การจัดเก็บสินค้าด้วยคอมพิวเตอร์
COMPUTERIZE WAREHOUSE CONTROL III

โดย

นายนริศ เชียวผ่อง

นายนิพนธ์ วัฒนาพูนชัย

นายพยุงศักดิ์ ใจคำ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษา
หลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้า
เจ้าคุณทหาร ลาดกระบัง
ปีการศึกษา 2535

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

032551

การจัดเก็บสินค้าด้วยคอมพิวเตอร์

Computerize warehouse control III

นาย นริศ เขียวผ่อง 34161212

นาย นิพนธ์ วัฒนาพูนชัย 34161213

นาย พยุงศักดิ์ ใจค่า 34161220

อาจารย์ที่ปรึกษา

อาจารย์ ภากร หุตะสังกาศ

Advisor

Assoc.Pakorn Hutasungkas

ปริญญานิพนธ์สำหรับปริญญาอุตสาหกรรมศาสตร์บัณฑิตย์

สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

032551

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจัดเก็บสินค้าด้วยคอมพิวเตอร์

Computerize warehouse control III

ผู้จัดทำ

1.	นายนริศ	เขียวผ่อง	34161212
2.	นายนิพนธ์	วัฒนาพูนชัย	34161213
3.	นายพยุงศักดิ์	ใจคำ	34161220

.....อาจารย์ที่ปรึกษา

(อาจารย์ ภากร หุตะสังกาศ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของเนื้อหาการประยุกต์ระบบไมโครคอนโทรลเลอร์ (Microcontrollor Application) ซึ่งนำเอาไมโครคอนโทรลเลอร์มาประยุกต์ใช้กับการควบคุมคลังสินค้าอัตโนมัติ โดยจะเป็นการนำสินค้าเข้า-ออกจากคลังสินค้า ซึ่งจะช่วยให้งานด้านการจัดเก็บ และ เคลื่อนย้ายสินค้า เป็นไปอย่างมีระบบประกอบกับการนำเอาระบบการจัดการฐานข้อมูล ที่มีประสิทธิภาพมาช่วยจัดสินค้าที่ตำแหน่งต่างๆ โดยสินค้าประเภทที่มีการเคลื่อนย้ายบ่อย จะถูกนำมาวาง ณ.ตำแหน่งที่ใกล้ทางเข้าหรือออกเพื่อประหยัดเวลาและพลังงานในการเคลื่อนย้ายสินค้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Abstract

This thesis is a fraction of microcontroller application. By use the microcontriller to control automatic warehouse model. The operation is to keep or get the product in/out from warehouse model. For make an advance and intelligence it will be use with a database which intelligence. The database know the product which frequency move in/out from the warehouse. Then this product will keep in a nearly position. From this it will save time and energy to move the product .

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทคัดย่อ	
ABSTRACT	
บทที่ 1. บทนำ	1-1
บทที่ 2. โครงสร้างของ WAREHOUSE MODEL.	2-1
2.1 การเรียกตำแหน่งของ MOTOR	2-1
2.2 การเรียกตำแหน่งของ LIMIT SWITCH	2-2
บทที่ 3. ลักษณะทางด้าน HARDWARE และ การออกแบบ	3-1
3.1 ไมโครโปรเซสเซอร์	3-1
3.2 การเชื่อมต่อระหว่าง ANT-32 กับวงจร DRIVE MOTOR และ WAREHOUSE LIMIT SW.	3-7
3.3 การเชื่อมต่อกับ IBM PC	3-16
3.4 ภาคจ่ายไฟ	3-17
3.5 แผนผังการเชื่อมต่อ	3-18
บทที่ 4. การออกแบบทาง Software	4-1
โปรแกรมการทำงาน	4-18
บทที่ 5. สรุป	5-1
5.1 สรุปผล	5-1
5.2 ปัญหาที่เกิดขึ้น และ ข้อเสนอแนะ	5-1
กิตติกรรมประกาศ	
หนังสืออ้างอิง	
ภาคผนวก	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

การพัฒนาเทคโนโลยีทางด้านไมโครคอมพิวเตอร์ในปัจจุบัน เป็นไปอย่างกว้างขวาง จะเห็นได้จากการนำเอาระบบไมโครคอมพิวเตอร์ไปใช้ในงานด้านต่างๆ เช่น งานด้านประมวลผลข้อมูลในทางธุรกิจ งานด้านการออกแบบทางสถาปัตยกรรมและศิลปกรรม งานทางด้านการค้นคว้าและวิจัย งานเกี่ยวกับการศึกษา และโดยเฉพาะอย่างยิ่งในด้านอุตสาหกรรม ไมโครคอมพิวเตอร์ได้เข้ามามีบทบาทอย่างมากในการควบคุมเครื่องจักรกล ให้สามารถทำงานแทนกำลังงานคน เพื่อผลิตผลิตภัณฑ์และบริการงานด้านต่างๆ ให้มีคุณภาพและปริมาณเพียงพอกับความต้องการของมนุษย์อย่างมากมาย และการพัฒนาของระบบไมโครคอมพิวเตอร์ยังคงมีต่อไปอย่างไม่หยุดยั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วัตถุประสงค์ของปริญญานิพนธ์

ตามที่ได้กล่าวมาแล้วว่าไมโครคอมพิวเตอร์ นับวันจะมีความสำคัญต่อระบบงานในด้านอุตสาหกรรม ดังนั้นปริญญานิพนธ์ฉบับนี้จึงเป็นการนำเอาไมโครคอมพิวเตอร์สำหรับงานควบคุมขนาดเล็กหรือที่เรียกว่า ไมโครคอนโทรลเลอร์ (MicroController) ในตระกูล MCS - 51 ซึ่งผลิตโดยบริษัท INTEL มาประยุกต์ใช้งานในระบบควบคุมสำหรับงานทางด้านคลังสินค้า หรือ ระบบการจัดเก็บสินค้าด้วยคอมพิวเตอร์ (Computerized Warehouse Control) ซึ่งมีความสำคัญเป็นอย่างยิ่งในระบบงานทางด้านอุตสาหกรรมในบ้านเราซึ่งกำลังพัฒนาไปอย่างมากซึ่งการนำระบบการจัดเก็บสินค้าด้วยเครื่องคอมพิวเตอร์มาใช้ในงานจะทำให้เกิดความสะดวกรวดเร็ว ลดต้นทุนในการผลิต

นอกจากนี้การพัฒนาระบบไมโครคอมพิวเตอร์เพื่องานควบคุมด้านอุตสาหกรรม และการพัฒนาระบบไมโครคอนโทรลเลอร์ เพื่องานควบคุมขนาดเล็กนี้ก็เป็นสิ่งที่น่าสนใจ เพราะจะทำให้สามารถเข้าใจกับพื้นฐานของระบบควบคุมได้มากขึ้นและเมื่อนำระบบควบคุมขนาดเล็กมาทำงานควบคู่กับระบบคอมพิวเตอร์เพื่อการจัดการเกี่ยวกับฐานข้อมูล ก็สามารถจะได้ระบบงานที่ใหญ่ขึ้น เพื่อนำไปใช้ในงานด้านอุตสาหกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขอบเขตของปริญญานิพนธ์

ในการทำปริญญานิพนธ์นี้เนื้อหาส่วนใหญ่จะเป็นการประยุกต์การใช้งาน Board MicroController ANT-32 ซึ่งเป็น Micro Controller ตระกูล MCS-51เพื่อใช้ในการControl Warehouse แต่เนื่องจากในโครงงานจริงจะเป็นการนำ ANT-32 ไปประยุกต์ใช้ร่วมกับ Micro Computer IBM-PC โดยที่ IBM-PC จะใช้เพื่อจัดเก็บระบบฐานข้อมูลของคลังสินค้าและ ANT-32 จะติดต่อสื่อสารกับ IBM-PC ผ่านทางช่องสื่อสารอนุกรม RS-232 โดยแบ่งงานออกเป็น 3 กลุ่มคือ

1.กลุ่มที่พัฒนาระบบฐานข้อมูลบน IBM-PC

2.กลุ่มพัฒนา CONVEYER และ ROBOT ARM

3.กลุ่มที่พัฒนาการควบคุม Warehouse โดยใช้ ANT-32

ดังนั้นขอบเขตของปริญญานิพนธ์ฉบับนี้จะเป็นการควบคุม Warehouse Model ซึ่งประกอบด้วยวงจรควบคุม MOTOR และ การเขียนโปรแกรมควบคุมการนำสินค้าเข้า-ออกจาก Warehouse ซึ่งรับเอาข้อมลความต้องการของผู้ใช้มาจาก IBM PC

โครงงานนี้เป็นการพัฒนาต่อเนื่องจากเทอมที่แล้ว โดยการเปลี่ยนบอร์ดคอนโทรลจาก JASS-31 และมีการทำงานเป็นแบบ Manual ไม่มีการ Link กับคอมพิวเตอร์เพื่อจัดทำ DATA BASE มาเป็น ANT-32 ซึ่งสามารถสื่อสารข้อมูลในการเคลื่อนย้ายสินค้าเข้าออก มีการจัดทำฐานข้อมูล ทำให้ทราบว่ามีสินค้าตัวใดบ้างที่ถูกเคลื่อนย้าย เข้า-ออก ความบ่อยครั้งในการเคลื่อนย้ายสินค้าตัวนั้นๆ ซึ่งสามารถทำให้ทราบว่ามีสินค้าตัวใดบ้างที่ควรจะสั่งเพิ่ม และมีสินค้าตัวใดบ้างที่ยังไม่ควรสั่ง ข้อมูลทั้งหลายดังที่กล่าวมานี้สามารถเรียกดูได้จาก คอมพิวเตอร์

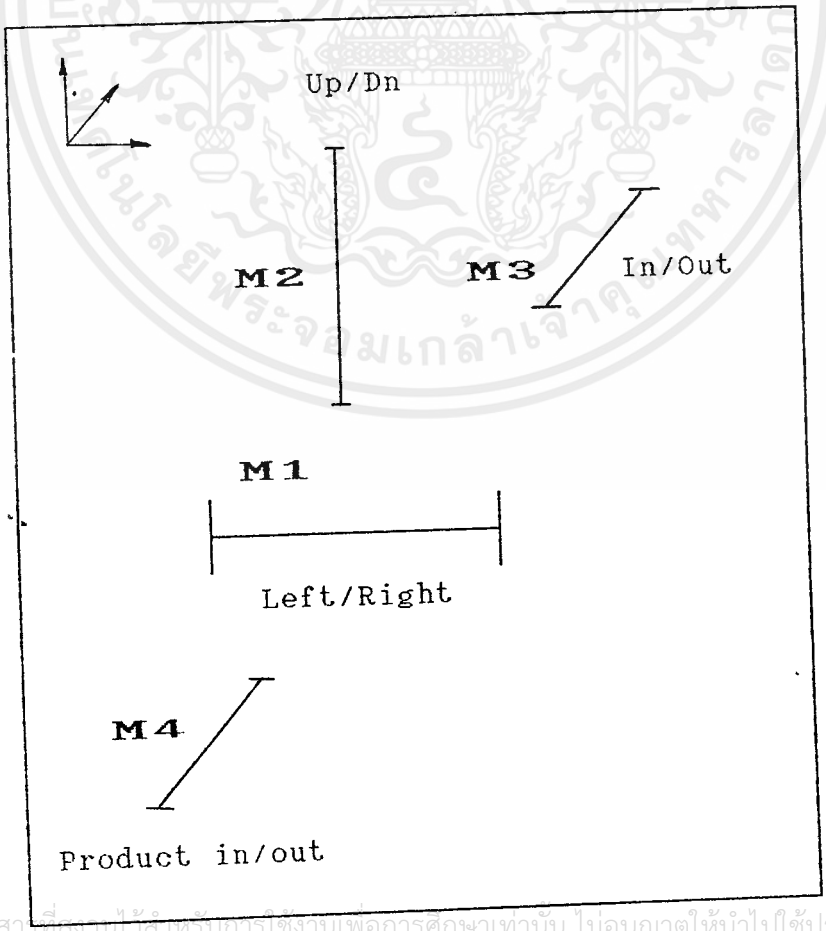
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

โครงสร้างของ WAREHOUSE MODEL

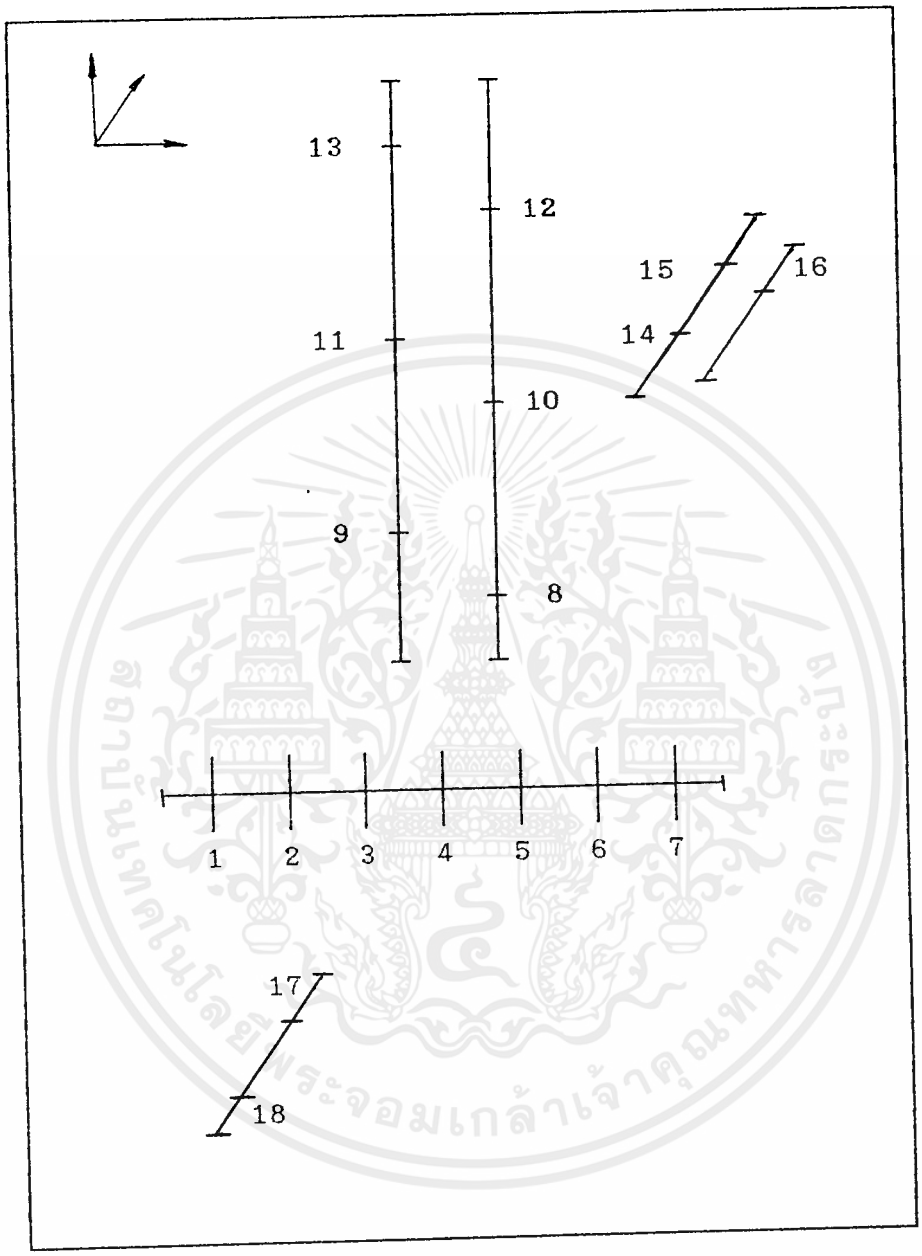
ลักษณะการจัดเก็บสินค้าของ WAREHOUSE จะมีลักษณะเป็นชั้น ๆ จำนวน 3 ชั้น โดยในแต่ละชั้นสามารภจัดเก็บได้จำนวน 7 ช่อง ผลิตจากวัสดุประเภท พีวีซี ซึ่งลักษณะการนำสินค้าเข้า หรือออกจากชั้นนี้จะเป็นในลักษณะลิฟท์ ขึ้น - ลง เคลื่อนที่เข้า - ออก และสามารถที่จะเลื่อนไปทางซ้ายและขวาได้ โดยการกลับทิศทางการหมุนของ Motor ซึ่งจะมีอยู่ 3 ตัว ในการกระทำแต่ละหน้าที่ Motor ที่ใช้เป็น DC.Motor แบบ Universal ใช้แรงดัน 8 โวลท์ ซึ่งการจะทำให้ MOTOR หมุนนี้จะขึ้นอยู่กับคำสั่งทางด้าน SOFTWAREเพื่อมา Control ซึ่งในการ Control นี้ เราจะต้องมี Sensor Switch (ใช้แบบปกติเปิด) ของแต่ละช่องเก็บเพื่อส่งสัญญาณ feedback control ผ่าน Port input มายัง CPU. เพื่อประมวลผลว่าควรจะหยุดที่ช่องเก็บช่องใดเพื่อการนำสิ่งของเข้าหรือออกจาก Warehouse ได้อย่างถูกต้อง

2.1 การเรียกตำแหน่งต่างๆ ของ Motor แสดงดังรูป

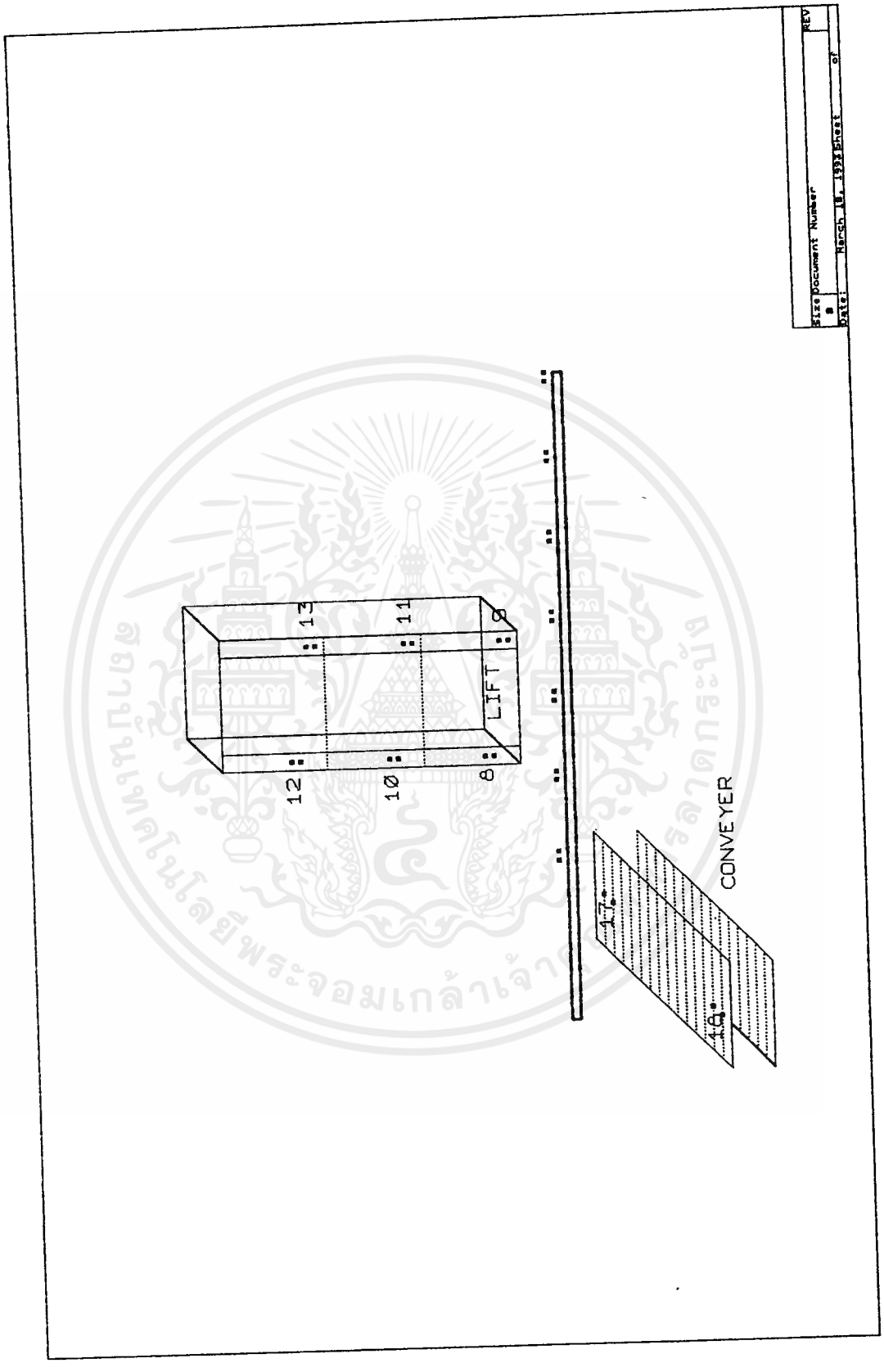


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2การเรียกตำแหน่งของ Limit switch



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

ลักษณะทางฮาร์ดแวร์(Hardware)

3.1 ไมโครโปรเซสเซอร์(Microprocessor)

ในการออกแบบการควบคุมขนาดเล็ก โดยทั่วๆไป สิ่งสำคัญที่ต้องพิจารณาเป็นอันดับแรกก็คือตัวไมโครโปรเซสเซอร์หรือซีพียู(CPU) ซึ่งจะทำหน้าที่เป็นตัวประมวลผล ของระบบควบคุม ไมโครโปรเซสเซอร์ที่มีอยู่ในท้องตลาดมีอยู่ด้วยกันหลายเบอร์ ซึ่งแต่ละเบอร์จะมีคุณสมบัติแตกต่างกันออกไป และเหมาะสมกับงานในแต่ละประเภท และความถนัดของผู้ใช้งาน สำหรับไมโครโปรเซสเซอร์ที่เลือกใช้นี้เป็นไมโครโปรเซสเซอร์ที่เหมาะสมกับงานควบคุมขนาดเล็ก หรือที่เรียกว่าไมโครคอนโทรลเลอร์(Mircrocontroller)ในตระกูล MCS-51 ซึ่งผลิตโดยบริษัท INTEL สำหรับเบอร์ที่เลือกใช้ คือเบอร์ 8032 AH

เพื่อเป็นการลดขั้นตอนการออกแบบในส่วนของตัวไมโครคอนโทรลเลอร์ ซึ่งการออกแบบบอร์ดไมโครคอนโทรลเลอร์นี้ มิใช่จุดประสงค์ของงานวิจัยนี้ แต่ได้มุ่งเน้นในการนำบอร์ดไมโครคอนโทรลเลอร์ไปประยุกต์ใช้งาน จึงได้เลือกใช้บอร์ดไมโครคอนโทรลเลอร์สำเร็จรูป ซึ่งปัจจุบันมีจำหน่ายอยู่หลายรุ่นให้เลือกใช้ตามความเหมาะสม ซึ่งงานวิจัยนี้ได้เลือกใช้บอร์ด ANT-32 EMBEDDED CONTROLLER BOARD ซึ่งเป็นผลงานผลิตและจำหน่ายโดย บริษัท ศิลารีเสิร์ช จำกัด ในบอร์ดดังกล่าวประกอบไปด้วยส่วนประกอบหลักๆ คือ ซีพียู(CPU),หน่วยความจำรอม (ROM) ,หน่วยความจำแรม (RAM) ,อินพุท/เอาท์พุท พอร์ต(INPUT/OUTPUT PORT) ,แอดเดรสดีโคดเดอร์(ADDRESS DECODER) ฯลฯ สำหรับรายละเอียด และ วงจรของบอร์ด ANT-32 ดูได้จาก ภาคผนวก ในการพิจารณาและออกแบบทางฮาร์ดแวร์ เพื่อการคอนโทรลแวร์เฮาส์ (Control Warehouse) จะพิจารณาเพื่อการเชื่อมต่อกับบอร์ด ANT-32 นี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โครงสร้างของ ANT-32

- ANT-32 คือ บอร์ดไมโครคอนโทรลเลอร์ ผลิตโดยบริษัท ศิลารีเสิร์ช ผลิตขึ้นโดยใช้ CPU ตระกูล MCS-51 คือ เบอร์ 8032

คุณสมบัติของบอร์ด

- ใช้กับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 (8031/8032)
- ความถี่สัญญาณนาฬิกา 11.0592 เมกะเฮิรตซ์
- หน่วยความจำอีพรอม 8 ถึง 32 กิโลไบต์ สำหรับโปรแกรม
- หน่วยความจำแรม 8 ถึง 32 กิโลไบต์ สำหรับเก็บข้อมูลสามารถแบคอัพข้อมูลได้โดยใช้แบตเตอรี่ลิเธี่ยม
- ใช้กับหน่วยความจำชนิด EEPROM ขนาด 8 ถึง 32 กิโลไบต์ ได้
- มีพอร์ตเบอร์ 8255 จำนวน 2 พอร์ต สำหรับการต่อไปใช้งานภายนอก
- มีวงจรพอร์ตอนุกรม (RS-232) สำหรับการต่อเข้ากับเครื่องไมโครคอมพิวเตอร์
- มีขั้วต่อสำหรับ PORT 1 ของไมโครคอนโทรลเลอร์โดยเฉพาะ
- มีขั้วต่อ SYSTEM BUS ทำให้ขยายระบบได้ง่าย และสามารถใช้กับบอร์ดขยายต่างๆ ที่จะมีขึ้นในอนาคต
- สามารถเลือกเบอร์หน่วยความจำ หรือกำหนดคุณสมบัติต่าง ๆ ของบอร์ดได้ด้วยจัมเปอร์
- ขนาดของบอร์ด 10.2 * 14.2 เซนติเมตร

การทำงาน

วงจรสมบูรณ์ของบอร์ด ANT-32 แสดงดังรูป 3.1 ประกอบไปด้วยไมโครคอนโทรลเลอร์ 8032 , หน่วยความจำ (ROM/RAM) , พอร์ตอินพุต/เอาต์พุต , ส่วนถอดรหัสแอดเดรสของหน่วยความจำ , พอร์ต , แบตเตอรี่สำรองสำหรับ RTC , RAM และ ส่วนแปลงแรงดันสำหรับพอร์ตอนุกรม RS-232

IC_1 เป็นชิพไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ของ INTEL เบอร์ 8032 ทำงานที่ความถี่ 11.0592 เมกะเฮิร์ตซ์ ทำการติดต่อกับหน่วยความจำ 3 ส่วนคือ IC_2 เป็นหน่วยความจำภายนอกที่ใช้กับโปรแกรม (external program memory) ใช้กับอีพรอมขนาด 8-32 กิโลไบต์ เบอร์ 2764 , 27128 หรือ 27256 โดยใช้จัมเปอร์ JP_3 ส่วนที่สอง คือ IC_3 เป็นหน่วยความจำที่ใช้เก็บข้อมูล(data memory) ใช้กับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แรม 8 ถึง 32 กิโลไบต์ เบอร์ 6264 หรือ 62256 เลือกขนาดด้วยจัมเปอร์ JP_4 ซึ่ง IC_3 นี้มีแบตเตอรี่แบคอัพสำหรับเก็บข้อมูลเมื่อไฟดับด้วย และส่วนที่สามคือ IC_4 เป็นหน่วยความจำที่ใช้ทั้งเก็บโปรแกรมและเก็บข้อมูล (program and data memory) ใช้กับอีพรอม หรือ EEPROM และแรมขนาด 8 ถึง 32 กิโลไบต์ โดยใช้อีพรอมเบอร์ 2764, 27256 หรือ EEPROM เบอร์ 2864(A) , 28256(A) และแรมเบอร์ 6264 , 62256 ซึ่งเลือกชนิดและขนาดของ IC_4 ด้วยจัมเปอร์ JP_4 และ JP_6 ในกรณีที่ต้องการใช้หน่วยความจำภายในที่ต้องใช้กับโปรแกรม (internal program memory) ภายใน IC_1 สำหรับชิพไมโครคอนโทรลเลอร์เบอร์ 8051 , 8052 , 8751 และ 8752 สามารถเลือกได้โดยจัมพ์เปอร์ JP_1

ส่วนของพอร์ตนอกจากพอร์ตภายใน (internal port) ของซีพียู คือ PORT 1 รวมทั้งขา INT 0 ,INT 1 ,T0 และ T1 แล้ว ยังมีพอร์ตภายนอกคือ IC_{11} และ IC_{12} เบอร์ 8255 ให้ใช้งานเพิ่มเติมอีกถึง 48 บิต

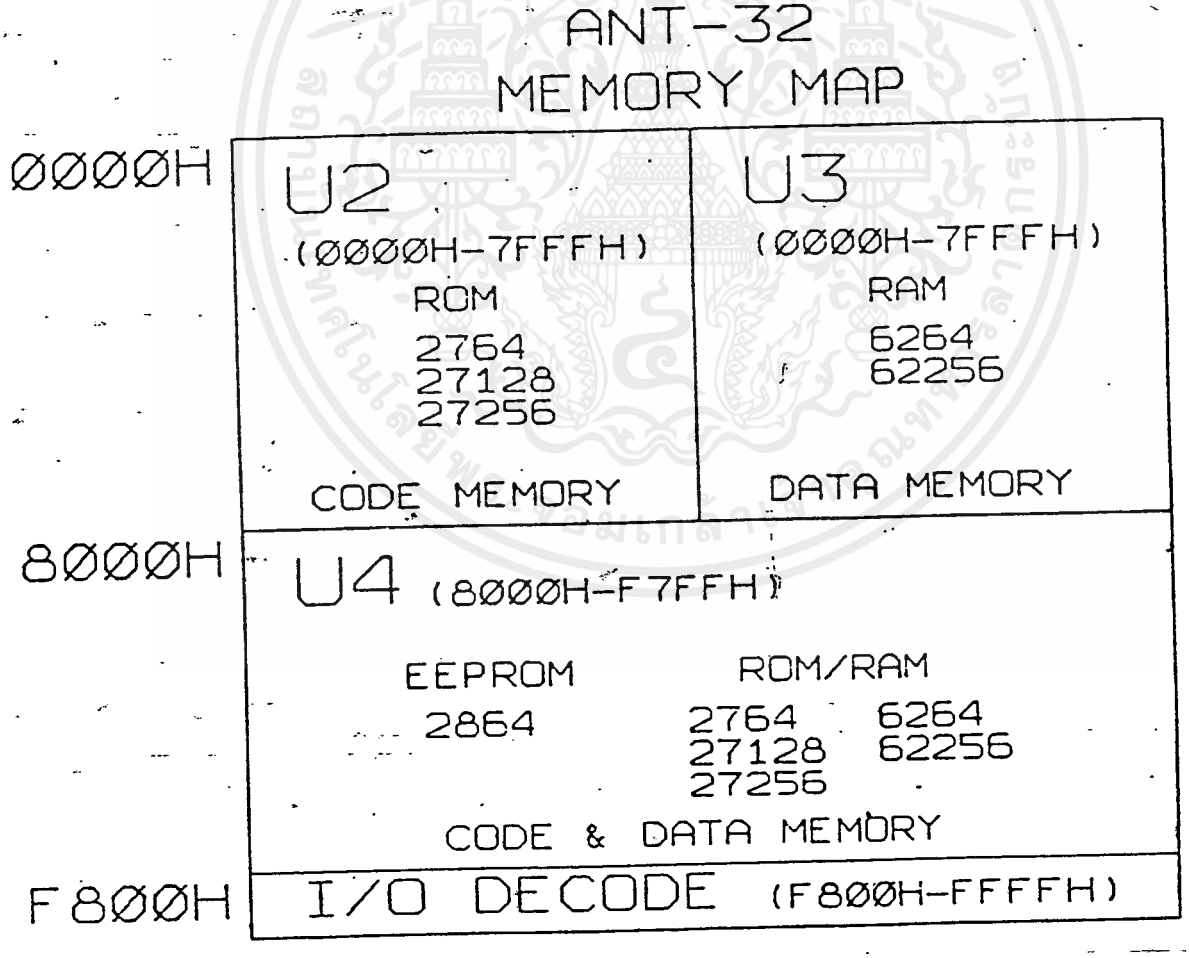
ส่วนถอดรหัสแอดเดรสของหน่วยความจำ สำหรับ IC_2 ใช้สายสัญญาณแอดเดรส A_{15} เพื่อกำหนดให้ IC_2 อยู่ที่แอดเดรส 000H-7FFFH ส่วน IC_3 และ IC_4 ใช้สายสัญญาณแอดเดรส A_{13} - A_{15} ร่วมกับ IC_7 จัดให้ IC_3 อยู่ที่แอดเดรส 0000H-7FFFH และ IC_4 อยู่ที่แอดเดรส 8000H-F7FFH ส่วนพอร์ต IC_{11} จัดให้อยู่ที่แอดเดรส F800H-F9FFH และ IC_{12} อยู่ที่แอดเดรส FC00H-FDFFH IC_7 นอกจากใช้ถอดรหัสแอดเดรสแล้ว ยังใช้สร้างสัญญาณอ่านข้อมูลสำหรับหน่วยความจำ IC_4 และบัฟเฟอร์สัญญาณรีเซ็ตซีพียูอีกด้วย สำหรับวงจรแบตเตอรี่สำรองข้อมูลใช้สำหรับเก็บข้อมูลของแรมIC_3 ในขณะไฟดับ (power fail) โดยใช้แบตเตอรี่ลิเธียม ซึ่งข้อดีของแบตเตอรี่ชนิดนี้คือ วัดวงจรง่ายไม่ต้องมีวงจรชาร์จ และมีอายุการใช้งานนานกว่าแบตเตอรี่ชนิดนิกเกิลแคดเมียมวงจรนี้สามารถสำรองทั้ง RTC และแรมได้นานนับปี การกำหนดสถานะการแบคอัพข้อมูลใน IC_2 กระทำได้ด้วยจัมพ์เปอร์ JP_7 นอกจากนั้นแล้ว IC_9 ยังมีวงจรตรวจจับแรงดัน V_{cc} เมื่อไฟตก วงจรในส่วนนี้จะถูกใช้สำหรับงานที่ต้องการเก็บค่าสถานะ หรือข้อมูลบางอย่างของระบบลงแรม ก่อนไฟดับแรงดันที่ตรวจจับกำหนดได้จากค่า R_5 และ R_6 และกำหนดให้อีนาเบิล หรือดีสเอเบิลได้ด้วยจัมพ์เปอร์ JP_9 ส่วนสัญญาณรีเซตซีพียูสามารถเลือกได้ว่าจะใช้วงจร R,C รีเซตหรือจาก IC_9 โดยจัมพ์เปอร์ JP_2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IC_{10} ทำหน้าที่ปรับระดับแรงดันสำหรับสัญญาณข้อมูลอนุกรมจากขา TXD, RXD ของ IC_1 ให้เป็นไปตามระดับแรงดันตามมาตรฐาน RS-232 ซึ่งจะใช้กับการสื่อสารรับส่งข้อมูลระหว่างบอร์ด ANT-32 กับเครื่องไมโครคอมพิวเตอร์

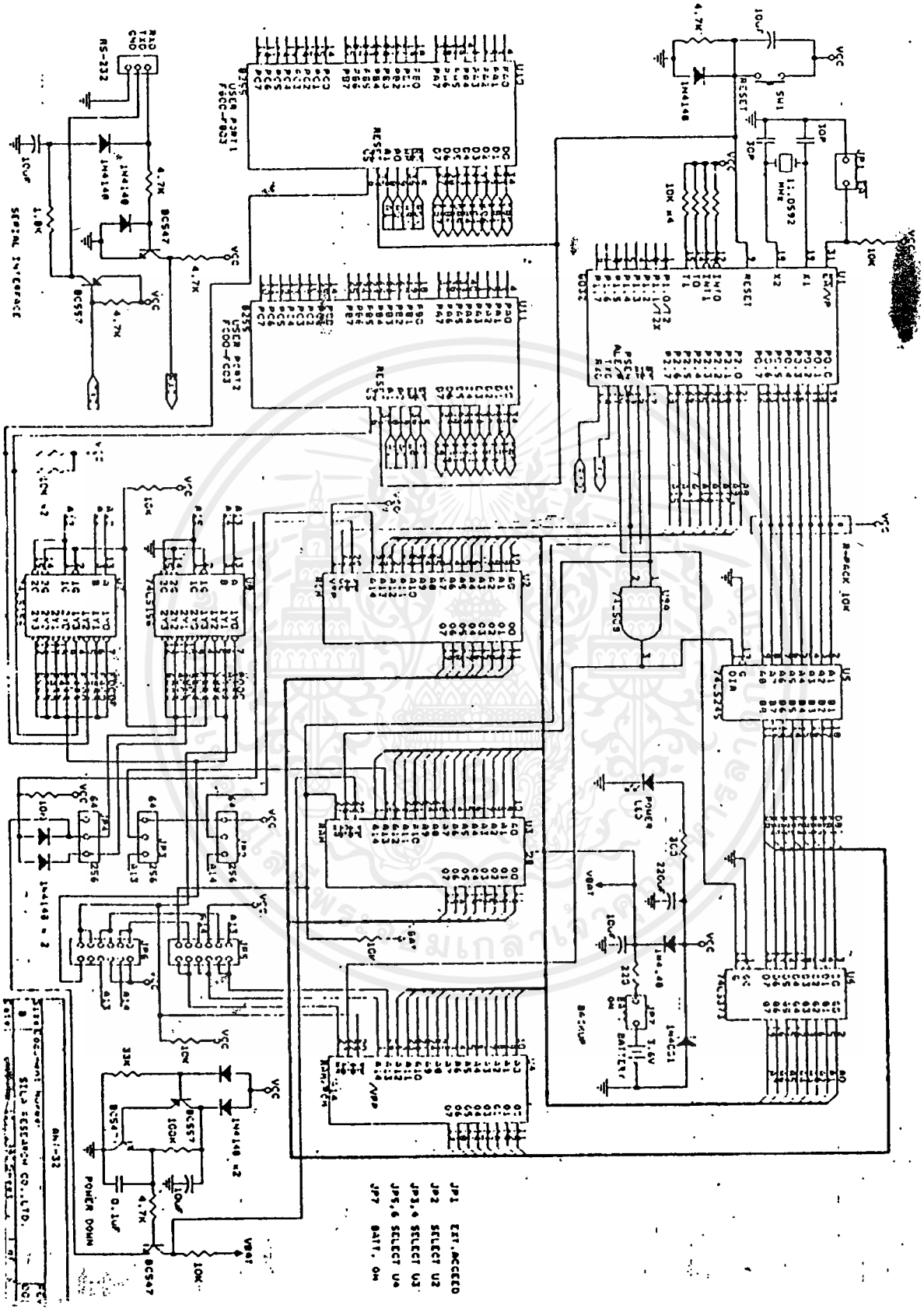
การจัดหน่วยความจำ

รูปที่ 3.2 แสดงถึงการจัดหน่วยความจำของบอร์ด ANT-32 ซึ่ง IC_2 เป็นส่วนของรอมมอนิเตอร์ และ IC_3 เป็นแรมแบคอัพ ทั้ง IC_2 และ IC_3 ถูกถอดระหัสที่แอดเดรสเดียวกันคือ 000H-7FFFH สำหรับ IC_4 สามารถใช้ได้ทั้งอีพรอม, EEPROM และแรมอยู่ในตำแหน่งแอดเดรส 800H-F7FFH และแอดเดรสช่วง F800H-FFFFH ถูกใช้สำหรับเป็นแอดเดรสของพอร์ต โดย IC_{11} อยู่ที่แอดเดรส F800H-F9FFH และ IC_{12} ที่แอดเดรส FC00H-FDFFH



รูปที่ 3.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ANT-32 SPECIFICATION

CPU	: 8032
CPU CLOCK	: 11.0502 MH$_z$
INTERNAL MEMORY	: 256 BYTE (RAM)
EXT.CODE MEMORY	: (U2) 8-32K SELECT 2764,27128,27256 (ROM)
EXT.DATA MEMORY	: (U3) 8-32K SELECT 6264,62256 (RAM)
EXT.CODE & DATA MEMORY	: (U4) 8-30K SELECT 2764,27128,27256 (ROM)
	: 2864 (EPROM)
	: 6264,62256 (RAM)
INTERNAL PORT	: 12 BIT I/O
EXTERNAL PORT	: USER1 8255 PORT I/O 24 BIT
	: USER2 8255 PORT I/O 24 BIT
BACKUP	: DATA MEMORY (U3) 52 HOURS
CHARGE TIME	: 48 HOURS
LANGUAGE	: MCS BASIC-52
	: ASSEMBLY (BY DOWNLOAD HEX FILE)
CONNECTOR	: 16P INTERNAL PORT
	: 26P USER1 PORT
	: 26P USER2 PORT
	: 2P POWER SUPPLY
	: 3P SERIAL INTERFACE (RS232)
SERIAL INTERFACE	: RS232
POWER	: 5V DC 290 mA (U4 NOT INCLUDE)
SIZE	: 5.25" * 3.9"
OPTION	: BATTERY NI-CAD 3.6V 60 mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การเชื่อมต่อระหว่าง ANT-32 กับวงจร DRIVE MOTOR และ WAREHOUSE LIMIT SWITCH

ในการเชื่อมต่อระหว่าง ANT-32 กับวงจร DRIVE MOTOR และ WAREHOUSE LIMIT SWITCH จะผ่านวงจรทางด้าน อินพุท/เอาท์พุท พอร์ต ของ บอร์ด ANT-32 ซึ่งใช้ IC#8255 PPi จำนวน 2 ตัว ผ่านออกมาทางด้าน DIP JUMPER (JP-1 และ JP-2)

3.2.1 การจัด INPUT/OUTPUT PORT สำหรับการเชื่อมต่อกับ WAREHOUSE

-การเชื่อมต่อกับ INPUT PORT กับ WAREHOUSE เป็นการนำเอาสถานะของ LIMIT SWITCH ของ WAREHOUSE ซึ่งเป็นแบบปกติปิด (NORMALLY CLOSED) มาเป็นตัวบ่งบอกถึงตำแหน่งพิกัดต่างๆ ของ WAREHOUSE ซึ่งSWITCH นี้จะเปลี่ยนสถานะเมื่อมีการสัมผัสจากอุปกรณ์ทางด้านฮาร์ดแวร์ และจะส่งผลไปยัง PORT INPUT เพื่อนำผลที่ได้นี้ไปป้อนให้กับ ซีพียู(CPU) ทำการประมวลผลเพื่อทำงานตามเงื่อนไขที่ได้วางไว้ต่อไป

-การเชื่อมต่อ OUTPUT PORT กับ WAREHOUSE เป็นการทำงานเกี่ยวกับการเลื่อนตำแหน่งต่างๆ ซึ่งใช้มอเตอร์เป็นเครื่องจักรในการทำงาน ดังนั้นจึงต้องผ่านวงจร DRIVE MOTOR ซึ่งสามารถที่จะกลับทิศทางการหมุนของมอเตอร์ เพื่อใช้ในการเลื่อนตำแหน่งได้โดยการ คอนโทรลยังบิต(BIT)ของ OUTPUT PORT ซึ่งลักษณะของวงจรDRIVE MOTOR แสดงได้ดังรูป 3.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

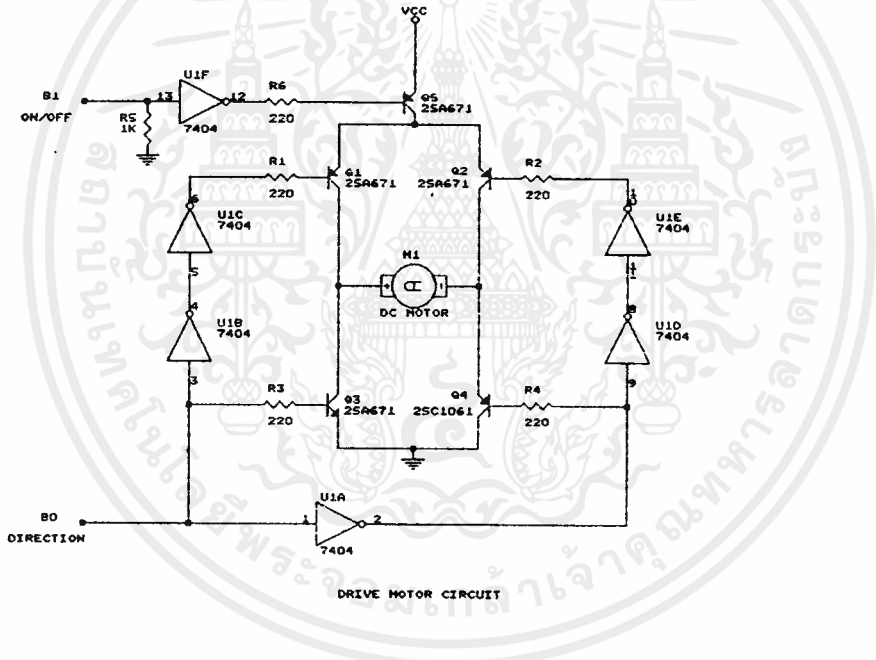
วงจรควบคุมทิศทางการหมุนของมอเตอร์

การทำงานของวงจร

เมื่อ B_o = "0" และ B_1 = "1" จะมีผลทำให้ทรานซิสเตอร์ Q_1,Q_4 และ Q_5 นำกระแส ส่วน Q_2,Q_3 จะไม่นำกระแส จึงมีผลทำให้ Motor หมุนไปในทิศทางตามเข็มนาฬิกา

เมื่อ B_o = "1" และ B_1 = "1" จะมีผลทำให้ทรานซิสเตอร์ Q_2,Q_3 และ Q_5 นำกระแส ส่วน Q_1,Q_4 จะไม่นำกระแส จึงมีผลทำให้ Motor หมุนไปในทิศทางทวนเข็มนาฬิกา

แต่เมื่อ B_1="0" ทรานซิสเตอร์ Q_5 จะไม่นำกระแสจึงทำให้ไม่มีกระแสไหลผ่าน Motor จึงทำให้ Motor หยุดหมุน



รูปที่ 3.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 ลักษณะการควบคุมการหมุนของมอเตอร์ เพื่อการเลื่อนตำแหน่ง

-การเลื่อน ซ้าย-ขวา ใช้มอเตอร์ M1 ในการเลื่อน โดยมี USER1 PB0 เป็นบิต (BIT) การควบคุมทิศทาง และ PB1 เป็นบิต(BIT)ควบคุมการหมุน/หยุดหมุน

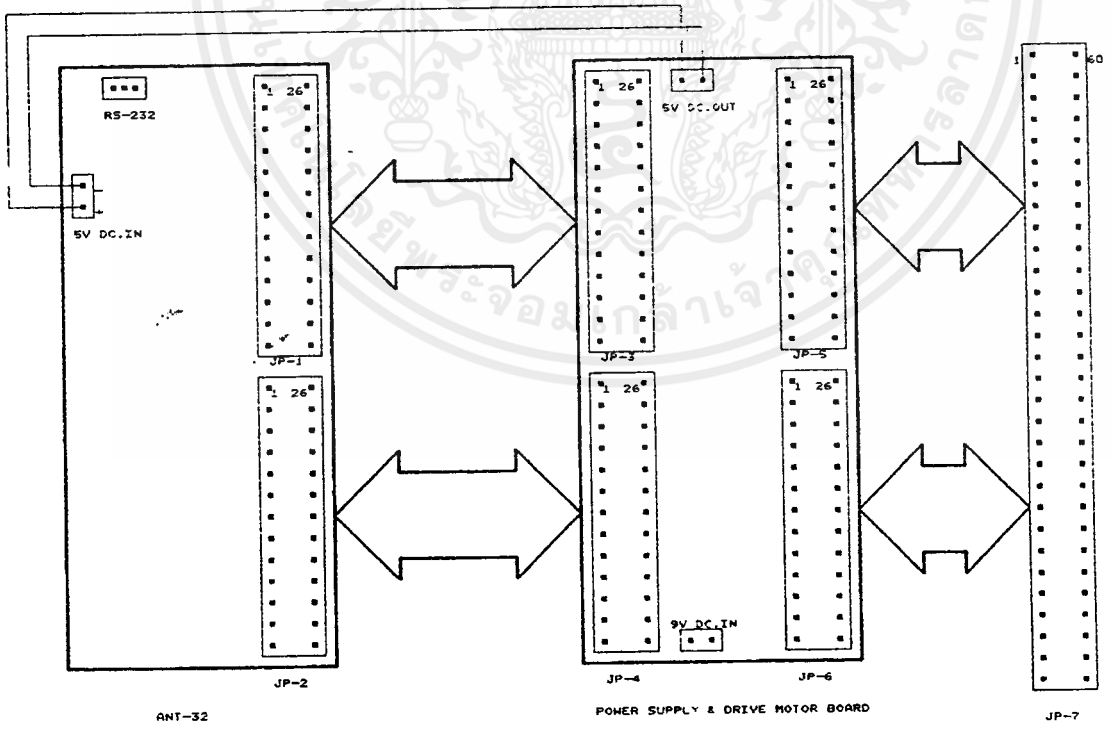
-การเลื่อน ขึ้น-ลง ใช้มอเตอร์ M2 ในการเลื่อน โดยมี USER1 PB2 เป็นบิต (BIT) การควบคุมทิศทาง และ PB3 เป็นบิต(BIT) ควบคุมการหมุน/หยุดหมุน

-การเลื่อน เข้า-ออก เพื่อจัดเก็บสินค้า ใช้มอเตอร์ M3 ในการเลื่อน โดยมี USER1 PB4 เป็นบิต(BIT) การควบคุมทิศทาง และ PB5 เป็นบิต(BIT) ควบคุมการหมุน/หยุดหมุน

-การเลื่อน เข้า-ออก ของ CONVEYOR ใช้มอเตอร์ M4 ในการเลื่อน โดยมี USER1 PB6 เป็นบิต(BIT) การควบคุมทิศทาง และ PB7 เป็นบิต(BIT) ควบคุมการหมุน/หยุดหมุน

3.2.3 ลักษณะของ DIP JUMPER และรายละเอียดของขาต่างๆ

ลักษณะของ DIP JUMPER ที่ต่อระหว่าง บอร์ดต่างๆ แสดงได้ดังรูป 3.4

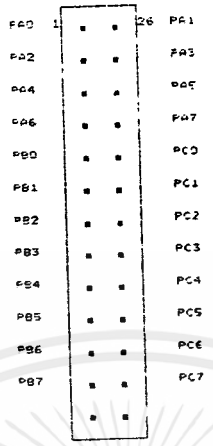


รูปที่ 3.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

032551

รายละเอียดของ JUMPER ต่างๆ มีดังนี้



รูปที่ 3.5

JP-1(8255 PPi USER 1)

พอร์ตที่ใช้งานของ 8255 PPI USER 1 คือ PORT B ADDRESS F801H ตั้งแต่ขาที่ 5-12 ดังมีรายละเอียดของแต่ละบิต (BIT) ดังนี้

ขา 5 PB0 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมทิศทาง M1

ขา 6 PB1 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมการหมุน M1

ขา 7 PB2 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมทิศทาง M2

ขา 8 PB3 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมการหมุน M2

ขา 9 PB4 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมทิศทาง M3

ขา 10 PB5 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมการหมุน M3

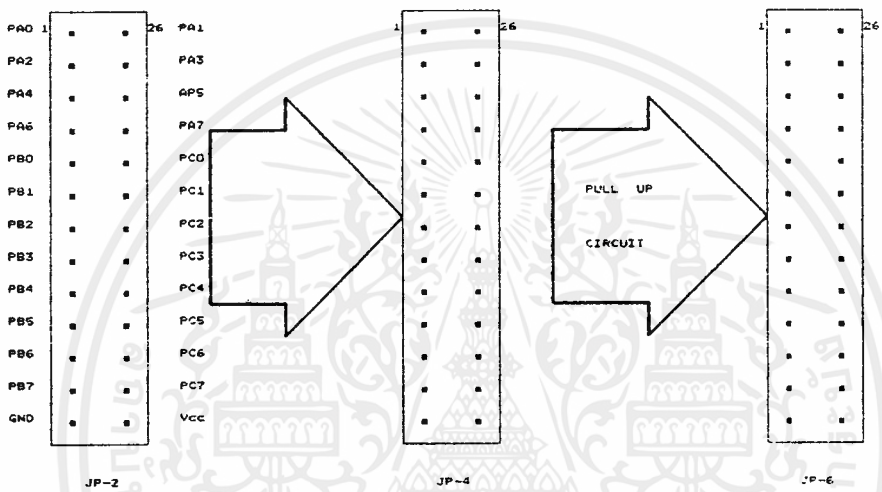
ขา 11 PB6 เป็น OUTPUT BIT ออกไปวงจร์ DRIVE MOTOR เพื่อควบคุมทิศทาง M4

ขา 12 PB7 เป็น OUTPUT BIT ออกไปวงจร DRIVE MOTOR เพื่อควบคุมการหมุน M4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

JP-2 (8255 PPI USER 2),JP-4,JP-6

USERPORT2เป็น PORT ที่สำหรับทางด้านINPUT เพื่ออ่านค่าจาก LIMIT SWITCH ตำแหน่งต่างๆ ของ WAREHOUSE โดยต่อผ่าน BOARD DRIVE MOTOR เพื่อต่อไปยังวงจร PULL UP โดยผ่าน DIP JUMPER JP-2 ไปยัง JP-4 ผ่านวงจร PULL UP ออกไปยัง JP-6 และ JP-7 ต่อไปยัง WAREHOUSE ตามลำดับ



รูปที่ 3.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

JP-2 มีรายละเอียดของแต่ละบิต(BIT) ดังนี้

ขา 1 PA 0 เป็น INPUT LIMIT SWITCH ตัวที่ 1

ขา 26 PA 1 เป็น INPUT LIMIT SWITCH ตัวที่ 2

ขา 2 PA 2 เป็น INPUT LIMIT SWITCH ตัวที่ 3

ขา 25 PA 3 เป็น INPUT LIMIT SWITCH ตัวที่ 4

ขา 3 PA 4 เป็น INPUT LIMIT SWITCH ตัวที่ 5

ขา 24 PA 5 เป็น INPUT LIMIT SWITCH ตัวที่ 6

ขา 4 PA 5 เป็น INPUT LIMIT SWITCH ตัวที่ 7

ขา 23 PA 6 NOT USED

ขา 5 PB 0 เป็น INPUT LIMIT SWITCH ตัวที่ 8

ขา 6 PB 1 เป็น INPUT LIMIT SWITCH ตัวที่ 9

ขา 7 PB 2 เป็น INPUT LIMIT SWITCH ตัวที่ 11

ขา 8 PB 3 เป็น INPUT LIMIT SWITCH ตัวที่ 11

ขา 9 PB 4 เป็น INPUT LIMIT SWITCH ตัวที่ 12

ขา 10 PB 5 เป็น INPUT LIMIT SWITCH ตัวที่ 13

ขา 11-12 PORT PB 6 - PB 7 NOT USED

ขา 22 PORT PC0 เป็น INPUT LIMIT SWITCH ตัวที่ 14

ขา 21 PORT PC1 เป็น INPUT LIMIT SWITCH ตัวที่ 15

ขา 20 PORT PC2 เป็น INPUT LIMIT SWITCH ตัวที่ 16

ขา 19 PORT PC3 เป็น INPUT LIMIT SWITCH ตัวที่ 17

ขา 18 PORT PC4 เป็น INPUT LIMIT SWITCH ตัวที่ 18

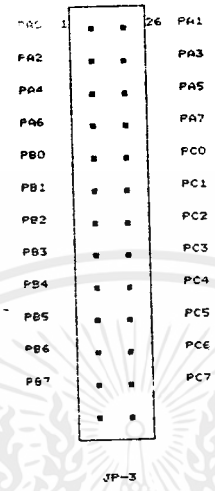
ขา 15-17 PORT PC5-7 NOT USED

ขา 13 เป็น GND

ขา 14 เป็น V_{cc}

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

JP-3 เป็น JUMPER รับสัญญาณออกจาก 8255 PPI USER1 เพื่อป้อนให้กับวงจร DRIVE MOTOR เพื่อควบคุมการทำงานของ WAREHOUSE ดังมีรายละเอียดของขาต่างๆ ดังนี้



รูปที่ 3.7

PORT A ขา 1-4 และขา 23-26 NOT USED

PORT C ขา 15-22 NOT USED

PORT B

ขา 5 PB0 เป็นสัญญาณควบคุมทิศทางของมอเตอร์ M1

ขา 6 PB1 เป็นสัญญาณควบคุมการหมุนของมอเตอร์ M1

ขา 7 PB2 เป็นสัญญาณควบคุมทิศทางของมอเตอร์ M2

ขา 8 PB3 เป็นสัญญาณควบคุมการหมุนของมอเตอร์ M2

ขา 9 PB4 เป็นสัญญาณควบคุมทิศทางของมอเตอร์ M3

ขา 10 PB5 เป็นสัญญาณควบคุมการหมุนของมอเตอร์ M3

ขา 11 PB6 เป็นสัญญาณควบคุมทิศทางของมอเตอร์ M4

ขา 12 PB7 เป็นสัญญาณควบคุมการหมุนของมอเตอร์ M4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลักษณะการ WIRING จาก JP-5 และ JP-6 ไปยัง JP-7

JP-5		ไปยัง	JP-7
M1	ขา 5 (JJ)		ขา 2
	ขา 6 (KK)		ขา 1
M2	ขา 7 (FF)		ขา 4
	ขา 8 (HH)		ขา 3
M3	ขา 9 (DD)		ขา 6
	ขา 10 (EE)		ขา 5
M4	ขา 11 (E)		ขา 27
	ขา 12 (F)		ขา 26

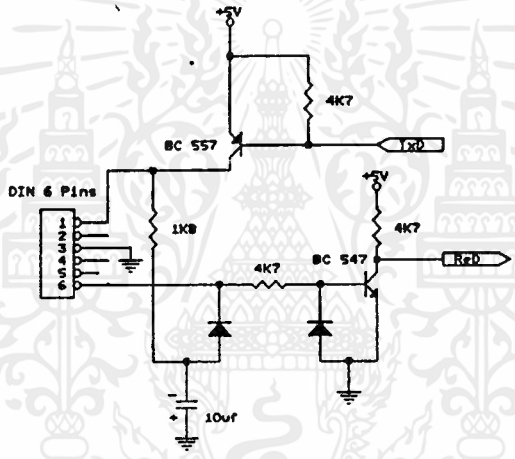
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

JP-6	ไปยัง	JP-7
ขา 1 (P)		ขา 18
ขา 2 (M)		ขา 20
ขา 3 (K)		ขา 22
ขา 4 (H)		ขา 24
ขา 5 (Y)		ขา 10
ขา 6 (V)		ขา 13
ขา 7 (X)		ขา 11
ขา 8 (S)		ขา 14
ขา 9 (W)		ขา 12
ขา 10 (R)		ขา 15
ขา 13 (AA)		ขา 31
ขา 18 (Z)		ขา 8
ขา 19 (BB)		ขา 9
ขา 20 (B)		ขา 7
ขา 21 (D)		ขา 30
ขา 22 (J)		ขา 28
ขา 24 (L)		ขา 23
ขา 25 (N)		ขา 21
ขา 26 (A)		ขา 19

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การเชื่อมต่อกับ IBM PC .

เป็นการสื่อสารแบบอนุกรมกันระหว่าง IBM PC กับ ANT-32 เพื่อให้ระบบควบคุมใช้งานได้กว้างขวางขึ้น ประกอบกับตัวไมโครคอนโทรลเลอร์ มีพอร์ตอนุกรมอยู่ในตัวจึงได้จัดเตรียมพอร์ตเพื่อการสื่อสารกับภายนอก โดยจะทำการสื่อสารกับ พี ซี คอมพิวเตอร์(PC COMPUTER) เพื่อใช้ในการตอบโต้กับผู้ใช้ โดยจะใช้คีบอร์ดและจอภาพของพีซีคอมพิวเตอร์เป็นคอนโซล(Console) นอกจากนั้นยังใช้สำหรับการพัฒนาซอฟแวร์ในอนาคตด้วย ในการสื่อสารกับ พีซี คอมพิวเตอร์ จะสื่อสารโดยใช้พอร์ตอนุกรมซึ่งมีมาตรฐานการสื่อสารเป็นแบบ RS-232 วงจรเชื่อมต่อ RS-232 กับ พีซี คอมพิวเตอร์ เป็นดังรูป 3.8

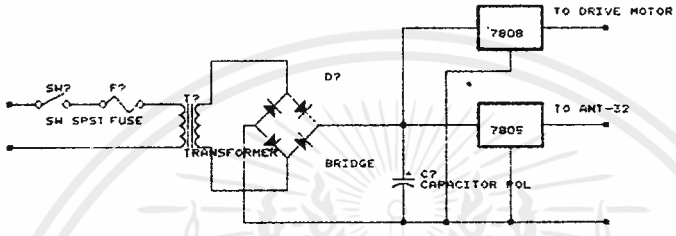


รูปที่ 3.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 ภาคจ่ายไฟ(POWER SUPPLY)

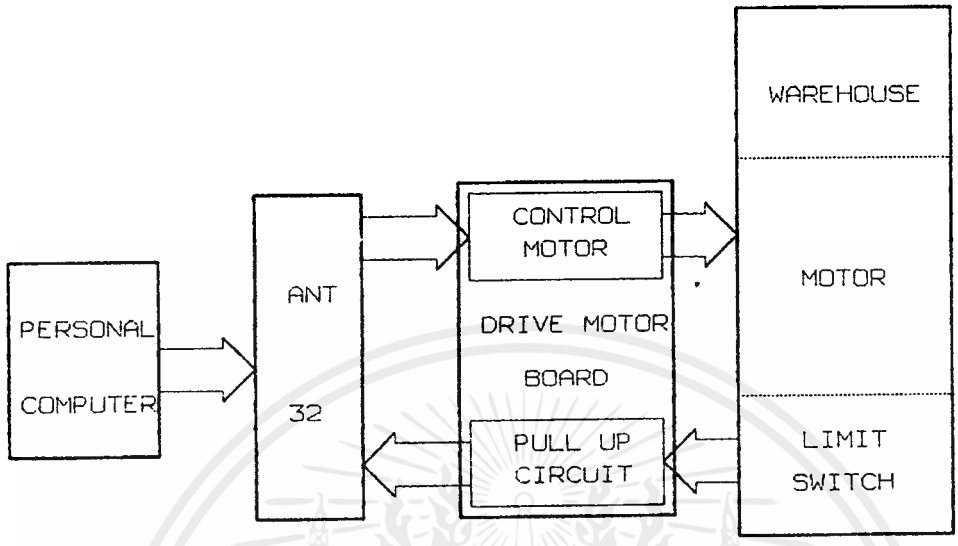
ภาคจ่ายไฟจะทำหน้าที่จ่ายแรงดันไฟฟ้า ให้กับส่วนต่างๆของวงจร ซึ่งภาคจ่ายไฟนี้จะใช้ LINEAR IC REGULATE เพื่อทำให้แรงดันคงที่ ที่ 5 โวลท์ เพื่อป้อนให้กับ บอร์ด ANT-32 และ ที่ 8 โวลท์ เพื่อป้อนให้กับวงจร DRIVE MOTOR วงจรแสดงดังรูป



รูปที่ 3.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 แผนผังการเชื่อมต่อ



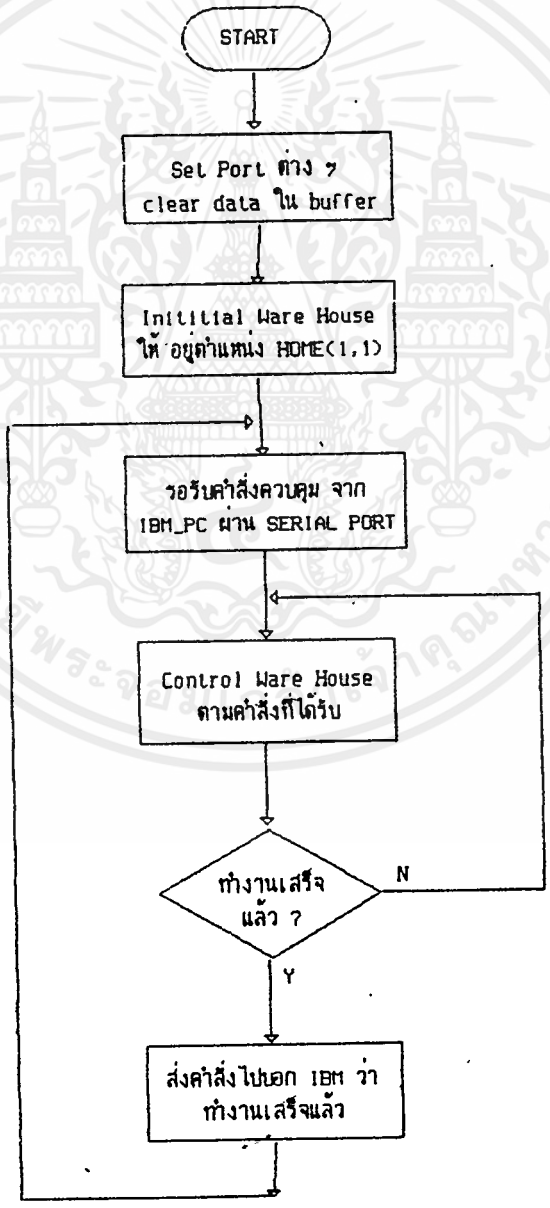
รูปที่ 3.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่4

การออกแบบทาง Software

การออกแบบจะแยกโปรแกรมจะแยกโปรแกรมออกเป็นโปรแกรมย่อย(Sub program) ต่างๆ ซึ่งจะถูกเรียกใช้จาก main program เป็นการทำให้ เขียนโปรแกรมได้อย่างมีประสิทธิภาพและมีขนาดเล็ก โดยการอธิบายเป็นลำดับไปเริ่มจาก main program ซึ่งจะเป็น loop control หลัก แสดงดัง flowchart



รูปที่ 4.1 Flowchart ของ Loop control หลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อธิบายการทำงานของโปรแกรมหลัก

เมื่อเปิดให้ Automatic Warehouse Control ทำงาน โปรแกรมจะทำการ Inititial I/O Port ต่างๆเพื่อใช้ในการ Control โดย 8255PPI จะเป็นหน่วย Input/Output ของระบบซึ่งมีอยู่ 2 ตัวคือ 8255 User1 อยู่ที่ Address F800H-F803H จะนำมาใช้เป็น Output Port สำหรับ Control Motor จำนวน 4 ตัวโดย Set ให้เป็น Output Port โดยส่ง Data = 80H ออกไปที่ Port F803H และ 8255 User2 อยู่ที่ Address FC00H-FC03H จะนำมาใช้เป็น Input Port สำหรับ Limit Switch เพื่อใช้ในการตรวจสอบตำแหน่งต่างๆของ Lift ที่ Row และ Column และ Conveyor นำสินค้าเข้า/ออก โดยส่ง Data = 9BH ออกไปที่ Port FC03H จากนั้นก็จะทำการ Clear Buffer ต่างๆที่ใช้เป็น Temporary Buffer ก็คือ

-Internal ram address 21H SWBUF1 สำหรับการใช้เป็น Buffer Limit Switch หมายเลข 1-7

-Internal ram address 22H SWBUF2 สำหรับการใช้เป็น Buffer Limit Switch หมายเลข 8-13

-Internal ram address 23H SWBUF3 สำหรับการใช้เป็น Buffer Limit Switch หมายเลข 14-20

-Internal ram adderss 24H M_BUF1 สำหรับการใช้เป็น Buffer ในการ Control Motor 1-4

-Internal ram adderss 25H M_BUF1 สำหรับการใช้เป็น Buffer ในการ Control Motor 5

เมื่อ Clear ค่าต่างๆใน Buffer แล้วก็จะรอรับคำสั่งควบคุมจาก IBM-PC ผ่านทาง Serial Port ซึ่งประกอบด้วยชุดคำสั่งต่างๆคือ

-Irc การนำเอาสินค้าบน Conveyor เข้าไปเก็บในช่องที่ แถว = r และคอลลัมน์ = c

-Orc การนำเอาสินค้า ออกจากช่องที่ แถว = r และคอลลัมน์ = c ไปไว้ที่ Conveyer ทางออก

-Hrc คำสั่งเลื่อน Lift ยกของมายังตำแหน่ง แถว = r และคอลลัมน์ = c

หลังจากที่ทำงานตามคำสั่งเสร็จแล้วก็จะส่งคำสั่งกลับไปบอก IBM-PC ว่าทำงานเสร็จแล้วพร้อมที่จะรับคำสั่งต่อไปมาทำงานโดยส่งคำสั่ง R (Ready) ไปยัง IBM-PC ซึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

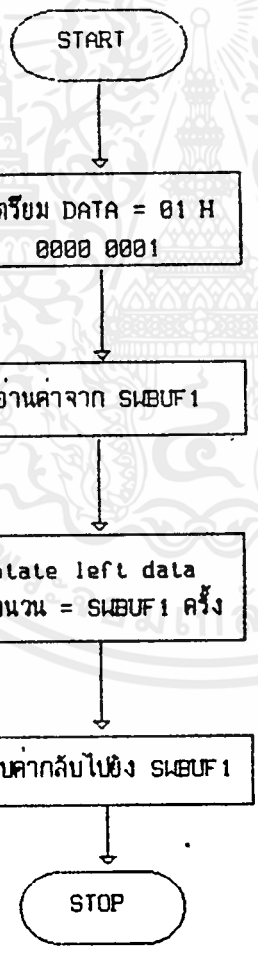
Main Program จะทำงานวนเพียงเท่านี้ ส่วน Sub Program ต่างๆจะได้แยกอธิบายออกเป็นส่วนๆ

การใช้งานของ Buffer แต่ละ Byte ว่านำไปใช้ทำอะไร เมื่อ CPU ได้รับคำสั่งมาจาก IBM-PC ก็จะนำค่า r, c มาเก็บไว้ยัง Address ต่างๆดังนี้

-r เก็บไว้ยัง SWBUF1 ค่า c อยู่ในช่วง 1-7

-c เก็บไว้ยัง SWBUF2 ค่า r อยู่ในช่วง 1-3

ในการหาตำแหน่งในแนว Column ดูรูป 4.2



รูป 4.2 แสดง Flowchart ในการหาตำแหน่ง Limit Switch ที่สัมพันธ์กับค่าใน SWBUF

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

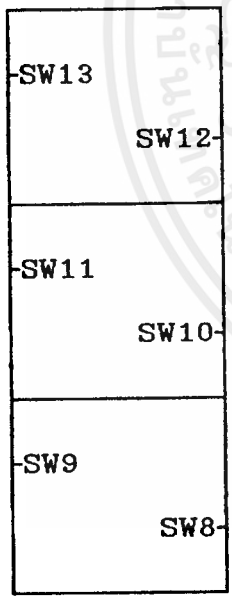
การทำเช่นนี้เพื่อความสะดวกในการตรวจสอบค่าที่อ่านได้จาก Limit switch หมายเลข 1-7 ที่อ่านค่าจาก 8255 User2 Port A โดยการนำมา Compare กันดูรูปที่ 4.3

Bit No.	0	1	2	3	4	5	6	7
Sw. No.	1	2	3	4	5	6	7	X

รูป 4.3
PORTSWA และ SW.No ต่างๆ

หากทำการ Rotate data $_{lsb}1000\ 0000_{msb}$ ไป 3 ครั้ง จะได้ $_{lsb}0001\ 0000_{msb}$ เอาไว้เตรียมสำหรับการ Compare กับ data ที่อ่านมาจาก PORTSWA ทั้งในกรณีนำสินค้าเข้า-ออก จาก Warehouse

ในการหาตำแหน่งแนวตั้งของ Warehouse พิจารณาจากรูป 4.4



Bit No.	7	6	5	4	3	2	1	0
Sw. No.	X	X	13	12	11	10	9	8

รูปที่ 4.4
SW. ตำแหน่งต่างๆบน Warehouse และ Bit ต่างๆ ที่ PORTSWB

ในการนำสินค้าเข้าและการนำสินค้าออกจาก Warehouse จะมีการเตรียมข้อมูลที่ SWBUF2 ต่างกันดังนี้ ยกตัวอย่างชี้ที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นำสินค้าเข้า

-ยก Lift ให้ถึง SW11

-เลื่อนสินค้าเข้า

-เลื่อน Lift ลงให้ถึง SW10

-เลื่อน Lift สินค้าออกจากช่องเก็บ

ต้องเตรียม Data ใน SWBUF2 เป็น 0000 1000

นำสินค้าออก

-ยก Lift ให้ถึง SW10

-เลื่อนสินค้าเข้า

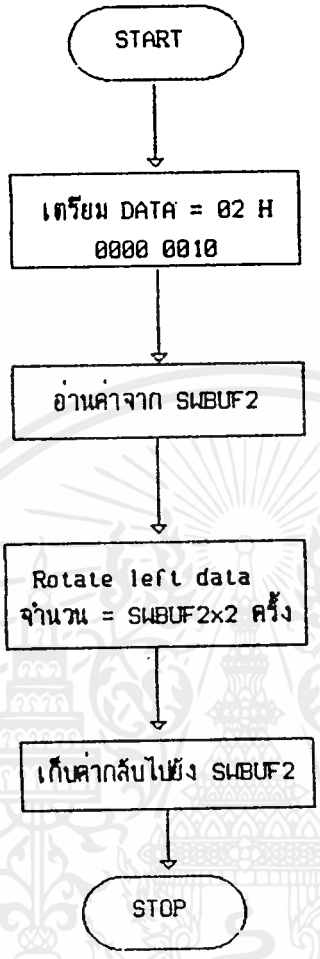
-ยก Lift ลงให้ถึง SW11

-เลื่อน Lift สินค้าออกจากช่องเก็บ

ต้องเตรียม Data ใน SWBUF2 เป็น 0000 0100

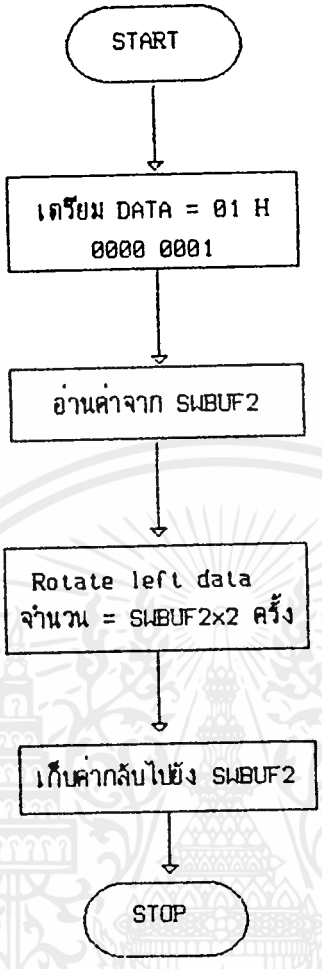
เขียนแสดงเป็น Flowchart ดังรูป 4.5 และ 4.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.5 Flowchart การเตรียม Data สำหรับ Compare ในการนำสินค้าเข้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.6 Flowchart การเตรียม Data สำหรับ Compere ในการนำสินค้าออก

การควบคุม Motor จะใช้ 8255 User1 PORTA เป็นสัญญาณในการควบคุมวงจร Drive motor DC ที่ต้องสามารถหมุนกลับทาง Motor ได้สัญญาณต่างๆเป็นดังรูป 4.6

	A	B	A	B	A	B	A	B
Bit No.	7	6	5	4	3	2	1	0
PortMA								

M4 M3 M2 M1

รูป 4.6 แสดง Bit ที่ใช้ Control Motor ที่ 8255 User1 PORTA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย M1 ควบคุมการเคลื่อนที่ในแนวนอน

M2 ควบคุมการเคลื่อนที่ในแนวตั้ง

M3 ควบคุมการเคลื่อนที่ในการนำสินค้าเข้า-ออกจากช่องเก็บสินค้า

M4 ควบคุมการเคลื่อนที่ของ Conveyor

A และ B คือการควบคุมการหมุนดังรูป 4.7

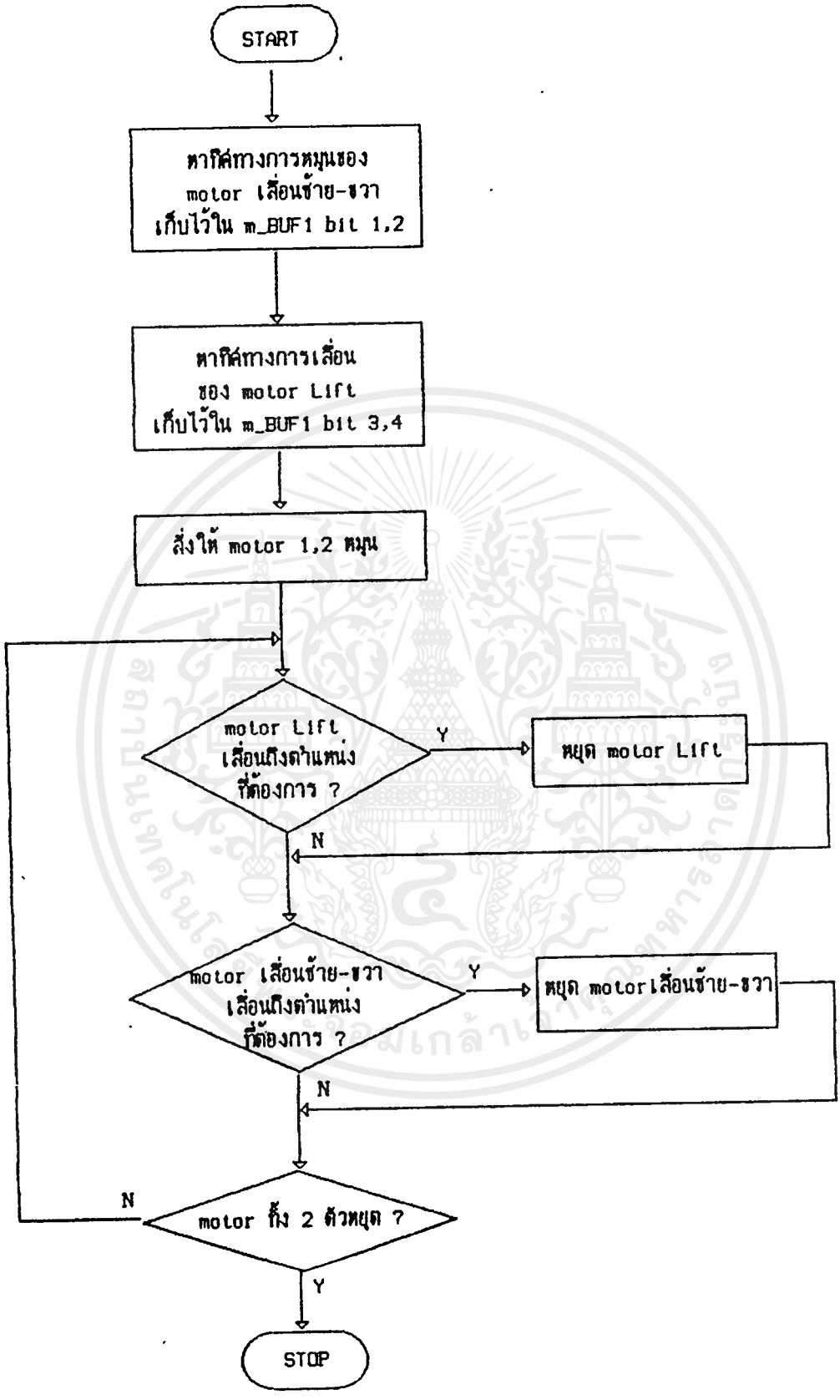
A	B	
0	0	หยุดหมุน
0	1	หยุดหมุน
1	0	หมุนซ้าย
1	1	หมุนขวา

รูปที่ 4.7
สัญญาณในการควบคุม Motor

การควบคุมมอเตอร์แบ่งออกเป็นส่วนย่อยๆ ดังนี้

Flowchart แสดงการเลื่อน Lift ยกของดังรูป 4.8 อธิบายดังนี้เริ่มจากการหาทิศทางการหมุนของ Motor ว่า AB จะต้องเป็น Format ไหนแล้วเก็บค่าที่จะใช้ ControlMotor1,Motor2 เอาไว้ที่ Bit0-3ของM_BUF1 จากนั้นก็ส่งค่าออกไปที่ 8255 User1 PortB เพื่อให้วงจร Drive Motor Drive Motor1, Motor2 ในทิศทางที่ต้องการ เมื่อ Motor หมุนแล้วต่อไปก็ทำการตรวจสอบดูว่า Motor หมุนไปจนถึงตำแหน่งที่ Set ไว้ที่ SWBUF1, SWBUF2 หรือยังหาก Motor ตัวใดหมุนจนถึงตำแหน่งที่ต้องการแล้วก็สั่งให้ Motor ตัวนั้นหยุดหมุนและ Set Status ของ Motor เอาไว้รอจนกว่า Motor จะหยุดหมุนทั้ง 2 ตัว ส่วนรายละเอียดในส่วนย่อยจะอธิบายต่อไป

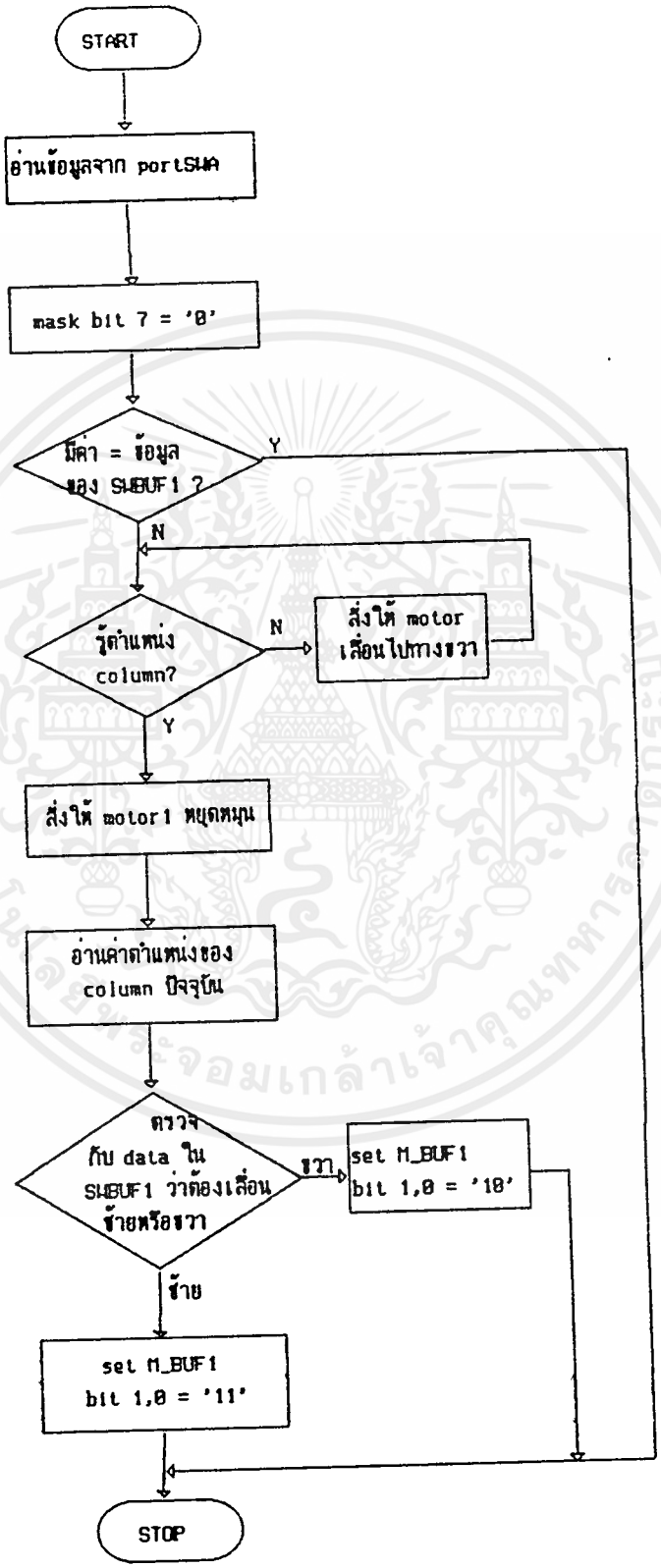
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.8 Loop ควบคุมการสั่งให้ Motor เลื่อนจนถึงตำแหน่ง Sensor Switch ที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Flowchart แสดงรายละเอียดในการหาตำแหน่ง Column ที่ต้องการเก็บสินค้า แสดงในรูป 4.9



รูป 4.9 Flowchart แสดงการหาตำแหน่ง Column

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จาก Flowchart รูป 4.9 อธิบายดังนี้ เริ่มจากการอ่านค่าที่ PORTSWA ซึ่งใช้สำหรับตรวจสอบสถานะ Limit Switch ในแนว Column เนื่องจาก Bit ที่ 7 ไม่ได้ใช้งานจึงต้องทำการ Mask ให้ = '0' แล้วนำเอาข้อมูลที่ได้มาเปรียบเทียบกับค่าใน SWBUF1 ซึ่งได้ทำการ Adjust ให้อยู่ในลักษณะเดียวกับตำแหน่ง Sensor Switch ที่ต้องการ หากมีค่าเท่ากันก็หมายความว่าไม่จำเป็นต้องเลื่อนตำแหน่งของ Lift เพราะอยู่ในตำแหน่งที่ต้องการอยู่แล้ว หากไม่เท่ากันก็จะทำการตรวจสอบต่ออีกว่าค่าที่อ่านจาก Limit Switch เท่ากับ 00H หรือไม่ เพราะว่าหากเท่ากับ 00H ก็หมายความว่าขณะนี้โปรแกรมไม่สามารถรู้ได้ว่า Lift ยกสินค้ากำลังอยู่ที่ Column ใด ต้องทำการสั่งให้ Motor 1 เลื่อนไปทางซ้ายแล้วคอยตรวจสอบจนกระทั่งค่าที่อ่านจาก PORTSWA ที่ค่าไม่เท่ากับ 00H แล้วสั่งให้ Motor 1 หยุดหมุน ตอนนี้โปรแกรมรู้แล้วว่า Lift สินค้าอยู่ที่ตำแหน่ง Column ใด ต่อไปต้องทำการตรวจสอบว่าจากตำแหน่งปัจจุบันนี้ต้องทำการ Control ให้ Motor1 เลื่อนไปทางซ้ายหรือเลื่อนไปทางขวาเพื่อที่จะเลื่อน Lift สินค้าไปยังตำแหน่งที่ต้องการซึ่ง Set ไว้ใน SWBUF1 การตรวจสอบทำโดยนำค่าที่ Set ไว้ใน SWBUF1 และค่าที่อ่านได้จาก PORTSWB มาลบกันแล้วตรวจดูสถานะของ Register PSW.7 ในตัว CPU หากต้องการเลื่อนไปทางขวาก็ทำการ Set Bit ที่ 1 และ 0 ของ M_BUF1 ให้เป็น '10' แต่หากต้องการเลื่อนไปทางซ้ายก็ทำการ Set Bit ที่ 1 และ 0 ของ M_BUF1 ให้เป็น '01'

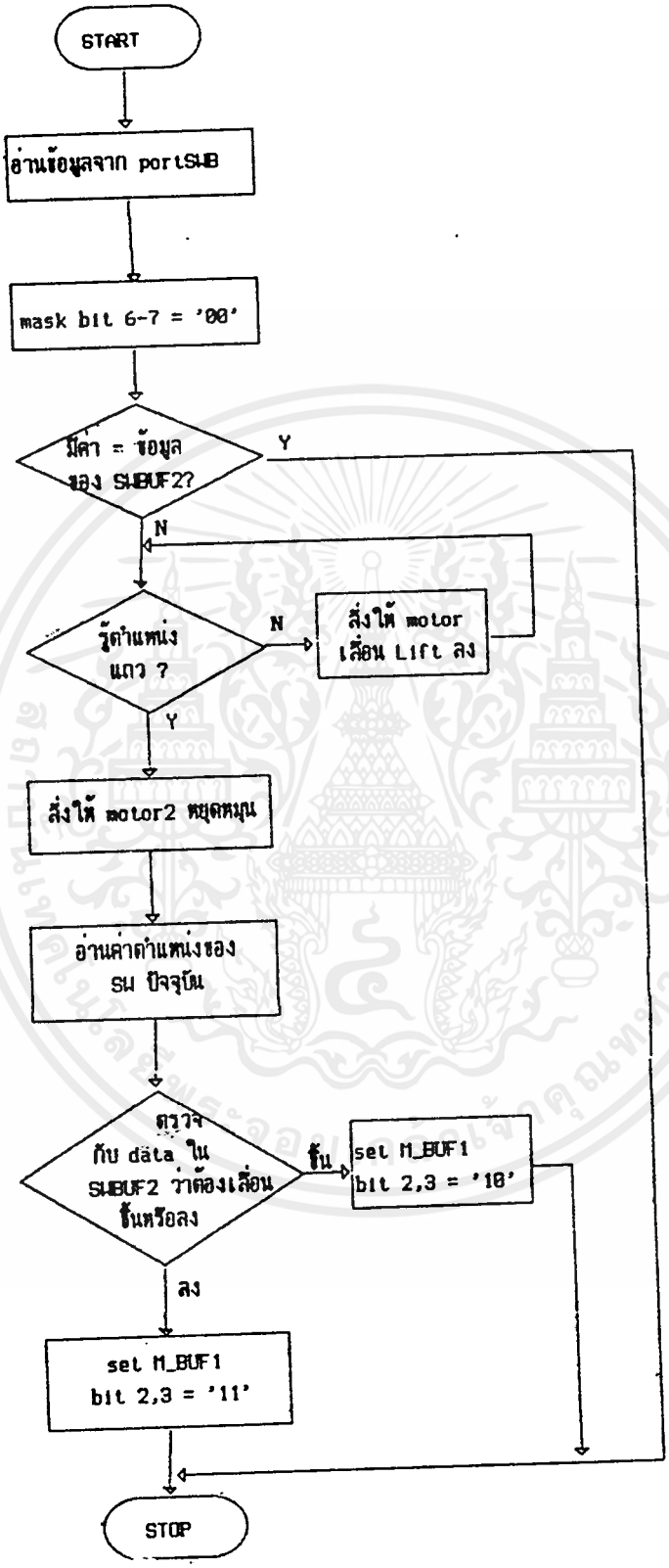
ในการหาตำแหน่งในแนวตั้งก็ทำคล้ายๆกันกับการหาตำแหน่งในแนว Column เริ่มจากการอ่านค่าที่ PORTSWB ซึ่งใช้สำหรับตรวจสอบสถานะ Limit Switch ในแนวตั้ง เนื่องจาก Bit ที่ 6-7 ไม่ได้ใช้งานจึงต้องทำการ Mask ให้ = '0' แล้วนำเอาข้อมูลที่ได้มาเปรียบเทียบกับค่าใน SWBUF2 ซึ่งได้ทำการ Adjust ให้อยู่ในลักษณะเดียวกับตำแหน่ง Limit Switch ที่ต้องการ หากมีค่าเท่ากันก็หมายความว่าไม่จำเป็นต้องเลื่อนตำแหน่งของ Lift เพราะอยู่ในตำแหน่งที่ต้องการอยู่แล้ว หากไม่เท่ากันก็จะทำการตรวจสอบต่ออีกว่าค่าที่อ่านจาก Limit Switch เท่ากับ 00H หรือไม่ เพราะว่าหากเท่ากับ 00H ก็หมายความว่าขณะนี้โปรแกรมไม่สามารถรู้ได้ว่า Lift ยกสินค้ากำลังอยู่ที่ Column ใด ต้องทำการสั่งให้ Motor 2 เลื่อนลงแล้วคอยตรวจสอบจนกระทั่งค่าที่อ่านจาก PORTSWB ที่ค่าไม่เท่ากับ 00H แล้วสั่งให้ Motor 2 หยุดหมุน ตอนนี้โปรแกรมรู้แล้วว่า Lift สินค้าอยู่ที่ตำแหน่งใด ต่อไปต้องทำการตรวจสอบว่าจากตำแหน่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปัจจุบันนี้ต้องทำการ Control ให้ Motor2 เลื่อนขึ้นหรือเลื่อนลงเพื่อที่จะเลื่อน Lift สินค้าไปยังตำแหน่งชั้นที่ต้องการซึ่ง Set ไว้ใน SWBUF2 การตรวจสอบทำโดยนำค่าที่ Set ไว้ใน SWBUF2 และค่าที่อ่านได้จาก PORTSWB มาลบกันแล้วตรวจดูสถานะของ Register PSW.7 ในตัว CPU หากต้องการเลื่อนไปทางขวาก็ทำการ Set Bit ที่ 2 และ 3 ของ M_BUF1 ให้เป็น '10' แต่หากต้องการเลื่อนไปทางซ้ายก็ทำการ Set Bit ที่ 1 และ 0 ของ M_BUF1 ให้เป็น '01' Flowchart แสดงดังรูปที่ 4.10 ในหน้าถัดไป

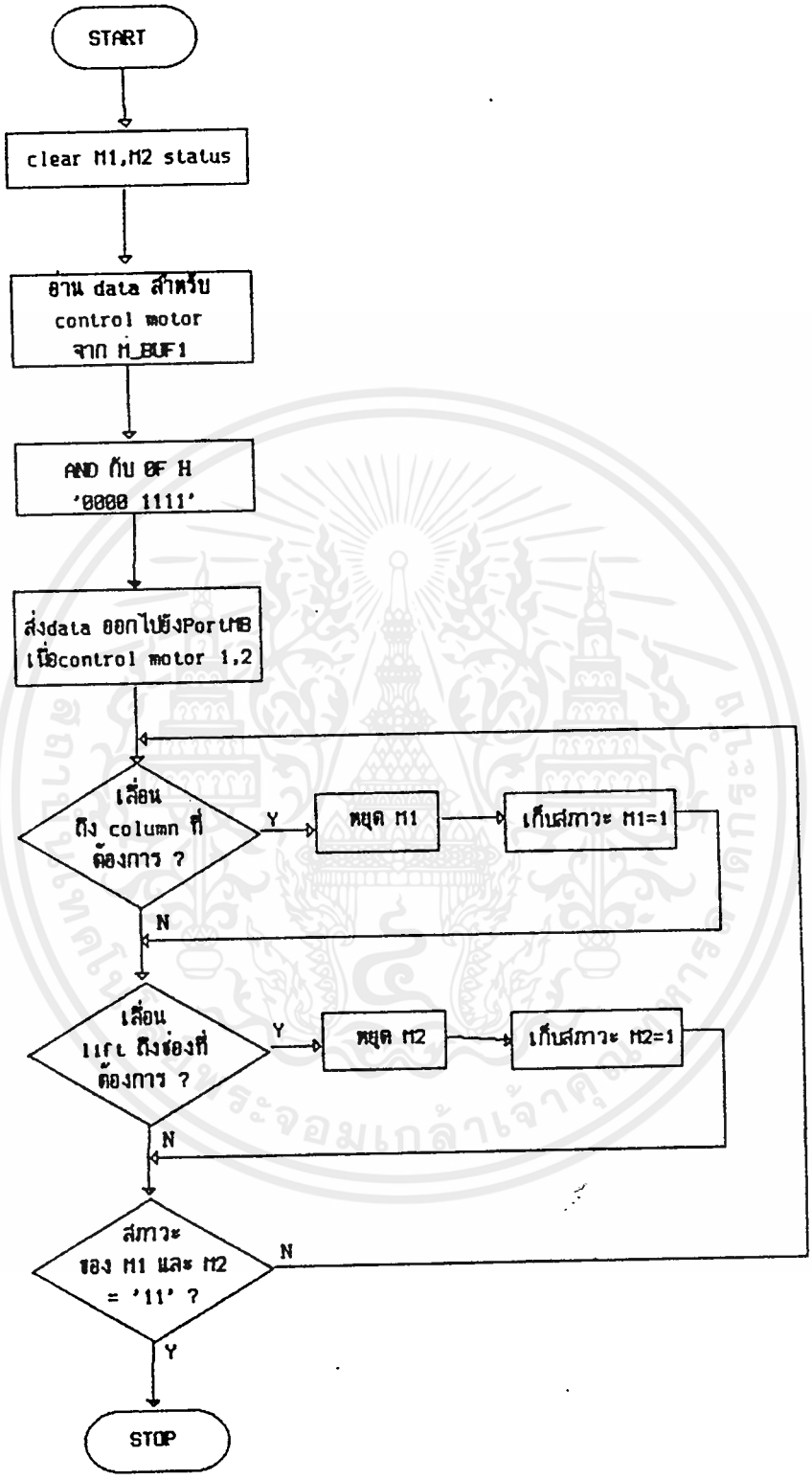
เมื่อรู้ตำแหน่งทิศทางการหมุนของ Motor1, Motor2 แล้วก็สั่งให้ Motor ทั้งสองตัวทำงานพร้อมกันได้เลย ดู Flowchart รูปที่ 4.11 ประกอบ โดยการนำเอาค่าใน M_BUF1 มา Mask Bit 4-7 ให้เป็น '0000' ก่อนแล้วจึงส่งออกไปยัง Port User1 A ซึ่งจะใช้ในการ Control วงจร Drive Motor จากนั้น CPU จะคอยตรวจสอบที่ PORTSW1, PORTSW2 ว่า Limitr Switch ที่ตำแหน่งที่ต้องการให้หยุดเปลี่ยนจาก Logic '0' เป็น Logic '1' หรือยังโดยการนำเอาค่าที่อ่านได้จาก PORTSW1 และ PORTSW2 มาเปรียบเทียบกับค่าใน SWBUF1 และ SWBUF2 ถ้าตำแหน่งชั้นถึงก่อนก็หยุด Motor2 แล้วทำการเก็บสถานะเอาไว้ รอจนอีกตำแหน่งหนึ่งถึงจุดที่ต้องการ เมื่อ Motor1 เลื่อน Lift สินค้าไปถึงตำแหน่ง Column ที่ต้องการแล้วก็หยุด Motor1 แล้วเก็บสภาวะเอาไว้ ในขณะที่กำลังตรวจสอบตำแหน่งของ Lift อยู่นั้นโปรแกรมก็จะวน Loop คอยตรวจดูค่าสถานะของ Motor1 และ Motor2 จนกว่าจะเท่ากับ '11' จึงจะถือว่า Lift สินค้าเลื่อนมาถึงตำแหน่งที่ต้องการแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.10 Flowchart แสดงการหาตำแหน่งในแนวนอน

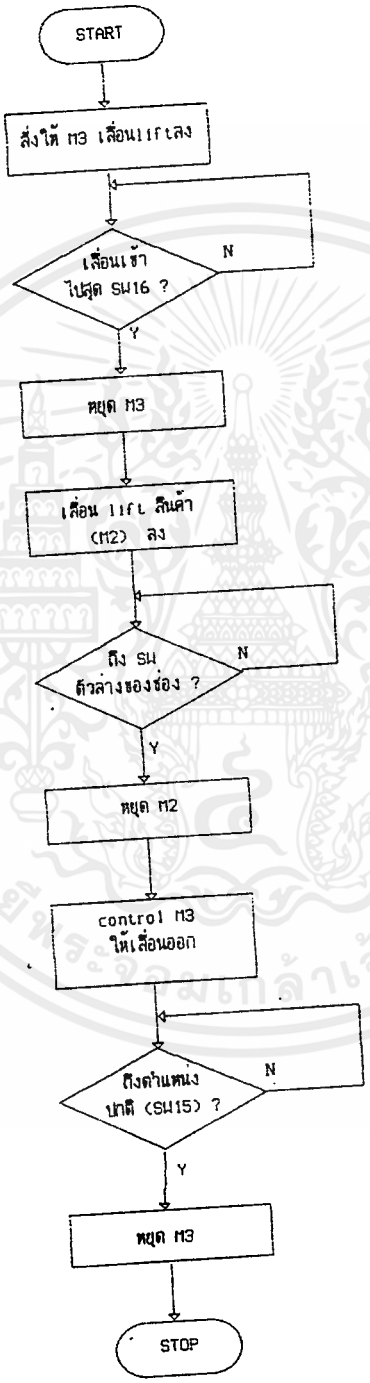
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.11 แสดง Flowchart ในการตรวจว่า Lift สินค้าเลื่อนมาถึงตำแหน่งที่ Set ไว้ที่ SWBUF1 และ SWBUF2 หรือยัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อ Lift สินค้าเลื่อนมาถึงตำแหน่งที่ต้องการเรียบร้อยแล้ว ต่อไปก็จะเป็นขั้นตอนในการนำสินค้าเข้าและออกจากช่องเก็บสินค้า Flowchart ในการนำสินค้าเข้า แสดงดังรูป 4.12



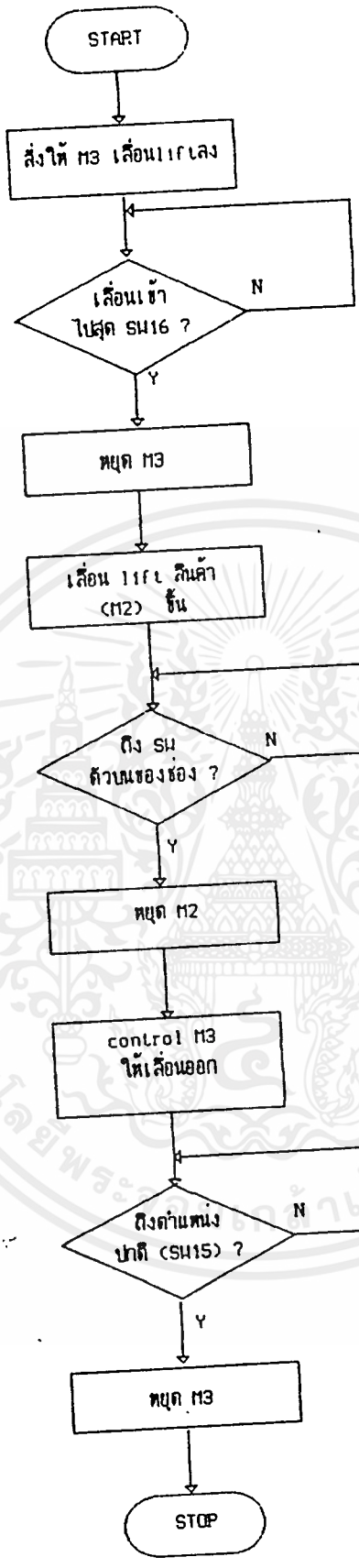
รูป 4.12 แสดง Flowchart ในการนำสินค้าเข้าช่องเก็บสินค้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Flowchart ในรูป 4.12 จะเริ่มจากการที่ Lift สินค้าเลื่อนมาอยู่ที่ตำแหน่งชั้นที่ต้องการนำเอาสินค้าเข้าเก็บเรียบร้อยแล้ว ซึ่งจะต้องอยู่ที่ตำแหน่ง Limit Switch ตัวบนคือ SW9, SW11 หรือ SW13 เท่านั้นจากนั้นก็จะทำการเลื่อนสินค้าเข้าไปเก็บในช่องเก็บสินค้า โดยการสั่งให้ Motor3 ทำงานและตรวจสอบการเข้าไปในสุดโดย Limit Switch SW16 เมื่อสามารถตรวจจับ Switch S16 จาก Logic '0' เป็น Logic '1' ได้แล้วก็ทำการหยุด Motor3 แล้วเลื่อน Lift สินค้าลงมา 1 ขั้น โดยการสั่งให้ Motor2 เลื่อน Lift สินค้าลงซึ่งจะมีผลทำให้สินค้าค้างอยู่บนชั้นวางจากนั้นก็สั่งให้ Motor3 เลื่อนเอา Lift สินค้าออกมาจากช่องเก็บ

ในการนำสินค้าออกจากช่องเก็บแสดง Flowchart ดังรูปที่ 4.13เริ่มจากการที่ Lift สินค้าเลื่อนมาอยู่ที่ตำแหน่งชั้นที่ต้องการนำเอาสินค้าเข้าเก็บเรียบร้อยแล้ว ซึ่งจะต้องอยู่ที่ตำแหน่ง Limit Switch ตัวล่างคือ SW8, SW10 หรือ SW12 เท่านั้นจากนั้นก็จะทำการเลื่อน Lift สินค้าเข้าไปเก็บในช่องเก็บสินค้า โดยการสั่งให้ Motor 3 ทำงานและตรวจสอบการเข้าไปในสุดโดย Limit Switch SW16 เมื่อสามารถตรวจจับ Switch S16 จาก Logic '0' เป็น Logic '1' ได้แล้วก็ทำการหยุด Motor3 แล้วเลื่อน Lift สินค้าขึ้นมา 1 ขั้น โดยการสั่งให้ Motor2 เลื่อน Lift สินค้าขึ้นซึ่งจะมีผลทำให้สินค้าถูกยกออกจากชั้นวางจากนั้นก็สั่งให้ Motor3 เลื่อนเอา Lift สินค้าออกมาจากช่องเก็บ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.13 แสดง Flowchart ในการนำสินค้าออกจากช่องเก็บสินค้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมในการติดต่อสื่อสาร

ในการติดต่อสื่อสารต้องมีการ Set ค่า Parameter ต่างๆคือ BAUD RATE, PARITY, DATA BIT, STOPBIT ก่อนซึ่งในโครงงานนี้ใช้ค่า Parameter ดังนี้

BAUD RATE : 9600

PARITY : NO

DATA BIT : 8

STOPBIT : 1

ซึ่งมีการ Set ค่าใน Subroutine BAUDS ดังนี้

```
BAUDS:    MOV   TMOD, #020H      ;Set timer 1 mode 2
          MOV   SCON, #52H       ;Set serial 8 Bit
          MOV   TH1, #BRAT48     ;Set Baud rate BRAT48 = #0FAH
          SETB  TR1              ;Timer 1 on
          RET
```

ก็จะทำการ Set Parameter ต่างๆตามมต้องการ

การรับ-ส่งข้อมูลใช้ Subroutine SBYTE, RBYTE

```
SBYTE:    JNB   TI, $            ;Check Data
          CLR   TI               ;Reset TI
          MOV   A, SBUF          ;Read data
          RET                    ;Return

RBYTE:    JNB   RI, $            ;Check data
          CLR   RI               ;Reset RI
          MOV   SBUF, A          ;Read data
          RET                    ;Return data in REGs A
```

ในการับข้อมูล Data ที่รับได้จะเก็บที่ Register A และในการส่งข้อมูลข้อมูลก็จะส่งมาทาง Register A เช่นกัน

Source Program ของ Software ทั้งหมดแสดงในหน้าถัดไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    1       ;PROGRAM PROJECT.ASM
                    2       ;BY J.PAYUNGSAK
                    3       ;   W.NIPON
                    4       ;   K.NARIS
                    5       ;DESCRIPTION
                    6       ;   FOR CONTROL WAREHOUSE MODEL FROM IBM-PC
                    7       ;   RS-232C
                    8       ;HARDWARE
                    9       ;   ANT-31 + MOTOR DRIVE AND CONTROL BOARD
                   10       ;DATE
                   11       ;   17/3/93
                   12
                   13       $DATE
                   14       $WIDTH  90
                   15       $LENGTH 55
                   16       $TITLE "PROJECT INS 2S 2nd SEMESTER"
                   17
F800=              18            PORTOUT  EQU 0F800H
FC00=              19            PORTIN   EQU 0FC00H
0080=              20            CW_OOOO  EQU 80H
009B=              21            CW_IIII  EQU 9BH
F800=              22            PORTMA   EQU PORTOUT
F801=              23            PORTMB   EQU PORTMA+1
F802=              24            PORTMC   EQU PORTMA+2
F803=              25            PORTMCW  EQU PORTMA+3
FC00=              26            PORTSWA  EQU PORTIN
FC01=              27            PORTSWB  EQU PORTSWA+1
FC02=              28            PORTSWC  EQU PORTSWA+2
FC03=              29            PORTSWCW EQU PORTSWA+3
002D=              30            HEXBUF   EQU  002DH
0030=              31            KEYBUF   EQU  0030H
0020=              32            BITADD   EQU  0020H
0021=              33            SWBUF1   EQU  0021H
0022=              34            SWBUF2   EQU  0022H
0023=              35            SWBUF3   EQU  0023H
0024=              36            M_BUF1   EQU  0024H
0025=              37            M_BUF2   EQU  0025H
00F4=              38            BRAT24   EQU  00F4H
00FA=              39            BRAT48   EQU  00FAH
00FD=              40            BRAT96   EQU  00FDH
                   41       $EJECT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    ;╔═══════════════════════════╗
                    ;║ STARTING ROM ADDRESS 0000H ║
                    ;╚═══════════════════════════╝
0000                        ORG   0000H
0000 020100         POWER_ON:LJMP  RES

                    ;╔═══════════════════════╗
                    ;║ STACK STARTING ADDRESS ║
                    ;╚═══════════════════════╝
006C                        ORG   006CH
006C                STACK:  DS    30

                    ;╔═══════════════╗
                    ;║ POWER ON DELAY ║
                    ;╚═══════════════╝
0100                        ORG   0100H
0100 7A40           RES:    MOV   R2, #40H
0102 7B00           RES1:   MOV   R3, #0
0104 DBFE                   DJNZ  R3, $
0106 DAFA                   DJNZ  R2, RES1
0108 75816C                 MOV   SP, #STACK
010B 020150                 LJMP  START

                    ;╔═══════════════╗
                    ;║ PROGRAM START ║
                    ;╚═══════════════╝
0150                        ORG   0150H
0150 120334         START:  LCALL BAUDS
0153 90F803                 MOV   DPTR, #PORTMCW ;INITITIAL PORT
0156 748Q                   MOV   A, #CW_0000    ;PORT 1 FOR CONTROL MOTOR
0158 F0                     MOVX  @DPTR, A
0159 90FC03                 MOV   DPTR, #PORTSWCW
015C 749B                   MOV   A, #CW_IIII    ;PORT 2 FOR CHECK POSITION
015E F0                     MOVX  @DPTR, A
015F E4             MAIN:   CLR   A
0160 F521                   MOV   SWBUF1, A
0162 F522                   MOV   SWBUF2, A
0164 F523                   MOV   SWBUF3, A
0166 F524                   MOV   M_BUF1, A
0168 F525                   MOV   M_BUF2, A
016A F520                   MOV   BITADD, A
016C 1203DF                 LCALL GET_CMD      ; CALL GET COMMAND

                    ;╔═══════════════════════════════╗
                    ;║           MOVE IN             ║
                    ;║ MOVE GOODS ON LIFT IN TO WARE HOUSE ║
                    ;╚═══════════════════════════════╝
016F                MOVE_I:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
016F 7430      90              MOV   A, #30H           ;0011 0000
0171 90F801    91              MOV   DPTR, #PORTMB     ;DRIVE MOTOR IN
0174 F0        92              MOVX  @DPTR, A
0175 90FC02    93              MOV   DPTR, #PORTSWC    ;READ TO CHECK IN LIMIT
0178 E0        94              MOVX  A, @DPTR
0179 5404      95              ANL   A, #04H           ;0000 0100
017B B404F1    96              CJNE  A, #04H, MOVE_I   ;0000 0100
017E 90F801    97              MOV   DPTR, #PORTMB     ;STOP MOTOR
0181 E4        98              CLR   A
0182 F0        99              MOVX  @DPTR, A
0183 E522     100              MOV   A, SWBUF2         ;READ OLD POSITION
0185 03       101              RR    A                 ;STEP DOWN 1 POS
0186 4480     102              ORL   A, #80H
0188 F522     103              MOV   SWBUF2, A
018A 120224   104              LCALL CTRL_UD
018D 120276   105              LCALL STOP_12           ;STOP MOTOR
0190 90F801   106      MOVE_I1: MOV   DPTR, #PORTMB
0193 7420     107              MOV   A, #20H           ;0010 0000
0195 F0       108              MOVX  @DPTR, A          ;DRIVE MOTOR OUT
0196 90FC02   109              MOV   DPTR, #PORTSWC
0199 E0       110              MOVX  A, @DPTR
019A 5402     111              ANL   A, #02H
019C B402F1   112              CJNE  A, #02H, MOVE_I1  ;0000 0010
019F 90F801   113              MOV   DPTR, #PORTMB     ;STOP MOTOR
01A2 E4       114              CLR   A
01A3 F0       115              MOVX  @DPTR, A
01A4 22       116              RET
              117      ;
              118      ;                MOVE OUT
              119      ;  MOVE GOODS FROM WAREHOUSE ON LIFT
              120      ;
01A5          121      MOVE_O:
01A5 7430     122              MOV   A, #30H           ;0011 0000
01A7 90F801   123              MOV   DPTR, #PORTMB     ;DRIVE MOTOR IN
01AA F0       124              MOVX  @DPTR, A
01AB 90FC02   125              MOV   DPTR, #PORTSWC    ;READ TO CHECK IN LIMIT
01AE E0       126              MOVX  A, @DPTR
01AF 5404     127              ANL   A, #04H  ;0000 0010
01B1 B404F1   128              CJNE  A, #04H, MOVE_O   ;#1111 1011
01B4 90F801   129              MOV   DPTR, #PORTMB      ;STOP MOTOR
01B7 E4       130              CLR   A
01B8 F0       131              MOVX  @DPTR, A
01B9 E522     132              MOV   A, SWBUF2         ;READ OLD POSITION
01BB 23       133              RL    A                 ;STEP UP 1 POS
01BC 4401     134              ORL   A, #01H
01BE F522     135              MOV   SWBUF2, A
01C0 120224   136              LCALL CTRL_UD
01C3 120276   137              LCALL STOP_12           ;STOP MOTOR
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
01C6             138        MOVE_01:
01C6 90F801      139                MOV   DPTR, #PORTMB
01C9 7420        140                MOV   A, #20H            ;0010 0000
01CB F0          141                MOVX  @DPTR, A           ;DRIVE MOTOR OUT
01CC 90FC02      142                MOV   DPTR, #PORTSWC
01CF E0          143                MOVX  A, @DPTR
01D0 5402        144                ANL   A, #02H
01D2 B402F1      145                CJNE  A, #02H, MOVE_01   ;1111 1101
01D5 90F801      146                MOV   DPTR, #PORTMB       ;STOP MOTOR
01D8 E4          147                CLR   A
01D9 F0          148                MOVX  @DPTR, A
01DA 22          149                RET
                 150
                 151        ;╔═══════════════════════════════════════════╗
                 152        ;║              CONTROL LEFT RIGHT           ║
                 153        ;║ CONTROL MOTOR TO MOVE LIFT TO THE POSITION ║
                 154        ;║ DEFINE IN ROW SWBUF1                      ║
                 155        ;╚═══════════════════════════════════════════╝
01DB 90FC00      156        CTRL_LR: MOV   DPTR, #PORTSWA     ;CHECK DIPSW#1
01DE E0          157                 MOVX  A, @DPTR           ;GET DATA
01DF F4          158                 CPL   A
01E0 4480        159                 ORL   A, #80H            ;BIT 7 NO DATA MUST BE MASK
01E2 B52103      160                 CJNE  A, SWBUF1, LR1     ;SWBUF = OFEH
01E5 E4          161                 CLR   A
01E6 4121        162                 AJMP  LR_END             ;POSITION OK
01E8 90FC00      163        LR1:     MOV   DPTR, #PORTSWA     ;READ POSITION AGAIN
01EB E0          164                 MOVX  A, @DPTR
01EC F4          165                 CPL   A
01ED 4480        166                 ORL   A, #80H
01EF B4FF08      167                 CJNE  A, #0FFH, LR2      ;ON SOME POSITION CAN CHECK
01F2 7403        168                 MOV   A, #03H            ;0000 0011
01F4 90F801      169                 MOV   DPTR, #PORTMB
01F7 F0          170                 MOVX  @DPTR, A           ;IT WILL <-----
01F8 21E8        171                 AJMP  LR1
01FA E4          172        LR2:     CLR   A
01FB 90F801      173                 MOV   DPTR, #PORTMB
01FE F0          174                 MOVX  @DPTR, A           ;STOP MOTOR
01FF 90FC00      175                 MOV   DPTR, #PORTSWA
0202 E0          176                 MOVX  A, @DPTR           ;READ POSITION
0203 F4          177                 CPL   A
0204 4480        178                 ORL   A, #80H            ;IF ON CORRECT POSITION
0206 B52105      179                 CJNE  A, SWBUF1,LR_CHK   ;STOP MOTOR AND SAVE MOTOR STATUS
0209 E4          180                 CLR   A
020A F524        181                 MOV   M_BUF1, A
020C 4121        182                 AJMP  LR_END
020E 90FC00      183        LR_CHK:  MOV   DPTR, #PORTSWA
0211 E0          184                 MOVX  A, @DPTR
0212 F4          185                 CPL   A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0213 4480    186            ORL   A, #80H
0215 A921    187            MOV   R1, SWBUF1
0217 99      188            SUBB  A, R1              ;COMPARE DIPSW AND SETDATA
0218 20D704  189            JB    PSW.7, CTRL_L
021B 7402    190            MOV   A, #02H            ;0000 0010 --->
021D 4121    191            AJMP  LR_END
021F 7403    192     CTRL_L: MOV  A, #03H            ;0000 0011 <---
0221 F524    193     LR_END: MOV  M_BUF1, A
0223 22      194            RET                      ;RETURN FROM <----> CONTROL
             195
             196     ;
             197     ;              CONTROL UP DOWN
             198     ; CONTROL MOTOR TO MOVE LIFT TO THE POSITION
             199     ; DEFINE IN COLUMN SWBUF2
             200     ;
0224 90FC01  201     CTRL_UD: MOV  DPTR, #PORTSWB     ;CHECK DIPSW#1
0227 E0      202            MOVX  A, @DPTR           ;GET DATA
0228 F4      203            CPL   A
0229 44C0    204            ORL   A, #0C0H           ;BIT 6,7 NO DATA MUST BE MASK
022B B52203  205            CJNE  A, SWBUF2, UD1     ;SWBUF = ?
022E E4      206            CLR   A
022F 4175    207            AJMP  UD_END             ;POSITION OK
0231 90FC01  208     UD1:   MOV   DPTR, #PORTSWB     ;READ POSITION AGAIN
0234 E0      209            MOVX  A, @DPTR
0235 F4      210            CPL   A
0236 44C0    211            ORL   A, #0C0H
0238 B4FF08  212            CJNE  A, #0FFH, UD2      ;ON SOME POSITION CAN CHECK
023B 740C    213            MOV   A, #0CH            ;0000 11XX
023D 90F801  214            MOV   DPTR, #PORTMB
0240 F0      215            MOVX  @DPTR, A           ;IT WILL MOVE UP
0241 4131    216            AJMP  UD1
0243 E4      217     UD2:   CLR   A
0244 90F801  218            MOV   DPTR, #PORTMB
0247 F0      219            MOVX  @DPTR, A           ;STOP MOTOR
0248 90FC01  220            MOV   DPTR, #PORTSWB
024B E0      221            MOVX  A, @DPTR           ;READ POSITION
024C F4      222            CPL   A
024D 44C0    223            ORL   A, #0C0H
024F B52207  224            CJNE  A, SWBUF2,UD_CHK   ;IF CORRECT POSITION
0252 E4      225            CLR   A                  ;STOP MOTOR AND SAVE MOTOR STATUS
0253 4524    226            ORL   A, M_BUF1
0255 F524    227            MOV   M_BUF1, A
0257 4175    228            AJMP  UD_END
0259 E4      229     UD_CHK: CLR  A
025A 90F801  230            MOV   DPTR, #PORTMB
025D F0      231            MOVX  @DPTR, A           ;STOP MOTOR
025E 90FC01  232            MOV   DPTR, #PORTSWB
0261 E0      233            MOVX  A, @DPTR
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0262 F4       234                 CPL   A
0263 44C0     235                 ORL   A, #0C0H           ;1100 0000
0265 A922     236                 MOV   R1, SWBUF2
0267 99       237                 SUBB  A, R1              ;COMPARE DIPSW AND SETDATA
0268 20D704   238                 JB    PSW.7, CTRL_D
026B 740C     239                 MOV   A, #0CH            ;0000 1100 Up
026D 4171     240                 AJMP  UD3
026F 7408     241       CTRL_D:   MOV   A, #08H            ;0000 1000 Dn
0271 4524     242       UD3:      ORL   A, M_BUF1
0273 F524     243                 MOV   M_BUF1, A
0275 22       244       UD_END:   RET                      ;RETURN FROM Up-Dn CONTROL
              245
              246
              247       ;╔═══════════════════════════════════════════════════╗
              248       ;║                STOP MOTOR 1, 2                    ║
              249       ;║ STOP MOTOR M1, M2 WHEN LIFT MOVE TO CORRECT POS.  ║
              250       ;╚═══════════════════════════════════════════════════╝
0276 E524     251       STOP_12:  MOV   A, M_BUF1          ; START MOTOR1, 2
0278 B40002   252                 CJNE  A, #00H, STOP_
027B 41CA     253                 AJMP  STOP_OK            ; DATA TO DRIVE MOTOR1, 2 = 0000
027D E524     254       STOP_:    MOV   A, M_BUF1
027F 540F     255                 ANL   A, #0FH      ;0000 1111
0281 90F801   256                 MOV   DPTR, #PORTMB
0284 F0       257                 MOVX  @DPTR, A
0285 E4       258                 CLR   A
0286 F520     259                 MOV   BITADD, A          ;CLEAR BIT ADDRESS
0288 90FC00   260       M1:       MOV   DPTR, #PORTSWA     ;START CHECK M1
028B E0       261                 MOVX  A, @DPTR
028C F4       262                 CPL   A
028D 4480     263                 ORL   A, #80H
028F B52112   264                 CJNE  A, SWBUF1, M2      ;M1 ISN'T ON CORRECT POSITION
0292 90F801   265                 MOV   DPTR, #PORTMB      ;M1 OK
0295 740C     266                 MOV   A, #0CH            ;0000 1100
0297 5524     267                 ANL   A, M_BUF1
0299 F524     268                 MOV   M_BUF1, A          ;SAVE M1 STATUS 00
029B F0       269                 MOVX  @DPTR, A           ;STOP M1 ONLY
029C 7401     270                 MOV   A, #01H
029E 4520     271                 ORL   A, BITADD
02A0 5403     272                 ANL   A, #03H    ;0000 0011
02A2 F520     273                 MOV   BITADD, A          ;SET M1 FLAG
02A4 90FC01   274       M2:       MOV   DPTR, #PORTSWB     ;START CHECK M2
02A7 E0       275                 MOVX  A, @DPTR
02A8 F4       276                 CPL   A
02A9 44C0     277                 ORL   A, #0C0H           ;0000 1100
02AB B522DA   278                 CJNE  A, SWBUF2, M1      ;M2 ISN'T ON CORRECT POSITION
02AE 7403     279                 MOV   A, #03H  ;0000 0011
02B0 5524     280                 ANL   A, M_BUF1
02B2 F524     281                 MOV   M_BUF1, A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
02B4 90F801     282             MOV   DPTR, #PORTMB
02B7 F0         283             MOVX  @DPTR, A          ;STOP M2 ONLY
02B8 7402       284             MOV   A, #02H
02BA 4520       285             ORL   A, BITADD
02BC 5403       286             ANL   A, #03H    ;0000 0011
02BE F520       287             MOV   BITADD, A         ;SET M2 FLAG
02C0 7403       288             MOV   A, #03H           ;CHECK FLAG = 0000 0011 ?
02C2 B520C3     289             CJNE  A, BITADD, M1
02C5 E4         290             CLR   A                 ;FLAG IS 0000 0011
02C6 90F801     291             MOV   DPTR, #PORTMB
02C9 F0         292             MOVX  @DPTR, A          ;STOP ALL MOTOR 1,2
02CA 22         293         STOP_OK: RET                ;STOP MOTOR 1,2 COMPLETE
                294
                295
                296         ;╔═══════════╗
                297         ;  DELAY LOOP
                298         ;╚═══════════╝
02CB 7800       299         DELAY:   MOV   R0, #000H
02CD 7900       300         DELAY1:  MOV   R1, #00H
02CF 7A10       301         DELAY2:  MOV   R2, #10H
02D1 DAFE       302                  DJNZ  R2, $
02D3 D9FA       303                  DJNZ  R1, DELAY2
02D5 D8F6       304                  DJNZ  R0, DELAY1
02D7 22         305                  RET
                306
                307         ;╔═══════════════════════════════════════════╗
                308         ;  REPEAT COMMAND WHICH RECIEVE FROM RS-232
                309         ;  EX : RECIEVE  I11  <FROM RS-232>
                310         ;       SEND     %11  <TO RS-232>
                311         ;╚═══════════════════════════════════════════╝
02D8 740D       312         EXECUTE: MOV   A, #13
02DA 12032C     313                  LCALL SBYTE
02DD 7425       314                  MOV   A, #25H   ;25H = '%'
02DF 12032C     315                  LCALL SBYTE
02E2 E522       316                  MOV   A, SWBUF2
02E4 2430       317                  ADD   A, #30H
02E6 12032C     318                  LCALL SBYTE
02E9 E521       319                  MOV   A, SWBUF1
02EB 2430       320                  ADD   A, #30H
02ED 12032C     321                  LCALL SBYTE
02F0 740D       322                  MOV   A, #13
02F2 12032C     323                  LCALL SBYTE
02F5 22         324                  RET
                325
                326         ;╔═══════════════════════════╗
                327         ;  GET PARAMETER ROW AND COLUMN
                328         ;╚═══════════════════════════╝
02F6 120324     329         GETR_C:  LCALL RBYTE   ;GET ROW, COL IN R1, R2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
02F9 F9        330              MOV   R1, A   ;ROW
02FA 120324    331              LCALL RBYTE
02FD FA        332              MOV   R2, A   ;COL
02FE E9        333      CHK_RC: MOV   A, R1          ; ROW IN A
02FF 9431      334              SUBB  A, #31H
0301 20D71D    335              JB    PSW.7, CHK_RC1          ; ROW < 1
0304 F9        336              MOV   R1, A
0305 7402      337              MOV   A, #02
0307 99        338              SUBB  A, R1
0308 20D716    339              JB    PSW.7, CHK_RC1
030B 09        340              INC   R1
030C 8922      341              MOV   SWBUF2, R1
030E EA        342              MOV   A, R2   ; COLUMN IN A
030F 9431      343              SUBB  A, #31H
0311 20D70D    344              JB    PSW.7, CHK_RC1          ; COL < 1
0314 FA        345              MOV   R2, A
0315 7406      346              MOV   A, #6
0317 9A        347              SUBB  A, R2                   ; COL > 7
0318 20D706    348              JB    PSW.7, CHK_RC1
031B 0A        349              INC   R2
031C 8A21      350              MOV   SWBUF1, R2
031E E4        351              CLR   A
031F 6123      352              AJMP  CHK_RC2
0321 74FF      353      CHK_RC1: MOV   A, #0FFH
0323 22        354      CHK_RC2: RET
               355
               356
               357      ;
               358      ; GET DATA 1 BYTE FROM RS-232
               359      ;
0324 3098FD    360      RBYTE:   JNB   RI, $
0327 C298      361               CLR   RI
0329 E599      362               MOV   A, SBUF
032B 22        363               RET
               364
               365      ;
               366      ; SEND DATA 1 BYTE TO RS-232
               367      ;
032C 3099FD    368      SBYTE:   JNB   TI, $       ;WAIT FOR SEND OK
032F C299      369               CLR   TI
0331 F599      370               MOV   SBUF, A
0333 22        371               RET
               372
               373      ;
               374      ;  COMMNUICATION SETUP
               375      ;  4800, N, 8, 1
               376      ;
0334 758920    377      BAUDS:   MOV   TMOD, #20H         ;TIMER 1 MODE 2
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0337 759852   378              MOV   SCON, #52H        ;SERIAL 8 BIT
033A 758DFA   379              MOV   TH1,  #BRAT48      ;BPS
033D D28E     380              SETB  TR1               ;TIMER 1 ON
033F 22       381              RET
              382
              383      ;
              384      ;  CONTROL GOODS ON LIFT TO MOVE IN WAREHOUSE
              385      ;
0340 A822     386      IN:     MOV   R0, SWBUF2        ;CHECK Up-Dn
0342 74FD     387              MOV   A, #0FDH          ;PREPARE 1111 1101 FOR ROTATE
0344 D804     388      IN_1:   DJNZ  R0, IN_2
0346 F522     389              MOV   SWBUF2, A         ;SAVE Up-Dn POSITION SWITCH
0348 614E     390              AJMP  IN_3              ;GOTO CHECK <---> SWITCH
034A 23       391      IN_2:   RL    A
034B 23       392              RL    A
034C 6144     393              AJMP  IN_1
034E A821     394      IN_3:   MOV   R0, SWBUF1
0350 74FE     395              MOV   A, #0FEH          ;PREPARE 1111 1110 FOR ROTATE
0352 D804     396      IN_4:   DJNZ  R0, IN_5
0354 F521     397              MOV   SWBUF1, A         ;SAVE <---> POSITION
0356 615B     398              AJMP  IN_6
0358 23       399      IN_5:   RL    A
0359 6152     400              AJMP  IN_4
035B 1201DB   401      IN_6:   LCALL CTRL_LR
035E 120224   402              LCALL CTRL_UD
0361 120276   403              LCALL STOP_12
0364 12016F   404              LCALL MOVE_I
0367 22       405              RET
              406      ;
              407      ;  MOVE GOODS IN WAREHOUSE ON LIFT
              408      ;
0368 A822     409      OUT:    MOV   R0, SWBUF2        ;CHECK Up-Dn
036A 74FE     410              MOV   A, #0FEH          ;PREPARE 1111 1110 FOR ROTATE
036C D804     411      OUT_1:  DJNZ  R0, OUT_2
036E F522     412              MOV   SWBUF2, A         ;SAVE Up-Dn POSITION SWITCH
0370 6176     413              AJMP  OUT_3             ;GOTO CHECK <---> SWITCH
0372 23       414      OUT_2:  RL    A
0373 23       415              RL    A
0374 616C     416              AJMP  OUT_1
0376 A821     417      OUT_3:  MOV   R0, SWBUF1
0378 74FE     418              MOV   A, #0FEH          ;PREPARE 1111 1110 FOR ROTATE
037A D804     419      OUT_4:  DJNZ  R0, OUT_5
037C F521     420              MOV   SWBUF1, A         ;SAVE <---> POSITION
037E 6183     421              AJMP  OUT_6
0380 23       422      OUT_5:  RL    A
0381 617A     423              AJMP  OUT_4
0383 1201DB   424      OUT_6:  LCALL CTRL_LR
0386 120224   425              LCALL CTRL_UD
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
0389 120276     426                 LCALL STOP_12
038C 1201A5     427                 LCALL MOVE_O
038F 22         428                 RET
                429
                430         ;╔═══════════════════════════════════╗
                431         ;║ MOVE WAREHOUSE TO ROW, COL 1, 1   ║
                432         ;╚═══════════════════════════════════╝
0390 7521FE     433         HOME:   MOV   SWBUF1, #0FEH          ;1111 1110  <---
0393 7522FE     434                 MOV   SWBUF2, #0FEH          ;1111 1110  Dn
0396 1201DB     435                 LCALL CTRL_LR               ;CONTROL <---->
0399 120224     436                 LCALL CTRL_UD               ;CONTROL Up-Dn
039C 120276     437                 LCALL STOP_12               ;STOP ON CORRECT POSITION
039F 22         438                 RET
                439
                440         ;╔═══════════════════════╗
                441         ;║ SEND PROMPT TO IBM-PC ║
                442         ;╚═══════════════════════╝
03A0 900438     443         PROMPT: MOV   DPTR, #READY
03A3 7800       444                 MOV   R0, #0
03A5 E8         445         PROMPT1: MOV  A, R0
03A6 93         446                 MOVC  A, @A+DPTR
03A7 12032C     447                 LCALL SBYTE
03AA 08         448                 INC   R0
03AB B43EF7     449                 CJNE  A, #3EH, PROMPT1
03AE 22         450                 RET
                451
                452
                453         ;╔═══════════════════════════════════╗
                454         ;║ MOVE GOODS ON CONVEYER TO LIFT    ║
                455         ;╚═══════════════════════════════════╝
03AF 120390     456         COV_I:  LCALL HOME
03B2 74C0       457         COV_I1: MOV   A, #0C0H          ;SLIDE CONV IN TO LIFT
03B4 90F801     458                 MOV   DPTR, #PORTMB
03B7 F0         459                 MOVX  @DPTR, A
03B8 90FC02     460                 MOV   DPTR, #PORTSWC
03BB E0         461                 MOVX  A, @DPTR
03BC 5408       462                 ANL   A, #08H
03BE B408F1     463                 CJNE  A, #08H, COV_I1 ; SW 17 NO DETECT
03C1 E4         464                 CLR   A                 ;STOP M3
03C2 90F801     465                 MOV   DPTR, #PORTMB
03C5 F0         466                 MOVX  @DPTR, A
03C6 22         467                 RET
                468
                469         ;╔═══════════════════════════════════╗
                470         ;║ MOVE GOODS ON LIFT TO CONVEYER    ║
                471         ;╚═══════════════════════════════════╝
03C7 120390     472         COV_O:  LCALL HOME
03CA 7480       473         COV_O2: MOV   A, #080H          ;SLIDE CONV OUT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
03CC 90F801     474             MOV   DPTR, #PORTMB  ;UNTIL SW 18 DETECT
03CF F0         475             MOVX  @DPTR, A
03D0 90FC02     476             MOV   DPTR, #PORTSWC
03D3 E0         477             MOVX  A, @DPTR
03D4 5410       478             ANL   A, #10H
03D6 B410F1     479             CJNE  A, #10H, COV_O2 ;SW 18 NOT DETECT
03D9 E4         480             CLR   A               ;STOP M3
03DA 90F801     481             MOV   DPTR, #PORTMB
03DD F0         482             MOVX  @DPTR, A
03DE 22         483             RET
                484
                485     ;┌──────────────────────────────────────┐
                486     ;│ GET COMMAND FROM RS-232              │
                487     ;│ Irc = MOVE GOODS IN ROW = r, COL = c │
                488     ;│ Orc = MOVE GOODS FROM ROW = r, COL = c│
                489     ;│ H   = HOME                           │
                490     ;│ S   = SEND GOODS ON LIFT TO CONVEYER │
                491     ;│ G   = GET  GOODS ON LIFT TO CONVEYER │
                492     ;└──────────────────────────────────────┘
03DF            493     GET_CMD:
03DF 1203A0     494             LCALL PROMPT
03E2 120324     495             LCALL RBYTE
03E5 F8         496             MOV   R0, A
03E6 B8490E     497     CMD_I:  CJNE  R0, #49H, CMD_O    ;49H = 'I'
03E9 1202F6     498             LCALL GETR_C
03EC 70F1       499             JNZ   GET_CMD
03EE 1202D8     500             LCALL EXECUTE
03F1 120340     501             LCALL IN
03F4 0203DF     502             LJMP  GET_CMD
03F7 B84F0E     503     CMD_O:  CJNE  R0, #4FH, CMD_H    ;4FH = 'O'
03FA 1202F6     504             LCALL GETR_C
03FD 70E0       505             JNZ   GET_CMD
03FF 1202D8     506             LCALL EXECUTE
0402 120368     507             LCALL OUT
0405 0203DF     508             LJMP  GET_CMD
0408 B8480E     509     CMD_H:  CJNE  R0, #48H, CMD_G    ; 48H = 'H'
040B 1202F6     510             LCALL GETR_C
040E 70CF       511             JNZ   GET_CMD
0410 1202D8     512             LCALL EXECUTE
0413 120390     513             LCALL HOME
0416 0203DF     514             LJMP  GET_CMD
0419 B84706     515     CMD_G:  CJNE  R0, #47H, CMD_S    ;GET FROM  CONV
041C 1203AF     516             LCALL COV_I
041F 0203DF     517             LJMP  GET_CMD
0422 B85306     518     CMD_S:  CJNE  R0, #53H, CMD_ERR ;SEND TO CONV
0425 1203C7     519             LCALL COV_O
0428 0203DF     520             LJMP  GET_CMD
042B 7445       521     CMD_ERR: MOV  A, #45H  ; NO COMMAND
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
042D 12032C     522                 LCALL SBYTE     ; SEND "E<CR>"
0430 740D       523                 MOV   A, #13
0432 12032C     524                 LCALL SBYTE
0435 0203DF     525                 LJMP  GET_CMD
                526
                527         ; COMMAND PROMPT
0438 436F6D6D   528         READY:  DB    'Command..>'
043C 616E642E
0440 2E3E
                529
0000=           530                 END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   bauds = 0334      bitadd = 0020      brat24 = 00F4
  brat48 = 00FA      brat96 = 00FD      chk_rc = 02FE
 chk_rc1 = 0321     chk_rc2 = 0323     cmd_err = 042B
   cmd_g = 0419       cmd_h = 0408       cmd_i = 03E6
   cmd_o = 03F7       cmd_s = 0422       cov_i = 03AF
  cov_i1 = 03B2       cov_o = 03C7      cov_o2 = 03CA
  ctrl_d = 026F      ctrl_l = 021F     ctrl_lr = 01DB
 ctrl_ud = 0224     cw_iiii = 009B     cw_oooo = 0080
   delay = 02CB      delay1 = 02CD      delay2 = 02CF
 execute = 02D8     get_cmd = 03DF      getr_c = 02F6
  hexbuf = 002D         home = 0390          in = 0340
    in_1 = 0344         in_2 = 034A        in_3 = 034E
    in_4 = 0352         in_5 = 0358        in_6 = 035B
  keybuf = 0030          lr1 = 01E8         lr2 = 01FA
  lr_chk = 020E       lr_end = 0221          m1 = 0288
      m2 = 02A4      m_buf1 = 0024      m_buf2 = 0025
    main = 015F      move_i = 016F     move_i1 = 0190
  move_o = 01A5     move_o1 = 01C6         out = 0368
   out_1 = 036C        out_2 = 0372       out_3 = 0376
   out_4 = 037A        out_5 = 0380       out_6 = 0383
  portin = FC00       portma = F800      portmb = F801
  portmc = F802      portmcw = F803     portout = F800
 portswa = FC00      portswb = FC01     portswc = FC02
portswcw = FC03     power_on = 0000      prompt = 03A0
 prompt1 = 03A5        rbyte = 0324       ready = 0438
     res = 0100         res1 = 0102       sbyte = 032C
   stack = 006C        start = 0150       stop_ = 027D
 stop_12 = 0276     stop_ok = 02CA     swbuf1 = 0021
 swbuf2 = 0022      swbuf3 = 0023        ud1 = 0231
     ud2 = 0243         ud3 = 0271     ud_chk = 0259
  ud_end = 0275
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุป

5.1 สรุปผลการทดลอง

การจัดเก็บสินค้าด้วยคอมพิวเตอร์ ในส่วนของการพัฒนาการควบคุม WAREHOUSE ด้วยการใช้ MICRO CONTROLLER นี้ ในส่วนของ HARDWARE และ SOFTWARE นี้ สามารถที่จะทำงานได้ในระดับหนึ่ง

5.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

ในส่วนของ WAREHOUSE เนื่องจากไม่มี DATA SHEET เราจึงต้องศึกษาการทำงานทางด้าน SENSOR SWITCH ตำแหน่งของ SWITCH แต่ละตัวและอุปกรณ์บางส่วนได้หลุดหรือขาดหายไป เราจึงทำการแก้ปัญหาด้วยการนำ LEGO ที่มีเหลืออยู่มาประยุกต์ใช้งานตามส่วนต่างๆ

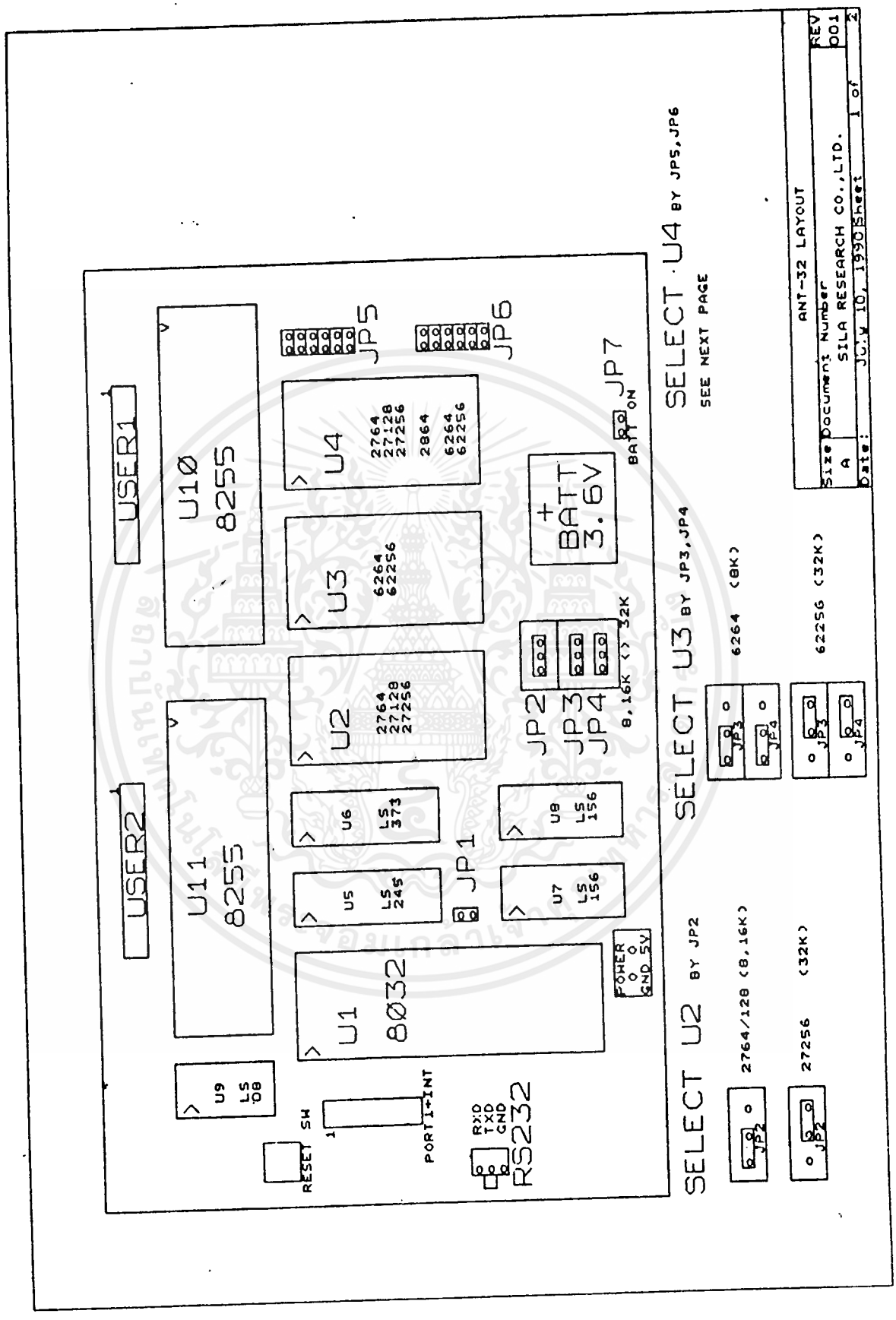
ในส่วนของทางด้าน SOFTWARE เนื่องจาก MICRO CONTROLLER ตระกูล MCS-51 เป็น CPU ตระกูลใหม่ ดังนั้นในการเขียนโปรแกรมจึงไม่สะดวกเท่าที่ควร.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

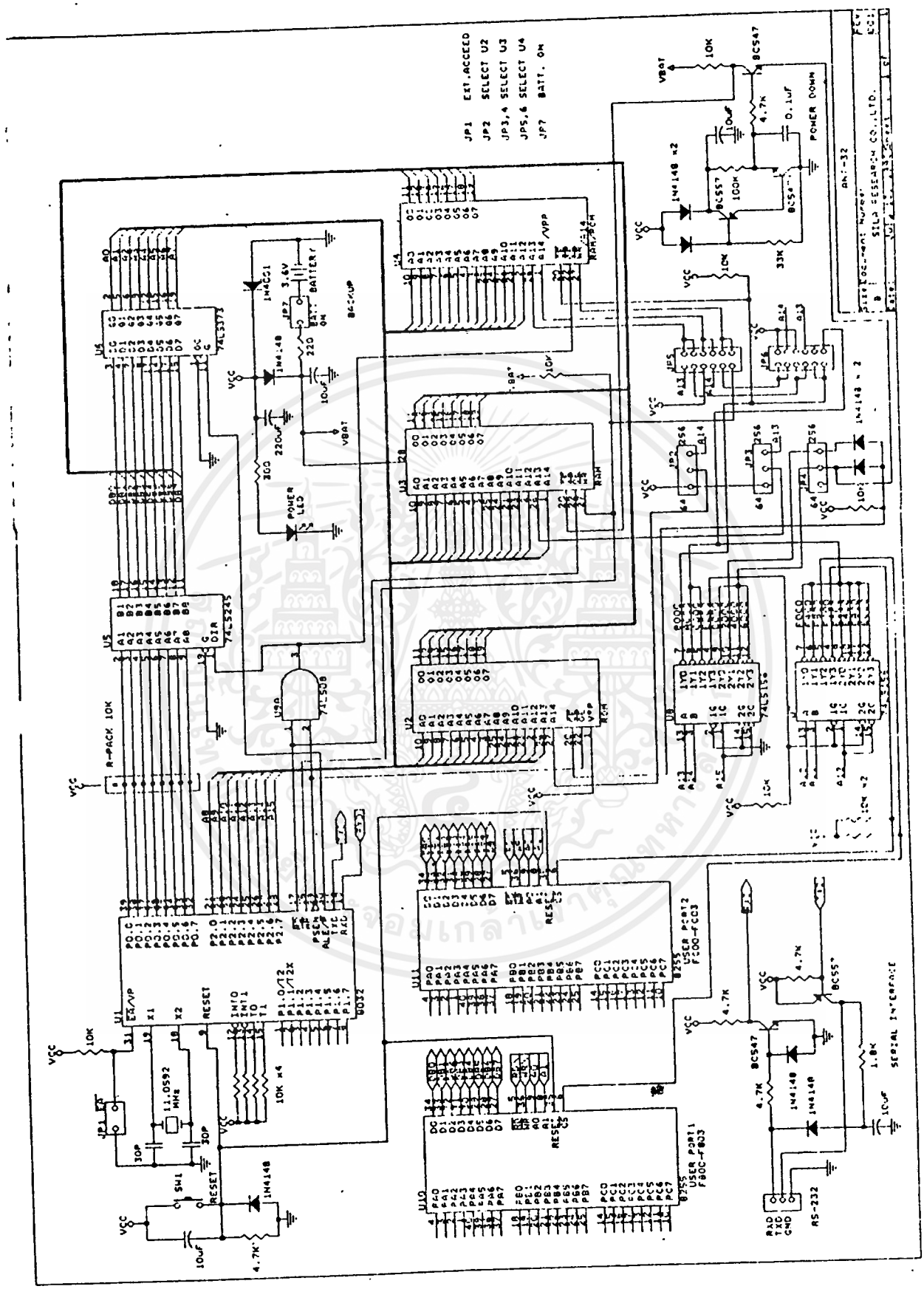
ANT-32 SPECIFICATION

CPU:	8032
CPU CLOCK:	11.0592 MHz
INTERNAL MEMORY:	256 BYTE (RAM)
EXT.CODE MEMORY:	(U2) 8-32K SELECT 2764,27128,27256 (ROM)
EXT.DATA MEMORY:	(U3) 8-32K SELECT 6264,62256 (RAM)
EXT.CODE & DATA MEMORY:	(U4) 8-30K SELECT 2764,27128,27256 (ROM)
	2864 (EEPROM)
	6264,62256 (RAM)
INTERNAL PORT:	12 BIT I/O
EXTERNAL PORT:	USER1 8255 PORT I/O 24 BIT
	USER2 8255 PORT I/O 24 BIT
BACKUP:	DATA MEMORY (U3) 52 HOUR
CHARGE TIME:	48 HOUR
LANGUAGE:	MCS BASIC-52
	ASSEMBLY (BY DOWNLOAD HEX FILE)
CONNECTOR:	16P INTERNAL PORT
	26P USER1 PORT
	26P USER2 PORT
	2P POWER SUPPLY
	3P SERIAL INTERFACE (RS232)
SERIAL INTERFACE:	RS232C
POWER:	5V DC 290mA (U4 NOT INCLUDE)
SIZE:	5.25" x 3.9"
OPTION:	BATTERY NI-CAD 3.6V 60mA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

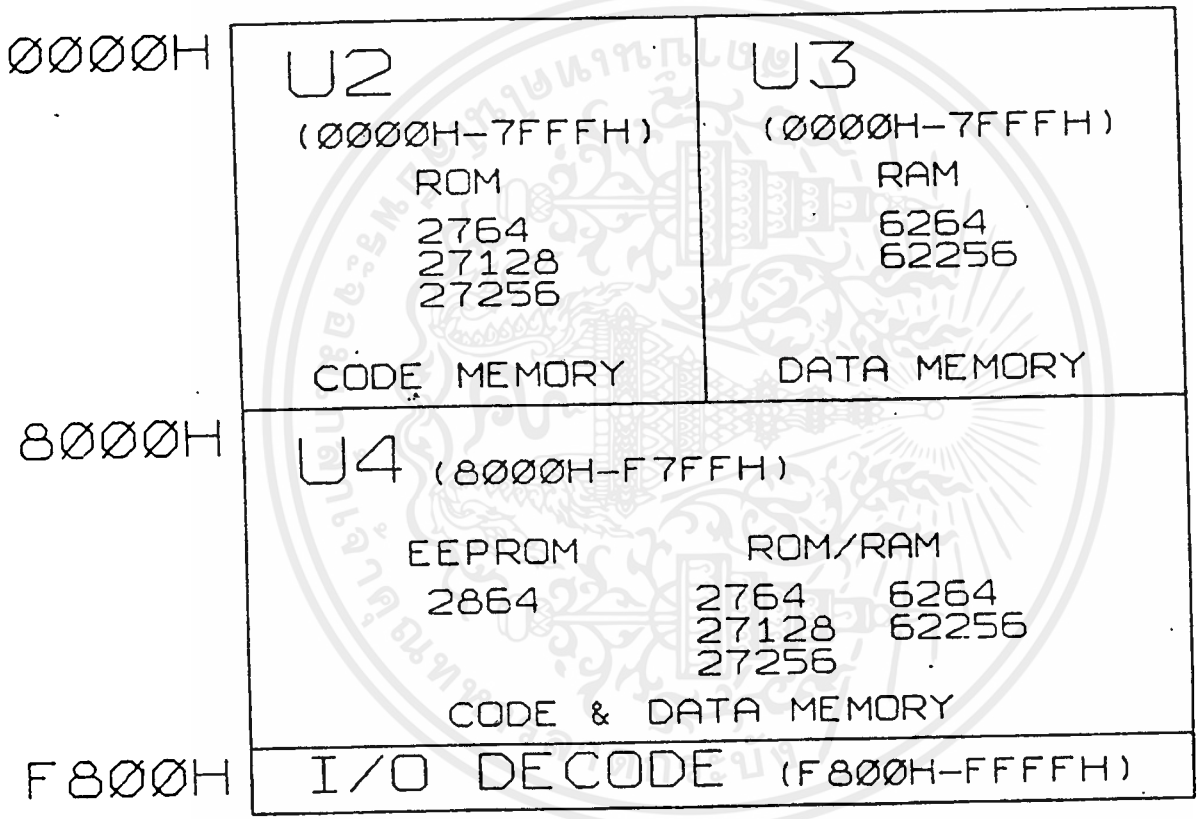


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

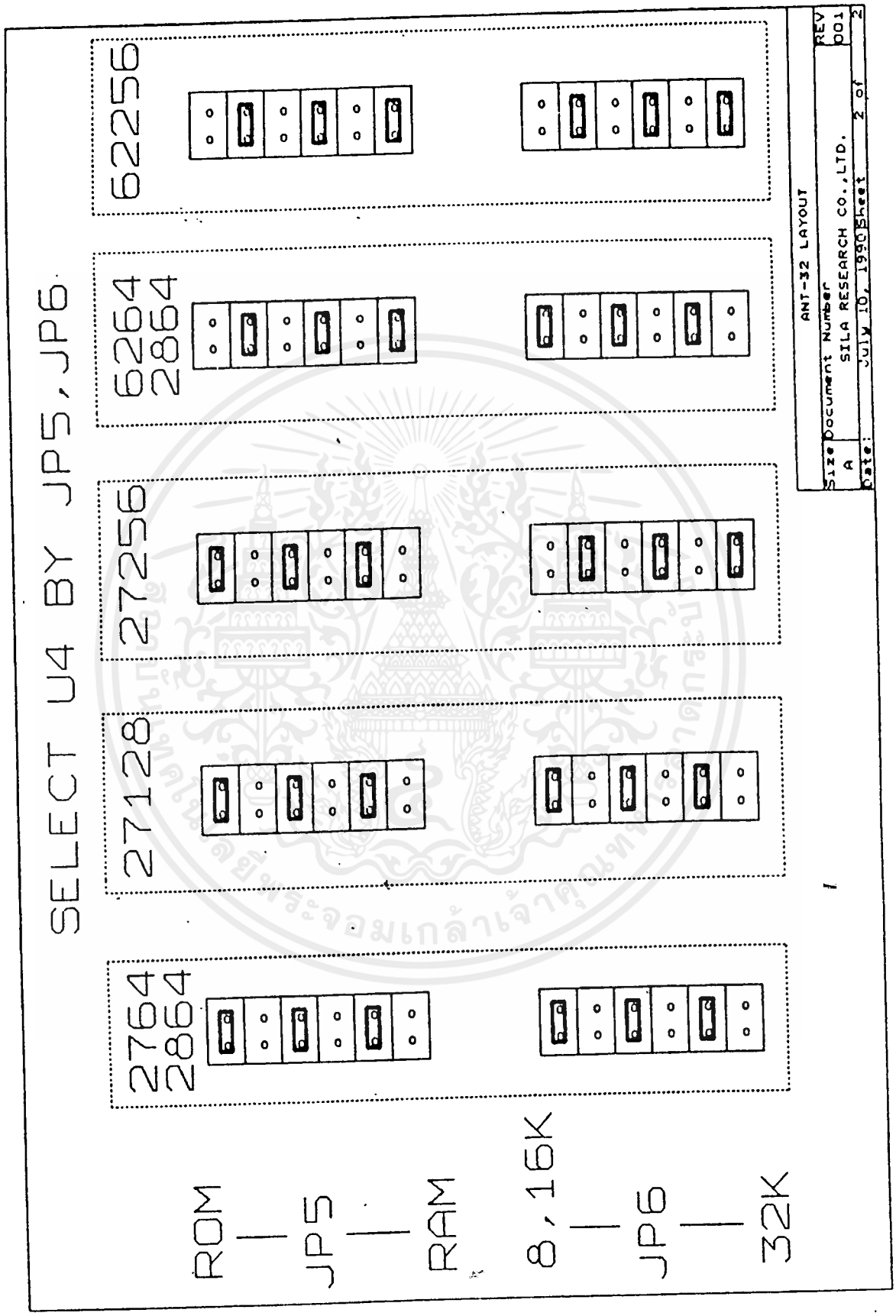


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

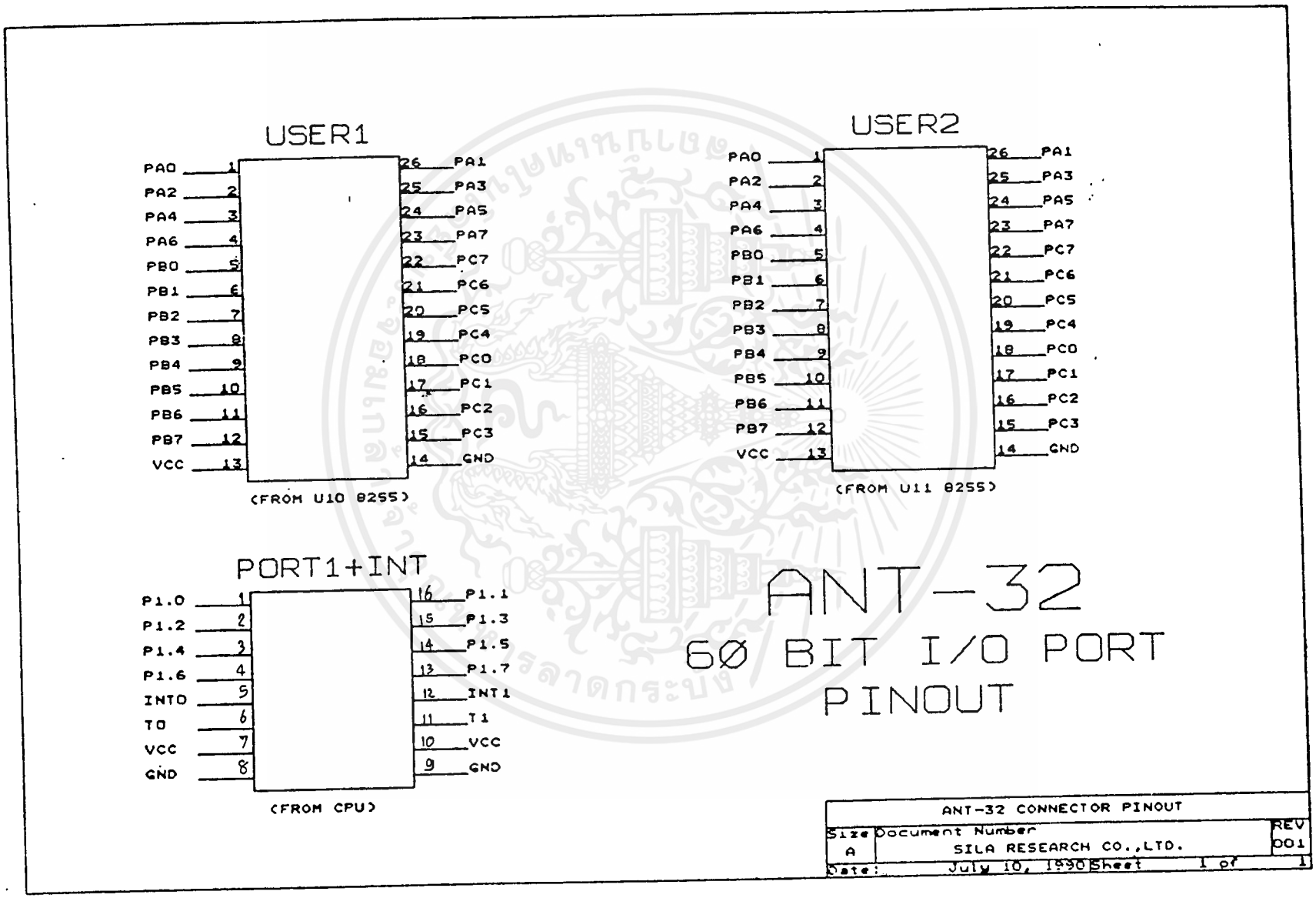
ANT-32
MEMORY MAP



ANT-32 MEMORY MAP		
Size A	Document Number SILA RESEARCH CO.,LTD.	REV 001
Date: July 10, 1990	Sheet 1 of 1	



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


USER1
PA0 1
PA2 2
PA4 3
PA6 4
PB0 5
PB1 6
PB2 7
PB3 8
PB4 9
PB5 10
PB6 11
PB7 12
VCC 13
26 PA1
25 PA3
24 PA5
23 PA7
22 PC7
21 PC6
20 PC5
19 PC4
18 PC0
17 PC1
16 PC2
15 PC3
14 GND
(FROM U10 8255)
USER2
PA0 1
PA2 2
PA4 3
PA6 4
PB0 5
PB1 6
PB2 7
PB3 8
PB4 9
PB5 10
PB6 11
PB7 12
VCC 13
26 PA1
25 PA3
24 PA5
23 PA7
22 PC7
21 PC6
20 PC5
19 PC4
18 PC0
17 PC1
16 PC2
15 PC3
14 GND
(FROM U11 8255)
PORT1+INT
P1.0 1
P1.2 2
P1.4 3
P1.6 4
INT0 5
T0 6
VCC 7
GND 8
16 P1.1
15 P1.3
14 P1.5
13 P1.7
12 INT1
11 T1
10 VCC
9 GND
(FROM CPU)
ANT-32
60 BIT I/O PORT
PINOUT
ANT-32 CONNECTOR PINOUT
Size A
Document Number
SILA RESEARCH CO.,LTD.
REV 001
Date: July 10, 1990 Sheet 1 of 1

8255 MODE Ø SUMMARY

CONTROL CODE IN MODE Ø

PORT A	PORT B	PORT CØ-C3	PORT C4-C7	CODE(HEX)
0	0	0	0	8Ø
0	0	0	1	88
0	0	1	0	81
0	0	1	1	89
0	1	0	0	82
0	1	0	1	8A
0	1	1	0	83
0	1	1	1	8B
1	0	0	0	90
1	0	0	1	98
1	0	1	0	91
1	0	1	1	99
1	1	0	0	92
1	1	0	1	9A
1	1	1	0	93
1	1	1	1	9B

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

intel

HARDWARE DESCRIPTION OF THE 8051, 8052 AND 80C51

INTRODUCTION

This chapter presents a comprehensive description of the on-chip hardware features of the MCS®-51 microcontrollers. Included in this description are

- The port drivers and how they function both as ports and, for Ports 0 and 2, in bus operations
- The Timer/Counters
- The Serial Interface
- The Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices
- The EPROM versions of the 8051AH, 8052AH, and 80C51BH

The devices under consideration are listed in Table 1. As it becomes unwieldy to be constantly referring to each of these devices by their individual names, we will adopt a convention of referring to them generically as 8051s and 8052s, unless a specific member of the group is being referred to, in which case it will be specifically named. The "8051s" include the 8051, 8051AH, and 80C51BH, and their ROMless and EPROM versions. The "8052s" are the 8052AH, 8032AH, and 8752BH.

Figure 1 shows a functional block diagram of the 8051s and 8052s.

Table 1. The MCS-51 Family of Microcontrollers

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-bit Timers	Ckt Type
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Special Function Registers

A map of the on-chip memory area called SFR (Special Function Register) space is shown in Figure 2. SFRs marked by parentheses are resident in the 8052s but not in the 8051s.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

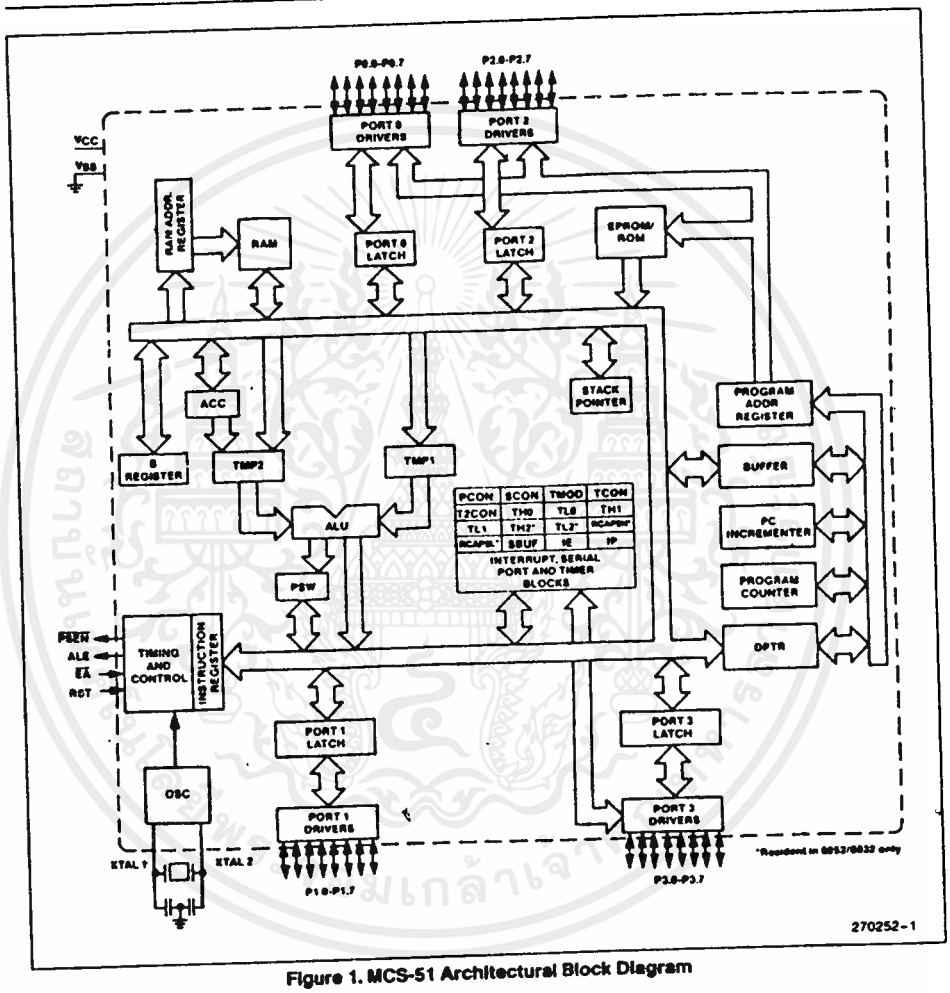


Figure 1. MCS-51 Architectural Block Diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Figure 2. SFR Map. (. . .) Indicates Resident in 8052s, not in 8051s

Note that not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in future MCS-51 products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

The functions of the SFRs are outlined below.

ACCUMULATOR

ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B REGISTER

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW register contains program status information as detailed in Figure 3.

STACK POINTER

The Stack Pointer Register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 3

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER REGISTERS

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit Counting registers for Timer/Counters 0, 1, and 2, respectively.

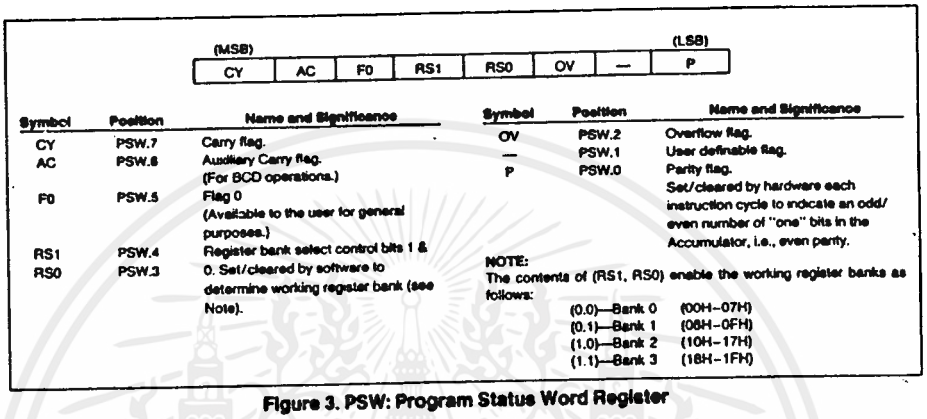
CAPTURE REGISTERS

The register pair (RCAP2H, RCAP2L) are the Capture registers for the Timer 2 "Capture Mode." In this mode, in response to a transition at the 8052's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode. More about Timer 2's features in a later section.

CONTROL REGISTERS

Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port. They are described in later sections.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Symbol	Position	Name and Significance
CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary Carry flag. (For BCD operations.)
F0	PSW.5	Flag 0 (Available to the user for general purposes.)
RS1	PSW.4	Register bank select control bits 1 &
RS0	PSW.3	0. Set/cleared by software to determine working register bank (see Note).

Symbol	Position	Name and Significance
OV	PSW.2	Overflow flag.
—	PSW.1	User definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/ even number of "one" bits in the Accumulator, i.e., even parity.

NOTE:
The contents of (RS1, RS0) enable the working register banks as follows:

(0.0)—Bank 0	(00H–07H)
(0.1)—Bank 1	(08H–0FH)
(1.0)—Bank 2	(10H–17H)
(1.1)—Bank 3	(18H–1FH)

Figure 3. PSW: Program Status Word Register

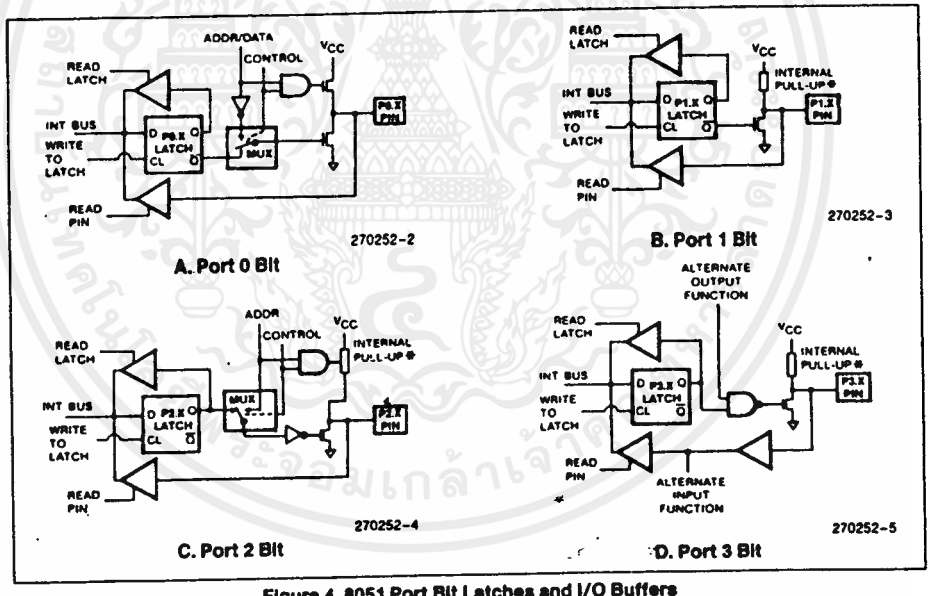


Figure 4. 8051 Port Bit Latches and I/O Buffers
*See Figure 5 for details of the internal pullup.

PORT STRUCTURES AND OPERATION

All four ports in the 8051 are bidirectional. Each consists of a latch (Special Function Registers P0 through P3), an output driver, and an input buffer.

The output drivers of Ports 0 and 2, and the input buffers of Port 0, are used in accesses to external memory. In this application, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise the Port 2 pins continue to emit the P2 SFR content.

All the Port 3 pins, and (in the 8052) two Port 1 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed on the following page.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Port Pin	Alternate Function
*P1.0	T2 (Timer/Counter 2 external input)
*P1.1	T2EX (Timer/Counter 2 Capture/Reload trigger)
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt)
P3.3	$\overline{INT1}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{WR}$ (external Data Memory write strobe)
P3.7	$\overline{RD}$ (external Data Memory read strobe)

*P1.0 and P1.1 serve these alternate functions only on the 8052.

The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise the port pin is stuck at 0.

I/O Configurations

Figure 4 shows a functional diagram of a typical bit latch and I/O buffer in each of the four ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal. More about that later.

As shown in Figure 4, the output drivers of Ports 0 and 2 are switchable to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses. During external memory accesses, the P2 SFR remains unchanged, but the P0 SFR gets 1s written to it.

Also shown in Figure 4, is that if a P3 bit latch contains a 1, then the output level is controlled by the signal labeled "alternate output function." The actual P3.X pin level is always available to the pin's alternate input function, if any.

Ports 1, 2, and 3 have internal pullups. Port 0 has open drain outputs. Each I/O line can be independently used as an input or an output. (Ports 0 and 2 may not be used as general purpose I/O when being used as the ADDR/DATA BUS). To be used as an input, the port bit latch must contain a 1, which turns off the output driver FET. Then, for Ports 1, 2, and 3, the pin is pulled high by the internal pullup, but can be pulled low by an external source.

Port 0 differs in not having internal pullups. The pullup FET in the P0 output driver (see Figure 4) is used only when the Port is emitting 1s during external memory accesses. Otherwise the pullup FET is off. Consequently P0 lines that are being used as output port lines are open drain. Writing a 1 to the bit latch leaves both output FETs off, so the pin floats. In that condition it can be used a high-impedance input.

Because Ports 1, 2, and 3 have fixed internal pullups they are sometimes called "quasi-bidirectional" ports. When configured as inputs they pull high and will source current (I_{IL}, in the data sheets) when externally pulled low. Port 0, on the other hand, is considered "true" bidirectional, because when configured as an input it floats.

All the port latches in the 8051 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are in fact sampled by their output buffers only during Phase 1 of any clock period. (During Phase 2 the output buffer holds the value it saw during the previous Phase 1). Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle. See Figure 39 in the Internal Timing section.

If the change requires a 0-to-1 transition in Port 1, 2, or 3, an additional pullup is turned on during S1P1 and S1P2 of the cycle in which the transition occurs. This is done to increase the transition speed. The extra pullup can source about 100 times the current that the normal pullup can. It should be noted that the internal pullups are field-effect transistors, not linear resistors. The pullup arrangements are shown in Figure 5.

In HMOS versions of the 8051, the fixed part of the pullup is a depletion-mode transistor with the gate wired to the source. This transistor will allow the pin to source about 0.25 mA when shorted to ground. In parallel with the fixed pullup is an enhancement-mode transistor, which is activated during S1 whenever the port bit does a 0-to-1 transition. During this interval, if the port pin is shorted to ground, this extra transistor will allow the pin to source an additional 30 mA.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

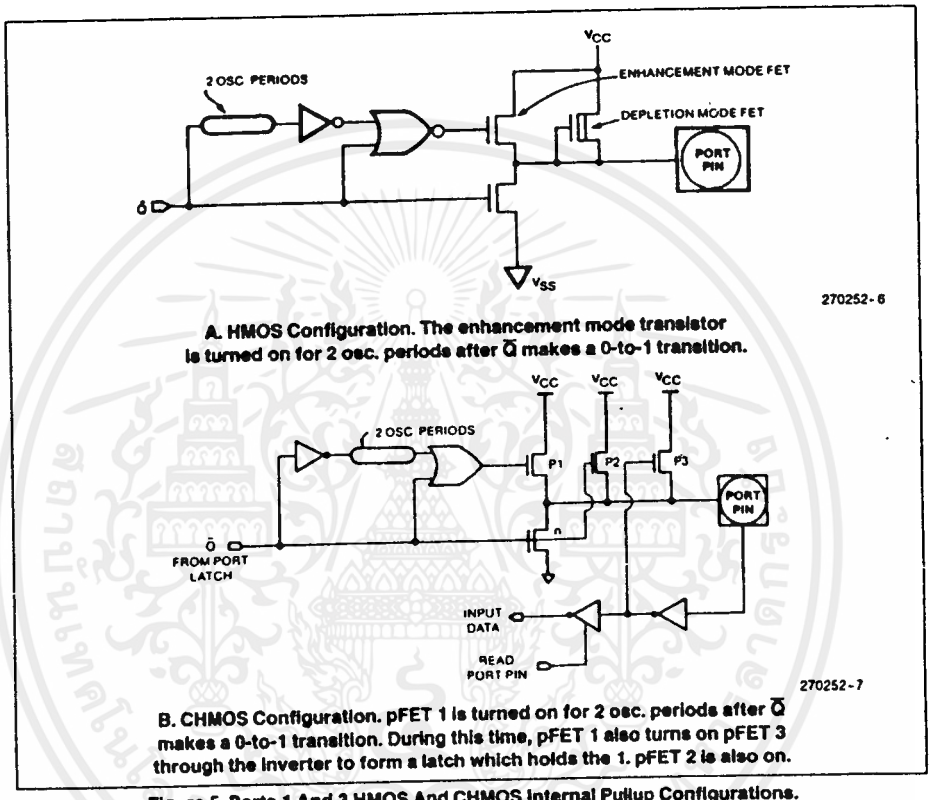


Figure 5. Ports 1 And 3 HMOS And CHMOS Internal Pullup Configurations. Port 2 Is Similar Except That It Holds The Strong Pullup On While Emitting 1s That Are Address Bits. (See Text, "Accessing External Memory".)

In the CHMOS versions, the pullup consists of three pFETs. It should be noted that an n-channel FET (nFET) is turned on when a logical 1 is applied to its gate, and is turned off when a logical 0 is applied to its gate. A p-channel FET (pFET) is the opposite: it is on when its gate sees a 0, and off when its gate sees a 1.

pFET1 in Figure 5 is the transistor that is turned on for 2 oscillator periods after a 0-to-1 transition in the port latch. While it's on, it turns on pFET3 (a weak pull-up), through the inverter. This inverter and pFET form a latch which hold the 1.

Note that if the pin is emitting a 1, a negative glitch on the pin from some external source can turn off pFET3, causing the pin to go into a float state. pFET2 is a very weak pullup which is on whenever the nFET is off, in traditional CMOS style. It's only about $1/_{10}$ the strength of pFET3. Its function is to restore a 1 to the pin in the event the pin *had* a 1 and lost it to a glitch.

Port Loading and Interfacing

The output buffers of Ports 1, 2, and 3 can each drive 4 LS TTL inputs. These ports on HMOS versions can be driven in a normal manner by any TTL or NMOS circuit. Both HMOS and CHMOS pins can be driven by open-collector and open-drain outputs, but note that 0-to-1 transitions will not be fast. In the HMOS device, if the pin is driven by an open-collector output, a 0-to-1 transition will have to be driven by the relatively weak depletion mode FET in Figure 5(A). In the CHMOS device, an input 0 turns off pullup pFET3, leaving only the very weak pullup pFET2 to drive the transition.

In external bus mode, Port 0 output buffers can each drive 8 LS TTL inputs. As port pins, they require external pullups to drive any inputs.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do which? The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

ANL	(logical AND, e.g., ANL P1, A)
ORL	(logical OR, e.g., ORL P2, A)
XRL	(logical EX-OR, e.g., XRL P3, A)
JBC	(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL	(complement bit, e.g., CPL P3.0)
INC	(increment, e.g., INC P2)
DEC	(decrement, e.g., DEC P2)
DJNZ	(decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV, PX.Y, C	(move carry bit to bit Y of Port X)
CLR PX.Y	(clear bit Y of Port X)
SETB PX.Y	(set bit Y of Port X)

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

ACCESSING EXTERNAL MEMORY

Accesses to external memory are of two types: accesses to external Program Memory and accesses to external Data Memory. Accesses to external Program Memory use signal PSEN (program store enable) as the read strobe. Accesses to external Data Memory use $\overline{RD}$ or $\overline{WR}$ (alternate functions of P3.7 and P3.6) to strobe the memory. Refer to Figures 36 through 38 in the Internal Timing section.

Fetches from external Program Memory always use a 16-bit address. Accesses to external Data Memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

Whenever a 16-bit address is used, the high byte of the address comes out on Port 2, where it is held for the duration of the read or write cycle. Note that the Port 2 drivers use the strong pullups during the entire time that they are emitting address bits that are 1s. This is during the execution of a MOVX @DPTR instruction. During this time the Port 2 latch (the Special Function Register) does not have to contain 1s, and the contents of the Port 2 SFR are not modified. If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the Port 2 SFR will reappear in the next cycle.

If an 8-bit address is being used (MOVX @Ri), the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signal drives both FETs in the Port 0 output buffers. Thus, in this application the Port 0 pins are not open-drain outputs, and do not require external pullups. Signal ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before $\overline{WR}$ is activated, and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding. If the user writes to Port 0 during an external memory fetch, the incoming code byte is corrupted. Therefore, do not write to Port 0 if external program memory is used.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{EA}$ is active; or
2) Whenever the program counter (PC) contains a number that is larger than 0FFFH (1FFFH for the 8052).

This requires that the ROMless versions have $\overline{EA}$ wired low to enable the lower 4K (8K for the 8032) program bytes to be fetched from external memory.

When the CPU is executing out of external Program Memory, all 8 bits of Port 2 are dedicated to an output function and may not be used for general purpose I/O. During external program fetches they output the high byte of the PC. During this time the Port 2 drivers use the strong pullups to emit PC bits that are 1s.

TIMER/COUNTERS

The 8051 has two 16-bit Timer/Counter registers: Timer 0 and Timer 1. The 8052 has these two plus one

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

more: Timer 2. All three can be configured to operate either as timers or event counters.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is $1/12$ of the oscillator frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or (in the 8052) T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is $1/24$ of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

In addition to the "Timer" or "Counter" selection, Timer 0 and Timer 1 have four operating modes from which to select. Timer 2, in the 8052, has three modes of operation: "Capture," "Auto-Reload" and "baud rate generator."

Timer 0 and Timer 1

These Timer/Counters are present in both the 8051 and the 8052. The "Timer" or "Counter" function is selected by control bits C/$\overline{T}$ in the Special Function Register TMOD (Figure 6). These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for both Timer/Counters. Mode 3 is different. The four operating modes are described in the following text.

MODE 0

Either Timer in Mode 0 is an 8-bit Counter with a divide-by-32 prescaler. This 13-bit timer is MCS-48 compatible. Figure 7 shows the Mode 0 operation as it applies to Timer 1.

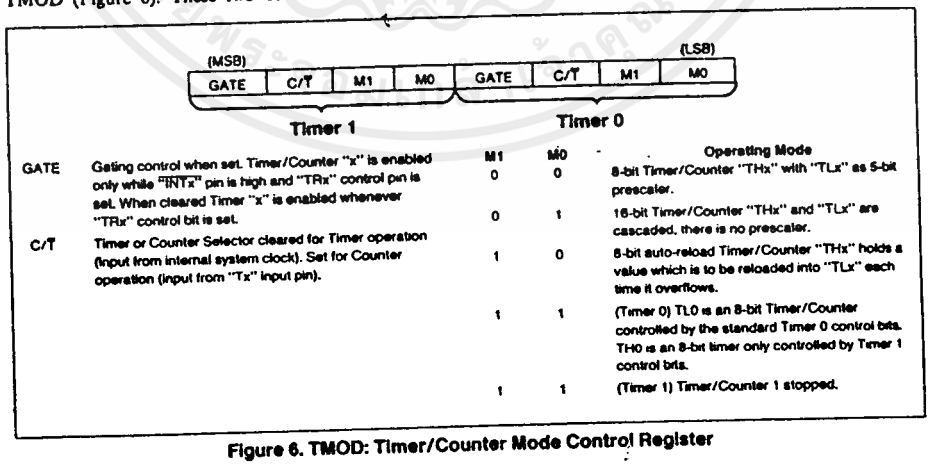
In this mode, the Timer register is configured as a 13-Bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or $\overline{INT1}$ = 1. (Setting GATE = 1 allows the Timer to be controlled by external input $\overline{INT1}$, to facilitate pulse width measurements.) TR1 is a control bit in the Special Function Register TCON (Figure 8). GATE is in TMOD.

The 13-Bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag (TR1) does not clear the registers.

Mode 0 operation is the same for Timer 0 as for Timer 1. Substitute TR0, TF0 and $\overline{INT0}$ for the corresponding Timer 1 signals in Figure 7. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

MODE 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.



GATE	Gating control when set. Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	Timer or Counter Selector cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating Mode
0	0	8-bit Timer/Counter "THx" with "TLx" as 5-bit prescaler.
0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded, there is no prescaler.
1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 stopped.

Figure 6. TMOD: Timer/Counter Mode Control Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

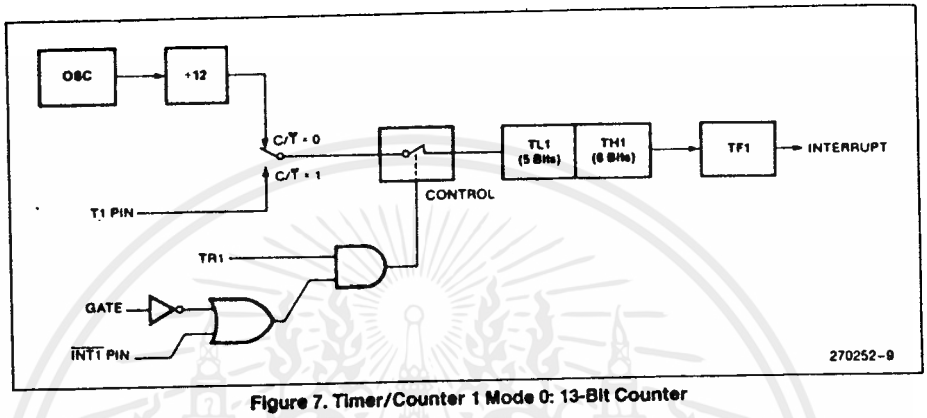


Figure 7. Timer/Counter 1 Mode 0: 13-Bit Counter

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

Figure 8.TCON: Timer/Counter Control Register

MODE 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 9. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

Mode 2 operation is the same for Timer/Counter 0.

MODE 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 10. TL0 uses the Timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, an 8051 can look like it has three Timer/Counters, and an 8052, like it has four. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

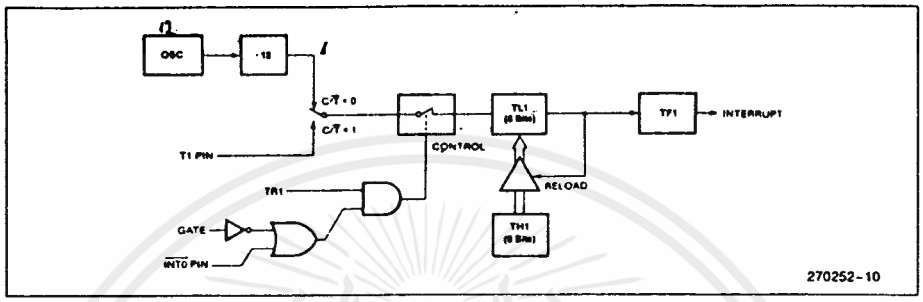


Figure 9. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload

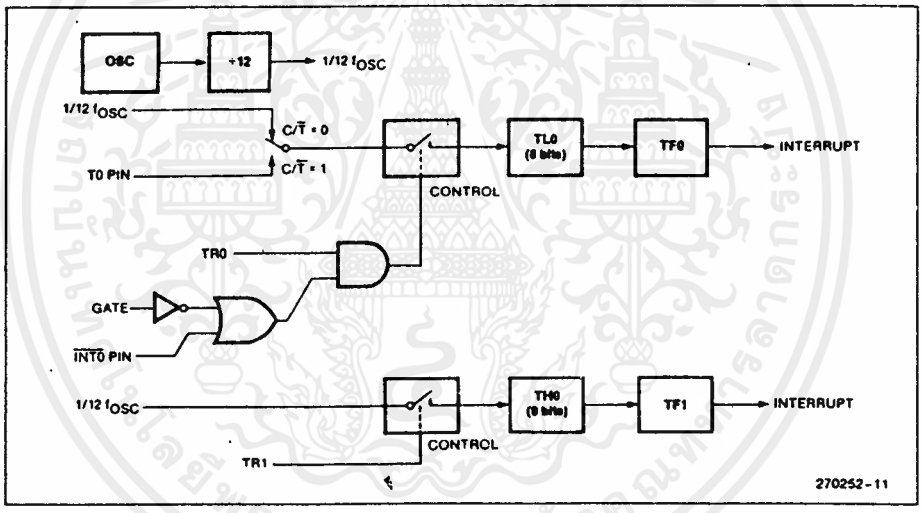


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Timer 2

Timer 2 is a 16-bit Timer/Counter which is present only in the 8052. Like Timers 0 and 1, it can operate either as a timer or as an event counter. This is selected by bit C/$\overline{T2}$ in the Special Function Register T2CON (Figure 11). It has three operating modes: "capture," "auto-load" and "baud rate generator," which are selected by bits in T2CON as shown in Table 2.

Table 2. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(MSB)							(LSB)
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/$\overline{T2}$	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Figure 11. T2CON: Timer/Counter 2 Control Register

In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. (RCAP2L and RCAP2H are new Special Function Registers in the 8052.) In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2, like TF2, can generate an interrupt.

The Capture Mode is illustrated in Figure 12.

In the auto-reload mode there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2.

The auto-reload mode is illustrated in Figure 13.

The baud rate generator mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

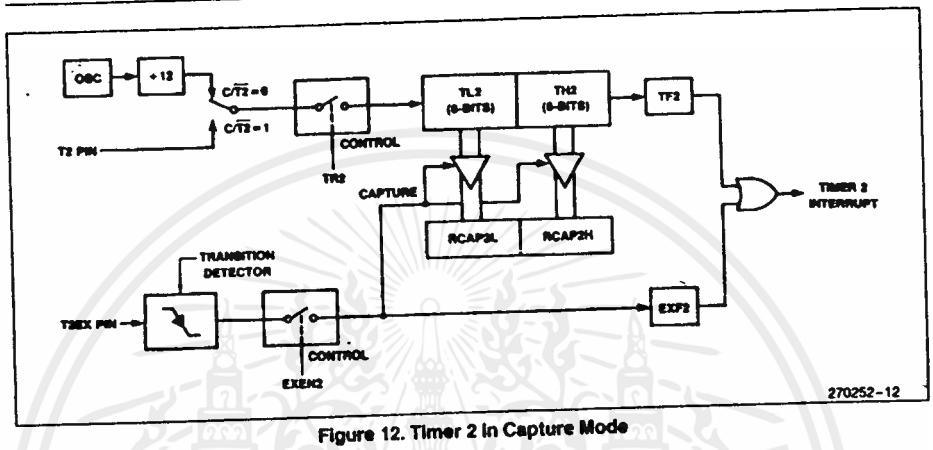


Figure 12. Timer 2 in Capture Mode

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

Mode 1: 10 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Functon Register SCON, while the stop bit is ignored. The baud rate is programmable to either $1/_{32}$ or $1/_{64}$ the oscillator frequency.

Mode 3: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 14. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

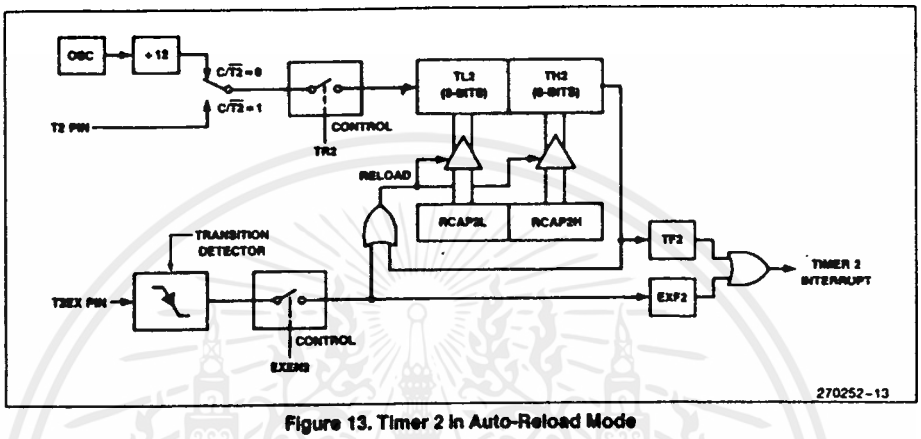


Figure 13. Timer 2 in Auto-Reload Mode

(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	f_{osc} /12
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{osc.}$/64 or $f_{osc.}$/32
1	1	3	9-bit UART	variable

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 14. SCON: Serial Port Control Register

Baud Rates

The baud rate in Mode 0 is fixed:

$$\text{Mode 0 Baud Rate} = \frac{\text{Oscillator Frequency}}{12}$$

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate is $1/_{64}$ the oscillator frequency. If SMOD = 1, the baud rate is $1/_{32}$ the oscillator frequency.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times (\text{Oscillator Frequency})$$

In the 8051, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate. In the 8052, these baud rates can be determined by Timer 1, or by Timer 2, or by both (one for transmit and the other for receive).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \text{(Timer 1 Overflow Rate)}$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{\text{Oscillator Frequency}}{12 \times [256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Figure 15 lists various commonly used baud rates and how they can be obtained from Timer 1.

Baud Rate	f_{OSC}	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes 1, 3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5	11.986 MHZ	0	0	2	1DH
110	6 MHZ	0	0	2	72H
110	12 MHZ	0	0	1	FEEBH

Figure 15. Timer 1 Generated Commonly Used Baud Rates

Using Timer 2 to Generate Baud Rates

In the 8052, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Figure 11). Note then the baud rates for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 16.

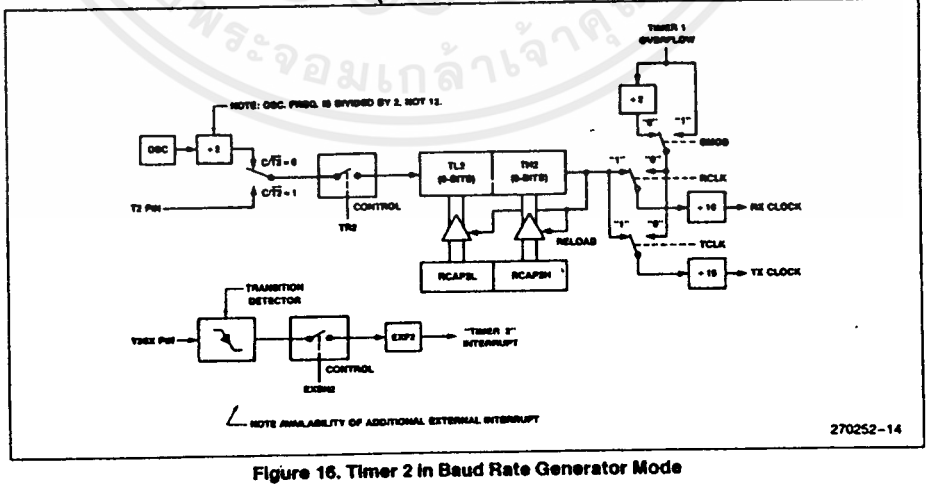


Figure 16. Timer 2 in Baud Rate Generator Mode

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at $1/12$ the oscillator frequency). As a baud rate generator, however, it increments every state time (thus at $1/2$ the oscillator frequency). In that case the baud rate is given by the formula

$$\text{Modes 1, 3 Baud Rate} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H, RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 16. This Figure is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in "timer" function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the Timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn't be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at $1/12$ the oscillator frequency.

Figure 17 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF," and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1 and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and RI = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

More About Mode 1

Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the 8051 the baud rate is determined by the Timer 1 overflow rate. In the 8052 it is determined either by the Timer 1 overflow rate, or the Timer 2 overflow rate, or both (one for transmit and the other for receive).

Figure 18 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

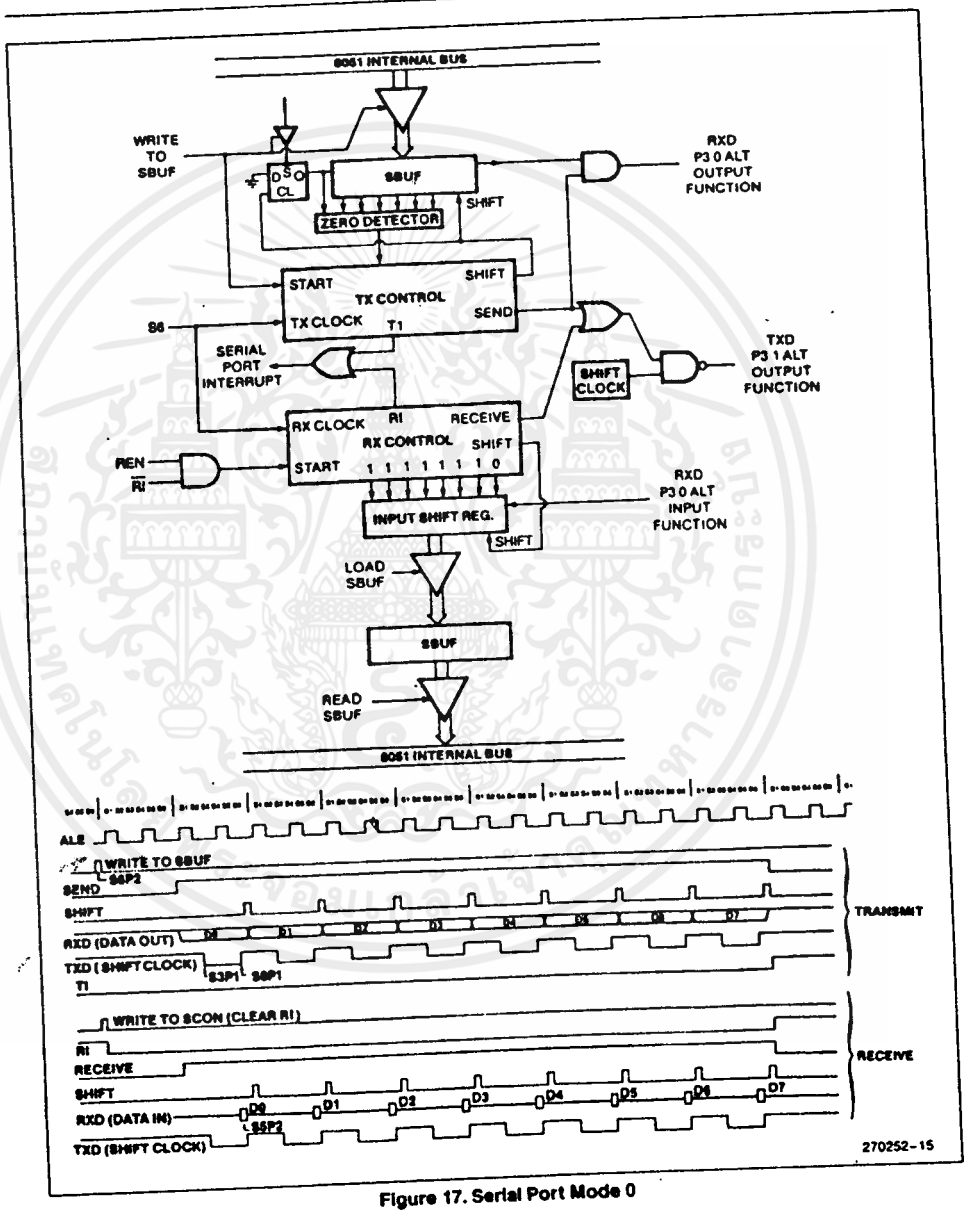


Figure 17. Serial Port Mode 0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

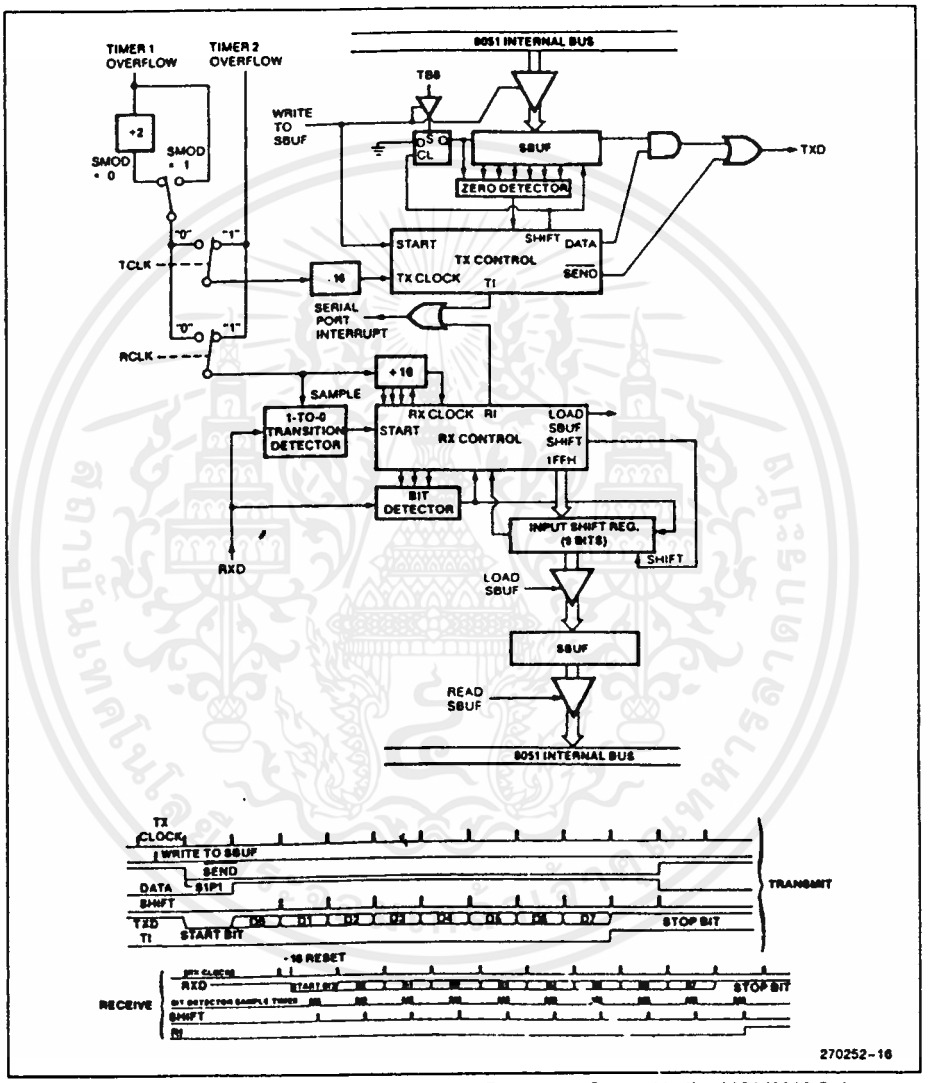


Figure 18. Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 8052/8032 Only.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

As data bits shift out to the right, zeroes are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI. This occurs at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register, (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.

1) RI = 0, and
2) Either SM2 = 0, or the received stop bit = 1

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RXD.

More About Modes 2 and 3

Eleven bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either $1/_{32}$ or $1/_{64}$ the oscillator frequency in Mode 2. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK.

Figures 19 and 20 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TXD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeroes are clocked in. Thus, as data bits shift out to the right, zeroes are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeroes. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

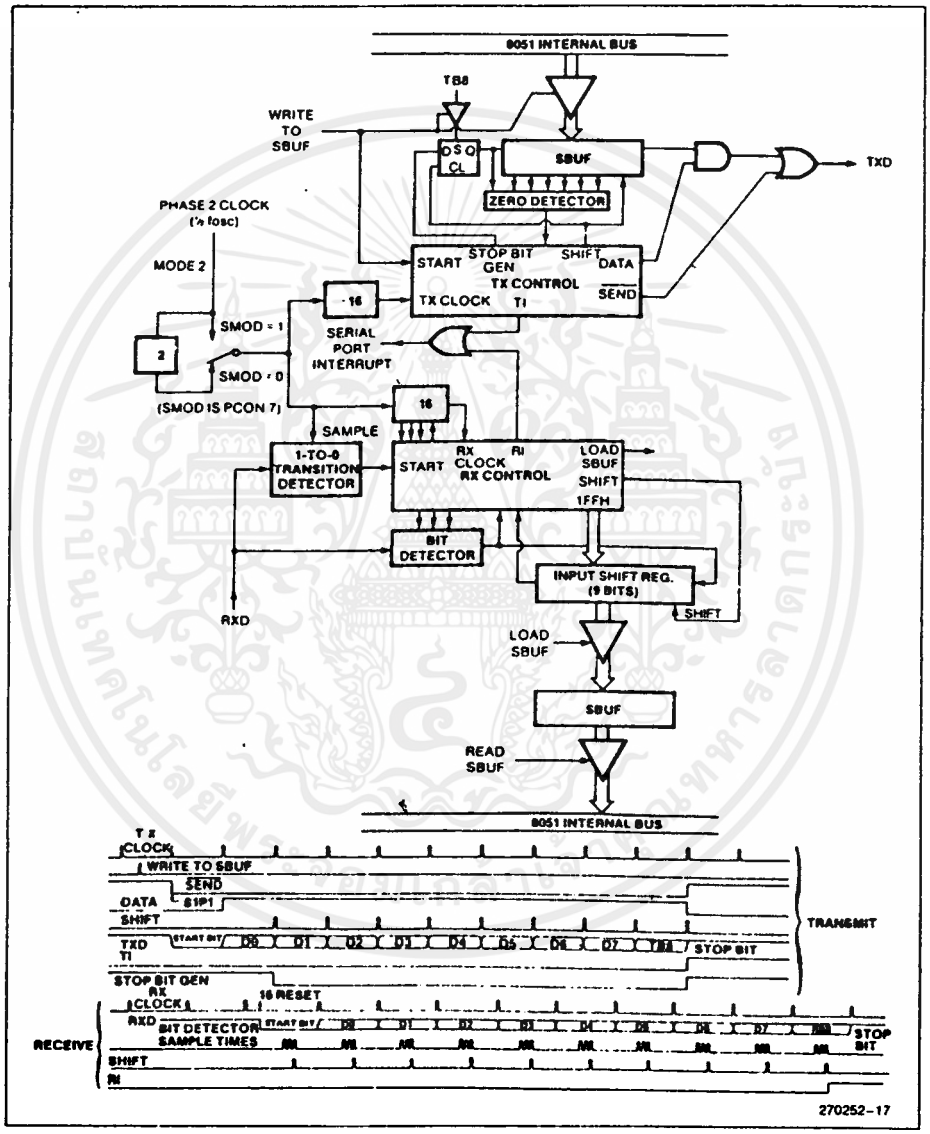


Figure 19. Serial Port Mode 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

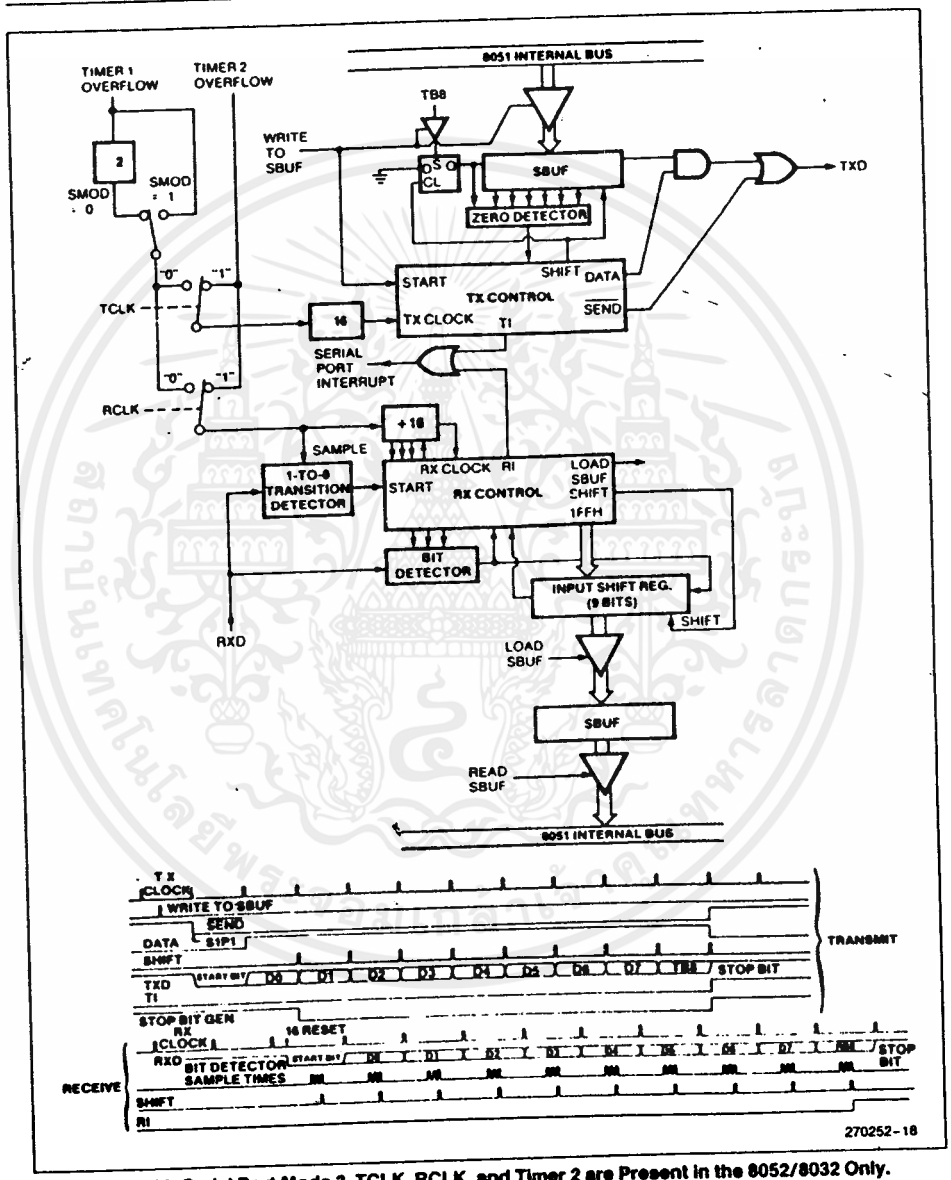


Figure 20. Serial Port Mode 3. TCLK, RCLK, and Timer 2 are Present in the 8052/8032 Only.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI = 0, and
2) Either SM2 = 0 or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Note that the value of the received stop bit is irrelevant to SBUF, RB8, or RI.

INTERRUPTS

The 8051 provides 5 interrupt sources. The 8052 provides 6. These are shown in Figure 21.

The External Interrupts $\overline{INT0}$ and $\overline{INT1}$ can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

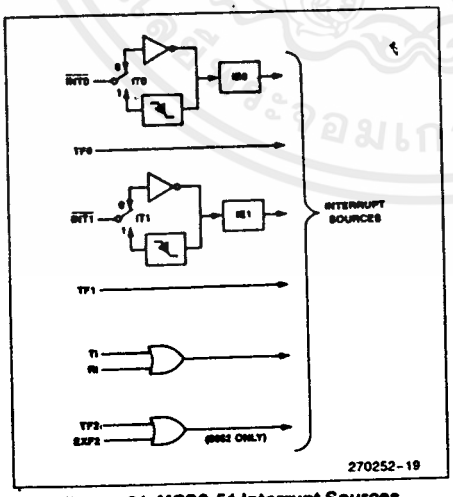


Figure 21. MCS®-51 Interrupt Sources

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers (except see Timer 0 in Mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

In the 8052, the Timer 2 Interrupt is generated by the logical OR of TF2 and EXF2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared in software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

(MSB)							(LSB)
EA	—	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 22. IE: Interrupt Enable Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 22). IE contains also a global disable bit, EA, which disables all interrupts at once.

Note in Figure 22 that bit position IE.6 is unimplemented. In the 8051s, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP (Figure 23). A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

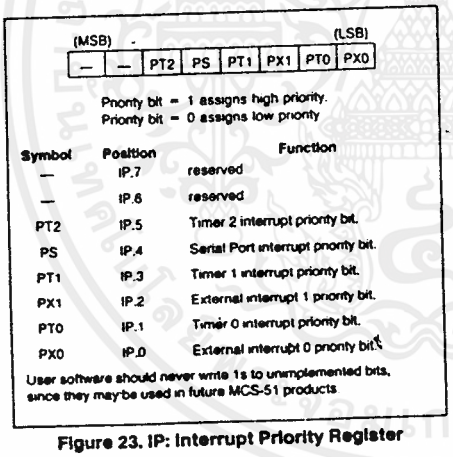


Figure 23. IP: Interrupt Priority Register

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

	Source	Priority Within Level
1.	IE0	(highest)
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI + TI	
6.	TF2 + EXF2	(lowest)

Note that the "priority within level" structure is only used to resolve *simultaneous requests of the same priority level.*

The IP register contains a number of unimplemented bits. IP.7 and IP.6 are vacant in the 8052s, and in the 8051s these and IP.5 are vacant. User software should not write 1s to these bit positions, since they may be used in future MCS-51 products.

How Interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. The 8052's Timer 2 interrupt cycle is different, as described in the Response Time Section. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be

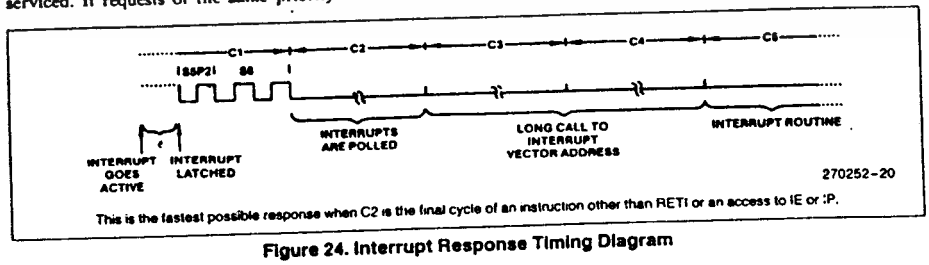


Figure 24. Interrupt Response Timing Diagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least *one more* instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note then that if an interrupt flag is active but not being responded to for one of the above conditions, and is not *still* active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The polling cycle/LCALL sequence is illustrated in Figure 24.

Note that if an interrupt of higher priority level goes active prior to S5P2 of the machine cycle labeled C3 in Figure 24, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port or Timer 2 flags. This has to be done in the user's software. It clears an external interrupt flag (IE0 or IE1) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below.

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT1 or IT0 in Register TCON. If ITx = 0, external interrupt x is triggered by a detected low at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is edge-triggered. In this mode if successive samples of the $\overline{INTx}$ pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The $\overline{INT0}$ and $\overline{INT1}$ levels are inverted and latched into the interrupt flags IE0 and IE1 at S5P2 of every machine cycle. Similarly, the Timer 2 flag EXF2 and the Serial Port flags RI and TI are set at S5P2. The values are not actually polled by the circuitry until the next machine cycle.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag TF2 is set at S2P2 and is polled in the same cycle in which the timer overflows.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 24 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

SINGLE-STEP OPERATION

The 8051 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-stop operation is to program one of the external interrupts (say, $\overline{INT0}$) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$  ;Wait Here Till INT0 Goes High
JB    P3.2,$  ;Now Wait Here Till it Goes Low
RETI          :Go Back and Execute One Instruction
```

Now if the $\overline{INT0}$ pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until $\overline{INT0}$ is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

RESET

The reset input is the RST pin, which is the input to a Schmitt Trigger.

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), *while the oscillator is running*. The CPU responds by generating an internal reset, with the timing shown in Figure 25.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase 2 of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

While the RST pin is high, ALE and PSEN are weakly pulled high. After RST is pulled low, it will take 1 to 2 machine cycles for ALE and PSEN to start clocking. For this reason, other devices can not be synchronized to the internal timings of the 8051.

Driving the ALE and $\overline{PSEN}$ pins to 0 while reset is active could cause the device to go into an indeterminate state.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF. The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate. Table 3 lists the SFRs and their reset values.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

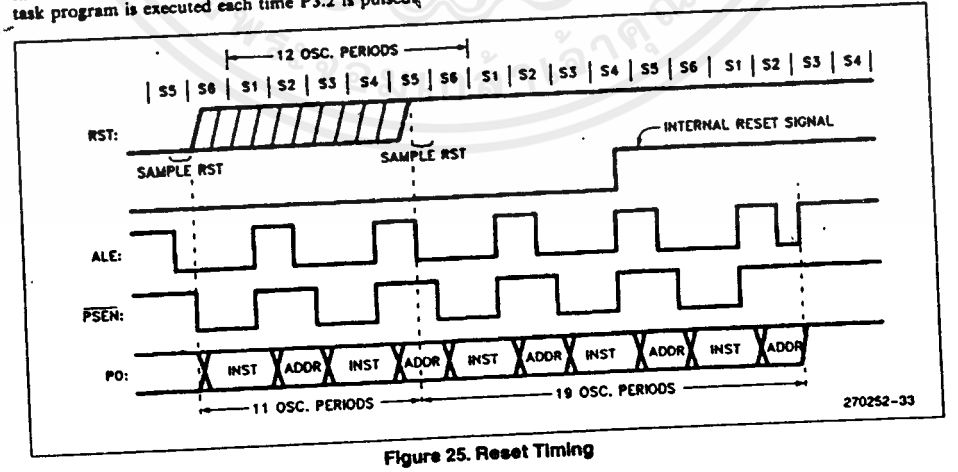


Figure 25. Reset Timing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 3. Reset Values of the SFRs

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0–P3	FFH
IP (8051)	XXX00000B
IP (8052)	XX000000B
IE (8051)	0XX00000B
IE (8052)	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2 (8052)	00H
TL2 (8052)	00H
RCAP2H (8052)	00H
RCAP2L (8052)	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXB
PCON (CHMOS)	0XXX0000B

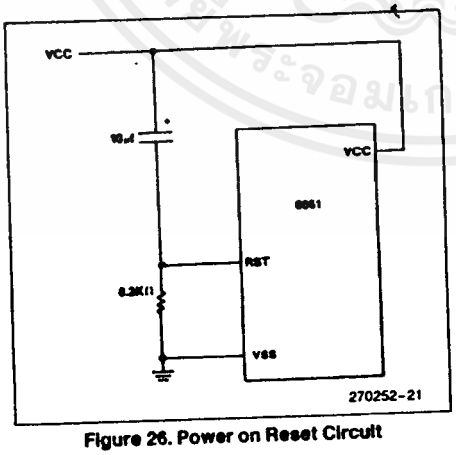


Figure 26. Power on Reset Circuit

POWER-ON RESET

For HMOS devices when V_{CC} is turned on an automatic reset can be obtained by connecting the RST pin to V_{CC} through a 10 μF capacitor and to V_{SS} through an 8.2 KΩ resistor (Figure 26). The CHMOS devices do not require this resistor although its presence does no harm. In fact, for CHMOS devices the external resistor can be removed because they have an internal pulldown on the RST pin. The capacitor value could then be reduced to 1 μF.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles.

On power up, V_{CC} should rise within approximately ten milliseconds. The oscillator start-up time will depend on the oscillator frequency. For a 10 MHz crystal, the start-up time is typically 1 ms. For a 1 MHz crystal, the start-up time is typically 10 ms.

With the given circuit, reducing V_{CC} quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited and will not harm the device.

NOTE:
The port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

POWER-SAVING MODES OF OPERATION

For applications where power consumption is critical the CHMOS version provides power reduced modes of operation as a standard feature. The power down mode in HMOS devices is no longer a standard feature and is being phased out.

CHMOS Power Reduction Modes

CHMOS versions have two power-reducing modes, Idle and Power Down. The input through which backup power is supplied during these operations is VCC. Figure 27 shows the internal circuitry which implements these features. In the Idle mode (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

clock signal is gated off to the CPU. In Power Down (PD = 1), the oscillator is frozen. The Idle and Power Down modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 26 details its contents.

In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

IDLE MODE

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated. ALE and $\overline{PSEN}$ hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into Idle.

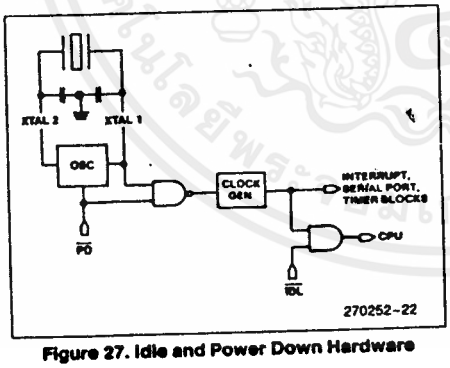


Figure 27. Idle and Power Down Hardware

(MSB)							(LSB)
SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3
—	PCON.6	(Reserved)
—	PCON.5	(Reserved)
—	PCON.4	(Reserved)
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).
In the HMOS devices the PCON register only contains SMOD. The other four bits are implemented only in the CHMOS devices. User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

Figure 28. PCON: Power Control Register

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the Idle mode is with a hardware reset. Since the clock oscillator is still running, the hardware reset needs to be held active for only two machine cycles (24 oscillator periods) to complete the reset.

The signal at the RST pin clears the IDL bit directly and asynchronously. At this time the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode. As shown in Figure 25, two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On-chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external Data RAM.

POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all func-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 4. EPROM Versions of the 8051 and 8052

Device Name	EPROM Version	EPROM Bytes	Ckt Type	VPP	Time Required to Program Entire Array
8051	(8751)	4K	HMOS	21.0V	4 minutes
8051AH	8751H	4K	HMOS	21.0V	4 minutes
80C51BH	87C51	4K	CHMOS	12.75V	13 seconds
8052AH	8752BH	8K	HMOS	12.75V	26 seconds

tions are stopped, but the on-chip RAM and Special Function Registers are held. The port pins output the values held by their respective SFRs. ALE and $\overline{PSEN}$ output lows.

The only exit from Power Down for the 80C51 is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM.

In the Power Down mode of operation, VCC can be reduced to as low as 2V. Care must be taken, however, to ensure that VCC is not reduced before the Power Down mode is invoked, and that VCC is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before VCC is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10 msec).

EPROM VERSIONS

The EPROM versions of these devices are listed in Table 4. The 8751H programs at VPP = 21V using one 50 msec $\overline{PROG}$ pulse per byte programmed. This results in a total programming time (4K bytes) of approximately 4 minutes.

The 8752BH and 87C51 use the faster "Quick-Pulse" programming™ algorithm. These devices program at VPP = 12.75V using a series of twenty-five 100 μs $\overline{PROG}$ pulses per byte programmed. This results in a total programming time of approximately 26 seconds for the 8752BH (8K bytes) and 13 seconds for the 87C51 (4K bytes).

Detailed procedures for programming and verifying each device are given in the data sheets.

EXPOSURE TO LIGHT

It is good practice to cover the EPROM window with an opaque label when the device is in operation. This is not so much to protect the EPROM array from inadvertent erasure, but to protect the RAM and other on-chip logic. Allowing light to impinge on the silicon die while the device is operating can cause logical malfunction.

Program Memory Locks

In some microcontroller applications it is desirable that the Program Memory be secure from software piracy. Intel has responded to this need by implementing a Program Memory locking scheme in some of the MCS-51 devices. While it is impossible for anyone to guarantee absolute security against all levels of technological sophistication, the Program Memory locks in the MCS-51 devices will present a formidable barrier against illegal readout of protected software.

One Lock Bit Scheme on 8751H

The 8751H contains a lock bit which, once programmed, denies electrical access by any external means to the on-chip Program Memory. The effect of this lock bit is that while it is programmed the internal Program Memory can not be read out, the device can not be further programmed, and it *can not execute external Program Memory.* Erasing the EPROM array deactivates the lock bit and restores the device's full functionality. It can then be re-programmed.

The procedure for programming the lock bit is detailed in the 8751H data sheet.

Two-Level Program Memory Lock Scheme

The 87C51 and 8752BH contain two Program Memory locking schemes: Encrypted Verify and Lock Bits.

Encrypted Verify: These devices implement a 32-byte EPROM array that can be programmed by the customer, and which can then be used to encrypt the program code bytes during EPROM verification. The EPROM verification procedure is performed as usual, except that each code byte comes out X-NORed with one of the 32 key bytes. The key bytes are gone through in sequence. Therefore, to read the ROM code, one has to know the 32 key bytes in their proper sequence.

Unprogrammed bytes have the value FFH. Therefore, if the Encryption Array is left unprogrammed all the key bytes have the value FFH. Since any code byte X-NORed with FFH leaves the code byte unchanged, leaving the Encryption Array unprogrammed in effect bypasses the encryption feature.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Lock Bits: Also on the chip are two Lock Bits which can be left unprogrammed (U) or programmed (P) to obtain the following features:

Bit 2	Bit 1	Additional Features
U	U	None
U	P	• Externally fetched code can not access internal Program Memory. • Further programming disabled.
P	U	(Reserved for Future definition.)
P	P	• Externally fetched code can not access internal Program Memory. • Further programming disabled. • Program verification is disabled.

When Lock Bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

ROM Protection

The 8051AHP and 80C51BHP are ROM Protected versions of the 8051AH and 80C51BH, respectively. To incorporate this Protection Feature, program verification has been disabled and external memory accesses have been limited to 4K. Refer to the data sheets on these parts for more information.

ONCE Mode

The ONCE ("on-circuit emulation") mode facilitates testing and debugging of systems using the device without the device having to be removed from the circuit. The ONCE mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored after a normal reset is applied.

THE ON-CHIP OSCILLATORS

HMOS Versions

The on-chip oscillator circuitry for the HMOS (HMOS-I and HMOS-II) members of the MCS-51 family is a single stage linear inverter (Figure 29), intended for use as a crystal-controlled, positive reactance oscillator (Figure 30). In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal.

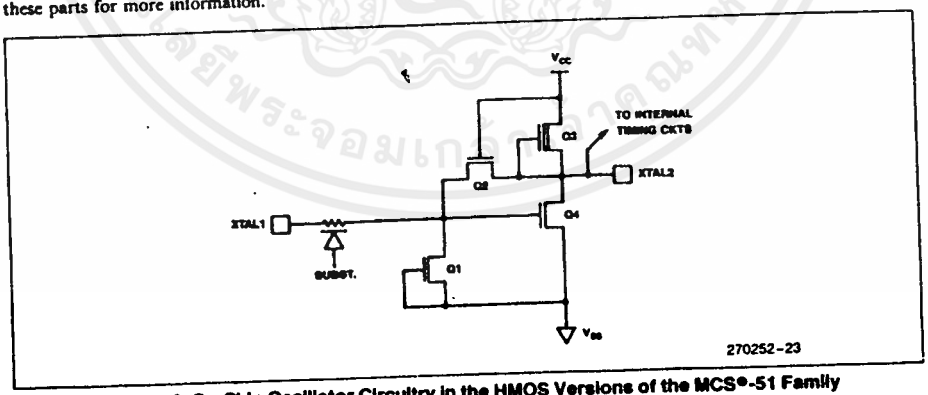


Figure 29. On-Chip Oscillator Circuitry in the HMOS Versions of the MCS®-51 Family

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

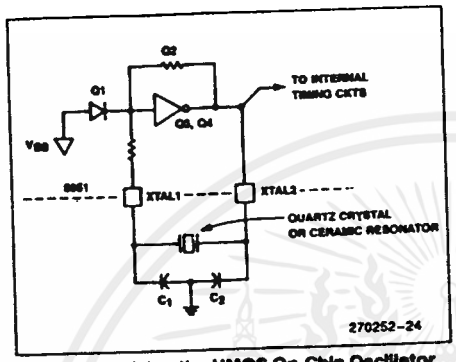


Figure 30. Using the HMOS On-Chip Oscillator

The crystal specifications and capacitance values (C1 and C2 in Figure 30) are not critical. 30 pF can be used in these positions at any frequency with good quality crystals. A ceramic resonator can be used in place of the crystal in cost-sensitive applications. When a ceramic resonator is used, C1 and C2 are normally selected to be of somewhat higher values, typically, 47 pF. The manufacturer of the ceramic resonator should be consulted for recommendations on the values of these capacitors.

In general, crystals used with these devices typically have the following specifications:

ESR (Equivalent Series Resistance)	see Figure 31
C_O (Shunt Capacitance)	7.0 pF max.
C_L (Load Capacitance)	30 pF ± 3 pF
Drive Level	1 MW

Frequency, tolerance and temperature range are determined by the system requirements.

A more in-depth discussion of crystal specifications, ceramic resonators, and the selection of values for C1 and C2 can be found in Application Note AP-155, "Oscillators for Microcontrollers," which is included in the *Embedded Control Applications Handbook*.

To drive the HMOS parts with an external clock source, apply the external clock signal to XTAL2, and ground XTAL1, as shown in Figure 32. A pullup resistor may be used (to increase noise margin), but is optional if VOH of the driving gate exceeds the VIH MIN specification of XTAL2.

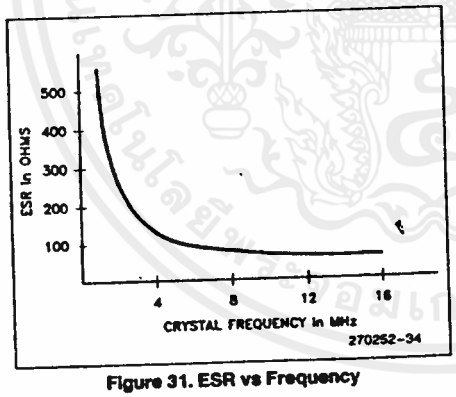


Figure 31. ESR vs Frequency

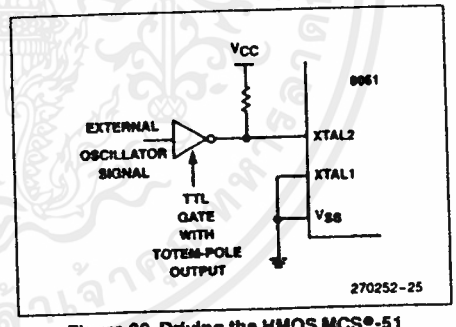


Figure 32. Driving the HMOS MCS®-51 Parts with an External Clock Source

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CHMOS VERSIONS

The on-chip oscillator circuitry for the 80C51BH, shown in Figure 33, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator in the same manner as the HMOS parts. However, there are some important differences.

One difference is that the 80C51BH is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON). Another difference is that in the 80C51BH the internal clocking circuitry is driven by the signal at XTAL1, whereas in the HMOS versions it is by the signal at XTAL2.

The feedback resistor R_f in Figure 33 consists of paralleled n- and p- channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2, which act as clamps to VCC and VSS, are parasitic to the R_f FETs.

The oscillator can be used with the same external components as the HMOS versions, as shown in Figure 34. Typically, C1 = C2 = 30 pF when the feedback element is a quartz crystal, and C1 = C2 = 47 pF when a ceramic resonator is used.

To drive the CHMOS parts with an external clock source, apply the external clock signal to XTAL1, and leave XTAL2 float, as shown in Figure 35.

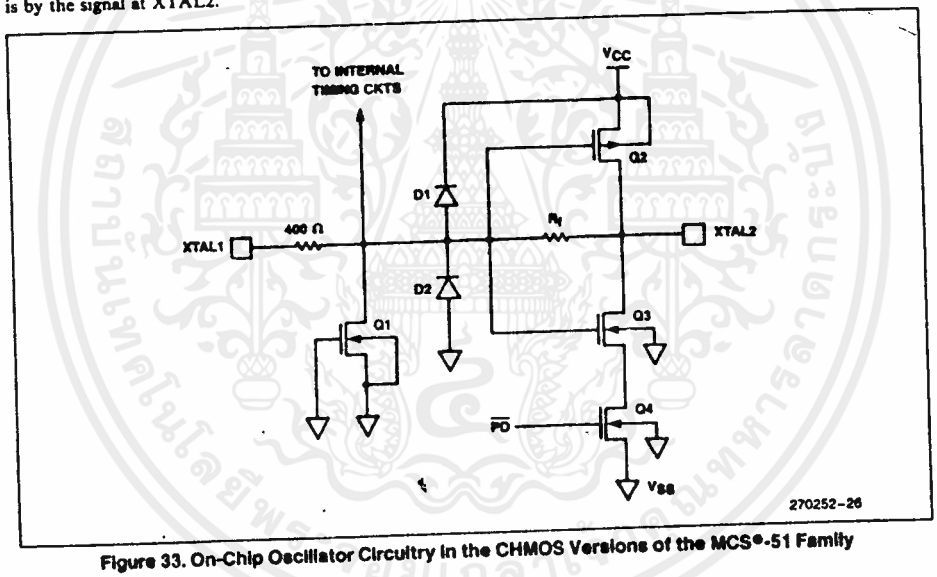


Figure 33. On-Chip Oscillator Circuitry in the CHMOS Versions of the MCS®-51 Family

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

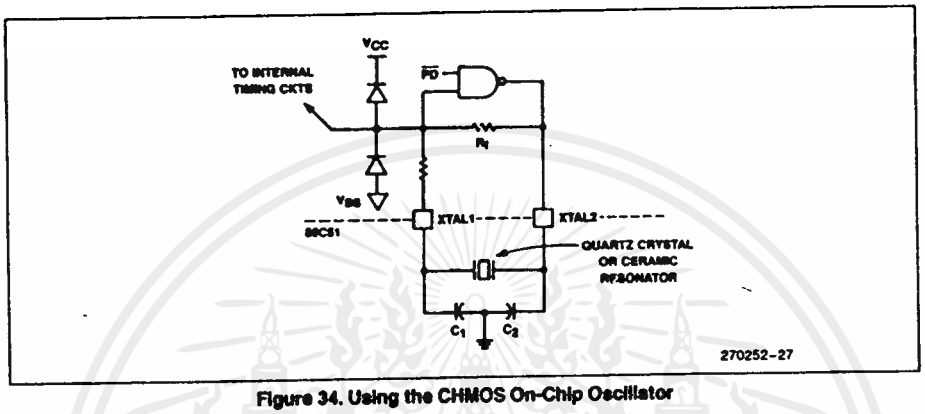


Figure 34. Using the CHMOS On-Chip Oscillator

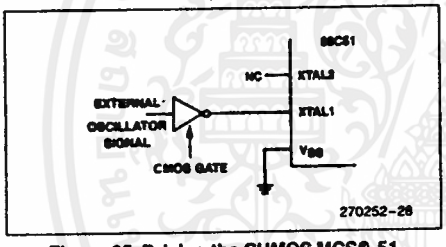


Figure 35. Driving the CHMOS MCS®-51 Parts with an External Clock Source

The reason for this change from the way the HMOS part is driven can be seen by comparing Figures 29 and 33. In the HMOS devices the internal timing circuits are driven by the signal at XTAL2. In the CHMOS devices the internal timing circuits are driven by the signal at XTAL1.

INTERNAL TIMING

Figures 36 through 39 show when the various strobe and port signals are clocked internally. The figures do not show rise and fall times of the signals, nor do they show propagation delays between the XTAL signal and events at other pins.

Rise and fall times are dependent on the external loading that each pin must drive. They are often taken to be something in the neighborhood of 10 nsec, measured between 0.8V and 2.0V.

Propagation delays are different for different pins. For a given pin they vary with pin loading, temperature, VCC, and manufacturing lot. If the XTAL waveform is taken as the timing reference, prop delays may vary from 25 to 125 nsec.

The AC Timings section of the data sheets do not reference any timing to the XTAL waveform. Rather, they relate the critical edges of control and input signals to each other. The timings published in the data sheets include the effects of propagation delays under the specified test conditions.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

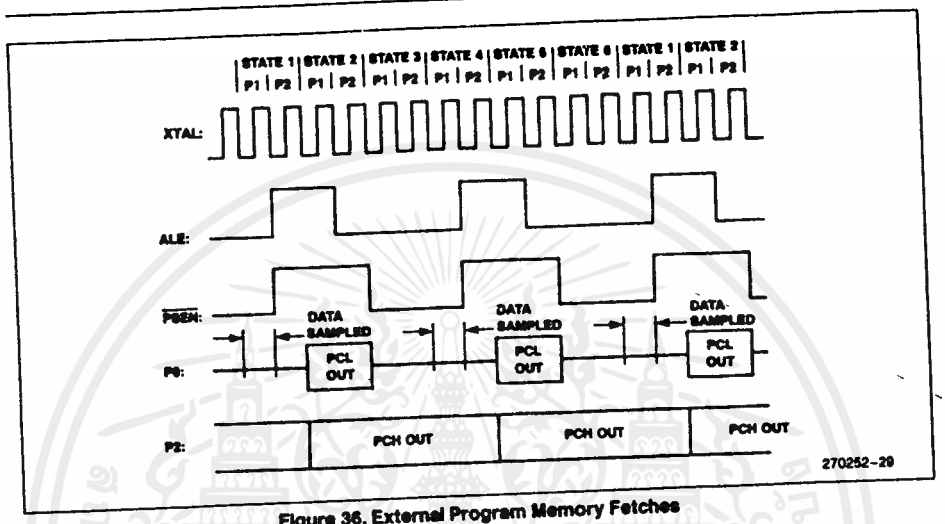


Figure 36. External Program Memory Fetches

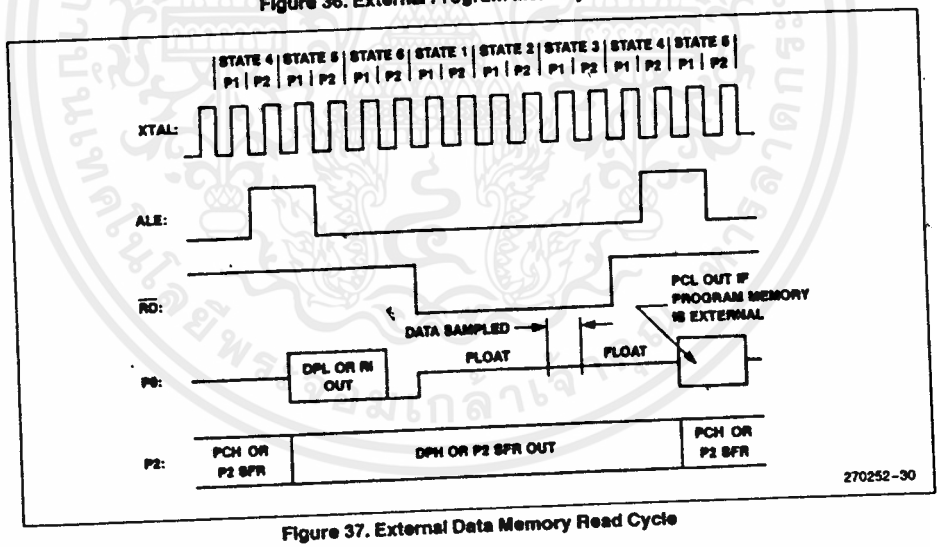


Figure 37. External Data Memory Read Cycle

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

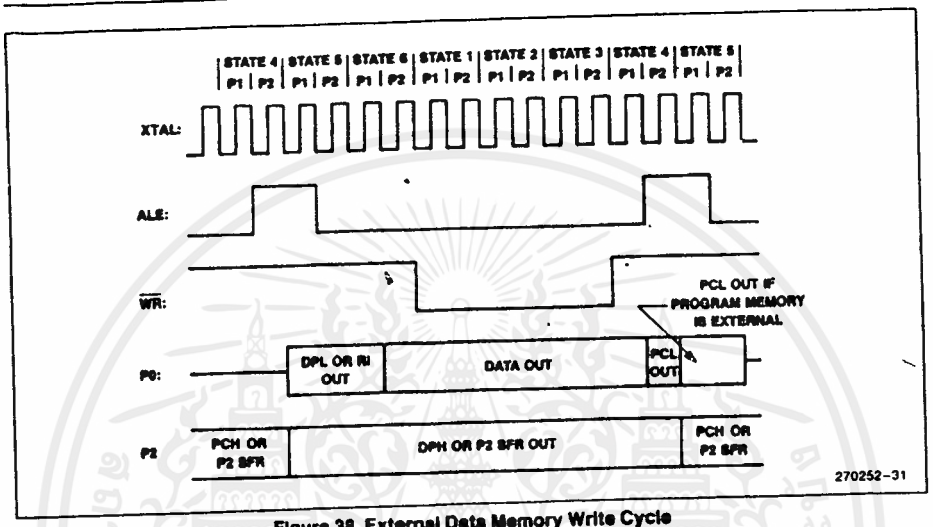


Figure 38. External Data Memory Write Cycle

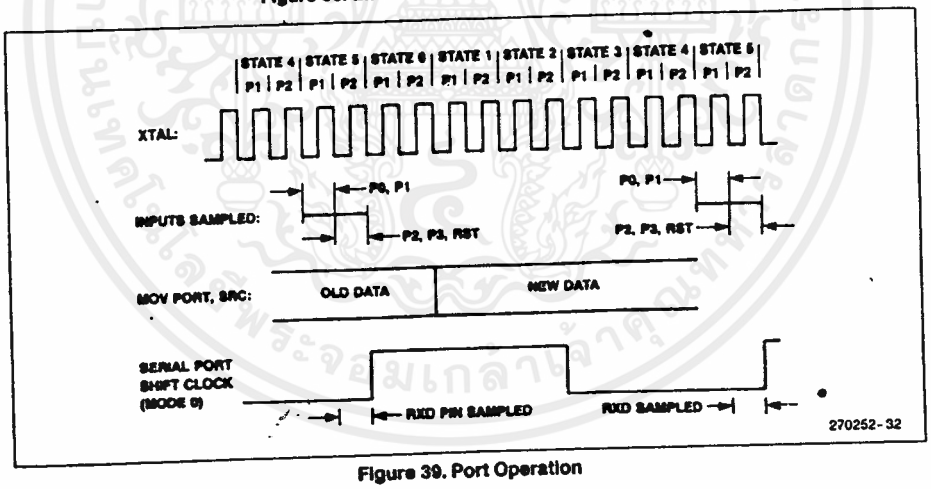


Figure 39. Port Operation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

1."An Inteoduction to the Intel MCS-51 Single Chip Microcontroller Family" , Intel Corporation , USA

2."Intel MCS-51 Family User'Manual" , Intel Corporation , USA

3."ANT-32 Embeded Controller Broad User's Manual" Sila Reserch Co.,Ltd.,Thailand.

4."8255 PPI User's Manaul"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ ภากร หุตะสังกาศ ผู้ให้คำปรึกษา และข้อแนะนำต่างๆ

ขอขอบคุณ อาจารย์ สุพรรณ กุลพาณิชย์ ที่ให้คำปรึกษา และอำนวยความสะดวกทาง

ด้าน WAREHOUSE MODEL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้