ระบบจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์

นาย สยาม ดีอุดมวงศา

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาฟิสิกส์ประยุกต์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2534/

DRUG ARRANGING MACHINE CONTROLLED BY MICROCOMPUTER

Mr. SIAM DEEUDOMVONGSA

A SPECIAL PROJECT SUBMITED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF APPLIED PHYSICS

FACULTY OF SCIENCES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1991

หัวข้อโครงงานพิเศษ	ระบบจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์
โดย	นาย สยาม ดีอุดมวงศา
ภาควิชา	ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา	อ. อนุพงศ์ สรงประภา
	อ. จิติ หนูแก้ว

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

_____ หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ดร. เสน่ห์ เอกะวิภาต)

คณะกรรมการโครงงานพิเศษ

_____ ประธานกรรมการ

(อ. สุวรรณ คูสำราญ)

_____ กรรมการ

(อ. อนุพงศ์ สรงประภา)

_____ กรรมการ

(อ. จิติ หนูแก้ว)

_____ กรรมการ

(อ. อนุชิต จารุวนาวัฒน์)

ลิขสิทธิ์ของภาควิชาฟิสิกส์ประยุกต์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อโครงงานพิเศษ	ระบบจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์
นักศึกษา	นาย สยาม ดีอุดมวงศา
อาจารย์ที่ปรึกษา	อ. อนุพงศ์ สรงประภา
	อ. จิติ หนูแก้ว
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	2534

บทคัดย่อ

ในโครงงานพิเศษนี้ได้สร้าง ระบบจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC ซึ่งในการจัดเรียงข้อมูลของยาได้ใช้โปรแกรมภาษา C โดยระบบสามารถเลือกขวดยาได้ 64 ชนิด และอัตราการจ่ายเม็ดยาโดยเฉลี่ยคือ 100 เม็ดต่อ 13.4 นาที

Special Project Title	Drug Arranging Machine Controlled by Microcomputer
Name	Mr. Siam Deeudomvongsa
Special Project Advisor	Mr. Anupong Srongprapa
	Mr. Jitti Nukaew
Department	Applied Physics
Academic year	1991

Abstract

In this special project we have constructed a drug arranging machine controlled by microcomputer IBM PC. C.programming language is used in data base management. The system can select up to 64 drug bottles and the average speed of releasing is 100 tabets per 13.4 minutes

กิตติกรรมประกาศ

โครงงานพิเศษนี้สามารถเสร็จสมบูรณ์ได้ ด้วยความช่วยเหลือจากบุคคลต่างๆ

คุณพ่อ และ คุณแม่ ซึ่งคอยให้กำลังใจ และให้ความอุปการะให้ได้รับการศึกษาจนถึงระดับอุดมศึกษา

อ. อนุพงศ์ สรงประภา ซึ่งคอยให้คำปรึกษาในรายละเอียดต่างๆ ของโครงงานและความช่วยเหลือทางด้านข้อมูลต่างๆ

อ. จิติ หนูแก้ว ซึ่งคอยให้คำปรึกษา ด้านการเชื่อมต่อไมโครคอมพิเตอร์

อ. อนุชิต จารุวนาวัฒน์ ซึ่งคอยให้คำปรึกษา ด้านโปรแกรมภาษา C

อ. วิชิต ศิริชติ ซึ่งคอยให้คำปรึกษา

อ. เกียรติศักดิ์ คมวัชระ ซึ่งคอยให้คำปรึกษา

คณะกรรมการทุกท่าน ที่ช่วยตรวจทานและแก้ไขรายงานโครงงานพิเศษนี้

และสุดท้าย ความช่วยเหลือที่ได้รับจากภาควิชา และเพื่อนๆ พี่ๆ น้องๆ ทุกท่าน

สยาม ดีอุดมวงศา

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1

บทนำ

ในปัจจุบันเครื่องคอมพิวเตอร์ได้เข้ามามีบทบาทในการทำงานต่าง ๆ มากมาย ไม่ว่าจะเป็นงานทางด้านธุรกิจ,วิศวกรรม,วิทยาศาสตร์ และการแพทย์เป็นต้น เราสามารถที่จะนำคอมพิวเตอร์มาทำการจัดเก็บข้อมูล ค้นหาข้อมูล และติดต่อกับอุปกรณ์ภายนอกได้ จากคุณสมบัติของคอมพิวเตอร์ที่กล่าวมา เราจึงนำเครื่องคอมพิวเตอร์มาประยุกต์เข้ากับระบบการจ่ายยาเม็ดในแผนกจ่ายยาตามโรงพยาบาล ซึ่งระบบนี้เรียกว่าระบบการจ่ายยาอัตโนมัติควบคุมด้วยไมโคร-คอมพิวเตอร์

ลักษณะทั่วไปของระบบการจ่ายยาตามโรงพยาบาลจะมีขั้นตอนดังนี้คือ เมื่อแพทย์ทำการตรวจและเขียนใบสั่งยา ใบสั่งยานี้จะถูกนำมาในแผนกชำระเงิน เมื่อชำระเงินเสร็จเรียบร้อยแล้วใบสั่งยาจะนี้ถูกนำมาในห้องจ่ายยาซึ่งจะมีเภสัชกรและผู้ช่วยประจำอยู่ โดยจะคอยจัดยาตามรายการในใบสั่งยานั้น หลังจากนี้เภสัชกรจะทำการตรวจยาทั้งหมดว่าถูกต้องตามรายการใบสั่งยาและทำการจ่ายยาให้แก่คนไข้ พร้อมทั้งบอกวิธีการใช้ยานั้นๆด้วย

ด้วยขั้นตอนของระบบการจ่ายยาดังกล่าวมานี้ เภสัชกรและผู้ช่วยเภสัชกรจะต้องรู้ถึงประ-เภทของยาทุกชนิด รวมทั้งตำแหน่งที่ตั้งของยาทุกชนิดแล้ว เภสัชกรและผู้ช่วยเภสัชกรจะต้องทำการนับจำนวนเม็ดของยาแต่ละชนิดตามใบสั่งยา ซึ่งโดยทั่วไปแล้วตามแผนกห้องยาจะมีเภสัชกรและผู้ช่วยเภสัชกรประจำอยู่ประมาณ 10-15 คน คอยทำหน้าที่จ่ายยานี้ ซึ่งถ้าจะพิจารณาถึงการทำงานของระบบการจ่ายยาทั้งหมดแล้ว ในแต่ละใบสั่งยาจะต้องใช้คนทำงานมากพอสมควร รวมถึงต้องใช้เวลาในการทำงานนานพอสมควรด้วยโดยจากการไปสำรวจการทำงานของแผนกห้องจ่ายยาตามโรงพยาบาลระดับจังหวัดของรัฐพบว่าโดยเฉลี่ยของการจ่ายยาในแต่ละใบสั่งยาจะใช้เวลาประ-มาณ 20-25 นาที ซึ่งใช้เวลานานมากเมื่อคิดถึงคนไข้ที่มีจำนวนมากในแต่ละวัน

ดังนั้นด้วยขั้นตอนของระบบดังกล่าว เราจึงคิดวิธีที่จะลดจำนวนคนที่ทำงานในห้องจ่ายยา รวมถึงการลดเวลาการจ่ายยา โดยใช้ไมโครคอมพิวเตอร์ในการจัดเก็บข้อมูลของยาแต่ละชนิดอย่างมีระเบียบ (sorting) และทำการค้นหาข้อมูลของยาแต่ละชนิดตามใบสั่งยา (searching) อย่างมีระบบ และทำการเชื่อมต่อเครื่องจ่ายยากับไมโครคอมพิวเตอร์ โดยภายในเครื่องจ่ายยาจะประกอบไปด้วยระบบการปิดเปิดขวดยา และระบบการนับเม็ดยา ซึ่งระบบทั้งหมดรวมเรียกว่า ระบบการจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์

โดยวัตถุประสงค์ของโครงงานนี้ เพื่อนำเครื่องไมโครคอมพิวเตอร์มาใช้ในการควบคุมการจ่ายยาอย่างอัตโนมัติ และเพื่อศึกษาถึงการเก็บข้อมูลอย่างเป็นระเบียบ (sorting) การค้นหาข้อมูล (searching) และเพื่อศึกษาถึงการเชื่อมต่ออุปกรณ์ภายนอกกับเครื่องไมโครคอมพิวเตอร์ โดยอุปกรณ์ภายนอกนี้ได้แก่ ระบบการนับ ระบบการปิดเปิด ระบบการตรวจจับ และลักษณะกลไกของระบบ

โดยมีวิธีการดำเนินงานดังนี้

1. ศึกษาระบบการจัดฐานข้อมูล การจัดเก็บข้อมูล (sorting) การค้นหาข้อมูล (searching)
2. ศึกษาถึงทฤษฎีการเชื่อมต่ออุปกรณ์ภายนอกกับเครื่องไมโครคอมพิวเตอร์
3. ออแบบระบบการนับ ระการปิดเปิด ระบบการตรวจจับ และกลไกของระบบ
4. เขียนโปรแกรมการจัดเก็บข้อมูล การค้นหาข้อมูล และการควบคุมระบบของอุปกรณ์ภายนอก
5. ทำการทดลองระบบทั้งหมดของการจ่ายยา และสรุปผล เสอนแนะ

บทที่ 2

การจัดระบบฐานข้อมูลของระบบการจ่ายยา

ในระบบการจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์นั้น การจัดระบบฐานข้อมูลเป็นสิ่งสำคัญที่ต้องมีอยู่ในระบบการจ่ายยาฯ เพื่อจะได้จัดเก็บข้อมูลอย่างเป็นระเบียบ และสะดวกต่อการค้นหาข้อมูล ดังนั้นจึงจำเป็นจะต้องศึกษาถึงการจัดระบบฐานข้อมูล

2.1 ฐานข้อมูล(data base)

ฐานข้อมูล(data base) คือที่รวมของข้อมูลที่เราเก็บรวบรวมเอาไว้ ในการเก็บข้อมูลด้วยคอมพิวเตอร์นั้น ฐานข้อมูลจะต้องประกอบด้วยแฟ้มข้อมูล(data file)จำนวนหนึ่ง แต่ละแฟ้มข้อมูลหรือไฟล์จะต้องมีชื่อไฟล์(file name) กำกับอยู่เพื่อสะดวกต่อการเรียกใช้ ข้อมูลที่เก็บไว้ในไฟล์แต่ละบรรทัดเรียกว่าเรคอร์ด(record) โดยข้อมูลแต่ละเรคอร์ดจะเก็บรายละเอียดในรูปแบบลักษณะที่เหมือนกัน โดยในแต่ละคอลัมน์ของเรคอร์ดจะรวบรวมข้อมูลประเภทเดียวกันไว้ซึ่งเรียกว่าดาต้าฟิลด์(data field) หรือฟิลด์ โดยแต่ละฟิลด์จะมีชื่อกำกับเรียกว่าชื่อฟิลด์(field name) ซึ่งสิ่งที่บรรจุอยู่ในแต่ละฟิลด์คือข้อมูล(data) ดังนั้นลักษณะการจัดแบ่งฟิลด์ต่างๆของไฟล์เราเรียกว่า โครงสร้างของไฟล์(file structure)

2.2 การจัดการฐานข้อมูล(data base management)

การจัดการฐานข้อมูลเป็นเรื่องที่เกี่ยวข้องกับการจัดการข้อมูลที่มีอยู่ โดยมีหัวใจสำคัญคือการจัดเก็บข้อมูลให้ง่ายที่สุด และสามารถเรียกใช้ข้อมูลนั้นๆได้อย่างสะดวก

โดยการทำงานพื้นฐานของการจัดการฐานข้อมูลประกอบด้วย

1. การสร้างไฟล์ข้อมูล
2. การเพิ่มเติมข้อมูลลงในไฟล์
3. การจัดเรียงข้อมูลในไฟล์
4. การค้นหาข้อมูลจากไฟล์

5. การจัดทำรายงานจากไฟล์ที่มีอยู่

6. การแก้ไขปรับปรุงข้อมูลในไฟล์

7. การลบข้อมูลออกจากไฟล์

เมื่อนำเอาการจัดการฐานข้อมูลมาใช้ในระบบจ่ายยาฯ สิ่งที่ต้องศึกษาถึงรายละเอียดของการจัดการฐานข้อมูลก็คือ วิธีการจัดเรียงข้อมูล(sorting) และวิธีการค้นหาข้อมูล(searching)

2.3 การจัดเรียงลำดับ(sorting)

การจัดเรียงลำดับในที่นี้หมายถึง การจัดเรียงข้อมูลให้เรียงลำดับตามคีย์ โดยจะเรียงจากค่าน้อยไปมาก เช่น รายชื่อผู้มีโทรศัพท์ในสมุดโทรศัพท์จะมีการเรียงลำดับตามตัวอักษร หรือ รายชื่อของยาชนิดต่างๆในระบบการจ่ายยาฯ เป็นต้น การจัดเรียงลำดับนี้แม้ว่าจะต้องใช้เวลาในการจัดเรียง แต่ผลจะช่วยทำให้การค้นหาข้อมูลในภายหลังทำได้สะดวกและรวดเร็วขึ้น ดังนั้นการจัดเรียงลำดับข้อมูลจึงเป็นงานสำคัญอีกอย่างหนึ่งในระบบงานที่ใช้ไมโครคอมพิวเตอร์ควบคุม

การจัดเรียงลำดับแบ่งออกเป็นหลายวิธีด้วยกัน กล่าวคือ

2.3.1 การจัดเรียงลำดับแบบเลือก (selection sort)

2.3.2 การจัดเรียงลำดับแบบแทรก (insertion sort)

2.3.3 การจัดเรียงลำดับแบบบับเบิล (bubble sort)

2.3.4 การจัดเรียงลำดับแบบเชลล์ (shell sort)

2.3.5 การจัดเรียงลำดับแบบเร็ว (quick sort)

โดยมีวิธีการจัดเรียงลำดับ ซึ่งจะกำหนดให้ข้อมูลมีคีย์เป็นจำนวนเต็ม ที่ถูกเก็บไว้ในแถวลำดับ A[1..n] และจะจัดเรียงให้ลำดับคีย์จากน้อยไปมาก นั่นคือ A[1]< A[2]<...<A[n] การจัดเรียงลำดับจะกล่าวรายละเอียดดังต่อไปนี้

2.3.1 การจัดเรียงลำดับแบบเลือก (selection sort)

การเรียงลำดับแบบเลือก วิธีนี้เป็นวิธีการจัดเรียงที่ง่ายที่สุด โดยเริ่มจากตำแหน่งของคีย์ที่มีค่าน้อยที่สุดสลับของตำแหน่งนั้นกับคีย์ที่อยู่ใน A[1] จะได้ A[1] มีค่าน้อยที่สุด ต่อไปหาตำแหน่ง

ของคีย์ที่มีค่าน้อยที่สุดในกลุ่ม A[2],A[3],...,A[n] แล้วสลับคีย์ของตำแหน่งนั้นกับคีย์ใน A[2] และให้ดำเนินการในลักษณะนี้ จนกระทั่งตำแหน่งคีย์ที่มีค่าท้ายสลับกับคีย์ในตำแหน่ง A[n-1] เช่น กำหนดข้อมูลเริ่มต้นดังนี้ [33,44,(11),77,55]

จะพบว่าคีย์ 11 น้อยสุด ดังนั้นจึงสลับ A[1] กับ A[3] ซึ่งจะได้ 11 [44,33,77,55]

เมื่อดำเนินการในลักษณะนี้ต่อไป จะได้ผลของการจัดเรียงสุดท้ายเป็น 11,33,44,55,77

2.3.2 การจัดเรียงลำดับแบบแทรก (insertion sort)

การจัดเรียงแบบนี้ได้ใช้เทคนิคมาจากลักษณะการจัดไพ่ในมือของผู้เล่น กล่าวคือ เมื่อผู้เล่นได้ไพ่ใบใหม่เพิ่มขึ้นมา เขาจะนำไพ่ใบนั้นไปแทรกในตำแหน่งที่เหมาะสม ซึ่งจะทำให้ไพ่ในมือบางส่วนขยับตำแหน่งออกไป

สำหรับการจัดลำดับแบบแทรกนั้นจะพิจารณา คีย์ในตำแหน่งที่ i(i=1,2,3,...,n) ถ้าคีย์ของ A[i] มีค่าน้อยกว่าที่อยู่ในตำแหน่งก่อนหน้า i แล้วจะนำคีย์ A[i]ไปแทรกในตำแหน่งที่ถูกต้อง ซึ่งจะมีผลทำให้คีย์ในตำแหน่งที่อยู่หลังตำแหน่งที่แทรกขยับออกไป

2.3.3 การจัดเรียงแบบบับเบิล (bubble sort)

การจัดเรียงโดยวิธีนี้จะเป็นการเปรียบเทียบคีย์ในตำแหน่งที่ติดกัน ถ้าคีย์ไม่อยู่ในลำดับที่ต้องการ ให้สลับคีย์ระหว่างตำแหน่งที่เปรียบเทียบ การเปรียบเทียบจะเริ่มจากคู่แรกคือ A[1]กับA[2] ต่อไปจะเป็น A[2]กับA[3] ซึ่งจะเปรียบเทียบในลักษณะนี้จนถึงตำแหน่งสุดท้าย และจากนั้นจะกลับไปเริ่มต้นการเปรียบเทียบแบบเดียวกันอีกจนกระทั่งจัดเรียงเรียบร้อยทุกตำแหน่ง

2.3.4 การจัดเรียงลำดับแบบเชลล์ (shell sort)

ผู้ที่คิดเทคนิคการจัดเรียงลำดับแบบนี้คือ D.L.SHELL โดยมีแนวความคิดในการจัดเรียงคือ ในแต่ละรอบของการพิจารณา จะแบ่งข้อมูลออกเป็นกลุ่ม แต่ละกลุ่มจะประกอบไปด้วยคีย์ที่อยู่ในตำแหน่งห่างกันเป็นระยะ d จากนั้นในแต่ละกลุ่มจะทำการจัดเรียงลำดับคีย์โดยจะวิธีการจัดเรียงแบบแทรก หรือแบบบับเบิล เมื่อจัดเรียงลำดับทุกกลุ่มเรียบร้อยแล้วจะลดระยะ d ลงไป และดำเนินการในแบบเดียวกันต่อไป ซึ่งจะกระทำในลักษณะนี้จนกว่าระยะ d เป็น 1

2.3.5 การจัดเรียงลำดับแบบเร็ว (quick sort)

การจัดเรียงลำดับแบบเร็ว จัดว่าเป็นวิธีการจัดเรียงลำดับที่ใช้เวลาน้อยเหมาะสำหรับการจัดเรียงลำดับที่มีจำนวนมาก โดยมีแนวความคิดในการจัดเรียงลำดับ คือ จะเลือกคีย์จากกลุ่มข้อมูลขึ้นมา 1 ค่า และใช้คีย์นั้นเป็นหลักในการแบบข้อมูลออกเป็นสองส่วน ซึ่งผลของการแบ่งจะได้ส่วนหนึ่งอยู่ในตำแหน่งตอนหน้า และคีย์ที่อยู่ในส่วนนี้จะมีค่าน้อยกว่าหรือเท่ากับคีย์ที่เลือก อีกส่วนหนึ่งจะอยู่ในตำแหน่งตอนหลังและคีย์ในส่วนนี้จะมีค่ามากกว่าหรือเท่ากับคีย์ที่เลือก เมื่อแบ่งเสร็จแล้วจะนำแต่ละส่วนไปแบ่งเป็นส่วนย่อยในลักษณะเดียวกันต่อไป จนกระทั่งทุกส่วนไม่สามารถแบ่งออกเป็นส่วนย่อยต่อไปอีก

2.4 การค้นหาข้อมูล (searching)

วิธีการค้นหาข้อมูลในระบบคอมพิวเตอร์ เราต้องให้ข้อความบางอย่าง หรือค่าของข้อมูลที่จะค้นหาแก่คอมพิวเตอร์ โดยจะเรียกค่านั้นว่าคีย์ และกลุ่มข้อมูลที่จะเข้าไปค้นหาจะมีเขตข้อมูลหนึ่งเป็นคีย์ด้วยเหมือนกัน โดยปกติคีย์ของกลุ่มข้อมูลในกลุ่มเดียวกันมักจะกำหนดให้ไม่ซ้ำกัน เมื่อคอมพิวเตอร์รับค่าคีย์ที่จะค้นหาแล้ว จะนำค่าคีย์นั้นไปเปรียบเทียบกับคีย์ของกลุ่มข้อมูลที่ต้องการค้นหา โดยลำดับการเปรียบเทียบขึ้นอยู่กับอัลกอริทึมที่ใช้

สำหรับวิธีการค้นหาแบ่งออกได้เป็นหลายวิธี ดังนี้คือ

2.4.1 การค้นหาแบบเรียงตามลำดับ (sequential search)

2.4.2 การค้นหาแบบไบนารี่ (binary search)

2.4.3 การค้นหาข้อมูลในไบนารี่เสิร์ชทรี

2.4.1 การค้นหาแบบเรียงตามลำดับ (sequential search)

สำหรับการค้นหาข้อมูลแบบเรียงตามลำดับนี้เป็นวิธีที่ง่ายที่สุด โดยจะนำค่าคีย์ที่จะค้นหาไปเปรียบเทียบกับคีย์ของข้อมูล ตั้งแต่ตัวแรกเรียงลำดับไปทีละตัวจนกว่าจะพบข้อมูลที่ต้องการ หรือเปรียบเทียบไปจนถึงตัวสุดท้าย ดังนั้นจะเรียกวิธีนี้ได้อีกแบบหนึ่งว่าการค้นหาแบบเชิงเส้น

2.4.2 การค้นหาแบบไบนารี่ (binary search)

การค้นหาแบบไบนารี่เป็นวิธีการค้นหาข้อมูลที่รวดเร็วกว่าแบบเรียงตามลำดับ แต่วิธีนี้จะใช้ได้ในกรณีที่ข้อมูลถูกเก็บเรียงตามลำดับคีย์จากมากไปน้อยแล้วเท่านั้น การค้นหาโดยวิธีนี้แต่ละครั้งในการเปรียบเทียบสามารถลดจำนวนข้อมูลได้ครั้งละประมาณครึ่งหนึ่งของจำนวนที่เหลือ โดยวิธีการค้นหาจะเป็นดังนี้คือ กำหนดข้อมูลเรียงลำดับคีย์จากน้อยไปมาก และการเปรียบเทียบคีย์จะเริ่มจากคีย์ที่อยู่ตำแหน่งกึ่งกลาง ซึ่งจุดนี้จะแบ่งข้อมูลออกเป็นสองส่วนเท่าๆกัน โดยส่วนแรกจะมีค่าคีย์น้อยกว่าค่าคีย์ที่ตำแหน่งกึ่งกลาง และส่วนหลังจะมีค่าคีย์มากกว่าค่าคีย์ที่ตำแหน่งกึ่งกลาง และการเปรียบเทียบคีย์ ถ้าค่าเป้าหมายน้อยกว่าค่าคีย์ที่เปรียบเทียบแสดงว่าข้อมูลที่ต้องการจะอยู่ในส่วนแรก และในทางตรงกันข้าม ถ้าค่าเป้าหมายมากกว่าค่าคีย์ที่เปรียบเทียบแสดงว่าข้อมูลที่ต้องการจะอยู่ในส่วนหลัง เมื่อทราบว่าข้อมูลที่ต้องการอยู่ในส่วนใด การเปรียบเทียบในครั้งต่อไปจะไปที่ตำแหน่งกึ่งกลางของส่วนนั้นซึ่งจะแบ่งข้อมูลออกเป็นสองส่วนเช่นเดียวกัน และดำเนินการในลักษณะนี้จนกว่าจะพบข้อมูลที่ต้องการ

การที่สามารถลดจำนวนข้อมูลลงได้ครั้งละประมาณครึ่งหนึ่งของที่เหลือ การค้นหาแบบไบนารี่จึงใช้เวลาในการค้นหาข้อมูลน้อยมาก แม้ว่าข้อมูลจะมีจำนวนมาก แต่การค้นหาแบบนี้ โครงสร้างข้อมูลต้องอยู่ในลักษณะที่สามารถเข้าถึงกึ่งกลางกลุ่มได้ง่าย มิฉะนั้นเวลาที่ใช้ในการค้นหาอาจจะไม่ลดลง ดังนั้นโครงสร้างข้อมูลที่จะค้นหาแบบไบนารี่ได้ดีคือ โครงสร้างแถวลำดับ

2.4.3 การค้นหาข้อมูลในไบนารี่เสิร์ชทรี

การค้นหาข้อมูลในไบนารี่เสิร์ชทรี จะนำค่าเป้าหมายไปเปรียบเทียบกับคีย์ในโหนด ถ้าค่าเป้าหมายน้อยกว่าค่าคีย์ที่เปรียบเทียบให้เลื่อนไปทางซ้ายของโหนดนั้น ในทางตรงกันข้าม ถ้าค่าเป้าหมายมากกว่าค่าคีย์ที่เปรียบเทียบให้เลื่อนไปทางขวาของโหนดนั้น เมื่อไปถึงโหนดใดให้เปรียบเทียบค่าคีย์และดำเนินการด้วยวิธีเดียวกันดังกล่าว การค้นหาจะสิ้นสุดเมื่อพบข้อมูลที่ต้องการ หรือสุดทางที่จะเลื่อนต่อไป ซึ่งแสดงว่าไม่พบข้อมูลที่ต้องการ

บทที่ 3

Interfacing to the IBM PC for he prototype board

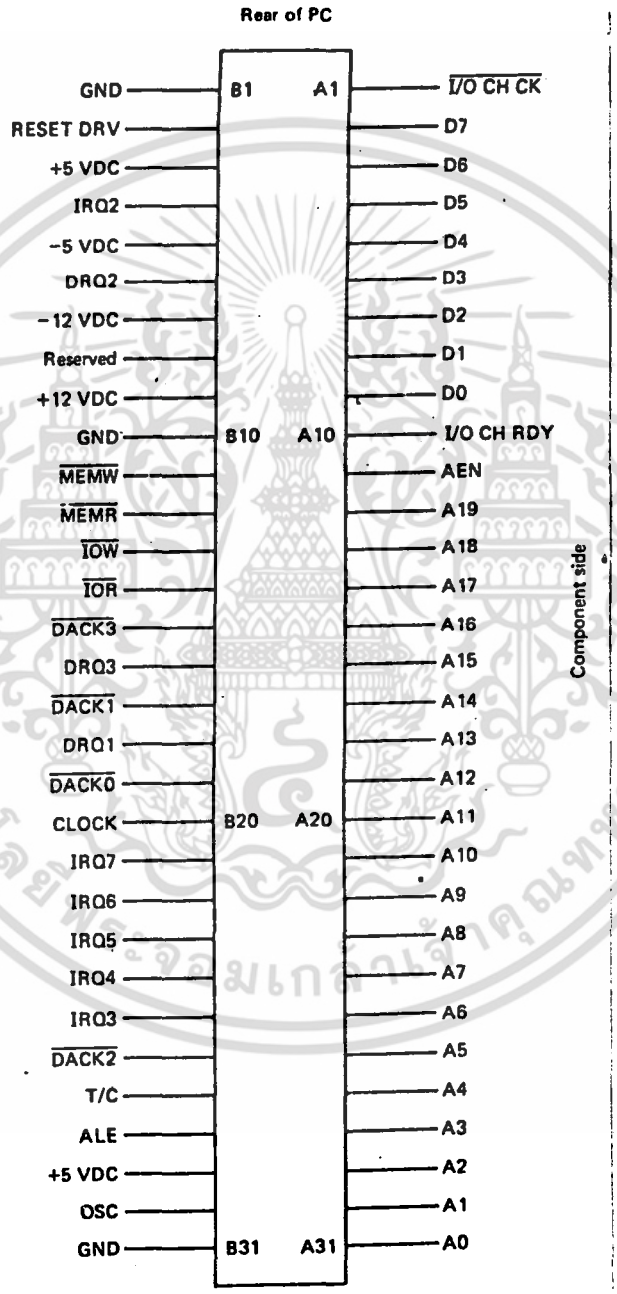
ภายใน IBM PC ได้มีการออกแบบให้สามารถเพิ่มวงจรอินเทอร์เฟสเข้าไปภายหลังได้ โดยผ่านทางสล๊อตที่อยู่บนเมนบอร์ด สำหรับสล๊อตบนเมนบอร์ดนี้จะมี 5 สล๊อต ซึ่งแต่ละสล๊อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็นข้างละ 31 ขา ส่วนการเรียกตำแหน่งของขาสล๊อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้าย หรือ ขวา) ของสล๊อต โดยขาที่อยู่ทางด้านซ้ายของสล๊อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา ส่วนขาที่อยู่ทางด้านขวาของสล๊อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา แต่ละขาของสล๊อตเหล่านี้จะต่อกับเส้นสัญญาณต่างๆบนเมนบอร์ด ทำให้การสร้างวงจรอินเทอร์เฟสกับ IBM PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาสล๊อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอดเดรส (Address bus) บัสข้อมูล (Data bus) บัสควบคุมการเขียน/อ่านข้อมูลจากหน่วยความจำหรือพอร์ท I/O เส้นสัญญาณการขออินเทอร์รัพท์ของวงจรอินเทอร์เฟส เส้นสัญญาณสำหรับการขอ DMA สัญญาณฐานเวลา (Timing signal) ต่างๆเส้นสัญญาณสำหรับการรีเฟรชหน่วยความจำ และเส้นสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O check) นอกจากนี้แล้วสล๊อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆที่ใช้ในระบบอีก คือ +5Vdc , -5Vdc , +12Vdc , -12Vdc และต่อเข้ากับกราวนด์ (Ground) ของระบบ ซึ่งรายละเอียดของขาสัญญาณต่างๆบนสล๊อตแสดงดังรูปที่3.1 IBM PC system bus

3.1 การอ้างแอดเดรสของพอร์ท I/O

การอ้างแอดเดรสของพอร์ท I/O ต่างๆนั้นจะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 มีทั้งสิ้น 64 K ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้นต้องใช้จำนวนเส้นแอดเดรสในการบัสแอดเดรส ในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A_0-A_{15} แต่สำหรับใน IBM PC ถูกออกแบบให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นดังนั้น คือ A_0-A_9 ดังนั้นในการอ้างแอดเดรสของพอร์ทของอุปกรณ์ หรือชิพซัพพอร์ทใดๆที่ใช้ร่วมกับ IBM PC จึงใช้จำนวนแอดเดรสเพียง 10 เส้น โดยเส้นแอดเดรสที่เหลือ คือ A_{10}-A_{15} นั้นจะไม่ถูกนำมาใช้

ฉะนั้นแอดเดรสตั้งแต่ A_0-A_9 จะมีแอดเดรสพอร์ททั้งสิ้น 1024 แอดเดรสพอร์ท ซึ่ง

ใน I/O port-address map จะแบ่งออกเป็น 2 ส่วน คือ แอดเดรสพอร์ท 512 แอดเดรส คือตั้งแต่ 0000H-01FFH จะเป็นแอดเดรสพอร์ทของ system board และแอดเดรสพอร์ท 512 แอดเดรสคือตั้งแต่ 0200H-03FFH จะใช้ใน feature card ports ซึ่งอยู่ในสล๊อตของ IBM PC ทั้ง 5 สล๊อต แต่สำหรับ Prototype board นั้นจะอยู่ตั้งแต่ 0300H-031FH ซึ่งมีเพียง 32 แอดเดรสพอร์ทเท่านั้น

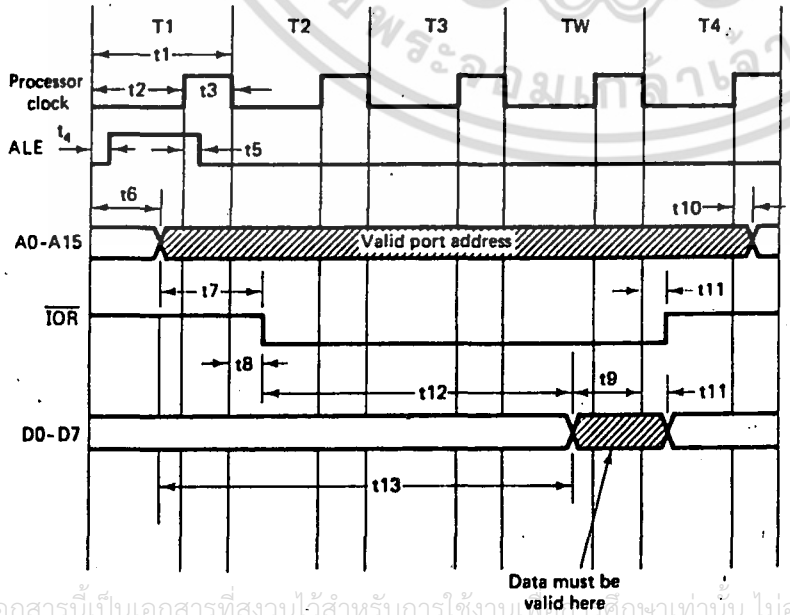


รูปที่ 3.1 แสดงขาต่างๆบนสล๊อต IBM PC

3.2 สัญญาณของอินพุท/เอ้าท์พุท พอร์ท (I/O port timing)

3.2.1 บัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O

ในขณะที่ 8088 เอ็กซ์คิวท์ (Excute) ชุดคำสั่ง IN ซึ่งเป็นชุดคำสั่งที่จะทำให้ 8088 ทำการอ่านข้อมูล จากพอร์ทที่กำหนดในส่วนของโอเปอร์แรนด์ (Operand) นั้น 8088 จะสร้างบัสไซเคิลในการอ่านข้อมูลจากพอร์ทเพื่อให้พอร์ทที่ถูกกำหนดนั้นส่งข้อมูลมาบนบัสข้อมูล บัสไซเคิลนี้จะเริ่มต้นในช่วงของคล็อก T_1 ซึ่งเป็นช่วงเวลาที่สัญญาณ ALE แอคทีฟ (ลอจิก "1") สัญญาณ ALE นี้จะถูกใช้เพื่อให้แสดงอุปกรณ์ที่ทำงานรวมกับ 8088 ทราบว่าข้อมูลที่อยู่ในบัสแอดเดรสในช่วงขอบลงของสัญญาณ ALE นั้นเป็นแอดเดรสของพอร์ทที่ 8088 ต้องการจะติดต่อด้วย (ในที่นี้ คือ แอดเดรสของพอร์ทที่ 8088 ต้องการอ่านข้อมูล) หลังจากนั้นในช่วงของคล็อก T_2 สัญญาณ IOR จะแอคทีฟ (ลอจิก "0") ซึ่งเป็นการแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าบัสไซเคิลนี้เป็นบัสไซเคิลในการอ่านข้อมูลจากพอร์ท (I/O port Read bus cycle) และเป็นการทำให้พอร์ทที่มีแอดเดรสตรงกับค่าแอดเดรสที่อยู่บนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสมูล เมื่อพอร์ทที่ถูกอ้างถึงโดยแอดเดรสบนบัสข้อมูลทำการส่งข้อมูลมาบนบัสข้อมูลแล้ว 8088 จะอ่านข้อมูลนั้น ในช่วงเริ่มต้นของคล็อก T_4 จากนั้นสัญญาณ IOR จะถูกปรับให้เป็นลอจิก "1" และจะสิ้นสุดการทำงานในบัสไซเคิลเมื่อสิ้นสุดช่วงเวลาของคล็อก T_4 จะเห็นได้ว่าโดยปกติบัสไซเคิลนี้จะใช้เวลาเท่ากับช่วงเวลาของคล็อก 4 ลูก แต่ภายใน IBM PC นั้นจะนับเพิ่มช่วงเวลา T_W ในบัสไซเคิลนี้ขึ้นอีก 1 ลูก ทำให้ช่วงเวลาในบัสไซเคิลเพิ่มขึ้นเป็น 1.05 sec โดย T_W นี้จะถูกเพิ่มเข้าไประหว่างช่วงต่อของคล็อก T_3 และ T_4 เพื่อให้พอร์ท I/O ซึ่งปกติมักมีความเร็วในการทำงานต่ำ สามารถที่จะส่งข้อมูลออกมาบนบัสข้อมูลได้ทัน



Symbol	Max.	Min.
t1	-	209.5
t2	-	124.5
t3	-	71.8
t4	15	
t5	15	-
t6	128	16
t7	-	91.5
t8	35	10
t9	-	42
t10	35	10
t11	-	10
t12	-	551.5
t13	-	668

*All times are in nanoseconds

รูปที่ 3.2 บัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O

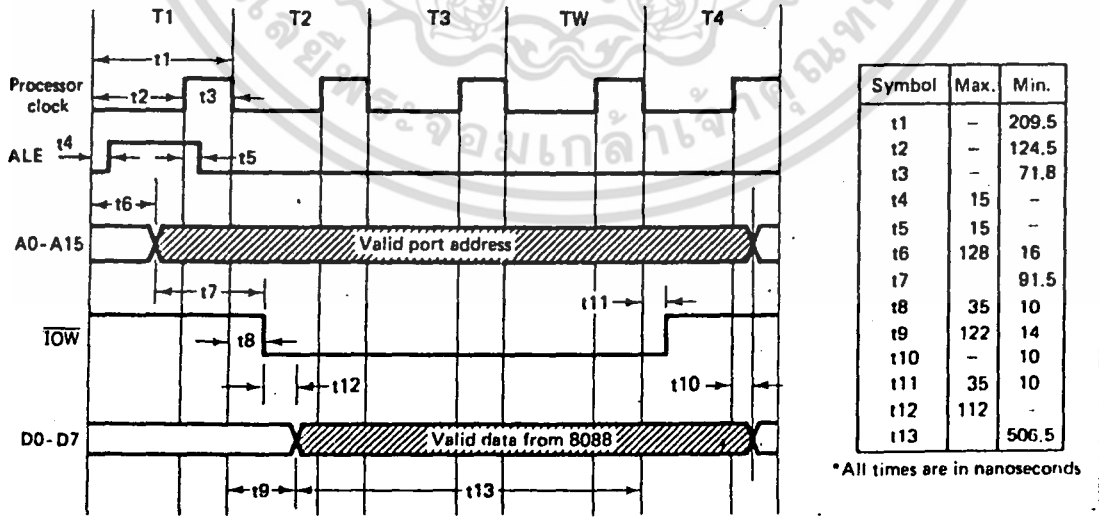
3.2.2 บัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท

ในขณะที่ 8088 เอ็กซ์คิวท์ คำสั่ง OUT ซึ่งเป็นชุดคำสั่งที่ทำให้ 8088 ทำการเขียนข้อมูลลงบนพอร์ทที่กำหนดในส่วนของโอเปอร์แรนด์นั้น 8088 จะสร้างบัสไซเคิลในการเขียนข้อมูลลงพอร์ท เพื่อให้พอร์ทที่ถูกกำหนดทำการรับข้อมูลที่อยู่บนบัสข้อมูล

บัสไซเคิลนี้จะเริ่มต้นในช่วงของคล็อก T_1 ซึ่งเป็นช่วงเวลาที่ ALE แอคทีฟ (ลอจิก "1") สัญญาณ ALE นี้จะถูกใช้เพื่อให้แสดงอุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าข้อมูลอยู่บนบัสแอดเดรส ในช่วงขอบขาลงของสัญญาณ ALE นั้นเป็นแอดเดรสพอร์ทที่ 8088 ต้องการจะติดต่อด้วย (ในที่นี้ คือ แอดเดรสของพอร์ท 8088 ที่ต้องการจะส่งข้อมูลให้)

หลังจากนั้นในช่วงคล็อก T_2 สัญญาณ IOW จะแอคทีฟ (ลอจิก"0") ซึ่งเป็นการแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าบัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท (I/O port Write bus cycle) จากนั้น 8088 จะทำการส่งข้อมูลที่ต้องการส่งให้พอร์ทที่กำหนดนั้นออกมาทางบัสข้อมูลในช่วงคล็อก T_4 สัญญาณ IOW จะถูกปรับให้กลับลอจิกเป็น "1" และสิ้นสุดการทำงานในบัสไซเคิล เมื่อสิ้นสุดการทำงานของคล็อก T_4

สำหรับในกรณีของบัสไซเคิลนี้ IBM PC จะทำการเพิ่ม T_W เข้าไประหว่าง T_3 และ T_4 เช่นเดียวกับกรณีของบัสไซเคิลในการอ่านข้อมูลจากพอร์ทเพื่อให้พอร์ท I/O สามารถที่จะทำงานในบัสไซเคิลได้ทัน



Symbol	Max.	Min.
t1	–	209.5
t2	–	124.5
t3	–	71.8
t4	15	–
t5	15	–
t6	128	16
t7		91.5
t8	35	10
t9	122	14
t10	–	10
t11	35	10
t12	112	–
t13		506.5

*All times are in nanoseconds

รูปที่ 3.3 บัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O

3.3 อุปกรณ์ช่วยติดต่อ 8255 PIA

ในการเชื่อมต่อระหว่าง IBM PC กับ prototype board สิ่งที่สำคัญที่นอกเหนือจากการ Decode Address Port สำหรับ prototype board แล้ว จะต้องมีอุปกรณ์ซึ่งใช้ในการช่วยติดต่อระหว่าง ซีพียู กับอุปกรณ์ภายนอก หรือเรียกว่า อุปกรณ์อินพุท/เอาท์พุท (I/O Devices) ซึ่งอุปกรณ์ช่วยติดต่อที่ทำหน้าที่ได้เช่นนี้คือ 8255 PIA (Programmable Interface Adapter)โดย 8255 PIA จะเป็นพอร์ทให้แก่ไมโครโปรเซสเซอร์ 8088 ได้ถึง 3 พอร์ท โดยจะเรียกพอร์ทต่างๆ เป็น พอร์ท A, พอร์ท B, พอร์ท C และที่พิเศษคือพอร์ททุกพอร์ทจะเป็นได้ทั้งพอร์ทอินพุทและพอร์ทเอาท์พุท

การกำหนดแอดเดรสพอร์ทของพอร์ทต่างๆบน 8255 PIA จากการ Decode แอดเดรสที่ขา y_0 มีแอดเดรสพอร์ทอยู่ที่ 300H-303H ซึ่งจะใช้ในการกำหนดพอร์ทต่างๆบน 8255

จากตาราง สัญญาณควบคุมการทำงานของ 8255

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
			Input operation (read)		
0	0	0	1	0	Port A to data bus
0	1	0	1	0	Port B to data bus
1	0	0	1	0	Port C to data bus
			Output operation (write)		
0	0	1	0	0	Data bus to port A
0	1	1	0	0	Data bus to port B
1	0	1	0	0	Data bus to port C
1	1	1	0	0	Data bus to control
			Disable function		
×	×	×	×	1	Data bus to three-state
1	1	0	1	0	Illegal condition
×	×	1	1	0	Data bus to three-state

ตารางที่ 3.1 แสดง Basic Operation of the Intel 8255 PIA

ดังนั้นจึงกำหนดให้ PORT A มีแอดเดรส 300H

PORT B มีแอดเดรส 301H

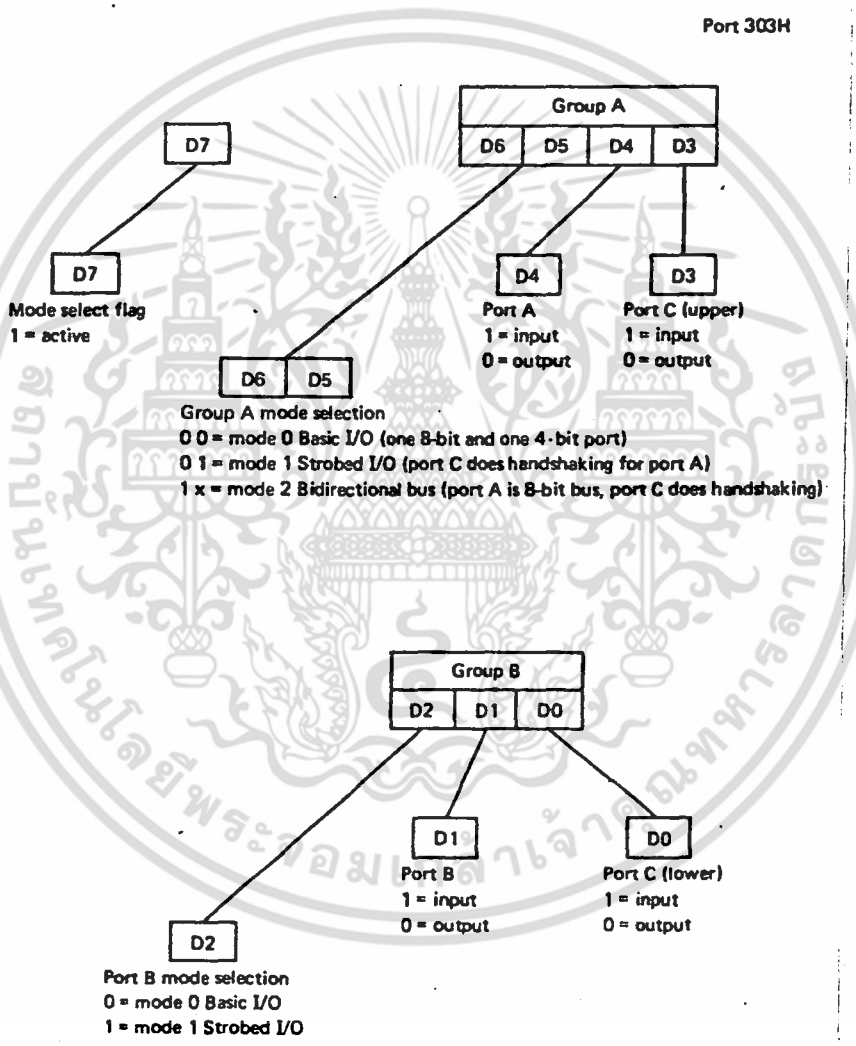
PORT C มีแอดเดรส 302H

REGISTER CONTROL WORD มีแอดเดรส 303H

3.3.1 การทำงานของ8255 ในโหมด 0 : Basic I/O

การเซ็ท 8255 ในโหมด 0 จะต้องส่ง control word ให้แก่รีจิสเตอร์ควบคุม เสียก่อน control word นี้จะกำหนดการทำงานให้แก่แต่ละพอร์ทของ 8255

รายละเอียดแต่ละบิตของรีจิสเตอร์ควบคุมของ 8255

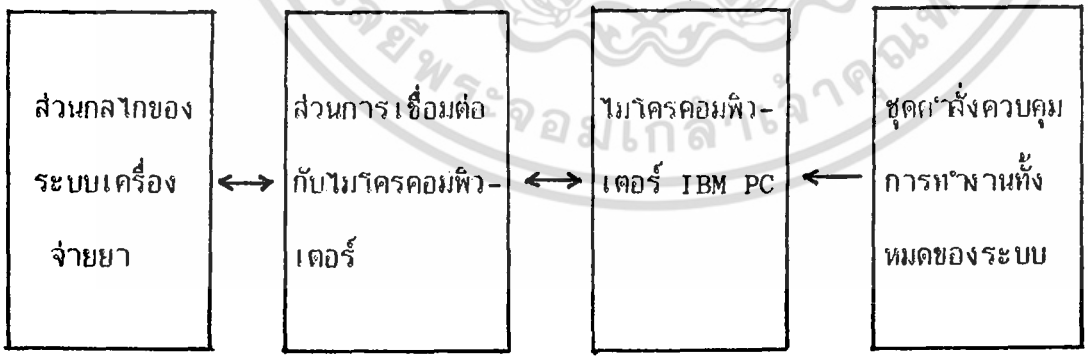


รูปที่ 3.4 แสดงรายละเอียดแต่ละบิตของรีจีสเตอร์ควบคุม 8255 PIA

บทที่ 4

ระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC

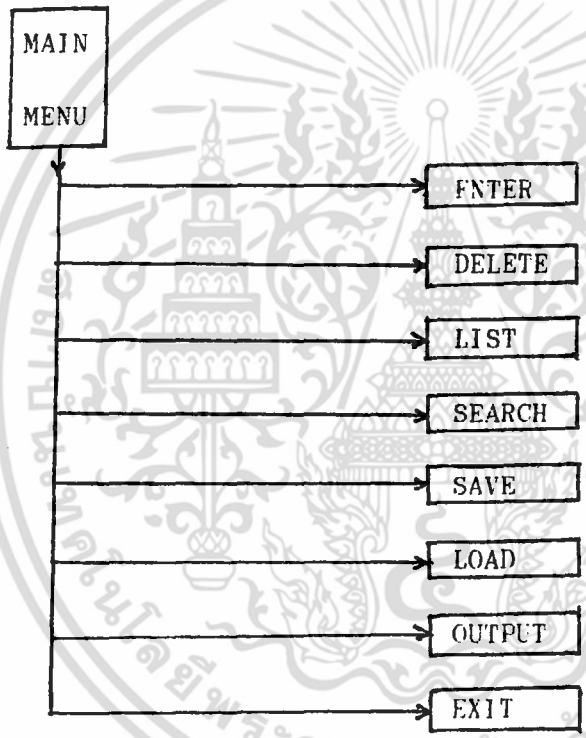
ภายในระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC นั้นจะประกอบไปด้วยส่วนที่สำคัญ 3 ส่วนโดยแต่ละส่วนมีความเกี่ยวข้องสัมพันธ์กัน ซึ่งจะกล่าวแต่ละส่วนดังนี้ ส่วนที่หนึ่งเป็นส่วนที่เกี่ยวกับชุดคำสั่งที่ใช้กับเครื่องไมโครคอมพิวเตอร์ IBM PC หรือที่เรียกว่าโปรแกรมควบคุมการทำงานของระบบ ซึ่งจะประกอบด้วยชุดคำสั่งที่เป็นเมนูเลือกการทำงานในตัวเลือกต่างๆ ซึ่งได้แก่ การเก็บข้อมูลของยา รหัสที่อยู่ของตำแหน่งของขวดยา และชุดคำสั่งที่ใช้ในการติดต่อกับตัวตรวจจับเม็ดยา และระบบปิดเปิดของขวดยาแต่ละขวด โดยส่วนที่เป็นชุดคำสั่งทั้งหมดนี้รวมเรียกว่าส่วนของซอฟแวร์ของระบบ ส่วนที่สองได้แก่ส่วนที่เกี่ยวข้องกับการเชื่อมต่อไมโครคอมพิวเตอร์ IBM PC กับอุปกรณ์ภายนอก ซึ่งจะเป็นการ์ด (Prototypeboard) ที่เสียบลงในสล๊อตของไมโครคอมพิวเตอร์ IBM PC และส่วนสุดท้ายคือส่วนของกลไกด้านเครื่องกล โดยจะมีระบบกลไกที่ทำให้เม็ดยาสามารถเคลื่อนที่อย่างเป็นระเบียบ ผ่านตัวตรวจจับทางแสง และระบบปิดของขวดยา เมื่อนำทั้งสามส่วนประกอบกันเข้าเป็นระบบจะทำให้ได้ระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC ซึ่งแสดงให้เห็นดังรูปที่ 4.1 และรายละเอียดต่างๆของแต่ละส่วนจะกล่าวในหัวข้อต่อไป



รูปที่ 4.1 แสดงภาพรวมของระบบเครื่องจ่ายยาอัตโนมัติ ควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC

4.1 ส่วนของซอฟแวร์ของระบบ

ส่วนของซอฟแวร์ของระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC จะประกอบด้วย ชุดคำสั่งที่ใช้เป็นเมนูหลักในการทำงานของระบบทั้งหมด ซึ่งได้แก่การเก็บข้อมูลของยาแต่ละชนิด(ชื่อยา และรหัสของตำแหน่งยา) การลบข้อมูล DELETE การ List ข้อมูล และอื่นๆอีกเป็นต้น ภายในชุดคำสั่งของเมนูหลักจะมีชุดคำสั่งย่อยที่เกี่ยวกับการทำงานของแต่ละตัวเลือก เช่น ชุดคำสั่งย่อยสำหรับการ ENTER ชุดคำสั่งย่อยสำหรับการ SEARCH เป็นต้น ซึ่งรายละเอียดจะแสดงให้เห็นดัง โฟลว์ชาร์ต การทำงานของชุดคำสั่งต่างๆของระบบ



รูปที่ 4.2 แสดงขั้นตอนการทำงานของชุดคำสั่งของระบบ

จากรูปที่ 4.2 แสดงขั้นตอนการทำงานของชุดคำสั่งของระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ โดยจะประกอบด้วยชุดคำสั่งย่อย (โปรแกรมย่อย) ต่างๆมากมายซึ่งจะกล่าวถึงหน้าที่ของแต่ละชุดคำสั่งย่อยดังนี้

ชุดคำสั่งย่อย ENTER

-จะจอง memory ใน heapเพื่อเก็บ data

-เก็บ data ในตัวแปร buffer

-เก็บ data จากตัวแปร buffer ลงใน memory ที่จองไว้ในลักษณะ linked list order sort

ชุดคำสั่งย่อย DELETE

-เป็นการลบ data ใน linked list โดยใช้ name เป็น key

ชุดคำสั่งย่อย list

-แสดง data ตามลำดับทั้งหมดที่อยู่ใน linked list

ชุดคำสั่งย่อย Search

-แสดง data เฉพาะชื่อที่ต้องการจะค้นหาใน linked list โดยใช้ nameของชื่อยาเป็น key

ชุดคำสั่งย่อย save

-เก็บ data ทั้งหมดใน memory ของ linked list ลงในไฟล์ใน disk

ชุดคำสั่งย่อย load

-เรียก data จากไฟล์ใน disk ลง memory ใน heap

ชุดคำสั่ง output

-ส่ง code ของยา และจำนวนเม็ดยาที่ต้องการออก port โดยใช้ name ของชื่อยาเป็น key

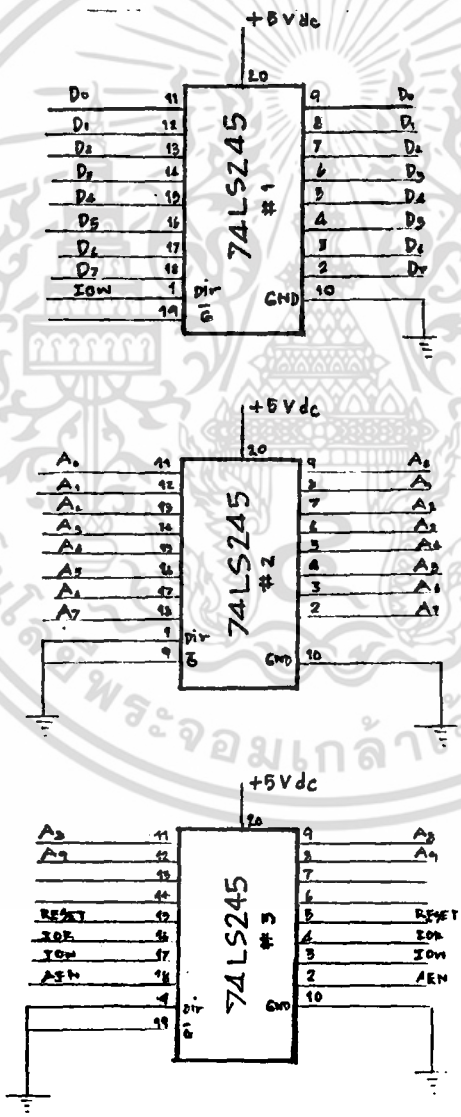
ชุดคำสั่ง exit

-ออกจาก program สู่ dos

4.2 <u>ส่วนการเชื่อมต่อกับไมโครคอมพิวเตอร์ IBM PC ของระบบเครื่องจ่ายยาอัตโนมัติ ควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC</u>

การเชื่อมต่อกับไมโครคอมพิวเตอร์ IBM PC ของระบบจะเป็นในรูปของ Prototype Board ที่ต่อเข้ากับ สล๊อตของไมโครคอมพิวเตอร์ IBM PC โดยจะประกอบด้วย Bus driver;decoder and Control logic;programmable I/O ports และ programmable counter ซึ่งจะแสดงรายละเอียดดังนี้

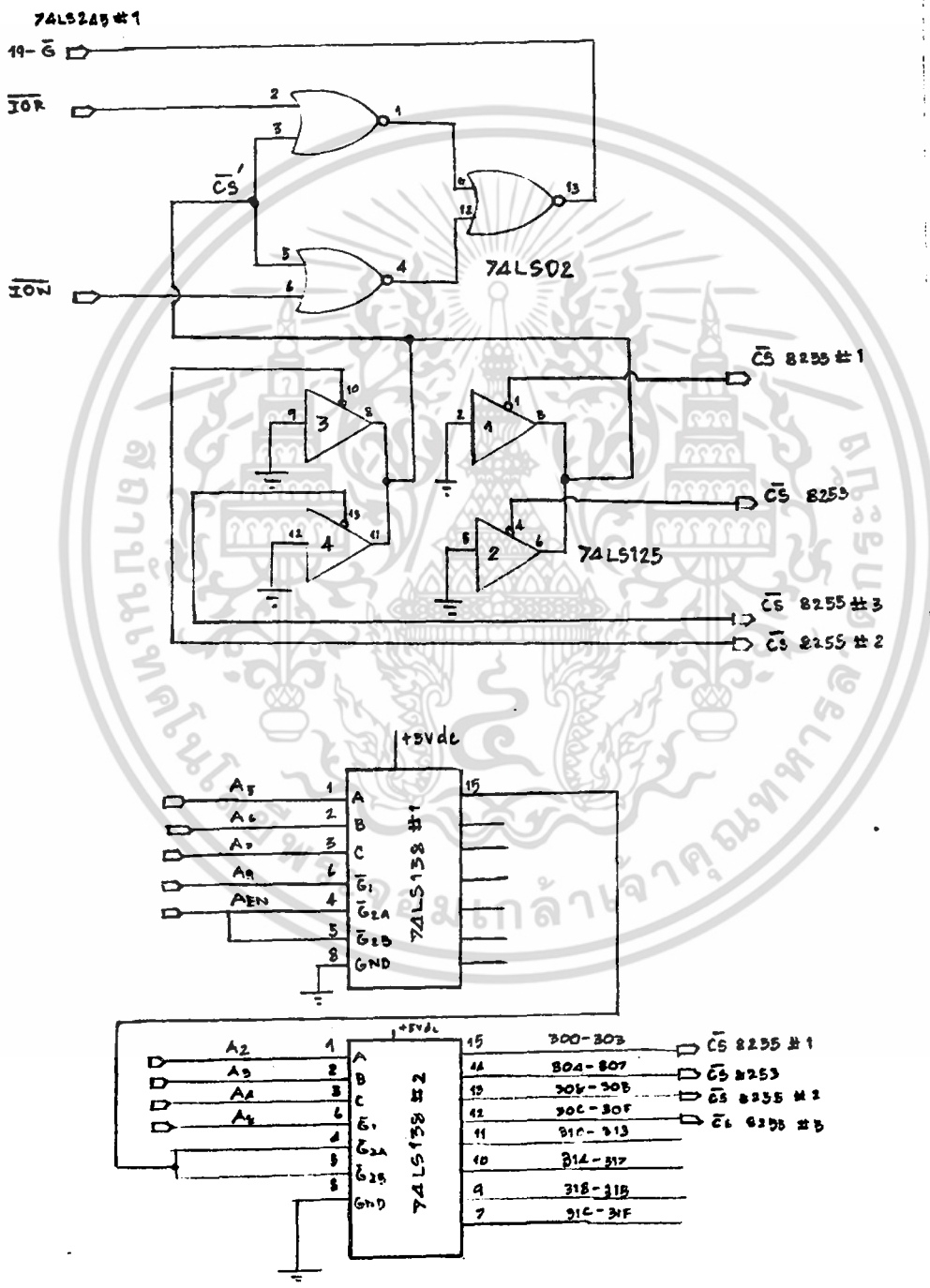
Bus driver จะประกอบด้วย IC #74LS245 (Transciver with 3 state Output) จำนวน 3 ตัวทำหน้าที่ในการเป็นบัฟเฟอร์และกำหนดทิศทางของสัญญาณต่างๆ โดยจะแสดงดังรูป 4.3



รูปที่ 4.3 แสดง Bus driver circuit

decoder and Control logic

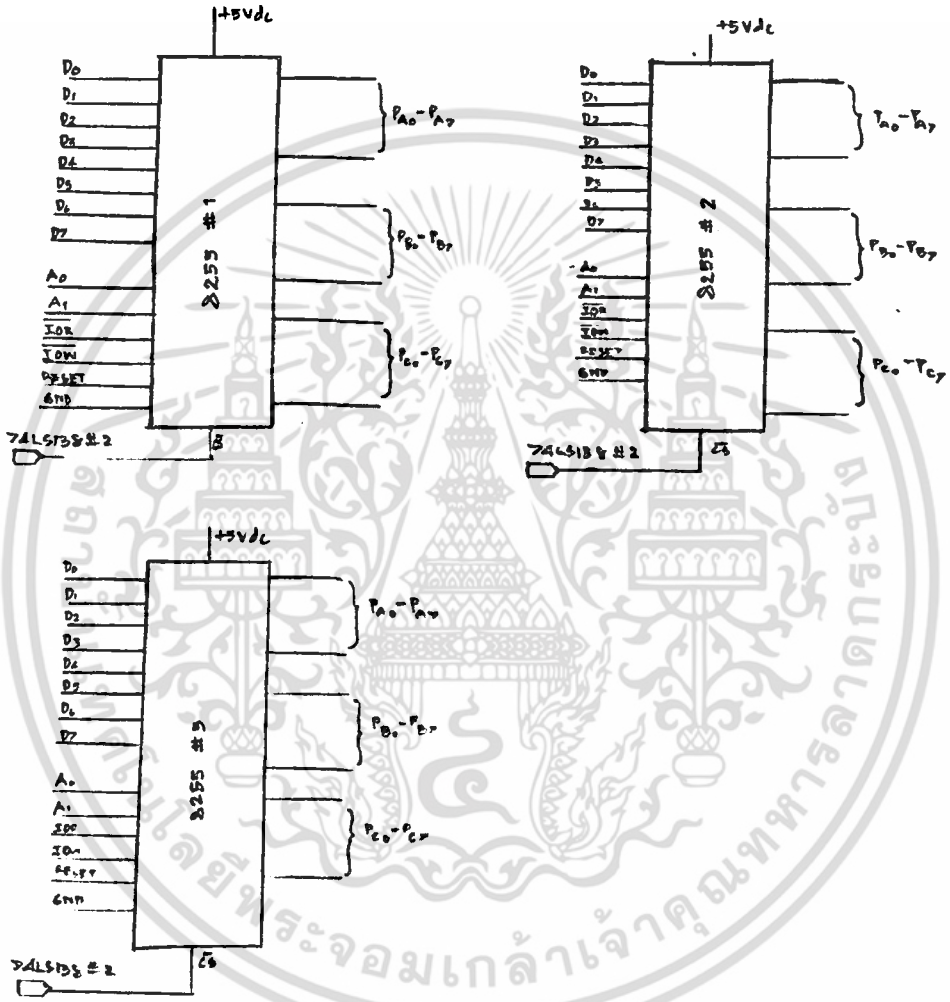
จะประกอบด้วย IC #74LS138 (3-Line-to-8-Line Decoder) จำนวน 2 ตัว ทำหน้าที่ในการ decoder address ports ที่ใช้ใน Prototype board และ IC #74LS02 (Positive NOR Gate) จำนวน 1 ตัว และ IC #74LS125 (Bus Buffer Gate with three state Output) จำนวน 1 ตัว ทำหน้าที่ในการเป็น Control logic โดยแสดงดังรูป 4.4



รูปที่ 4.4 แสดง decoder and Control logic circuit

Programmable I/O ports

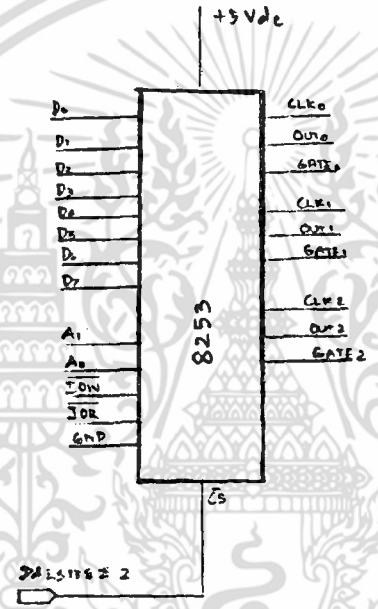
จะประกอบด้วย IC #8255 PPI(Programmable Peripheral Interface) จำนวน 3 ตัว ทำหน้าที่ช่วยติดต่อกับอุปกรณ์ภายนอก I/O devices โดยแสดงดังรูป 4.5



รูปที่ 4.5 แสดง Programmable I/O ports circuit

Programmable Counter

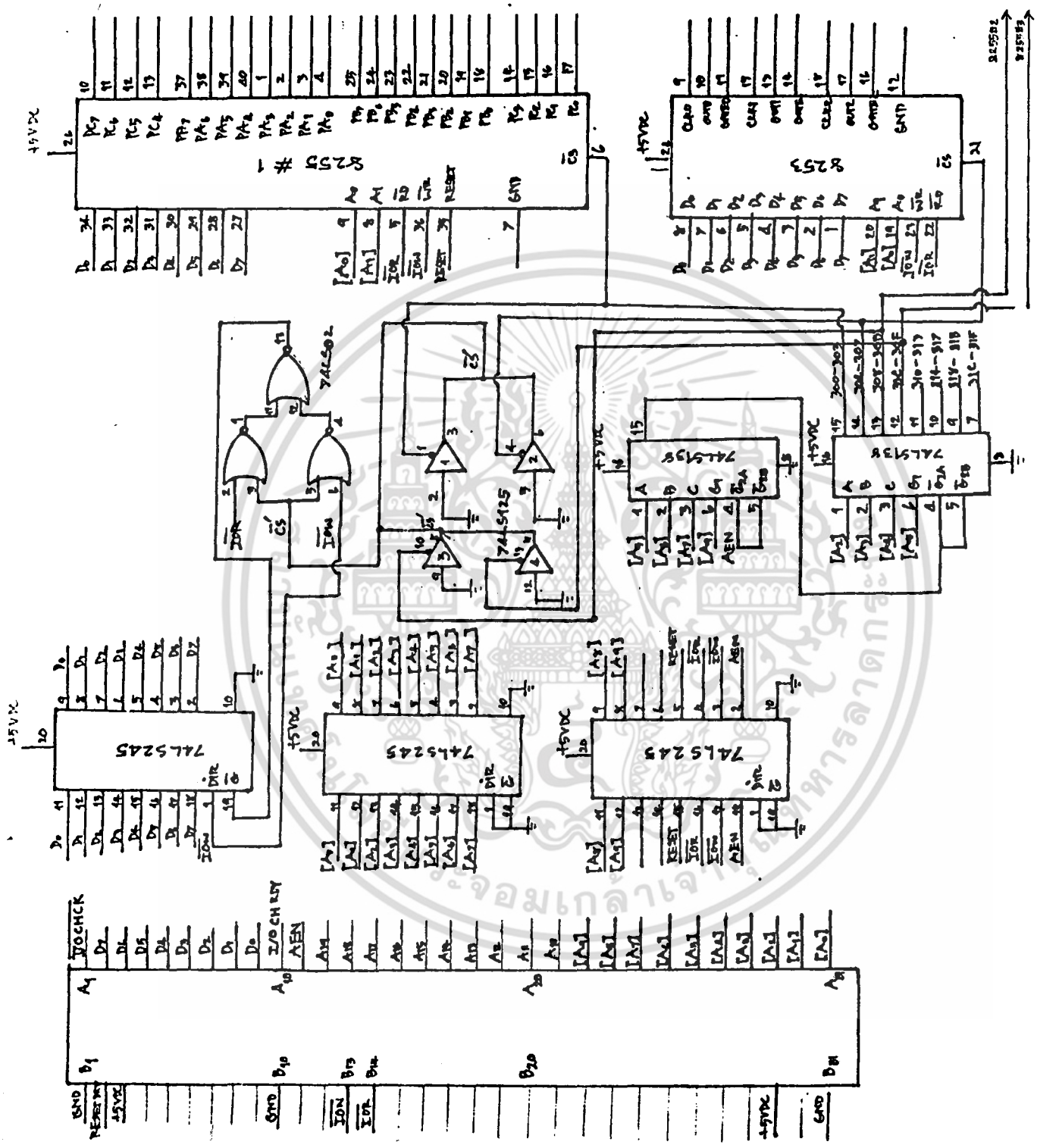
จะเป็น IC #8253 (Programmable counter) ซึ่งจำทำหน้าที่เป็น counter ของ Prototype Board แสดงดังรูป 4.6



รูปที่ 4.6 แสดง Programmable counter circuit

เมื่อนำแต่ละส่วนของ Prototype Board ประกอบกันจะได้ วงจรของ Prototype Board

ดังรูป 4.7

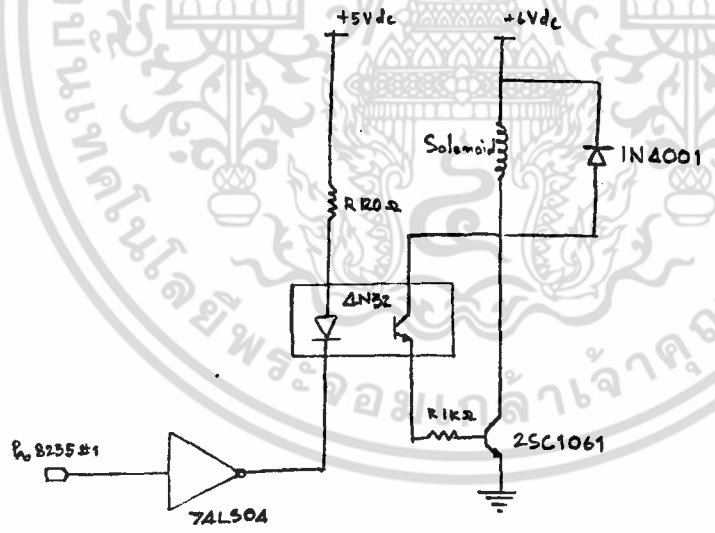


รูปที่ 4.7 แสดง Prototype Board circuit ของระบบเครื่องจ่ายยาอัตโนมัติ

ควบคุมด้วยไมโครคอมพิวเตอร์

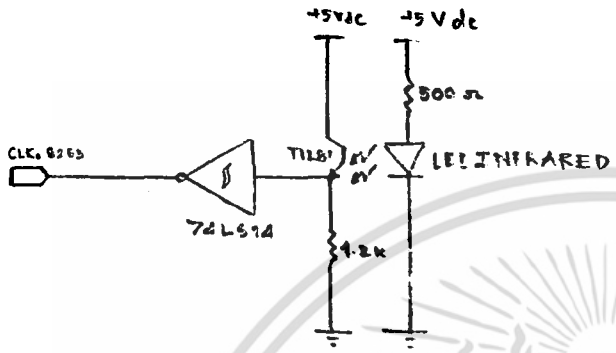
จากรูปที่ 4.7 เมื่อมีสัญญาณจาก Address bus,Data bus,IOR,IOW AEN,RESET สัญญาณเหล่านี้จะตอ้งผ่านเข้าส่วนของ Bus driver circuit จากนั้นเส้นสัญญาณต่างๆ จะไปทำหน้าที่ต่างๆ ต่อไป โดย Address bus A_0-A_9 และ AEN จะเป็นสัญญาณที่ใช้ในการ decode address ports ที่จะใช้ใน prototype board คือที่ address ports 300-31F โดยจะใช้ IC#74LS138 จำนวน 2 ตัวในการ decode address ports และเส้นสัญญาณ IOR IOW และ address ports ที่เลือกใช้คือ 300-30F จะต้องผ่านเข้าไปในส่วนของ control logic เพื่อทำให้เป็นเส้นสัญญาณ chip-select ต่างๆไปเลือก programmable I/O ports #8255 แต่ละตัว และไปเลือก programmable counter #8253 จากนั้น IC #8255 และ IC #8253 จะไปเชื่อมต่อกับอุปกรณ์ที่ใช้เป็นตัวตรวจจับเม็ดยา และอุปกรณ์ที่ใช้เป็นตัวปิดเปิดขวดยา ต่อไป

ส่วนวงจรที่เกี่ยวกับการปิดเปิดวาวล์(Solenoid switch) จะเป็นดังนี้



โดยรายละเอียดของวงจรประกอบด้วย IC #74LS04 เป็นอินเวิอด์เตอร์ #2SC1061,#4N32 เป็นPhototransistor NPN Darlington ทำหน้าที่ขับ Solinoid switch และ IN4001 ทำหน้าที่บังคับให้กระแสไหลผ่าน Solinoid switch ได้ทางเดียว

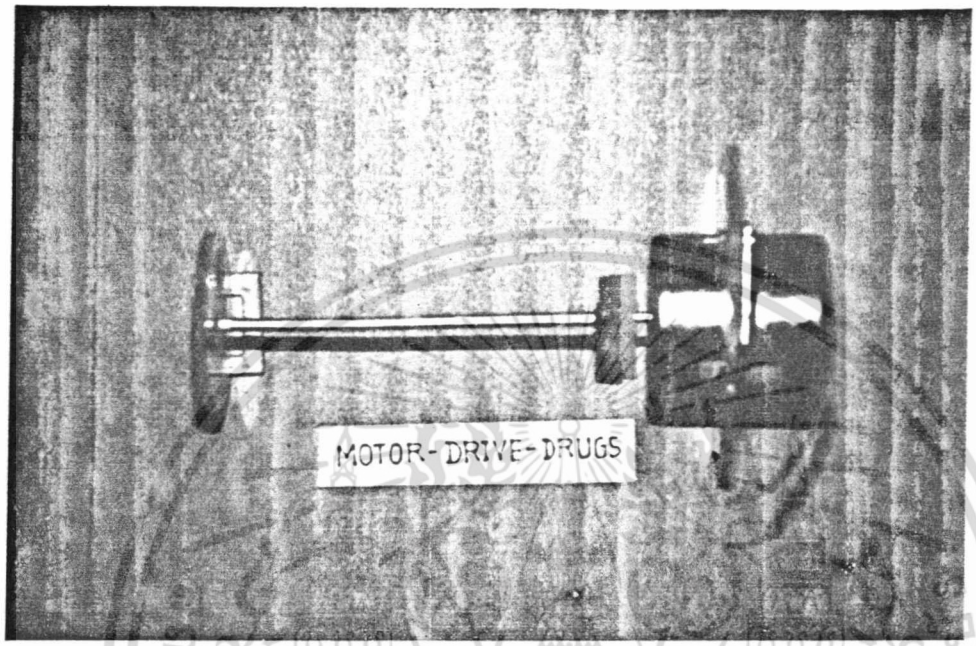
ส่วนของวงจรที่ใช้เป็นตัว Detector จะประกอบด้วย

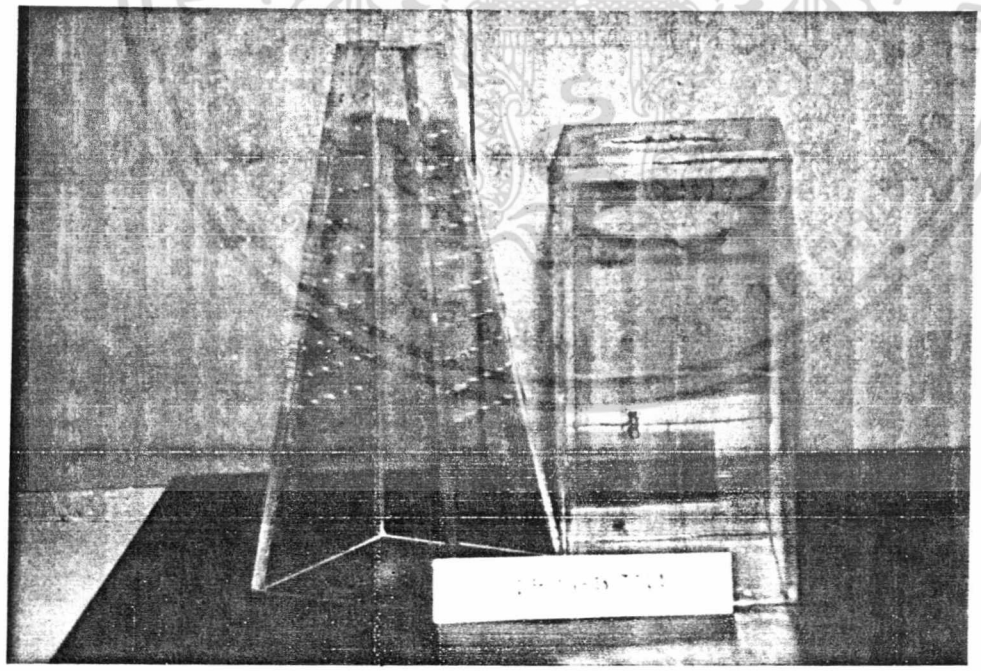


TIL81 และ LED INFRARED เป็นตัว Detector และมี #74LS14 เป็น Schmitt-Trigger Invertor ทำหน้าที่ให้สัญญาณ Pulse ที่เกิดจาก Detector ดีมากขึ้น

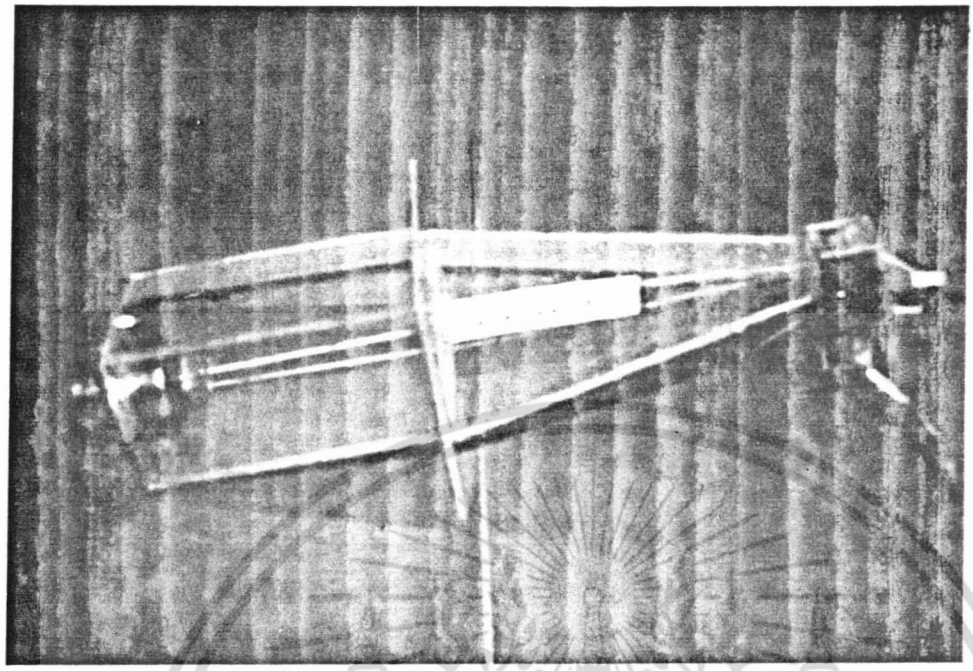
4.3 ส่วนที่เป็น MECHANIC ของระบบ

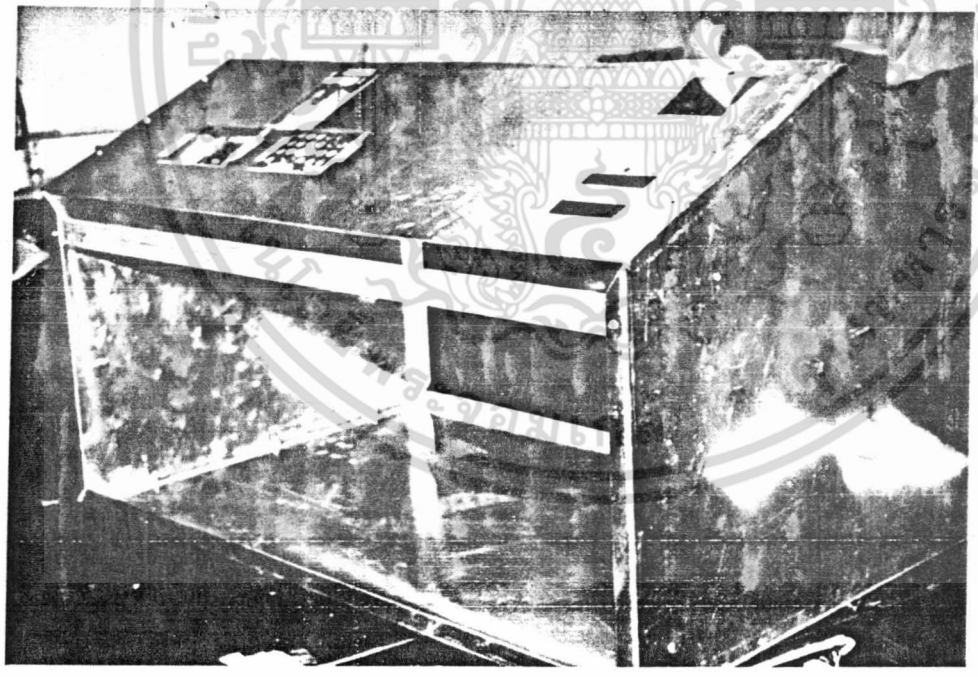
MECHANIC ของระบบจะประกอบด้วยขวดยา และตัวถัง การจัดวางเม็ดยาเพื่อให้เคลื่อนตัวอย่างเป็นระเบียบ ซึ่งจะแสดงดังรูปที่ 4.8 (ก),(ข)





รูปที่ 4.8 (ก) แสดงส่วนกลไก MOTOR DRIVE DRUG and DRUG BOTTLE

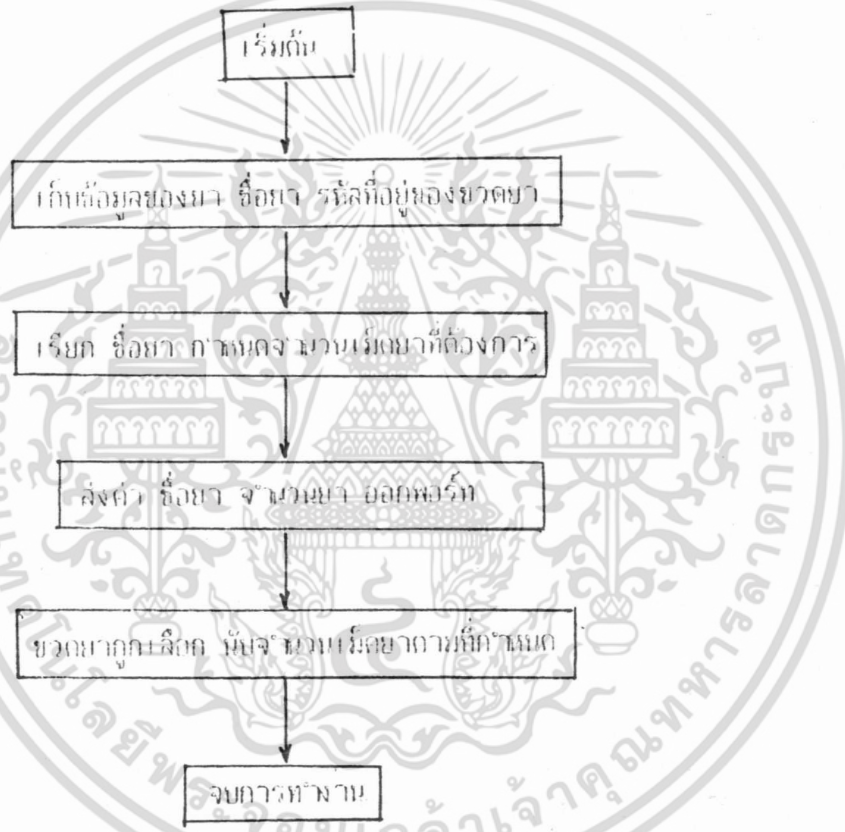




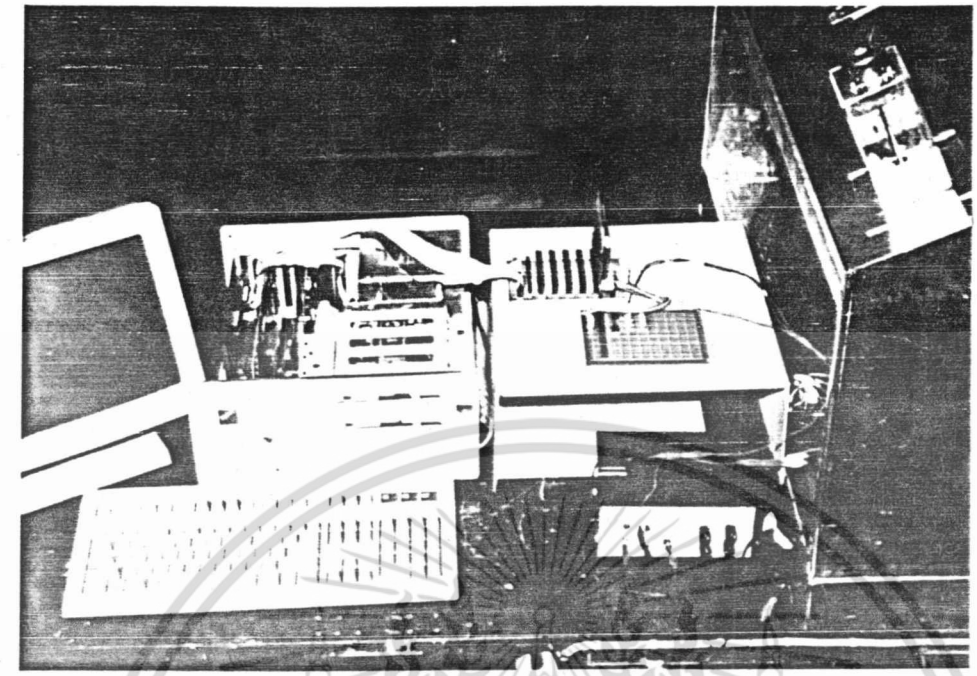
รูปที่ 4.8 (ข) แสดงส่วนของขวดจ่ายยาและฐานวางขวดยา

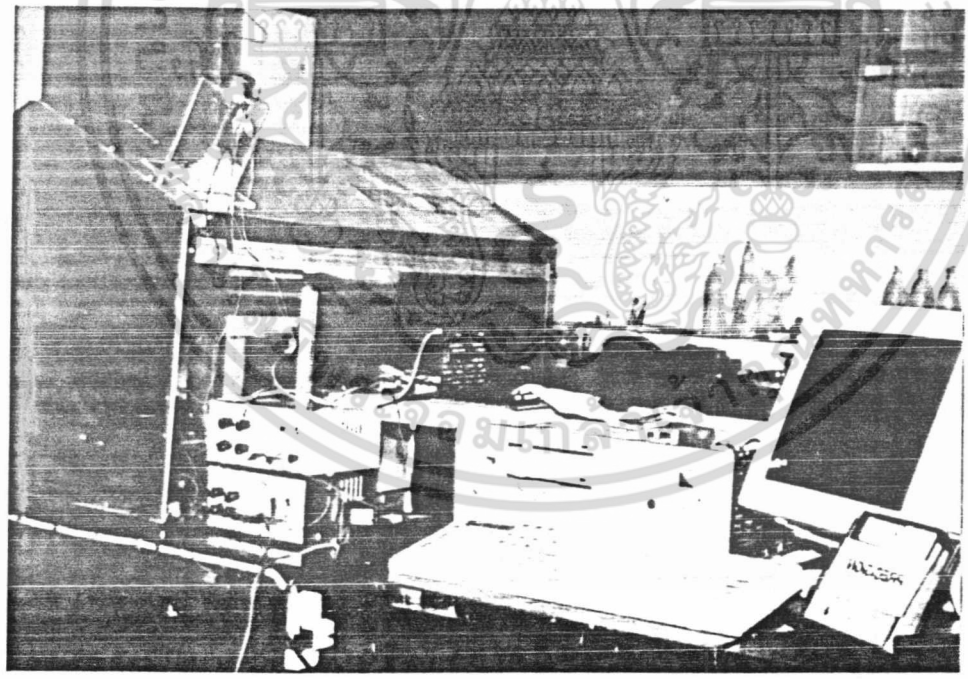
4.4 หลักการควบคุมการทำงานของระบบเครื่องจ่ายยาฯ

การควบคุมการทำงานของระบบเริ่มจากการเก็บข้อมูลของยา (ชื่อยา รหัสที่อยู่ของขวดยา)ไว้ในฐานข้อมูลของระบบ เมื่อมีการใช้งานของเครื่องจ่ายยาฯในแต่ละครั้งจะต้องเรียกข้อมูลยา(ชื่อยา)ตามใบสั่งยานั้นๆ จากนั้นจึงกำหนดจำนวนเม็ดยาที่ต้องการ ซึ่งหลังจากนี้คอมพิวเตอร์จะส่งค่าข้อมูลไปยังพอร์ทของ 8255 PIA เพื่อทำการเลือกขวดยาและให้ขวดยาที่ถูกเลือกทำการจ่ายยาตาม ที่กำหนด โดยการควบคุมการทำงานของระบบทั้งหมดแสดงดังรูปที่ 4.9



รูปที่ 4.9 แสดง Flow Chart การทำงานของระบบ





รูปที่ 4.10 แสดงภาพโดยรวมของระบบทั้งหมด

บทที่ 5

การทดสอบการใช้งาน

ในระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ IBM PC ได้ออกแบบให้สามารถควบคุมการทำงานของขวดยาได้ 64 ขวด โดยการทำงานของแต่ละขวดตั้งแต่เริ่มต้นเรียกหาข้อมูลยาและส่งค่าเริ่มต้นในการนับจำนวนเม็ดยา จะใช้เวลาในการทำงานประมาณ 0.05 วินาที จากนั้นเวลาที่เหลือทั้งหมดจะอยู่ในส่วนของกลไกการจัดเรียงเม็ดยาให้ไหลลงจากขวดที่ละหนึ่งเม็ด เพื่อที่จะสามารถนับจำนวนเม็ดยาได้ โดยเวลาที่ใช้ในส่วนนี้แสดงในตารางที่ 5.1

อัตราการหมุน motor drive drug รอบ/นาที	จำนวนเม็ดยา 10 เม็ด ใช้เวลา (นาที)	อัตราการหมุน motor drive drug รอบ/นาที	จำนวนเม็ดยา 10 เม็ด ใช้เวลา (นาที)
1875	1.30 1.13 1.10 1.75 1.15	2142	1.75 2.14 2.03 1.96 2.40

ตารางที่ 5.1 แสดงผลการนับจำนวนเม็ดยา 10 เม็ด ด้วยอัตราการหมุนมอเตอร์ขับเม็ดยาต่างๆ

โดยลักษณะของเม็ดยาเป็นรูปทรงรี โดยมีขนาด ยาว 0.5cm กว้าง 1.0cm สูง 0.5cm และลักษณะของรูที่ให้เม็ดยาหล่นลงเป็นรูปวงกลม โดยมีเส้นผ่านศูนย์กลาง 1.5 cm

บทที่ 6

สรุปผลการทดสอบการใช้งานของโครงงานพิเศษ และ ข้อเสนอแนะ

จากการทดสอบการใช้งานของระบบเครื่องจ่ายยาอัตโนมัติควบคุมด้วยไมโครคอมพิวเตอร์ ได้ผลการทดสอบดังนี้คือ เวลาที่ใช้ในการเลือกขวดยาแต่ละขวดประมาณ 0.05 วินาที และเวลาในการนับจำนวนเม็ดยา 10 เม็ด กับการหมุนมอเตอร์ขับเม็ดยา 1875 รอบ/นาทีและ 2142 รอบ/นาทีโดยเฉลี่ยเป็น 1.29 นาที และ 2.06 นาที ตามลำดับ

จากผลการทดสอบการใช้งานสรุปได้ว่าในการจ่ายยาจำนวน 10 เม็ด โดยใช้ระบบของโครงงานพิเศษนี้จะใช้เวลาในการจ่ายยาประมาณ 1.34 - 2.11 นาที เมื่อเทียบกับการจ่ยยาโดยคน ในการจ่ายยา 10 เม็ดจะใช้เวลา 0.5 นาที ดังนั้นในการจ่ายยาโดยใช้ระบบตามโครงงานพิเศษนี้จะช้ากว่าการจ่ายยาโดยใช้คน ปัญหาการล่าช้านี้เกิดจากกลไกการจ่ายยาซึ่งในการออกแบบที่ผ่านมาได้คำนึงถึงการที่จะให้เม็ดยาไหลออกมาอย่างไม่ติดขัดและการนับเม็ดยาจะไม่สามารถกระทำได้อย่างถูกต้องถ้ามีเม็ดยาไหลออกมาจากขวดพร้อมกันครั้งละหลายๆเม็ด

ข้อเสนอแนะในการปรับปรุงและพัฒนาโครงงานพิเศษนี้ มีดังนี้คือ

ในส่วนของซอฟแวร์ ควรปรับปรุงโปรแกรมการใช้งานในส่วนที่เกี่ยวข้องกับผู้ใช้(เภสัชกร) ให้สะดวกต่อการใช้เพื่อลดความซับซ้อน

ในส่วนของ Prototype Board ถ้าเพิ่มเติมให้มีเอาท์พุทพอร์ทจำนวนมากขึ้นก็สามารถนำไปควบคุมการจ่ายยาในจำนวนมากตามต้องการได้

ในส่วนของกลไกการจ่ายยา เป็นปัญหาที่สำคัญที่จะต้องมีการพัฒนาและแก้ไข สำหรับระบบที่ทำนี้สามารถแก้ไขทำให้มียาหล่นเพียงครั้งละ 1 เม็ด แต่ก็ยังติดปัญหาของช่วงเวลาการจ่ายยาที่ใช้เวลามาก ถ้ามีการควบคุมให้อัตราการหมุนของมอเตอร์ที่ใช้เลื่อนตำแหน่งของช่องปล่อยเม็ดยา ให้มีอัตราที่ช้าลง การไหลของเม็ดยาควรจะมีอัตราที่สูงขึ้น แต่เนื่องจากมอเตอร์ที่ใช้ในระบบเป็น DC มอเตอร์ เมื่อลดแรงดันจะลดอัตราการหมุนของมอเตอร์ให้มีกำลังน้อยลง ในโอกาสต่อไปถ้ามีการปรับปรุงไปใช้เซอร์โวมอเตอร์ ซึ่งสามารถหมุนในอัตราที่ช้าโดยมีกำลังสูง ก็จะได้ทราบระบบนี้มีความเหมาะสมหรือไม่ และจะมีระบบอื่นใดที่เหมาะสมกว่า

ภาคผนวก ก. โปรแกรมควบคุมการทำงานของระบบ

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

struct address {
   char name[30];
   char number[4];
   char code[4];
   struct address *next;   /* pointer to next entry */
   struct address *prior;  /* pointer to next entry */
} list_entry;

struct address *start;  /* pointer to first entry in list */
struct address *last;   /* pointer to last entry */
struct address *find(char *);

void tomenu(void), enter(void), search(void), save(void);
void load(void), list(void);
void delete(struct address **, struct address **);
void dls_store(struct address *i, struct address **start,
       struct address **last);
void inputs(char *, char *, int), display(struct address *);
int menu_select(void);
```

```
void main(void)
{
   start = last = NULL;  /* initialize top and bottom pointers */
   for(;;)   {
      clrscr();
      switch(menu_select())   {
 case 1: enter();
    break;
 case 2: delete(&start, &last);
    break;
 case 3: list();
    break;
 case 4: search();
    break;
 case 5: save();    /* save list to disk */
    break;
 case 6: load();    /* read from disk */
    break;
 case 7: exit(0);
      }
   }
}

/* Select an operation. */
menu_select(void)
{
   char s[80];
   int c;
```

```
    printf("1. Enter a name\n");
    printf("2. Delete a name\n");
    printf("3. List the file\n");
    printf("4. Search\n");
    printf("5. Save the file\n");
    printf("6. Load the file\n");
    printf("7. Quit\n");
    do   {
       printf("\nEnter your choice: ");
       gets(s);
       c = atoi(s);
    } while(c<0 || c>7);
    return c;
}

void tomenu(void)
{
    printf("pass anykey to continue");
    for(;;) {
       if(kbhit()) return;
    }
}

/* Enter names and address. */
void enter(void)
{
    struct address *info;
```

```
    for(;;)   {
        info = (struct address *)malloc(sizeof(list_entry));
        if(!info)   {
  printf("\nout of memory");
  return;
        }
        inputs("enter name: ", info->name,30);
        if(!info->name[0]) break;  /* stop entering */
        inputs("enter intilial number : ", info->number,4);
        inputs("enter code: ", info->code,4);
        dls store(info, &start, &last);
    } /* entry loop */
}

/* This function will input a string up to the length in count
   and will prevent the string from being overrun.
*/
void inputs(char *promt, char *s, int count)
{
   char p[255];

   do   {
      printf(promt);
      gets(p);
      if(strlen(p)>count) printf("\ntoo long\n");
   } while(strlen(p)>count);
   strcpy(s, p);
```

```
}

/* Create a doubly linked list in sorted order.
*/
void dls_store(
   struct address *i,    /* new element */
   struct address **start, /* first element in list */
   struct address **last /* last element in list */
)
{
   struct address *old, *p;

   if(*last==NULL)   { /* first element in list */
      i->next = NULL;
      i->prior = NULL;
      *last = i;
      *start = i;
      return;
   }
   p = *start;  /* start at top of list */
   old = NULL;
   while(p)   {
      if(strcmp(p->name, i->name)<0)   {
old = p;
p = p->next;
      }
      else   {
if(p->prior)   {
```

```
            p->prior->next = i;
            i->next = p;
            i->prior = p->prior;
            p->prior = i;
            return;
        }
        i->next = p; /* new first element */
        i->prior = NULL;
        p->prior = i;
        *start = i;
        return;
            }
        }
        old->next = i; /* put on end */
        i->next = NULL;
        i->prior = old;
        *last = i;
}

/* Remove an element from the list. */
void delete(struct address **start, struct address **last)
{
    struct address *info, *find();
    char s[80];

    printf("enter name: ");
    gets(s);
    info = find(s);
```

```
    if(info)   {
        if(*start==info)   {
  *start=info->next;
  if(*start) (*start)->prior = NULL;
  else *last = NULL;
        }
        else   {
  info->prior->next = info->next;
  if(info!=*last)
     info->next->prior = info->prior;
  else
     *last = info->prior;
        }
        free(info);   /* return memory to system */
        tomenu();
    }
}

/* Find an address. */
struct address *find( char *name)
{
   struct address *info;

   info = start;
   while(info)   {
      if(!strcmp(name, info->name)) return info;
      info = info->next; /* get next address */
   }
```

```
    printf("name not found\n");
    return NULL;  /* not found */
}

/* Display the entire list. */
void list(void)
{
    struct address *info;
    const char cc="c";

    info = start;
    while(info)  {
        display(info);
        info = info->next; /* get next address */
        if(cc==getch()) continue;
    }
    printf("\n");
}

/* This function actually prints the fields in each address. */
void display(struct address *info)
{
    printf("%s\n", info->name);
    printf("%s\n", info->number);
    printf("%s\n", info->code);
    printf("\n");
}
```

```
/* Lood for a name in the list. */
void search(void)
{
   char name[40];
   struct address *info, *find();

   printf("enter name to find: ");
   gets(name);
   info = find(name);
   if(!info) printf("not found\n");
   else display(info);
   tomenu();
}

/* Save the file to disk. */
void save(void)
{
   struct address *info;
   FILE *fp;

   fp = fopen("data.xxx", "w+");
   if(!fp)   {
      printf("cannot open file\n");
      exit(1);
   }
   printf("\nsaving file\n");
   info = start;
   while(info)   {
```

```
        fwrite(info, sizeof(struct address), 1, fp);
        info = info->next; /* get next address */
    }
    fclose(fp);
}

/* Load the address file. */
void load()
{
    struct address *info;
    FILE *fp;

    fp = fopen("data.xxx", "r+");
    if(!fp)  {
        printf("cannot open file\n");
        exit(1);
    }

    /* free any previously allocated memory */
    while(start)  {
        info = start->next;
        free(info);
        start = info;
    }

    /* reset top and bottom pointers */
    start = last = NULL;
    printf("\nloading file\n");
```

```
    while(!feof(fp))    {
        info = (struct address *) malloc(sizeof(struct address));
        if(!info)   {
printf("out of memory");
return;
        }
        if(1!=fread(info, sizeof(struct address), 1, fp)) break;
        dls_store(info, &start, &last);
    }
    fclose(fp);
}
```

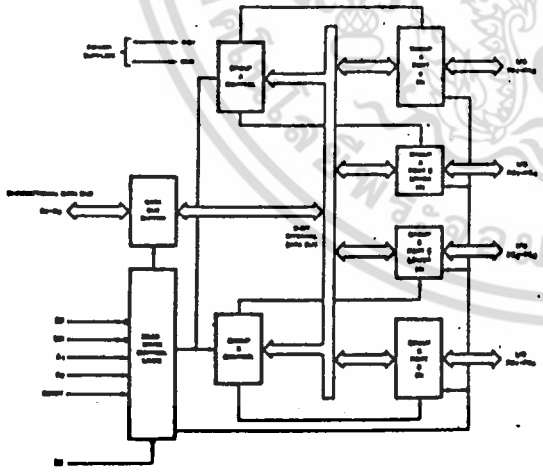
ภาคผนวก ข. Data Sheet

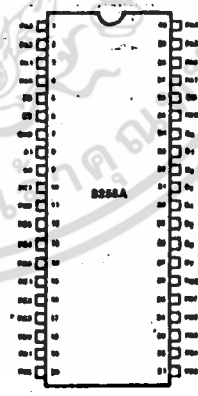
intel

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- Reduces System Package Count
- Improved DC Driving Capability
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

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





8255A FUNCTIONAL DESCRIPTION

General

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

Data Bus Buffer

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

Read/Write and Control Logic

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

($\overline{CS}$)

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

($\overline{RD}$)

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

($\overline{WR}$)

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

(A_0 and A_1)

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

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS → 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS → 3-STATE

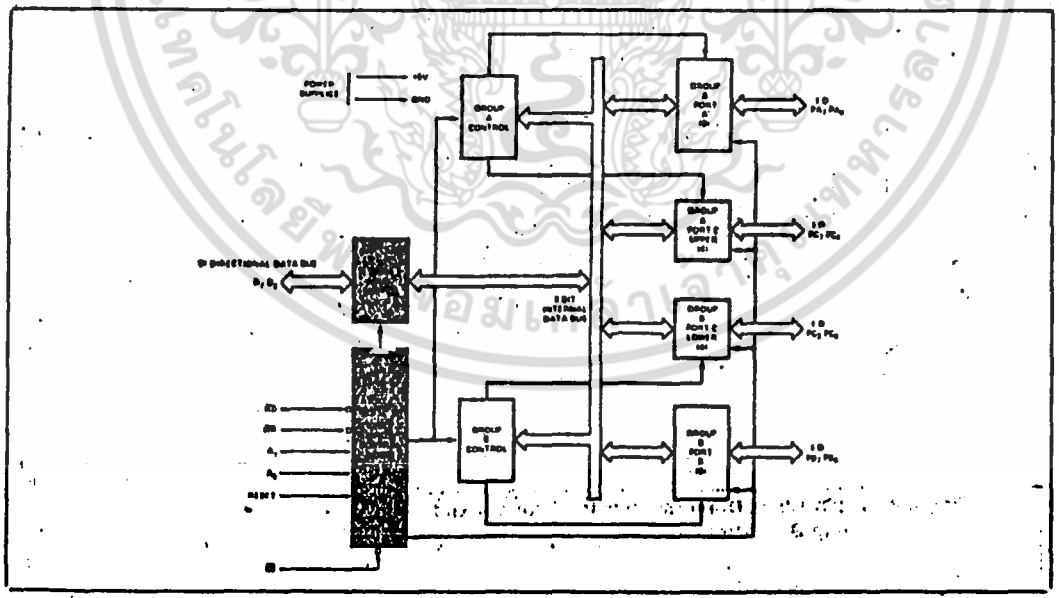


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Fupctions

(RESET)

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

Group A and Group B Controls

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

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

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

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

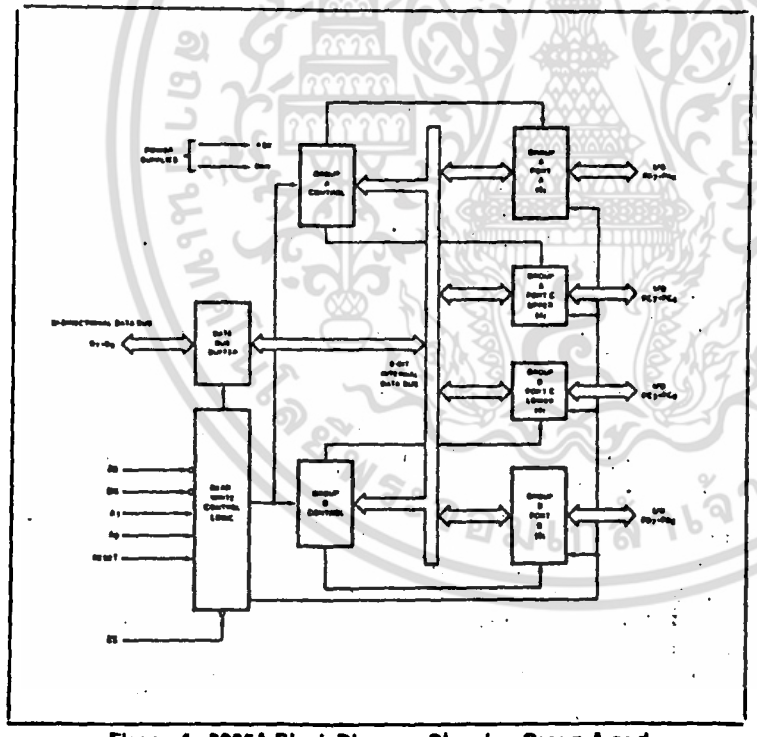
Ports A, B, and C

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

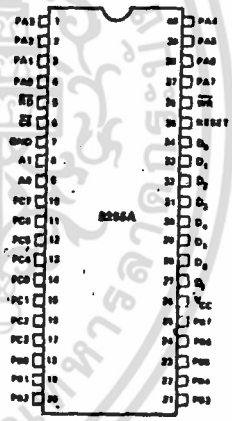
Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

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

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



PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

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

8255A OPERATIONAL DESCRIPTION

Mode Selection

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

Mode 0 – Basic Input/Output
Mode 1 – Strobed Input/Output
Mode 2 – Bi-Directional Bus

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

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

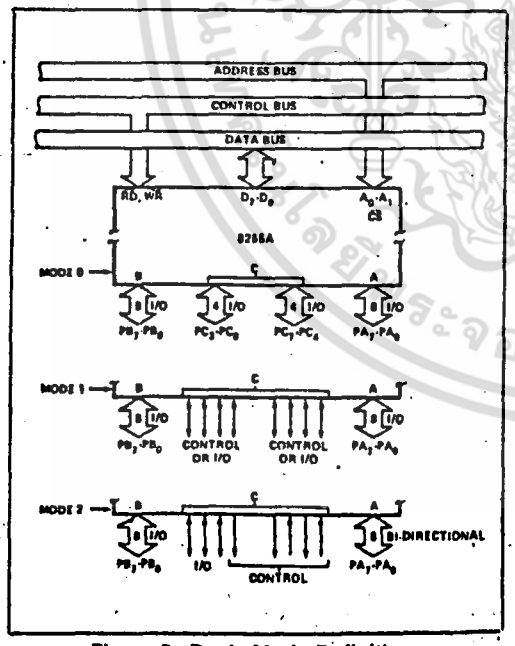


Figure 5. Basic Mode Definitions and Bus Interface

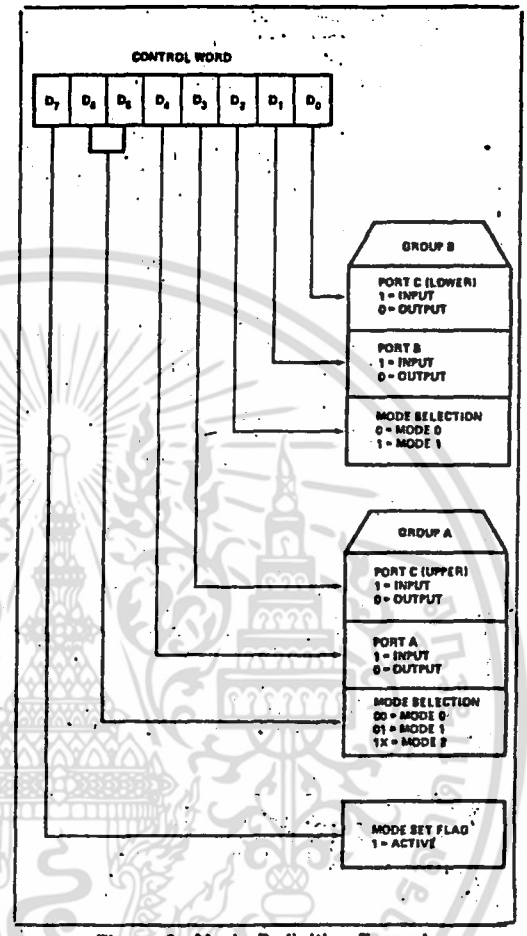


Figure 6. Mode Definition Format

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

Single Bit Set/Reset Feature

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

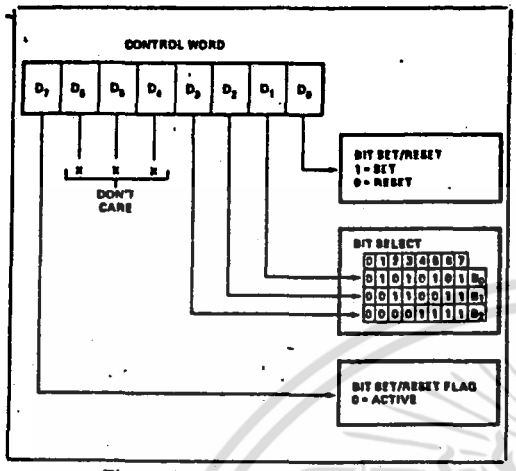


Figure 7. Bit Set/Reset Format

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

Interrupt Control Functions

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

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

INTE flip-flop definition:

(BIT-SET) – INTE is SET – Interrupt enable
(BIT-RESET) – INTE is RESET – Interrupt disable

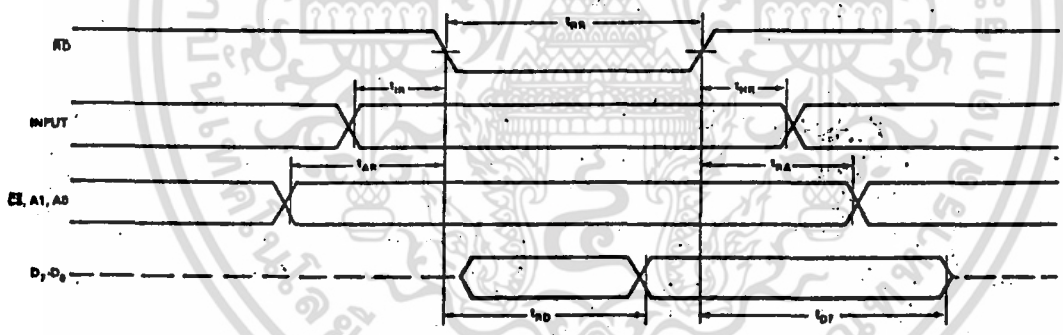
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

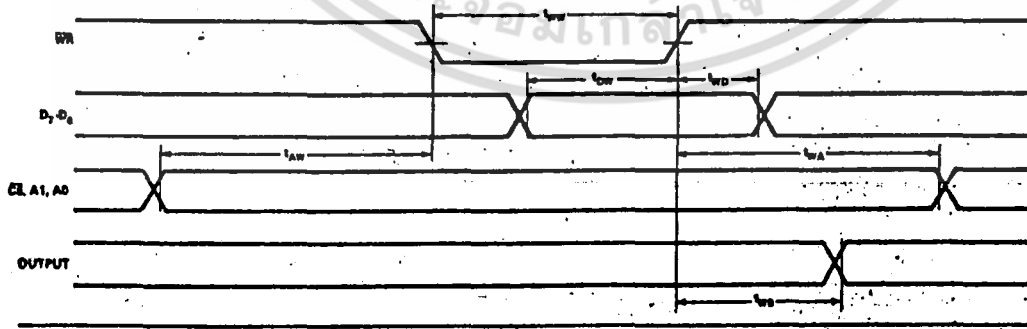
MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

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



MODE 0 (Basic Input)

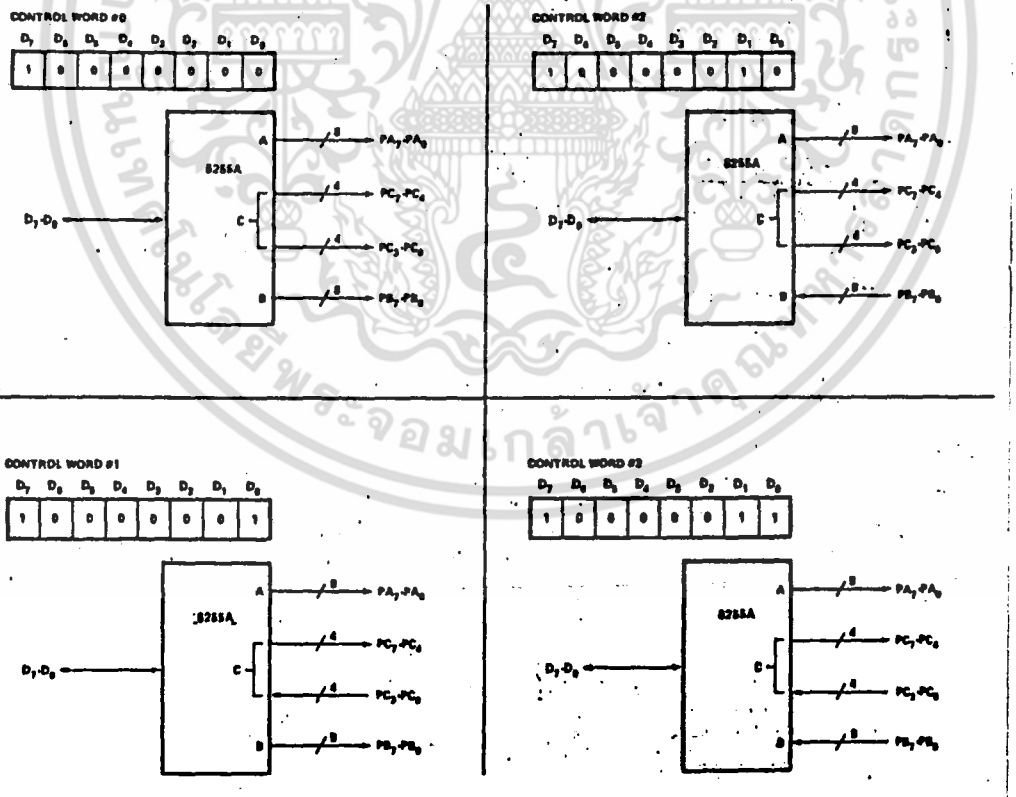


MODE 0 (Basic Output)

MODE 0 Port Definition

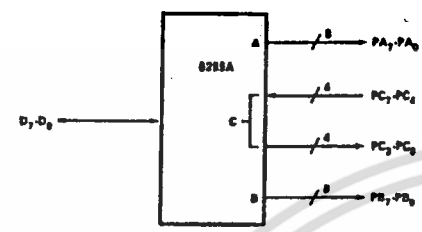
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations



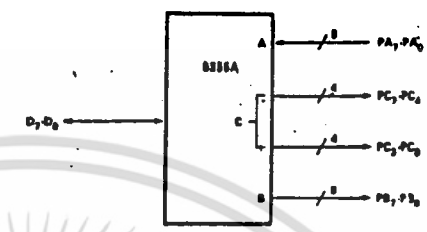
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



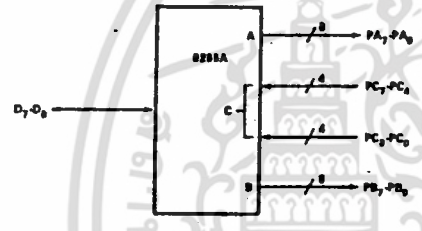
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



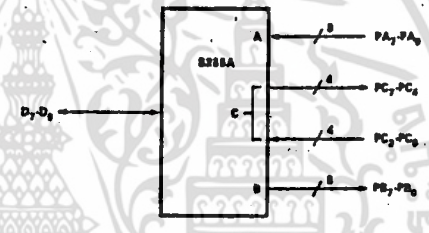
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



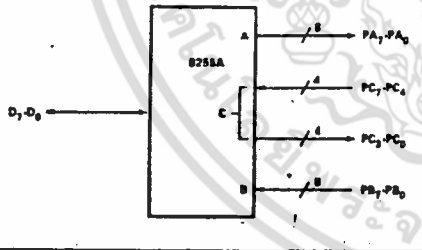
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0

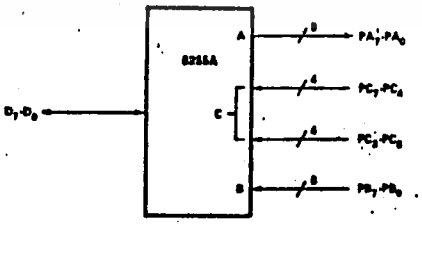


CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0

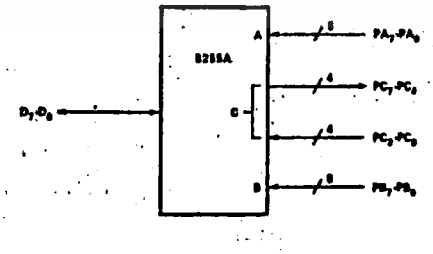
CONTROL WORD #7

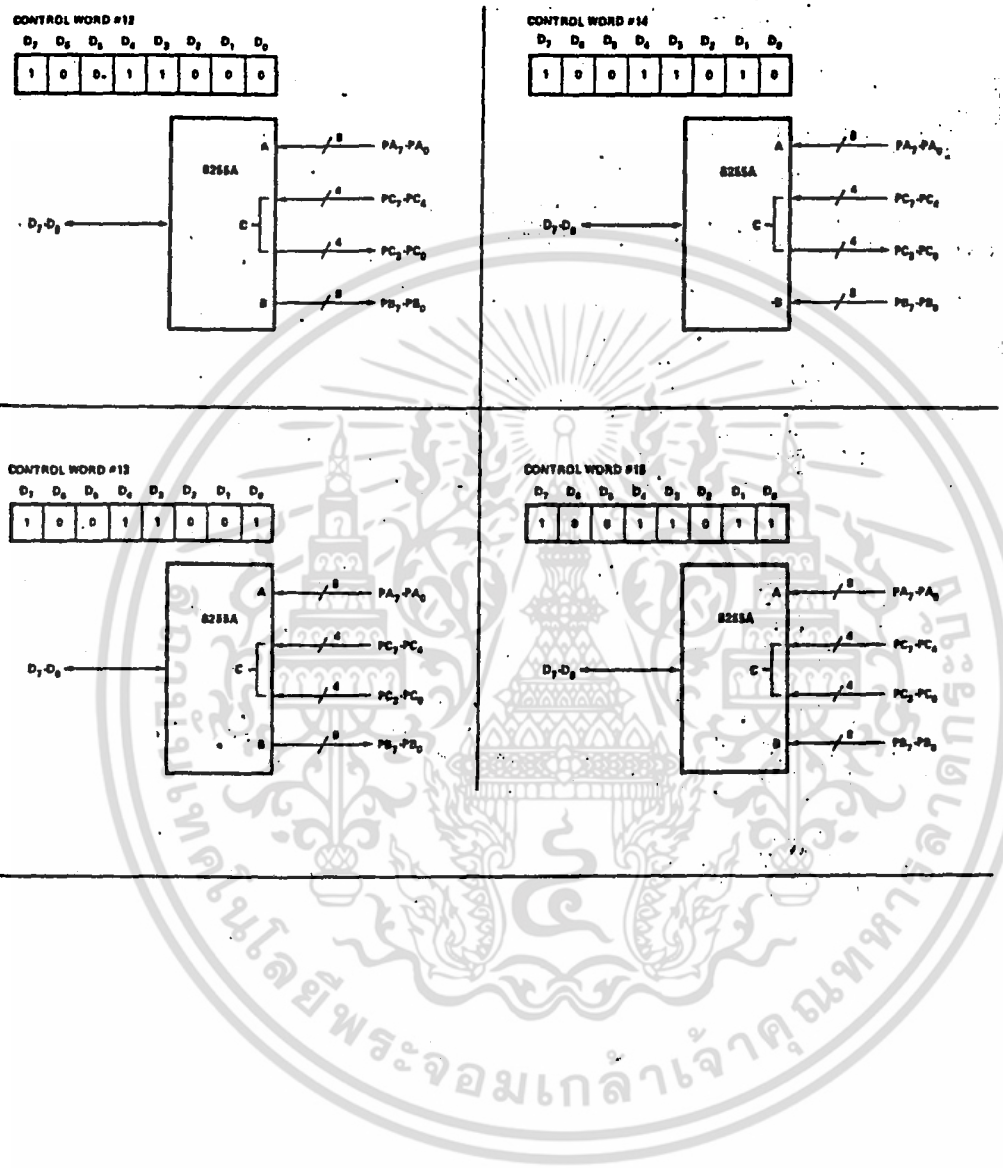
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1





8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2.6 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

The Intel® 8253 is a programmable counter/timer device designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2.6 MHz. All modes of operation are software programmable.

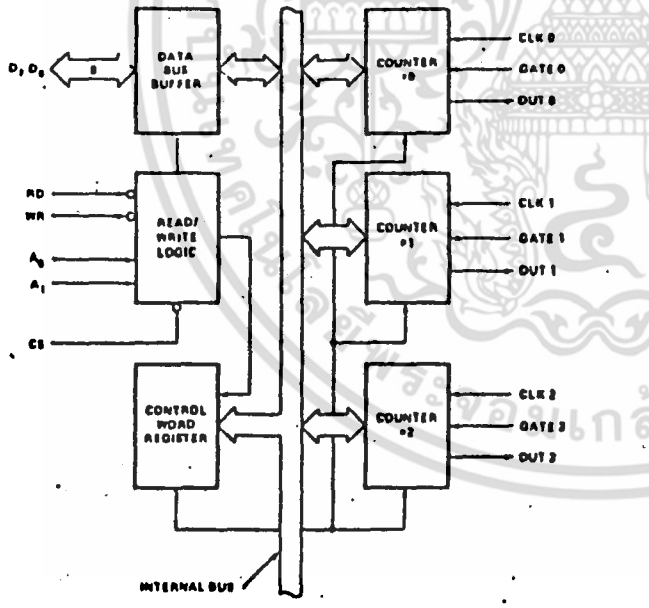


Figure 1. Block Diagram

Figure 2. Pin Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5 V to +7 V
Power Dissipation ..... 1 Watt

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ±10%) *

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}+.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to .45V
I_{CC}	V_{CC} Supply Current		140	mA	

CAPACITANCE (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%, GND = 0V) *

Bus Parameters (Note 3)

READ CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AR}	Address Stable Before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay From $\overline{READ}$[4]		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time Between READ and Any Other Control Signal	1		1		μs

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AW}	Address Stable Before $\overline{WR}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{WR}$	30		30		ns
t_{WW}	$\overline{WR}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{WR}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{WR}$	40		30		ns
t_{RV}	Recovery Time Between $\overline{WR}$ and Any Other Control Signal	1		1		μs

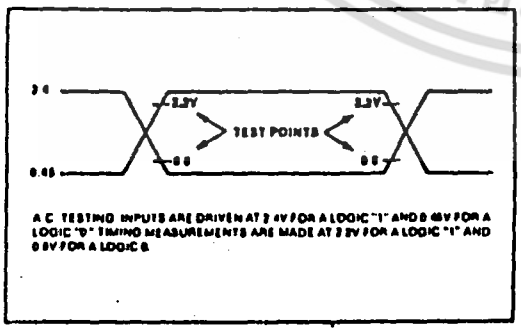
CLOCK AND GATE TIMING

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK↑	100		100		ns
t_{GH}	Gate Hold Time After CLK↑	50		50		ns
t_{OD}	Output Delay From CLK↓ [4]		400		400	ns
t_{ODG}	Output Delay From Gate↓ [4]		300		300	ns

NOTES:
1. I_{OL} = 2.2 mA.
2. I_{OH} = −400 μA.
3. AC timings measured at V_{OH} 2.2, V_{OL} = 0.8.
4. C_L = 150pF.

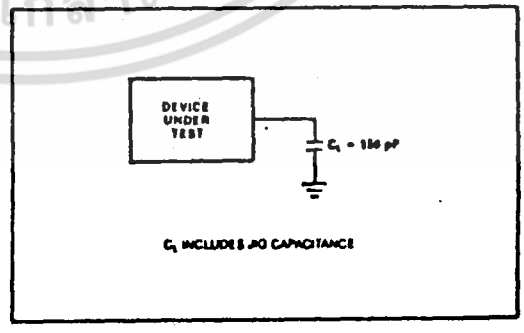
* For Extended Temperature EXPRESS, use M8253 electrical parameters.

A.C. TESTING INPUT, OUTPUT WAVEFORM



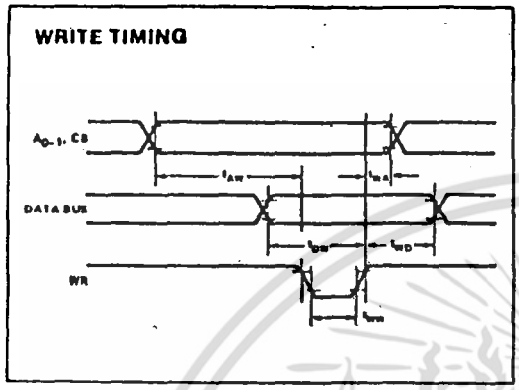
A.C. TESTING INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". TIMING MEASUREMENTS ARE MADE AT 2.2V FOR A LOGIC "1" AND 0.8V FOR A LOGIC 0.

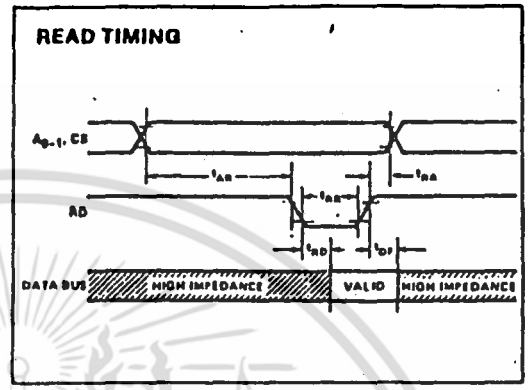
A.C. TESTING LOAD CIRCUIT



C_L INCLUDES JIG CAPACITANCE

WAVEFORMS





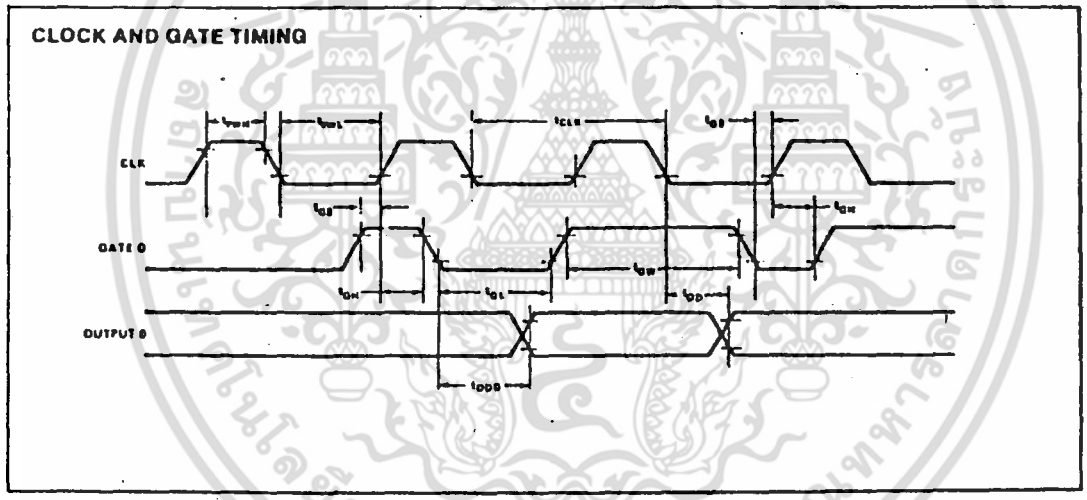


Photo Detectors
Transistor Output

MRD300
MRD310

PHOTO DETECTORS
TRANSISTOR OUTPUT
NPN SILICON

. . designed for application in industrial inspection, processing and control, counters, . . sorters, switching and logic circuits or any design requiring radiation sensitivity, and stable characteristics.

- Popular TO-18 Type Package for Easy Handling and Mounting
- Sensitive Throughout Visible and Near Infrared Spectral Range for Wider Application
- Minimum Light Current 4 mA at H = 5 mW/cm$_2$ (MRD300)
- External Base for Added Control
- Annular Passivated Structure for Stability and Reliability

CASE 82-05
METAL

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
Collector-Emitter Voltage	V_{CEO}	50	Volts
Emitter-Collector Voltage	V_{ECO}	7	Volts
Collector-Base Voltage	V_{CBO}	80	Volts
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.27	mW mW/°C
Operating Temperature Range	T_A	−65 to +125	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C

STATIC ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Collector Dark Current (V_{CE} = 20 V, H = 0) T_A = 25°C	I_{CEO}	—	5	25	nA
T_A = 100°C		—	4		μA
Collector-Base Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CBO}$	80	120	—	Volts
Collector-Emitter Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CEO}$	50	85	—	Volts
Emitter-Collector Breakdown Voltage (I_E = 100 μA)	$V_{(BR)ECO}$	7	8.5	—	Volts

OPTICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Light Current (V_{CC} = 20 V, R_L = 10 Ohms) Note 1	MRD300	I_L	4	7	— mA
	MRD310		1	3.5	—
Light Current (V_{CC} = 20 V, R_L = 100 Ohms) Note 2	MRD300	I_L	—	2.5	— mA
	MRD310		—	0.8	—
Photo Current Rise Time (Note 3) (R_L = 100 Ohms, I_L = 1 mA peak)		t_r		2	2.5 μs
Photo Current Fall Time (Note 3) (R_L = 100 Ohms, I_L = 1 mA peak)		t_f	—	2.5	4 μs

NOTES 1. Radiation flux density (H) equal to 5 mW/cm^2 emitted from a tungsten source at a color temperature of 2870 K.
2. Radiation flux density (H) equal to 0.5 mW/cm^2 (pulsed) from a GaAs (gallium arsenide) source at λ = 940 nm.
3. For unsaturated response time measurements, radiation is provided by pulsed GaAs (gallium arsenide) light-emitting diode (λ = 940 nm) with a pulse width equal to or greater than 10 microseconds (see Figure 2) I_L = 1 mA peak.

TYPICAL CHARACTERISTICS

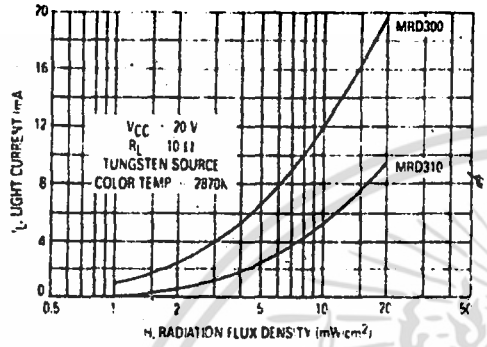


Figure 1. Light Current versus Irradiance

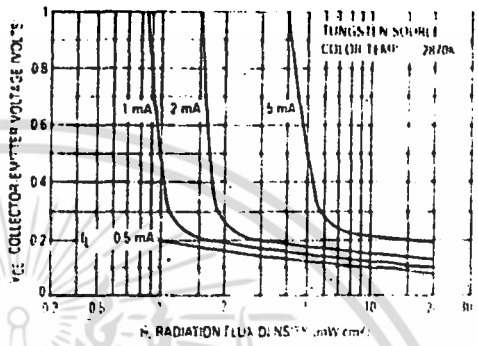


Figure 2. Collector-Emitter Saturation Characteristic

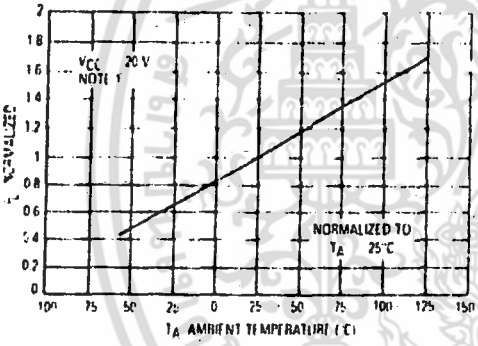


Figure 3. Normalized Light Current versus Temperature

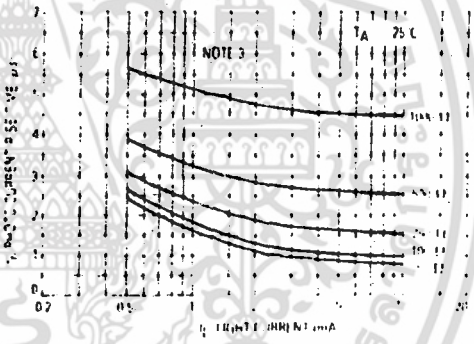


Figure 4. Rise Time versus Light Current

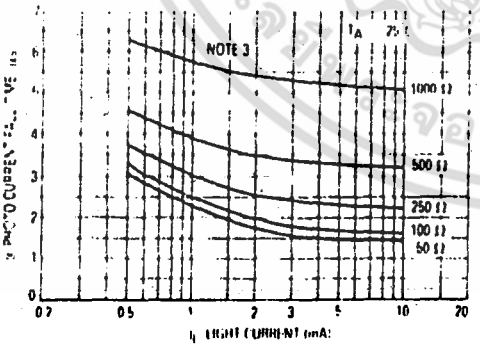


Figure 5. Fall Time versus Light Current

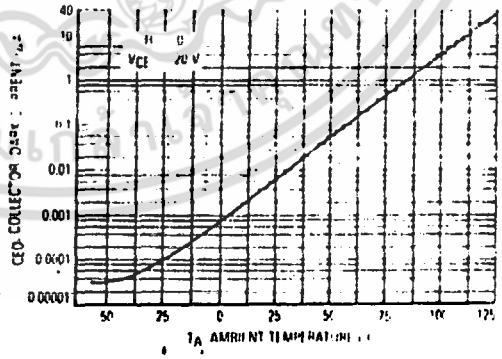


Figure 6. Dark Current versus Temperature

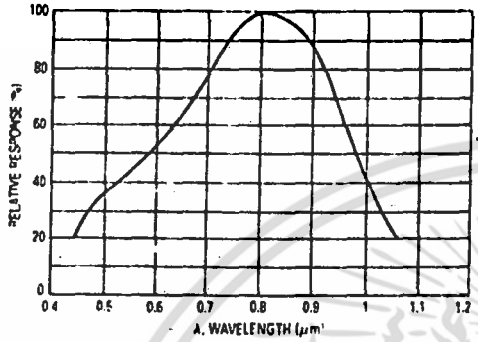


Figure 7. Constant Energy Spectral Response

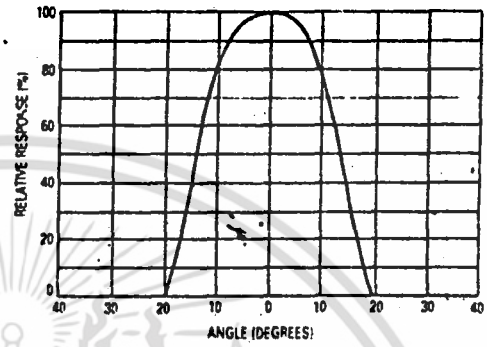


Figure 8. Angular Response

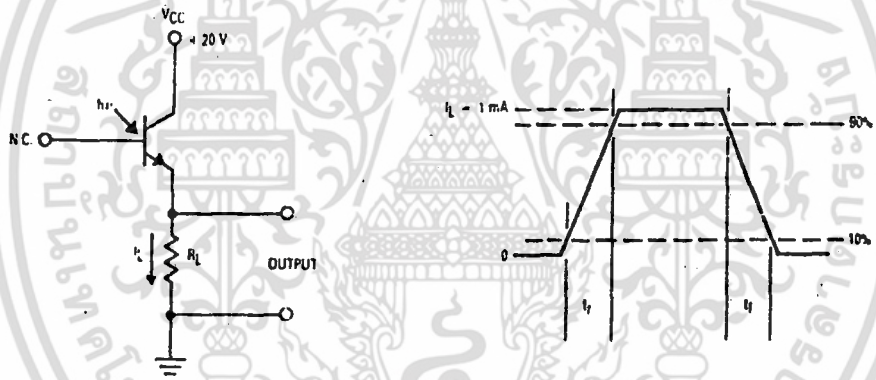
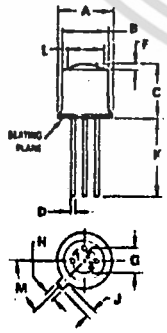


Figure 9. Pulse Response Test Circuit and Waveform

OUTLINE DIMENSIONS



NOTES
1. LEADS WITHIN .13 mm (.005) RADIUS OF TRUE POSITION AT SEATING PLANE, AT MAXIMUM MATERIAL CONDITION.
2. PIN 3 INTERNALLY CONNECTED TO CASE.

STYLE 1.
PIN 1. EMITTER
2. BASE
3. COLLECTOR

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	5.31	5.84	0.209	0.230
B	4.52	4.95	0.178	0.195
C	6.07	6.60	0.185	0.255
D	0.41	0.48	0.016	0.019
F	—	1.14	—	0.045
G	2.54 BSC		0.100 BSC	
H	0.71	1.17	0.028	0.046
J	0.84	1.22	0.033	0.048
K	12.70	—	0.500	—
L	3.35	4.01	0.132	0.158
M	45° BSC		45° BSC	

CASE 82-05
METAL

ประวัติผู้เขียน

นาย สยาม ดีอุดมวงศา เกิดวันที่ 10 ธันวาคม พ.ศ. 2510

การศึกษามัธยมต้นและมัธยมปลาย โรงเรียน เซนต์จอห์น วิทยาลัย

การศึกษาระดับอุดมศึกษา สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง คณะวิทยาศาสตร์ สาขาฟิสิกส์ประยุกต์ และสำเร็จการศึกษาในปีการศึกษา 2533