gotoxy(25,13);printf("Code   : ");
                gets(var1.code);
                gotoxy(25,15);printf("Name   : ");
                gets(var1.name);
                gotoxy(25,17);printf("Price  : ");
                gets(m_price);cur_off();
                var1.price1=atoi(m_price);
                fseek(fp,(long)((atoi(rec)-1)*sizeof var1),0);
                while(fread(&var2,sizeof var2,1,fp)==1)
                {
                        if(ferror(fp))
                        {
                                clrscr();
                                box(2,1,79,25,0);
                                name();
                                gotoxy(24,14);
                                printf("Error in reading\n");
                                fclose(fp);
                                getch();
                                return;
                        }
                        loc1=ftell(fp)-sizeof var2;
                        fseek(fp,loc1,0);
                        fwrite(&var1,sizeof var1,1,fp);
                        if(ferror(fp))
                        {
                                clrscr();
                                box(2,1,79,25,0);
```

```
                                        name();
                                        gotoxy(24,14);
                                        printf("Error in writing\n");
                                        fclose(fp);
                                        getch();
                                        return;
                                }
                                var1=var2;
                                fseek(fp,ftell(fp),0);
                        }
                        fwrite(&var1,sizeof var1,1,fp);
                        rewind(fp);
                }
                }
                if((atoi(rec))>nfile)
                {
                clrscr();
                box(2,1,79,25,0);
                name();
                gotoxy(27,18);printf("DON'T HAVE NUMBER : %d",atoi(rec));
                gotoxy(27,24);printf(" PLESS ANY KEY TO RETURN ");
                getch();
                }
        }while(rec[0]!='\0');
        printf("\007");
        fclose(fp);
}

/*-----------------------------------------------------------------*/

delrec()
{
        FILE *fp;
        char rec[4],m_price[6],file_name[10];
        long n_record;
        int i,n_rec;
        struct data *pt;
        clrscr();
        box(2,1,79,25,0);
        name();
        gotoxy(24,14);printf("Enter file name : ");
        bigcur();gets(file_name);cur_off();
        ffile(file_name);
        do
        {
                if((fp=fopen(file_name,"r+b"))==NULL)
                {
                        clrscr();
                        box(2,1,79,25,0);
```

```
            name();
            printf("\007");
            gotoxy(24,14);printf("Error in open file\n");
            getch();
            return;
    }
    clrscr();
    box(2,1,79,25,0);
    name();
    gotoxy(20,14);
    printf("Enter the NUMBER to be deleted ");
    bigcur();gets(rec);cur_off();
    if(rec[0]!='\0')
    {
    if((atoi(rec))<=nfile)
    {
            fseek(fp,0,2);
            n_record=ftell(fp)/sizeof var;
            if((pt=malloc(n_record*sizeof var))==NULL)
            {
                    clrscr();
                    box(2,1,79,25,0);
                    name();
                    gotoxy(24,14);printf("Out of memory\n");
                    getch();
                    return;
            }
            rewind(fp);
            for(i=1;i<=n_record-1;++i)
            {
                    if(ftell(fp)==(long)(atoi(rec)-1)*sizeof var)
                    fseek(fp,sizeof var,1);
                    fread(pt,sizeof var,1,fp);
                    if(ferror(fp))
                    {
                            clrscr();
                            box(2,1,79,25,0);
                            name();
                            gotoxy(24,14);
                            printf("Error in read file");
                            getch();
                            return;
                    }
                    ++pt;
            }
            if((fp=fopen(file_name,"w+b"))==NULL)
            {
                    clrscr();
                    gotoxy(24,14);
```

```
                        printf("Error in open file\n");
                        getch();
                        return;
                }
                for(i=1;i<=n_record-1;++i)
                --pt;
                for(i=1;i<=n_record-1;++i)
                {
                        fwrite(pt,sizeof var,1,fp);
                        if(ferror(fp))
                        {
                                clrscr();
                                name();
                                gotoxy(24,14);
                                printf("Error in writing\n");
                                getch();
                                return;
                        }
                        ++pt;
                }
                rewind(fp);
                clrscr();
        }
        }
    }while(rec[0]!='\0');
    printf("\007");
    fclose(fp);
}
/*-------------------------------------------------------------------*/
editrec()
{
    FILE *fp;
    int n,line,i;
    char rec[4],ch,m_number[5],m_code[7],m_name[40],m_price[4],file_name
[10];
    clrscr();
    box(2,1,79,25,0);
    name();
    gotoxy(24,14);printf("Enter file name : ");
    bigcur();gets(file_name);cur_off();
    ffile(file_name);
    if((fp =fopen(file_name,"r+b")) ==NULL)
    {
        clrscr();
        box(2,1,79,25,0);
        name();
        gotoxy(24,14);printf("Error in open file");
        printf("\007");
        getch();
```

```
        return;
}
do
{
        clrscr();
        box(2,1,79,25,0);
        name();
        gotoxy(17,14);
        line=17;
        printf("Enter the number of record to be edited :");
        bigcur();gets(rec);cur_off();
        n=atoi(rec);
        ch='0';
        if((rec[0]=='0'||rec[0]=='1'||rec[0]=='2'||rec[0]=='3'
        ||rec[0]=='4'||rec[0]=='5'||rec[0]=='6'||rec[0]=='7'
        ||rec[0]=='8'||rec[0]=='9')&&n<=nfile)
        {
                fseek(fp,(long)((n-1)*sizeof var),0);
                fread(&var,sizeof var,1,fp);
                if(ferror(fp))
                {
                        clrscr();
                        name();
                        gotoxy(24,14);printf("Error in file");
                        getch();
                        return;
                }
                        clrscr();
                        box(2,1,79,25,0);
                        name();
                        gotoxy(7,9);printf("OLD RECORD");
                        gotoxy(4,11);printf("numBer : %s\n",var.number);
                        gotoxy(4,13);printf("Code   : %s\n",var.code);
                        gotoxy(4,15);printf("Name   : %s\n",var.name);
                        gotoxy(4,17);printf("Price  : %d\n",var.price1);
                do
                {
                        gotoxy(3,24);
                        for(i=3;i<=74;i++)
                        putch(205);
                        gotoxy(20,20);
                        printf("Press the uppercase charracter for editing ");
                        bigcur();ch=getch();cur_off();
                        gotoxy(20,20);
                        printf("                                          ");
                        ch=toupper(ch);
                        if(ch=='B'||ch=='C'||ch=='N'||ch=='P')
                        {
                        gotoxy(50,9);printf("Enter new data");
```

```
                        switch(ch)
                        {
                                case 'B':
                                        gotoxy(45,11);printf("Number : ");
                                        gotoxy(53,11);
                                        bigcur();gets(m_number);cur_off();
                                        if(m_number[0]!='\0')
                                        strcpy(var.number,m_number);
                                        break;
                                case 'C':
                                        gotoxy(45,13);printf("Code : ");
                                        bigcur();gets(m_code);cur_off();
                                        if(m_code[0]!='\0')
                                        strcpy(var.code,m_code);
                                        break;
                                case 'N':
                                        gotoxy(45,15);printf("Name : ");
                                        bigcur();gets(m_name);cur_off();
                                        if(m_name[0]!='\0')
                                        strcpy(var.name,m_name);
                                        break;
                                case 'P':
                                        gotoxy(45,17);printf("Price : ");
                                        bigcur();gets(m_price);cur_off();
                                        if(m_price[0]!='\0')
                                        var.price1=atoi(m_price);
                                        break;
                        }
                        line=line+2;
                        }
                        gotoxy(5,24);
                        printf(" Q=Quit , SPACBAR = Edit other fields , Enter
= Edit other reccords ");
                        ch=getch();
                }while(toupper(ch)==' ');
                fseek(fp,(long)((n-1)*sizeof var),0);
                fwrite(&var,sizeof var,1,fp);
                if(ferror(fp))
                {
                        clrscr();
                        name();
                        gotoxy(24,14);printf("Error in writing");
                        fclose(fp);
                        printf("\007");
                        getch();
                        return;
                }
        }
        rewind(fp);
```

```
        }while(toupper(ch) == '\r');
        printf("\007");
        fclose(fp);
}
/*-------------------------------------------------------------------*/
ffile(char file_name[10])
{
        FILE *fp;
        nfile=0;
        if((fp =fopen(file_name,"rb")) ==NULL)
        {
                return;
        }
        while(fread(&var,sizeof var,1,fp) ==1)
        {
                if(ferror(fp))
                {
                        return;
                }
                nfile=++nfile;
        }
        fclose(fp);
}
/*-------------------------------------------------------------------*/

void init_8250(void)
{
        outportb(LCR,0x80);   /* set DLAB in Line control register to 1 */
        outportb(DLL,0x0C);   /* set baud rate to 9600 divisor = 000Ch */
        outportb(DLM,0x00);   /* set Divisor latch upper to 00 */
        outportb(MDC,0x00);   /* set Modem control bit 4 (loopback) to 0 */
        outportb(LCR,0x03);   /* set 8250 to 9600 8 N 1 */
}
void send_one_char(BYTE data)
{
        while(!(inportb(LST) & 0x20));
        outportb(TXbuffer,data);
}


/*************************** receive_one_char
*****************************/
/*      return received from RXbuffer                                   */
/*      if not received data it return 0                        */
/*      return type : BYTE                                      */
/*****************************************************************
*****/
BYTE receive_one_char(void)
{
```

```
        BYTE data;
        if (inportb(LST) & 0x01) data = inportb(RXbuffer);
        else data = 0;
        return data;
}

/****************** read_kb ******************************/
/*      get data from keybord one "byte"                 */
/*      get interger from keybord and change to BYTE     */
/*      return type : BYTE                               */
/*********************************************************/
BYTE read_kb(void)
{
        return ((BYTE)getch());
}

/******************** box ********************************/
/*       draw box at (left-1,up-1) and (right+1,low+1)  */
/*       if will set window use (left,up)and(gight,low) */
/*       is edge of window                              */
/*********************************************************/
void clearbox(int left,int up,int right,int low)
{
        int i,j;
        for (i = left;i <= right;i++)
        {
                for (j = up; j <= low;j++)
                {
                        gotoxy(i,j);
                        putch(' ');
                }
        }
}
void blackbox(int left ,int up,int right,int low)
{
        int i,j;
        for (i = left;i <= right;i++)
        {
                for (j = up; j <= low;j++)
                {
                        gotoxy(i,j);
                        putch(' ');
                }
        }
}

void wall(void)
{
   int i,j;
```

```
   textattr(0x79);
   for(i = 1;i <=80;i++)
        for (j = 1;j <=24;j++)
           putch('ฑ'); /* ascii 178 */
   textattr(07);
}

void screen(void)
{
        textattr(0x1f);
        clrscr();
        wall();
        blackbox(27,3,55,5);
        blackbox(7,8,36,18);
        blackbox(47,8,76,18);
        blackbox(7,21,76,24);
        textattr(0x1f);
        blackbox(25,2,53,4);
        blackbox(5,7,34,17);
        blackbox(45,7,74,17);
        blackbox(5,20,74,23);
        gotoxy(30,3);
        cprintf(" ฑ AUTOMATIC MENU ฑ ");
        textattr(0x1e);
        gotoxy(7,9);
        cprintf(" ฑ INVENTOR ฑ ");
        gotoxy(9,11);
        cprintf("1.UTHAI SKULUDOMKANG");
        gotoxy(9,13);
        cprintf("2.AEKKAPOT BOONYARATTAPUN");
        gotoxy(9,15);
        cprintf("3.ORAN SRISAWET");
        gotoxy(47,9);
        cprintf(" ฑ ADVISOR ฑ ");
        gotoxy(50,11);
        cprintf("Dr. MANUT SUNGWORASILL");
        gotoxy(47,13);
        cprintf(" ฑ CLASS ฑ ");
        gotoxy(50,15);
        cprintf("3R/1  ROOM ");
        gotoxy(31,21);
        cprintf(" ELECTRONIC ENGINEER ");
        gotoxy(15,22);
        cprintf("KING'S MONGKUT INSTITUTE OF TECHNOLOGY
LADKRABANG");

}
void bye(void)
{
```

```
        int i;
        textattr(0x17);
        blackbox(25,4,55,21);
        window(25,4,55,21);
        for(i = 1;i <=52;i++) cprintf("");
        window(1,1,80,25);
}
/*AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA*/

decode_data(void)
{
        int i,j;
        BYTE data_in_serial;
        WORD TX_P,RX_P;
        int up_xpos=2,up_ypos=2,low_xpos=2,low_ypos=15;
        int data;

        i=0;
        do
        {
                if(kbhit()!=0)
                {
                        data_kb=getch();
                }


                data_in_serial = receive_one_char();
                if (data_in_serial!=0 && data_in_serial!='*')
                {
                        data_port[i]=data_in_serial;
                        if(data_port[i]=='#')
                        {
                                data_kb='Q';
                                i--;
                        }
                        i++;
                }

        }while(data_kb!='Q' && data_kb!=ESC1);
        list_menu=i/4;
        j=0;
        table[0]=data_port[j];
        j++;
        table[1]=data_port[j];
        table[2]='\0';
        j++;
        for(list=0;list<list_menu;list++)
        {
```

```
            code_food[list][0]=data_port[j];
            j++;
            code_food[list][1]=data_port[j];
            j++;
            code_food[list][2]=data_port[j];
            code_food[list][3]='\0';
            j++;
            amoung[list][0]=data_port[j];
            amoung[list][1]='\0';
            j++;
      }

}

/*AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA*/
```

ภาคผนวก ค

(DATA SHEET)

National Semiconductor

MM54C922/MM74C922 16-Key Encoder
MM54C923/MM74C923 20-Key Encoder

general description

These CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two key roll over is provided between any two switches.

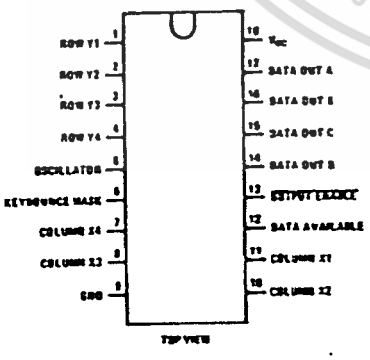
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- TRI-STATE outputs LPTTL compatible
- Wide supply range 3V to 15V
- Low power consumption

connection diagrams

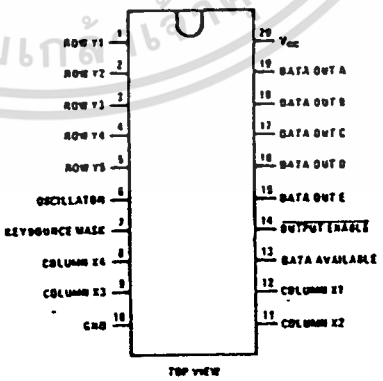
Dual-In-Line Package



TOP VIEW

Order Number MM54C922N
or MM74C922N
See Package 20

Dual-In-Line Package

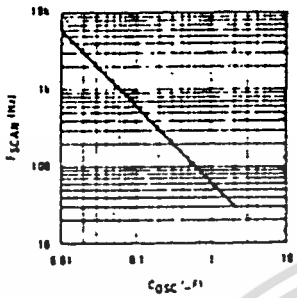


TOP VIEW

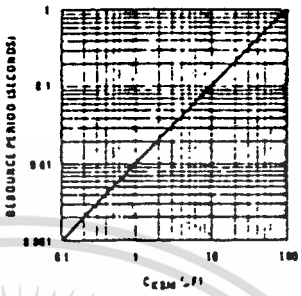
Order Number MM54C923N
or MM74C923N
See Package 20A

typical performance characteristics (con't)

Typical F_{SCAN} vs C_{OSC}

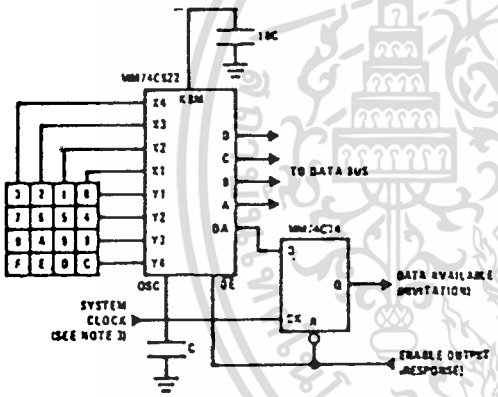


Typical Debounce Period vs C_{KBM}

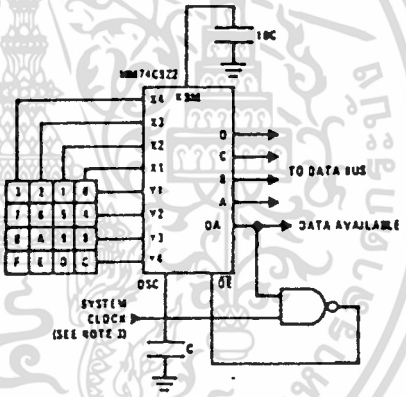


typical applications

Synchronous Handshake (MM74C922)

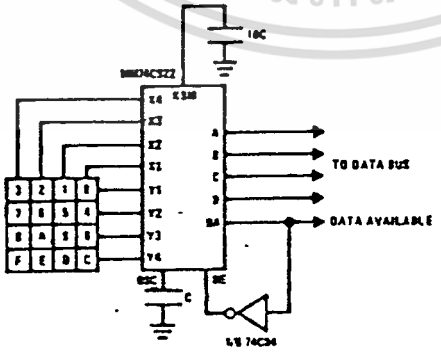


Synchronous Data Entry Onto Bus (MM74C922)



Outputs are enabled when valid entry is made and go into TRI-STATE when key is released.

Asynchronous Data Entry Onto Bus (MM74C922)



Outputs are in TRI STATE until key is pressed, then data is placed on bus. When key is released, outputs return to TRI-STATE.

Keyboard Suppliers

Mini Key Series KL
Digitran Company
Pasadena, California

Computronics Engineering
7235 Hollywood Blvd
Hollywood, California 90046

Note 3: The keyboard may be synchronously scanned by omitting the capacitor at osc. and driving osc. directly if the system clock rate is lower than 10 kHz.

absolute maximum ratings

Voltage at Any Pin	V_{CC} − 0.3V to V_{CC} + 0.3V	Package Dissipation	500 mW
Operating Temperature Range		Operating V_{CC} Range	3V to 15V
MM54C922, MM54C923	−55°C to +125°C	V_{CC}	18V
MM74C922, MM74C923	−40°C to +85°C	Lead Temperature (Soldering, 10 seconds)	300°C
Storage Temperature Range	−65°C to +150°C		

dc electrical characteristics

Min/max limits apply across temperature range unless otherwise noted

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
CMOS TO CMOS						
V_{T+}	Positive Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq$ 0.7 mA	3	3.6	4.3	V
		V_{CC} = 10V, $I_{IN} \geq$ 1.4 mA	6	6.8	8.6	V
		V_{CC} = 15V, $I_{IN} \geq$ 2.1 mA	9	10	12.9	V
V_{T-}	Negative Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq$ 0.7 mA	0.7	1.4	2	V
		V_{CC} = 10V, $I_{IN} \geq$ 1.4 mA	1.4	3.2	4	V
		V_{CC} = 15V, $I_{IN} \geq$ 2.1 mA	2.1	5	6	V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,	3.5	4.5		V
		V_{CC} = 10V	8	9		V
		V_{CC} = 15V,	12.5	13.5		V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V,		0.5	1.5	V
		V_{CC} = 10V,		1	2	V
		V_{CC} = 15V,		1.5	2.5	V
I_{rp}	Row Pull Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	μA
		V_{CC} = 10V		−10	−20	μA
		V_{CC} = 15V		−22	−45	μA
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 5V, I_O = −10μA	4.5			V
		V_{CC} = 10V, I_O = −10μA	9			V
		V_{CC} = 15V, I_O = −10μA	13.5			V
$V_{OUT(0)}$	Logical "0" Output Voltage	V_{CC} = 5V, I_O = 10μA			0.5	V
		V_{CC} = 10V, I_O = 10μA			1	V
		V_{CC} = 15V, I_O = 10μA			1.5	V
R_{on}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	V_{CC} = 5V, V_O = 0.5V		500	1400	Ω
		V_{CC} = 10V, V_O = 1V		300	700	Ω
		V_{CC} = 15V, V_O = 1.5V		200	500	Ω
I_{CC}	Supply Current	V_{CC} = 5V, Osc at 0V		0.55	1.1	mA
		V_{CC} = 10V		1.1	1.9	mA
		V_{CC} = 15V		1.7	2.6	mA
$I_{IN(1)}$	Logical "1" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V	V_{CC} − 1.5			V
		74C, V_{CC} = 4.75V	V_{CC} − 1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V			0.8	V
		74C, V_{CC} = 4.75V			0.8	V
$V_{OUT(1)}$	Logical "1" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA	2.4			V
		74C, V_{CC} = 4.75V, I_O = −360μA	2.4			V
$V_{OUT(0)}$	Logical "0" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA			0.4	V
		74C, V_{CC} = 4.75V, I_O = −360μA			0.4	V

dc electrical characteristics (con't)

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
OUTPUT DRIVE (See 54C/74C Family Characteristics Data Sheet)						
I_{SOURCE}	Output Source Current (P Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

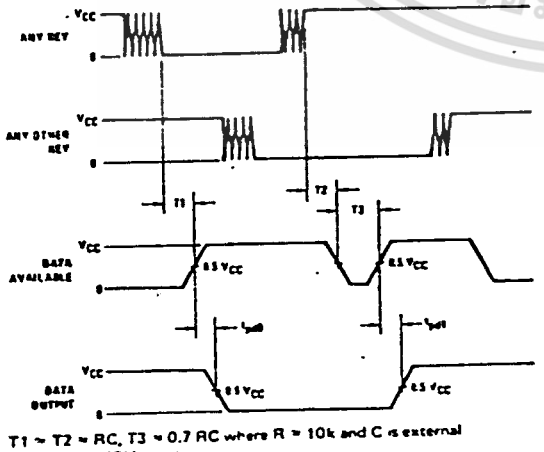
ac electrical characteristics T_A = 25°C

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
t_{pd0}-t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF, *(Figure 1)*				
		V_{CC} = 5V		60	150	ns
		V_{CC} = 10V		35	80	ns
		V_{CC} = 15V		25	60	ns
t_{0H}-t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 5 pF, *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		80	200	ns
		V_{CC} = 10V C_L = 10 pF		65	150	ns
		V_{CC} = 15V		50	110	ns
t_{H0}-t_{H1}	Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF, *(Figure 2)*				
		V_{CC} = 5V R_L = 10k		100	250	ns
		V_{CC} = 10V C_L = 50 pF		55	125	ns
		V_{CC} = 15V		40	90	ns
C_{IN}	Input Capacitance	Any Input, (Note 2)		5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance	Any Output, (Note 2)		10		pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

switching time waveforms



T1 ≈ T2 ≈ RC, T3 ≈ 0.7 RC where R = 10k and C is external capacitor at KBM input.

FIGURE 1

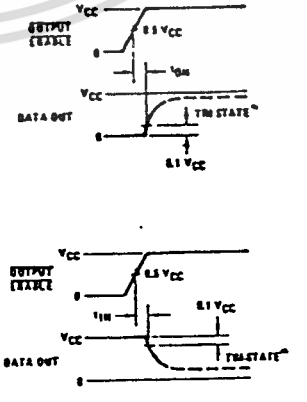
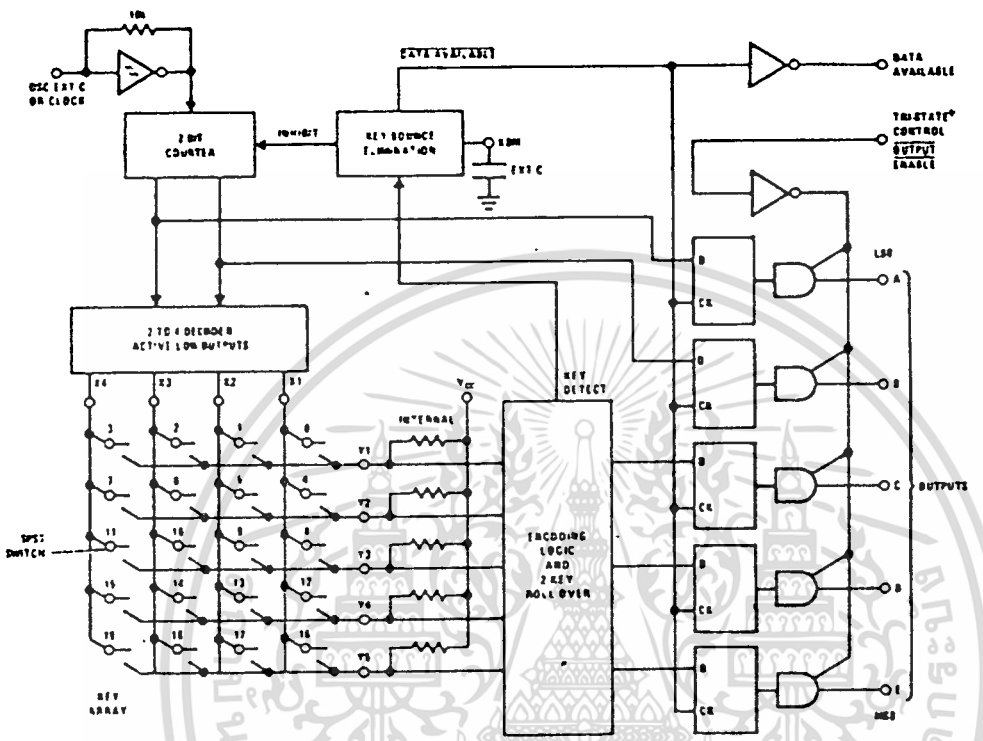


FIGURE 2

block diagram



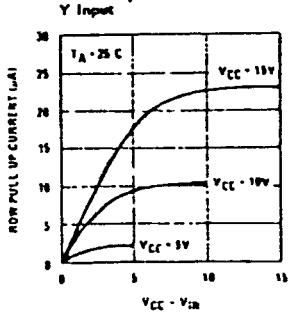
truth table

SWITCH POSITION	0 Y1,X1	1 Y1,X2	2 Y1,X3	3 Y1,X4	4 Y2,X1	5 Y2,X2	6 Y2,X3	7 Y2,X4	8 Y3,X1	9 Y3,X2	10 Y3,X3	11 Y3,X4	12 Y4,X1	13 Y4,X2	14 Y4,X3	15 Y4,X4	16 Y5*,X1	17 Y5*,X2	18 Y5*,X3	19 Y5*,X4
OUTPUT A	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
OUTPUT B	0	0	*	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
OUTPUT C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
OUTPUT D	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
OUTPUT E*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

*Omit for MM54C922/MM74C922

typical performance characteristics

Typical I_{rp} vs V_{IN} at Any Y Input



Typical R_{on} vs V_{OUT} at Any X Output