



ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญาโท



ชื่อหัวข้อ ดิสก์ไดรฟ์ 3.5 นิ้ว สำหรับซิงเกิลบอร์ด

3.5 Disk Drive Controller for Single Board

ชื่อนักศึกษา 1. นายจิตร ฝึกทอง รหัสประจำตัว 40031105
2. น.ส.นงคณา สมบูรณ์ศรี รหัสประจำตัว 40031113
3. น.ส.มณีนี คอภจันทร์ รหัสประจำตัว 40031122
4. น.ส.จิรายุส แก้วละศรี รหัสประจำตัว 40031128

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชา วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา อาจารย์ไพรัชวุฒิ สุวรรณจันทร์

อาจารย์ที่ปรึกษาร่วม อาจารย์สุชนา อาจารย์สม

คณะกรรมการสอบปริญญาโท		ลายมือชื่อ
1. อาจารย์ไพรัชวุฒิ	สุวรรณจันทร์	
2. อาจารย์สุชนา	อานาญ	
3. อาจารย์โกศล	ตราฐ	
4. อาจารย์พงษ์เกียรติ	เชษฐพิทักษ์สกุล	
5. อาจารย์ไพบุลย์	พวงวงศ์ตระกูล	

วัน/เดือน/ปีที่สอบ วันพุธที่ 28 เมษายน พ.ศ. 2542 เวลา 9.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

นพ.

3310

2541



ภาควิชารับรองแล้ว

ธีระพล เทพหัสดิน ณ อยุธยา

คณบดี

วันที่ / เดือน พ.ศ. 2542

เลขหมาย 32803
เลขทะเบียน 10 ส.ย. 2542

ปริญญานิพนธ์

ดิสก์ไดรฟ์ 3.5 นิ้ว สำหรับซิงเกิลบอร์ด

3.5" DISK DRIVE CONTROLLER FOR SINGLE BOARD



นายฐิติ

พิกุลทอง

นางสาวนงกัญช สมุทรคนตรี

นางสาวมาลินี ดอกจันทร์

นายวิเชียร

แก้วณะศรี

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานี้เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปีการศึกษา 2541

ปริญญานิพนธ์

เรื่อง ดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด

3.5" Disk Drive Controller for Single Board

วัตถุประสงค์

1. เพื่อศึกษาการทำงานของดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
2. เพื่อออกแบบชุดควบคุมดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
3. เพื่อสร้างชุดควบคุมดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
4. เพื่อทดลองและทดสอบดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
5. เพื่อนำดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด ไปใช้งาน

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้เกี่ยวกับการทำงานของดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
2. ได้วงจรต้นแบบดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
3. ได้เครื่องต้นแบบดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
4. ได้ผลการทดลองและทดสอบดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด
5. ได้นำดิสก์ไดรฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด ไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

I

ชื่อหัวข้อ	ดิสก์ไครฟ์ 3.5 นิ้ว สำหรับซิงเกิลบอร์ด	
นักศึกษา	นายฐิติ	พิกุลทอง
	นางสาวนงคันทน์	สมุทรคนตรี
	นางสาวมาลินี	ดอกจันทร์
	นายวิเชียร	แก้วณะศรี
อาจารย์ที่ปรึกษา	อาจารย์พีระวุฒิ	สุวรรณจันทร์
อาจารย์ที่ปรึกษาร่วม	อาจารย์สุชิน	อาจหาญ
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต	
สาขาวิชา	วิศวกรรมโทรคมนาคม	
ปีการศึกษา	2541	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบและการสร้างดิสก์ไครฟ์ 3.5 นิ้วสำหรับซิงเกิลบอร์ด ซึ่งสามารถใช้ในการบันทึกข้อมูลที่อยู่ภายในหน่วยความจำของซิงเกิลบอร์ด และสามารถส่งข้อมูลจากฟลอปปีดิสก์ไปยังหน่วยความจำของซิงเกิลบอร์ดได้ โครงการนี้ประกอบด้วย 2 ส่วน คือ ส่วนควบคุมการโอนย้ายข้อมูล และส่วนของภาค FDC (Floppy Disk Controller) ซึ่งจะช่วยให้เข้าใจหลักการทำงานของดิสก์ไครฟ์ 3.5 นิ้ว สำหรับซิงเกิลบอร์ดได้ดียิ่งขึ้น และสามารถนำไปประยุกต์ใช้ในการบันทึกโปรแกรมลงบนดิสก์เกตต์ขนาด 3.5 นิ้วได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

II

Thesis Title	3.5" Disk Drive Controller for Single Board	
Students	Mr.Thiti	Phikuntong
	MissNongnuch	Samuthdontri
	MissMalinee	Dokchan
	Mr.Wichian	Kaewnasri
Advisor	Mr.Peerawut	Suwanjan
Co-Advisor	Mr.Suchin	Artharn
Education Level	Bachelor of Science in Industrial Education	
Program in	Telecommunication Engineering	
Academic Year	1998	

ABSTRACT

This thesis presents a designing and implementation of 3.5" disk drive controller for single board, which can save a program from the memory of single board to a 3.5" diskette and load sent datas from a floppy disk to memory of single board. This project consists of two parts, which are the transfer data control and FDC (Floppy Disk Controller). This project helps the user to better understand the principle of 3.5" disk drive controller for single board and can be applied to use for saving program on 3.5" diskette.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงลงได้ด้วยดี เนื่องมาจากความร่วมมือของสมาชิกภายในกลุ่มทุกท่าน ขอขอบคุณอาจารย์ที่ปรึกษาปริญญานิพนธ์และคณาจารย์ภาควิชาครุศาสตร์วิศกรรมทุกท่านที่กรุณาให้ความอนุเคราะห์เครื่องมือ และอุปกรณ์ รวมทั้งยังให้คำปรึกษาแนะนำ แนวความคิด ความรู้ต่างๆ แนวทางการแก้ไขปัญหา ในการจัดทำปริญญานิพนธ์ ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม สำนักหอสมุดกลาง ที่อำนวยความสะดวกในการค้นคว้าข้อมูล และเพื่อนๆ ทุกคนที่ให้ความช่วยเหลือ สุดท่ายที่ควรระลึกถึงอย่างยิ่ง บิดา และมารดาที่เป็นผู้ให้ความสนับสนุน ทางด้านการศึกษา และเป็นผู้ให้กำลังใจด้วยดีตลอดมา ตั้งแต่อดีตจนถึงปัจจุบัน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญรูป	IX
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์	1
1.2 ชัดความสามารถของโครงการ	1
1.3 เนื้อหาโดยสังเขป	1
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 กล่าวนำ	3
2.2 ฟลอปปีดิสก์	3
2.2.1 โครงสร้างของดิสก์เกตต์	3
2.2.2 แทร็กและเซกเตอร์	4
2.2.3 การเก็บข้อมูลในแผ่นฟลอปปีดิสก์	4
2.2.4 การมอดูเลตของข้อมูล	5
2.2.5 การฟอร์แมตฟลอปปีดิสก์	7
2.2.6 โครงสร้างของดิสก์ไครฟ์	8
2.2.7 ข้อกำหนดรายละเอียดที่สำคัญของดิสก์ไครฟ์	12
2.2.8 โครงสร้างดิสก์ตามแบบของดอส	12
2.3 วงจรควบคุมดิสก์ไครฟ์	25
2.3.1 โครงสร้างของวงจรควบคุมดิสก์ไครฟ์	25
2.3.2 ปรากฏการณ์การเพี้ยนแบบเลื่อนจุดยอด	26
2.3.3 การตรวจสอบข้อมูลผิดพลาดในวงจรควบคุมดิสก์ไครฟ์	26
2.3.4 โครงสร้างและการทำงานของ FDC μ PD765A	29
2.3.5 FDC (Floppy Disk Controller)	32

สารบัญ (ต่อ)

เรื่อง	หน้า
2.4 ไมโครคอนโทรลเลอร์ตระกูล MCS-51	44
2.4.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51	44
2.4.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์	44
2.4.3 การทำงานของ 8051	46
2.5 ไมโครโปรเซสเซอร์ Z-80	47
2.5.1 โครงสร้างซีพียู Z-80	47
2.5.2 รีจิสเตอร์ใช้งานทั่วไป (General Purpose Register)	50
2.5.3 แอคคิวมูเลเตอร์และรีจิสเตอร์สถานะ (Accumulator and Flag Register)	51
2.5.4 รีจิสเตอร์ใช้งานเฉพาะอย่าง (Special Purpose Register)	58
2.5.5 ขาและสัญญาณสำหรับการเชื่อมต่อ	62
2.5.6 แผนผังเวลาของไมโครโปรเซสเซอร์	66
บทที่ 3 การออกแบบ การสร้าง และการทำงาน	69
3.1 โครงสร้าง	70
3.1.1 นำข้อมูลที่อยู่ในหน่วยความจำของซิงเกิลบอร์ดมาเก็บไว้ในแผ่นฟลอปปีดิสก์	70
3.1.2 นำข้อมูลจากแผ่นฟลอปปีดิสก์มาเก็บไว้ในหน่วยความจำของซิงเกิลบอร์ด	70
3.2 แผนผังการทำงานของวงจร	70
3.2.1 ซีพียู	71
3.2.2 หน่วยความจำ	71
3.2.3 Bus Controller	71
3.2.4 FDC	72
3.3 ภาคนหน่วยความจำ	73
3.4 ภาคควบคุม	73
3.5 FDC โปรเซสเซอร์	74
3.6 วงจรกำเนิดสัญญาณนาฬิกา	80
3.7 ภาคบัฟเฟอร์	81
3.8 ดีโคเดอ์	82
3.9 ภาคเอ็ดพัตรจิสเตอร์	84

สารบัญ (ต่อ)

เรื่อง	หน้า
3.10 การทำงานภาคซอฟต์แวร์	85
3.10.1 การตั้งค่าเริ่มต้นฮาร์ดแวร์	87
3.10.2 การรับคำสั่งและการเขียนอ่านข้อมูล	88
3.11 ภาคอินเตอร์เฟส	89
บทที่ 4 การทดลอง และผลการทดลอง	93
4.1 การทดลองภาค Bus Controller	93
4.1.1 การทดลองโอนย้ายข้อมูลจากซิงเกิลบอร์ด ไปเก็บไว้ในแรมของวงจร	93
4.1.2 การอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ด	95
4.1.3 การทดลองกำหนดข้อมูลลงในตำแหน่งหน่วยความจำ MCS-51 แล้วโอนย้ายไปยังซิงเกิลบอร์ด	97
4.1.4 การทดลองการกำหนดข้อมูลลงในตำแหน่งหน่วยความจำของซิงเกิลบอร์ด แล้วโอนย้ายไปยัง MCS-51	100
4.1.5 การทดลองโอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51	103
4.1.6 การป้อนค่าต่างๆ ในการโอนย้ายข้อมูล	114
4.2 การทดลองภาค FDC โปรเซสเซอร์	124
4.3 การทดลองภาคกำเนิดสัญญาณนาฬิกา	125
4.4 การทดลองภาคดีโค้ดเดอร์	126
4.5 การทดลองภาคเอาต์พุตรีจิสเตอร์	126
บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา	128
5.1 บทสรุป	128
5.2 ปัญหาและแนวทางแก้ไข	128
5.2.1 แหล่งอ้างอิงข้อมูล	128
5.2.2 อุปกรณ์ใช้งาน	130
5.2.3 จุดบกพร่องจากการทำงาน	131
5.2.4 หัวอ่าน/เขียน ไม่เคลื่อนที่จากคำสั่ง	131
5.2.5 เกิดสัญญาณอินเตอร์รัพต์ในการเคลื่อนหัวอ่าน/เขียน	131
5.2.6 ความเร็วในการส่งผ่านข้อมูล ไม่เพียงพอ	132

สารบัญ (ต่อ)

เรื่อง	หน้า
5.3 แนวทางการพัฒนา	132
ภาคผนวก ก เครื่องต้นแบบ	134
ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์	139
ภาคผนวก ค ผังการทำงานและโปรแกรม	152
ภาคผนวก ง รายละเอียดและคุณสมบัติของอุปกรณ์	171
ภาคผนวก จ คู่มือการใช้งาน	229
บรรณานุกรม	231
ประวัติผู้แต่ง	232



สารบัญตาราง

ตาราง	หน้า
ตารางที่ 2.1 ขนาดของคลัสเตอร์	16
ตารางที่ 2.2 คุณลักษณะของไฟล์ที่ระบุด้วยบิต	24
ตารางที่ 2.3 สถานะการทำงานของบิตต่างๆ ใน Main Status Register	35
ตารางที่ 2.4 การทำงานที่สภาวะต่างๆ ที่ขา Chip Select	35
ตารางที่ 2.5 รูปแบบสัญญาณที่ใช้เชื่อมต่อข้อมูลใน Adapter Board กับ FDC	39
ตารางที่ 3.1 การทำงานของ U4 ในสภาวะต่างๆ	77
ตารางที่ 3.2 รายละเอียดการแสดงสภาวะของ Main Register ในแต่ละบิต	77
ตารางที่ 3.3 สถานะการทำงานของสัญญาณอ่าน, เขียน และสัญญาณตำแหน่ง	82
ตารางที่ 3.4 การกำหนดค่าบิตที่ทำงานในแต่ละบิต	83
ตารางที่ 3.5 สภาวะลอจิกของสัญญาณที่นำมาสร้างเป็นสัญญาณนาฬิกา	85
ตารางที่ 3.6 การกำหนดบิตการทำงานของเอาต์พุตรีจิสเตอร์	85
ตารางที่ 4.1 รูปแบบคำสั่ง Command	115
ตารางที่ 4.2 บิตต่างๆ ที่ได้จากภาคเอาต์พุตรีจิสเตอร์	127
ตารางที่ 4.3 ผลการทดลองภาคเอาต์พุตรีจิสเตอร์	127

สารบัญรูป

รูป	หน้า
รูปที่ 2.1 การจัดแท่งและเชกเตอร์ภายในดิสก์เกตต์	4
รูปที่ 2.2 หัวอ่านบันทึกดิสก์ในขณะที่ทำการบันทึก	5
รูปที่ 2.3 การมอดูเลชันโดยวิธี FM	6
รูปที่ 2.4 เปรียบเทียบการมอดูเลชันแบบ FM (ก) กับการมอดูเลชันแบบ MFM (ข)	7
รูปที่ 2.5 ตำแหน่งของข้อมูล เมื่อทำการฟอร์แมตแผ่นฟลอปปีดิสก์แบบ 3740	9
รูปที่ 2.6 การเก็บสถานะภาพของดิสก์เกตต์ในมาส์กอินเด็กซ์แอดเดรส	10
รูปที่ 2.7 โครงสร้างพื้นฐานของดิสก์ไครฟ์	10
รูปที่ 2.8 สัญญาณในการติดต่อระหว่างดิสก์ไครฟ์และวงจรควบคุมดิสก์ไครฟ์	11
รูปที่ 2.9 ส่วนประกอบของดิสก์ที่คอจัดการ	15
รูปที่ 2.10 บุตรเรคคอร์ด	21
รูปที่ 2.11 ขนาดของแต่ละส่วนในดิสก์	24
รูปที่ 2.12 แผนผังของวงจรควบคุมดิสก์ไครฟ์	27
รูปที่ 2.13 ปรัชญาการณณ์เลื่อนจุดยอด	28
รูปที่ 2.14 ตำแหน่งสัญญาณของไอซี μ PD765A	29
รูปที่ 2.15 หน้าที่ของแต่ละสัญญาณของ μ PD765A	30
รูปที่ 2.16 สถานภาพของ μ PD765A ที่รีจิสเตอร์แสดงสถานะหลัก	31
รูปที่ 2.17 แผนผังการทำงานภายในของ 765 FDC	33
รูปที่ 2.18 การจัดตำแหน่งขาของ FDC	34
รูปที่ 2.19 การเชื่อมต่อขาสัญญาณต่างๆ	40
รูปที่ 2.20 วงจรอนุญาตการทำงานของ ดิสก์ไครฟ์และควบคุมการทำงานของมอเตอร์	41
รูปที่ 2.21 วงจรเขียนข้อมูลลงในแผ่นดิสก์	42
รูปที่ 2.22 หน่วยการทำงานพื้นฐานของ MCS-51	45
รูปที่ 2.23 การกำหนดหน้าที่ขาสัญญาณของไอซี 8051	47
รูปที่ 2.24 โครงสร้างภายในของซีพียู Z-80	48
รูปที่ 2.25 รีจิสเตอร์ของซีพียู (Z-80)	49
รูปที่ 2.26 การจับรีจิสเตอร์	50
รูปที่ 2.27 การกระทำทางคณิตศาสตร์แบบ Double Precision	51

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 2.28 การทำงานทางคณิตศาสตร์แบบลอจิก	52
รูปที่ 2.29 รูปแบบของแฟลค	53
รูปที่ 2.30 ผลของแฟลค C หลังจากทำคำสั่งบวกและลบ	54
รูปที่ 2.31 เมื่อแฟลค P/V ทำหน้าที่เป็นพาริตีแฟลค	55
รูปที่ 2.32 การทำงานของแฟลค V	55
รูปที่ 2.33 บิตเครื่องหมายของตัวเลข	56
รูปที่ 2.34 การเซตแฟลคตัวทศช่วย	57
รูปที่ 2.35 ตัวอย่างการทำงานของแฟลค Z	57
รูปที่ 2.36 การเพิ่มขึ้นของข้อมูลในโปรแกรมเคาน์เตอร์	59
รูปที่ 2.37 การทำงานของสแตคและสแตคพอยท์เตอร์	60
รูปที่ 2.38 การทำงานของรีจิสเตอร์ I	61
รูปที่ 2.39 ขาต่างๆ ของชิพยู Z-80	63
รูปที่ 2.40 วงจรสร้างสัญญาณตอบรับการอินเตอร์รัพต์	63
รูปที่ 2.41 ตัวอย่างแผนผังเวลาของไมโคร โปรเซสเซอร์	67
รูปที่ 3.1 การเชื่อมต่ออุปกรณ์ภายนอกเข้ากับวงจร	70
รูปที่ 3.2 แผนผังการทำงานของวงจร	71
รูปที่ 3.3 แผนผังการทำงานของวงจรภาค FDC	72
รูปที่ 3.4 วงจรภาคหน่วยความจำ	74
รูปที่ 3.5 การทดลองและการออกแบบไมโคร โปรเซสเซอร์ MCS-51	75
รูปที่ 3.6 แผนผังการทำงานภายในของไมโคร โปรเซสเซอร์เบอร์ 8272A	76
รูปที่ 3.7 การออกแบบวงจรภาค FDC	79
รูปที่ 3.8 วงจรกำเนิดสัญญาณนาฬิกา	80
รูปที่ 3.9 วงจรกำเนิดสัญญาณนาฬิกา 8 MHz	80
รูปที่ 3.10 ค่าที่ได้จากการหารสัญญาณนาฬิกาในแต่ละเฮดท์พุต	81
รูปที่ 3.11 วงจรภาคบัฟเฟอร์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ในการค้า	81
รูปที่ 3.12 วงจรภาคดีโค๊ดเดอร์มิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการแก้ไข	83
รูปที่ 3.13 วงจรเฮดท์พุตรีจิสเตอร์	84

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 3.14 แผนผังการทำงานหลัก	86
รูปที่ 3.15 แผนผังการตั้งค่าเริ่มต้นของวงจร	87
รูปที่ 3.16 แผนผังการทำงานการรับคำสั่งและปฏิบัติงาน	89
รูปที่ 3.17 การเชื่อมต่อขาสัญญาณของ MCS-51	91
รูปที่ 3.18 วงจรขยายพอร์ตการทำงาน	92
รูปที่ 4.1 แผนผังการทำงานของการโอนย้ายข้อมูลจากซิงเกิลบอร์คไปเก็บไว้ในแรมของวงจร	94
รูปที่ 4.2 โปรแกรมการกำหนดข้อมูลลงในตำแหน่งหน่วยความจำเฉพาะ	94
รูปที่ 4.3 แผนผังการทำงานของการอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ค	95
รูปที่ 4.4 โปรแกรมการอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ค	96
รูปที่ 4.5 แผนผังการทำงานของการกำหนดข้อมูลลงในตำแหน่งหน่วยความจำ MCS-51 แล้วโอนไปยังซิงเกิลบอร์ค	97
รูปที่ 4.6 โปรแกรมกำหนดข้อมูลในหน่วยความจำแล้วโอนย้ายไปยังซิงเกิลบอร์ค	99
รูปที่ 4.7 แผนผังการทำงานของการกำหนดข้อมูลลงในหน่วยความจำ MCS-51 แล้วโอนไปยังซิงเกิลบอร์ค	100
รูปที่ 4.8 แผนผังการทำงานของการกำหนดข้อมูลลงในหน่วยความจำซิงเกิลบอร์ค แล้วโอนไปยัง MCS-51	103
รูปที่ 4.9 แผนผังการทำงานของการโอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51	104
รูปที่ 4.10 โปรแกรมการโอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51	113
รูปที่ 4.11 โปรแกรมที่ป้อนให้ Z-80	114
รูปที่ 4.12 โปรแกรมการป้อนค่าตำแหน่งเริ่มต้นและตำแหน่งสุดท้ายในการโอนย้ายข้อมูล	123
รูปที่ 4.13 โปรแกรมที่ป้อนให้ Z-80	124
รูปที่ 4.14 โปรแกรมป้อนคำสั่งการทำงานของ FDC Processor	124
รูปที่ 4.15 ความถี่ 8 MHz, 4 MHz และ 2MHz ที่ได้จากวงจรกำเนิดสัญญาณนาฬิกา	125
รูปที่ 4.16 ผลจากการทำงานภาคดีไค์เตอร์	126
รูปที่ 4.17 โปรแกรมการส่งค่าคงที่ให้กับเอาต์พุตรีจิสเตอร์	127
รูปที่ ก.1 ภาคจ่ายกำลังไฟฟ้า	135
รูปที่ ก.2 วงจรภาคควบคุมการโอนย้ายข้อมูล	135

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ ก.3 ภาคขยายพอร์ต Z-80	136
รูปที่ ก.4 ภาค FDC Controller	136
รูปที่ ก.5 ชิงเกิลบอร์ด MPF-1P	137
รูปที่ ก.6 กล่อง (ด้านหน้า)	137
รูปที่ ก.7 กล่อง (ด้านหลัง)	138
รูปที่ ข.1 วงจรภาคจ่ายกำลังไฟ	140
รูปที่ ข.2 ลายวงจรพิมพ์ภาคจ่ายกำลังไฟ (ด้านบน)	140
รูปที่ ข.3 ลายวงจรพิมพ์ภาคจ่ายกำลังไฟ (ด้านล่าง)	140
รูปที่ ข.4 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาคจ่ายกำลังไฟ	140
รูปที่ ข.5 วงจรภาคควบคุมการโอนย้ายข้อมูล	141
รูปที่ ข.6 ลายวงจรพิมพ์ภาคควบคุมการโอนย้ายข้อมูล (ด้านบน)	142
รูปที่ ข.7 ลายวงจรพิมพ์ภาคควบคุมการโอนย้ายข้อมูล (ด้านล่าง)	143
รูปที่ ข.8 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาคควบคุมการโอนย้ายข้อมูล	144
รูปที่ ข.9 วงจรภาคขยายพอร์ต Z-80	145
รูปที่ ข.10 ลายวงจรพิมพ์ภาคขยายพอร์ต Z-80	146
รูปที่ ข.11 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาคขยายพอร์ต Z-80	147
รูปที่ ข.12 วงจรภาค FDC Controller	148
รูปที่ ข.13 ลายวงจรพิมพ์ภาค FDC Controller (ด้านบน)	149
รูปที่ ข.14 ลายวงจรพิมพ์ภาค FDC Controller (ด้านล่าง)	150
รูปที่ ข.15 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาค FDC Controller	151
รูปที่ ค.1 ผังการทำงานของโปรแกรม	153

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความเป็นมา และความสำคัญของปริญญานิพนธ์

ปัญหาในการเขียนโปรแกรมบนซิงเกิลบอร์ด (Single Board) ที่มองเห็นได้ชัดคือ การที่เขียนโปรแกรมบนซิงเกิลบอร์ดแล้วไม่สามารถนำเอาโปรแกรมมาเก็บเองไว้ใช้ได้นานๆ หรือเรียกกลับมาใช้ได้อีก เนื่องจากการเก็บโปรแกรมลงบนแรม (Random Access Memory : RAM) จะทำให้โปรแกรมหายไปเมื่อปิดเครื่อง แนวทางแก้ไขคือ การเก็บข้อมูลลงในแผ่นดิสก์อย่างที่เราใช้กันโดยทั่วไปในปัจจุบัน ซึ่งทำให้สามารถนำเอาโปรแกรมที่เคยเขียนแล้ว และต้องการพัฒนา หรือต้องการนำเอามาใช้ใหม่เรียกกลับมาได้โดยไม่ต้องเขียนใหม่ ในกรณีที่ต้องการทำงานเชิงวิจัยหรือทดลองที่จำเป็นต้องใช้ซิงเกิลบอร์ดมาเป็นส่วนประกอบในการเก็บข้อมูล สามารถหลีกเลี่ยงอัตราเสี่ยงความปลอดภัยของข้อมูลได้โดยใช้แผ่นดิสก์เป็นตัวเก็บข้อมูลแทน ทำให้ได้ข้อมูลที่ชัดเจน และมีความถูกต้องของข้อมูลมากกว่าการเก็บข้อมูลที่อยู่ในแรมของซิงเกิลบอร์ด ในการออกแบบวงจรหากทำให้เป็นอุปกรณ์อินพุต/เอาต์พุตตัวหนึ่งของระบบ จะทำให้สามารถพัฒนาโปรแกรมที่นำมาควบคุมการทำงานได้อย่างอิสระ และเต็มความสามารถตามความต้องการของผู้ใช้งาน

1.2 ขีดความสามารถของโครงการ

สามารถบันทึกโปรแกรมหรือข้อมูลที่บันทึกอยู่ภายในหน่วยความจำของซิงเกิลบอร์ด โดยบันทึกในรูปแบบของ Text File ลงบนแผ่นฟลอปปี้ดิสก์ขนาด 3.5 นิ้ว ในโครงสร้างการจัดบันทึกของคอส และสามารถส่งข้อมูลจากฟลอปปี้ดิสก์ไปยังหน่วยความจำของซิงเกิลบอร์ด โดยการอ่านจากแฟ้มข้อมูลที่บันทึกไว้ได้

1.3 เนื้อหาโดยสังเขป

ในปริญญานิพนธ์ฉบับนี้ประกอบด้วยเนื้อหาในส่วนต่างๆ ดังนี้

บทที่ 2 ทฤษฎีและหลักการ กล่าวถึงองค์ประกอบที่สำคัญของปริญญานิพนธ์ ซึ่งประกอบด้วยส่วนย่อย 4 เรื่อง ดังนี้คือ สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) ฟลอปปีดิสก์ ในหัวข้อนี้จะนำเสนอเนื้อหาในส่วนของโครงสร้างทั้งทางด้านกายภาพ และโครงสร้างในมุมมองของการเขียนโปรแกรม รวมทั้งการจัดวางข้อมูลซึ่งมีการแบ่งออกเป็น ส่วนต่างๆ ตามหลักการวางโครงสร้างของคอส การถอดรหัสของข้อมูลที่ถูกรับที่ได้อ่านและเนื้อหา อื่นๆ ที่ควรทราบ

2) ดิสก์ไครฟ์ จะกล่าวเชื่อมโยงควบคุมกับตัวควบคุมฟลอปปีดิสก์ (Floppy Disk Controller : FDC) หรือวงจรภาคที่ทำหน้าที่ควบคุมการทำงานของภาคขับแผ่นฟลอปปีดิสก์ (Floppy Disk Drive : FDD) ในหัวข้อย่อยจะแสดงรูปแบบการทำงานของ FDC ว่ามีหลักการทำงานแบบใด โดย ครอบคลุมถึงส่วนของการมอดูเลตข้อมูลที่จะนำไปบันทึกลงบนแผ่นฟลอปปีดิสก์แบบต่างๆ และ ส่วนของการทำงานทางด้านกายภาพของดิสก์ไครฟ์

3) ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วยคุณสมบัติ, โครงสร้างภายใน และ การทำงานของไมโครคอนโทรลเลอร์

4) ไมโครโปรเซสเซอร์ Z-80 เป็นการศึกษาการทำงานของตัวไมโครโปรเซสเซอร์ Z-80 ซึ่ง ใช้เป็นไมโครโปรเซสเซอร์ที่ทำหน้าที่เป็นซีพียูในเชิงเกิลบอร์ด โดยจะกล่าวถึงโครงสร้าง, หลักการ ทำงาน และคาบเวลาการทำงานของตัวไมโครโปรเซสเซอร์เพื่อใช้อ้างอิงการทำงาน และการเชื่อมต่อวงจรเข้าไปควบคุมการทำงาน

บทที่ 3 การออกแบบและการสร้าง ซึ่งจะกล่าวถึงการออกแบบ และการสร้างวงจร รวมถึง โปรแกรมที่ใช้ในการควบคุมการทำงานซึ่งจะแบ่งได้ออกเป็น 4 ภาควงจร คือ ภาค FDC, ภาค ควบคุม, ภาคหน่วยความจำ และ ภาคอินเตอร์เฟส โดยจะนำเสนอรูปแบบการสร้างและออกแบบ วงจรในส่วนของภาค FDC และภาคอินเตอร์เฟส จะมีการวางโครงสร้างและรายละเอียดการเขียน โปรแกรมที่ทำหน้าที่ควบคุมประกอบด้วย

บทที่ 4 การทดลอง และผลการทดลอง มีเนื้อหาเกี่ยวกับการตรวจสอบความถูกต้องในการ ทำงานของภาคต่างๆ ผลที่ได้รับ และการทดลองเมื่อนำเอาส่วนต่างๆ รวมเข้าด้วยกัน

บทที่ 5 บทสรุป ปัญหา แนวทางการแก้ไขและพัฒนา จะบ่งบอกถึงสมรรถนะการทำงาน, ความผิดพลาด และจุดบกพร่องที่พบในวงจร หรือโปรแกรมที่จัดทำขึ้น รวมถึงที่มาของสาเหตุ และ แนวทางแก้ไข

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์

ภาคผนวก ค ผังการทำงานและโปรแกรม

ภาคผนวก ง รายละเอียดและคุณสมบัติของอุปกรณ์

ภาคผนวก จ คู่มือการใช้งาน

บทที่ 2

ทฤษฎี และหลักการ

2.1 กล่าวนำ

เนื้อหาของปริญาณิพนธ์ในบทนี้จะเป็นทฤษฎีและหลักการที่นำมาใช้ประกอบการสร้างโครงการ โดยประกอบด้วยฟลอปปีดิสก์, ดิสก์ไครฟ์, ไมโครคอนโทรลเลอร์ตระกูล MCS-51 และไมโครโปรเซสเซอร์ Z-80 ซึ่งจะได้กล่าวถึงดังต่อไปนี้

2.2 ฟลอปปีดิสก์

ไมโครคอมพิวเตอร์ 16 บิต เชื่อมต่อกับฟลอปปีดิสก์โดยการด์ที่ทำหน้าที่ควบคุมการทำงานของดิสก์ไครฟ์ คือ μ PD765 หรือ 8272 มีชื่อเป็นทางการว่า “ตัวควบคุมฟลอปปีดิสก์ (FDC)” ซึ่งเป็นชิปที่มีขีดความสามารถควบคุมการทำงานของฟลอปปีดิสก์ได้ดี และสามารถโปรแกรมการทำงานให้เป็นไปตามความต้องการได้ รายละเอียดของการเชื่อมต่อเพื่อให้เห็นโครงสร้าง และรายละเอียดการใช้งานของ 8272A สามารถปรับปรุงหรือทดลองนำไปใช้ ทำให้เกิดการเชื่อมต่อกับไมโครโปรเซสเซอร์ตระกูลอื่นได้โดยไม่ยาก

2.2.1 โครงสร้างของดิสก์เกตต์

ในระบบไมโครคอมพิวเตอร์นิยมเก็บข้อมูลโดยใช้ดิสก์ไครฟ์ 2 ชนิด ได้แก่ ฟลอปปีดิสก์ และฮาร์ดดิสก์ ความแตกต่างระหว่างดิสก์เกตต์ทั้งสองคือเนื้อสารที่ใช้สร้าง ฟลอปปีดิสก์สร้างขึ้นจากแผ่นงานไมลาร์ซึ่งเคลือบด้วยสารแม่เหล็กหรือออกไซด์ เนื่องจากโครงสร้างของฟลอปปีดิสก์เป็นไมลาร์จึงมีความอ่อนตัวเราจึงเรียกว่า ฟลอปปีดิสก์ หรือบางทีก็จะเรียกว่าดิสก์เกตต์ ส่วนฮาร์ดดิสก์เป็นแผ่นโลหะที่เคลือบด้วยออกไซด์ โครงสร้างของฟลอปปีดิสก์ที่ใช้งานกันทั่วไปนั้นจะประกอบด้วยส่วนสำคัญ 3 ส่วน ดังนี้

1) ตัวแผ่นดิสก์ (Magnetic Disk) มีลักษณะเป็นแผ่นกลมทำด้วยสารโพลีเอสเตอร์ แล้วเคลือบด้วยสารแม่เหล็กบางๆ

2) ซองหุ้ม (Jacket) เป็นซองสี่เหลี่ยมที่บรรจุแผ่นดิสก์เอาไว้ทำจากสารพีวีซี โดยมีหน้าที่เป็นส่วนป้องกันไม่ให้ฝุ่นละอองหรือไขมันไปทำลายผิวดิสก์ และป้องกันประจุไฟฟ้าสถิต

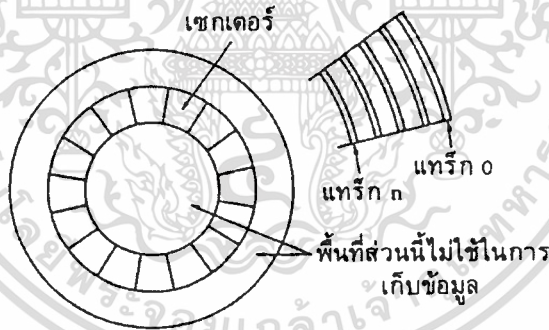
3) แผ่นทำความสะอาด (Liner) เป็นแผ่นที่ทอจากไฟเบอร์มีความหนาประมาณ 0.2 มิลลิเมตรกั้นระหว่างซองหุ้มกับตัวแผ่นดิสก์ ที่ทำหน้าที่รักษาความสะอาดของดิสก์เกตต์ที่มีการนำไปใช้

2.2.2 แทร็กและเซกเตอร์

การอ่านและเขียนข้อมูลของดิสก์เกตต์ทำในขณะที่ดิสก์เกตต์หมุนอยู่ ดังนั้นกลุ่มของข้อมูลที่บรรจุในดิสก์เกตต์อยู่ในแนวแถบกลมเรียกว่า “แทร็ก (Track)” ดังแสดงในรูปที่ 2.1 โดยระหว่างแทร็กที่เก็บข้อมูลจะมีแทร็กที่ไม่ใช้ในการเก็บข้อมูลคั่นไว้ เพื่อลดความผิดพลาดในการอ่านหรือเขียนข้อมูล จำนวนแทร็กต่อความยาวตัดขวางเป็นการบอกความจุของดิสก์เกตต์วิธีหนึ่ง เช่น ดิสก์เกตต์มีขนาด 3.5 นิ้ว ที่ใช้ในไมโครคอมพิวเตอร์จะมีแทร็กข้อมูล 80 แทร็ก และเนื่องจากใน 1 แทร็กจะสามารถเก็บข้อมูลได้มาก ดังนั้นในแต่ละแทร็กจะทำการแบ่งออกเป็นหลายเซกเตอร์ (Sector) เพราะสะดวกในการค้นหาข้อมูล โดยดิสก์เกตต์ขนาด 3.5 นิ้ว จะมีจำนวน 18 เซกเตอร์ต่อหนึ่งแทร็ก โดยในหนึ่งเซกเตอร์มีข้อมูล 512 ไบต์ ความจุของดิสก์เกตต์หาได้จาก

ความจุ = จำนวนหน้า × จำนวนเซกเตอร์ต่อแทร็ก × จำนวนแทร็ก × จำนวนไบต์ต่อเซกเตอร์

ดังนั้นดิสก์เกตต์ขนาด 3.5 นิ้ว ที่ใช้ในไมโครคอมพิวเตอร์ 16 บิต จะมีความจุเท่ากับ $2 \times 18 \times 80 \times 512 = 1,474,560$ ไบต์

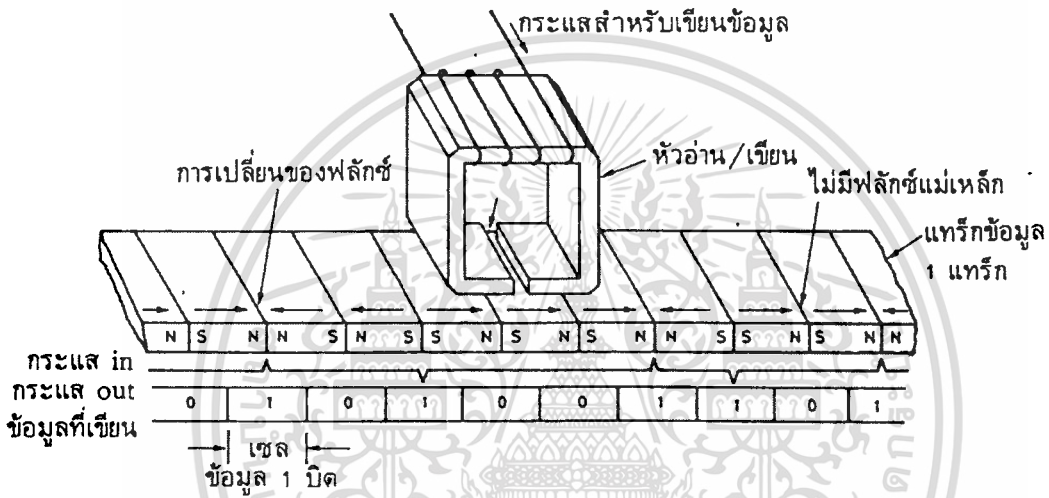


รูปที่ 2.1 การจัดแทร็กและเซกเตอร์ภายในดิสก์เกตต์

2.2.3 การเก็บข้อมูลในแผ่นฟลอปปีดิสก์

การเก็บข้อมูลในฟลอปปีดิสก์ทำได้โดยการเหนี่ยวนำของสารแม่เหล็กให้เกิดขั้วแม่เหล็กบนแผ่น โดยหัวอ่านดิสก์มีวิธีการอ่านหรือเขียนฟลอปปีดิสก์ หัวอ่านจะถูกกด (Loaded) จนสัมผัสกับดิสก์เกตต์ ซึ่งหัวอ่านเขียนจะประกอบด้วยขดลวดพันบนแกนเฟอไรต์ ซึ่งจะมีช่องอากาศ (Air Gap) อยู่ในช่วงที่สัมผัสกับดิสก์เกตต์ดังรูปที่ 2.2 การเขียนข้อมูลลงบนดิสก์เกตต์ทำได้โดยการเหนี่ยวนำสารแม่เหล็กบนดิสก์เกตต์ ทำให้เกิดทิศทางของฟลักซ์ที่แตกต่างกัน โดยวิธีการเขียนลอจิก “1” จะเขียนโดยการกลับทิศทางของกระแสที่ไหลผ่านหัวอ่านหรือเขียน เพื่อที่ทำให้ฟลักซ์

แม่เหล็กบนแผ่นมีทิศทางตรงข้ามจากเดิมในขณะที่ลอจิก “0” จะไม่มีการกลับทิศของฟลักซ์แม่เหล็ก ในทางกลับกันเมื่อกระทำการอ่านข้อมูลจากดิสก์เกตต์ เมื่อหัวอ่านพบข้อมูลลอจิก “1” จะทำให้เกิดกระแสไหลตามกฎของเลนส์ เมื่อผ่านวงจรรีเลย์ทรอนิกส์เราก็สามารถอ่านค่าข้อมูลที่ถูกต้องออกมาได้ ข้อมูลที่บรรจุบนดิสก์เกตต์จะมีการสร้างรูอินเด็กซ์ (Index Hold) ขึ้น ขณะที่ดิสก์เกตต์หมุนมาถึงตำแหน่งรูอินเด็กซ์แสดงว่าเป็นตำแหน่งของเซกเตอร์แรกของทุกแทร็ก โดยการตรวจจับจะกระทำโดยตัวดิสก์ไดรฟ์



รูปที่ 2.2 หัวอ่านบันทึกดิสก์ในขณะทำการบันทึก

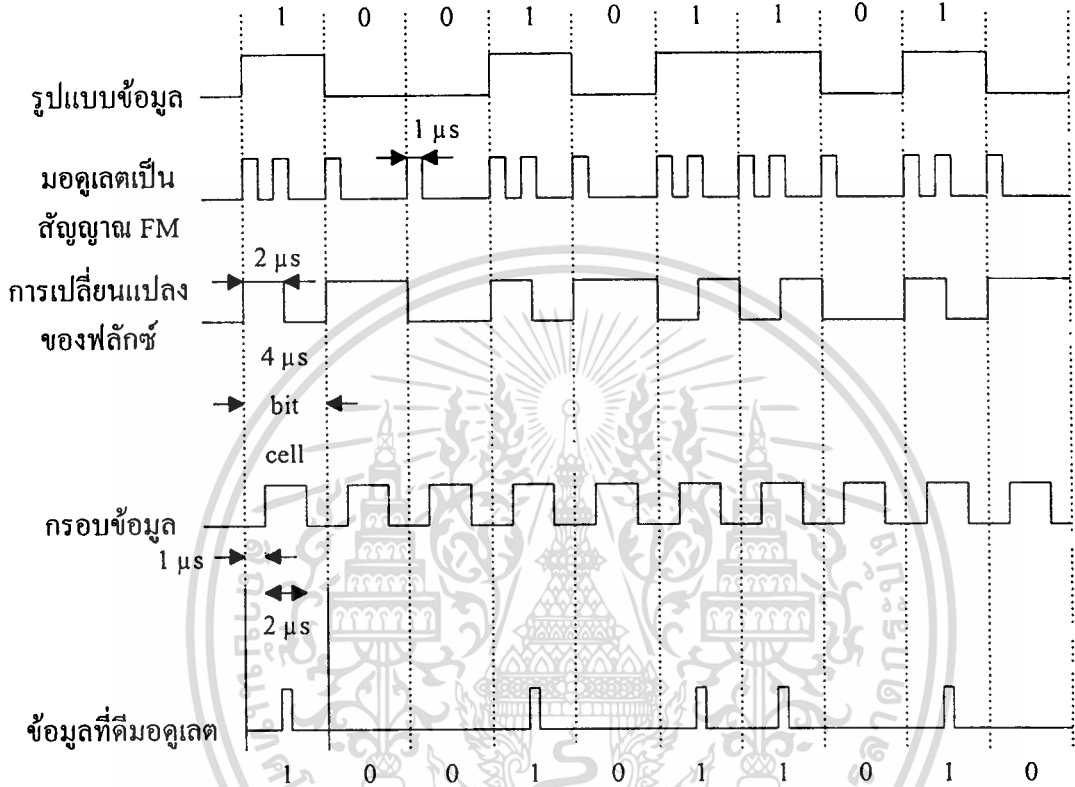
2.2.4 การมอดูเลตของข้อมูล

การเก็บข้อมูลมีความจำเป็นที่จะต้องทำการมอดูเลต เพื่อลดความผิดพลาดในการอ่านและเขียนข้อมูล เนื่องจากความเร็วของดิสก์เกตต์ไม่คงที่ และเพื่อที่จะเพิ่มความจุในการเก็บข้อมูลของดิสก์เกตต์ ซึ่งหลักการของการมอดูเลตที่ใช้ในฟลอปปี้ดิสก์ทำโดยการผสมสัญญาณนาฬิกากับข้อมูล วิธีการมอดูเลตที่ใช้ในฟลอปปี้ดิสก์มีหลายวิธีดังนี้

1) การมอดูเลตทางความถี่ (Frequency Modulation : FM) วิธีการนี้ใช้การบันทึกข้อมูลลงบนดิสก์เกตต์แบบความหนาแน่นเท่าเดียว (Single Density) หลักการมอดูเลตทำโดยการใส่ข้อมูลระหว่างสัญญาณนาฬิกาดังรูปที่ 2.3

จะเห็นว่า การมอดูเลตแบบ FM จะมีอัตราการบันทึกของข้อมูลต่ำ เพราะอัตราการบันทึกข้อมูลขึ้นอยู่กับจำนวนครั้งของการเปลี่ยนแปลงของฟลักซ์ต่อวินาที และในขณะที่ทำการส่งลอจิก “1” ข้อมูลที่ถูกมอดูเลตจะมีอัตราการเปลี่ยนแปลงของฟลักซ์สูงกว่าขณะทำการส่งลอจิก “0”

ดังนั้นสัญญาณนาฬิกาหรืออัตราการส่งของข้อมูลจะต้องมีค่าเพียงครึ่งหนึ่งของอัตราการบันทึกข้อมูลของดิสก์เกตต์



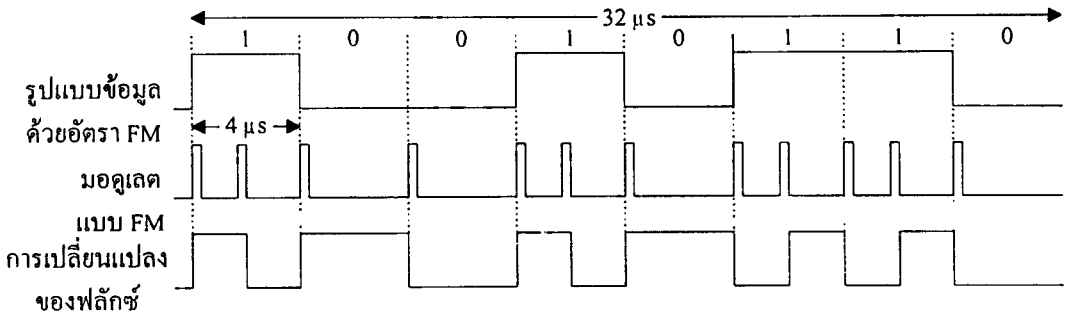
รูปที่ 2.3 การมอดูเลตโดยวิธี FM

2) การมอดูเลตทางความถี่ที่มีการดัดแปลง (Modified Frequency Modulation : MFM) วิธีนี้ใช้ในการบันทึกข้อมูลลงบนดิสก์เกตต์แบบความหนาแน่น 2 เท่า (Double Density) ซึ่งพัฒนาหลักการมาจากวิธี FM วิธี MFM จะพยายามกำจัดสัญญาณนาฬิกาที่ไม่จำเป็นออกไป โดยมีหลักการของการมอดูเลตดังแสดงในรูปที่ 2.4

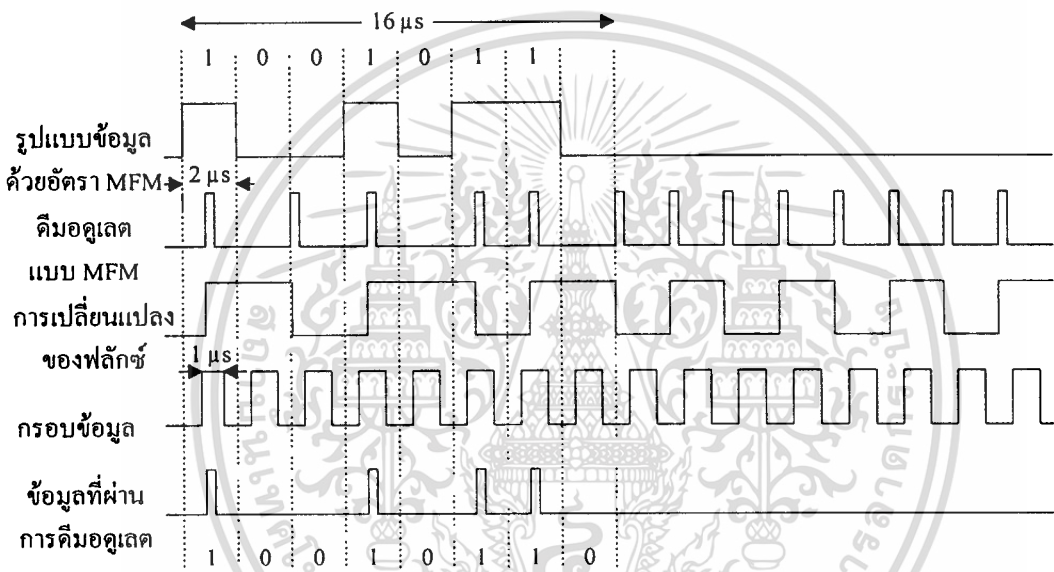
2.1) เมื่อบันทึกข้อมูลลอจิก “1” ทำการสร้างพัลส์ในช่วงตรงกลางของบิตเซลล์เหมือนวิธี FM แต่ตัดสัญญาณนาฬิกาออก

2.2) เมื่อบันทึกลอจิก “0” ทำการส่งสัญญาณนาฬิกาขั้วลบเมื่อข้อมูลก่อนหน้าเป็นลอจิก “1”

เอกสารนี้เป็นการมอดูเลตแบบ MFM จะได้อัตราการส่งข้อมูลมีค่าเท่ากับอัตราการบันทึกของฟลอปปีการดิสก์การแก้ไขใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)



(ข)

รูปที่ 2.4 เปรียบเทียบการมอดูเลตแบบ FM(ก) กับการมอดูเลตแบบ MFM(ข)

3) การมอดูเลตทางความถี่ที่มีการดัดแปลงจาก MFM (Modified Modified Frequency Modulation : $M^2\text{FM}$) วิธีนี้ดัดแปลงจากวิธี MFM เล็กน้อย คือในกรณี $M^2\text{FM}$ จะส่งสัญญาณนาฬิกาเมื่อข้อมูลที่ส่งเป็นลอจิก "0" ติดต่อกัน 2 บิตขึ้นไป

2.2.5 การฟอร์แมตฟลอปปีดิสก์

ในการใช้งานฟลอปปีดิสก์ก่อนการบันทึกข้อมูลลงบนฟลอปปีดิสก์จำเป็นต้องทำการฟอร์แมต (Format) แผ่นนั้นเสียก่อน การฟอร์แมต คือ การกำหนดตำแหน่งของแทร็ก เซกเตอร์และจำนวนความจุต่อเซกเตอร์ เนื่องจากการเก็บข้อมูลลงบนฟลอปปีดิสก์จะต้องเก็บข้อมูลที่ช่วยในการค้นหาข้อมูลที่ต้องการ ในการฟอร์แมตจะมีการจัดเนื้อที่ในฟลอปปีดิสก์ดังรูปที่ 2.5

1) **พรีแอมเบิล** เป็นข้อมูลที่เก็บในช่วงที่ดิสก์ไดรฟ์ตรวจจับรูอินเด็กซ์ โดยทำการบันทึกข้อมูล FF และ 00

2) **มาสก์อินเด็กซ์แอดเดรส** เป็นข้อมูล 1 ไบต์ ใช้สำหรับบอกสถานะของข้อมูลในแทร็กนั้น ดังแสดงในรูปที่ 2.6

3) **ช่องว่าง 1 โฟสอินเด็กซ์** เป็นช่องว่างระหว่างมาสก์อินเด็กซ์แอดเดรสกับเซกเตอร์แรกของแทร็ก

4) **เซกเตอร์** เป็นช่วงที่ใช้เก็บข้อมูล โดยในแต่ละเซกเตอร์มีการจัดรูปแบบดังนี้

หมายเลข ID ของเรคคอร์ด เป็นส่วนของข้อมูลที่ใส่บอกตำแหน่งของเซกเตอร์นั้นๆ ความจุของข้อมูลต่อเซกเตอร์ และค่า CRC (Cyclic Redundancy Check) ที่ใช้ในการตรวจสอบความถูกต้องของการบันทึกและการอ่านข้อมูล

ช่องว่างที่ 2 ช่องว่าง ID เป็นช่องว่างไว้คั่นหมายเลขเรคคอร์ด ID กับฟิลด์ข้อมูล

ฟิลด์ข้อมูล เป็นส่วนที่ใช้บันทึกข้อมูลโดย จะใช้ 1 ไบต์แรกบอกสถานะภาพของข้อมูลและ 2 ไบต์สุดท้ายเก็บค่า CRC

ช่องว่าง 3 ช่องว่างข้อมูล เป็นช่วงที่เว้นไว้คั่นระหว่างแต่ละเซกเตอร์

ช่องว่าง 4 พรีอินเด็กซ์ เป็นช่วงที่เว้นจากเซกเตอร์สุดท้ายไปจนถึงพรีแอมเบิล

2.2.6 โครงสร้างของดิสก์ไดรฟ์

ส่วนประกอบที่สำคัญของดิสก์ไดรฟ์แสดงดังรูปที่ 2.7 ซึ่งประกอบด้วย

1) **มอเตอร์ขับเคลื่อน** เป็นมอเตอร์ที่ทำหน้าที่ในการหมุนดิสก์เกตต์ ซึ่งสปินเดิลมอเตอร์จะต้องมีความเร็วที่เที่ยงตรงมาก

2) **สเต็ปมอเตอร์** เป็นมอเตอร์ที่ใช้ในการเลื่อนหัวอ่านหรือเขียนไปยังแทร็กที่ต้องการ ซึ่งวิธีการเลื่อนหัวอ่านเขียนที่นิยมใช้ได้แก่ การใช้แผ่นแถบโลหะที่บาง เคลื่อนที่ไปมาได้ ดังรูปที่ 2.7

3) **ตัวตรวจสอบแทร็ก 0** ทำหน้าที่ตรวจสอบหัวอ่านหรือเขียนอยู่ในตำแหน่งแทร็ก 0 หรือไม่

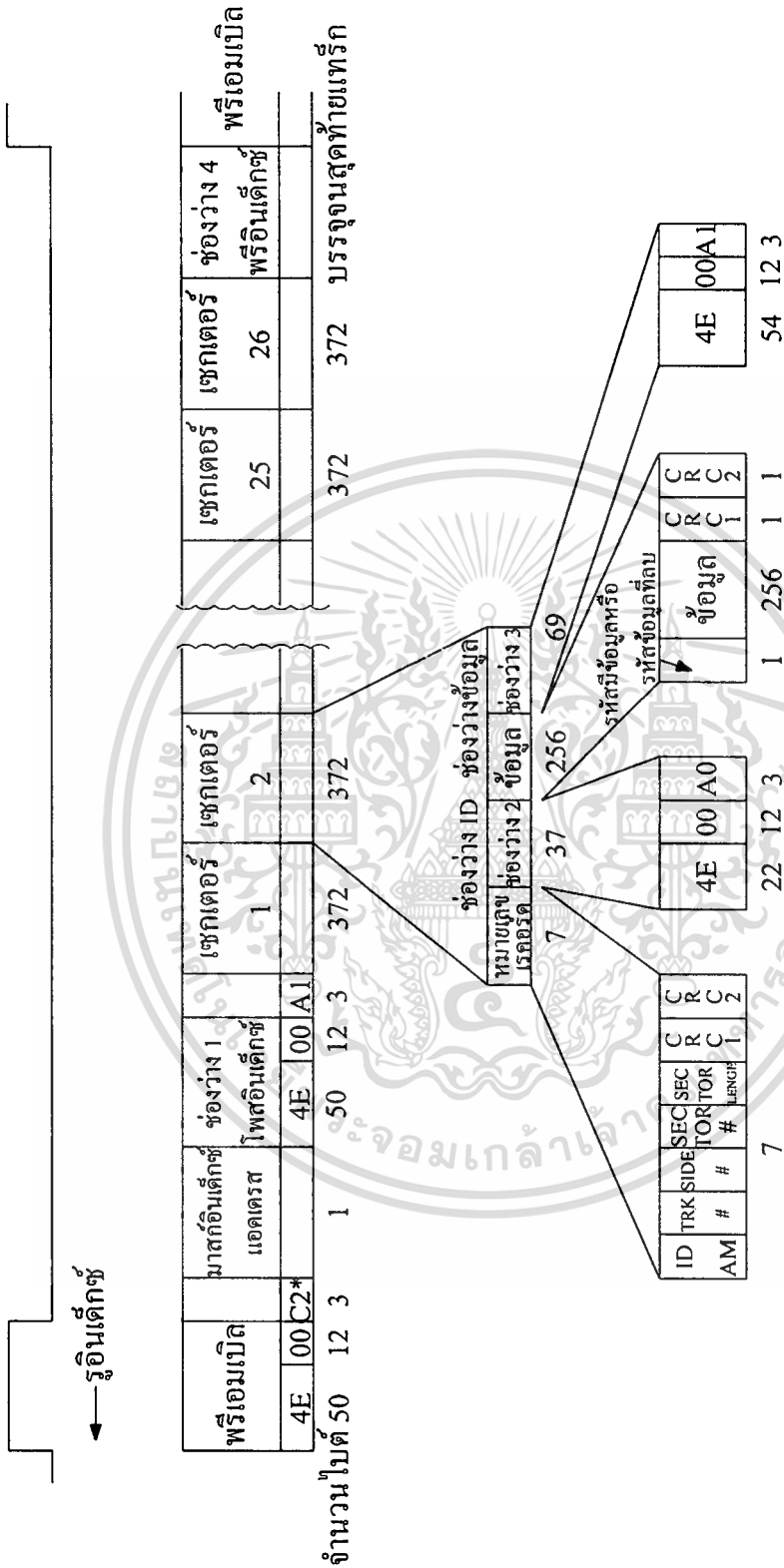
4) **ตัวตรวจจับรูอินเด็กซ์** มีหน้าที่ตรวจสอบดิสก์เกตต์จะหมุนไปถึงที่ตำแหน่งเริ่มต้นของ เซกเตอร์แรกหรือยัง

5) **ตัวตรวจจับการป้องกันการเขียน** ทำหน้าที่ตรวจสอบแผ่นดิสก์เกตต์มีการป้องกันการเขียนข้อมูลลงไปหรือไม่

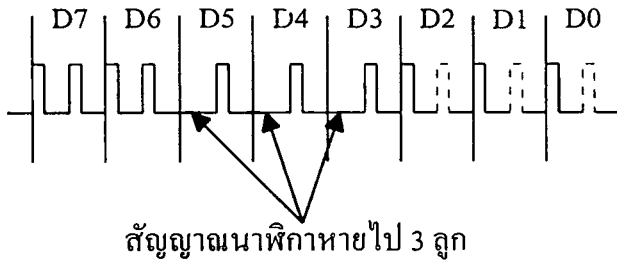
6) **หัวอ่านเขียน** คือ หัวอ่านเขียนดิสก์เกตต์

สำหรับการติดต่อระหว่างดิสก์ไดรฟ์กับวงจรควบคุมดิสก์ไดรฟ์ที่มีสัญญาณที่ใช้ในการติดต่อ ดังรูปที่ 2.8 ซึ่งมีรายละเอียดดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

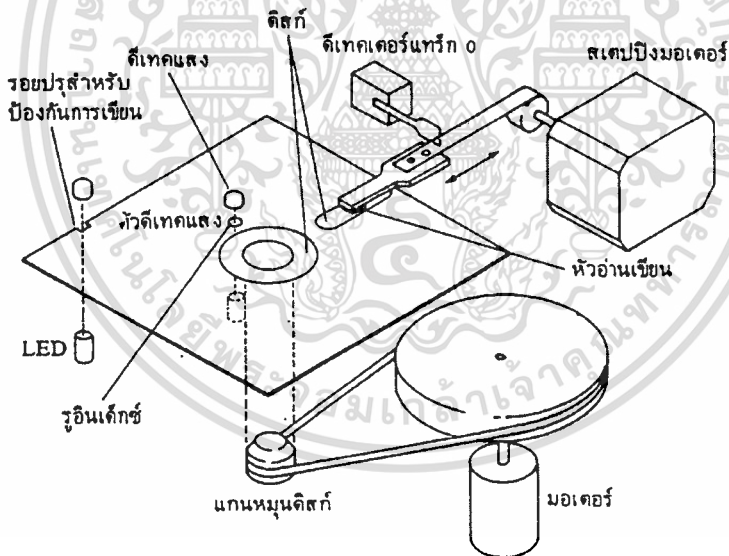


รูปที่ 2.5 ตำแหน่งของข้อมูล เมื่อทำการฟอร์แมตแผ่นฟลอปปีดิสก์แบบ 3740



D2	D1	D0	มาสก์แอดเดรส
0	0	0	ข้อมูลถูกลบ
0	1	1	มีข้อมูล
1	0	0	อินเด็กซ์
1	1	0	ID

รูปที่ 2.6 การเก็บสถานะภาพของดิสก์เกิดในมาสก์อินเด็กซ์แอดเดรส

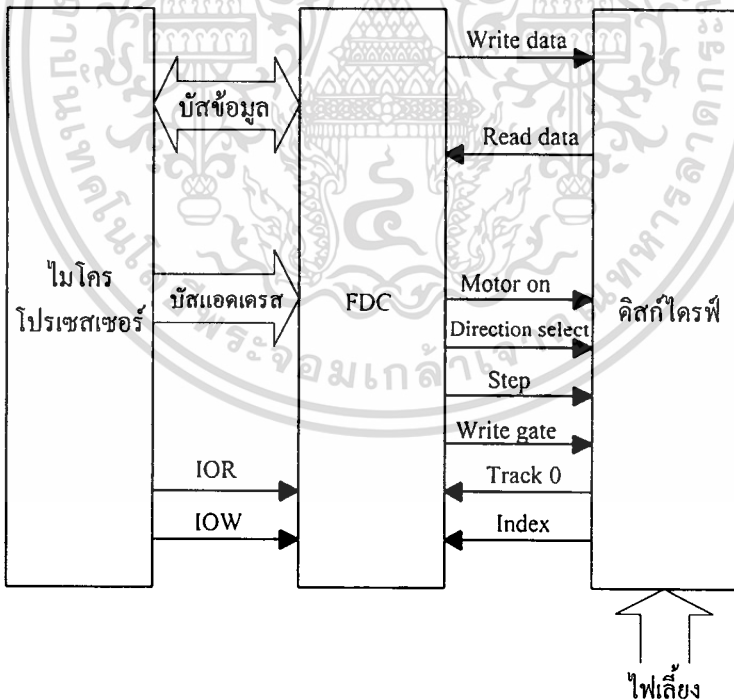


รูปที่ 2.7 โครงสร้างพื้นฐานของดิสก์ไดรฟ์

- 1) การอินทิเกรตมอเตอร์ เป็นสัญญาณควบคุมสปินเดิลมอเตอร์เพื่อให้ดิสก์เกิดหมุน
 - 2) การเลือกไดรฟ์ เป็นสัญญาณที่เป็นตัวบอกว่ามีการติดต่อระหว่างวงจรควบคุมดิสก์ไดรฟ์กับดิสก์ไดรฟ์
 - 3) การเขียนข้อมูล คือ ข้อมูลที่ส่งมาจากวงจรควบคุมดิสก์ไดรฟ์เพื่อที่จะเขียนลงบนดิสก์เกิด
- เอกสารนี้เป็นทรัพย์สินทางปัญญาของสำนักงานส่งเสริมการค้าในต่างประเทศ ณ นครเซี่ยงไฮ้ โดยจะดำเนินการค้าสำหรับรูปที่ 2.8 สัญญาณในการติดต่อระหว่างดิสก์ไดรฟ์และวงจรควบคุมดิสก์ไดรฟ์ โดยข้อมูลที่ใช้

ส่งมาจะเป็นซีแควนเชียล

- 4) การอ่านข้อมูล คือ ข้อมูลส่งจากดิสก์ไครฟ์ไปยังวงจรควบคุมดิสก์ไครฟ์ เป็นข้อมูลที่อ่านจากดิสก์เกตต์ที่ต้องการ โดยเป็นข้อมูลซีแควนเชียล
- 5) อินาเบิลการเขียน เป็นสัญญาณที่อนุญาตให้มีการเขียนข้อมูลลงบนดิสก์เกตต์
- 6) เลือกว่าน เป็นสัญญาณกำหนดหัวอ่านหรือเขียน ว่าต้องการทำการอ่านหรือเขียนข้อมูลจากหัวอ่านด้านใดในการอ่านเขียนดิสก์เกตต์ 2 ด้าน
- 7) ทิศทาง เป็นสัญญาณที่ใช้ควบคุมการเลื่อนตำแหน่งของหัวอ่านหรือเขียนว่าจะให้เลื่อนออกหรือเข้าจากตำแหน่งแทร็กเดิม เพื่อค้นหาแทร็กใหม่
- 8) สเตป เป็นสัญญาณกำหนดให้มีการเลื่อนหัวอ่านและเขียนไป 1 แทร็ก ในทิศทางที่กำหนดโดยสัญญาณ Direction
- 9) แทร็ก 0 เป็นสัญญาณที่ส่งจากดิสก์ไครฟ์ไปยังวงจรควบคุมดิสก์ไครฟ์ เพื่อบอกว่าตำแหน่งหัวอ่านและเขียนอยู่ในตำแหน่งแทร็ก 0



รูปที่ 2.8 สัญญาณในการติดต่อระหว่างดิสก์ไครฟ์และวงจรควบคุมดิสก์ไครฟ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเฉพาะเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาต
 10) อินเด็กซ์ เป็นสัญญาณที่ส่งจากดิสก์ไครฟ์ไปยังวงจรควบคุมดิสก์ไครฟ์ เมื่อรูอินเด็กซ์ถูก
 ตรวจจับได้

11) ป้องกันการเขียน เป็นสัญญาณที่ส่งจากดิสก์ไครฟ์ไปยังวงจรควบคุมดิสก์ไครฟ์เพื่อบอกว่า ดิสก์เกตต์มีการป้องกันการเขียนข้อมูล

2.2.7 ข้อกำหนดรายละเอียดที่สำคัญของดิสก์ไครฟ์

- 1) **Track to Track Time** คือ เวลาที่ใช้ในการเลื่อนหัวอ่านหรือเขียนไป 1 แตร็ก
- 2) **Seek Time** คือ เวลาที่ใช้ในการเลื่อนหัวอ่านเขียนจากแตร็กใดแตร็กหนึ่ง ไปยังแตร็กที่ต้องการ โดยค่าเวลานี้มีค่าเท่ากับจำนวนแตร็กที่เลื่อนคูณกับ Track to Track Time
- 3) **Setting Time** คือ ช่วงเวลาที่ต้องรอให้หัวอ่านหรือเขียนหยุดนิ่งหลังจากที่ทำการค้นหาเพื่อทำการอ่านหรือเขียนได้
- 4) **Head Load Time** คือ เวลาที่ใช้ในการให้ดิสก์เกตต์หมุน

2.2.8 โครงสร้างดิสก์ตามแบบของดอส

ดอส (DOS) เป็นระบบปฏิบัติการที่สร้างแนวทางการใช้ดิสก์ไว้ ระบบปฏิบัติการแต่ละตัวที่ใช้กับพีซีมีหลายชนิด ซึ่งล้วนมีรูปแบบของตัวเอง และเนื่องจากดอสเป็นระบบปฏิบัติการที่ผู้ใช้พีซีส่วนมากจะนำมาใช้งาน ดังนั้นจึงจะกล่าวถึงแนวทางที่ดอสใช้ในการจัดการดิสก์เท่านั้น

ในที่นี้จะศึกษาความรู้ขั้นพื้นฐานในการที่ดอสใช้ดิสก์, ข้อกำหนดทางเทคนิคที่อยู่ภายใต้การควบคุมของดอส และสำรวจว่าข้อมูลในไฟล์ต่างๆ ที่ดอสจัดการมีลักษณะอย่างไร โดยเฉพาะจากรูปแบบของข้อมูลที่นิยมที่สุดอันได้แก่ ไฟล์รหัสแอสกี

1) เริ่มรู้จักดิสก์ของดอส

จากพื้นฐานของดิสก์ทำให้ทราบว่าดิสก์เป็นสื่อเก็บข้อมูลที่มี 3-4 มิติ 3 มิติแรก ได้แก่ แตร็ก หรือ ซีลินเดอร์ (เป็นมิติรัศมี) ด้าน (เป็นมิติแนวตั้ง) และเชกเตอร์ (เป็นมิติวงกลม) ซึ่งเป็นตัวบอกตำแหน่งของแต่ละเชกเตอร์บนดิสก์ ขนาดความจุของแต่ละเชกเตอร์เป็นมิติที่ 4 และจากผลคูณของ 3 มิติแรกจะได้จำนวนเชกเตอร์ทั้งหมดในดิสก์ ซึ่งเป็นขอบเขตเนื้อที่ที่ดอสสามารถใช้ดิสก์ได้ และเมื่อนำจำนวนเชกเตอร์กับขนาดของเชกเตอร์มาคูณกันจะให้ความจุของดิสก์เป็นจำนวนไบต์

เชกเตอร์ เป็นหน่วยทำงานขั้นพื้นฐานของดิสก์ซึ่งเป็นหน่วยที่เล็กที่สุดที่ใช้ในการอ่านและเขียนข้อมูลลงดิสก์ กล่าวคือ ในการอ่านหรือเขียนดิสก์แต่ละครั้งจะกระทำที่ละเชกเตอร์ โดยการกำหนดเชกเตอร์ที่ต้องการอ่านหรือเขียน

ดอสจะกำหนดให้เชกเตอร์บนดิสก์อยู่เรียงตามลำดับเป็นแถวไป จากเชกเตอร์แรกถึงเชกเตอร์สุดท้าย และดอสจะกำหนดหมายเลขให้เชกเตอร์ตามลำดับจาก 0 สำหรับเชกเตอร์ในด้านแรกของซีลินเดอร์แรก จนถึงเชกเตอร์สุดท้ายของซีลินเดอร์สุดท้าย ดังนั้นการทำงานและการกำหนดพื้นที่ที่ใช้เชกเตอร์ของดอสจะทำตามเลขประจำเชกเตอร์เหล่านี้เรียงตามลำดับ และเมื่อทำการเขียนหรืออ่านข้อมูลลงดิสก์ ดอสจะทำการแปลงเลขเชกเตอร์ให้เป็นตำแหน่งในดิสก์จริงๆ

นอกจากนี้ คอายังทำหน้าที่เป็นตัวกลางระหว่างโปรแกรมที่ต้องการใช้ข้อมูลกับการทำงานในหน่วยของเซกเตอร์ว่าจะต้องการอ่านเซกเตอร์ใด และเป็นจำนวนกี่เซกเตอร์

การที่คอสมองว่าดิสก์ประกอบด้วยเซกเตอร์ที่เรียงตามลำดับนั้น ทำให้คอจัดโครงสร้างของดิสก์ง่ายขึ้น แต่มีข้อเสียได้แก่ กรณีที่ข้อมูลที่ต้องการอ่านอยู่บนซีลินเดอร์ทำให้ต้องเสียเวลาในการเลื่อนหัวอ่านข้อมูล เมื่อเปรียบเทียบกับกรณีที่ข้อมูลที่ต้องการอ่านอยู่ในเซกเตอร์ที่อยู่ในซีลินเดอร์เดียวกัน แบบหลังจะอ่านได้เร็วกว่า แต่การที่คอเปลี่ยนการจัดการดิสก์ให้อยู่ในรูปของเซกเตอร์นี้ ทำให้คอไม่รู้และไม่สนใจว่าเซกเตอร์แต่ละเซกเตอร์อยู่บนซีลินเดอร์เดียวกันหรือไม่ นอกจากนี้ลักษณะการทำงานภายใต้คอเช่นนี้ ทำให้คอรุ่นที่ต่ำกว่าเวอร์ชัน 4.0 มีข้อจำกัดในเรื่องของขนาดความจุของดิสก์

2) ข้อจำกัด 32 เมกะไบต์ในคอรุ่นที่ต่ำกว่าเวอร์ชัน 4.0

การที่คอจัดโครงสร้างของดิสก์เป็นเส้นตรงเรียงตามลำดับนั้นทำให้เกิดข้อจำกัดขึ้น คือคอจะใช้ดิสก์ได้เพียง 32 เมกะไบต์

ข้อจำกัดเกิดขึ้นจากขนาดเซกเตอร์มาตรฐานที่คอกำหนดคือ 512 ไบต์ และเลขประจำเซกเตอร์ที่คอกำหนดขึ้น โดยใช้ตัวเลขเหล่านี้ตามแบบจำนวนเต็มขนาด 16 บิต ซึ่งจะมีตัวเลขได้ประมาณ 64 k เซกเตอร์ และเนื่องจากเซกเตอร์มีขนาด 512 ไบต์ จึงทำให้ขนาดสูงสุดของดิสก์ที่เป็นไปได้คือ 32 M (64 k คูณ 512 ได้ประมาณ 32 ล้าน)

ทางแก้ไขข้อจำกัดนี้ที่ปรากฏในคอเวอร์ชัน 4.0 คือ การเปลี่ยนแปลงระบบไฟล์ให้ใช้เนื้อที่ในการเก็บหมายเลขเซกเตอร์ได้มากขึ้น (เดิมใช้เพียง 16 บิต)

ก่อนที่จะนำดิสก์มาใช้งานจะต้องทำการฟอร์แมตดิสก์ด้วยคอเสียก่อน ซึ่งหมายความว่ามีการกำหนดและจัดโครงสร้างของดิสก์ตามที่คอต้องการด้วยการใช้โปรแกรม FORMAT โดยที่คำสั่ง FORMAT จะทำให้ดิสก์อยู่ในสถานะที่คอคาดว่าดิสก์จะต้องเป็น ซึ่งก็ขึ้นอยู่กับชนิดของคอด้วยว่าเป็นดิสก์เกตต์ หรือฮาร์ดดิสก์ หลังจากนั้น FORMAT จะวางรูปแบบโครงสร้างของคอลงในดิสก์รวมถึงโครงสร้างที่ใช้ในการกำหนดวิธีการและตำแหน่งที่จะจัดเก็บไฟล์

3) การฟอร์แมตทางกายภาพและตรรกะ

การฟอร์แมตดิสก์มี 2 แบบ ได้แก่ การฟอร์แมตทางกายภาพ (Physical Format) และการฟอร์แมตทางตรรกะ (Logical Format) ซึ่งจะอธิบายถึงความแตกต่างระหว่าง 2 แบบ ดังนี้

3.1) การฟอร์แมตทางกายภาพ เกี่ยวข้องกับการสร้างเซกเตอร์บนดิสก์ ซึ่งจะเสร็จสมบูรณ์เมื่อมีการทำเครื่องหมายตำแหน่งที่ของเซกเตอร์ และในตำแหน่งที่สามารถใช้เก็บข้อมูลลงบนดิสก์ โดยในส่วนที่ใช้เก็บข้อมูลจะทำการใส่ข้อมูลล่อไว้ โดยปกติแล้วดิสก์เกตต์ใหม่จะยังไม่ทำการฟอร์แมตทางกายภาพมา ในขณะที่ฮาร์ดดิสก์ใหม่จะผ่านการฟอร์แมตทางกายภาพมาแล้ว ที่มีการนำไปใช้

3.2) การฟอร์แมตทางตรรกะ เป็นการทำให้ดิสก์มีมาตรฐานเดียวกับระบบปฏิบัติการที่จะใช้งาน นั่นคือ เมื่อคอสฟอร์แมตดิสก์แล้วจะสร้างโครงสร้างทางตรรกะตามที่คอสใช้ การฟอร์แมตทางตรรกะเป็นแนวทางที่คอสหรือระบบปฏิบัติการอื่นๆ ใช้สำหรับการเข้าถึงดิสก์

เราสามารถใช้อำสั่ง FORMAT ที่ติดมากับแผ่นคอสในการฟอร์แมตดิสก์ได้ แต่การใช้อำสั่งนี้กับดิสก์เกตต์และฮาร์ดดิสก์จะมีความแตกต่างกัน เพราะการใช้ FORMAT กระทำต่อดิสก์จะทำการฟอร์แมตทั้งทางกายภาพ และทางตรรกะด้วย เนื่องจากเนื้อที่ในดิสก์เกตต์จะถูกควบคุมโดยระบบปฏิบัติการที่ทำการฟอร์แมต (ในที่นี้คือคอส) ในขณะที่การใช้ฮาร์ดดิสก์นั้น เราสามารถแบ่งเนื้อที่ภายในฮาร์ดดิสก์ให้ถูกควบคุมด้วยระบบปฏิบัติการหลายๆ ระบบได้ ดังนั้นในการออกคำสั่ง FORMAT บนฮาร์ดดิสก์จะทำการฟอร์แมตเฉพาะแบบตรรกะเท่านั้น เนื่องจากการฟอร์แมตทางกายภาพจะส่งผลไปรบกวนส่วนอื่นๆ ของดิสก์ที่คอสไม่ได้ดูแล

ขณะที่คอสไม่มีโปรแกรมฟอร์แมตฮาร์ดดิสก์ทางกายภาพ แต่สามารถใช้โปรแกรม IBM Advanced Diagnostics ในการฟอร์แมตแบบกายภาพได้ เพราะในการใช้งานจริงอาจเกิดเหตุการณ์ผิดปกติที่ทำให้ต้องฟอร์แมตดิสก์ใหม่

โปรแกรม FORMAT ของคอสใช้อำสั่งพิเศษของไบออส (BIOS) เพื่อฟอร์แมตดิสก์ทีละแท่ง กลไกของการฟอร์แมตทางกายภาพต้องการฟอร์แมตให้ทุกเซกเตอร์ในแต่ละแท่งใช้ระบบปฏิบัติการเดียวกัน เทคนิคการฟอร์แมตทีละแท่งสามารถนำมาใช้เป็นพื้นฐานสำหรับการป้องกันการก๊อปปี้ไฟล์ได้

การทำงานของโปรแกรม FORMAT กับดิสก์เกตต์ จะกำหนดให้ข้อมูลทุกไบต์ในแต่ละเซกเตอร์มีค่าเป็น F6 เนื่องจากคำสั่ง FORMAT จะเขียนค่า F6 ทับทุกไบต์ในดิสก์เกตต์ จึงทำให้ข้อมูลเก่าในดิสก์เกตต์ถูกลบไปทั้งหมด ดังนั้นเราจึงไม่สามารถกู้ข้อมูลเก่าคืนมาได้เมื่อมีการฟอร์แมตแผ่น อย่างไรก็ตามโปรแกรม FORMAT จะไม่เขียนทับข้อมูลเก่าลงในฮาร์ดดิสก์ ดังนั้นจึงสามารถกู้ข้อมูลเก่าคืนมาได้หลังจากฟอร์แมตฮาร์ดดิสก์ใหม่

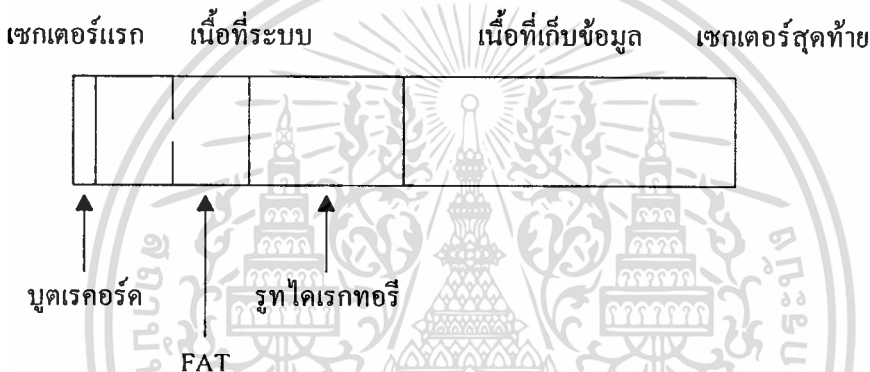
4) โครงสร้างของดิสก์ตามแบบของคอส

คอสแบ่งพื้นที่ในดิสก์ออกเป็น 2 ส่วน คือ พื้นที่สำหรับระบบที่คอสใช้เก็บข้อมูลหลักเกี่ยวกับดิสก์ และพื้นที่สำหรับจัดเก็บข้อมูลจริง พื้นที่สำหรับระบบจะเป็นเนื้อที่ส่วนเล็กๆ ประมาณ 0.3 เปอร์เซ็นต์ (สำหรับฮาร์ดดิสก์) ถึง 2 เปอร์เซ็นต์ (สำหรับฟลอปปีดิสก์) พื้นที่สำหรับระบบที่คอสใช้งานจะแบ่งออกเป็น 3 ส่วน คือ พื้นที่ของส่วนที่ทำการบูต, เนื้อที่สำหรับตารางกำหนดพื้นที่ของไฟล์ (File Allocation Table : FAT) และเนื้อที่ของไดเรกทอรี (Directory)

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของ บริษัท เทคโนโลยีสารสนเทศ จำกัด 4.1) บูตหรือบูตเรคอร์ด (Boot Record) เป็นส่วนแรกสุดของดิสก์ ส่วนนี้จะเป็นโปรแกรมการคำนวณพื้นที่ของดิสก์ และใช้เนื้อที่ในการเก็บเพียง 1 เซกเตอร์ บูตเรคอร์ดจะทำงานในช่วงเริ่มต้น

เพื่อจะนำคอสเข้าสู่หน่วยความจำของคอมพิวเตอร์ เรียกกระบวนการเริ่มต้นในการนำคอสเข้าสู่ระบบนี้ว่า การบูต (Booting) เวลาฟอร์แมตแผ่นดิสก์ ถ้าออกคำสั่งพร้อมด้วย /s การฟอร์แมตจะทำการโหลดคอสลงไปในแผ่นใหม่ด้วย เรียกว่า การฟอร์แมตระบบ (System Format)

สังเกตว่าบูตเรคอร์ดจะปรากฏอยู่ในแผ่นดิสก์ที่ฟอร์แมตแล้วทุกแผ่น แม้ว่าจะไม่ได้ใช้คำสั่ง /s ก็ตาม แต่สำหรับแผ่นที่ไม่ได้ฟอร์แมตด้วยคำสั่ง /s ก็จะไม่มีการคอส ซึ่งเป็นความสามารถของส่วนบูตเรคอร์ดที่จะตรวจสอบว่าแผ่นที่ใช้อยู่มีคอสหรือไม่ ถ้าหากไม่มีก็จะรายงานข้อผิดพลาดให้ทราบ ซึ่งก็ต้องเปลี่ยนเอาแผ่นที่ฟอร์แมตระบบไว้แล้วใส่เข้าไปแทน



รูปที่ 2.9 ส่วนประกอบของดิสก์ที่คอสจัดการ

4.2) ตารางที่กำหนดพื้นที่ของแฟ้มข้อมูล หรือเรียกว่า FAT หน้าที่ของ FAT คือใช้เป็นตารางสำหรับเก็บสถานะว่าเนื้อที่ในดิสก์ส่วนใดถูกใช้งานไปแล้วและเนื้อที่ส่วนใดยังว่างอยู่สำหรับจัดเก็บข้อมูลใหม่ลงไปได้ และเพื่อให้การใช้เนื้อที่ในดิสก์เป็นไปอย่างมีระบบ คอสได้แบ่งส่วนของดิสก์ออกเป็นส่วนๆ เรียกว่า คลัสเตอร์ (Cluster) โดยที่ขนาดของคลัสเตอร์จะแตกต่างกันไปตามชนิดของดิสก์ โดยจะมีขนาดเล็กที่สุดคือ 512 ไบต์ หรือ 0.5 k ในส่วนต่อไปนี้จะแสดงขนาดของคลัสเตอร์ในดิสก์เกิดแบบต่างๆ เอาไว้

ข้อมูลที่ปรากฏในแต่ละแถวของ FAT จะมีขนาด 12 บิตสำหรับดิสก์เกิด ถ้าข้อมูลใน FAT มีขนาดยาวขึ้น คอสก็สามารถเก็บจำนวนคลัสเตอร์ได้มากขึ้น แต่ไม่ว่าคลัสเตอร์จะมีขนาดใดก็ตาม คอสจะแบ่งส่วนของข้อมูลในดิสก์ออกเป็นคลัสเตอร์ย่อยๆ และใช้คลัสเตอร์นี้เป็นหน่วยที่กำหนดพื้นที่สำหรับไฟล์ โดยแต่ละคลัสเตอร์จะมีช่องประจำตัวอยู่ใน FAT และในช่องนี้จะบรรจุตัวเลขที่แสดงว่าคลัสเตอร์นั้นมีข้อมูลบรรจุอยู่หรือเป็นคลัสเตอร์ว่าง กล่าวคือถ้าค่าใน FAT ของคลัสเตอร์ใด

มีค่าเป็น 0 หมายความว่าคลัสเตอร์นั้นไม่ได้บรรจุข้อมูลใดๆ แต่ถ้าเป็นเลขอื่นหมายความว่ามิข้อมูล
บรรจุอยู่ในคลัสเตอร์นั้น และค่าตัวเลขนั้นจะระบุถึงคลัสเตอร์ตัวต่อไปที่บรรจุข้อมูลส่วนที่เหลือใน
ไฟล์นั้น จะเห็นว่าเริ่มต้นไฟล์ที่คลัสเตอร์หมายเลข 2 ซึ่งบรรจุข้อมูลหมายเลข 3 ไว้ หมายความว่า
ส่วนแรกของไฟล์จะบรรจุคลัสเตอร์หมายเลข 2 ส่วนถัดไปจะบรรจุคลัสเตอร์หมายเลข 3, 4, 5, 6
และ 7 ตามลำดับ สังเกตว่าคลัสเตอร์หมายเลข 7 จะบรรจุส่วนสุดท้ายของไฟล์ไว้ จึงไม่มีหมายเลข
คลัสเตอร์เชื่อมต่อ ในกรณีนี้คอสจะใส่รหัสพิเศษที่บอกจุดสิ้นสุดของไฟล์ไว้ในส่วนของคลัสเตอร์
7 ใน FAT

ตารางที่ 2.1 ขนาดของคลัสเตอร์

ดิสก์	ขนาดของคลัสเตอร์
3.5 นิ้ว, 1.44 MB	0.5 k
3.5 นิ้ว, 720 kB	1.0 k
5.25 นิ้ว, 1.2 MB	0.5 k
5.25 นิ้ว, 360 kB	1.0 k

ข้อมูลที่ปรากฏในแต่ละแถวของ FAT จะมีขนาด 12 บิตสำหรับดิสก์เกตต์ ถ้าข้อมูลใน FAT
มีขนาดยาวขึ้น คอสก็สามารถเก็บจำนวนคลัสเตอร์ได้มากขึ้น แต่ไม่ว่าคลัสเตอร์จะมีขนาดใดก็ตาม
คอสจะแบ่งส่วนของข้อมูลในดิสก์ออกเป็นคลัสเตอร์ย่อยๆ และใช้คลัสเตอร์นี้เป็นหน่วยที่กำหนด
พื้นที่สำหรับไฟล์ โดยแต่ละคลัสเตอร์จะมีช่องประจำตัวอยู่ใน FAT และในช่องนี้จะบรรจุตัวเลขที่
แสดงว่าคลัสเตอร์นั้นมีข้อมูลบรรจุอยู่หรือเป็นคลัสเตอร์ว่าง กล่าวคือถ้าค่าใน FAT ของคลัสเตอร์ใด
มีค่าเป็น 0 หมายความว่าคลัสเตอร์นั้นไม่ได้บรรจุข้อมูลใดๆ แต่ถ้าเป็นเลขอื่นหมายความว่ามิข้อมูล
บรรจุอยู่ในคลัสเตอร์นั้น และค่าตัวเลขนั้นจะระบุถึงคลัสเตอร์ตัวต่อไปที่บรรจุข้อมูลส่วนที่เหลือใน
ไฟล์นั้น จะเห็นว่าเริ่มต้นไฟล์ที่คลัสเตอร์หมายเลข 2 ซึ่งบรรจุข้อมูลหมายเลข 3 ไว้ หมายความว่า
ส่วนแรกของไฟล์จะบรรจุคลัสเตอร์หมายเลข 2 ส่วนถัดไปจะบรรจุคลัสเตอร์หมายเลข 3, 4, 5, 6
และ 7 ตามลำดับ สังเกตว่าคลัสเตอร์หมายเลข 7 จะบรรจุส่วนสุดท้ายของไฟล์ไว้ จึงไม่มีหมายเลข
คลัสเตอร์เชื่อมต่อ ในกรณีนี้คอสจะใส่รหัสพิเศษที่บอกจุดสิ้นสุดของไฟล์ไว้ในส่วนของคลัสเตอร์
7 ใน FAT

จากความสำคัญของ FAT คอสจึงเก็บก๊อปปี้สำรองของ FAT ไว้อีก 1 ก๊อปปี้ เพื่อให้สามารถ
นำมาใช้งานได้ในกรณีที่ดิสก์ส่วนที่เก็บ FAT ตัวจริงเกิดเสียหายขึ้น

4.3) รุทไดเรกทอรี (Root Directory) ซึ่งเป็นไดเรกทอรีของไฟล์ที่ดิสก์ทุกแผ่นต้องมี โดยที่ดิสก์จะมีไดเรกทอรีย่อย (Subdirectory) ลงไปอีกหรือไม่ก็ได้ แต่จำเป็นต้องมีรุทไดเรกทอรีเสมอ

ข้อมูลเกี่ยวกับไฟล์ทั้งหมดในดิสก์ถูกบันทึกไว้ในไดเรกทอรี ข้อมูลดังกล่าวประกอบด้วยชื่อไฟล์ 8 ตัวอักษร นามสกุล 3 ตัวอักษร ขนาดของไฟล์ และวันที่พร้อมทั้งเวลาที่บันทึกข้อมูลครั้งสุดท้าย ซึ่งข้อมูลทั้งหมดนี้สามารถเรียกดูได้ด้วยคำสั่ง DIR นอกจากนี้ยังมีข้อมูลอีก 2 ส่วน เกี่ยวกับไฟล์ที่เก็บไว้ในไดเรกทอรีส่วนแรก คือ หมายเลขของคลัสเตอร์แรกที่เก็บไฟล์นั้น อีกส่วนหนึ่ง คือคุณลักษณะพิเศษ ได้แก่ ไฟล์ระบบ (System File) หรือไฟล์ที่ซ่อนไว้ (Hidden File) และมีคุณสมบัติอีกบางประการที่สามารถระบุไว้ในไดเรกทอรีได้ นั่นคือ การระบุให้เป็นไฟล์ชนิดที่อ่านได้เพียงอย่างเดียว (Read Only File) เพื่อป้องกันไม่ให้มีการเปลี่ยนแปลงหรือลบไฟล์ทิ้งได้ เป็นต้น

ส่วนของรุทไดเรกทอรีจะเหมือนกับส่วนอื่นๆ ก็จะมีขนาดเฉพาะไปสำหรับการฟอร์แมตแต่ละแบบ ขนาดนี้จะเป็นตัวกำหนดว่ารุทไดเรกทอรีจะมีไฟล์ได้เป็นจำนวนกี่ไฟล์ โดยที่ข้อมูลสำหรับไฟล์ (หรือไดเรกทอรีย่อย) แต่ละชุดจะใช้เนื้อที่ 32 ไบต์ ดังนั้นจึงสามารถเก็บข้อมูลได้ 16 ชุดใน 1 เซกเตอร์ นั่นคือดิสก์ขนาด 3.5 นิ้ว (1.44 MB) ซึ่งใช้เนื้อที่ 14 เซกเตอร์สำหรับรุทไดเรกทอรีสามารถสร้างไฟล์ได้ 224 (มาจาก 16×14) ไฟล์ ถ้าเป็นฮาร์ดดิสก์จะสามารถมีได้มากกว่านั้น ซึ่งโดยส่วนใหญ่แล้วฮาร์ดดิสก์จะกันเนื้อที่ไว้ 32 เซกเตอร์ จึงมีไฟล์ได้ถึง 512 ไฟล์ (มาจาก 16×32) ในรุทไดเรกทอรี

ส่วนที่กินเนื้อที่ส่วนใหญ่ของดิสก์คือส่วนที่ใช้เก็บข้อมูลจริง พื้นที่ของข้อมูลจะแบ่งออกเป็นคลัสเตอร์ (ซึ่งมีขนาด 1 เซกเตอร์ หรือมากกว่า) สำหรับดิสก์ทุกแผ่นคลัสเตอร์จะมีขนาดเดียวกัน แต่ระหว่างดิสก์ที่ฟอร์แมตต่างกันคลัสเตอร์ก็จะมีขนาดต่างกันไป ข้อมูลในแต่ละไฟล์จะถูกบันทึกลงใน 1 คลัสเตอร์หรือมากกว่า (โดยที่ส่วนที่บอกว่าไฟล์ใดอยู่ในคลัสเตอร์ใดจะเก็บไว้ใน FAT) ข้อมูลในแต่ละไฟล์จะกระจายอยู่ตามคลัสเตอร์ที่ไม่ต่อเนื่องกัน ถึงแม้ว่าคอสมอสจะพยายามจัดเป็นไฟล์ให้อยู่ในคลัสเตอร์ที่ต่อเนื่องกันในตอนที่เริ่มต้นก็ตาม แต่เมื่อมีการแก้ไขข้อมูลในไฟล์ และเพิ่ม/ลบไฟล์ ข้อมูลจะเริ่มกระจายกระจายออกไป ซึ่งถ้าถึงสภาวะที่ข้อมูลในไฟล์เดียวกันอยู่กระจายกันมากเกินไปจะทำให้การอ่านไฟล์ช้าลงอย่างเห็นได้ชัด เรียกสภาวะการแตกกระจายนี้ว่าการแฟร็กเมนต์ (Fragment) ซึ่งสามารถใช้คำสั่ง SD (Speed Disk) ในนอร์ตันยูทิลิตี้แก้ไขปัญหาดังกล่าวกับฮาร์ดดิสก์ได้

การจัดการไดเรกทอรีย่อยมีหลักการคือ การเก็บไดเรกทอรีย่อยลงในดิสก์จะกระทำเหมือนกับไฟล์ทั่วไป คือ เก็บไว้ในเนื้อที่ส่วนข้อมูลของดิสก์ โดยที่ตำแหน่งคลัสเตอร์ของไดเรกทอรีย่อยนี้จะบันทึกอยู่ใน FAT เช่นเดียวกับไฟล์อื่นๆ แต่เมื่อคอสมอสเข้ามาใช้เนื้อที่ในส่วนไดเรกทอรีย่อย คอสมอสจะมองข้อมูลนี้เช่นเดียวกับรุทไดเรกทอรี (ก็จะมองทีละ 32 ไบต์ สำหรับชื่อและคุณลักษณะของไฟล์

ในไคเรกทอรีย่อนั้น) แต่ไคเรกทอรีย่อยและรูทไคเรกทอรีมีข้อแตกต่างกัน คือ ในแต่ละระบบจะมีรูทได้ตัวเดียว แต่มีไคเรกทอรีย่อยได้หลายตัว นอกจากนี้แล้ว รูทยังมีขนาดและความจุคงที่ ในขณะที่ไคเรกทอรีย่อยจะเหมือนกับไฟล์ทั่วไปที่มีขนาดได้ต่างๆ กัน

5) รูปแบบของไฟล์

ไฟล์ที่เก็บอยู่ในดิสก์ทั่วไปจะมีรูปแบบเฉพาะ ได้แก่ ลักษณะโครงสร้างของข้อมูลในไฟล์นั้น ส่วนนามสกุล 3 ตัวของไฟล์สามารถใช้เป็นตัวกำหนดรูปแบบ และการใช้เพิ่มข้อมูล โดยที่นามสกุลบางอย่างก็ถือเป็นมาตรฐานในการกำหนดชนิดของแฟ้มข้อมูลโดยตรง ในขณะที่นามสกุลบางอย่างก็ไม่มีการใช้ที่กำหนดตายตัว

ส่วนใหญ่นามสกุลที่มีการกำหนดตายตัวจะขึ้นอยู่กับซอฟต์แวร์ที่ใช้งานด้วย เช่น คอสจะกำหนดไว้ว่า ไฟล์ทำงานของคอส (Executable) จะต้องลงท้ายด้วย .COM หรือ .EXE ถ้าเป็น แบตช์ไฟล์ (Batch File) ใช้นามสกุล .BAT ส่วนนามสกุลของไฟล์ชนิดอื่นๆ นั้นไม่มีการกำหนดมาตรฐานแน่นอน แต่การตั้งชื่อนามสกุลตามความนิยมจะทำให้การใช้งานเป็นระบบมากขึ้น เช่น การกำหนดให้โปรแกรมเบสิกลงท้ายด้วย .BAS โปรแกรมภาษาปาสคาลลงท้ายด้วย .PAS เป็นต้น

นอกจากนี้ยังสามารถเข้าไปดูข้อมูลแท็ก ในไฟล์ได้ด้วยเครื่องมือประเภทด๊ิบก หรือ NU ประโยชน์ที่ได้จากการเข้าไปดูข้อมูลที่เห็นได้ชัด ได้แก่ การกู้ไฟล์ที่ลบทิ้งไป หรือการซ่อมแซมข้อมูลโดยตรงเข้าไปในดิสก์

โดยทั่วไป ไฟล์ที่ใช้เก็บข้อมูลจะมีโครงสร้างภายในเฉพาะตัวที่ไม่สามารถดูเข้าใจได้ ยกเว้นส่วนที่เป็นข้อความ เช่น ชื่อและที่อยู่ปรากฏในไฟล์ของฐานข้อมูล เป็นต้น แต่ส่วนประกอบที่ใช้เก็บฟอร์แมตโครงสร้างของไฟล์ ผู้ใช้ทั่วไปไม่สามารถเข้าใจได้

6) ไฟล์ข้อความรหัสแอสกี

ไฟล์ข้อความแอสกีเป็นรูปแบบของไฟล์อีกแบบหนึ่งซึ่งเป็นชนิดของไฟล์ที่ใช้กันมากที่สุด หรือเรียกว่า ไฟล์แอสกี (ASCII file) ขณะที่ซอฟต์แวร์ส่วนใหญ่มีวิธีการบันทึกข้อมูลแบบพิเศษเฉพาะตัว ไฟล์แอสกีเป็นแบบที่ใช้กันทั่วไปในโปรแกรมใดๆ ก็ได้ ไฟล์แอสกีได้รับการออกแบบให้เก็บข้อความธรรมดาเหมือนกับข้อความในหนังสือ ซอฟต์แวร์ส่วนใหญ่ เช่น เวิร์ดโปรเซสเซอร์ (Wordprocessor) สเปรดชีต (Spreadsheet) ระบบจัดการฐานข้อมูล ฯลฯ จะสร้างไฟล์ที่มีรูปแบบเฉพาะ ซึ่งประกอบด้วยส่วนที่เป็นข้อมูล และโครงสร้างที่ใช้ในการควบคุมรูปแบบของข้อมูล ดังนั้นเมื่อมีความจำเป็นที่ต้องมีการโอนย้ายข้อมูลระหว่างซอฟต์แวร์ ก็สามารถใช้ไฟล์แอสกีเป็นตัวกลางได้ กล่าวคือ ในระบบซอฟต์แวร์จะมีเครื่องมือที่ใช้เปลี่ยนโครงสร้างเป็นไฟล์แอสกี โดยตัดเอาส่วนควบคุมที่ยุ่งยากออก เหลือเฉพาะข้อความธรรมดา เมื่อทำการโหลดข้อมูลจากไฟล์แอสกีเข้าสู่ระบบก็จะใช้เครื่องมืออีกตัวหนึ่ง เป็นต้น

ลักษณะซอฟต์แวร์ที่ทำงานกับไฟล์แอสกีล้วนๆ เช่น ตัวแปรภาษาต่างๆ จะทำงานร่วมกับไฟล์ที่บรรจุโปรแกรมเอาไว้ ซึ่งก็เป็นไฟล์แอสกี ตัวอย่างที่เห็นบ่อยๆ คือ แบตช์ไฟล์ เป็นต้น

ข้อมูลในไฟล์ข้อความหรือไฟล์แอสกี ประกอบด้วยอักขระ 2 แบบคือ อักขระแอสกีที่เป็นตัวพยัญชนะธรรมดา และตัวอักษรควบคุม อักขระที่เป็นข้อความธรรมดาเป็นข้อมูลพื้นฐานในไฟล์แอสกี ในขณะที่ตัวอักษรควบคุมจะเป็นตัวบอกว่าแฟ้มข้อมูลมีรูปแบบอย่างไร เช่น ใช้ในการกำหนดการแบ่งบรรทัดและย่อหน้า เป็นต้น

ปัจจุบันไม่มีกฎหมายตายตัวเกี่ยวกับวิธีที่ซอฟต์แวร์ต่างๆ ใช้ไฟล์แอสกี โดยเฉพาะส่วนที่เกี่ยวกับอักขระควบคุม นั่นคือ การตีความหมายของอักขระควบคุมซอฟต์แวร์แต่ละตัวจะแตกต่างกันไป ในที่นี้จะกล่าวถึงเฉพาะอักขระควบคุมที่ใช้กันมาก และความหมายเป็นที่ยอมรับกันทั่วไป

1. ตัวอักษรควบคุมที่กำหนดจุดจบของบรรทัด คือ CR (Carriage Return) กับ LF (Line Feed) ได้แก่ รหัส 13 (หรือ 0D ในฐานสิบหก) และ 10 (หรือ 0A ในฐานสิบหก) ตามลำดับ

2. ตัวอักษรควบคุมที่กำหนดจุดสิ้นสุดของไฟล์ คือ Ctrl-Z ได้แก่ รหัส 26 (หรือ 1A ในฐานสิบหก) ตารางแอสกีส่วนใหญ่จะเรียกตัวอักษรตัวนี้ว่า SUB แต่เนื่องจากความหมายในการใช้งาน คือการกำหนดจุดสิ้นสุดของไฟล์ ดังนั้นจึงมักเรียกดัชนีตัวนี้ว่า EOF (End Of File) แทน

3. TAB ใช้แทนการเว้นช่องว่างหลายๆ ตัว TAB มีรหัสหมายเลข 9 หรือในตารางจะเรียกว่า HT ซึ่งย่อมาจาก Horizontal Tab ส่วนใหญ่แล้ว TAB จะมีความยาว 8 ตัวอักษร (ช่องว่าง) Form Feed ใช้บอกจุดจบของหน้า และให้เริ่มหน้าใหม่ คือรหัสหมายเลข 12 (หรือ 0C ในฐานสิบหก) ซึ่งในตารางแอสกีเรียกตัวอักษรนี้ว่า FF

วิธีการแบ่งข้อความเป็นย่อหน้าที่ใช้ส่วนใหญ่มี 2 วิธี วิธีแรกจะกำหนดจุดจบของแต่ละบรรทัดด้วย CR และ LF และเมื่อจบย่อหน้าจะใส่บรรทัดว่างไว้ 1 บรรทัด กล่าวคือ จะมี CR และ LF 2 ชุดอยู่ติดต่อกัน ตัวอย่าง สมมติให้มีไฟล์ที่มีความยาว 2 บรรทัด ดังนี้

Columbus sailed the ocean blue

In fourteen hundred and ninety two.

ข้อมูลที่เก็บในไฟล์ จะประกอบด้วยอักขระควบคุมด้วยดังนี้

Columbus sailed the ocean blue <CR> <LF>

In fourteen hundred and ninety two. <CR> <LF> <EOF>

เอกสารนี้ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิธีที่ 2 จะไม่มีอักขระควบคุมระหว่างบรรทัด กล่าวคือ ในแต่ละย่อหน้าจะมีเฉพาะข้อความเรียงต่อกัน และจบย่อหน้าด้วย CR และ LF ในรูปแบบที่ 2 นี้เป็นหน้าที่ของซอฟต์แวร์ที่ใช้ไฟล์นั้นอยู่ว่าจะต้องทำการควบคุมการขึ้นบรรทัดใหม่เอง เช่น อาจจะใช้วิธีการนับตัวอักษร เป็นต้น จากตัวอย่างรูปแบบทั้งสองทำให้ทราบความยุ่งยากซึ่งเกิดขึ้นในการโอนย้ายข้อมูลระหว่างซอฟต์แวร์ได้ เพราะแม้ว่าจะเป็นการใช้ไฟล์แอสกีเหมือนกัน แต่รูปแบบในการจัดย่อหน้าก็อาจจะแตกต่างกัน อย่างไรก็ตามไฟล์แอสกีถือเป็นตัวกลางที่ดีที่สุดในขณะนี้ หากผู้ใช้งานมีความรู้พื้นฐานเกี่ยวกับรูปแบบการใช้อักขระควบคุมในไฟล์แอสกีของซอฟต์แวร์ที่กำลังใช้งานอยู่ ก็พอที่จะแก้ไขปัญหาที่อาจเกิดขึ้นได้โดยไม่ยากนัก อีกสิ่งหนึ่งที่จะกล่าวถึงคือ การใช้อักขระควบคุมจุดจบของไฟล์ หรือ EOF หรือ Ctrl-Z ก็ยังไม่มีความเป็นมาตรฐานเดียวกัน ตัวอย่างความแตกต่างกันรูปแบบแรกเป็นดังเช่นตัวอย่างที่เราแสดงไปแล้วคือ <EOF> จะถูกใส่ไว้ในตอนจุดจบของไฟล์ ในกรณีนี้ขนาดของไฟล์ที่ปรากฏในไดเรกทอรีจะเท่ากับขนาดไฟล์จริงซึ่งประกอบด้วยส่วนของข้อความและอักขระควบคุม ซึ่งนับคู่สำหรับตัวอย่างนี้จะได้เท่ากับ 70 ไบต์ แต่ก็มีซอฟต์แวร์บางตัวที่เก็บข้อมูลที่ละกลุ่ม เช่น เก็บกลุ่มละ 128 ไบต์ ในลักษณะแบบนี้ EOF ก็จะอยู่ที่จุดจบของข้อความจริงๆ ดังแสดงในตัวอย่าง ในขณะที่ขนาดของไฟล์ที่ปรากฏอยู่ในไดเรกทอรีจะเท่ากับ 128 ไบต์ ส่วนอีกรูปแบบหนึ่งที่อาจพบคือ ซอฟต์แวร์จะไม่บันทึก EOF เอาไว้ แต่จะสามารถทราบได้ถึงจุดสิ้นสุดของไฟล์ได้โดยการตรวจสอบจากขนาดของไฟล์ในไดเรกทอรี ซึ่งในกรณีนี้ขนาดของไฟล์ที่บันทึกเอาไว้จะต้องเป็นขนาดของไฟล์จริงๆ (ที่ไม่นับ EOF ด้วย)

2.2.9 เจาะลึกรายละเอียดของดิสก์

1) รายละเอียดโครงสร้างของดิสก์

ในหัวข้อนี้จะเจาะลึกเกี่ยวกับวิธีที่คอยจัดโครงสร้างของดิสก์ เนื้อหาเหล่านี้จะช่วยให้ทราบถึงการใช้ดิสก์ รวมทั้งช่วยให้สามารถแก้ไขปัญหาเมื่อเกิดข้อผิดพลาดขึ้นได้

คอสแบ่งดิสก์เป็น 2 ส่วน ได้แก่ ส่วนของระบบ และส่วนที่ใช้เก็บข้อมูล ส่วนของระบบมี 3 ส่วน ได้แก่ บูตเรคคอร์ด, FAT และรูทไดเรกทอรี

1.1) บูตเรคคอร์ด เป็นส่วนแรกของดิสก์ ใช้สำหรับเก็บโปรแกรมสั้นๆ ที่ใช้เริ่มต้นกระบวนการเริ่มการทำงานของคอส (การบูต) บูตเรคคอร์ดจะมีในดิสก์ทุกแผ่น แม้ว่าแผ่นนั้นจะใช้บูตไม่ได้ก็ตาม โปรแกรมการบูตมีขนาดเล็กที่สามารถใส่ลงในเซกเตอร์เดียวได้ แต่ถึงแม้ว่าโปรแกรมบูตจะมีขนาดใหญ่ขึ้น คอสก็ออกแบบมาให้รองรับการขยายเนื้อที่ที่จะใช้เก็บบูตเรคคอร์ดได้

ข้อมูลที่อยู่ในบูตเรคคอร์ดสามารถใช้โปรแกรมดีบักเพื่อตรวจสอบเนื้อหาในบูตเรคคอร์ดอย่างง่ายๆ โดยการออกคำสั่ง 2 คำสั่ง คือ L0001 ซึ่งจะทำให้การอ่านบูตเรคคอร์ดจากดิสก์ไดรฟ์ A ลงบน

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน่วยความจำ และคำสั่ง D 0 L 200 จะทำการแสดงข้อมูลบูตเรคคอร์ดในรูปแบบเลขฐาน 16 และรหัสแอสกี ตารางที่ 2.2 แสดงข้อมูลข้างต้นของคอสเวอร์ชัน 4.0

```

0000 EB 3C 90 4D 53 57 49 4E 34 2E 31 00 02 01 01 00 <.MSWIN4.1.....
0010 02 E0 00 40 0B F0 09 00 12 00 02 00 00 00 00 00 ...@.....
0020 40 0B 00 00 00 00 29 CF 0A 10 15 46 44 43 5F 50 @.....)....FDC_P
0030 52 4F 4A 45 43 54 46 41 54 31 32 20 20 20 20 FA 33 ROJECTFAT12 .3
0040 C9 8E D1 BC FC 7B 16 07 BD 78 00 C5 76 00 1E 56 .....{...x..v..V
0050 16 55 BF 22 05 89 7E 00 89 4E 02 B1 0B FC F3 A4 .U."...~..N.....
0060 06 1F BD 00 7C C6 45 FE 0F 8B 46 18 88 45 F9 38 ....|.E...F..E.8
0070 4E 24 7D 22 8B C1 99 E8 77 01 72 1A 83 EB 3A 66 N$)"...w.r....:f
0080 A1 1C 7C 66 3B 07 8A 57 FC 75 06 80 CA 02 88 56 ..|f;..W.u.....V
0090 02 80 C3 10 73 ED 33 C9 8A 46 10 98 F7 66 16 03 ....s.3..F...f..
00A0 46 1C 13 56 1E 03 46 0E 13 D1 8B 76 11 60 89 46 F..V..F....v..`F
00B0 FC 89 56 FE B8 20 00 F7 E6 8B 5E 0B 03 C3 48 F7 ..V.. ....^...H.
00C0 F3 01 46 FC 11 4E FE 61 BF 00 07 E8 23 01 72 39 ..F..N.a....#.r9
00D0 38 2D 74 17 60 B1 0B BE D8 7D F3 A6 61 74 39 4E 8-t..}....}.at9N
00E0 74 09 83 C7 20 3B FB 72 E7 EB DD BE 7F 7D AC 98 t...;r.....)..
00F0 03 F0 AC 84 C0 74 17 3C FF 74 09 B4 0E BB 07 00 .....t.<.t.....

```

รูปที่ 2.10 บูตเรคคอร์ด

จากรูป ข้อความที่บอกว่าไม่สามารถบูตระบบได้พร้อมด้วยชื่อไฟล์ คือ IBMBIO.COM และ IBMDOS.COM เพราะโปรแกรมบูตจะทำหน้าที่ตรวจหาไฟล์ทั้งสองนี้จากไดเรกทอรีว่ามีหรือไม่ ซึ่งไม่มีก็จะแสดงข้อความดังที่เห็นออกมา ในส่วนต้นของโปรแกรมข้อมูลจะบอกเวอร์ชันของคอส ได้แก่ “IBM 4.0” เอาไว้ นอกจากนี้ก็มีข้อมูลที่บ่งบอกคุณลักษณะของดิสก์ เช่น จำนวนไบต์ต่อเซกเตอร์ จำนวนเซกเตอร์ต่อแทร็ก เป็นต้น รวมทั้งขนาดของ FAT และไดเรกทอรี ในตอนจบของบูตเรคคอร์ดลงท้ายด้วยข้อมูล 55 AA (ฐาน 16) สำหรับดิสก์ที่ถูกฟอร์แมตด้วยคอสรุ่นที่ใหม่กว่า 2.0

ในการทำงานคอสจำเป็นต้องรับรู้และควบคุมลักษณะของดิสก์ทั้งหมด ซึ่งไม่ค่อยมีปัญหาสำหรับคอสเวอร์ชันแรกๆ เพราะมีการฟอร์แมตดิสก์เพียงไม่กี่รูปแบบ จึงสามารถเก็บคุณลักษณะต่างๆ เหล่านี้ไว้ในตัวคอสได้เลย และคอสก็อ่านรหัส 1 ไบต์ในดิสก์ เพื่อที่จะใช้เลือกเอาข้อมูลดังกล่าวออกจากคอส หลังจากนั้นจึงมีการดัดแปลงให้เก็บคุณลักษณะของดิสก์ไว้ในบูตเรคคอร์ดแทน ซึ่งสามารถเรียกเอาข้อมูลเหล่านี้ออกมาดูได้ ดังแสดงในรูปที่ 2.10

โปรแกรมบูตจะทำการบูตระบบด้วยการอ่านบูตเรคคอร์ด ดังนั้นผู้ที่ต้องการศึกษาการทำงานของโปรแกรมนี้อาจสามารถดักเรียกดูได้โดยออกคำสั่ง U (สำหรับการ Unassembler ได้แก่ การแปลงภาษาเครื่องกลับเป็นภาษาแอสเซมบลี) แล้วตรวจจากโปรแกรมแอสเซมบลีว่าโปรแกรมบูตมีการออกคำสั่งให้กระโดดไปอ่านข้อมูลที่ใด (ซึ่งเป็นข้อมูลของบูตเรคคอร์ด) แล้วจึงออกคำสั่ง U ใน

ตำแหน่งนั้นอีก ดังเช่น ข้อมูลในบูตเรคคอร์ดที่ปรากฏในรูปที่ 2.10 ได้มาจากการออกคำสั่ง 2 คำสั่ง คือ U 0 L 2 และ U 3 E

1.2) FAT ทำหน้าที่ควบคุมการใช้พื้นที่แฟ้มข้อมูล พื้นที่ในดิสก์จะถูกแบ่งเป็นคลัสเตอร์ ซึ่งขนาดของคลัสเตอร์จะแตกต่างกันตามชนิดและการฟอร์แมตของดิสก์ โดยที่ทุกคลัสเตอร์จะมีเลขประจำตัวเรียงตามลำดับ จาก 2 สำหรับคลัสเตอร์แรก (คลัสเตอร์ 0 กับ 1 เก็บไว้สำหรับคอส) และข้อมูลพร้อมกับสถานะของคลัสเตอร์ก็จะเก็บไว้ใน FAT ดังนี้

สมัยก่อนมีการกำหนดเนื้อที่ 12 บิตสำหรับเก็บค่าแต่ละช่องของ FAT จากความใหญ่ของ 12 บิต ทำให้มีค่าที่แตกต่างกันได้ 4 k หรือประมาณ 4000 ค่า จึงทำให้จำนวนคลัสเตอร์ถูกจำกัดไว้ที่ 4000 กว่าคลัสเตอร์ แต่เมื่อดิสก์เริ่มมีขนาดใหญ่ขึ้นจึงได้มีการดัดแปลงให้ขนาดของช่องใน FAT ใหญ่ขึ้นเป็น 16 บิต ซึ่งจะทำให้มีคลัสเตอร์ได้กว่าหมื่นคลัสเตอร์ ไม่ว่าขนาดของ FAT จะระบุให้ใช้แบบใด (ระหว่าง 12 บิตกับ 16 บิต) แต่โครงสร้างวิธีการดูแลจัดการข้อมูลของ FAT ก็ยังเหมือนกัน

ถ้าข้อมูลของ FAT ในช่องมีค่าเป็น 0 แสดงว่าคลัสเตอร์นั้นยังเป็นคลัสเตอร์ว่าง ซึ่งสามารถนำไปใช้เก็บข้อมูลได้ แต่ถ้าคลัสเตอร์ใดเก็บข้อมูลอยู่แล้ว ค่าใน FAT สำหรับคลัสเตอร์นั้นจะมีได้ 2 ประเภท คือ อาจจะเก็บหมายเลขคลัสเตอร์ที่เก็บข้อมูลส่วนต่อไปของไฟล์นั้น หรือเก็บรหัสพิเศษ (อันได้แก่ FFF สำหรับ 12 บิต หรือ FFFF สำหรับ 16 บิต) ที่แสดงว่าคลัสเตอร์นั้นเก็บข้อมูลส่วนสุดท้ายของไฟล์ก็ได้ แต่ละไฟล์จะเก็บหมายเลขของคลัสเตอร์แรกไว้ในไคเรกทอรี นอกจากนี้หากดิสก์ส่วนใดเกิดความบกพร่อง หรือที่เรียกว่ามีส่วนของแทร็กเสีย (Bad Track) อยู่ ก็จะมีค่าข้อมูลใน FAT สำหรับคลัสเตอร์นั้นเป็น FF7 (สำหรับ FAT ขนาด 12 บิต) หรือ FFF7 (สำหรับ FAT ขนาด 16 บิต) และมีการจองรหัส FF0 ถึง FFF (หรือ FFF0 ถึง FFFF สำหรับ FAT ขนาด 16 บิต) ไว้สำหรับการใช้งานในอนาคต

ถึงแม้ว่าไฟล์ทั่วไปจะเริ่มใช้คลัสเตอร์ตั้งแต่หมายเลข 2 ขึ้นไป แต่ในตาราง FAT จะเริ่มเก็บข้อมูลตั้งแต่ช่องแรก (คลัสเตอร์ 0) โดยคลัสเตอร์ 0 และ 1 จะใช้เป็นช่องเก็บข้อมูลทางเทคนิคของดิสก์ ตัวอย่างเช่น ไบต์แรกของ FAT (คลัสเตอร์ 0) จะเก็บรหัสที่ระบุวิธีการฟอร์แมตดิสก์ เช่น ถ้าตัวรหัสเท่ากับ F8 หมายถึงฮาร์ดดิสก์ เป็นต้น และเนื่องจากความสำคัญของ FAT จึงจำเป็นต้องมี FAT ชุดสำรองเก็บไว้ในดิสก์อีกชุดหนึ่งเพื่อกันไว้ในกรณีที่ FAT ส่วนแรกเกิดเสียหายไป

1.3) รูทไคเรกทอรี ซึ่งเก็บต่อจาก FAT ไคเรกทอรีทำงานเหมือนกับตารางข้อมูลซึ่งมีขนาดของแต่ละช่องเท่ากับ 32 ไบต์ สำหรับรายละเอียดเกี่ยวกับไฟล์และไคเรกทอรีย่อยต่างๆ

ข้อมูลในไคเรกทอรีประกอบด้วย ชื่อไฟล์ขนาด 8 ไบต์ นามสกุล 3 ไบต์ ขนาดของไฟล์ วัน และเวลาที่สร้างไฟล์ หมายเลขคลัสเตอร์เริ่มต้น และรหัสที่ใช้ระบุคุณลักษณะของไฟล์ ซึ่งใช้เนื้อที่ 22 ไบต์ โดยเหลือสำรองไว้ 10 ไบต์สำหรับการใช้งานที่อาจเพิ่มเติมเข้ามาในอนาคต มีสิ่งที่น่าสนใจ

ในไดเรกทอรี คือ ถ้าไบต์แรกที่ใช้เก็บชื่อไฟล์มีค่าเป็น 0 แสดงว่าไดเรกทอรีช่องนั้นยังไม่ได้ใช้เก็บอะไร แต่ถ้ามีค่า E5 หมายความว่าไฟล์นั้นได้ถูกลบทิ้งไปแล้ว (แต่ที่จริงข้อมูลในไฟล์ยังคงอยู่) ดังนั้นในการทำงานของโปรแกรมที่ใช้กู้ไฟล์จะทำการสำรวจไดเรกทอรีสำหรับช่องที่ชื่อไฟล์ขึ้นต้นด้วย E5 แต่สังเกตว่า E5 จะเข้ามาทับตัวอักษรตัวแรกของชื่อไฟล์ โดยที่เหลือของชื่อยังเหมือนเดิม

โดยปกติแล้วสำหรับไฟล์และไดเรกทอรีย่อยจะแยกชื่อและนามสกุลออกเป็นคนละฟิลด์ แต่เมื่อใช้กับชื่อของดิสก์ (Volume Label) จะรวมเป็นฟิลด์เดียวที่มีขนาด 11 ตัวอักษร นั่นคือ เมื่อเราออกคำสั่งให้ดิสก์ (ด้วยคำสั่ง DIR หรือ CHKDSK) จะเห็นว่าชื่อดิสก์ไม่แยกออกจากกันด้วยจุด เหมือนกับชื่อของไฟล์ปกติ

เนื้อที่ขนาด 4 ไบต์ถูกใช้สำหรับเก็บขนาดของไฟล์ ซึ่งมีความกว้างพอที่จะสร้างตัวเลขที่มีขนาดใหญ่ ทำให้แน่ใจได้ว่าไฟล์จะไม่ถูกจำกัดขนาดด้วยขนาดที่สามารถบันทึกลงในไดเรกทอรี สังเกตว่า จะเก็บขนาดของไฟล์เฉพาะกับไฟล์ปกติเท่านั้น โดยที่ช่องที่ใช้เก็บชื่อดิสก์หรือไดเรกทอรีย่อยจะมีค่าขนาดของไฟล์เป็น 0 ซึ่งขัดกับความเป็นจริง เพราะไดเรกทอรีย่อยมีการเก็บข้อมูลจริงๆ จึงมีขนาดไม่ใช่ 0 แต่จะไม่ทำการบันทึกขนาดไว้ในตารางไดเรกทอรี

สำหรับวันและเวลาที่ข้อมูลในไดเรกทอรีแต่ละชุดกำหนดให้บันทึกตั้งแต่วันที่ 1 มกราคม 1980 จนถึงสิ้นปี 2099 เวลาที่บันทึกจะมีวามคลาดเคลื่อนจากเวลาจริงไม่เกิน 2 วินาที แต่อาจจะแสดงเวลาออกมาเป็นหลักชั่วโมงและนาทีเท่านั้น คอสใช้เนื้อที่ 16 บิตในการบันทึกวันที่ และอีก 16 บิตสำหรับเวลา โดยจะจัดข้อมูลทั้งสองนี้อยู่ติดกันเป็น 4 ไบต์และใช้สูตรเฉพาะในการแปลงค่าออกมาเป็นวันเวลาดังนี้

$$\text{วันที่} = \text{วัน} + 32 \times \text{เดือน} + 512 \times (\text{ปี}-1980)$$

$$\text{เวลา} = \text{วินาที}/2 + 32 \times \text{นาฬิกา} + 1024 \times \text{ชั่วโมง}$$

ข้อมูล 1 ไบต์ที่ใช้เก็บคุณลักษณะของไฟล์สามารถใช้กำหนดคุณลักษณะได้ 8 อย่าง (เพราะ 1 ไบต์มี 8 บิต) กล่าวคือถ้าบิตใดถูกเซตไว้แสดงว่าไฟล์นั้นมีคุณลักษณะตามที่ระบุด้วยบิตนั้น โดยที่ในขณะที่ใช้ไปแล้ว 6 บิต และเหลือไว้อีก 2 บิตสำหรับใช้ในอนาคต การใช้งานทั้ง 6 บิตนี้ เป็นดังตารางที่ 2.2

ไฟล์ที่ซ่อนไว้ในไฟล์ระบบ คือ คำสั่งคอสบางชนิด เช่น DIR, COPY และ DEL จะไม่รู้จักรหัสไฟล์ 2 ชนิดนี้ นั่นคือ ถ้ากำหนดให้ไฟล์ใดมีชนิดเป็นไฟล์ระบบแล้วออกคำสั่ง DIR ก็จะไม่ปรากฏชื่อไฟล์นั้นออกมา แต่บางคำสั่ง เช่น TYPE จะรู้จักไฟล์ทั้งสองชนิดนี้ อาจจะต้องออกคำสั่ง TYPE การค้า IBM DOS.COM จากแผ่นระบบก็ได้

ตารางที่ 2.2 คุณลักษณะของไฟล์ที่ระบุด้วยบิต

บิต	คุณลักษณะ
1	เป็นชื่อของดิสก์ (Disk Volume Label)
2	เป็นไคเรกทอรีย่อย
3	เป็นไฟล์ชนิดอ่านอย่างเดียว
4	เป็นไฟล์ที่ได้มีการแก้ไขไปแล้ว
5	เป็นไฟล์ที่ซ่อนไว้ (Hidden File)
6	เป็นไฟล์ระบบ (System File)
7-8	ยังไม่ได้ใช้

Menu 3.2 Technical information

Drive A:

Type of disk:

Double – sided,quad-density, 18-sector,high-capacity

Basic storage capacity:

1.4 Million characters (megabytes)

100%of disk space is free

Logical dimension:

Sectors are made up of 512 bytes

Tracks are made of 18 sectors per disk side

There are 2 disk sides

The disk is made up of 80 tracks

Space for files is allocated in clusters of 1 sector

There are 20847 clusters

The disk's root directory can hold 244 files

Press any key to continue...

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
รูปที่ 2.11 ขนาดของแต่ละส่วนในดิสก์
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รุตไคเรททอรีจะเหมือนกับส่วนอื่นๆ ในระบบ ก็จะมีขนาดคงที่ ดังนั้นคอสจึงสามารถรู้ได้ว่า จะพบจุดเริ่มต้นของไคเรททอรีที่ใด และจุดเริ่มต้นข้อมูล (เป็นส่วนที่อยู่ต่อจากรุตไคเรททอรี) อยู่ที่ใด และเนื่องจากขนาดที่คงที่นี้ จึงทำให้รุตไคเรททอรีมีจำนวนไฟล์ที่จำกัด ซึ่งตรงข้ามกับไคเรททอรีย่อย เพราะเราเก็บไคเรททอรีย่อยไว้ในส่วนของข้อมูล จึงมีสถานะเหมือนไฟล์ทั่วไป ทำให้ไม่มีความจำกัดในเรื่องของขนาด ดังนั้นถ้าจำเป็นต้องเก็บไฟล์เป็นจำนวนมาก ก็จะต้องสร้างไคเรททอรีย่อยขึ้นเพื่อหลีกเลี่ยงปัญหาดังกล่าว

แม้ว่าส่วนของระบบดิสก์จะมีขนาดคงที่ แต่ขนาดนั้นก็ขึ้นอยู่กับชนิดและการฟอร์แมตดิสก์ ดังนั้นถ้าอยากทราบว่าดิสก์ที่ใช้อยู่มีเนื้อที่ในแต่ละส่วนเป็นขนาดเท่าใดก็สามารถใช้โปรแกรม NU ตรวจสอบได้ เช่น ตัวอย่างที่แสดงในตารางที่ 2.4 เป็นข้อมูลของดิสก์เกดต์ 3.5 นิ้ว ความจุ 1.44 MB

ส่วนของดิสก์ซึ่งมีขนาดใหญ่ที่สุดคือ ส่วนที่ใช้เก็บข้อมูล เนื่องจากไฟล์มีได้หลายขนาด แต่จะจัดเก็บในแบบเต็มเซกเตอร์ 512 ไบต์ และเซกเตอร์ก็ประกอบขึ้นเป็นคลัสเตอร์ ดังนั้นจึงมีการเหลือเนื้อที่ว่างตอนท้ายของแต่ละเซกเตอร์ และมีเซกเตอร์ว่างในตอนท้ายของคลัสเตอร์ ส่วนที่ว่างเปล่าเหล่านี้แท้จริงแล้วยังมีข้อมูลของไฟล์เก่าค้างอยู่ในกรณีที่ไฟล์ใหม่ใช้เนื้อที่ไม่หมด

2.3 วงจรควบคุมดิสก์ไครฟ์

วงจรควบคุมดิสก์ไครฟ์เป็นอุปกรณ์ที่ทำหน้าที่เชื่อมต่อระหว่างไมโครคอมพิวเตอร์และดิสก์ไครฟ์ โดยมีหน้าที่หลักคือ การอินเตอร์เฟสกับไมโครคอมพิวเตอร์ โดยให้ไมโครคอมพิวเตอร์มองอุปกรณ์ดิสก์ไครฟ์เป็นพอร์ตที่สามารถควบคุมการทำงาน และบันทึกข้อมูลโดยผ่านพอร์ต และนำข้อมูลที่ได้จากไมโครคอมพิวเตอร์มาทำการมอดูเลต และส่งข้อมูลแบบซีเควนเซียลไปให้เกดต์ไครฟ์ นอกจากนี้ยังมีวงจรควบคุมการทำงานดิสก์ไครฟ์ในการอ่าน บันทึกข้อมูล และตรวจสอบความผิดพลาดข้อมูลที่เกิดขึ้นเพื่อส่งให้แก่ไมโครคอมพิวเตอร์

2.3.1 โครงสร้างของวงจรควบคุมดิสก์ไครฟ์

โครงสร้างของวงจรควบคุมดิสก์ไครฟ์ (Disk Controller) ที่ใช้ในไมโครคอมพิวเตอร์ 16 บิต แสดงดังรูปที่ 2.10 อธิบายได้ดังนี้

1) บัฟเฟอร์ (Buffer) เป็นอุปกรณ์ที่คั่นระหว่างบัสข้อมูลของไมโครคอมพิวเตอร์กับบัสข้อมูลของวงจรควบคุมดิสก์ไครฟ์

2) ตัวควบคุมฟลอปปีดิสก์ หรือ FDC เป็นส่วนสำคัญที่สุดในวงจร ซึ่งทำหน้าที่รับคำสั่งจากไมโครคอมพิวเตอร์ในการควบคุมดิสก์ไครฟ์ และนำข้อมูลที่ต้องการบันทึกมามอดูเลต แล้วส่งให้กับดิสก์ไครฟ์ หรือนำข้อมูลที่อ่านจากดิสก์ไครฟ์มาทำการดีมอดูเลต เพื่อส่งให้กับไมโครคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) พอร์ตควบคุม (Digital Control Port) เป็นอุปกรณ์แลตซ์ ทำหน้าที่เป็นรีจิสเตอร์เอาต์พุต โดยรับคำสั่งจากไมโครคอมพิวเตอร์เพื่อเลือกคิสก์ไครฟ์ที่ไมโครคอมพิวเตอร์ที่ต้องการติดต่อ และควบคุมการอินเทอร์รัพต์ รวมทั้งรีเซตวงจรควบคุมคิสก์ไครฟ์

4) วงจรลอครหัส ทำหน้าที่เปลี่ยนรหัสจากพอร์ตควบคุม เป็นสัญญาณควบคุมการทำงานของคิสก์ไครฟ์

5) วงจรกำหนดช่วงเวลาและสัญญาณนาฬิกา (Clock and Timing Circuit) เป็นวงจรที่กำเนิดสัญญาณนาฬิกา และช่วงเวลาต่างๆ ของวงจร

6) วงจรชดเชยการเขียน วงจรที่ทำหน้าที่ชดเชยผลของความเพี้ยนจากการเลื่อนจุดยอด (Peak Shift Distortion)

2.3.2 ปรากฏการณ์การเพี้ยนแบบเลื่อนจุดยอด

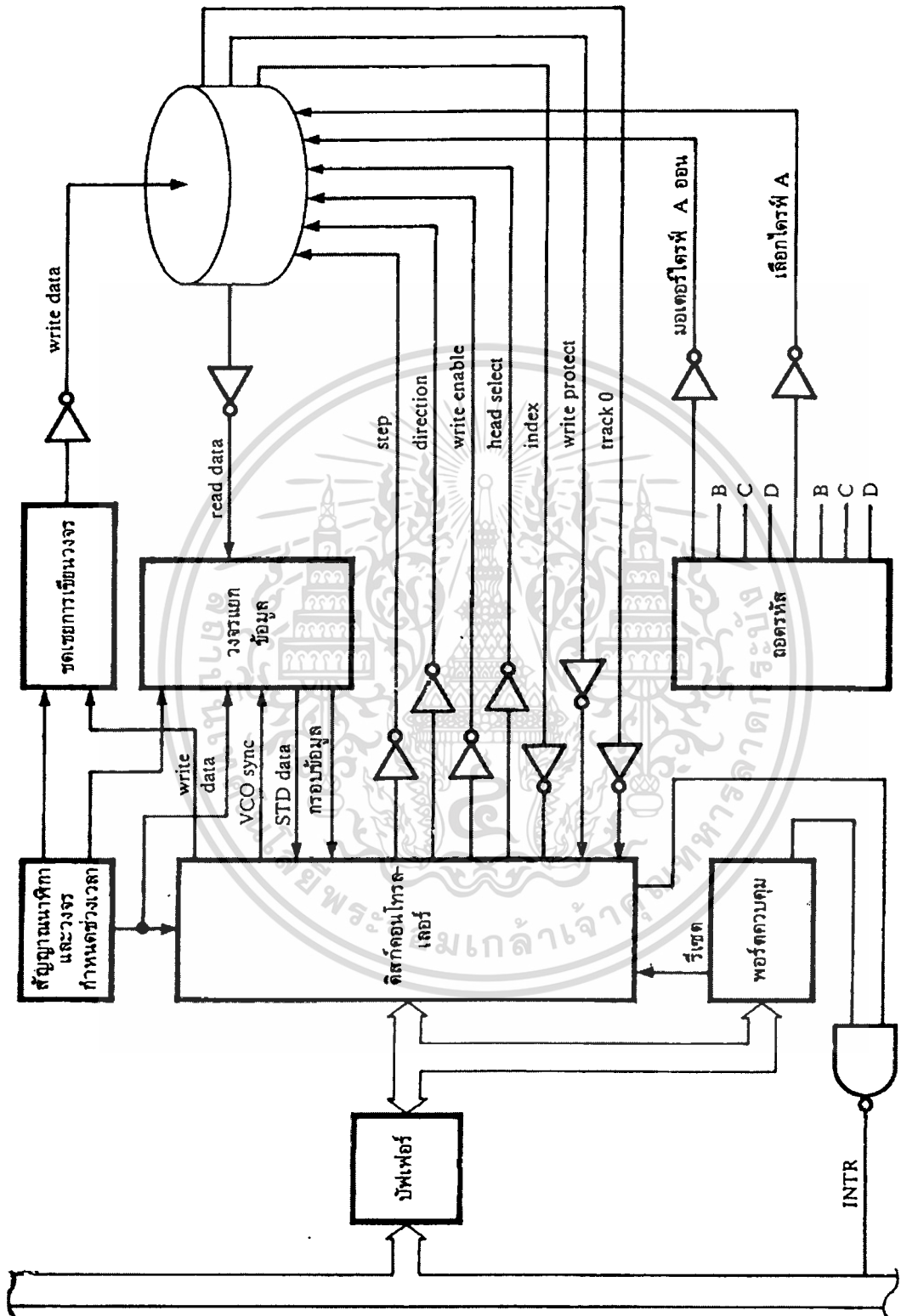
การเก็บข้อมูลในแผ่นฟลอปปี้ดิสก์ โดยการเหนี่ยวนำสารแม่เหล็กให้เกิดขั้วแม่เหล็กขึ้น โดยการสร้างสนามแม่เหล็กจากหัวบันทึก และเมื่อทำการอ่านข้อมูลก็ใช้สนามแม่เหล็กบนคิสก์เกิดเหนี่ยวนำให้เกิดกระแสบนหัวอ่าน รูปที่ 2.13 แสดงสัญญาณที่ทำการบันทึกและการอ่าน จะเห็นได้ว่าจากลักษณะสมบัติของสนามแม่เหล็กบนคิสก์เกิดจะทำให้ช่วงเวลาของการอ่านข้อมูล แตกต่างกับการเขียนข้อมูล ซึ่งปรากฏการณ์นี้จะมีผลกับข้อมูลในคิสก์เกิดที่บรรจุในแทร็กด้านใน เพราะการเก็บข้อมูลจะทำการเก็บจำนวนข้อมูลเท่ากันทุกๆ แทร็ก ในขณะที่แทร็กด้านในจะมีเส้นรอบวงน้อยกว่าแทร็กด้านนอก ทำให้ความหนาแน่นของข้อมูลสูงกว่าแทร็กด้านนอก และปัญหานี้จะมีผลในกรณีทำการมอดูเลชันแบบ MFM เพราะมีความหนาแน่นของข้อมูลสูง ผลของการเลื่อนจุดยอดจึงทำให้เกิดความผิดพลาดในการอ่านข้อมูล

ดังนั้นในวงจรควบคุมคิสก์ไครฟ์จึงจะต้องมีวงจรชดเชยการเขียน ในการบันทึกข้อมูลโดยทำการควบคุมข้อมูลที่ถูกบันทึกให้ “เร็ว” (Early) หรือ “ช้า” (Late) จากตำแหน่งเดิม วงจรแยกข้อมูล (Data Separator) ทำหน้าที่รับข้อมูลที่ได้จากการอ่านจากคิสก์ไครฟ์ ซึ่งถูกทำการมอดูเลตมาแยกสัญญาณนาฬิกาออก เพื่อส่งให้กับวงจรควบคุมคิสก์ไครฟ์

2.3.3 การตรวจสอบข้อมูลผิดพลาดในวงจรควบคุมคิสก์ไครฟ์

การตรวจสอบข้อมูลผิดพลาดเป็นหน้าที่สำคัญของวงจรควบคุมคิสก์ไครฟ์ ซึ่งจะถูกระทำโดยตัววงจรควบคุมคิสก์ไครฟ์ หลักการตรวจสอบทำโดยวิธีเช็คผลบวก (Check Sum)

CRC เป็นการตรวจสอบความผิดพลาดของข้อมูลในคิสก์เกิด โดยทุกๆ เซกเตอร์จะมี CRC อยู่ 2 ไบต์ บิตของข้อมูลจะถูกหารโดยโพลิโนเมียลตัวกำเนิด $G(x)$ เช่น $G(x) = x^7 + x^5 + x + 1$ เศษส่วนที่เหลือ คือ CRC ซึ่งจะถูเก็บไว้ใน 2 ไบต์หลังสุดของข้อมูล เมื่อทำการอ่านข้อมูลจากคิสก์เกิดใดๆ ขึ้นอีกทั้งยังมีให้ดูแปลงเนื้อหาและตัวอย่างอ้างอิงถึงจากเอกสารทุกครั้งที่มีารนำไปใช้ คิสก์เกิดรวมทั้งค่า CRC ถ้าเศษส่วนของการหาข้อมูลด้วย $G(x)$ ไม่เท่ากับ CRC แสดงว่าเกิดความ



รูปที่ 2.12 แผนผังของวงจรควบคุมคิสก์โครฟี

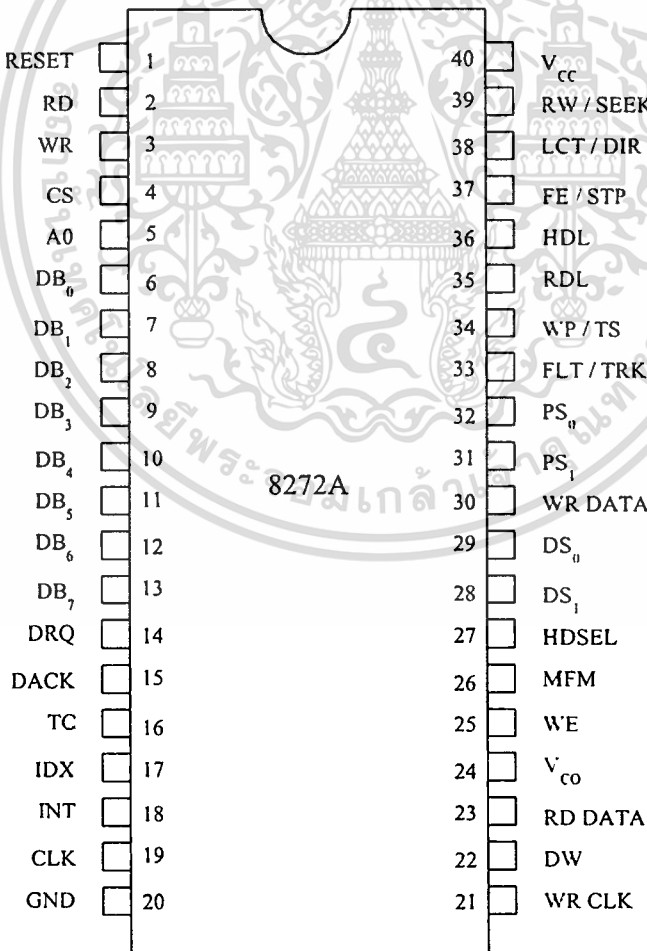
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่ควรนำออกไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.4 โครงสร้างและการทำงานของ FDC μ PD765A

μ PD765A เป็นชิปที่มีขา 40 ขา โดยทำหน้าที่เป็นตัวควบคุมฟลอปปีดิสก์ไครฟ์ได้ 2 แบบ คือ FM และ MFM ตัว μ PD765A จะสร้างสัญญาณควบคุมโดยมีวงจรการแลกเปลี่ยนข้อมูล และการชดเชย การเขียน (Write Precompensate) ประกอบการติดต่อส่งข้อมูลทำโดยใช้ DMA (Direct Memory Access) หรือการอินเทอร์รัพต์

μ PD765A มีคำสั่งในการควบคุมดิสก์ไครฟ์อยู่ 15 คำสั่ง ได้แก่ Read Data, Write Data, Recalibrate, Read ID, Write Deleted Data, Sense Interrupt Status, Read Selected Data, Format a Track, Sense Drive Status, Read a Track, Seek, Specify, Scan Equal, Scan High or Equal, Scan Low or Equal รายละเอียดของคำสั่งจะได้กล่าวถึงต่อไป

การจัดขาของ μ PD765A และหน้าที่ของสัญญาณต่างๆ แสดงดังรูปที่ 2.12 และรูปที่ 2.13

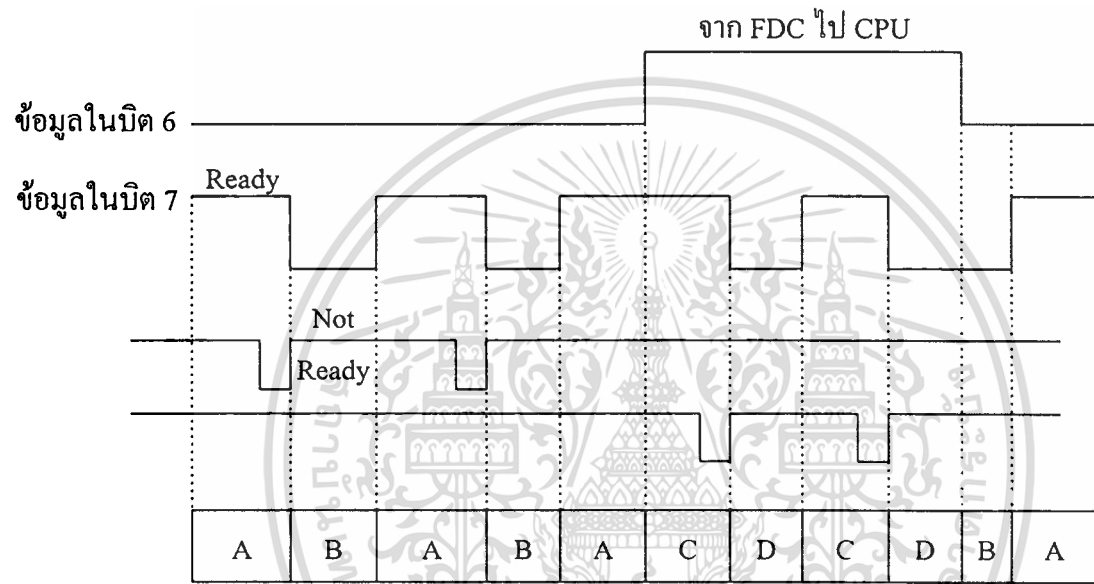


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.14 ตำแหน่งสัญญาณของไอซี μ PD765A

บิต 7 บอกว่าข้อมูลในรีจิสเตอร์ข้อมูลพร้อมที่จะติดต่อกับซีพียู

การมีข้อมูลในรีจิสเตอร์สถานะนี้ก็เพื่อให้ซีพียูติดต่อกับ FDC ในช่วงเวลาที่ถูกต้อง เป็นการตรวจสอบซึ่งกันและกัน กล่าวคือ FDC สามารถติดต่อกับคิสก์ไครฟ์ได้ 4 ตัว ดังนั้นต้องมีการแบ่งช่วงเวลาเพื่อไม่ให้เกิดการผิดพลาดว่า ขณะนี้การเขียนหรืออ่านในรีจิสเตอร์ข้อมูลเป็นของคิสก์ไครฟ์ตัวใด พิจารณาแผนผังเวลารูปที่ 2.16 คู่มือการแบ่งการทำงานในแต่ละไครฟ์



- โดยที่
- A = รีจิสเตอร์ข้อมูลพร้อมสำหรับการเขียนโดย CPU
 - B = รีจิสเตอร์ข้อมูลไม่พร้อมสำหรับการเขียนโดย CPU
 - C = รีจิสเตอร์ข้อมูลพร้อมสำหรับการอ่านโดย CPU
 - D = รีจิสเตอร์ข้อมูลไม่พร้อมสำหรับการอ่าน

รูปที่ 2.16 สถานภาพของ μ PD765A ที่รีจิสเตอร์แสดงสถานะหลัก

2) รีจิสเตอร์ข้อมูล (Data Register) เป็นรีจิสเตอร์ที่สำคัญเพราะทำให้ซีพียูติดต่อกับ FDC ได้ โดยซีพียูสามารถสั่งการทำงานโดยส่งคำสั่งเข้ามาบอก FDC ผ่านทางรีจิสเตอร์นี้ และเมื่อ FDC ทำงานแล้วจะให้ผลลัพธ์ของการทำงานโดยบอกกลับทางรีจิสเตอร์นี้อีกเช่นกัน การทำงานซีพียูจะส่งคำสั่งมาโดยคำสั่งมีทั้งสั้นให้ใช้ได้ 15 คำสั่ง ดังนั้นซีพียูจะต้องส่งรหัสคำสั่งและพารามิเตอร์ของการทำงาน เช่น ถ้าต้องการให้อ่านข้อมูลจะต้องส่งรหัสคำสั่งที่ชื่อ Read Data เมื่อส่งรหัสคำสั่งแล้ว

ต้องให้พารามิเตอร์อื่นตามมา เช่น ดิสก์ที่ให้อ่านมีการเก็บข้อมูลแบบใด มี 2 หน้าทีหรือไม่ ให้อ่านทีหน้าอะไร แทร็กเท่าไร เซกเตอร์เท่าไร เซกเตอร์ละกี่ไบต์ ให้อ่านกี่เซกเตอร์ ดังนั้นการทำงานจึงแบ่งเป็น 3 ขั้นตอนดังนี้

2.1) ขั้นตอนการรับคำสั่ง (Command Phase) FDC จะรับคำสั่งและข้อมูลที่กำหนดการทำงานจากตัวซีพียู

2.2) ขั้นตอนการเอกซิกิวต์ (Execute Phase) FDC แปลความหมายคำสั่งแล้วจะปฏิบัติตามคำสั่ง

2.3) ขั้นตอนส่งผลลัพธ์ (Result Phase) FDC ส่งผลลัพธ์กลับให้ซีพียูผ่านทางรีจิสเตอร์ข้อมูล ซึ่งซีพียูต้องอ่านจากรีจิสเตอร์นี้ไปตีความ สิ่งที่น่าสนใจคือ การติดต่อกับ FDC จะใช้เพียงรีจิสเตอร์ตัวเดียว แต่เมื่อโปรแกรมคำสั่งส่งไปให้ FDC จะต้องใช้หลายไบต์ เช่น คำสั่ง Read Data ต้องใช้ถึง 9 ไบต์ และได้ผลลัพธ์กลับคืนมาถึง 7 ไบต์ ดังนั้นโครงสร้างของรีจิสเตอร์ข้อมูลนี้จะมีโครงสร้างเป็นสแต็ก ซึ่งสามารถเขียนเข้ามาได้หลายไบต์ และอ่านกลับไปได้หลายไบต์เช่นกัน การเขียนหรืออ่านจะกระทำเหมือนกับการติดต่อกับพอร์ตอินพุตเอาต์พุตแบบปกติ

2.3.5 FDC (Floppy Disk Controller)

ใน System Board นั้นดิสก์ไครฟ์จะมีความทำงานที่พอร์ตต่างๆ ดังนี้

Output Register ที่ 3F2H

Status Register ที่ 3F4H

Data Register ที่ 3F5H

หัวใจของดิสก์ไครฟ์คือ ไมโครโปรเซสเซอร์เบอร์ μ PD765 หรือเป็นไมโครโปรเซสเซอร์เบอร์อื่นที่ทำหน้าที่เป็น FDC แสดงดังรูปที่ 2.17 โดย FDC สามารถที่จะขยายการควบคุมการทำงานของ FDC ได้ถึง 4 ตัว

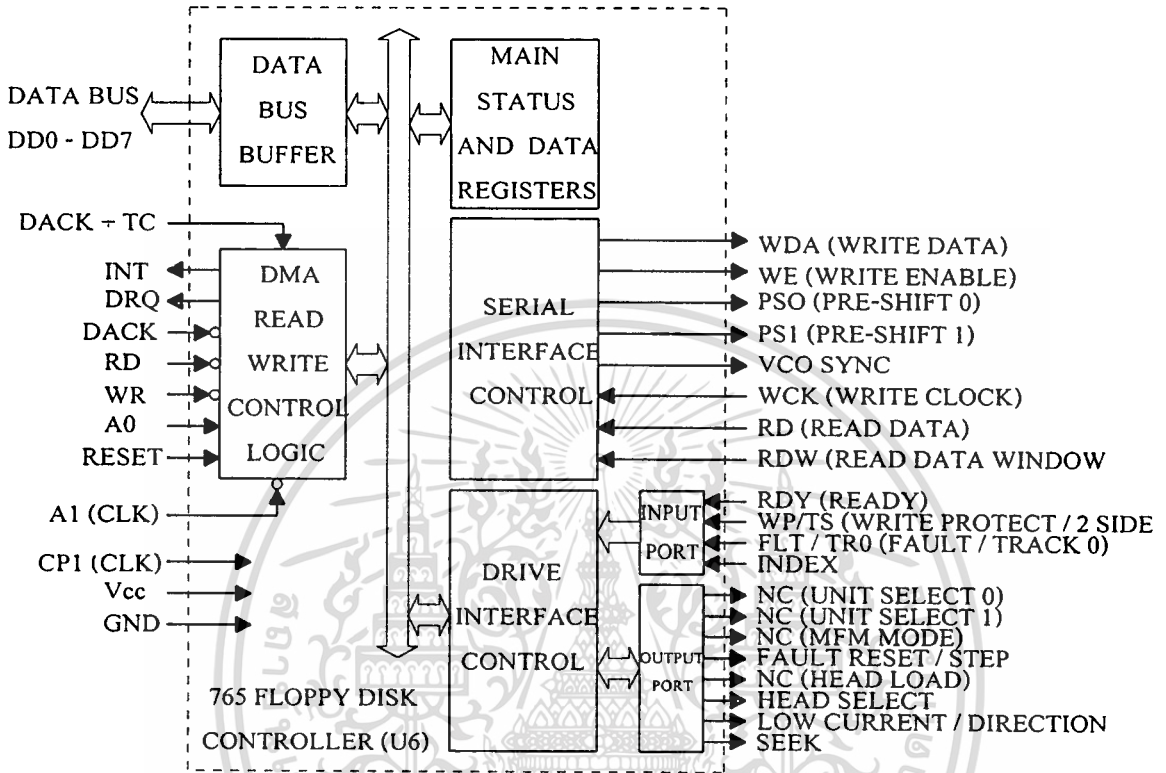
จากรูปที่ 2.17 เป็นสถาปัตยกรรมที่ทำให้ง่ายเข้ากับการขดเชยการเขียนและการอ่านข้อมูลแบบ PLL (Phase Locked Loop) FDC จะประกอบด้วย Data Bus Interface, DMA Read/Write Control Logic

รีจิสเตอร์ที่ทำหน้าที่ Status, Data และอินเตอร์เฟสแบบอนุกรมสำหรับการควบคุมไครฟ์และไครฟ์อินเตอร์เฟส ซึ่งจะทำหน้าที่ควบคุมการเขียน/อ่าน และควบคุมหัวอ่าน

สัญญาณ Hand Shaking จะอนุญาตให้ DMA Interface ติดต่อกับ DMA Controller ซึ่งจะอยู่บน Mainboard

เอกสารนี้ FDC สามารถทำงานได้ทั้งมี DMA และไม่มี DMA ในการทำงานแบบไม่มี DMA นั้น FDC จะมีการร้องขออินเตอร์รัพต์กับซีพียู 8088 บน System Board ทุกครั้งที่มีการส่งถ่ายข้อมูล ในโหมด

การทำงานแบบที่มี DMA นั้นซีพียูจะส่งคำสั่งไปยัง FDC และ DMA



รูปที่ 2.17 แผนผังการทำงานภายในของ 765 FDC

FDC (อ้างอิง μPD765) สามารถรับคำสั่งการทำงานได้ทั้งหมด 15 คำสั่ง รวมทั้งคำสั่งการ Format Disk, Read Data, Read ID, Reading a Track, Reading Delete Data และการสั่งการอินเตอร์รัพต์ และสถานะใดก็ได้ซึ่งส่งข้อมูลแบบ 8 บิต ซึ่งมีการกำหนดรูปแบบไว้แล้ว

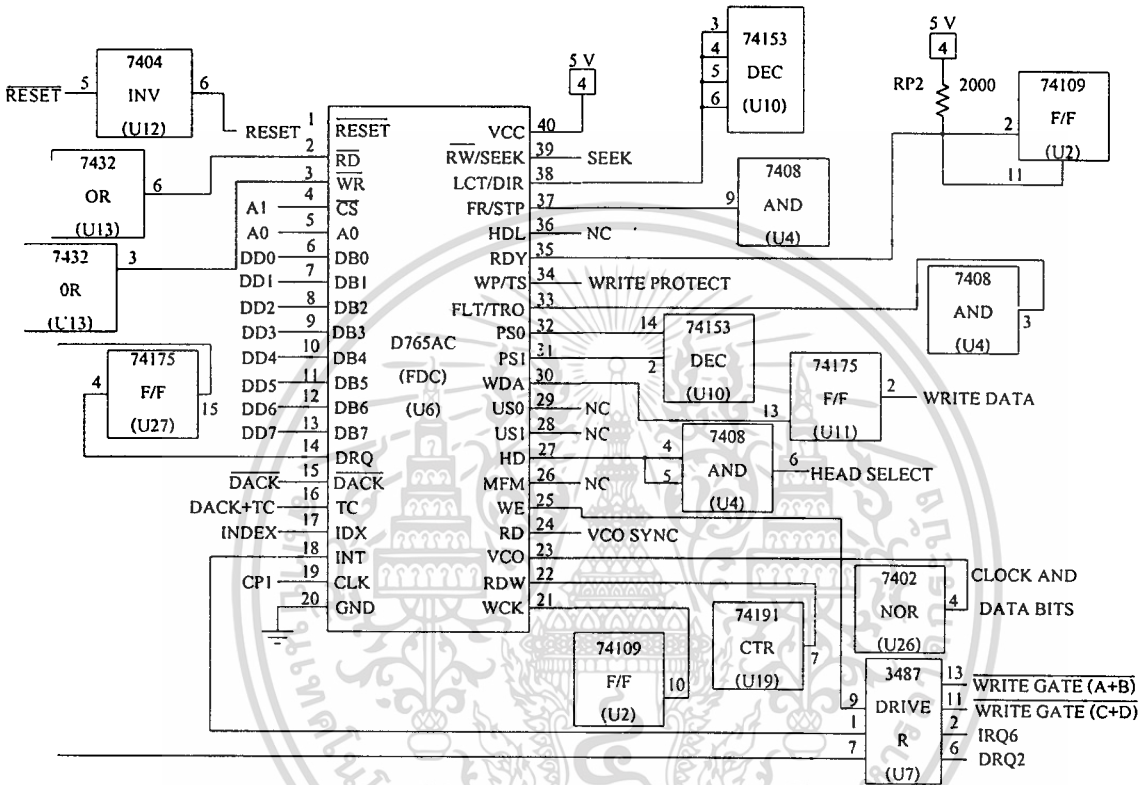
การ Sync ข้อมูลและการตรวจสอบความผิดพลาดของข้อมูลจะถูกทำงานอัตโนมัติโดยไอซี FDC สำหรับอุปกรณ์ที่ติดต่อกับภายนอกจะใช้สัญญาณนาฬิกาของระบบ และรวมถึงสัญญาณนาฬิกาที่ใช้ในการเขียนการอ่านข้อมูลสำหรับ FDC ด้วย

รูปที่ 2.18 แสดงลักษณะการจัดตำแหน่งขาของ FDC และการควบคุมของ IBM PC ซึ่งแสดงให้เห็นว่าบางขาไม่ได้ใช้งานดังนี้คือ ขา 26 (MFM) จะเป็นแฟล็กเอาต์พุต โดยจะทำหน้าที่แสดงการทำงานในโหมด MFM แต่การแสดงสถานะนี้ไม่มีความจำเป็นต้องใช้ ดังนั้น ขา 26 จึงไม่มีการต่อใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในระบบ แต่การเลือกใคร่ 0 จะถูกเลือกโดย Main Board อยู่แล้ว 2 ขานี้ไม่ใช้งาน

และสุดท้ายคือไขมัน 36 (HDL) จะไม่ใช้งานเพราะหัวอ่านเขียนจะสัมผัสกับแผ่นดิสก์อยู่แล้ว



รูปที่ 2.18 การจัดตำแหน่งขาของ FDC

1. Stack Stores Data

รีจิสเตอร์แสดงสถานะหลักและรีจิสเตอร์ข้อมูลสามารถควบคุมได้โดยซีพียู 8088 ที่ตำแหน่ง 3F4H และ 3F5H โดยรีจิสเตอร์แสดงสถานะจะมีขนาด 8 บิต สามารถอ่านได้เพียงอย่างเดียว โดยจะแสดงสถานะการทำงานของ FDC ซึ่งนำมาใช้งานบางเวลาเท่านั้น สำหรับรีจิสเตอร์ข้อมูลในการใช้งานนั้น จะมีการเลือกใช้งาน 1 ใน 4 รีจิสเตอร์ ซึ่งมีโครงสร้างเป็นแบบสแตก โดยประกอบด้วยสแตกต่างๆ ดังนี้

- ### 3. Parameters

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

3. Parameters

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- #### 4. Disk Drive Status information

2) รีจิสเตอร์แสดงสถานะหลัก

ข้อมูลที่ผ่านการอ่านออกหรือเขียนเข้าไปในตัว FDC Register โดยการเลือกการทำงานผ่านขา A0, RD, WR, และ A1 (Chip Select) โดย 8 บิตที่อ่านได้จากรีจิสเตอร์แสดงสถานะหลักจะมีการกำหนดค่าไว้ดังนี้ คือ

ตารางที่ 2.3 สถานะการทำงานของบิตต่างๆ ในรีจิสเตอร์แสดงสถานะหลัก

บิตที่	ชื่อ	สถานะการทำงาน
0	Drive 0 Busy	Drive 0 อยู่ในสภาวะ SEEK MODE
1	Drive 1 Busy	Drive 1 อยู่ในสภาวะ SEEK MODE
2	Drive 2 Busy	Drive 2 อยู่ในสภาวะ SEEK MODE
3	Drive 3 Busy	Drive 3 อยู่ในสภาวะ SEEK MODE
4	FDC Busy	FDC อยู่ในระหว่างการเขียนอ่าน
5	Non-DMA Mode	FDC ไม่ได้ทำงานใน DMA เป็น 1 เมื่ออยู่ในระหว่างการประมวลผล เป็น 0 เมื่อสิ้นสุดการทำงาน
6	Data I/O	แสดงทิศทางข้อมูลระหว่าง Data Register กับ FDC เป็น 1 เมื่อส่งข้อมูลจาก Data Register ไปยังซีพียู
7	MASTER RQST	แสดงเมื่อ Data Register พร้อมที่จะรับหรือส่งข้อมูล ใช้งานร่วมกับบิตที่ 6 เมื่อตรวจ Hand Shacking ของสถานะ Ready และทิศทางกับซีพียู

ขา Chip Select ควรอยู่ที่สภาวะแอกทีฟ Low เพื่อให้มีการทำงานที่ขา A0, RD, WR

ตารางที่ 2.4 การทำงานที่สภาวะต่างๆ ของขา Chip Select

ขา A0	ขา RD	ขา WR	ฟังก์ชัน
0	0	1	อ่านข้อมูลจากรีจิสเตอร์แสดงสถานะ
1	0	1	อ่านข้อมูลจากรีจิสเตอร์ข้อมูล
1	1	0	เขียนข้อมูลไปยังรีจิสเตอร์ข้อมูล

3) ขั้นตอนการทำงานของ FDC

ข้อมูลต่างๆ ที่มีการส่งระหว่างซีพียูและ FDC ในการควบคุมดิสก์ไครฟ์ให้ปฏิบัติตามคำสั่งนั้น ในการโปรแกรมคำสั่งจะประกอบด้วยขั้นตอนการออกคำสั่ง, ขั้นตอนการปฏิบัติงาน และขั้นตอนการแสดงผลการทำงาน

รหัสคำสั่งจะถูกส่งไปเป็นอันดับแรกโดยส่งผ่านไบต์คำสั่งไปตามลำดับชั้น ในลำดับการออกคำสั่งนี้ไอซี FDC จะรับเอาข้อมูลที่สำคัญในการประมวลผลไปปฏิบัติ โดยการประมวลผลโปรแกรมที่จะส่งไปยังดิสก์จะมีการเก็บข้อมูลไว้ภายในรีจิสเตอร์ข้อมูล โดยจะเกิดเหตุการณ์เหล่านี้ขึ้นภายหลังจากการที่ระบบบูทริเซต และมีการรับคำสั่งการทำงานเข้ามาเสร็จสิ้นแล้วเท่านั้น

ในลำดับขั้นตอนการปฏิบัติงานเป็นการทำงานตามคำสั่ง โดยลำดับการทำงานจะสิ้นสุดลงก็ต่อเมื่อ ข้อมูลไบต์สุดท้ายถูกส่งออกไปเป็นที่เรียบร้อยแล้ว (สังเกตจากขาสัญญาณ TC ของ 8273 DMA ที่ป้อนให้กับ FDC) หรือมีการทำงานผิดพลาด

หลังจากที่การทำงานในส่วนของการปฏิบัติงานเสร็จสิ้นแล้ว FDC ก็จะเข้าสู่การทำงานของส่วนการแสดงผล ในลำดับนี้จะมีการแสดงสถานะการทำงานต่างๆ ให้กับซีพียู 8088 ทุกๆ ไบต์ (ปกติมี 7 ไบต์) ทำหน้าที่ในการแสดงผลการทำงานซึ่งต้องอ่านทั้งหมดโดยซีพียู หากยังอ่านข้อมูลสถานะทั้งหมดเหลือค้างอยู่จะต้องไม่มีการส่งคำสั่งการทำงานใหม่เข้าไปในตัว FDC หลังจากที่ซีพียูอ่านข้อมูลที่ได้จาก FDC ทั้งหมดแล้ว FDC จะกลับเข้าสู่การทำงานของการรับคำสั่งใหม่อีกครั้งและพร้อมที่จะปฏิบัติงานตามคำสั่งต่อไป

การเขียนข้อมูลหรืออ่านข้อมูลจากรีจิสเตอร์ข้อมูลของ FDC ในลำดับขั้นตอนการทำงานของการออกคำสั่ง และการแสดงผลการทำงานนั้นจะถูกควบคุมการทำงานโดยซีพียู ซึ่งจะประมวลการเลือกทำงานมาจากรีจิสเตอร์แสดงสถานะหลักของ FDC อีกทีหนึ่ง บิตที่ 6 และ 7 ควรจะมีสถานะ 0 และ 1 ตามลำดับก่อนที่จะมีการส่งคำสั่งให้กับไอซี FDC ลำดับการอ่านสถานะการทำงานบิตที่ 6 และ 7 ควรจะมีสถานะ 1 ทั้งคู่ เพื่อที่จะใช้ในการอ่านข้อมูลจากรีจิสเตอร์แสดงสถานะหลักของ FDC

ข้อมูลที่ถูกส่งหรือรับมาจาก FDC ที่เกิดขึ้นระหว่างการทำงานในช่วงการปฏิบัติ ในระหว่างที่ DMA ส่งข้อมูลให้กับขา 14 ของ FDC (μ PD765) สัญญาณจะถูกส่งผ่าน 74LS175 Flip Flop และส่งข้อมูลผ่านต่อไปยังขาที่ 6 ของไอซี 3487 Drive เป็นสัญญาณ DRQ2 สัญญาณ DRQ2 นี้ ซีพียูจะใช้ในการร้องขอการทำงานของ DMA, 8273 DMA บน System Board จะตอบสนองโดยการสร้างสัญญาณ DMA Acknowledge 2 (DACK 2) โดยสัญญาณนี้จะกลายเป็นสัญญาณ DACK

และ DACK&TC สัญญาณ DACK จะต่อเข้ากับขา 15 ของ μ PD765 เป็นสัญญาณการตอบรับการคำสั่ง (Acknowledge) ที่ขา RD และ WR จะถูกควบคุมจาก 8273 DMAC ในระหว่างการถ่ายเทข้อมูลไปใช้

สัญญาณ DRQ จะถูกรีเซตหลังจากการส่งข้อมูลชุดสุดท้ายไปแล้วขา 18 INT จะกลับมาเป็นลอจิก High อีกครั้งทำให้สัญญาณ IRQ6 ออกจากขา 2 ของ U6 โดยสัญญาณ IRQ6 จะแสดงสถานะการทำงานในลำดับขั้นการปฏิบัติงานสิ้นสุดลงแล้ว และเริ่มการทำงานในลำดับขั้นการแสดงผล ช่วยให้โปรแกรมทราบว่าให้มีการเรียก Routine มาทำการตรวจสอบข้อมูลการแสดงผลและรีเซตการร้องขอการอินเตอร์รัพต์ สัญญาณ Terminal Count (TC) ที่ออกมาจาก 8237 DMAC จะทำการ AND ข้อมูลกับสัญญาณ Invert ของ DACK2 ซึ่งจะทำให้เกิดเป็นสัญญาณ DACK&TC ต่อเข้ากับขา 16 ของ μ PD765 โดยใช้ในการบอกการเริ่มต้นการนับและส่งข้อมูลของ DMA

สำหรับการส่งข้อมูลที่ไม่ใช่ DMA การร้องขอการถ่ายเทข้อมูลจะใช้บิตที่ 7 ของรีจิสเตอร์แสดงสถานะหลัก และสัญญาณอินเตอร์รัพต์ที่ขา 18 ของ μ PD765 ทั้งสองตัวนี้จึงมีสถานะเป็น High สัญญาณ INT จะกลายเป็นสัญญาณ IRQ6 ที่ขา 18 สำหรับการอินเตอร์รัพต์ไปยัง IBM PC System โดยบิตที่ 7 (Master Request) จะมีการตรวจสอบจากโปรแกรมเพื่อใช้ในการนำมาร่วมตัดสินใจในการตรวจสอบสถานะระบบ เมื่อสัญญาณ IRQ6 ถูกจัดส่งให้กับ 8088 มันจะตอบสนองโดยการอ่านข้อมูลจาก μ PD765 (สัญญาณ RD แอคทีฟ Low) หรือเขียนข้อมูลลงไปใน μ PD765 (สัญญาณ WR แอคทีฟ Low) ในระหว่างนี้จะไม่มีการร้องขอการส่งข้อมูล (Transfer Interrupt) เมื่อมีการส่งข้อมูล แต่เมื่อซีพียูได้รับสัญญาณ Transfer Interrupt แสดงว่าขณะนี้ข้อมูลถูกส่งไปครบตามจำนวนแล้ว โดยสัญญาณที่ขา 16 DACK&TC จะมีสถานะ High สัญญาณนี้จะเกิดก่อนที่สัญญาณไบต์สุดท้ายจะมีการส่งหรือรับ

เมื่อขา 16 ของ μ PD765 มีสถานะ High เป็นการแสดงว่า FDC หยุดการร้องขอการโอนย้ายข้อมูลโดย μ PD765 จะทำการอ่านหรือเขียนข้อมูลจากแผ่นดิสก์ จนกระทั่งการทำงานเสร็จสิ้นข้อมูลที่ถูกรับหลังจากการเกิดสัญญาณ TC จะไม่ถูกนำไปใช้งาน และในกรณีที่การเขียนข้อมูลลงแผ่นดิสก์ หลังจากการเกิดสัญญาณ TC ข้อมูลที่ถูกส่งจะกลายเป็น 0 และส่วนที่ยังคงเหลือในเซกเตอร์จะกลายเป็น 0 ทั้งหมด

4) FDC (μ PD765) Electronics

FDC μ PD765 จะทำงานทั้งหมด 6 ประการกับดิสก์ไดรฟ์ คือ อันดับแรก เลือกการทำงานของดิสก์ไดรฟ์หนึ่งในจำนวน 4 ตัว, สองคือ การควบคุมตำแหน่งแทร็ก โดยการส่งสเตปพัลส์ไปควบคุมหัวอ่านให้เคลื่อนที่ไปยังตำแหน่งที่ต้องการเขียนหรืออ่านได้ FDC จะทำการเก็บตำแหน่งของแทร็กและหัวอ่านไว้ เพื่อใช้ในการคำนวณในการหาระยะสเตปการทำงานของตำแหน่งหัวอ่านไปยังตำแหน่งแทร็กอื่น, สามคือ การตรวจดูข้อมูลในแทร็กจนกระทั่งมีการส่งข้อมูลการเลือกเซกเตอร์มาให้ ระยะเวลาที่ใช้ในการค้นหาลูกจากแทร็กหนึ่งถึงแทร็กหนึ่งจะใช้เวลา 6 ms, สี่ ทำหน้าที่ควบคุมการโหลดหัวอ่าน โดยใช้เวลา 35 ms เพราะหัวอ่านจะคงค้างเพื่อเซตค่า 15 ms ก่อนที่จะทำ

การเขียนหรืออ่าน และหัวจะทำการไหลหลังจากการทำงานเขียนหรืออ่าน ดังนั้นระยะเวลาการไหลหัวจะใช้เวลา 35 ms ก่อนการอ่านหรือเขียน ดังได้กล่าวแล้วว่าหัวอ่านจะคงค้างก่อนการเขียนหรืออ่าน 15 ms ในขณะที่มอเตอร์จะมีการใช้เวลาเริ่มต้นการทำงาน 250 ms ก่อนที่จะทำการหมุน, หัว การอ่านข้อมูลของ FDC จะเป็นสัญญาณผสม (สัญญาณนาฬิกาและข้อมูล) โดยจะต้องนำมาแบ่งแยกข้อมูลอีกครั้งหนึ่งด้วยอัตรา 500 kb/sec สำหรับ Double Density Disk ข้อมูลที่ถูกส่งมาแบบอนุกรมจะประกอบด้วยข้อมูล 8 บิต และส่งเข้าหน่วยความจำทุกๆ 16 μ s การแยกข้อมูลออกจากสัญญาณนาฬิกาจะใช้วงจร PLL หลังจากข้อมูลเกิดการ Sync กันแล้ว จะเริ่มทำการแยกข้อมูลออกจากสัญญาณนาฬิกา FDC จะใช้ช่วงข้อมูลใน Data Windows เหล่านี้มาสร้างข้อมูลใหม่อีกครั้งก่อนที่จะการบันทึกข้อมูลลงในดิสก์

ในลำดับสุดท้าย FDC (μ PD765) ใช้ในการตรวจสอบ CRC (Cyclic Redundancy Check) เพื่อใช้ในการหาข้อผิดพลาดในการบันทึก

5) Disk Drive Electronics

ดิสก์ไดรฟ์จะถูกต่อเข้ากับ Adapter Board เพื่อทำการเปลี่ยนคำสั่งที่อยู่ในรูปดิจิทัลให้อยู่ในรูปของการเคลื่อนไหวทางกายภาพ (เช่น การเลือกดิสก์, การเคลื่อนหัวอ่าน, และการไหลหัวอ่าน) และจัดเตรียมความพร้อมในการทำงานและแจ้งสถานะการทำงาน (เช่น ความพร้อมที่จะเขียนหรืออ่าน การเขียนข้อมูลผิดพลาด มีการป้องกันการเขียน) โดยแจ้งไปยัง Adapter Board การทำงานเหล่านี้จะเป็นรูปแบบของแอนะลอก ตารางที่ 2.5 จะอธิบายรูปแบบสัญญาณที่ใช้เชื่อมต่อข้อมูลใน Adapter Board กับ FDC

6) System Board Interface

ตำแหน่ง 10 บิตจะถูกต่อเข้ากับ Adapter Board เป็นลักษณะ I/O แบบ 3 สถานะ ที่ตำแหน่ง 3F2H จะเป็นไอซี 74LS245 ซึ่งทำหน้าที่เป็นอินพุตริจิสเตอร์ (U30 จากรูปที่ 2.19) บริเวณด้านล่างซ้ายของรูปที่ 2.19 เป็นตำแหน่ง 3F4H สำหรับ FDC Main Status Register และตำแหน่ง 3F5H สำหรับ FDC Data Register ตำแหน่งทั้ง 3 นั้นจะใช้ Address Bit A4-A9 เป็นบิตด้านสูง A3 เป็นด้านต่ำ

ดิจิตอลเอาต์พุตริจิสเตอร์ (Digital Output Register) U30 มีลักษณะการทำงานแบบ 2 ทิศทาง ควบคุมโดยขา S/R ที่ขา 1 สัญญาณนี้จะได้มาจากอุปกรณ์รอบๆ ในวงจรมันเอง ตำแหน่ง A3 และสัญญาณ AEN จะถูกนำมารวมเข้าด้วยกันโดย 74LS02 NOR Gate (U29) ต่อเข้ากับขา 11 และ 12 ของไอซี 74LS30 8 อินพุต NAND Gate (U28) อีก 6 อินพุตจะถูกต่อเข้ากับ A4-A9 โดยตรง ขา 8 ของ U28 จะนำมา AND กับสัญญาณ DACK จาก U12 (74LS04) และนำมาแยกออกเป็น 2 ทาง โดยต่อเข้ากับ 74LS32 (U13) ด้านบน (ขา2) ใช้ในการควบคุมการเขียนที่ขา 3 ของ FDC

μPD765 ซึ่งจะสร้างสัญญาณนาฬิกาผ่านทาง 74LS273 (D flip-flop) U17 และส่งไปควบคุมการทำงานของมอเตอร์และการเลือกไครฟ์

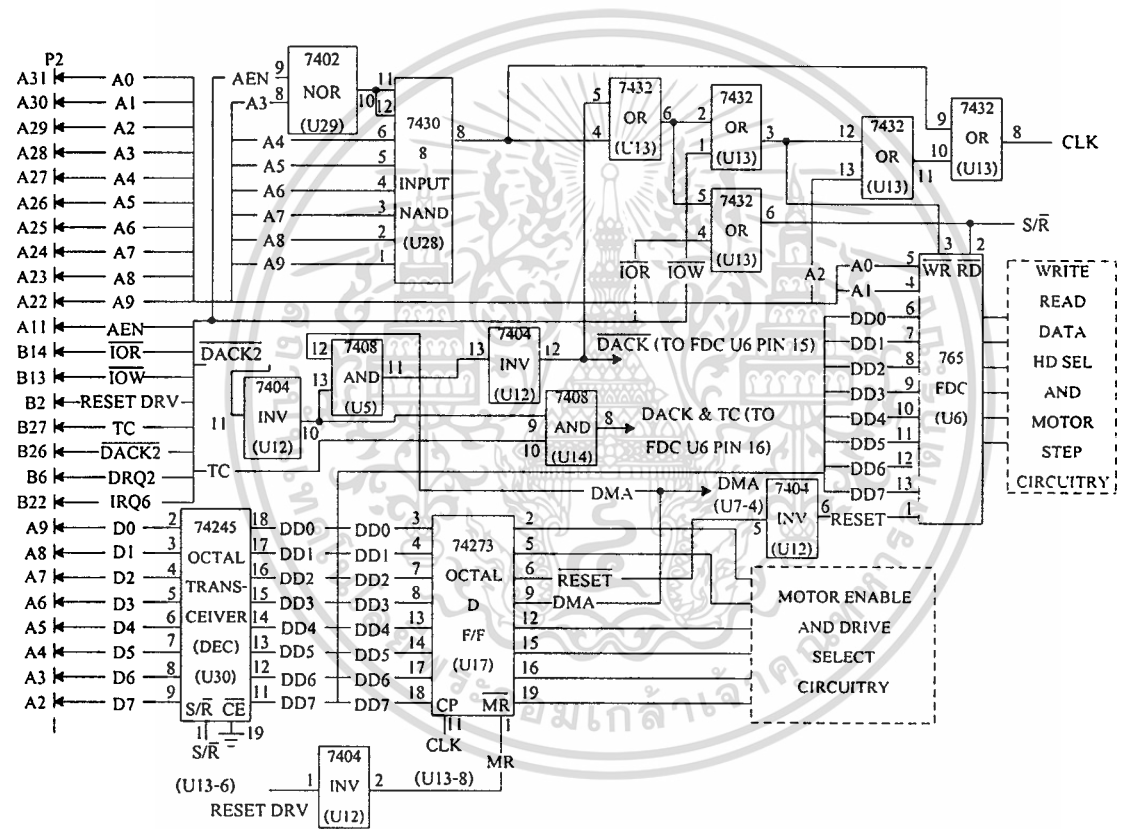
ตารางที่ 2.5 รูปแบบสัญญาณที่ใช้เชื่อมต่อข้อมูลใน Adapter Board กับ FDC

สัญญาณ	สถานะ	ขา	รายละเอียด
Drive Select	เอาต์พุต	12, 14	เลือกการทำงานของไครฟ์ โดยไครฟ์อื่นๆ จะถูกหยุดการทำงาน
Motor Enable	เอาต์พุต	10, 16	ควบคุมการหมุนมอเตอร์ของไครฟ์ที่ถูกเลือก
Step	เอาต์พุต	20	สั่งให้ไครฟ์ที่ถูกเลือกทำการเคลื่อนที่หัวอ่านไปในทิศทางที่ถูกกำหนดโดยขาสัญญาณ Direction (P18)
Direction	เอาต์พุต	18	ควบคุมทิศทางการเคลื่อนที่ของหัวอ่าน โดยเป็น High เมื่อหมุนไป และเป็น Low เมื่อหมุนกลับ
SEL HD	เอาต์พุต	32	เป็น High เมื่อเลือกหัว 1, เป็น Low เมื่อเลือกหัว 0
Write Data	เอาต์พุต	22	เป็นตัวสัญญาณที่จะทำให้เกิดการเปลี่ยนแปลงของแม่เหล็ก (Flux) ที่อยู่บนแผ่นดิสก์ โดยจะเขียนได้ก็ต่อเมื่อ Write Enable แอ็คทีฟ
Write Gate	เอาต์พุต	24	ไม่อนุญาตให้มีการเขียนจนกว่าจะมีสถานะ Low
Index	อินพุต	8	เป็นพัลส์ที่ใช้แสดงการตรวจพบตำแหน่ง Index ของดิสก์ในระหว่างที่ดิสก์หมุน
Read Data	อินพุต	30	เป็นพัลส์ข้อมูลที่ได้โดยการเปลี่ยนแปลงฟลักซ์ในแผ่นดิสก์ที่ถูกเลือกให้ทำงาน
Track 0	อินพุต	26	แอ็คทีฟเมื่อมีการเขียนหรืออ่านบนแทร็ค 0
Write Protect	อินพุต	28	จะแอ็คทีฟในกรณีที่แผ่นดิสก์มีการ Protect

อีกส่วนหนึ่งของ U13 (AND Gate) จะทำหน้าที่รับสัญญาณเอาต์พุตจาก U14 เข้าที่ขา 5 โดยจะนำสัญญาณดังกล่าวมา AND กับสัญญาณ IOR สร้างเป็นสัญญาณ S/R ออกที่ขา 6 สัญญาณนี้จะทำหน้าที่ควบคุมทิศทางของข้อมูลที่ป้อนเข้าไปยัง U30 (74LS245) สัญญาณนี้จะยอมให้มีการส่งไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลจาก U6 ไปยังบัคซาย I/O ซึ่งจะไม่มีการทำงานในส่วนนี้หาก A1 (Chip Select) มีสถานะลอจิก High

ใน Address อีก ตำแหน่ง คือ 3F4H และ 3F5H มีการเปลี่ยนแปลงไปตามบิต A0 ซึ่งจะใช้ในการเลือกข้อมูล หรือ Status (ขา 5) เมื่อ A3 เป็น Low และ A4 ถึง A9 เป็น High (00111110XXX) A2 ควรจะเป็น High เนื่องจากเป็นสัญญาณพิกากควบคุมโดยบิตดังกล่าว ดังนั้นข้อมูลที่ส่งจะเปลี่ยนได้ใหม่เป็น (001111101XX) และการกำหนดค่าที่บิต A0 กำหนดได้ดังนี้คือ หากเป็น Data Register A0 = 1, และ Status Register A0 = 0



รูปที่ 2.19 การเชื่อมต่อขาสัญญาณต่างๆ

7) Drive Motor Control Circuitry

รูปที่ 2.20 เป็นวงจรที่ใช้ในการต่อเพื่ออนุญาตการทำงานของดิสก์ไดรฟ์ และควบคุมการ

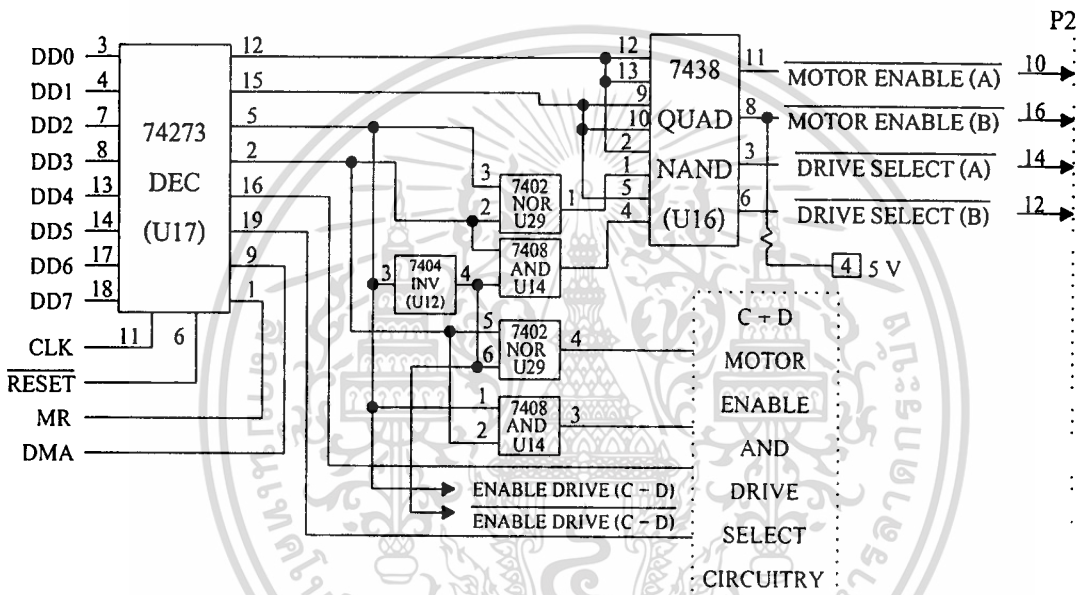
ทำงานของมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่งการอนุญาตและเลือกการทำงานของดิสก์ไครฟ์จะถูกส่งผ่านมาทาง DMA ผ่านทาง 74LS273 (D flip-flop) U17 โดยเอาต์พุตที่ออกทางขา 2, 5, 12, 15, 16 และขา 19 นั้นจะถูกผ่านวงจรลอจิกเพื่อสร้างวงจรในการเลือกดิสก์ไครฟ์ขึ้นมา

ที่ขา 5 ของ U17 (74LS273) จะแอกทีฟ High เพื่อใช้ในการ Enable Drive C&D สัญญาณจะถูกส่งผ่านทางขา 3 ส่งผ่าน 74LS02 ต่อไปยัง 74LS38 (บัฟเฟอร์) U16



รูปที่ 2.20 วงจรอนุญาตการทำงานของดิสก์ไครฟ์ และควบคุมการทำงานของมอเตอร์

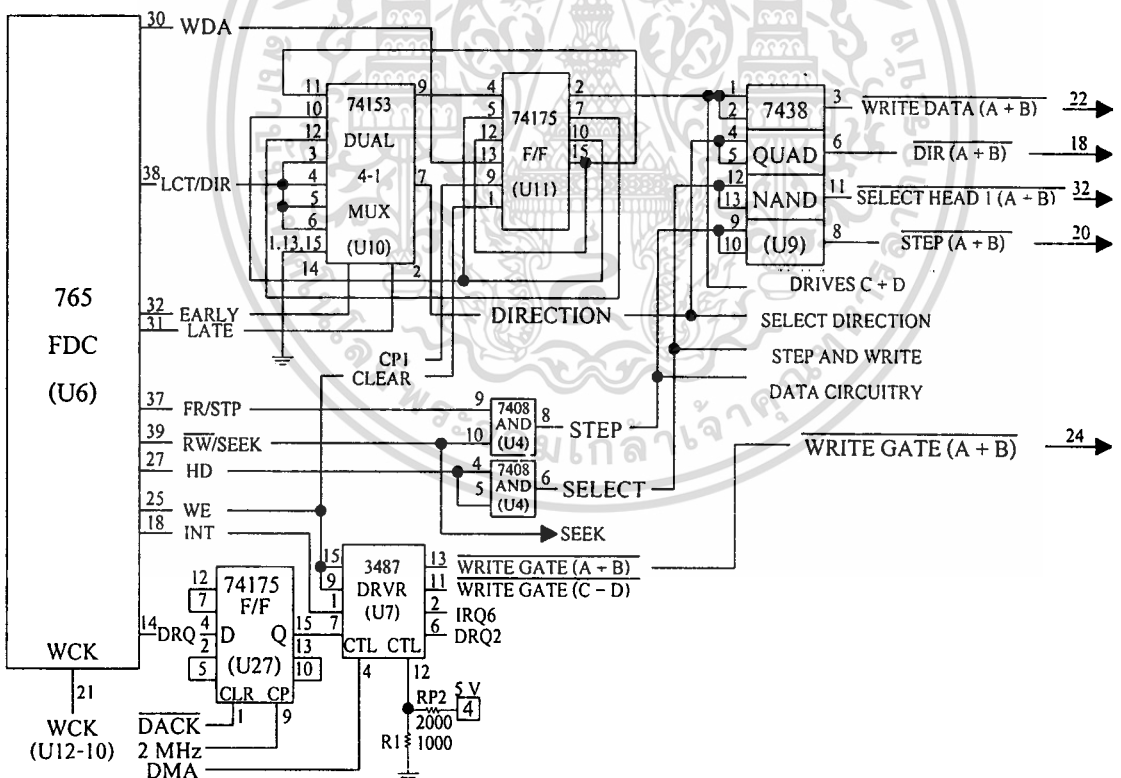
8) Disk Drive Write Circuitry

รูปที่ 2.21 เป็นแผนผังการทำงานในการเขียนข้อมูลลงในดิสก์ วงจรนี้จะทำหน้าที่สร้างสัญญาณ Clock/Data โดยข้อมูลจะส่งผ่านมาจากขา 12 ของ P2 มันจะสร้างสัญญาณ Direction (DIR A&B) ออกทางขา 18 ของ P2 และสัญญาณในการเลือกหัวอ่าน 0 หรือ 1 ใน P2 ขา 32 และสัญญาณสเตปพัลส์ที่ขา 20

ในส่วนของ 74LS153 (4 to 1 Multiplex) U10 และ 74LS 175 (D flip-flop) U11 จะเป็นตัวทำหน้าที่สร้างสัญญาณ MFM Serial Clock/Data ส่งผ่านไปจาก U11 ขาที่ 2 ออกไปสองทางคือ วงจรขับ A&B และ C&D drive สัญญาณนี้จะส่งไปเข้ายังขา 1 และ 2 ของ 74LS38 (NAND Gate) U9 เพื่อสร้างสัญญาณแอกทีฟ Low ขึ้น (Write Data A&B) ที่ขา 3 ในขณะที่ μ PD765 กำลังอยู่ในส่วนคำสั่ง ข้อมูลคำสั่งขนาด 9 ไบต์ จะถูกส่งเข้าไปใน FDC และข้อมูลนี้จะมีส่วนของการเลือก

ไครฟ์ และหมายเลขแทร็ก (Cylinder), หัวอ่าน, หมายเลขเชกเตอร์, ขนาดของเชกเตอร์ที่ใช้, หมายเลขเชกเตอร์สุดท้ายของแทร็ก, ช่องว่างระหว่างเชกเตอร์ (ช่องว่างสำหรับช่อง VCO Synchronization) และขนาดของเชกเตอร์

เมื่อเข้าสู่ช่วงการประมวลผลเริ่มต้นและ FDC เริ่มสั่งการโหลดหัวอ่าน รอประมาณ 35 ms ในการโหลดหัวอ่าน หลังจากนั้นเริ่มการอ่านเชกเตอร์ ID เมื่อตรงตามที่ได้จากการกำหนดของคำสั่ง FDC จะอ่านข้อมูลที่จะทำการเขียนตามช่วงเวลาของ 8088 CPU และส่งข้อมูลเข้าไปเขียนยังแผ่นดิสก์ผ่านทางหัวแม่เหล็ก ข้อมูลจะถูกเขียนผ่านไปทางหัวอ่านในเชกเตอร์ที่ถูกกำหนดเมื่อช่องข้อมูล (Data Field) เต็ม 765 FDC จะคำนวณค่า CRC ของข้อมูลและทำการบันทึกค่า CRC ขนาด 2 ไบต์ ต่อจากชุดข้อมูล FDC จะทำการอ่าน ID Field ของเชกเตอร์ และทำการเปรียบเทียบกับค่า CRC หากพบความผิดพลาดและ Error Flag (บิตที่ 5) ถูกเซตขึ้นในรีจิสเตอร์สถานะ จะยุติการทำงานของคำสั่งการเขียนทันที การส่งถ่ายข้อมูลระหว่าง CPU กับ FDC จะเกิดขึ้นทุกๆ 15 μ s



รูปที่ 2.21 วงจรเขียนข้อมูลลงในแผ่นดิสก์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับภาควิชางานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 เมื่อเข้าสู่ช่วงการประมวลผลเริ่มต้นและ FDC เริ่มสั่งการโหลดหัวอ่าน รอประมาณ 35 ms
 ในการโหลดหัวอ่าน หลังจากนั้นเริ่มการอ่านเชกเตอร์ ID เมื่อตรงตามที่ได้จากการกำหนดของ

คำสั่ง FDC จะอ่านข้อมูลที่จะทำการเขียนตามช่วงเวลาของ 8088 ซีพียู และส่งข้อมูลเข้าไปเขียนยังแผ่นดิสก์ผ่านทางหัวแม่เหล็ก ข้อมูลจะถูกเขียนผ่านไปทางหัวอ่านในเซกเตอร์ ที่ถูกกำหนดเมื่อช่องข้อมูลเต็ม 765 FDC จะคำนวณค่า CRC ของข้อมูลและทำการบันทึกค่า CRC ขนาด 2 ไบต์ ต่อจากชุดข้อมูล FDC จะทำการอ่าน ID Field ของเซกเตอร์ และทำการเปรียบเทียบกับค่า CRC หากพบความผิดพลาดและ Error Flag (บิตที่ 5) ถูกเซตขึ้นในรีจิสเตอร์สถานะจะยุติการทำงานของคำสั่งการเขียนทันที การส่งถ่ายข้อมูลระหว่าง ซีพียู กับ FDC จะเกิดขึ้นทุกๆ $15\ \mu\text{s}$

ในระหว่างการหาส่วนที่จะทำการเขียน FDC ขา 38 จะเป็นตัวกำหนดทิศทางของการเคลื่อนที่ของหัวอ่านโดยหากเป็นลอจิก High จะทำให้หัวเคลื่อนเข้า และลอจิก Low เพื่อเริ่มทำการเขียนข้อมูลจากแทร็คด้านในของดิสก์

FDC ขา 31 และ 32 จะใช้ในการสร้างวงจรชดเชยโดยใช้ 74LS153 U10 และ 74LS175 U11 ต่อร่วมกับสัญญาณนาฬิกา และสัญญาณบอกจังหวะการเขียนข้อมูล เพื่อในกรณีที่อ่านข้อมูลกลับออกมาจะสามารถที่จะแยกส่วนของข้อมูลและส่วนที่เป็นสัญญาณนาฬิกาออกมาได้ สัญญาณนาฬิกาบอกการเขียนข้อมูล (WCK) จะรับเข้าที่ขา 21 ของ 765 U6 (รูปที่ 2.17) จะสร้างสัญญาณนาฬิกาขนาด 250 ns ความถี่ 1MHz ในระหว่างที่ติดสอยอยู่ในช่วงการเขียนข้อมูลจะเร็วปกติหรือช้า นั้นจะขึ้นอยู่กับสัญญาณการเขียนข้อมูลนั่นเอง

ในระหว่างขั้นตอนการเขียนนี้ U6 ขา 37 จะแอคทีฟ โดยเมื่อเริ่มต้นเขียนสัญญาณ Fault Reset จะถูกส่งไปทำการรีเซต Fault flip-flop ซึ่งอยู่ในตัว Disk Drive เมื่อมีการค้นหาจุดเริ่มต้น ขา 37 จะทำหน้าที่เป็นสเตปพัลส์เพื่อเคลื่อนที่ตำแหน่งของหัวอ่านให้ไปในตำแหน่งแทร็คที่ต้องการได้ ระหว่างการค้นหา ขา 39 จะแอคทีฟ High สัญญาณที่ออกจากขา 37 และ 39 จะนำมา AND กัน โดย 74LS08 U4 เพื่อใช้ในการหาจำนวนสเตปพัลส์ที่จะส่งเข้าไปทำการค้นหา สัญญาณสเตปพัลส์จะส่งผ่าน 74LS38 NAND Gate U9 เพื่อสร้างสัญญาณแอคทีฟ Low (Step A&B) Pulse

FDC ขา 27 จะกำหนดการเลือกหัวอ่านโดยเมื่อเป็น High จะเป็นการเลือกหัวอ่าน 1 สัญญาณนี้จะส่งไป Invert โดย 74LS38 NAND U9 เพื่อสร้างสัญญาณแอคทีฟ Low ออกทางขา 11 ของ U9

ขา 25 ของ U6 จะส่งสัญญาณ Write Enable โดยผ่าน 3487 U7 สร้างสัญญาณแอคทีฟ Low สัญญาณจะส่งผ่านทางขา 13 และขา 11 ตามลำดับ

2.4 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ผลิตโดยบริษัทอินเทล มีการนำมาใช้ประโยชน์กันอย่างมากมายในปี ค.ศ. 1980 ต่อมาบริษัท Phillips และ Siemens ได้รับลิขสิทธิ์ในการผลิตจำหน่าย และได้มีการเพิ่มประสิทธิภาพมากขึ้น ทำให้ในปัจจุบันมีไมโครคอนโทรลเลอร์ที่มีมาตรฐานมาจาก MCS-51 ของบริษัทอินเทลเป็นจำนวนมาก

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วย ไมโครคอนโทรลเลอร์หลายรุ่น ซึ่งมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน ต่างกันเพียงขนาดของหน่วยความจำภายในและหน่วยความจำภายนอกเท่านั้น ในการใช้งานสามารถเลือกใช้ได้ตามความต้องการและความเหมาะสม

2.4.1 คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51

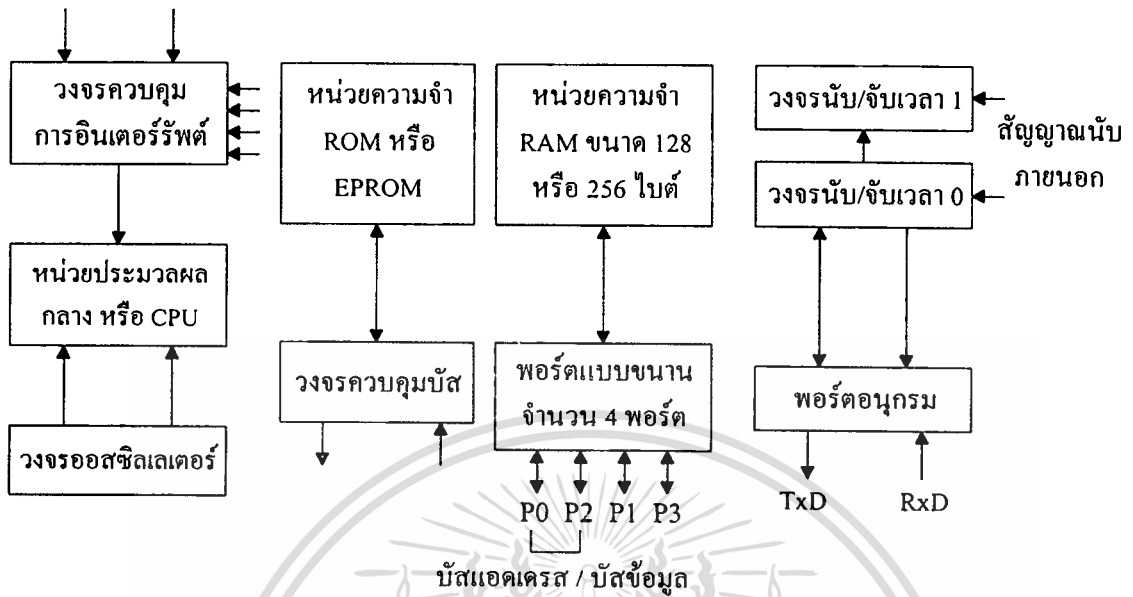
1. หน่วยความจำประมวลผลกลางขนาด 8 บิต
2. หน่วยความจำโปรแกรมภายใน (Program Memory) ขนาด 4 กิโลไบต์
3. หน่วยความจำข้อมูลภายใน (Data Memory) ขนาด 128 ไบต์
4. อีแอดแอดแอดของหน่วยความจำโปรแกรมได้ถึง 64 กิโลไบต์
5. อีแอดแอดแอดของหน่วยความจำข้อมูลได้ถึง 64 กิโลไบต์
6. หน่วยความจำโปรแกรมและข้อมูลที่อยู่ภายนอกชิพ แยกจากกันอย่างละ 64 กิโลไบต์
7. มีพอร์ตอินพุตและเอาต์พุต แบบขนานจำนวน 4 พอร์ต (32 บิต) แยกกันอย่างอิสระ
8. มีวงจรมีและจับเวลา ขนาด 16 บิต 2 ชุด ทำงานได้ 4 โหมด
9. มีพอร์ตการสื่อสารอนุกรม (Universal Asynchronous Receiver Transmitter : UART) รับส่งข้อมูลได้ในเวลาเดียวกัน (Full Duplex) สามารถเลือกรูปแบบการส่งได้ 4 โหมด
10. รับสัญญาณอินเทอร์รัพต์ได้ 6 แหล่ง กระโดดไปตอบสนองการทำงานได้ 5 ตำแหน่ง
11. มีวงจรรอสวิตช์เลเตอร์ภายใน
12. นำข้อมูลมา AND, OR หรือทำ Complement ได้ทั้งแบบ 8 บิตและ 1 บิต

2.4.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์

ภายในไมโครคอนโทรลเลอร์ จะประกอบขึ้นด้วยเกตชนิดต่างๆ เช่น AND, OR, NOT ซึ่งเกตเหล่านี้จะนำเอาแม้ออกแบบให้มีหน้าที่การทำงานต่างๆ เช่น วงจรลอกรหัส วงจรสร้างสัญญาณนาฬิกา เป็นต้น หน่วยการทำงานพื้นฐานของไมโครคอนโทรลเลอร์เบอร์ต่างๆ แสดงดังรูปที่ 2.22

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณอินเทอร์รัพต์ภายนอก



รูปที่ 2.22 หน่วยการทำงานพื้นฐานของ MCS-51

โครงสร้างภายในของไมโครคอนโทรลเลอร์ ประกอบด้วย 3 ส่วนหลักๆ ดังนี้

1) CPU (Central Processing Unit) ทำหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่นๆ เรียกว่า วงจรควบคุม (Control Unit) สัญญาณที่สร้างจากวงจรควบคุม ได้แก่ สัญญาณสำหรับการติดต่อกับหน่วยความจำ, อุปกรณ์รับข้อมูลเข้าหรือส่งข้อมูลออก ซึ่งส่วนควบคุมการขัดจังหวะและส่วนควบคุมบัสก็เป็นส่วนหนึ่งของวงจรควบคุมด้วย การสร้างสัญญาณจากวงจรควบคุมจากซีพียูจะทำการสร้างสัญญาณโดยการถอดรหัสจากคำสั่งที่กำหนดไว้ และสัญญาณที่สร้างขึ้นมาจะอ้างอิงกับสัญญาณนาฬิกาที่สร้างจากวงจรออสซิลเลเตอร์เพื่อให้ทุกๆ ส่วนทำงานประสานกันอย่างถูกต้อง

ในซีพียูประกอบด้วย Arithmetic Logic Unit : ALU ทำหน้าที่ประมวลผลข้อมูล เช่น การบวก, ลบ, คูณ, หรือหารข้อมูล แล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำที่ต้องการ

2) หน่วยความจำ (Memory) มีไว้สำหรับจัดจำข้อมูล ซึ่งในการนำข้อมูลเข้าและออกจากหน่วยความจำ จำเป็นต้องรู้ตำแหน่งของหน่วยความจำ (Address) ในการนำข้อมูลเข้าไปเก็บในหน่วยความจำ เรียกว่า การเขียนข้อมูล และการนำข้อมูลออกจากหน่วยความจำ เรียกว่า การอ่านข้อมูล ในไมโครคอนโทรลเลอร์ MCS-51 ข้อมูลในแต่ละตำแหน่งจะมีขนาด 8 บิต ดังนั้นแต่ละตำแหน่งของหน่วยความจำจะสามารถเก็บข้อมูลที่มีค่าได้ระหว่าง 00000000₂ หรือ 11111111₂ หรือการค่า 00H ถึง 0FFH ในการติดต่อกับหน่วยความจำจะต้องมีสัญญาณ 3 กลุ่ม คือองเอกสารทุกครั้งที่มีการนำไปใช้

2.1) ตำแหน่งที่ต้องการติดต่อกับหน่วยความจำ ซึ่ง MCS-51 สามารถติดต่อกับหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลได้สูงสุดชนิดละ 65,536 ตำแหน่ง (64 Kbytes) ดังนั้นการอ้างตำแหน่งของหน่วยความจำจะต้องใช้เส้นแสดงตำแหน่งในเลขฐาน 2 ทั้งหมด 16 เส้น (2^{16} เท่ากับ 65,536)

2.2) ข้อมูลที่อ่านหรือเขียนกับหน่วยความจำในตำแหน่งที่เราต้องการ

2.3) สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำ เพื่อบอกกับหน่วยความจำว่าต้องการอ่านข้อมูลหรือเขียนข้อมูลโดยวงจรลอจิกคำสั่งจะทำการสร้างสัญญาณควบคุมจากคำสั่งที่อ่านเข้ามาจากหน่วยความจำโปรแกรม

3) อุปกรณ์อินพุต/เอาต์พุต (Input/Output Device) เป็นส่วนที่ใช้ส่งข้อมูลเข้าหรือนำข้อมูลออกจาก MCS-51 ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกได้ อุปกรณ์อินพุต/เอาต์พุตได้แก่ 4 I/O Port, Timer/Counter 0, Timer/Counter 1, Serial Port

3.1) 4 I/O Port หรือพอร์ตแบบขนาน เป็นที่ใช้สำหรับใช้รับส่งข้อมูลซึ่งเป็นสัญญาณดิจิทัลเข้าหรือออกจากตัว MCS-51 มีทั้งหมด 4 พอร์ต โดยแต่ละพอร์ตจะรับส่งข้อมูลได้ 8 บิต มีพอร์ต P0, P1, P2 และ P3 บางพอร์ตจะใช้งานมากกว่า 1 อย่างก็ได้

3.2) Timer/Counter 0 และ Timer/Counter 1 เป็นวงจรนับที่สามารถทำการนับจำนวนไจเคลตของสัญญาณที่ต่อจากภายนอก MCS-51 หรือจำนวนของสัญญาณนาฬิกาภายใน MCS-51 ก็ได้ สามารถตั้งค่าเริ่มต้นของการนับและการอ่านค่าการนับได้โดยซีพียู

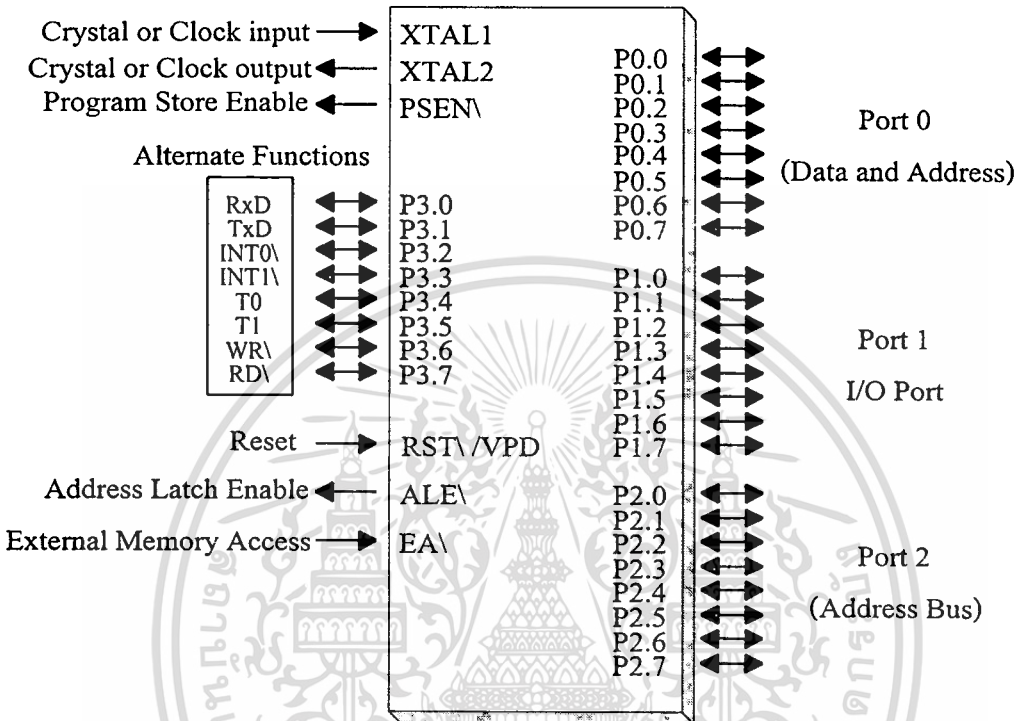
3.3) Serial Port หรือพอร์ตอนุกรม ซีพียูจะอ่านและเขียนข้อมูลกับพอร์ตอนุกรมเป็นแบบ 8 บิต แต่ข้อมูลจะถูกส่งออกจาก MCS-51 เรียงไปทีละบิต ออกจากขา TXD และในการรับข้อมูลก็จะรับทีละบิตทางขา RXD แล้วจัดเรียงใหม่เป็น 8 บิต เพื่อให้ซีพียูอ่านไปใช้งานต่อไป

ใน MCS-51 มีพอร์ตให้ใช้งานได้หลายแบบทำให้สะดวกแก่การนำไปใช้งานต่างๆ ได้มากมาย ในการจะนำพอร์ตไปใช้งานจะต้องทำการเขียนโปรแกรมมาควบคุม

2.4.3 การทำงานของ 8051

คำสั่งแต่ละคำสั่งของ 8051 จะใช้เวลาทำงาน 1, 2 หรือ 3 Machine Cycle แล้วแต่ว่าเป็นคำสั่งประเภทใด 1 Machine Cycle จะใช้เวลา 12 ไจเคลตของสัญญาณนาฬิกา ดังนั้นแต่ละคำสั่งของ 8051 จะใช้เวลาทำงาน 12, 24 หรือ 36 ไจเคลตของสัญญาณนาฬิกานั้นเอง แต่ Machine Cycle จะถูกแบ่งออกเป็น 6 สภาวะ คือ S1, S2, S3, S4, S5 และ S6 แต่ละสภาวะจะประกอบด้วย 2 ไจเคลตของสัญญาณนาฬิกา ในไจเคลตแรกจะเรียกว่าเฟส 1 (P1) และไจเคลตที่ 2 เรียกว่าเฟส 2 (P2) ในแต่ละเฟสจะนับตั้งแต่ขอบล่างของสัญญาณนาฬิกาถึงขอบล่างของสัญญาณนาฬิกาที่อยู่ถัดไป เมื่อ 8051 ทำงานเสร็จ 1 Machine Cycle จะเริ่มทำงาน State 1 Phase 1 (S1P1) ของไจเคลตต่อไปใน 1 Machine

Cycle วงจร Timer and Control จะสร้างสัญญาณ ALE ออกมา 2 ไชเกิดเพื่อ Fetch คำสั่งเข้าไป 2 ครั้งเสมอ



รูปที่ 2.23 การกำหนดหน้าที่ขาสัญญาณของไอซี 8051

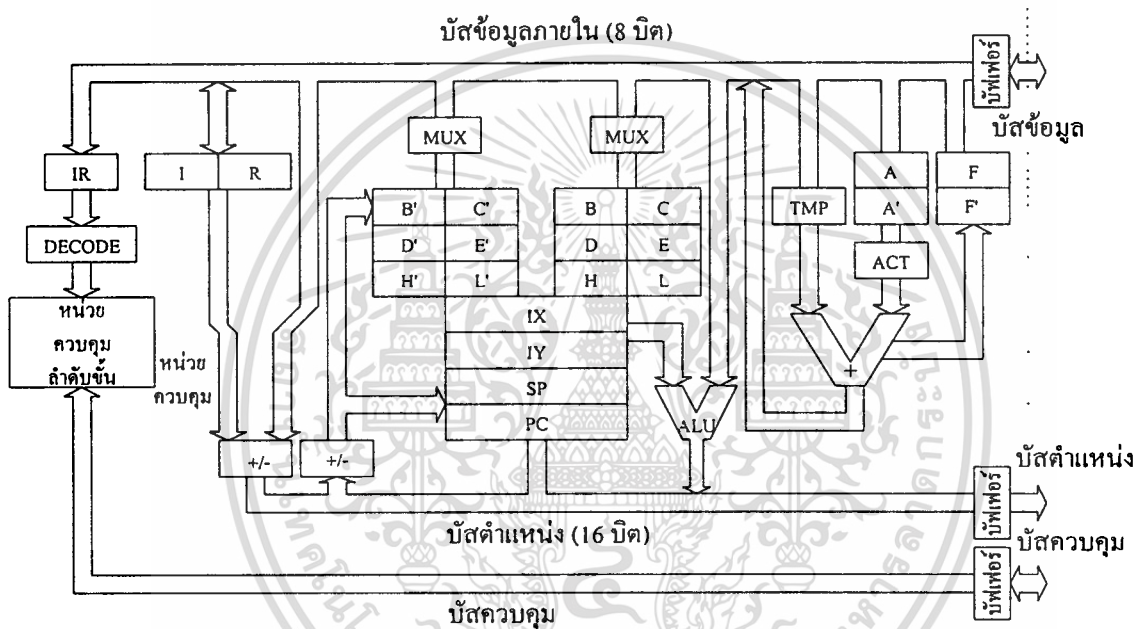
2.5 ไมโครโปรเซสเซอร์ Z-80

ไมโครโปรเซสเซอร์เบอร์ Z-80 เป็นไมโครโปรเซสเซอร์ขนาด 8 บิต ที่ได้พัฒนามาจากไมโครโปรเซสเซอร์เบอร์ 8080 ของบริษัทอินเทล โดยได้ทำการแก้ไขข้อบกพร่องบางอย่างของไมโครโปรเซสเซอร์เบอร์ 8080 เช่น ทำให้มีคำสั่งมากขึ้นวิธีการอ้างถึงตำแหน่งข้อมูล (Addressing Mode) ใหม่ มีระบบฮาร์ดแวร์ที่มีความสามารถ มีความสะดวกในการใช้งานมากขึ้น และซีพียู Z-80 ยังสามารถใช้ซอฟต์แวร์ของระบบที่ใช้กับซีพียู 8080 ได้อีกด้วย

2.5.1 โครงสร้างซีพียู Z-80

โครงสร้างของซีพียู Z-80 และรีจิสเตอร์ต่างๆ ที่จำเป็นในการเขียนโปรแกรมเพื่อให้ไมโครโปรเซสเซอร์ทำงานตามที่ต้องการ แสดงดังรูปที่ 2.24 จากรูปเห็นได้ว่าซีพียูมีบัสอยู่ 3 ชนิด คือ บัสตำแหน่งบัสข้อมูล และบัสควบคุม ซึ่งบัสเหล่านี้ใช้สำหรับการเชื่อมต่อซีพียูกับอุปกรณ์การคำนวณ ไม่ว่าจะเป็นใด ๆ ทั้งสิ้น อีกทั้งยังมีให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภายนอกและภายใน ไมโครโปรเซสเซอร์ประกอบด้วยวงจรพื้นฐานคือ ALU แอคคิวเมเตอร์ รีจิสเตอร์ตำแหน่ง และแฟลก นอกจากนี้ยังมีรีจิสเตอร์ใช้งานต่างๆ อีกคือ B, C, D, E, H, L, B', C', E', H' และ L' ส่วนรีจิสเตอร์ตำแหน่งคือ PC, SP, IX และ IY ซึ่งรีจิสเตอร์ต่างๆ เหล่านี้ จะกล่าวถึง โดยละเอียดต่อไป ทางด้านซ้ายสุดคือส่วนของหน่วยควบคุม ซึ่งส่วนนี้มีหน้าที่ถอดรหัสคำสั่งแล้ว ส่งสัญญาณควบคุมไปตามส่วนต่างๆ ทั้งภายในซีพียูและภายนอกซีพียู รวมทั้งรับสัญญาณควบคุม จากภายนอกเข้ามาด้วย



รูปที่ 2.24 โครงสร้างภายในของซีพียู Z-80

สัญญาณควบคุมซีพียูของ Z-80 ภายนอกมี 13 สัญญาณ ซึ่งแบ่งได้ 2 อย่าง คือ สัญญาณควบคุมซีพียู และสัญญาณควบคุมระบบ (CPU and System Control) ซึ่งสัญญาณต่างๆ ทำให้มีผลต่อซีพียู และมีผลต่อระบบไมโครคอมพิวเตอร์ด้วย บัสข้อมูลเป็นบัสที่มีขนาด 8 บิต ใช้เป็นทางเดินของข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำ หรืออุปกรณ์อินพุต, เอาต์พุตต่างๆ บัสตำแหน่งเป็นบัสที่มีขนาด 16 บิต เพื่อใช้ในการอ้างอิงถึงตำแหน่งของหน่วยความจำ ดังนั้นทำให้สามารถอ้างอิงถึงหน่วยความจำภายนอกได้ 2 ตำแหน่ง (65536 ตำแหน่ง หรือ 64k) คือตั้งแต่ตำแหน่งที่ 0-65535 บล็อกที่มีเครื่องหมาย +, - ที่อยู่ทางด้านซ้ายของรีจิสเตอร์ หมายถึงการเพิ่มหรือลดข้อมูล (Increment Decrement) ที่มีอยู่ในรีจิสเตอร์ตำแหน่งหรือคู่ของรีจิสเตอร์ต่างๆ คือ SP, PC, BC, DE, HL ซึ่งรีจิสเตอร์ต่างๆ เหล่านี้เป็นรีจิสเตอร์ในการกำหนดตำแหน่งโดยตรง (Pure Address Register) ใช้

ของการอ้างถึงตำแหน่งของหน่วยความจำแบบตรง (Direct Address Mode) ซีพียู Z-80 มีคำสั่งที่เกี่ยวกับอินพุต, เอาต์พุตโดยเฉพาะ ไม่ใช่ลักษณะการอินพุต, เอาต์พุตแบบ Memory-Mapped (Memory-Map I/O ใช้ส่วนหนึ่งของหน่วยความจำเพื่อเป็นตำแหน่งของอุปกรณ์อินพุต, เอาต์พุต) ในการทำให้ไมโครโปรเซสเซอร์ทำงานตามที่ต้องการทำได้โดยการเขียนโปรแกรมเก็บไว้ในหน่วยความจำ จากนั้นให้ไมโครโปรเซสเซอร์อ่านคำสั่งมาจากหน่วยความจำเพื่อมาปฏิบัติ การปฏิบัติคำสั่งต่างๆ นั้นไม่จำเป็นต้องเข้าถึงส่วนรายละเอียดต่างๆ ของไมโครโปรเซสเซอร์ทั้งหมด โดยจะสนใจเฉพาะรีจิสเตอร์ต่างๆ ที่เกี่ยวข้องกับการเขียนโปรแกรมเท่านั้น รีจิสเตอร์ภายในที่สามารถอ่านหรือเขียนได้มีถึง 208 บิต โดยแยกเป็นกลุ่มของรีจิสเตอร์ขนาด 8 บิต 18 รีจิสเตอร์ และรีจิสเตอร์ขนาด 16 บิต อีก 4 รีจิสเตอร์ รีจิสเตอร์ต่างๆ ภายใน Z-80 เป็นลักษณะของสแตคแรม และรีจิสเตอร์เหล่านี้แบ่งออกเป็น 3 ประเภท คือ

- 1) รีจิสเตอร์ใช้งานทั่วไป
- 2) แอคคิวมูเลเตอร์และรีจิสเตอร์สถานะ
- 3) รีจิสเตอร์ใช้งานเฉพาะอย่าง

การจัดรีจิสเตอร์ภายในของซีพียู Z-80 แสดงดังรูปที่ 2.25

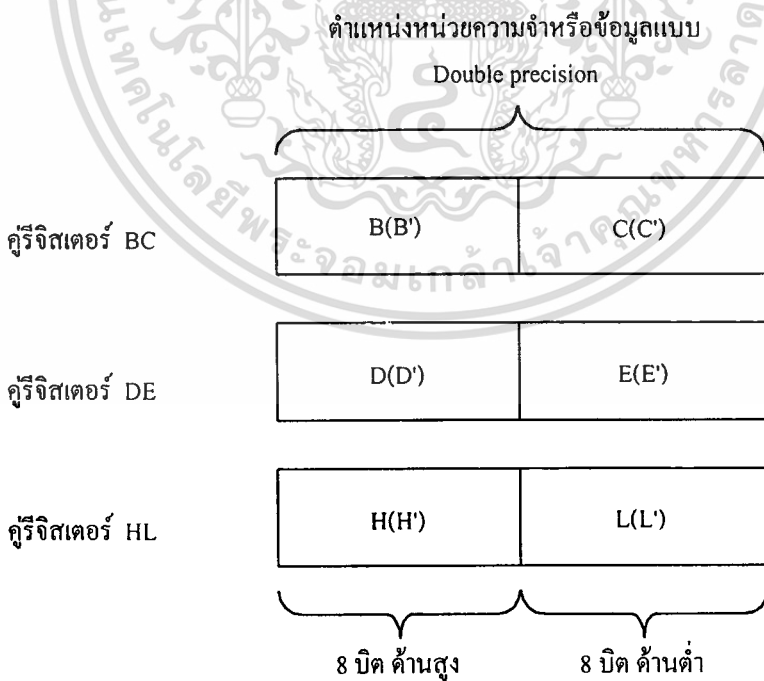
รีจิสเตอร์หลัก		รีจิสเตอร์สำรอง		รีจิสเตอร์ใช้งาน ทั่วไป
แอคคิวมูเลเตอร์	แฟลก	แอคคิวมูเลเตอร์	แฟลก	
A	F	A'	F'	
B	C	B'	C'	
D	E	D'	E'	
H	L	H'	L'	

อินเตอร์รัพท์เวกเตอร์	เมมโมรีรีเฟรช	รีจิสเตอร์ใช้งาน เฉพาะอย่าง
I	R	
อินเด็กซ์รีจิสเตอร์ IX		
อินเด็กซ์รีจิสเตอร์ IY		
สแตคพอยน์เตอร์		
โปรแกรมเคาน์เตอร์		

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้นำเอกสารนี้ไปเผยแพร่หรือใช้ซ้ำโดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

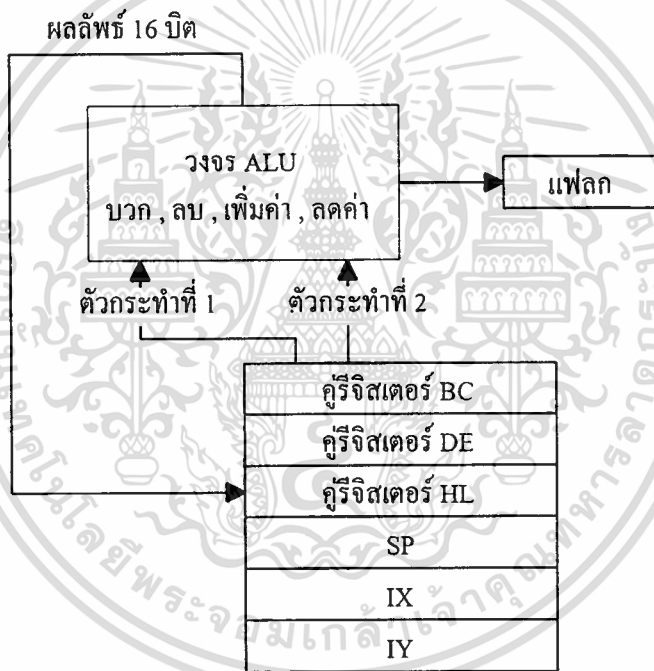
2.5.2 รีจิสเตอร์ใช้งานทั่วไป (General Purpose Register)

รีจิสเตอร์ใช้งานทั่วไปในซีพียู Z-80 มี 2 กลุ่มที่เหมือนกัน แต่ละกลุ่มประกอบด้วย รีจิสเตอร์ขนาด 8 บิต จำนวน 6 รีจิสเตอร์ กลุ่มหนึ่งคือกลุ่มของรีจิสเตอร์หลัก (Main Register Set) ซึ่งประกอบด้วย B, C, D, E, H, L ส่วนอีกกลุ่มหนึ่งคือชุดของรีจิสเตอร์สำรอง (Alternate Register Set) ซึ่งประกอบด้วย B', C', D', E', H', L' ประโยชน์ของการมีรีจิสเตอร์ 2 ชุดนี้ คือ เพื่อให้ผู้เขียนโปรแกรมสามารถสลับระหว่างข้อมูลในชุดของรีจิสเตอร์ทั้งสองได้อย่างรวดเร็ว ทั้งนี้เพื่อความเร็วในการทำงานของโปรแกรม เนื่องจากการเข้าถึงข้อมูลของโปรแกรมภายในรีจิสเตอร์ของซีพียูมีความรวดเร็วกว่าการเข้าถึงข้อมูลในหน่วยความจำภายนอก นอกจากนี้รีจิสเตอร์สำรองอาจใช้เพื่อเก็บข้อมูลในรีจิสเตอร์หลัก เมื่อซีพียูถูกอินเตอร์รัพต์จากภายนอก รีจิสเตอร์ใช้งานทั่วไปนี้ อาจใช้เป็นที่เก็บข้อมูลขนาด 8 บิตแบบธรรมดา หรือใช้เก็บข้อมูลขนาด 16 บิตโดยการจับคู่กัน (Pair) ระหว่างรีจิสเตอร์ก็ได้ ซึ่งคู่รีจิสเตอร์ของกลุ่มรีจิสเตอร์หลัก คือ BC, DE, HL ส่วนคู่รีจิสเตอร์อีกชุดหนึ่งคือ BC', DE' และ HL' ประโยชน์ของคู่รีจิสเตอร์ เช่น ใช้เป็นตัวกำหนดตำแหน่ง (Pointer) ของข้อมูลในหน่วยความจำ หรือใช้เพื่อกระทำทางคณิตศาสตร์แบบ 16 บิตที่เรียกว่า Double-Precision Arithmetic



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับ... **รูปที่ 2.26 การจับคูรีจิสเตอร์** ... ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การกระทำทางคณิตศาสตร์แบบ Double-Precision หมายถึง การบวก การลบเลข เพิ่มค่าขึ้น 1 (Increment) การลดค่าลง 1 (Decrement) ของข้อมูลขนาด 16 บิต ที่อยู่ในรีจิสเตอร์ หรืออยู่ในรีจิสเตอร์แบบ 16 บิต ซึ่งการกระทำทางคณิตศาสตร์หรือลอจิกส่วนมากของซีพียู Z-80 เป็นการกระทำของข้อมูลขนาด 8 บิต ซึ่งเรียกว่า Single-Precision ซึ่งการกระทำทางคณิตศาสตร์ขนาด 16 บิตของ Z-80 นี้ทำให้สามารถเปลี่ยนแปลงตัวกำหนดตำแหน่งได้สะดวก โดยการเปลี่ยนแปลงข้อมูลภายในรีจิสเตอร์ และนอกจากนี้ยังเห็นได้ว่าถ้าไม่มีการบวกลบเลขแบบ 16 บิต จะต้องทำการบวกลบเลขแบบ 8 บิตถึง 2 ครั้งซึ่งจะทำให้การเขียนโปรแกรมยุ่งยากขึ้น รูปที่ 2.27 แสดงถึงการใช้รีจิสเตอร์เพื่อการกระทำแบบ Double-Precision



รูปที่ 2.27 การกระทำทางคณิตศาสตร์แบบ double precision

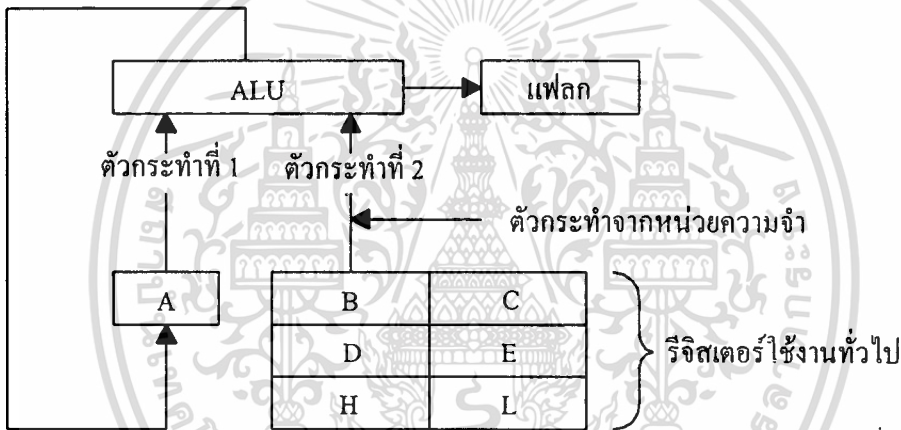
2.5.3 แอคคิวมูเลเตอร์และรีจิสเตอร์สถานะ (Accumulator and Flag Register)

จากโครงสร้างของซีพียูเห็นได้ว่าโปรเซสเซอร์ทั่วไปจะต้องมีแอคคิวมูเลเตอร์เพื่อเป็นที่เก็บข้อมูลตัวกระทำหลัก (Main Operand) ตัวหนึ่ง เพื่อที่จะทำทางคณิตศาสตร์หรือลอจิกกับข้อมูลที่มาจากรีจิสเตอร์อื่นๆ หรือข้อมูลที่มาจากหน่วยความจำภายนอก และผลจากการกระทำนั้นก็จะถูกนำมาเก็บไว้ในแอคคิวมูเลเตอร์ ส่วนรีจิสเตอร์สถานะหรือแฟล็ก ใช้เพื่อแสดงสถานะของแอคคิวมูเลเตอร์หรือรีจิสเตอร์ที่เกี่ยวข้อง การกระทำนี้แสดงได้ดังรูปที่ 2.28 ไมโครโปรเซสเซอร์ Z-80 มีรีจิสเตอร์ที่ทำหน้าที่เก็บข้อมูลแอคคิวมูเลเตอร์ 2 ตัวและแฟล็ก 2 ตัว แต่มีเพียงอย่างละ

ตัวเดียวเท่านั้นที่สามารถทำงานกับหน่วยคณิตศาสตร์และลอจิกได้ คือ แอคคิวมูเลเตอร์และแฟลกที่อยู่ในส่วนของรีจิสเตอร์หลัก ซึ่งเรียกอ่าว่า A และ F ส่วนรีจิสเตอร์ A' และ F' ที่อยู่ในรีจิสเตอร์สำรองใช้เพื่อเป็นที่เก็บข้อมูลชั่วคราวของแอคคิวมูเลเตอร์ A และแฟลก F เมื่อถึงคราวจำเป็นเท่านั้น ซึ่งคู่ของแอคคิวมูเลเตอร์และแฟลกทั้งสองชุดนี้ คือ AF และ AF' สามารถสลับข้อมูลที่อยู่ในรีจิสเตอร์ทั้งคู่ระหว่างกันได้ โดยใช้คำสั่งในกลุ่มการแลกเปลี่ยนข้อมูล (Exchange group) เหมือนกับการแลกเปลี่ยนข้อมูลระหว่างรีจิสเตอร์หลักและรีจิสเตอร์สำรองของกลุ่มรีจิสเตอร์ใช้งานทั่วไป

ผลลัพธ์ 8 บิต จากการกระทำของ

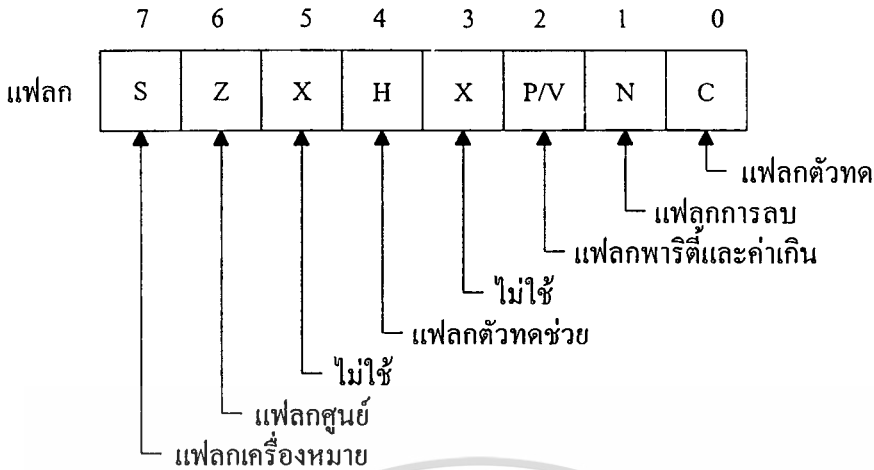
คณิตศาสตร์หรือลอจิก



รูปที่ 2.28 การทำงานทางคณิตศาสตร์และลอจิก

รีจิสเตอร์สถานะหรือแฟลกเป็นรีจิสเตอร์ขนาด 8 บิต ซึ่งแต่ละบิตใช้แสดงสถานะของซีพียูที่จะเกิดขึ้นหลังจากที่ทำคำสั่งทางคณิตศาสตร์ คำสั่งทางลอจิกหรือการทำคำสั่งอื่นๆ ของซีพียูบิตต่างๆ ของแฟลกใช้แสดงสถานะที่แตกต่างกัน ซึ่งรูปแบบของรีจิสเตอร์นี้แสดงดังรูปที่ 2.29

บิต 3 และบิต 5 ของแฟลกไม่ได้ถูกใช้เพื่อแสดงสถานะ ดังนั้นค่าในบิตทั้ง 2 นี้จะเป็นอะไรก็ได้ ข้อมูลที่บิตต่างๆ ของแฟลกอาจเป็น 0 หรือ 1 ก็ได้แล้วแต่ผลของการกระทำคำสั่งต่างๆ ของซีพียู กลุ่มคำสั่งที่มีผลต่อแฟลก เช่น คำสั่งในกลุ่มของการกระทำ ทางคณิตศาสตร์และลอจิก กลุ่มของการเลื่อนและหมุนข้อมูล กลุ่มของการทดสอบบิต เป็นต้น และในการทำงานข้าม (Jump or Branch) แบบมีเงื่อนไข ซีพียูก็จะทำการทดสอบข้อมูลที่บิตต่างๆ ของแฟลกนี้ด้วยว่าเป็นไปตามเงื่อนไขที่ต้องการหรือไม่ รายละเอียดของแฟลกต่างๆ จะอธิบายได้ดังต่อไปนี้ ให้นำไปใช้ประโยชน์ด้านการคำนวณ



รูปที่ 2.29 รูปแบบของแฟลก

บิต 3 และบิต 5 ของแฟลกไม่ได้ถูกใช้เพื่อแสดงสถานะ ดังนั้นค่าในบิตทั้ง 2 นี้จะเป็นอะไรก็ได้ ข้อมูลที่บิตต่างๆ ของแฟลกอาจเป็น 0 หรือ 1 ก็ได้แล้วแต่ผลของการกระทำคำสั่งต่างๆ ของซีพียู กลุ่มคำสั่งที่มีผลต่อแฟลก เช่น คำสั่งในกลุ่มของการกระทำทางคณิตศาสตร์และลอจิก กลุ่มของการเลื่อนและหมุนข้อมูล กลุ่มของการทดสอบบิต เป็นต้น และในการทำงานข้าม (Jump or Branch) แบบมีเงื่อนไข ซีพียูก็จะทำการทดสอบข้อมูลที่บิตต่างๆ ของแฟลกนี้ด้วยว่าเป็นไปตามเงื่อนไขที่ต้องการหรือไม่ รายละเอียดของแฟลกต่างๆ จะอธิบายได้ดังต่อไปนี้

1) แฟลกตัวทศ (Carry Flag)

แฟลกตัวทศหรือแฟลก C แฟลกนี้จะเซตหรือรีเซตขึ้นอยู่กับผลการทำงานของคำสั่งกลุ่มต่างๆ เช่น กลุ่มคณิตศาสตร์ กลุ่มของการเลื่อนหมุนข้อมูล ตัวอย่างเช่นการบวกเลขฐานสอง 8 บิตสองจำนวน หรือการลบเลขฐานสอง 8 บิตจำนวน ดังรูปที่ 2.30

$$\begin{array}{rcl}
 & 01101011 & 01001010 \\
 + & 11011110 & + 00111101 \\
 \hline
 C = [1] & 01001001 & C = [0] 10001111 \\
 \text{(ก)} & & \text{(ข)}
 \end{array}$$

$\begin{array}{r} 01101011 \\ - 00101001 \\ \hline C = [1] 01000010 \end{array}$ (ค)	$\begin{array}{r} 01011010 \\ - 10001100 \\ \hline C = [1] 11001001 \end{array}$ (ง)
---	---

รูปที่ 2.30 (ก), (ข) ผลของแฟลก C หลังจากทำคำสั่งบวก
(ค), (ง) ผลของแฟลก C หลังจากทำคำสั่งลบ

จากรูปที่ 2.30 เห็นได้ว่าการบวกได้ผลลัพธ์เกิน 8 บิต ดังนั้นบิตที่เกินมาเป็นบิตที่ 9 จะทำให้แฟลก C เซตเป็น 1 ส่วนรูปที่ 2.30 (ข) ผลของการบวกไม่เกิน 8 บิต แฟลก C รีเซตเป็น 0 ในทำนองเดียวกันเมื่อทำคำสั่งลบ แฟลก C นี้จะเป็นแฟลกที่แสดงการขอยืม ดังรูปที่ 2.30 (ง) เมื่อการลบต้องการขอยืมแฟลก C จะเซตเป็น 1 แต่ถ้าไม่มีการขอยืมแฟลก C นี้ก็จะรีเซตเป็น 0 แต่ถ้าซีพียูทำคำสั่งกลุ่มลอจิกไม่ว่าคำสั่งใดก็ตาม แฟลก C นี้จะรีเซตเป็น 0 เสมอ

2) แฟลกการลบ (Negative Flag)

แฟลกการลบจะเซตหรือรีเซตขึ้นอยู่กับคำสั่งที่ใช้เกี่ยวข้องกับทางคณิตศาสตร์ นั่นคือเมื่อทำคำสั่งประเภทการลบแฟลกนี้จะเซต เช่น เมื่อซีพียูทำคำสั่ง SUB CP DEC เป็นต้น แต่ถ้าซีพียูทำคำสั่งที่เป็นการบวก เช่น ADD INC แฟลกนี้จะรีเซต แฟลกการลบนี้มีประโยชน์มากเมื่อทำคำสั่ง DAA (Decimal Adjust Accumulator) จะทำให้ซีพียูรู้ว่าคำสั่งก่อนหน้านี้เป็นคำสั่งประเภทการบวกหรือการลบ เพื่อที่จะได้ทำการปรับด้วยการบวกหรือการลบด้วย 6 เพื่อให้ได้เลข BCD ที่ต้องการ

3) แฟลกพาริตี/ค่าเกิน (Parity/Overflow Flag)

เป็นแฟลกที่อยู่บิต 2 ของแฟลกริजิสเตอร์ แฟลกนี้มีประโยชน์สองอย่างคือ เมื่อซีพียูทำคำสั่งลอจิก แฟลกนี้จะเป็แฟลกพาริตี แต่ถ้าซีพียูทำคำสั่งคณิตศาสตร์แฟลกนี้จะเป็แฟลกค่าเกิน ในกรณีของแฟลกพาริตีบิตนี้จะเซตเป็น 1 ถ้าผลของการกระทำทางลอจิกแล้วทำให้จำนวน 1 ของแอกคิวมูลเตอร์เป็นจำนวนคู่ (Even Parity) และรีเซตเป็น 0 ถ้าจำนวน 1 ในแอกคิวมูลเตอร์เป็นจำนวนคี่ (Odd Parity) ตัวอย่างดังรูป 2.31

ในกรณีที่ทำงานเป็นแฟลกค่าเกิน (Overflow หรือ V) คือเมื่อซีพียูทำคำสั่งทางคณิตศาสตร์ แฟลก V นี้จะเซตเมื่อผลลัพธ์จากการกระทำมีค่าเกิน และรีเซตเมื่อผลลัพธ์มีค่าไม่เกิน ค่าเกินนี้พิจารณาได้จากผลลัพธ์ที่จะได้ทีแอกคิวมูลเตอร์ เนื่องจากระบบตัวเลขที่ใช้ในไมโครโปรเซสเซอร์เป็นแบบ 2's Complement Sign Binary นั่นคือ บิตสูงสุดเป็นบิตเครื่องหมายและที่เหลือเป็นขนาดของเลข ดังนั้นถ้าแอกคิวมูลเตอร์มีขนาด 8 บิต มันจะสามารถแสดงค่าเลขฐานสองได้ระหว่าง 01111111

$$\begin{array}{r}
 10110110 \\
 \text{AND } \underline{10000101} \\
 \hline
 \text{ผลลัพธ์ } 10000100 \rightarrow \text{"1" มี 2 บิต เป็นจำนวนคู่ : แฟล็ก P = 1}
 \end{array}$$

0 1 1 0 1 0 0 1
 OR 0 0 1 1 0 0 0 1
 ผลลัพธ์ 0 1 1 1 1 0 0 1 → “1” มี 5 บิต เป็นจำนวนคี่ : แฟล็ก P = 0

รูปที่ 2.31 เมื่อแฟลค P/V ทำหน้าที่เป็นพาริตีแฟลค

$$\begin{array}{rcl}
 +127 & & 01111111 \\
 +\underline{64} & & +\underline{01000000} \quad (\text{ก}) \\
 +191 & V[1] & 10111111 \\
 -125 & & 10000011 \\
 -\underline{126} & & +\underline{10000010} \quad (\text{ข}) \\
 -251 & V[1] & 00000101 \\
 \\
 +32 & & 00100000 \\
 +\underline{32} & & +\underline{00100000} \quad (\text{ค}) \\
 +64 & V[0] & 01000000
 \end{array}$$

รูปที่ 2.32 การทำงานของแฟลค V

ได้ดังนี้คือ ถ้าบิตเครื่องหมายของทั้งตัวตั้งและตัวบวกเป็น “0” (+) แล้วผลลัพธ์มีบิตเครื่องหมายเป็น

“1” (-) แสดงว่าค่าเกิน (รูปที่ 2.32 (ก)) ในทำนองเดียวกันถ้าบิตเครื่องหมายของทั้งตัวตั้งและตัวบวก เป็น “1” (-) ทั้งคู่ แล้วผลลัพธ์มีบิตเครื่องหมายเป็น “0” (+) แสดงว่าเกิดค่าเกิน นอกจากนี้แฟลก V จะรีเซ็ต ดังนั้นถ้ากำหนดให้บิตเครื่องหมายตัวตั้ง คือ As บิตเครื่องหมายตัวบวก Bs และบิตเครื่องหมายผลลัพธ์ คือ Cs ดังรูปที่ 2.33 ดังนั้นกรณีการเกิดค่าเกินสามารถเขียนเป็นสมการได้ดังนี้ คือ

$$V = A_s \cdot B_s \cdot C_s + A_s \cdot B_s \quad (2.1)$$

AS	
BS	
CS	

รูปที่ 2.33 บิตเครื่องหมายของตัวเลข

เมื่อ V คือ สถานะการเกิดค่าเกิน ถ้า V เป็นลอจิก 1 แสดงว่าเกิดค่าเกิน ถ้าเป็นลอจิก 0 แสดงว่าไม่เกิดค่าเกิน ในกรณีของการเกิดค่าเกินเราจะต้องพิจารณาแฟลก C เป็นบิตเครื่องหมาย และบิตเครื่องหมายเดิมจะกลายเป็นบิตสูงสุดของค่าตัวเลข (Magnitude)

4) แฟลกตัวทศช่วย (Half Carry Flag)

แฟลกตัวทศช่วยมีประโยชน์ในการกระทำทางคณิตศาสตร์ของตัวเลข BCD โดยที่แฟลกนี้จะเชตเมื่อมีการขอยืมของการกระทำตัวเลข 4 บิตทางด้านต่ำ (4 Least Significant Bit) ประโยชน์ของแฟลกนี้ใช้เพื่อการปรับผลลัพธ์ของตัวเลขที่เกิดจากการกระทำทางคณิตศาสตร์ของตัวเลข BCD ตัวอย่างของการเชตแฟลกตัวทศช่วยนี้แสดงได้ดังรูปที่ 2.34

	H [0]	
11 BCD	0001 0001	
+22 BCD	+0010 0010	
33 BCD	0011 0011	แฟลก H = 0

H [1]

49 BCD	0100 1001	
+ 28 BCD	+0010 1000	
71 BCD (ไม่ถูกต้อง)	0111 0001	แฟล็ก H = 1
+ 6	+0000 0110	ถ้ามีการปรับโดยคำสั่ง DAA
77 BCD	0111 0111	โดยการ +6 เข้าที่ 4 บิต ด้านต่ำ

(ข) แสดงการทำงานของแฟล็ก H ร่วมกับคำสั่ง DAA

รูปที่ 2.34 การเซตแฟล็กตัวทศช่วย

แฟล็กตัวทศช่วย และแฟล็กการลบเป็นแฟล็กเพื่อใช้ประโยชน์ส่วนใหญ่ในการกระทำทางคณิตศาสตร์ของเลข BCD และแฟล็กทั้งสองนี้ไม่ได้ถูกทดสอบโดยคำสั่งของการทำงานข้ามแบบมีเงื่อนไขเหมือนกับแฟล็กอื่นทั่วไป

5) แฟล็กศูนย์ (Zero Flag)

แฟล็กศูนย์จะเซตถ้าผลของการปฏิบัติคำสั่งแล้วทำให้ผลลัพธ์ในรีจิสเตอร์ที่อ้างถึงมีค่าเป็น 0 และรีเซตถ้าผลลัพธ์ในรีจิสเตอร์ที่อ้างถึงไม่เป็น 0 ตัวอย่างของการทำงานของแฟล็กศูนย์แสดงดังรูปที่ 2.35

```

01101001
AND  00000000
ผลลัพธ์ 00000000   แฟล็ก Z = 1 เพราะผลลัพธ์หลังจากการ AND เป็น 0

11111111
+ 00000000
C [1] 00000000   แฟล็ก Z = 1 และ แฟล็ก C = 1

```

รูปที่ 2.35 ตัวอย่างการทำงานของแฟล็ก Z

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง การนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจากสถาบันฯ ถือว่าผิดกฎหมาย

แฟล็กศูนย์นี้มีประโยชน์มากในการทำงานข้ามแบบมีเงื่อนไขโดยใช้คำสั่งต่างๆ เช่น คำสั่ง JP Z, nn JR Z, e JR NZ, nn เป็นต้น แฟล็กศูนย์โดยทั่วไปใช้ในประโยชน์ดังนี้ เช่น

1. ใช้สำหรับเปรียบเทียบเลข 2 จำนวน
2. ใช้ทดสอบบิต
3. ใช้ทดสอบการลดค่าตัวนับ

ซึ่งประโยชน์ต่างๆในการใช้งานของแฟลกศูนย์จะให้เห็นในการเขียนโปรแกรมที่ทำงานแบบวนรอบ

6) แฟลกเครื่องหมาย (Sign Flag)

แฟลกนี้ใช้ในการตรวจสอบบิตเครื่องหมาย ซึ่งเครื่องหมายบวกหรือลบนี้แสดงโดยตัวเลขที่บิต 7 ของแอดคิวมูลเตอร์ ถ้าบิต 7 เท่ากับ 0 แสดงว่าค่าเป็นบวก และถ้าบิต 7 เท่ากับ 1 แสดงว่าเป็นค่าลบ แฟลกเครื่องหมายนี้จะเซตถ้าหลังจากการปฏิบัติคำสั่งให้ผลลัพธ์ที่มีเครื่องหมายเป็นลบ และรีเซตเมื่อผลลัพธ์ที่ได้เครื่องหมายเป็นบวก

แฟลกต่างๆ จะเซตหรือรีเซตโดยผลการทำงานของคำสั่งในกลุ่มต่างๆ ดังนั้นเราสามารถทราบได้ว่าแฟลกใดจะมีผลเนื่องมาจากคำสั่งใด และคำสั่งใดทำการทดสอบที่แฟลกไหนโดยการดูได้จากตารางของชุดคำสั่งกลุ่มต่างๆ ซึ่งจะได้กล่าวถึงในเรื่องของชุดคำสั่ง

2.5.4 รีจิสเตอร์ใช้งานเฉพาะอย่าง (Special Purpose Register)

ในไมโครโปรเซสเซอร์ Z-80 มีรีจิสเตอร์ใช้งานเฉพาะอย่าง ดังที่แสดงไว้ในรูปที่ 2.25 ซึ่งประกอบด้วย PC, SP, IX, IY, I และ R ซึ่ง รีจิสเตอร์ต่างๆ เหล่านี้บางตัวได้กล่าวมาบ้างแล้ว แต่ในส่วนนี้จะได้กล่าวถึงอีกครั้งโดยอ้างถึงชิพ Z-80 โดยเฉพาะ

1) โปรแกรมเคาน์เตอร์ (PC)

เป็นรีจิสเตอร์ขนาด 16 บิต มีหน้าที่สำหรับเก็บตำแหน่งของคำสั่งต่อไปที่จะถูกเฟตช์ออกมาจากหน่วยความจำ คำสั่งของชิพ Z-80 เป็นคำสั่งที่มีความยาว 1, 2, 3 หรือ 4 ไบต์ ดังนั้นข้อมูลใน PC อาจถูกเพิ่มขึ้นครั้งละ 1, 2, 3 หรือ 4 โดยอัตโนมัติ ขึ้นอยู่กับความยาวของคำสั่งที่ถูกปฏิบัติก่อนหน้านี้ ถ้าคำสั่งที่มีความยาวต่างๆ กันถูกวางไว้เป็นลำดับดังรูปที่ 2.36 ค่าใน PC หลังจากทีคำสั่งนั้นถูกปฏิบัติไปแล้ว คือค่าที่อยู่ทางด้านขวา

จากรูปจะเห็นได้ว่าค่าของ PC จะเพิ่มขึ้นไปตามลำดับเรื่อยๆ นอกจากว่าเมื่อชิพทำการคำสั่งในกลุ่มของการทำงานข้าม (Jump) จะทำให้ค่าใน PC ไม่เปลี่ยนไปตามลำดับ แต่มันจะเปลี่ยนไปเป็นค่าตามตำแหน่งที่ต้องการกระโดดไปโดยคำสั่งทำงานข้ามนั้น และในอีกกรณีหนึ่ง คือ เมื่อชิพถูกขัดจังหวะ (Interrupt) เปลี่ยนไปแบบไม่เรียงลำดับเช่นกัน ข้อมูลใน PC นี้ไม่สามารถนำมากระทำทางคณิตศาสตร์และลอจิกได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งหน่วย	ข้อมูลใน	ข้อมูลใน PC หลัง
ความจำภายนอก	หน่วยความจำ	จากจบคำสั่ง

0100	คำสั่งที่ 1 (1 ไบต์)	0101
0101	คำสั่งที่ 2 (2 ไบต์)	0103
0103	คำสั่งที่ 3 (3 ไบต์)	0106
0106	คำสั่งที่ 4 (4 ไบต์)	010A
010A	คำสั่งที่ 5 (2 ไบต์)	010C

รูปที่ 2.36 การเพิ่มขึ้นของข้อมูลในโปรแกรมเคาน์เตอร์

2) สแตคพอยน์เตอร์ (SP)

สแตคพอยน์เตอร์ของซีพียู Z-80 เป็นรีจิสเตอร์ที่มีขนาด 16 บิต มีหน้าที่ใช้ชี้ตำแหน่งบนสุดของสแตค ซึ่งสแตคของระบบไมโครคอมพิวเตอร์ Z-80 เป็นสแตคแบบใช้หน่วยความจำภายนอก โดยใช้ส่วนหนึ่งของแรม ซึ่งหน้าที่ของสแตคนี้มีไว้ใช้เพื่อเป็นที่เก็บข้อมูลชั่วคราว ซึ่งข้อมูลนี้อาจจะมาจากรีจิสเตอร์ต่างๆ ข้อมูลที่เก็บลงในสแตคนี้อาจถูกเก็บเข้าหรือถูกนำออกโดยอัตโนมัติ เช่น เมื่อซีพียูทำคำสั่ง CALL หรือเมื่อซีพียูถูกขัดจังหวะมันจะเก็บค่าของ PC เข้าไปในสแตคอัตโนมัติ และเมื่อซีพียูทำคำสั่ง RET, RETI หรือ RETN ข้อมูลของ PC ที่เก็บไว้ยังสามารถนำเข้าหรือนำออกจากสแตคโดยใช้คำสั่งก็ได้ การนำข้อมูลเข้าหรือออกจากสแตคนี้ตำแหน่งบนสุดของสแตคจะถูกชี้โดยข้อมูลที่อยู่ในสแตคพอยน์เตอร์เสมอ ตัวอย่างดังรูป 2.37

ตำแหน่ง	หน่วยความจำ	ข้อมูลใน SP	ตำแหน่ง	หน่วยความจำ	ข้อมูลใน SP
0000	ข้อมูล A	0103	0100	ข้อมูล B	0101
0101			0101		
0102			0102		
0103			0103		
0104			0104		

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่สามารถนำข้อมูล A ขนาด 16 บิต (ก) PUSH ข้อมูล A ขนาด 16 บิต (ข) PUSH ข้อมูล B ขนาด 16 บิต

ตำแหน่ง	หน่วยความจำ	ข้อมูลใน SP
0100		0103
0101		
0102		
0103	ข้อมูล A	
0104	ข้อมูล A	

(ค) POP ข้อมูล B

รูปที่ 2.37 แสดงการทำงานของสแตกและสแตกพอยท์เตอร์

สังเกตได้ว่าข้อมูลที่อยู่ใน SP คือตำแหน่งบนสุดของสแตกที่มีข้อมูลอยู่เสมอ

3) อินเด็กซ์รีจิสเตอร์ (Index Register)

ไมโครโปรเซสเซอร์ Z-80 มีอินเด็กซ์รีจิสเตอร์ขนาด 16 บิต 2 ตัว คือ IX และ IY ซึ่งทั้งสองตัวนี้มีลักษณะการทำงานที่เหมือนกันทุกประการ คือ ใช้ในคำสั่งที่มีการอ้างถึงตำแหน่งข้อมูลแบบอินเด็กซ์ โดยที่ตำแหน่งของข้อมูลใช้งาน (Effective Address) ที่อยู่ในหน่วยความจำกำหนดโดยการนำข้อมูลที่อยู่ในอินเด็กซ์รีจิสเตอร์ที่ใช้เป็นตำแหน่งฐาน (Base Address) บวกกับระยะห่าง (Displacement) ขนาด 8 บิตที่บรรจุอยู่ในส่วนของคำสั่ง ดังนั้นอาจเขียนเป็นสมการได้ คือ

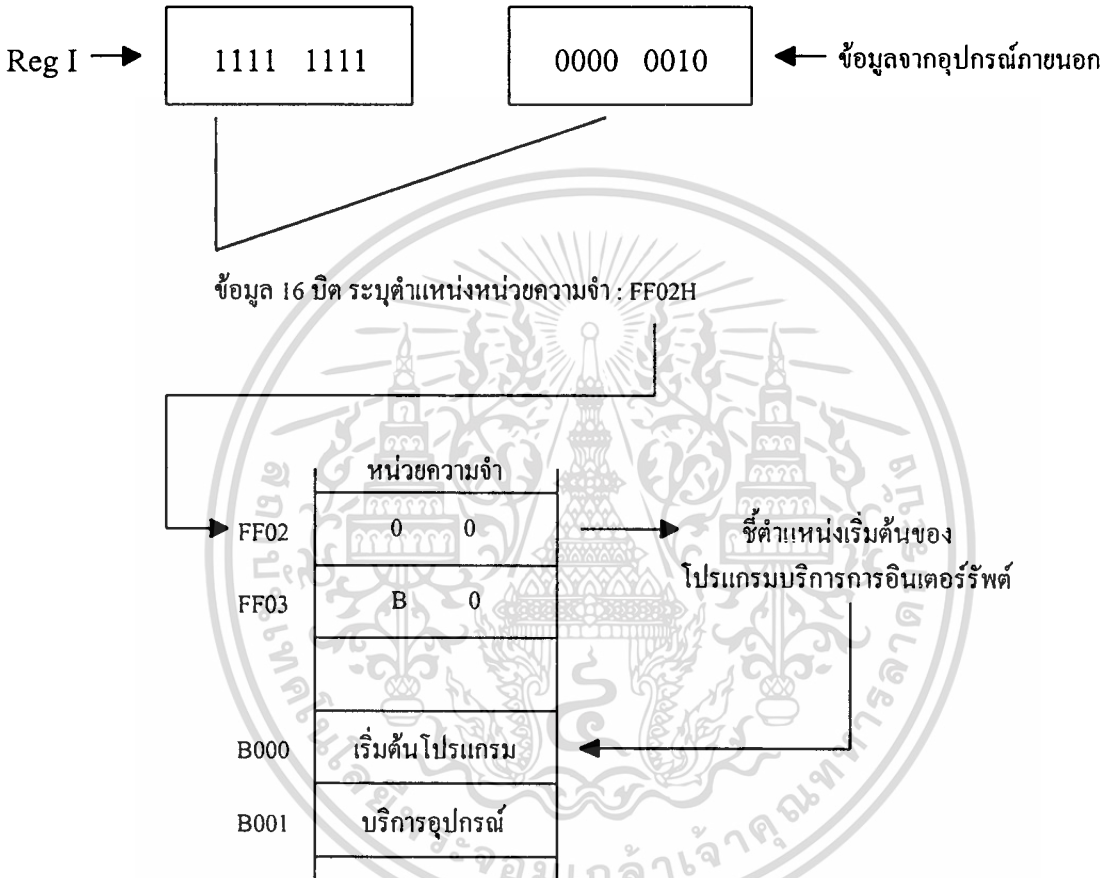
$$\text{Effective Address} = \text{Base address} + \text{Displacement} \quad (2.2)$$

การทำงานแบบอินเด็กซ์นี้มีประโยชน์มากในการเขียนโปรแกรมเพื่อให้ทำงานแบบวนรอบตามที่ต้องการ

4) อินเตอร์รัพต์เวกเตอร์รีจิสเตอร์ (I)

เป็นรีจิสเตอร์ขนาด 8 บิตที่สามารถโหลดข้อมูลขนาด 8 บิตลงไปได้ ข้อมูลในรีจิสเตอร์นี้ใช้เป็นส่วนหนึ่งของตำแหน่งหน่วยความจำที่จะอ้างถึง โดยใช้เป็นข้อมูล 8 บิตด้านสูง (High Byte) เพื่อใช้ชี้ตำแหน่งของตารางเวกเตอร์ (Vector Table) และเมื่อนำข้อมูลอีก 8 บิต มาต่อทางด้านต่ำ (Low Byte) ก็จะกลายเป็น 16 บิตที่ใช้ชี้ตำแหน่งของหน่วยความจำในตารางเวกเตอร์ ข้อมูล 8 บิตทางด้านต่ำที่จะมาต่อเป็นข้อมูลที่ส่งมาจากอุปกรณ์อินพุต, เอาต์พุต เมื่ออุปกรณ์อินพุต, เอาต์พุต ส่งสัญญาณเพื่อทำการอินเตอร์รัพต์การทำงานตามปกติของซีพียู และถ้าซีพียูตอบรับการอินเตอร์รัพต์

นั้นอุปกรณ์ตัวที่ส่งสัญญาณการอินเทอร์รัพต์จะส่งข้อมูลขนาด 8 บิตมาให้ซึ่งข้อมูลนี้คือข้อมูลขนาด 8 บิตด้านต่ำที่จะมารวมกับข้อมูลที่อยู่ในรีจิสเตอร์ I เพื่อชี้ตำแหน่งตารางเวกเตอร์ และข้อมูลที่อยู่ในหน่วยความจำตามตำแหน่งในตารางที่อ้างถึง 2 ไบต์จะเป็นตำแหน่งเริ่มต้นของโปรแกรมบริการการอินเทอร์รัพต์ของอุปกรณ์ตัวนั้นๆ ลักษณะการทำงานดังกล่าวนี้อธิบายได้ดังรูปที่ 2.36



รูปที่ 2.38 การทำงานของรีจิสเตอร์ I

จากรูปที่ 2.38 กำหนดข้อมูลในรีจิสเตอร์ I มีค่าเท่ากับ FFH และข้อมูลที่ส่งมาจากอุปกรณ์ภายนอกและมาต่อข้อมูลของรีจิสเตอร์ I คือ 02H เมื่อข้อมูลทั้งสองรวมกันได้ FF02H ซึ่งเป็นตำแหน่งของหน่วยความจำภายนอกที่จะอ้างถึง หน่วยความจำในส่วนนี้เรียกว่า ตารางเวกเตอร์ โดยที่ในตารางนี้จะบรรจุตำแหน่งของโปรแกรมบริการการอินเทอร์รัพต์ไว้ ดังนั้นค่าในรีจิสเตอร์ I หนึ่งค่าสามารถกำหนดตำแหน่งการบริการของการอินเทอร์รัพต์ได้ถึง 128 ตำแหน่ง จากรูป ตำแหน่งที่ FF02H และ FF03H จะบรรจุตำแหน่งเริ่มต้นของโปรแกรมบริการอุปกรณ์ที่ส่งสัญญาณการอินเทอร์รัพต์เข้ามา ดังนั้นโปรแกรมบริการของอุปกรณ์จึงอยู่ที่ตำแหน่ง B000H รีจิสเตอร์ I นี้ใช้เพื่อ

การบริการอินเตอร์รัพต์ในโหมด 2 ของการบริการอินเตอร์รัพต์ของซีพียู Z-80 ซึ่งมี 3 โหมด การอินเตอร์รัพต์แบบต่างๆ ของซีพียูจะได้กล่าวถึงโดยละเอียดในเรื่องของการอินเตอร์รัพต์

5) รีเฟรชรีจิสเตอร์ (R)

เมื่อซีพียู Z-80 ต่อกับหน่วยความจำแบบไดนามิก ซึ่งหน่วยความจำแบบนี้ต้องมีการอัดซ้ำ (Refresh) ตลอดเวลา (ปกติทุกๆ 2 ms) เพื่อให้ข้อมูลในหน่วยความจำนี้ยังคงอยู่ ซีพียู Z-80 นี้ใช้รีจิสเตอร์ R เพื่อบรรจุตำแหน่งของหน่วยความจำเพื่อทำการอัดซ้ำ รีจิสเตอร์ R มีขนาด 7 บิต และข้อมูลในรีจิสเตอร์จะเพิ่มขึ้นทีละ 1 โดยอัตโนมัติหลังจากที่ทำการเฟรชคำสั่งหนึ่งๆ แล้วข้อมูลในรีเฟรชรีจิสเตอร์ถูกส่งออกไปทางด้านล่างของบัสตำแหน่ง (A6-A0) ในขณะที่สัญญาณรีเฟรชแอกทีฟ และซีพียูไม่ได้ Access หน่วยความจำ รีจิสเตอร์ R โดยปกติผู้เขียนโปรแกรมไม่จำเป็นต้องใช้ นอกจากต้องการตรวจสอบเท่านั้น วิธีการเชื่อมต่อระหว่างซีพียู Z-80 กับหน่วยความจำแบบไดนามิกแรมจะได้กล่าวถึงในโอกาสต่อไป

2.5.5 ขาและสัญญาณสำหรับการเชื่อมต่อ

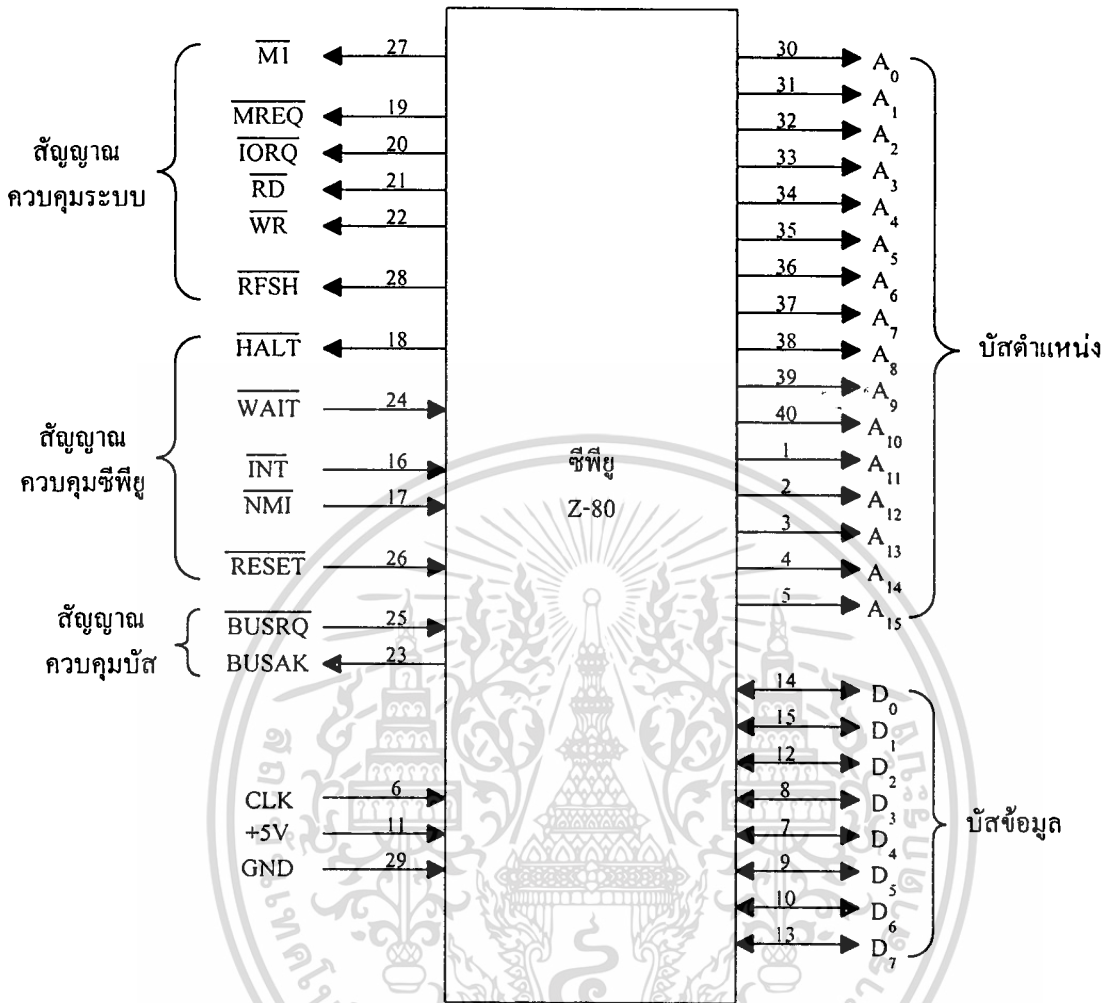
ไมโครโปรเซสเซอร์ Z-80 บรรจุอยู่ในไอซีขนาดมาตรฐานอุตสาหกรรม (Industry Standard) แบบ Dual In-Line Package (DIP) หรือที่เรียกว่าแบบดินตะขาบ 40 ขา ขาต่างๆ แสดงไว้ในรูปที่ 2.39

กลุ่มสัญญาณต่างๆ ของไมโครโปรเซสเซอร์แบ่งออกได้เป็น 3 กลุ่มคือ กลุ่มของสายสัญญาณเพื่อกำหนดตำแหน่ง (Address Bus) คือ A15-A0 กลุ่มของสายสัญญาณข้อมูล (Data Bus) คือ D7-D0 และกลุ่มของสายสัญญาณควบคุม (Control Bus) คือสายสัญญาณทั้งหมดที่เหลือ ยกเว้นขาแหล่งจ่ายไฟและสัญญาณนาฬิกา หน้าที่ของขาต่างๆ จะได้อธิบายในรายละเอียดต่อไป

A15-A0 เป็นสายสัญญาณกำหนดตำแหน่ง (Address Bus) โดยที่ A0 เป็นบิตทางด้านต่ำ (LSB) ขาเหล่านี้เป็นเอาต์พุตแบบสามสถานะ (Tri-State) และจะให้แอกทีฟที่ลอจิก 1 บัสนี้มีด้วยกันทั้งหมด 16 สาย ดังนั้นจึงสามารถติดต่อกับหน่วยความจำได้ถึง 65536 ตำแหน่ง (64 Kbytes) นอกจากนั้นยังสามารถใช้ในการกำหนดตำแหน่งของพอร์ตอินพุต. เอาต์พุต เมื่อใช้คำสั่งกลุ่มอินพุต, เอาต์พุตได้ โดยใช้ 8 บิตด้านต่ำ (A7-A0) เพื่อแสดงตำแหน่งของพอร์ต ดังนั้นจึงสามารถกำหนดพอร์ตอินพุตได้ 256 พอร์ต หรือกำหนดพอร์ตเอาต์พุตได้ถึง 256 พอร์ตเช่นกัน และในช่วงเวลารีเฟรช บัสด้านต่ำ 7 บิต (A6-A0) จะใช้แสดงตำแหน่งของหน่วยความจำแบบไดนามิกที่จะได้รับการรีเฟรช

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

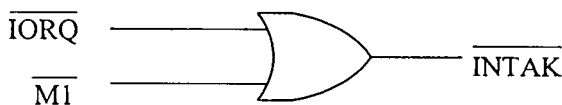
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.39 แสดงขาต่างๆ ของซีพียู Z-80

D7-D0

เป็นสายสัญญาณข้อมูล (Data Bus) D0 เป็นทิศทางด้านต่ำลักษณะเป็นบัสแบบสองทิศทางแบบสามสถานะ ขนาด 8 บิต และแอสท์ฟที่ลอจิก 1 ใช้เพื่อเป็นเส้นทางผ่านของข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำหรืออุปกรณ์อินพุต, เอาต์พุตต่างๆ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น รูปที่ 2.40 วงจรสร้างสัญญาณตอบรับการอินเทอร์รัพต์

- M1 (Machine Cycle One) เป็นขาเอาต์พุตและแอกทีฟที่ลอจิก 0 เมื่อขาอินพุตที่ $\overline{A0}$ ซึ่งให้เห็นว่าขณะนี้กำลังอยู่ในสถานะของการเฟรชคำสั่ง และถ้าเป็นคำสั่งที่มีรหัส 2 ไบต์ ส่วนของ M1 จะถูกสร้างขึ้นขณะเฟรชในแต่ละไบต์ ลักษณะของคำสั่งที่มีขนาด 2 ไบต์ เช่นคำสั่งที่มีรหัสที่เริ่มต้นด้วย CBH, DDH, EDH หรือ FDH นอกจากนั้นสัญญาณ M1 นี้จะใช้ร่วมกับ IORQ เพื่อสร้างสัญญาณตอบรับการอินเตอร์รัพต์ (Interrupt Acknowledge) โดยใช้วงจรลอจิกง่ายๆ ดังรูป 2.38
- MREQ (Memory Request) เป็นขาเอาต์พุตแบบสามสถานะและแอกทีฟที่ลอจิก 0 เมื่อสายสัญญาณนี้แอกทีฟ บอกให้ทราบว่าขณะนี้ไมโครโปรเซสเซอร์ต้องการติดต่อกับหน่วยความจำเพื่ออ่านหรือเขียนข้อมูล โดยที่ตำแหน่งของหน่วยความจำจะปรากฏอยู่บนบัสตำแหน่งแล้ว
- IORQ (Input/Output Request) เป็นขาเอาต์พุตแบบสามสถานะและแอกทีฟที่ลอจิก 0 เมื่อสายสัญญาณนี้แอกทีฟ บอกให้ทราบว่าขณะนี้ทางด้านไบต์ค่า ($A7-A0$) ของบัสตำแหน่งบรรจุดำแหน่งของพอร์ตที่จะส่งถ่ายข้อมูลระหว่างไมโครโปรเซสเซอร์กับอุปกรณ์อินพุต, เอาต์พุต นอกจากนั้นจะใช้ร่วมกับสัญญาณ M1 เพื่อตอบรับการอินเตอร์รัพต์ดังรูปที่ 2.40 และขณะนี้เวกเตอร์ของการอินเตอร์รัพต์จะส่งผ่านเข้ามาในบัสข้อมูลเพื่อกำหนดตำแหน่งของโปรแกรมบริการการอินเตอร์รัพต์
- RD (Memory Read) เป็นขาเอาต์พุตแบบสามสถานะและแอกทีฟที่ลอจิก 0 สัญญาณนี้เพื่อชี้ว่าขณะนี้ไมโครโปรเซสเซอร์ต้องการอ่านข้อมูลจากหน่วยความจำ หรือจากอุปกรณ์อินพุต, เอาต์พุต
- WR (Memory Write) เป็นขาเอาต์พุตแบบสามสถานะ และแอกทีฟที่ลอจิก 0 เมื่อสัญญาณนี้แอกทีฟชี้ว่าขณะนี้ไมโครโปรเซสเซอร์ต้องการเขียนข้อมูลเข้าหน่วยความจำ หรือเข้าอุปกรณ์อินพุต, เอาต์พุต
- RFSH (Refresh) เป็นขาเอาต์พุตแอกทีฟที่ลอจิก 0 RFSH เป็นสัญญาณเพื่อชี้ว่าขณะนี้บัสตำแหน่งทางด้านต่ำ 7 บิต ($A6-A0$) บรรจุดำแหน่งหน่วยความจำแบบไดนามิกแรมที่จะรีเฟรช และสัญญาณ MREQ ในช่วงนี้จะนำไปใช้เป็นสัญญาณสำหรับอ่านเพื่อรีเฟรช (Refresh Read) ไดนามิกแรมทั้งหมดที่ใช้ในระบบ
- HALT (Halt State) เป็นขาเอาต์พุตแอกทีฟที่ลอจิก 0 เป็นสัญญาณเพื่อชี้ว่าขณะนี้ไมโครโปรเซสเซอร์ปฏิบัติคำสั่ง HALT จากโปรแกรม และกำลังรอสัญญาณการอินเตอร์รัพต์ชนิดนอนมาสเคเบิลหรือมาสเคเบิล (เมื่อสั่งให้ยอมรับ) โดยจากการดำเนินการดังกล่าวไม่ว่ากรณีใดๆ อุปกรณ์ภายนอก ถ้าได้รับสัญญาณการอินเตอร์รัพต์แล้วจึงทำงานต่อไปได้ ในขณะที่

ที่หยุดรอ (Halted) นี้ ซีพียูจะกระทำคำสั่ง NOP (No-Operation) เพื่อให้มีการเฟตช์คำสั่งซึ่งจะไม่ทำให้การรีเฟรชหยุดชะงักลง

WAIT

(Wait) เป็นขาอินพุตแอกทีฟที่ลอจิก 0 เป็นสัญญาณเพื่อชี้ว่าการส่งถ่ายข้อมูลระหว่างไมโครโปรเซสเซอร์และหน่วยความจำ หรืออุปกรณ์อินพุต, เอาต์พุต ยังไม่เรียบร้อย และทำให้ไมโครโปรเซสเซอร์หยุดรอตามเท่าที่ขาอินพุตนี้แอกทีฟอยู่ ดังนั้นสัญญาณนี้จะใช้เพื่อให้หน่วยความจำหรืออุปกรณ์อินพุต, เอาต์พุตที่มีความเร็วใดๆ สามารถทำงานให้เข้าจังหวะกันได้พอดี (Synchronized) กับไมโครโปรเซสเซอร์

INT

(Interrupt Request) เป็นขาอินพุตแอกทีฟที่ลอจิก 0 สัญญาณ INT นี้เป็นสัญญาณที่สร้างมาจากอุปกรณ์อินพุต, เอาต์พุต เพื่อต้องการอินเตอร์รัพต์การทำงานตามปกติของไมโครโปรเซสเซอร์ สัญญาณร้องขอนี้จะถูกตรวจสอบเมื่อถึงสเตตสุดท้ายของคำสั่ง และไมโครโปรเซสเซอร์จะจดจำไว้ ถ้าหากว่าโปรแกรมกำหนดให้มีการยอมรับสัญญาณการอินเตอร์รัพต์ได้ (Enable Interrupt) โดย IFF1 ถูกเซตเป็น 1 และไม่มีการขอใช้บัสเสียก่อน คือ ขา BUSRQ ต้องไม่แอกทีฟ เมื่อไมโครโปรเซสเซอร์รับสัญญาณอินเตอร์รัพต์มันจะตอบสนองโดยการส่งสัญญาณ IORQ ออกมาในช่วงเวลา M1 เพื่อเป็นการตอบรับการอินเตอร์รัพต์ (Interrupt Acknowledge) ในช่วงไทม์เกิดของคำสั่งต่อมาส่วนรายละเอียดของการอินเตอร์รัพต์จะอธิบายในเรื่องอินเตอร์รัพต์

NMI

(Non Maskable Interrupt) เป็นขาอินพุตและแอกทีฟที่ขอบพัลส์ขาลง (Negative edge Trigger) สัญญาณที่ขา NMI นี้มีลำดับความสำคัญสูงกว่าสัญญาณที่ขา INT ไมโครโปรเซสเซอร์จะทำการตรวจสอบขาอินพุตนี้ที่สเตตสุดท้ายของคำสั่ง เช่นเดียวกับขา INT แต่จะไม่ขึ้นอยู่กับการตั้งค่า IFF เมื่อไมโครโปรเซสเซอร์ได้รับสัญญาณที่ขา NMI จะทำให้เริ่มต้นการทำงานใหม่ที่ตำแหน่ง 0066H ส่วนค่าในโปรแกรมเคาน์เตอร์ที่ชี้ตำแหน่งของคำสั่งต่อไปก่อนที่ซีพียูจะถูกอินเตอร์รัพต์จะถูกเก็บไว้ในสแตค (ที่แรม) เพื่อที่ซีพียูสามารถกลับมาทำงานต่อได้ หลังจากที่ทำโปรแกรมบริการการอินเตอร์รัพต์เสร็จสิ้นแล้ว ในขณะที่ซีพียูอยู่ในจังหวะ Wait มันจะไม่รับสัญญาณ NMI นี้ สัญญาณ NMI มีลำดับความสำคัญต่ำกว่าสัญญาณ BUSRQ ดังนั้นในขณะที่ซีพียูกำลังทำโปรแกรมบริการการอินเตอร์รัพต์อยู่มันสามารถรับสัญญาณ BUSRQ ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RESET	เป็นอินพุตแอกทีฟที่ลอจิก 0 เมื่อไมโครโปรเซสเซอร์ได้รับสัญญาณรีเซต จะทำให้ค่าในโปรแกรมเคาน์เตอร์เริ่มต้นที่ 0 และตั้งต้นการทำงานของไมโครโปรเซสเซอร์ใหม่ และในส่วนอื่นๆ เป็นดังนี้ 1. จัดอินเตอร์รัพท์ฟลิปฟล็อป (IFF) ให้อยู่ในสภาพที่ไม่ยอมรับการอินเตอร์รัพท์แบบมาสเคเบิล ($IFF1 = IFF2 = 0$) 2. เซตรีจิสเตอร์ I = 00H 3. เซตรีจิสเตอร์ R = 00H 4. เซตให้เป็นการอินเตอร์รัพท์โหมด 0 ในช่วงเวลาของการรีเซต บัสข้อมูล บัสตำแหน่ง จะอยู่ในสถานะอิมพีแดนซ์สูง ส่วนบัสควบคุมจะอยู่ในสภาพที่ไม่แอกทีฟ (Inactive)
BUSRQ	(Bus Request) เป็นขาอินพุตแอกทีฟที่ระดับ 0 สัญญาณ BUSRQ นี้มีผลให้บัสตำแหน่ง และสัญญาณข้อมูล และสัญญาณควบคุมที่เป็นขาเอาต์พุตแบบสามสถานะอยู่ในสถานะอิมพีแดนซ์สูง จากนั้นบัสต่างๆ จะถูกควบคุมโดยอุปกรณ์ภายนอก ไมโครโปรเซสเซอร์จะตรวจสอบสัญญาณการขอใช้บัสนี้ทุกๆ สิบนาโนวินาทีสุดท้ายของทุกเมกซ์ซีนไซเคิลของคำสั่ง เมื่อพบการขอใช้บัส ซีพียูจะตอบสนองในไซเคิลถัดไป
BUSAK	(Bus Acknowledge) เป็นขาเอาต์พุตแอกทีฟที่ระดับ 0 สัญญาณนี้ใช้สำหรับตอบรับการขอใช้บัส และแสดงว่าขณะนี้บัสตำแหน่ง บัสข้อมูล และสัญญาณควบคุมที่เป็นเอาต์พุตแบบสามสถานะอยู่ในสถานะอิมพีแดนซ์สูงแล้ว อุปกรณ์ควบคุมภายนอกสามารถเข้ามาควบคุมบัสได้
Φ	เป็นขาที่รับสัญญาณนาฬิกาซึ่งเป็นเพียงเฟสเดียว ใช้ระดับสัญญาณแบบ TTL และต้องการตัวต้านทานเพื่อ Pull Up ค่า 330 โอห์มหนึ่งตัว เพื่อต่อกับแหล่งจ่ายไฟ 5 โวลต์ Z-80 ทำงานได้ที่สัญญาณนาฬิกาไม่เกิน 2.2 MHz Z-80A ทำงานได้ไม่เกิน 4 MHz และ Z-80 ทำงานได้ไม่เกิน 6 MHz

2.5.6 แผนผังเวลาของไมโครโปรเซสเซอร์

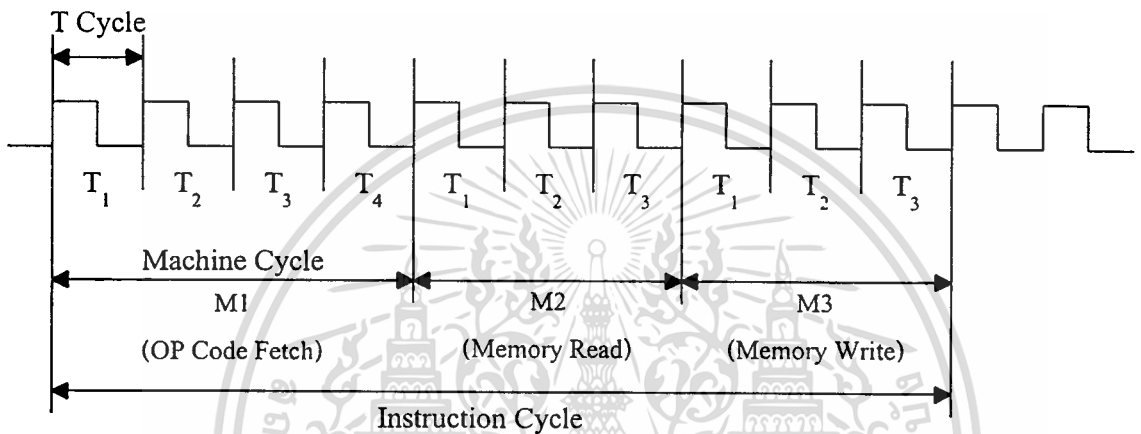
ไมโครโปรเซสเซอร์ Z-80 จะปฏิบัติตามคำสั่งต่างๆ อย่างมีขั้นตอนที่แน่นอน ขั้นตอนเหล่านี้ประกอบด้วยการทำงานพื้นฐานต่างๆ คือ

- การอ่าน-เขียนหน่วยความจำ (Memory Read or Write)

เอกสารนี้ การอ่าน-เขียนอุปกรณ์อินพุต, เอาต์พุต (I/O Device Read or Write)

ไม่มีการรับประกันการอินเตอร์รัพท์ (Interrupt Acknowledge)

การทำงานของคำสั่งต่างๆ เกิดจากลำดับการทำงานพื้นฐานนี้ ขบวนการทำงานพื้นฐานแต่ละส่วนอาจใช้เวลา 3-6 คาบ กว่าเสร็จสิ้นสมบูรณ์หรืออาจขยายออกไปได้มากกว่านี้ เพื่อให้การทำงานของไมโครโปรเซสเซอร์พอดีกับความเร็วของอุปกรณ์ภายนอก คาบเวลาของสัญญาณหนึ่งๆ นี้ คือ T ไชเกิล และขบวนการพื้นฐานหนึ่งๆ จะเรียกว่าแมชชีนไชเกิล (Machine Cycle) ตัวอย่างแผนผังเวลาของคำสั่งหนึ่งซึ่งแสดงดังรูป



รูปที่ 2.41 ตัวอย่างแผนผังเวลาของไมโครโปรเซสเซอร์

จากรูปที่ 2.41 เป็นคำสั่งที่ประกอบด้วย 3 แมชชีนไชเกิล (M1, M2 และ M3) แมชชีนไชเกิลที่ 1 (M1) ของคำสั่งใดๆ เป็นไชเกิลของการเฟตช์ ซึ่งอาจมี 4, 5 หรือ 6 T ไชเกิล ยกเว้นในกรณีที่เพิ่มเวลาออกไปเนื่องจากการรอคอย (Wait State) ไชเกิลของการเฟตช์ (M1) ใช้เพื่ออ่านรหัสคำสั่งต่อไปที่จะกระทำ ลำดับต่อมาทำการเคลื่อนย้ายข้อมูลระหว่างไมโครโปรเซสเซอร์กับหน่วยความจำหรืออุปกรณ์อินพุต, เอาต์พุต ซึ่งจะใช้เวลาตั้งแต่ 3 ถึง 5 คาบเวลา เว้นแต่มีการรอคอยเช่นกัน แมชชีนไชเกิลที่ตามหลัง M1 จะมีหรือไม่ก็ได้แล้วแต่นิยามของคำสั่ง เช่น คำสั่ง LD r,r จะใช้เพียง 1 แมชชีนไชเกิลก็สามารถทำงานได้ แผนผังเวลาซึ่งปรากฏอยู่ในการทำงานพื้นฐานต่างๆ ทั้งที่มีช่วงการรอคอยและไม่มี การรอคอย ซึ่งแมชชีนไชเกิลที่มีใช้สามารถสรุปได้ดังนี้

1. ไชเกิลการเฟตช์รหัสคำสั่ง (Instruction OPCode Fetch Cycle (M1))
2. ไชเกิลการอ่านหรือเขียนหน่วยความจำ (Memory Data Read or Write Cycle)
3. ไชเกิลการอ่านหรือเขียนข้อมูลกับอุปกรณ์อินพุต, เอาต์พุต (I/O Read or Write Cycle)
4. ไชเกิลการขอใช้บัสหรือการตอบสนองการขอใช้บัส (Bus Request/Acknowledge Cycle)

เอกสารนี้เป็นเอกสารทบทวนเนื้อหาสำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ในการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. ไซเกิลการขออินเทอร์รัพต์ และการตอบรับการอินเทอร์รัพต์ (Interrupt Request/Acknowledge Cycle)
6. ไซเกิลการขอและตอบรับการอินเทอร์รัพต์แบบนอนมาสเคเบิล (Non-Maskable Interrupt Request/Acknowledge Cycle)
7. ไซเกิลการออกจากคำสั่ง HALT (Exit From a HALT Instruction)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

การออกแบบ การสร้างและการทำงาน

ผู้ที่เคยผ่านงานกับการใช้ซิงเกิลบอร์ดมาแล้วจะพบว่า ปัญหาหลักอย่างหนึ่งในการทำงานกับซิงเกิลบอร์ดคือ อุปกรณ์ที่ใช้เก็บข้อมูลซึ่งในซิงเกิลบอร์ดจะใช้หน่วยความจำประเภทแรมในการเก็บข้อมูลรวมถึงโปรแกรมที่ใช้งาน หน่วยความจำประเภทนี้ข้อมูลที่ถูเก็บไว้จะถูกลบหายไปเมื่อตัดไฟเลี้ยงออกจากซิงเกิลบอร์ด ดังนั้นการเก็บข้อมูลจึงไม่คงทนถาวรเมื่อเทียบกับการเก็บข้อมูลโดยอุปกรณ์เก็บข้อมูลชนิดอื่นๆ

ในการพัฒนาโปรแกรมหรือการเขียนโปรแกรมโดยใช้ซิงเกิลบอร์ด โปรแกรมที่เขียนนั้นจะถูกจัดเก็บไว้ในหน่วยความจำแรม ทำให้สามารถเห็นถึงจุดด้อยในการพัฒนาโปรแกรมโดยใช้ซิงเกิลบอร์ด ในกรณีที่สามารถทำการถ่ายโอนข้อมูลระหว่างคอมพิวเตอร์กับซิงเกิลบอร์ดได้ การพัฒนาโปรแกรมจะไม่ใช่ปัญหามากนัก ในทางตรงข้ามหากผู้พัฒนาโปรแกรมไม่มีคอมพิวเตอร์หรือซิงเกิลบอร์ดไม่ได้สร้างจุดเชื่อมต่อข้อมูลข่าวสารกับเครื่องคอมพิวเตอร์ จะถือได้ว่าเป็นอุปสรรคใหญ่ของการพัฒนาโปรแกรม เพราะหมายถึงจะต้องป้อนโปรแกรมบนเครื่องซิงเกิลบอร์ดใหม่ทุกครั้งที่เปิดเครื่อง หรือการที่ต้องป้อนโปรแกรมใหม่ทุกครั้งที่มีการพัฒนาโปรแกรมครั้งละมากกว่าหนึ่งโปรแกรม ทำให้การพัฒนาโปรแกรมให้มีประสิทธิภาพเป็นไปได้ยาก

อีกปัญหาหนึ่งที่สืบเนื่องจากปัญหาเรื่องอุปกรณ์เก็บข้อมูลคือ ข่าวสารข้อมูลที่ได้จากการประมวลผลของตัวซิงเกิลบอร์ดนั้น การนำเอาข่าวสารข้อมูลเหล่านั้นมาแสดงผลทั้งหมดเป็นไปได้ยาก เนื่องจากขีดจำกัดในเรื่องของการแสดงผลของซิงเกิลบอร์ด หรือการนำเอาข่าวสารข้อมูลไปประมวลผลโดยใช้โปรแกรมประยุกต์หลายๆ โปรแกรมเพื่อใช้ในการวิเคราะห์ประมวลผล ขีดจำกัดของหน่วยความจำของซิงเกิลบอร์ดคือ การถ่ายโอนข้อมูลระหว่างคอมพิวเตอร์กับซิงเกิลบอร์ด สถานที่ใช้งานเพื่อเก็บข่าวสาร โดยใช้ซิงเกิลบอร์ดล้วนเป็นตัวแทนสำคัญในการเลือกเก็บข่าวสารโดยใช้ซิงเกิลบอร์ดเป็นตัวประมวลผลข้อมูล

จากเหตุผลที่กล่าวมาทั้งหมดนี้เพื่อที่จะให้ถึงอุปกรณ์ที่ใช้เก็บข่าวสารข้อมูลที่สามารถเก็บจุครบพร้อมดังกล่าวโดยจะต้องมีประสิทธิภาพในเรื่องของความคงทนของการเก็บข่าวสารข้อมูลได้ดีกว่าแรมและมีความจุในการเก็บข้อมูลที่มากกว่า รวมทั้งต้องมีความน่าเชื่อถือในการจัดเก็บข้อมูลได้ดีทัดเทียมกับแรมโดยอุปกรณ์เก็บข้อมูลที่อ้างอิงถึงคือ จานแม่เหล็กแบบอ่อนหรือที่รู้จักกันในชื่อของฟลอปปีดิสก์ที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 โครงสร้าง

จากหลักการที่กล่าวมาแล้วในตอนต้น ทำให้สามารถทราบถึงรูปแบบการทำงานอย่างคร่าวๆ ของตัววงจรที่จะทำการออกแบบได้ โดยแบ่งหน้าที่ของวงจรออกเป็น 2 ประการดังนี้ คือ

3.1.1 นำข้อมูลที่อยู่ในหน่วยความจำของซิงเกิลบอร์ดมาเก็บไว้ในแผ่นฟลอปปีดิสก์

ตัวแปรในการทำงานส่วนนี้คือ ตำแหน่งหน่วยความจำเริ่มต้น และตำแหน่งหน่วยความจำสุดท้ายของซิงเกิลบอร์ดที่ผู้ใช้งานต้องการบันทึกไว้ในแผ่นฟลอปปีดิสก์ ชื่อของแฟ้มข้อมูลที่จะทำการจัดเก็บ และที่สำคัญอีกประการหนึ่งคือ รูปแบบที่ใช้ในการบันทึกข้อมูลลงบนแผ่นฟลอปปีดิสก์ ควรเป็นรูปแบบเดียวกันกับที่ใช้ในเครื่องคอมพิวเตอร์

3.1.2 นำข้อมูลจากแผ่นฟลอปปีดิสก์มาเก็บไว้ในหน่วยความจำของซิงเกิลบอร์ด

หน้าที่ในส่วนนี้ตัวแปรคือ ชื่อของแฟ้มข้อมูลที่ใช้ในการเก็บข่าวสารข้อมูล ตำแหน่งของหน่วยความจำเริ่มต้นของซิงเกิลบอร์ดว่าจะให้เริ่มบันทึกข่าวสารที่ได้จากการอ่านแฟ้มข้อมูลที่ตำแหน่งใด

สิ่งที่พิจารณาเป็นอันดับต่อไปคือ การติดต่อกับอุปกรณ์ภายนอกอุปกรณ์ที่จะทำการเชื่อมต่อเข้ากับตัววงจรหลักจะมีอยู่ 2 อย่าง คือ ตัวซิงเกิลบอร์ด และ ดิสก์ไดรฟ์



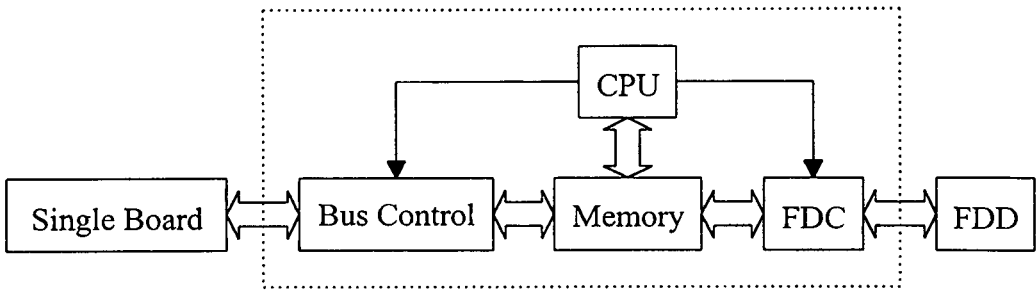
รูปที่ 3.1 การเชื่อมต่ออุปกรณ์ภายนอกเข้ากับวงจร

จากรูปที่ 3.1 ตัววงจรหลักจะเป็นตัวเชื่อมต่อข้อมูลข่าวสารระหว่างซิงเกิลบอร์ดกับดิสก์ไดรฟ์เข้าด้วยกัน ซึ่งการเชื่อมโยงในแต่ละด้านนี้จะมีอุปกรณ์และรายละเอียดวงจรแบ่งย่อยลงไปอีก ซึ่งจะกล่าวในหัวข้อต่อไป

3.2 แผนผังการทำงานของวงจร

จากรูปที่ 3.2 เป็นรายละเอียดในส่วนของวงจรหลักของรูปที่ 3.1 ซึ่งมีส่วนประกอบย่อยออกเป็น 4 ภาคด้วยกัน ดังนี้ คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



โดยที่

CPU : Central Processing Unit

FDC : Floppy Disk Controller

FDD : Floppy Disk Drive

รูปที่ 3.2 แผนผังการทำงานของวงจร

3.2.1 ซีพียู

เป็นส่วนที่ทำหน้าที่ควบคุมและลำดับขั้นตอนการทำงานของวงจรทั้งหมด เปรียบเสมือนสมองของวงจรทั้งหมด

3.2.2 หน่วยความจำ

ใช้ในการพักข้อมูลข่าวสารระหว่างการโอนย้ายข้อมูลของซิงเกิลบอร์ดไปยังแผ่นฟลอปปีดิสก์โดยตรง จะใช้เวลาในการถ่ายโอนข้อมูลมาก เนื่องจากจะต้องมีการควบคุมทางด้านกลไกของตัวขับเคลื่อนฟลอปปีดิสก์ในการกำหนดตำแหน่งที่จะบันทึก นอกจากนั้นยังต้องเสียเวลาไปกับการการคำนวณค่าต่างๆ ในการบันทึก ซึ่งหากเป็นการถ่ายโอนข้อมูลระหว่างหน่วยความจำจะสามารถถ่ายโอนข้อมูลได้ในระยะเวลาอันสั้น ซึ่งช่วยให้สามารถใช้งานซิงเกิลบอร์ดได้ทันทีหลังจากการโอนย้ายข้อมูลเสร็จสิ้นโดยไม่ต้องสนใจว่าการบันทึกข้อมูลลงบนแผ่นดิสก์เสร็จสิ้นแล้วหรือไม่

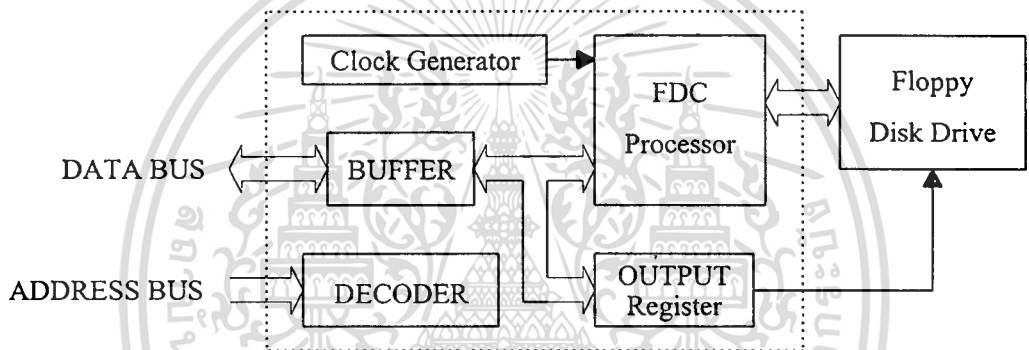
3.2.3 Bus Controller

ใช้ในการควบคุมบัสข้อมูลที่อยู่บนซิงเกิลบอร์ด โดยอาศัยหลักการควบคุมบัสโดยใช้ซีพียูจากภายนอกแทนซีพียูที่อยู่ภายในซิงเกิลบอร์ด ซึ่งอ้างอิงการใช้ซิงเกิลบอร์ดที่ใช้ซีพียูเบอร์ Z-80 เป็นตัวประมวลผลกลาง โดย Bus Controller จะสร้างสัญญาณร้องขอการใช้บัส และจัดสัญญาณตำแหน่งและข่าวสารข้อมูลส่งไปควบคุมการทำงานแทนที่ซีพียู(Z-80) ในซิงเกิลบอร์ด ซึ่งหน้าที่หลักๆ ของ Bus Controller นอกจากจะสร้างสัญญาณต่างๆ ครอบคลุมและทำงานแทนตัวซีพียูบนไมโครมินิเตอร์ หงส์สีน อีกทั้งห้ามมิให้ตดแบบสั่นโยนให้ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซิงเกิลบอร์ดและยังทำหน้าที่เป็นส่วนที่นำเอาข้อมูลจากแผ่นฟลอปปีดิสก์มาเก็บไว้ในตัวแรมของซิงเกิลบอร์ด และส่งผ่านข้อมูลจากแรมบนซิงเกิลบอร์ดไปให้กับ หน่วยความจำของวงจร

3.2.4 FDC

ใช้เป็นสื่อกลางในการควบคุมการทำงานของ FDD (Floppy Disk Drive) ซึ่งจะเป็นตัวช่วยในการคำนวณ และควบคุมการทำงานของ FDD ให้ง่ายยิ่งขึ้นหน้าที่หลักของ FDC ก็จะเป็นตัวจัดการทุกๆ อย่างที่เกี่ยวข้องกับ FDD ตามคำสั่งของซีพียูไม่ว่าจะเป็นเรื่องการจัดสัญญาณควบคุม การอ่านค่าสถานะความพร้อม และผลจากการทำงานของ FDD ซึ่งรายละเอียดจะกล่าวถึงในหัวข้อของฟลอปปีดิสก์คอนโทรลเลอร์



รูปที่ 3.3 แผนผังการทำงานของวงจรภาค FDC

วงจรภาค FDC มีองค์ประกอบสำคัญซึ่งแสดงดังรูปที่ 3.3 มี 5 ส่วนดังนี้ คือ

1) FDC โปรเซสเซอร์ ดังที่ได้กล่าวแล้วว่าหน้าที่หลักของภาค ฟลอปปีดิสก์คอนโทรลเลอร์จะเป็นส่วนที่ใช้คำนวณ และควบคุมทุกๆ อย่างที่เกี่ยวข้องกับดิสก์ไครฟ์ โดยในวงจรที่ออกแบบขึ้นได้ใช้ โปรเซสเซอร์ที่สร้างขึ้นมาทำงานด้านนี้โดยตรง คือไอซีเบอร์ 8272A ของบริษัท Intel (เทียบได้กับเบอร์ μ PD765 ของ NEC และ SMSC) ซึ่งเป็น Single / Double Density ฟลอปปีดิสก์คอนโทรลเลอร์โดยตรง ดังนั้นการคำนวณการควบคุมฟลอปปีดิสก์ไครฟ์จะถูกส่งงานจากฟลอปปีดิสก์คอนโทรลเลอร์เป็นหลัก

2) วงจรกำเนิดสัญญาณนาฬิกา เป็นภาคที่ผลิต Clock Pulse ที่เหมาะสมให้แก่ FDC โปรเซสเซอร์ซึ่งเนื่องจาก FDC โปรเซสเซอร์ไม่มีวงจรกำเนิดสัญญาณนาฬิกาภายใน (Built in Clock Generator) จำเป็นต้องอาศัยวงจรกำเนิดสัญญาณนาฬิกาจากภายนอก อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) บัฟเฟอร์ ในส่วนนี้เป็นเพียงกันชนข่าวสารข้อมูลที่จะนำเข้าหรือออกจาก FDC โปรเซสเซอร์เพื่อป้องกันไม่ให้อาสาข้อมูลออกจาก FDC โปรเซสเซอร์ออกไปรบกวนข่าวสารข้อมูลที่เกิดจากอุปกรณ์อื่นๆในวงจรในขณะที่ซีพียูหลักไม่ได้ติดต่อกับ FDC โปรเซสเซอร์

4) ดีโค้ดเดอร์ เป็นการกำหนดตำแหน่งการทำงานของวงจรภาค FDC ซึ่งกำหนดให้มีการทำงาน 3 ตำแหน่งตามจำนวนรีจิสเตอร์ที่สำคัญของ FDC ดังนี้ คือ

4.1) รีจิสเตอร์แสดงสถานะ กำหนดไว้ที่ตำแหน่ง AFF0H

4.2) รีจิสเตอร์ข้อมูล กำหนดไว้ที่ตำแหน่ง AFF1H

4.3) เอาต์พุตรีจิสเตอร์ กำหนดไว้ที่ตำแหน่ง AFF3H

โดยวงจรภาค ดีโค้ดเดอร์ จะเป็นตัวสร้างสัญญาณเพื่อกำหนดรีจิสเตอร์ทำงานตามตำแหน่งของตัวมันเอง

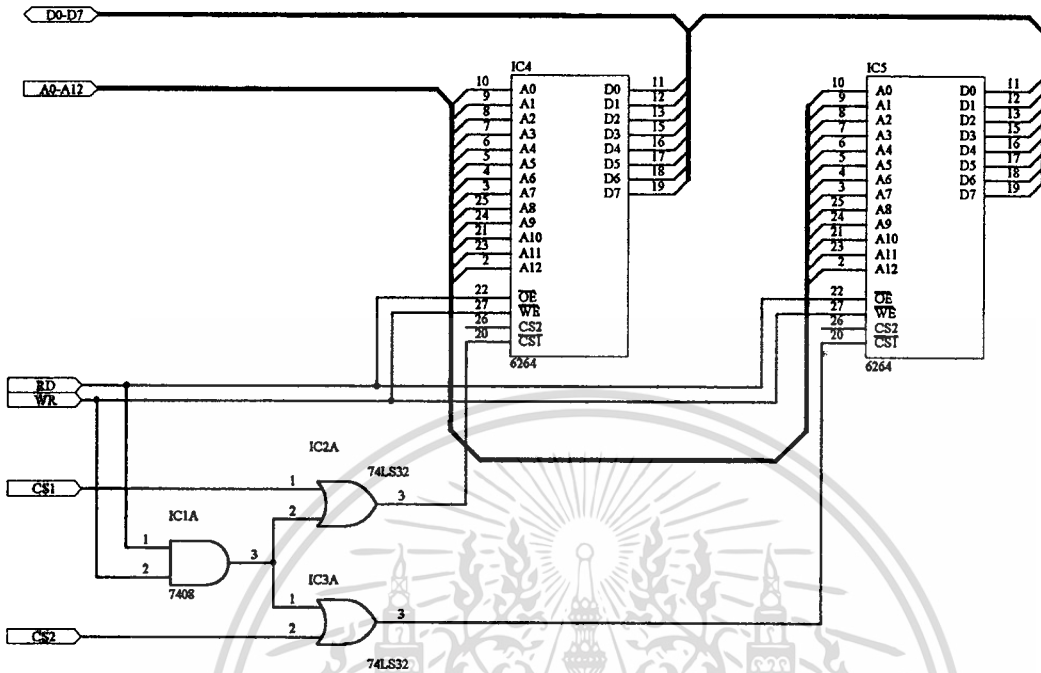
5) เอาต์พุตรีจิสเตอร์ เป็นรีจิสเตอร์ที่สร้างขึ้นจากภายนอก ถือเป็นองค์ประกอบหนึ่งของ FDC เนื่องจากโดยปกติแล้วในตัวของ FDC โปรเซสเซอร์นั้นจะมีรีจิสเตอร์ภายในเพียง 2 ตัว นั่นคือ รีจิสเตอร์แสดงสถานะ และ รีจิสเตอร์ข้อมูล ซึ่ง 2 ตัวนี้จะทำหน้าที่รับคำสั่งและส่งผลให้แก่ตัวซีพียูหลักในวงจรเท่านั้น แต่ยังไม่มีส่วนที่จะตอบการทำงานของมอเตอร์ ของฟลอปปีดิสก์ไดรฟ์ และเลือกว่า FDC ตัวใดเป็นตัวทำงาน เนื่องจาก FDC โปรเซสเซอร์เบอร์ 8272A นั้นสามารถที่จะควบคุม FDC ได้ 4 ตัวด้วยกันจึงจำเป็นต้องมีตัวกำหนดว่าการทำงานเลือกที่จะควบคุม FDC ตัวใด

3.3 ภาคนวความจำ

ในภาคของหน่วยความจำจะใช้ไอซีเบอร์ 6264 ซึ่งเป็นแรมขนาด 8 k จำนวน 2 ตัว ซึ่งออกแบบให้มีการ ดีโค้ดเดอร์ ที่ ตำแหน่ง 6000H-7FFFH และ 8000H-9FFFH ของไอซีตัวที่ 1 และ 2 ตามลำดับ ซึ่งจะใช้เป็นทีก์เก็บข้อมูลในระหว่างการนำข้อมูลจาก Z-80 มาเก็บไว้, ใช้ในการเขียนข้อมูลลงในดิสก์ และใช้ในการนำข้อมูลที่ อ่านได้จากดิสก์ไปเก็บไว้เพื่อที่จะนำไปเก็บไว้ในแรมของ Z-80 ในสภาวะการอ่านข้อมูลจากดิสก์ ดังรูปที่ 3.4

3.4 ภาคควบคุม (Controller)

ในการประมวลผลซึ่งใช้ไมโครโปรเซสเซอร์ MCS-51 ซึ่งทำหน้าที่ในการนำข้อมูลจาก Z-80 ไปส่งให้อ่าน-เขียนดิสก์ไดรฟ์ ซึ่งได้ทดลองและการออกแบบไมโครโปรเซสเซอร์ MCS-51 เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ดังรูปที่ 3.5



รูปที่ 3.4 วงจรภาคหน่วยความจำ

3.5 FDC โปรเซสเซอร์

ในภาค FDC หัวใจหลักของวงจร ได้แก่ ไมโครโปรเซสเซอร์เบอร์ 8272A ซึ่งเป็นไมโครโปรเซสเซอร์ที่ถูกออกแบบให้เป็น FDC โดยเฉพาะ ไมโครโปรเซสเซอร์เบอร์ดังกล่าวถูกนำมาใช้กับเครื่องคอมพิวเตอร์ PC ในช่วง 8080 ถึง 80386 ซึ่งเป็นคอมพิวเตอร์ขนาด 16 บิต โดย 8272A เป็นไมโครโปรเซสเซอร์ขนาด 40 ขาที่มีวงจรที่เกี่ยวข้องกับการควบคุมดิสก์ไดรฟ์เกือบครบ โดยมีการเสริมวงจรประกอบเพิ่มเข้าไปในบางส่วน

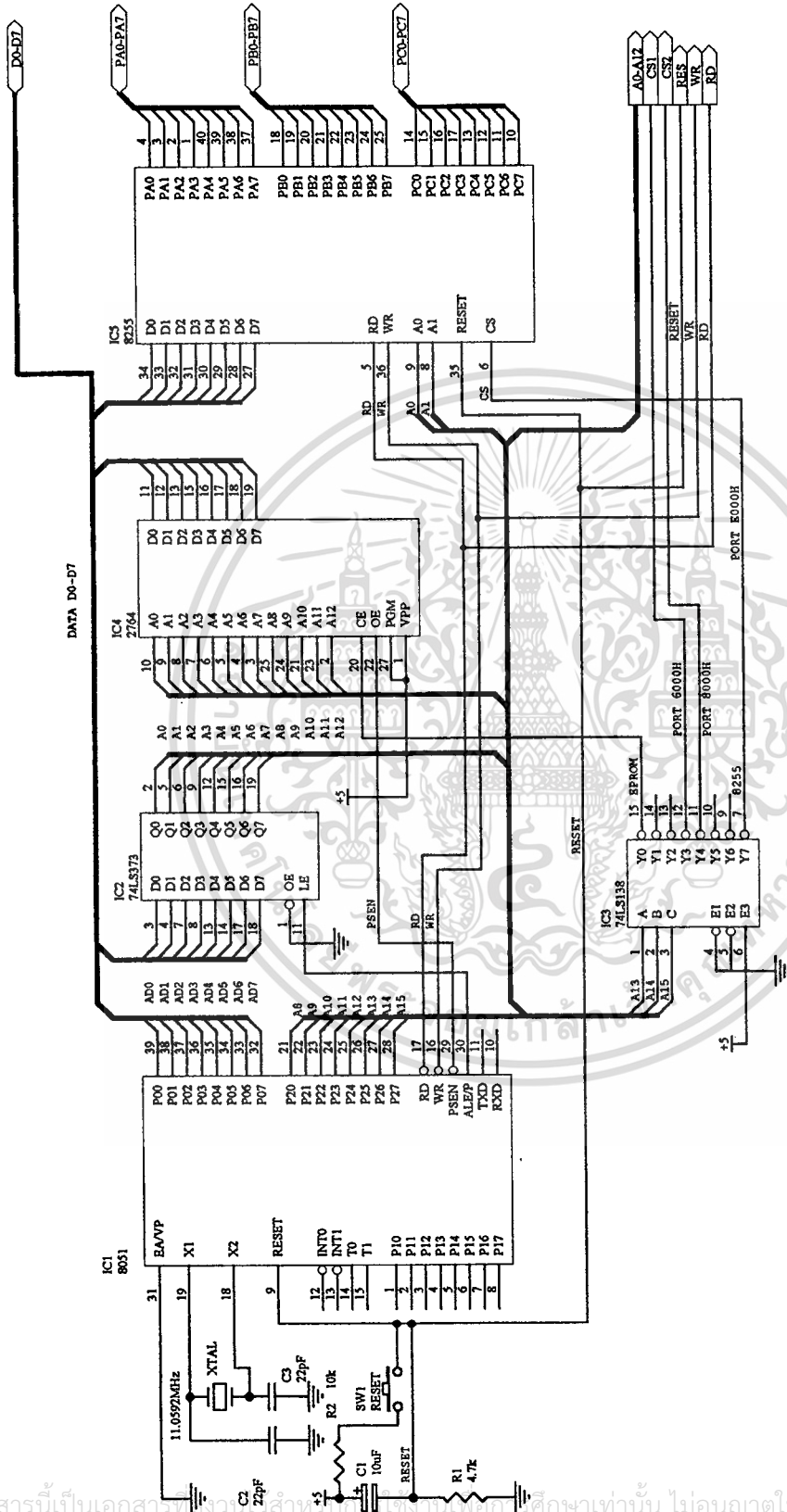
อันดับแรกจะกล่าวถึงส่วนของรีจิสเตอร์ หรือแหล่งข้อมูลของไมโครโปรเซสเซอร์เบอร์ 8272A ซึ่งคือ U4 ในวงจรที่อ้างอิง โดยต่อไปนี้จะเรียกไมโครโปรเซสเซอร์เบอร์ 8272A ว่า U4 เพื่อความกระชับรัดในการอธิบายการทำงาน

รีจิสเตอร์ของ U4 หรือกล่าวได้ว่าเป็นส่วนของฐานข้อมูลหรือส่วนที่เชื่อมต่อเข้ากับบัสข้อมูลของระบบ จะมี 2 ชนิดด้วยกัน คือ

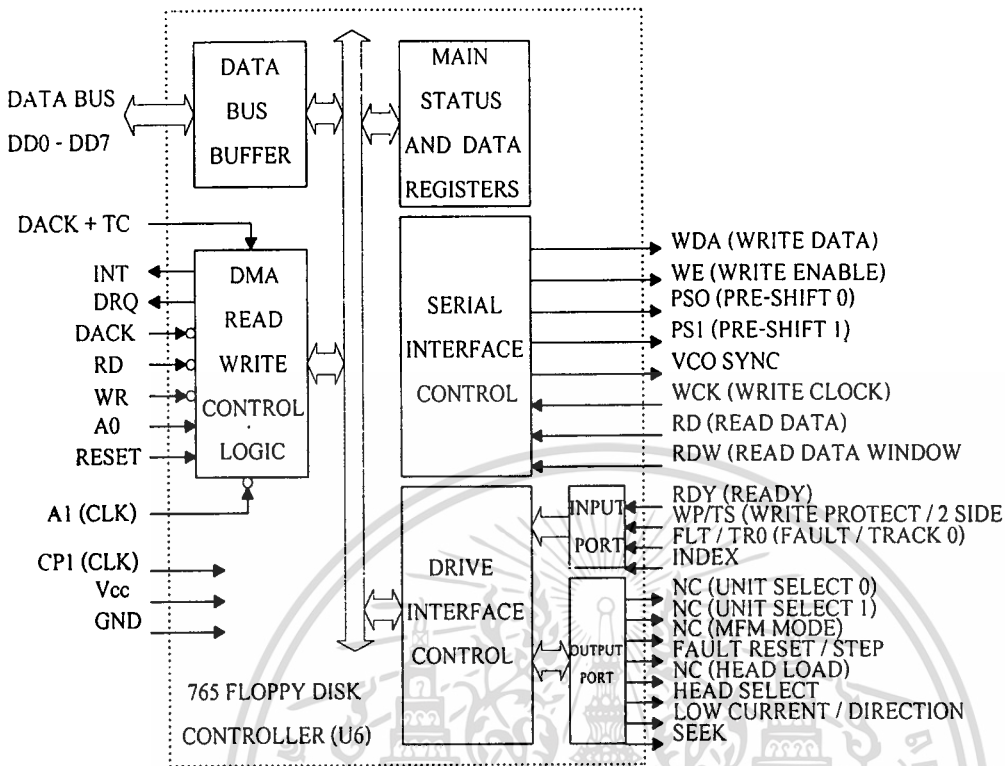
1. รีจิสเตอร์ข้อมูล

2. รีจิสเตอร์แสดงสถานะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 การทดลองและการออกแบบไมโครโปรเซสเซอร์ MCS-51



รูปที่ 3.6 แผนผังการทำงานภายในของไมโคร โปรเซสเซอร์เบอร์ 8272A

โดยรีจิสเตอร์ 2 ชนิดนี้จะอยู่ในบัสข้อมูลชุดเดียวกัน แต่ตำแหน่งการทำงานต่างกันซึ่งจะเห็นว่า U4 จะมีขา A0 เป็นขาตำแหน่งเพียงตัวเดียว ซึ่งจะใช้ในการเลือกรีจิสเตอร์ 2 ตัว ซึ่งบิต 1 บิต จะสามารถแทนค่าตำแหน่งได้เพียง 2 ตำแหน่ง ตามสมการ 2^N ดังนั้น เมื่อแทนค่า 1 เข้าไป จะได้

$$2^N = 2^1 = 2 \text{ ตำแหน่ง} \quad (3.1)$$

ซึ่งใช้แทนค่ารีจิสเตอร์ 2 ตัวดังกล่าว โดยที่เมื่อ A0 มีค่าเป็นสถานะลอจิก "0" จะเป็นการติดต่อกับรีจิสเตอร์แสดงสถานะและเมื่อ A0 มีค่าเป็นสถานะลอจิก "1" เป็นการติดต่อกับข้อมูลรีจิสเตอร์

จากตารางที่ 3.1 จะเห็นว่าเมื่อเอาสัญญาณอ่านและเขียนเข้ามาสัมพันธ์กับสัญญาณ A0 จะได้สถานะการทำงาน 3 แบบ คือ การอ่านข้อมูลจากรีจิสเตอร์แสดงสถานะ, การอ่านข้อมูลจากรีจิสเตอร์ข้อมูล และการเขียนข้อมูลจากรีจิสเตอร์ข้อมูลซึ่งจะสังเกตได้ว่ารีจิสเตอร์แสดงสถานะจะสามารถอ่านข้อมูลได้เพียงอย่างเดียวเท่านั้น

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.1 การทำงานของ U4 ในสถานะต่างๆ

A0	RD	WR	สถานะการทำงาน
0	0	1	อ่านข้อมูลรีจิสเตอร์แสดงสถานะ
1	0	1	อ่านข้อมูลรีจิสเตอร์ข้อมูล
1	1	0	เขียนข้อมูลรีจิสเตอร์ข้อมูล

หน้าที่ของรีจิสเตอร์แต่ละตัวจะมีหน้าที่ต่างกันดังนี้ คือรีจิสเตอร์ข้อมูลจะเป็นตัวสื่อสารข้อมูล รวมทั้งคำสั่งและผลที่เกิดจากการการทำงานทำให้ชิพียูหลักรับทราบ ในขณะที่รีจิสเตอร์แสดงสถานะจะเรียกว่ารีจิสเตอร์แสดงสถานะหลักเพื่อไม่ให้ซ้ำซ้อนในการเรียกชื่อรีจิสเตอร์แสดงสถานะซึ่งใช้เรียกเป็นหน่วยหนึ่งของการอ่านค่าผลของรีจิสเตอร์ข้อมูลดังนั้นเมื่อกล่าวถึงรีจิสเตอร์แสดงสถานะหลักจะหมายถึงการอ้างตำแหน่งของ A0 เป็นสถานะลอจิก "0" นั่นเอง สำหรับรีจิสเตอร์แสดงสถานะหลักจะเป็นส่วนที่แสดงสภาวะการทำงานหลักๆ โดยทั่วไปของ U4 เช่น การพร้อมรับคำสั่ง การแสดงทิศทางของข้อมูลข่าวสารที่ชิพียูติดต่อกับรีจิสเตอร์ข้อมูลของ U4 แสดงรายละเอียดการแสดงสถานะในแต่ละบิตของรีจิสเตอร์แสดงสถานะหลักดังตารางที่ 3.2

ตารางที่ 3.2 รายละเอียดการแสดงสภาวะของรีจิสเตอร์หลักในแต่ละบิต

บิต	ชื่อ	สถานะการทำงาน
0	Drive 0 Busy	Drive 0 อยู่ในสภาวะ Seek Mode
1	Drive 1 Busy	Drive 1 อยู่ในสภาวะ Seek Mode
2	Drive 2 Busy	Drive 2 อยู่ในสภาวะ Seek Mode
3	Drive 3 Busy	Drive 3 อยู่ในสภาวะ Seek Mode
4	FDC Busy	คำสั่งในการอ่าน/เขียน อยู่ในระหว่างการปฏิบัติ
5	Non DMA Mode	FDC อยู่ในสภาวะการทำงานแบบ Non DMA
6	Data I/O	ทิศทางการรับส่งข้อมูลระหว่าง FDC กับชิพียูโดยหากบิตนี้มีสถานะเป็น "1" หมายถึงกำลังส่งข้อมูลจาก FDC ให้กับชิพียูและเมื่อเป็น "0" หมายถึงการส่งข้อมูลจากชิพียูให้แก่ FDC
7	Master RQST	แสดงว่าขณะนี้ FDC พร้อมที่จะรับหรือส่งข้อมูลกับตัวชิพียู

ในลักษณะการรับส่งข้อมูลกับรีจิสเตอร์ทั้งสองตัวนี้จะแตกต่างจากการออกคำสั่งการทำงานกับไมโครโปรเซสเซอร์เบอร์อื่นๆ กล่าวคือ การนำคำสั่งมาทำงานจะเป็นลักษณะของวัฏจักร (Machine Cycle) โดยการนำเอาคำสั่ง (Fetch) มาอ่านทีละคำสั่งโดยเรียงตำแหน่งไปเรื่อยๆ เมื่ออ่านคำสั่ง 1 คำสั่งแล้วจึงนำมาปฏิบัติงานไปครั้งละคำสั่ง (Execution) แต่ในกรณีของไมโครโปรเซสเซอร์เบอร์ 8272A นั้น การป้อนคำสั่งการทำงานจะไม่เป็นการกำหนดค่าลงเป็นตำแหน่งเรียงกันไป แต่จะเป็นการป้อนคำสั่งแบบสแตค (Stack) คือ ป้อนคำสั่งเข้าไปที่ตำแหน่งเดียวกันโดยไมโครโปรเซสเซอร์ 8272A จะรับคำสั่งทั้งหมดก่อนจึงเริ่มปฏิบัติงานโดยเอาคำสั่งทั้งหมดไปปฏิบัติงาน เช่น การสั่งให้ FDD ทำการเขียนข้อมูลก็จะป้อนคำสั่งไป 8 คำสั่ง เมื่อ U4 รับคำสั่งแล้วนำไปปฏิบัติงานจะส่งค่า Result กลับมา 9 ค่า จากตำแหน่งเดียวกัน ทั้งหมดที่กล่าวมาเป็นการกล่าวทำให้ทราบถึงลักษณะการทำงานของรีจิสเตอร์ภายในของ U4 ซึ่งจะกล่าวรายละเอียดในส่วนของการเขียนโปรแกรมควบคุมต่อไป

สำหรับหน้าที่ของไมโครโปรเซสเซอร์เบอร์ 8272A นี้จะมี 6 หน้าที่หลักๆ คือ

1) ควบคุมการเลือก FDD เนื่องจากคุณสมบัติของไมโครโปรเซสเซอร์เบอร์ 8272A นั้นสามารถควบคุมฟลอปปีดิสก์ไดรฟ์ถึง 4 ตัวด้วยกัน ดังนั้นมันจะทำการเลือกควบคุมว่าจะให้ดิสก์ไดรฟ์ตัวใดเป็นตัวทำงาน แต่จะเลือกใช้ฟลอปปีดิสก์ไดรฟ์เพียงตัวเดียวเท่านั้นจึงไม่สำคัญมากนัก

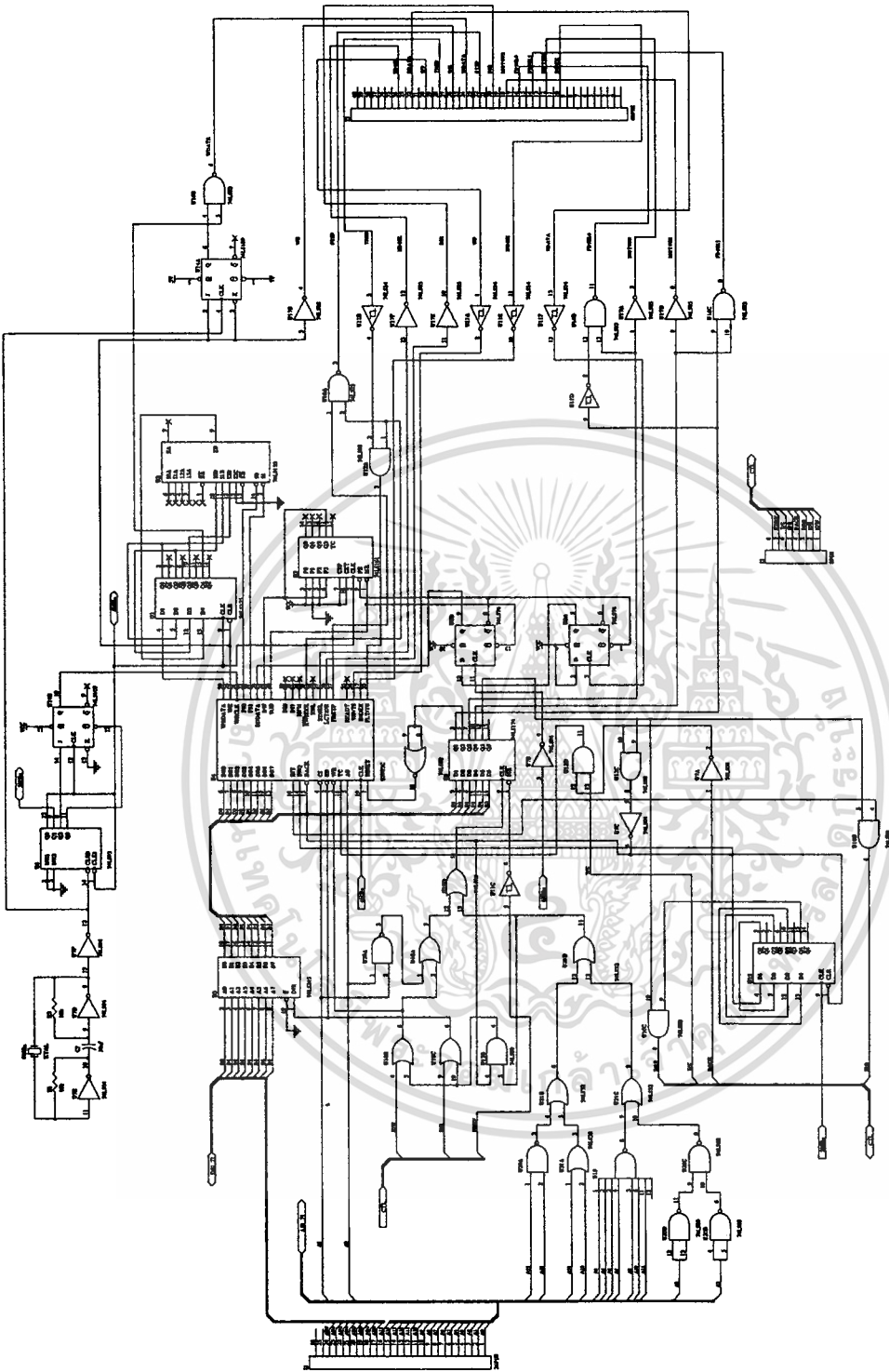
2) ควบคุมการเลือกแทร็คเป็นตัวควบคุมตำแหน่งของหัวเขียนหรืออ่านของฟลอปปีดิสก์ไดรฟ์เลื่อนไปยังแทร็คที่ต้องการบนแผ่นฟลอปปีดิสก์ถูกต้องตามกำหนด ช่วยทำให้ไม่ต้องมีการคำนวณค่าเวลา และจำนวนสัญญาณนาฬิกาที่จะต้องเข้าไปควบคุมฟลอปปีดิสก์ไดรฟ์

3) คอยแสดงสถานะ (Monitor) ของข้อมูลแทร็ค

4) ควบคุมระยะเวลาที่ใช้เตรียมความพร้อมของหัวเขียนหรืออ่านของฟลอปปีดิสก์ไดรฟ์

5) แยกสัญญาณข้อมูลรวม (Composite Signal) ซึ่งประกอบด้วยคาบสัญญาณนาฬิกา (Clock) และสัญญาณข้อมูลออกจากกัน

6) ตรวจสอบความผิดพลาดในการอ่าน, เขียนข้อมูลลงบนแผ่นฟลอปปีดิสก์ไดรฟ์ เรียกได้ว่าไมโครโปรเซสเซอร์ 8272A ตัวเดียวก็แทบจะครอบคลุมการทำงานของภาค FDC ทั้งหมดดังที่ได้กล่าวมาแล้วข้างต้นรูปที่ 3.7 แสดงวงจรในส่วนของภาค FDC

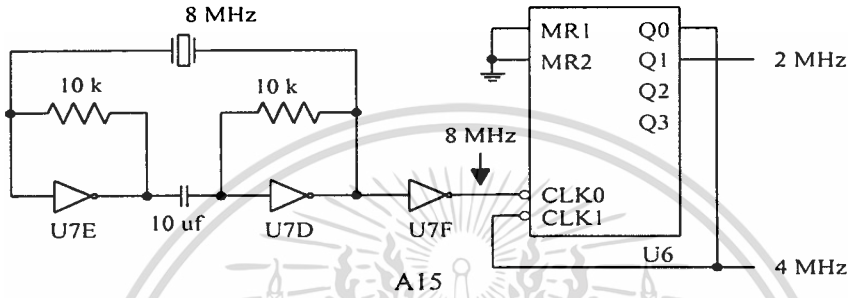


รูปที่ 3.7 วงจรภาค FDC

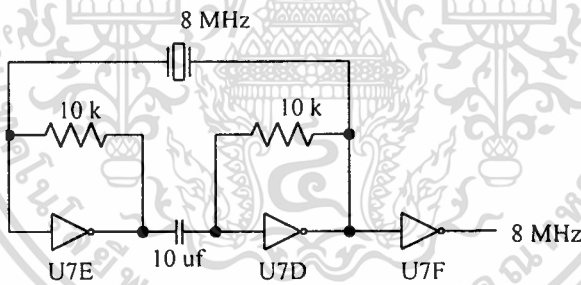
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6 วงจรกำเนิดสัญญาณนาฬิกา

ตามคุณสมบัติทั่วไปของไมโครโปรเซสเซอร์ คือ ไม่มีการสร้างสัญญาณนาฬิกาภายในตัวเอง ซึ่งเป็นข้อแตกต่างหนึ่งกับไมโครคอนโทรลเลอร์ ดังนั้นจึงจำเป็นที่จะต้องสร้างวงจรถ่ายทอดสัญญาณนาฬิกาขึ้นภายนอก

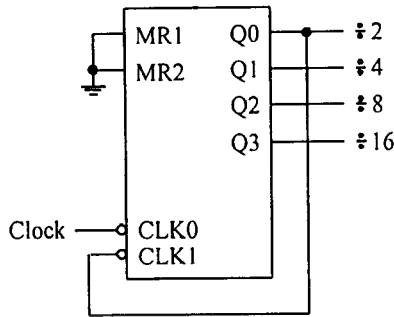


รูปที่ 3.8 วงจรกำเนิดสัญญาณนาฬิกา



รูปที่ 3.9 วงจรกำเนิดสัญญาณนาฬิกา 8 MHz

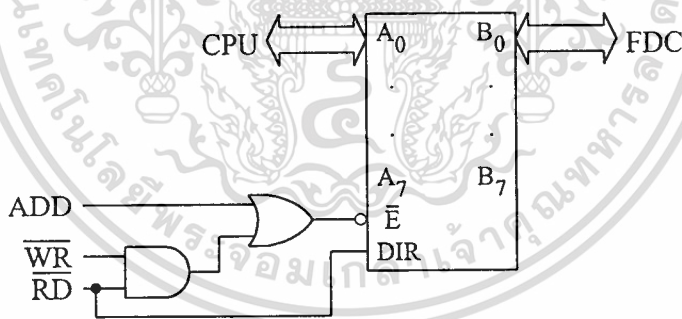
ในวงจรภาค FDC จะใช้สัญญาณนาฬิกา 3 ความถี่ด้วยกัน ดังนี้ คือ 2 MHz, 4 MHz และ 8 MHz โดยตัวกำเนิดสัญญาณนาฬิกาหลักใช้ U7 ประกอบด้วยคริสตอลความถี่ 8 MHz สร้างเป็นความถี่ขนาด 8 MHz ขึ้นมาเป็นความถี่หลัก ดังแสดงในรูปที่ 3.8 การออกสิจเลขความถี่โดยใช้คริสตอลเป็นตัวควบคุมให้ความถี่ที่ได้ทางด้านเอาต์พุตมีเสถียรภาพกว่าการใช้ R, C เป็นตัวออกสิจเลข และเมื่อได้ความถี่ 8 MHz แล้วความถี่ที่เหลือ คือ 2 MHz และ 4 MHz สามารถสร้างขึ้นมาโดยการนำเอาความถี่ 8 MHz มาหารโดยใช้ไอซีประเภทเคาน์เตอร์มาเป็นตัวหาร ซึ่งในที่นี้จะใช้ไอซีเบอร์ 74LS93 4 บิต ไบนารีเคาน์เตอร์โดยต่อ Q0 เข้ากับ CLK2 จะทำให้ไอซีนับค่าต่อเนื่อง ดังนั้นจะได้ค่าทางเอาต์พุตของไอซี U6 ดังรูปที่ 3.9



รูปที่ 3.10 ค่าที่ได้จากการหารสัญญาณนาฬิกาในแต่ละเอาต์พุต

ดังนั้นที่ 2 MHz และ 4 MHz จะได้จากการนำเอาสัญญาณ 8 MHz มาหาร 4 และหาร 2 ตามลำดับ ดังนั้นจะได้สัญญาณนาฬิกาครบทั้ง 3 ความถี่

3.7 ภาคบัฟเฟอร์



รูปที่ 3.11 วงจรภาคบัฟเฟอร์

ดังที่กล่าวมาแล้วว่าบัฟเฟอร์เป็นเสมือนกันชนระหว่างข่าวสารข้อมูล โดยเป็นตัวป้องกันไม่ให้มีข้อมูลออกมาเมื่อไม่ได้ทำการติดต่อกับ FDC และไม่ให้ข้อมูลที่ไม่ได้เป็นของ FDC ส่งผ่านไป

อีกประเด็นหนึ่งที่สำคัญ คือชิพที่ถูกเลือกใช้ในโครงงานนี้เป็นไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งจะมีการรวมตัวกันระหว่างข้อมูลกับตำแหน่ง 8 บิตล่างเข้าด้วยกัน โดยแยกข้อมูลกับตำแหน่งออก โดยการใช้อุปกรณ์ภายนอกช่วยอีกครั้งหนึ่ง การที่จะทราบว่าขณะนี้สิ่งที่

MCS-51 ส่งออกมาเป็นข้อมูลหรือตำแหน่ง โดยการพิจารณาที่สัญญาณการอ่านและการเขียนหากมีตัวใดตัวหนึ่งเกิดเป็นสถานะลอจิก "0" (Active) หมายถึงสิ่งที่ปรากฏเป็นข้อมูล แสดงสถานะดังตารางที่ 3.3

ตารางที่ 3.3 สถานะการทำงานของสัญญาณอ่าน, เขียน และสัญญาณตำแหน่ง

RD	WR	ADD	สถานะการทำงาน
0	1	0	อ่านข้อมูลจาก FDC
1	0	0	เขียนข้อมูลจาก FDC

จะเห็นว่า Address Decoder จะต้องแอกทีฟเป็น "0" เท่านั้น แสดงว่าเป็นข้อมูลที่ต้องการติดต่อกับ FDC ซึ่งถ้าหาก U5 ทำงานได้ก็ต่อเมื่อขา Enable เป็น "0" ดังนั้นมีตัวแปร 3 ตัว คือ RD, WR และ ADD เมื่อนำเอาตัวแปรทั้ง 3 มาเข้าสมการลอจิก

$$F = (RD \cdot WR) + ADD \quad (3.2)$$

จะได้ส่วนที่แอกทีฟ "0" ทางด้านเอาต์พุตเพียง 2 สถานะดังตารางที่ 3.3 อีกจุดหนึ่ง คือ ที่ขา DIR มีสถานะเป็น "1" จะทำให้ข้อมูลด้าน A (CPU) ถูกส่งเข้าไปด้าน B (FDC) และข้อมูลจากทางด้าน B จะถูกส่งมายังด้าน A เมื่อ DIR เป็น "0" ด้วยเหตุที่ว่าต้องการให้มีข้อมูลออกจากภาค FDC ก็ต่อเมื่อมีการอ่านข้อมูลเท่านั้น ในกรณีอื่นๆ จะไม่อนุญาตให้มีข้อมูลออก ดังนั้นจึงสามารถต่อสัญญาณ RD เข้ากับขา DIR ได้โดยตรง และส่งผลให้ข้อมูลจากทางด้าน B ส่งออกทางด้าน A ในกรณีที่มีการอ่านข้อมูลโดยอัตโนมัติ

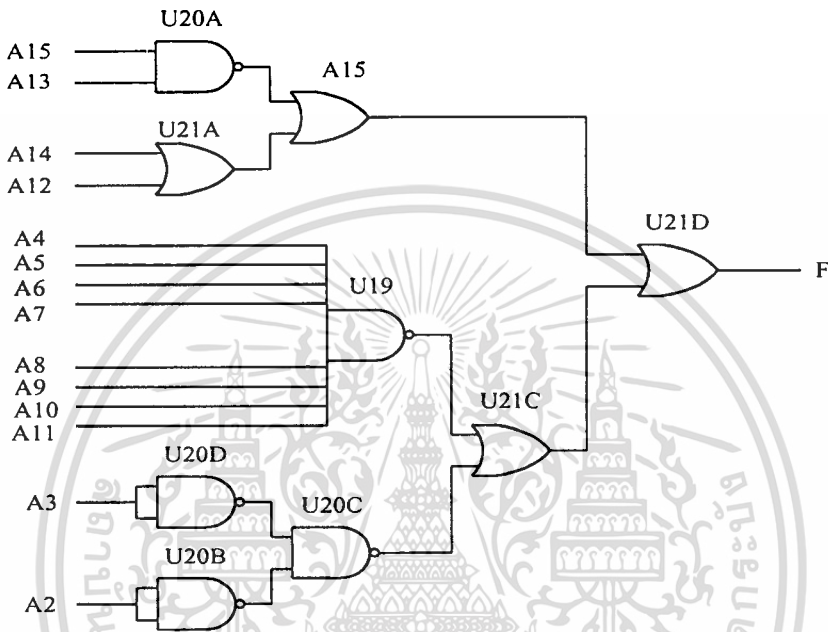
3.8 ดีโค้ดเดอร์

หรือเรียกว่าวงจรกำหนดตำแหน่งการทำงานวงจรถูกออกแบบคล้ายๆ กับวงจรดีโค้ดเดอร์ โดยทั่วไป ซึ่งในวงจรภาค FDC มีตำแหน่งการทำงาน 3 ตำแหน่ง โดยกำหนดไว้ดังนี้คือ

1. AFF0H - แสดงสถานะรีจิสเตอร์
2. AFF1H - รีจิสเตอร์ข้อมูล

เอกสารนี้ 3. AFF3H - เอาต์พุตรีจิสเตอร์ใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งจะเห็นว่าการกำหนดการทำงานทั้ง 3 ตำแหน่งจะมีข้อมูล 14 บิตเหมือนกัน คือ 1010 (A) 1111 (F) 1111(F) และ 00XX ดังนั้นการกำหนดว่าเป็นการสั่งงานที่ภาค FDC จะกำหนดตำแหน่งทั้ง 14 บิต คือ 101011111100B จะได้วงจรภาคดีโคเดอร์ดังนี้



รูปที่ 3.12 วงจรภาคดีโคเดอร์

ตารางที่ 3.4 การกำหนดค่าบิตที่ทำงานในแต่ละบิต

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2
1	0	1	0	1	1	1	1	1	1	1	1	0	0

จากวงจรในรูปที่ 3.12 ได้สมการทางด้านเอาต์พุตดังนี้ คือ

$$F = [(A15 \cdot A13) + (A14 + A12)] + [(A4 \cdot A5 \cdot A6 \cdot A7 \cdot A8 \cdot A9 \cdot A10 \cdot A11) + (A3 \cdot A2)] \quad (3.3)$$

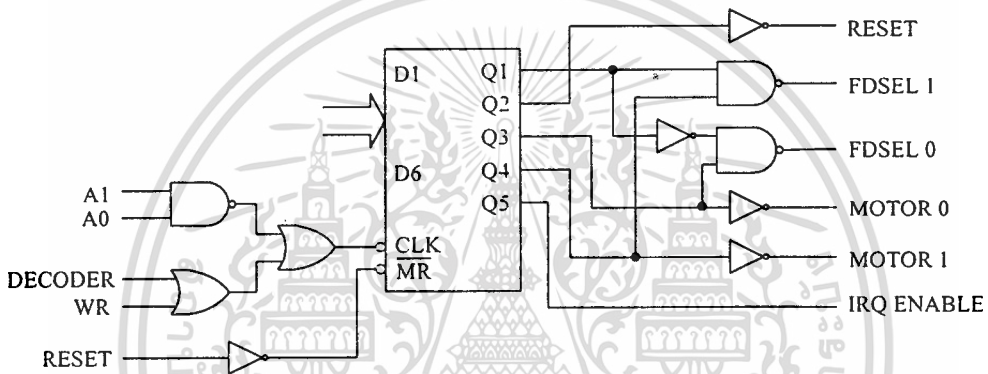
โดยกำหนดให้ F มีค่าแอกทีฟที่ลอจิก "0" ทดลองแทนค่าสภาวะลอจิกที่ทำงานจากตารางที่

เอกสารนี้เป็นเอกสารที่ควรรักษาไว้สำหรับใช้ในการงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
3.4 ของภาค FDC ลงในสมการ จะได้
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\begin{aligned}
 F &= [(1 \cdot 1) + (0 + 0)] + [1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1] + (0 \cdot 0) \\
 &= [0 + 0] + [0 + 0] \\
 &= 0
 \end{aligned}$$

ซึ่งสัญญาณลอจิก "0" ที่ได้ทางเอาต์พุตของวงจรดีโค้ดเดอร์จะนำไปประกอบกับส่วนอื่นๆ เช่น สร้างเป็นสัญญาณ IOR, IOW เป็นต้น

3.9 ภาควงจรเอาต์พุตรีจิสเตอร์



รูปที่ 3.13 วงจรเอาต์พุตรีจิสเตอร์

ในส่วนของเอาต์พุตรีจิสเตอร์จัดเป็นองค์ประกอบหนึ่งของภาควงจร FDC โปรเซสเซอร์ที่ทำงานได้สมบูรณ์แบบยิ่งขึ้น ซึ่งภาควงจรนี้จัดได้ว่าเป็นจุดต่อของไมโครโปรเซสเซอร์เบอร์ 8272A ซึ่งในระยะหลังๆ นี้ FDC โปรเซสเซอร์จะรวมเอาวงจรจัดการเกี่ยวกับเอาต์พุตรีจิสเตอร์รวมเข้ากับตัวไมโครโปรเซสเซอร์เลย ซึ่งทำให้วงจรมีขนาดเล็กลง และ FDC โปรเซสเซอร์สามารถต่อเข้ากับฟลอปปี้ดิสก์ไดรฟ์ได้โดยตรง แต่ราคาของไมโครโปรเซสเซอร์แบบที่มีเอาต์พุตรีจิสเตอร์ภายในจะมีราคาที่สูงกว่า (ประมาณ 4 เท่า) เมื่อเทียบกับแบบที่ไม่มีเอาต์พุตรีจิสเตอร์ภายใน

ในภาควงจรเอาต์พุตรีจิสเตอร์จะประกอบด้วย 74LS174 เป็นหัวใจหลักในการทำงาน ซึ่ง 74LS174 หรือ U9 นี้เป็น D-ฟลิปฟล็อปจะรับและส่งข้อมูลเมื่อได้รับสัญญาณนาฬิกาที่ขา CLK จากรูปที่ 3.10 จะเห็นว่าสัญญาณ CLK ของ U9 จะถูกสร้างขึ้นมาจาก 4 สัญญาณด้วยกัน คือ A1, A0, ดีโค้ดเดอร์ และ WR โดยกำหนดให้สัญญาณทั้ง 4 สัญญาณนี้จะต้องแอสท์ฟทั้งหมด กล่าวคือ

จะต้องมีลอจิกดังตารางที่ 3.5 สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.5 สภาวะลอจิกของสัญญาณที่นำมาสร้างเป็นสัญญาณนาฬิกา

A1	A0	Decoder	WR
1	1	0	0

เมื่อสภาวะลอจิกเป็นไปตามที่กำหนดจะทำให้ข้อมูลทางด้านอินพุต (D1–D6) ไปปรากฏอยู่ทางด้านเอาต์พุต (Q1 – Q6) ได้

ทางด้านเอาต์พุตจะมีการนำบิตแต่ละบิตไปต่อใช้งาน โดยแยกอธิบายแต่ละบิต ดังตารางที่

3.6

ตารางที่ 3.6 การกำหนดบิตการทำงานของเอาต์พุตรีจิสเตอร์

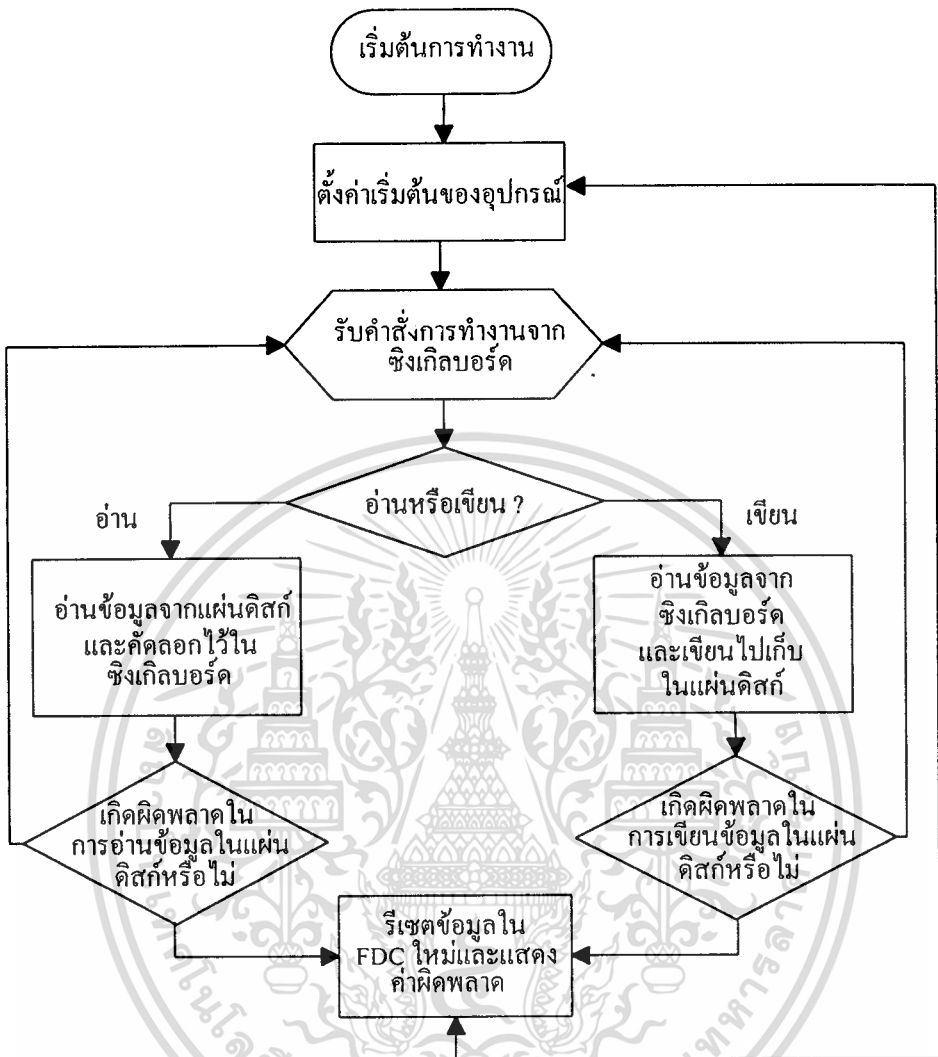
บิต	ชื่อ	หน้าที่
Q1	FDSEL	เป็นบิตที่ใช้ในการเลือกจะให้ FDD ตัวใดเป็นตัวทำงาน โดยเมื่อบิตนี้เป็น "0" เลือกไครฟ์ 0 "1" เลือกไครฟ์ 1
Q2	RESET	เป็นบิตที่ใช้ในการเลือกจะให้ FDD หยุดทำงาน และเริ่มต้นสภาวะของตัวเองเป็นค่าเริ่มต้นใหม่อีกครั้ง
Q3	MT0	เป็นบิตที่กำหนดให้มอเตอร์ใน FDD ตัวที่ 0 ทำงาน
Q4	MT1	เป็นบิตที่กำหนดให้มอเตอร์ใน FDD ตัวที่ 1 ทำงาน
Q5	IREQ	อนุญาตให้มีการใช้ DMA

3.10 การทำงานภาคซอฟต์แวร์

ในส่วนซอฟต์แวร์ที่ใช้ในการควบคุม FDC คอนโทรลเลอร์นั้นถือได้ว่าเป็นสมองส่วนหนึ่งที่จะช่วยให้การควบคุมดิสก์ไครฟ์เป็นไปอย่างเต็มประสิทธิภาพ

โปรแกรมการทำงานของ FDC คอนโทรลเลอร์นั้นเริ่มจากการตั้งค่าสภาวะเริ่มต้น การเตรียมความพร้อมก่อนการใช้งานเช่นเดียวกับไมโครโปรเซสเซอร์ชนิดอื่นๆ โดยมีแผนผังการทำงานของภาคควบคุมดังรูปที่ 3.11

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



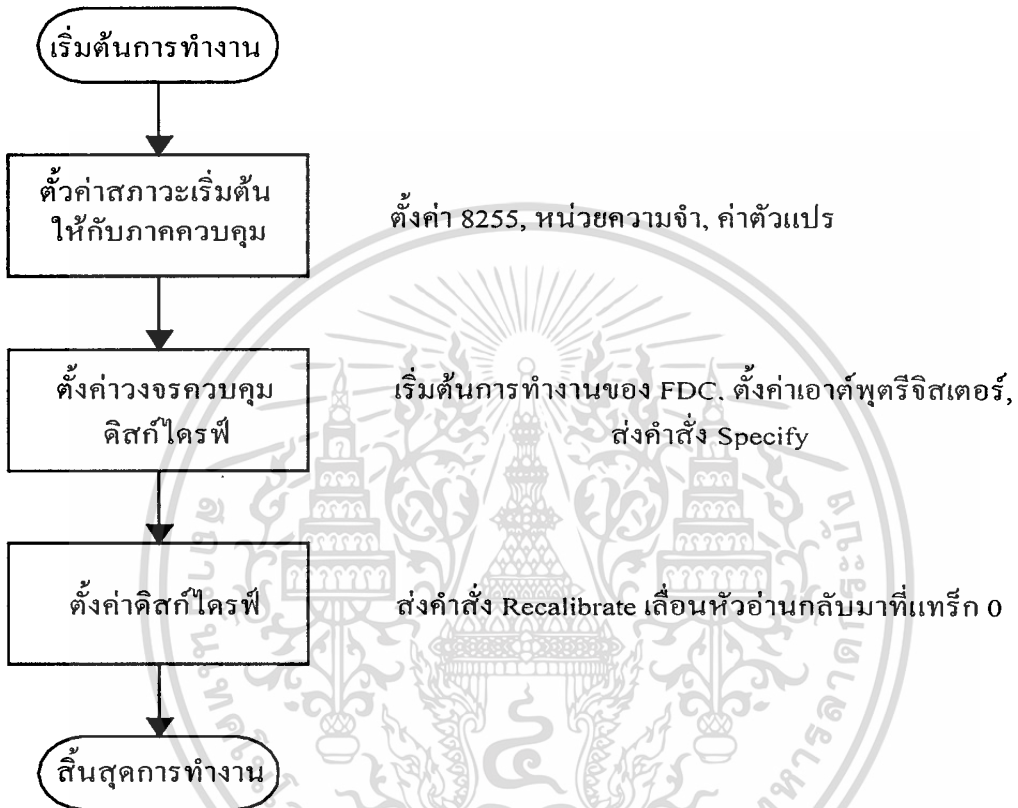
รูปที่ 3.14 แผนผังการทำงานหลักภาค FDC

การออกแบบส่วนของซอฟต์แวร์ที่ใช้ในการควบคุม ฟลอปปีดิสก์คอนโทรลเลอร์นั้นจะเริ่มจากการตั้งค่าสถานะของฮาร์ดแวร์ในระบบทั้งหมด เพื่อให้ฮาร์ดแวร์มีความพร้อมที่จะตอบสนองคำสั่งการทำงานเช่น การตั้งค่าตำแหน่งหัวอ่านของ FDD (Floppy Disk Drive) หรือการตั้งค่าของไอซีต่างๆ เช่น 8255 เป็นต้น

เมื่อตั้งค่าต่างๆ ของฮาร์ดแวร์เป็นที่เรียบร้อยแล้ว สิ่งที่ทำต่อคือ การรอรับคำสั่งจาก Z-80 หรือซิงเกิลบอร์ด ว่าต้องการที่จะทำการอ่านข้อมูลจากแผ่นฟลอปปีดิสก์มาเก็บในหน่วยความจำของซิงเกิลบอร์ด หรือต้องการนำเอาข้อมูลที่มีอยู่ในหน่วยความจำของซิงเกิลบอร์ด มาเก็บไว้ในแผ่นฟลอปปีดิสก์ เมื่อรับคำสั่งการทำงานเข้ามาแล้ว จะให้วงจรปฏิบัติตามคำสั่ง โดยเป็นรูปแบบการอ่าน/เขียน เมื่อการปฏิบัติงานในการอ่านหรือเขียนผ่านไปแล้ว จะมีการประเมินผลการทำงานว่า

เสร็จสมบูรณ์หรือไม่ โดยถ้าหากมีข้อผิดพลาดก็จะให้มีการแสดงค่าผิดพลาดขึ้นมาในรูปแบบของไฟแสดงผล โดยการออกคำสั่งไปยังฮาร์ดแวร์

3.10.1 การตั้งค่าเริ่มต้นฮาร์ดแวร์



รูปที่ 3.15 แผนผังการตั้งค่าเริ่มต้นของวงจร

การตั้งค่าเริ่มต้น เริ่มจากการตั้งค่าสถานะเริ่มต้นให้กับภาคควบคุมซึ่งได้แก่ 8255, หน่วยความจำ. ค่าตัวแปรเริ่มต้นต่างๆ ของวงจร เนื่องจากคอนโทรลเลอร์เป็นส่วนที่จะทำการรองรับข้อมูลของวงจรทั้งหมด จึงเป็นส่วนแรกที่จะทำการตั้งค่าเริ่มต้น จากนั้นจึงเริ่มทำการ ตั้งค่าเริ่มต้นการทำงานของวงจรภาค FDC ซึ่งจะทำการรีเซตการทำงานของ FDC โปรเซสเซอร์ซึ่งจะทำให้ FDC โปรเซสเซอร์พร้อมที่จะทำงาน และพร้อมที่จะรับคำสั่งจากคอนโทรลเลอร์ จากนั้นจึงเริ่มทำการตั้งค่าเริ่มต้นของเอาต์พุตริจิสเตอร์ ซึ่งจะมีผลต่อการทำงานของวงจร แล้วจึงทำการตั้งค่าคาบเวลาการทำงานของหัวอ่านให้แก่ FDC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในลำดับต่อมาคือ การเลื่อนตำแหน่งของหัวอ่านกลับมาที่ตำแหน่ง แทร็ก 0 เพื่อเป็นค่าของหัวอ่านเริ่มต้น เนื่องจากเมื่อเปิดเครื่องครั้งแรก FDC โปรเซสเซอร์ยังไม่มีค่าตำแหน่งที่อยู่ของหัวอ่านของ ฟลอปปีดิสก์ใด ๆ จึงต้องตั้งค่าให้เป็น แทร็ก 0 ก่อนที่จะเขียนโปรแกรมควบคุมต่อไป

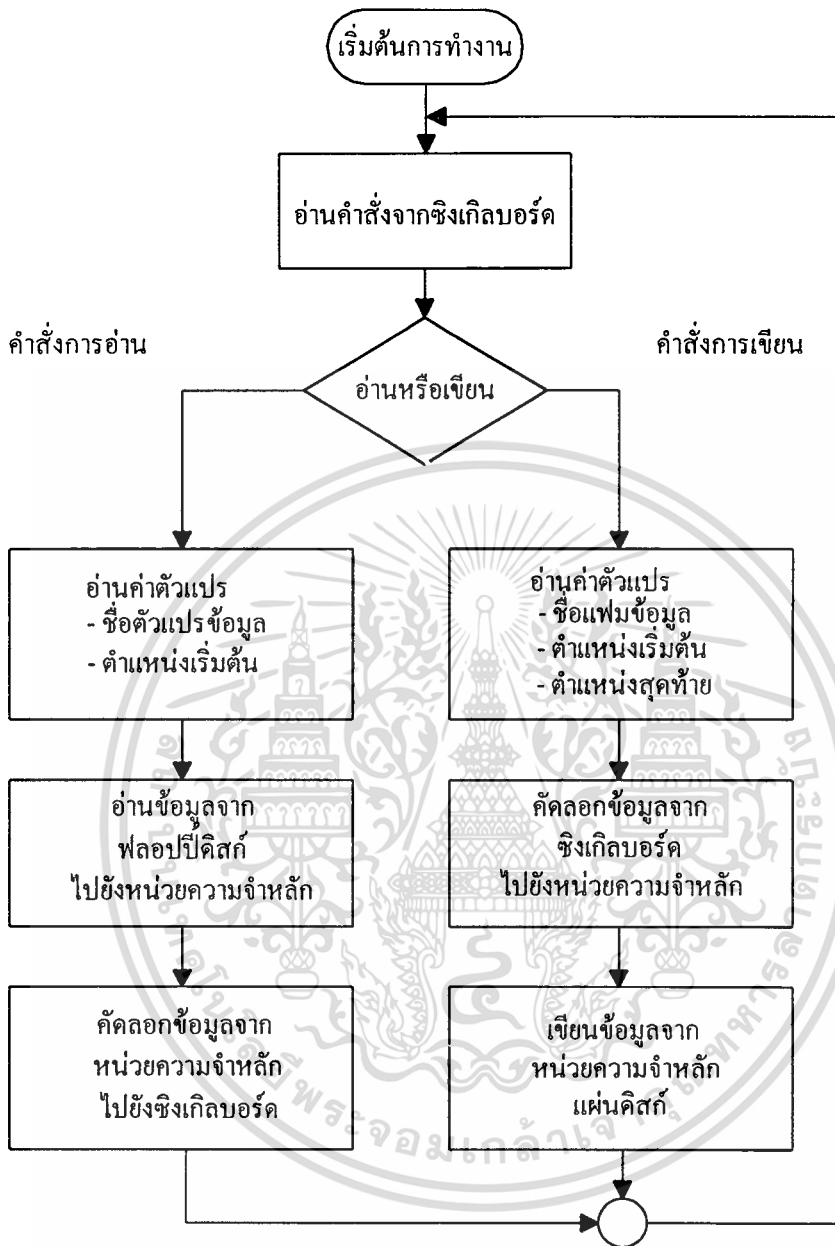
3.10.2 การรับคำสั่งและการเขียน อ่านข้อมูล

การรับคำสั่งการทำงานว่าจะเป็นการบันทึกข้อมูลลงบนแผ่นฟลอปปีดิสก์ หรือ การอ่านข้อมูลออกจากแผ่นฟลอปปีดิสก์นั้น จะเป็นการรอรับคำสั่งการป้อนค่าข้อมูลจากซิงเกิลบอร์ด โดยการทำงานของแต่ละส่วนจะรับข้อมูลดังนี้ คือ ในการอ่านข้อมูลจากแผ่นฟลอปปีดิสก์นั้น จะรับค่าตัวแปรเข้ามาใช้ในการประมวลผล 2 ตัว คือ ชื่อของแฟ้มข้อมูลและตำแหน่งของหน่วยความจำเริ่มต้นในซิงเกิลบอร์ดที่จะทำการคัดลอกข้อมูลเข้าไปเก็บไว้

ในด้านของการเขียนข้อมูลลงแผ่นฟลอปปีดิสก์นั้นจะเริ่มจากการรับชื่อแฟ้มข้อมูล, ตำแหน่งหน่วยความจำเริ่มต้นในซิงเกิลบอร์ด และตำแหน่งหน่วยความจำสุดท้ายของซิงเกิลบอร์ดที่จะทำการคัดลอกข้อมูล จะเห็นว่าการเขียนข้อมูลลงแผ่นฟลอปปีดิสก์ จะใช้ตัวแปรเพิ่มขึ้นอีกหนึ่งตัวแปร คือ ตำแหน่งสุดท้ายของหน่วยความจำ เนื่องจากการคัดลอกข้อมูลจากหน่วยความจำของซิงเกิลบอร์ดนั้นมีจำนวนผู้ใช้จำกัดปริมาณข้อมูลไว้จำนวนหนึ่ง ซึ่งต่างกับการอ่านข้อมูลที่ได้จากแผ่นฟลอปปีดิสก์ เนื่องจากการอ่านข้อมูลจากแผ่นฟลอปปีดิสก์นั้นขนาดของแฟ้มข้อมูลจะเป็นตัวกำหนดขนาดตำแหน่งสุดท้ายของแฟ้มข้อมูลโดยอัตโนมัติ

หลังจากการรับค่าตัวแปรแล้ว จะพิจารณาการทำงานทางด้านของการบันทึกข้อมูลลงในแผ่นฟลอปปีดิสก์ (ด้าน Write Command) โปรแกรมจะเริ่มทำการสั่งให้คัดลอกข้อมูลในตำแหน่งหน่วยความจำเริ่มต้นไปจนถึงหน่วยความจำสุดท้ายที่ถูกระบุไว้มาเก็บในหน่วยความจำหลักของวงจร เนื่องจากการคัดลอกข้อมูลระหว่างหน่วยความจำไปยังหน่วยความจำจะสามารถทำได้รวดเร็วกว่าและลดอัตราการเสี่ยงในเรื่องการผิดพลาดของข้อมูลได้ดีกว่าการที่จะคัดลอกข้อมูลมาเก็บในแผ่นฟลอปปีดิสก์โดยตรง เมื่อการคัดลอกข้อมูลสิ้นสุดลงระบบจะนำเอาข้อมูลที่ได้นั้นไปเก็บไว้ในแผ่นฟลอปปีดิสก์

ทางด้านการอ่านข้อมูลจากแผ่นฟลอปปีดิสก์จะมีขั้นตอนตรงกันข้ามกับการบันทึกข้อมูลลงในแผ่นฟลอปปีดิสก์คือเมื่อรับค่าตัวแปรแล้ว โปรแกรมจะทำการอ่านข้อมูลจากแผ่นฟลอปปีดิสก์ก่อนแล้วจึงนำข้อมูลที่อ่านได้ไปเก็บไว้ในหน่วยความจำหลัก เมื่อแฟ้มข้อมูลถูกอ่านมาเก็บไว้จนหมดแล้ว โปรแกรมจึงเริ่มสั่งให้มีการถ่ายโอนข้อมูลจากหน่วยความจำหลักไปยังหน่วยความจำของซิงเกิลบอร์ดในตำแหน่งเริ่มต้น (Start Address) ที่รับมาจากการกำหนดค่าตัวแปร



รูปที่ 3.16 ฟังก์ชันการทำงานการรับคำสั่งและปฏิบัติงาน

3.11 ภาคอินเตอร์เฟส

ภาค อินเตอร์เฟสแบ่งออกเป็น Z-80 กับ MCS-51

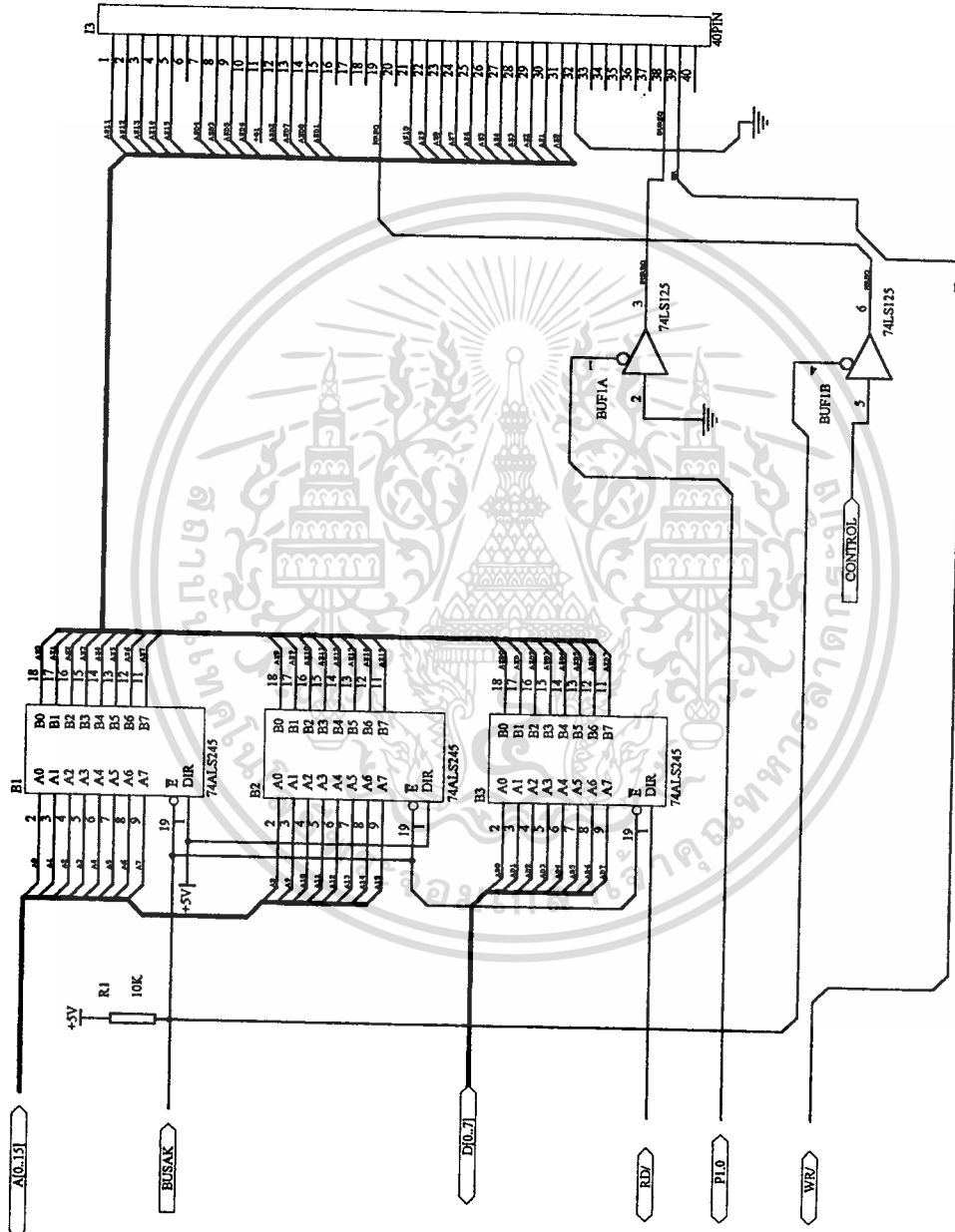
ในการเชื่อมต่อบัสตำแหน่งและบัสข้อมูล และสัญญาณควบคุมต่างๆ ระหว่าง Z-80 และ MCS-51 มีการออกแบบโดยใช้ไอซีบัฟเฟอร์คือเบอร์ 74145 ในการแยกบัสตำแหน่งและสัญญาณการควบคุมต่างๆ ในการเชื่อมต่อระหว่าง Z-80 กับ MCS-51 เพราะว่าไอซีบัฟเฟอร์นี้จะมีสถานะการไปใช้

ทำงาน 3 สถานะ คือ สถานะลอจิก “0” สถานะลอจิก “1” และสถานะ “High Impedance” ซึ่งในสถานะที่ปกติ คือ ไม่มีการเขียนและการอ่านดิสก์บัสตำแหน่ง, บัสข้อมูล และสัญญาณควบคุมระหว่าง Z-80 กับ MCS-51 จะต้องอยู่ในสถานะ High Impedance แต่เมื่อมีการเขียนและการอ่านดิสก์ก็จะมีสถานะเป็นลอจิก “0” และ “1” เท่านั้น

สำหรับสัญญาณควบคุมของ Z-80 นั้น จะใช้สัญญาณ MREQ, WR, RD, BUSRQ และ BUSAK ในการเชื่อมต่อกับขาสัญญาณของ MCS-51 ดังรูปวงจรที่ได้ออกแบบ

สำหรับ Z-80 นั้น ได้ออกแบบเพื่อขยายพอร์ตการทำงานของบอร์ด MPF-1 โดยขยาย พอร์ต 8255 ซึ่งสามารถขยายได้ถึง 256 พอร์ต ดังรูปที่ 3.18





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.17 การเชื่อมต่อขาสัญญาณของ MCS-51

บทที่ 4

การทดลอง และผลการทดลอง

จากที่ได้ทราบการทำงานโดยทั่วไปของวงจรดิจิทัล 3.5 นี้สำหรับซิงเกิลบอร์ดมาแล้ว ในบทที่ 3 ซึ่งทำให้เราทราบถึงหลักการต่างๆ ไป และความสำคัญของการทำงานแต่ละภาค การทดลองจะทำการทดลองไปครั้งละภาค ตามคุณสมบัติทางทฤษฎีในบทที่ 3 ดังนี้ คือ

4.1 Bus Controller

- 1) การโอนข้อมูลจากซิงเกิลบอร์ดไปเก็บในแรมของวงจร
- 2) การโอนข้อมูลจากแรมของวงจรไปเก็บในซิงเกิลบอร์ด

4.2 FDC (Floppy Disk Controller)

- 1) ทดลอง FDC Processor
- 2) ทดลองภาคกำเนิดสัญญาณนาฬิกา
- 3) ทดลองภาค Decoder
- 4) ทดลองภาค Output Register
- 5) ทดลองการทำงานรวมของภาค FDC

4.3 ทดลองการทำงานโดยรวมของวงจรทั้งหมด

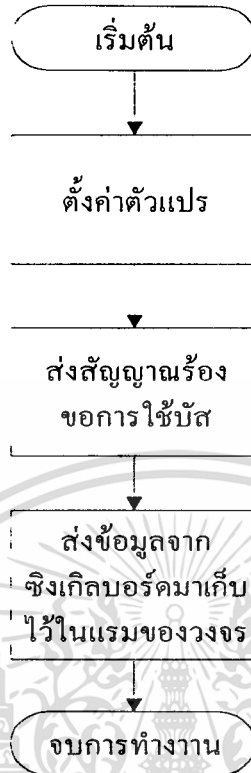
- 1) ทดลองบันทึกข้อมูลที่ได้จากซิงเกิลบอร์ดไปเก็บไว้ในแผ่นฟลอปปีดิสก์
- 2) ทดลองนำข้อมูลที่ได้จากฟลอปปีดิสก์มาเก็บไว้ในแรมของซิงเกิลบอร์ด

4.1 Bus Controller

4.1.1 การทดลองในส่วนการโอนย้ายข้อมูลจากซิงเกิลบอร์ดไปเก็บไว้ในแรมของวงจร

- 1) การกำหนดข้อมูลลงในตำแหน่งหน่วยความจำเฉพาะ

เป็นการกำหนดค่าข้อมูลลงในตำแหน่งหน่วยความจำโดยสามารถกำหนดตำแหน่งได้ ตั้งแต่ตำแหน่ง F000H-FFFFH การทดลองนี้จะกำหนดเฉพาะบางตำแหน่งเท่านั้น



รูปที่ 4.1 แผนผังการทำงานของ การโอนย้ายข้อมูลจากซิงเกิลบอร์ดไปเก็บในแรมของวงจร

2) โปรแกรมทดสอบการกำหนดข้อมูลลงในตำแหน่งหน่วยความจำเฉพาะ

WRITE:	MOV	R5, #00H
	CLR	P1.0
	LCALL	DELAY
	MOV	DPTR, #0F800H
LOOPW:	MOV	A, R5
	MOVX	@DPTR, A
	INC	DPL
	INC	R5
	CJNE	R5, #0FFH, LOOPW
	RET	
DELAY:	MOV	R1, #50H
DELAY1:	MOV	R2, #0FFH
	DJNZ	R2, \$
	DJNZ	R1, DELAY1
	RET	

เอกสารนี้เป็นเอกสารที่ 4.2 โปรแกรมการกำหนดข้อมูลลงในตำแหน่งหน่วยความจำเฉพาะ ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

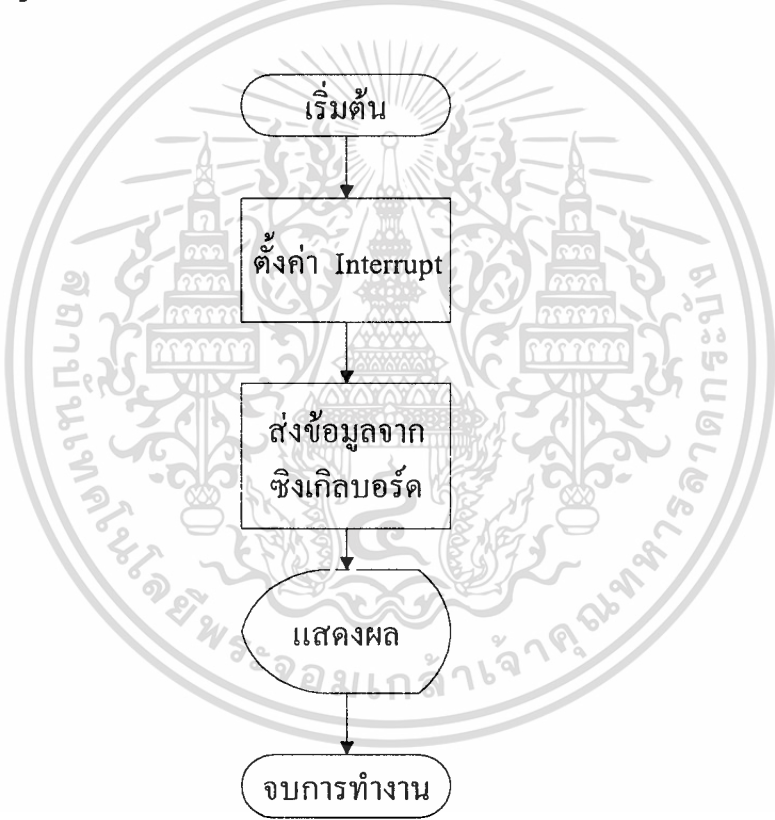
3) ผลการทดลอง

ข้อมูลในตำแหน่ง F800H-F8FFH มีข้อมูล 00H - FFH

<M> F800H : 00 01 02 03

<M> F8FFH : FC FD FE FF

4.1.2 การอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ด



รูปที่ 4.3 แผนผังการทำงานของ การอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ด

1) โปรแกรมทดลอง

READ:	CLR	P1.0
	MOV	R6, #00H
	LCALL	DELAY
	MOV	DPTR, #0F800H
	PUSH	DPL

```

LOOPR:  PUSH    DPH
        POP     DPH
        POP     DPL
        MOVX    A,@DPTR
        MOV     R6,A
        INC     DPL
        PUSH    DPL
        PUSH    DPH
        LCALL   DELAYTIME
        LCALL   DELAYTIME
        MOV     DPTR,#PORTA
        MOV     A,R6
        MOVX    @DPTR,A
        CJNE    R6,#0FH,LOOPR
        SETB    P1.0
        RET

DELAY:   MOV     R1,#50H
DELAY1:  MOV     R2,#0FFH
        DJNZ    R2,$
        DJNZ    R1,DELAY1
        RET

DELAYTIME: MOV    R1,#0FFH
DELAYTIME1: MOV    R2,#0FFH
        DJNZ    R2,$
        DJNZ    R1,DELAYTIME1
        RET

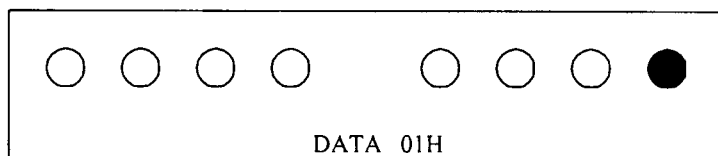
```

รูปที่ 4.4 โปรแกรมการอ่านข้อมูลตำแหน่งหน่วยความจำจากซิงเกิลบอร์ด

2) ผลการทดลอง

ผลการทดลองที่ได้คือ ข้อมูลที่อยู่ในตำแหน่ง F800H-F80FH จะถูกแสดงผลออกมาที่พอร์ต A เป็นข้อมูล

00H 01H 02H 03H 04H 05H 06H 07H 08H 09H 0AH 0BH 0CH 0DH 0EH 0FH



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานที่เฉพาะเจาะจงเท่านั้น ไม่อนุญาตให้拿去ไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.3 การทดลองกำหนดข้อมูลลงในตำแหน่งหน่วยความจำ MCS-51 แล้วโอนย้ายไปยังซิงเกิลบอร์ด

เป็นการกำหนดค่าข้อมูลลงในตำแหน่งหน่วยความจำ โดยสามารถกำหนดตำแหน่งได้ตั้งแต่ตำแหน่ง F000H-FFFFH โดยการทดลองจะกำหนดข้อมูลในแรมของ MCS-51 แล้วค่อยโอนย้ายไปยังตำแหน่งหน่วยความจำของซิงเกิลบอร์ดตามต้องการ



รูปที่ 4.5 แผนผังการทำงานของ การกำหนดข้อมูลในหน่วยความจำแล้วโอนย้ายไปยังซิงเกิลบอร์ด

1) โปรแกรมการทดลอง

MODE1	EQU	80H
CONTROL	EQU	0C003H
PORTA	EQU	0C000H
PORTB	EQU	0C001H
PORTC	EQU	0C002H

```

ADD51HR    EQU        24H
ADD51LR    EQU        25H
ADDZ80HW   EQU        26H
ADDZ80LW   EQU        27H
BYTEWRITE  EQU        32H
BYTEREAD   EQU        33H

        ORG          0000H
        MOV          DPTR, #CONTROL
        MOV          A, #MODE1
        MOVX         @DPTR, A

;-----INITIAL PORT-----
        MOV          A, #00H
        MOV          DPTR, #PORTA
        MOVX         @DPTR, A
        MOV          DPTR, #PORTB
        MOVX         @DPTR, A
        MOV          DPTR, #PORTC
        MOVX         @DPTR, A
        MOV          P1, #0FFH

;-----INITIAL DATA ADDRESS-----
        MOV          ADD51HR, #80H
        MOV          ADD51LR, #00H
        MOV          ADDZ80HW, #0F8H
        MOV          ADDZ80LW, #00H
        MOV          BYTEREAD, #10H
        MOV          BYTEWRITE, #0FFH

;-----MAIN-----;
MAINTOZ80: LCALL     WRITETOZ80
          LCALL     DELAYTIME
          LCALL     TRANSFERZ80
          LCALL     DELAYTIME
          LJMP      END1
WRITETOZ80: SETB     P1.0
          LCALL     DELAY
          MOV        R3, BYTEWRITE
          MOV        DPTR, #ADDREAD51
          MOV        R5, #02H
LOOPW1:   MOV        A, R5
          MOVX       @DPTR, A
          INC        DPL
          INC        R5
          INC        R5
          DJNZ       R3, LOOPW1
          RET
TRANSFERZ80:
          SETB       P1.0
          MOV        R4, BYTEREAD
          DEC        R4
          MOV        DPH, ADDZ80HW
          MOV        DPL, ADDZ80LW
          PUSH        DPL

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งยังมีให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      DPH,ADD51HR
MOV      DPL,ADD51LR
MOVX     A,@DPTR
MOV      10H,A
CLR      P1.0
LCALL    DELAY
POP      DPL
POP      DPH
MOV      A,10H
MOVX     @DPTR,A
SETB     P1.0
PUSH     DPH
PUSH     DPL
LOOPRRZ80: POP    DPL
           POP    DPH
           INC    DPL
           PUSH   DPL
           MOV    DPH,ADD51HR      ; ADDRESS MCS-51 SEND
           MOVX   A,@DPTR          ; TRANSFER DATA SEND
           CLR    P1.0
           LCALL  DELAY
           MOV    DPH,ADDZ80HW      ; ADDRESS Z-80 RECEIVER
           MOVX   @DPTR,A          ; TRANSFER DATA RECEIVER
           SETB   P1.0
           MOV    R5,A
           POP    DPL
           PUSH   DPH
           PUSH   DPL
           MOV    A,R5
           MOV    DPTR,#PORTC
           MOVX   @DPTR,A
           LCALL  DELAY
           DJNZ   R4,LOOPRRZ80
           LJMP   END1
DELAY:    MOV    R1,#025H
DELAY1:   MOV    R2,#0FFH
           DJNZ   R2,$
           DJNZ   R1,DELAY1
           RET
DELAYTIME: MOV   R1,#25H
DELAYTIME1:MOV   R2,#0FFH
           DJNZ   R2,$
           DJNZ   R1,DELAYTIME1
           RET
END1:     SETB   P1.0
           CLR    P1.7
           SJMP   END1
           END

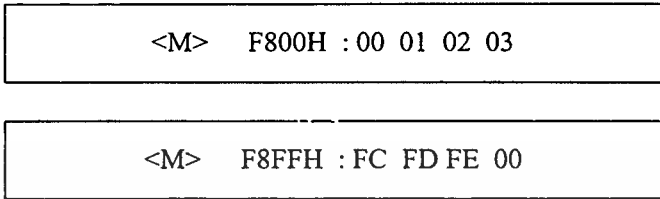
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
รูปที่ 4.6 โปรแกรมการกำหนดข้อมูลในหน่วยความจำแล้วโอนย้ายไปยังซีงเกิลบอร์ด
 ไม่ว่ากรณีใดๆ ทั้งสิ้น ผู้จัดทำไม่รับผิดชอบต่อเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2) ผลการทดลอง

ข้อมูลที่อยู่ในตำแหน่ง 8000H สามารถโอนย้ายไปยังตำแหน่ง F800H ได้ 255 ไบต์

ข้อมูลในตำแหน่ง F800H-F8FFH มีข้อมูล 00H - FFH



4.1.4 การทดลองกำหนดข้อมูลในตำแหน่งหน่วยความจำของซิงเกิลบอร์ด แล้วโอนย้ายไปยัง MCS-51

เป็นการกำหนดค่าของข้อมูลลงในตำแหน่งหน่วยความจำของ MCS-51 โดยสามารถกำหนดตำแหน่งได้ตั้งแต่ตำแหน่ง 6000H-9FFFH การทดลองจะกำหนดข้อมูลในแรมของซิงเกิลบอร์ด แล้วจึงโอนย้ายไปยังตำแหน่งหน่วยความจำของ MCS-51 ตามต้องการ โดยดูผลการทดลองจาก Port Display



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่สามารถใด ๆ ทั้งสิ้น ถือว่าหน่วยงานนี้ให้ข้อมูลเฉพาะเท่านั้น ไม่สามารถนำข้อมูลไปใช้

รูปที่ 4.7 แผนผังการทำงานของ การกำหนดข้อมูลลงในหน่วยความจำแล้ว โอนย้ายไปยัง MCS-51

1) โปรแกรมการทดลอง

```

MODE1      EQU      80H
CONTROL     EQU      0C003H

PORTA       EQU      0C000H
PORTB       EQU      0C001H
PORTC       EQU      0C002H

```

```

ADDZ80HIGH EQU      20H
ADD51HIGH  EQU      21H
ADDZ80LOW  EQU      22H
ADD51LOW   EQU      23H

```

```

BYTEWRITE  EQU      32H
BYTEREAD   EQU      33H

```

```

ORG        0000H
MOV        DPTR, #CONTROL
MOV        A, #MODE1
MOVX       @DPTR, A

```

```

;-----INITIAL PORT-----;

```

```

MOV        A, #00H
MOV        DPTR, #PORTA
MOVX       @DPTR, A
MOV        DPTR, #PORTB
MOVX       @DPTR, A
MOV        DPTR, #PORTC
MOVX       @DPTR, A
MOV        P1, #0FFH

```

```

;-----INITIAL DATA ADDRESS-----;

```

```

MOV        ADD51HIGH, #60H
MOV        ADD51LOW, #00H
MOV        ADDZ80HIGH, #0F0H
MOV        ADDZ80LOW, #00H
MOV        BYTEREAD, #10H
MOV        BYTEWRITE, #0FFH

```

```

;-----MAIN-----;

```

```

MAINTO51:  LCALL      WRITETO51
           LCALL      DELAYTIME
           LCALL      TRANFER51

```

```

;-----WRITE-----;

```

```

WRITETO51: CLR        P1.0

```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        LCALL    DELAY
        MOV      DPTR,#ADDREADZ80
        PUSH     DPH
        MOV      R3,BYTEWRITE
        MOV      R5,#00H
LOOPW:  MOV      A,R5
        MOVX     @DPTR,A
        INC      DPL
        INC      R5
        DJNZ     R3,LOOPW
        RET

```

; -----TRANSFER DATA RAM TO MCS-51-----

```

TRANSFER51:CLR      P1.0
        LCALL    DELAY
        MOV      R4,BYTEREAD
        DEC      R4
        MOV      DPH,ADD51HIGH
        MOV      DPL,ADD51LOW
        PUSH     DPH
        PUSH     DPL
        MOV      DPH,ADDZ80HIGH
        MOV      DPL,ADDZ80LOW
        MOVX     A,@DPTR
        POP      DPL
        POP      DPH
        MOVX     @DPTR,A
        PUSH     DPH
        PUSH     DPL

LOOPRR51: POP      DPL
        POP      DPH
        INC      DPL
        PUSH     DPL
        MOV      DPH,ADDZ80HIGH ; ADDRESS Z-80 SEND
        MOVX     A,@DPTR        ; TRANSFER DATA SEND
        SETB     P1.0
        MOV      DPH,ADD51HIGH
        MOVX     @DPTR,A        ; TRANSFER DATA RECEIVER
        MOVX     A,@DPTR
        MOV      R5,A
        POP      DPL
        PUSH     DPH
        PUSH     DPL
        MOV      A,R5
        MOV      DPTR,#PORTC
        MOVX     @DPTR,A
        CLR      P1.0
        LCALL    DELAY
        DJNZ     R4,LOOPRR51
        LJMP     END1

```

```

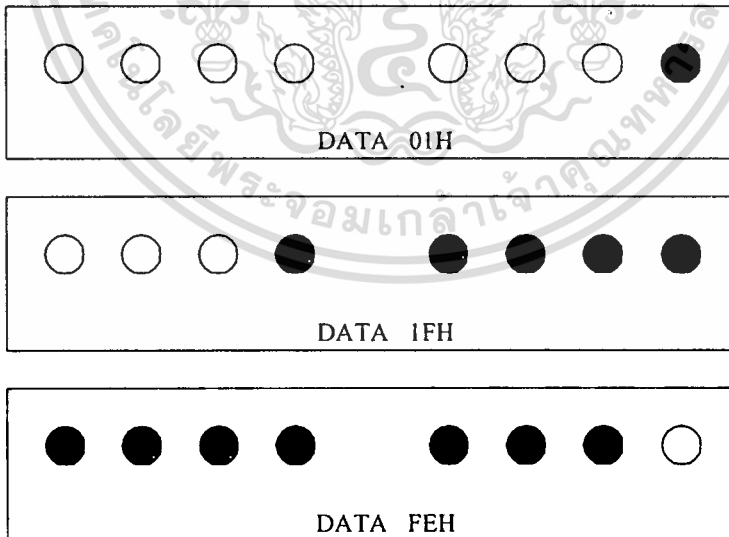
;-----
DELAY:      MOV        R1,#025H
DELAY1:     MOV        R2,#0FFH
            DJNZ       R2,$
            DJNZ       R1,DELAY1
            RET
            DELAYTIME: MOV        R1,#25H
DELAYTIME1: MOV        R2,#0FFH
            DJNZ       R2,$
            DJNZ       R1,DELAYTIME1
            RET
END1:      SETB        P1.0
            CLR        P1.7
            SJMP       END1
            END

```

รูปที่ 4.8 โปรแกรมกำหนดข้อมูลในหน่วยความจำของซิงเกิลบอร์ดแล้วโอนย้ายไปยัง MCS-51

2) ผลการทดลอง

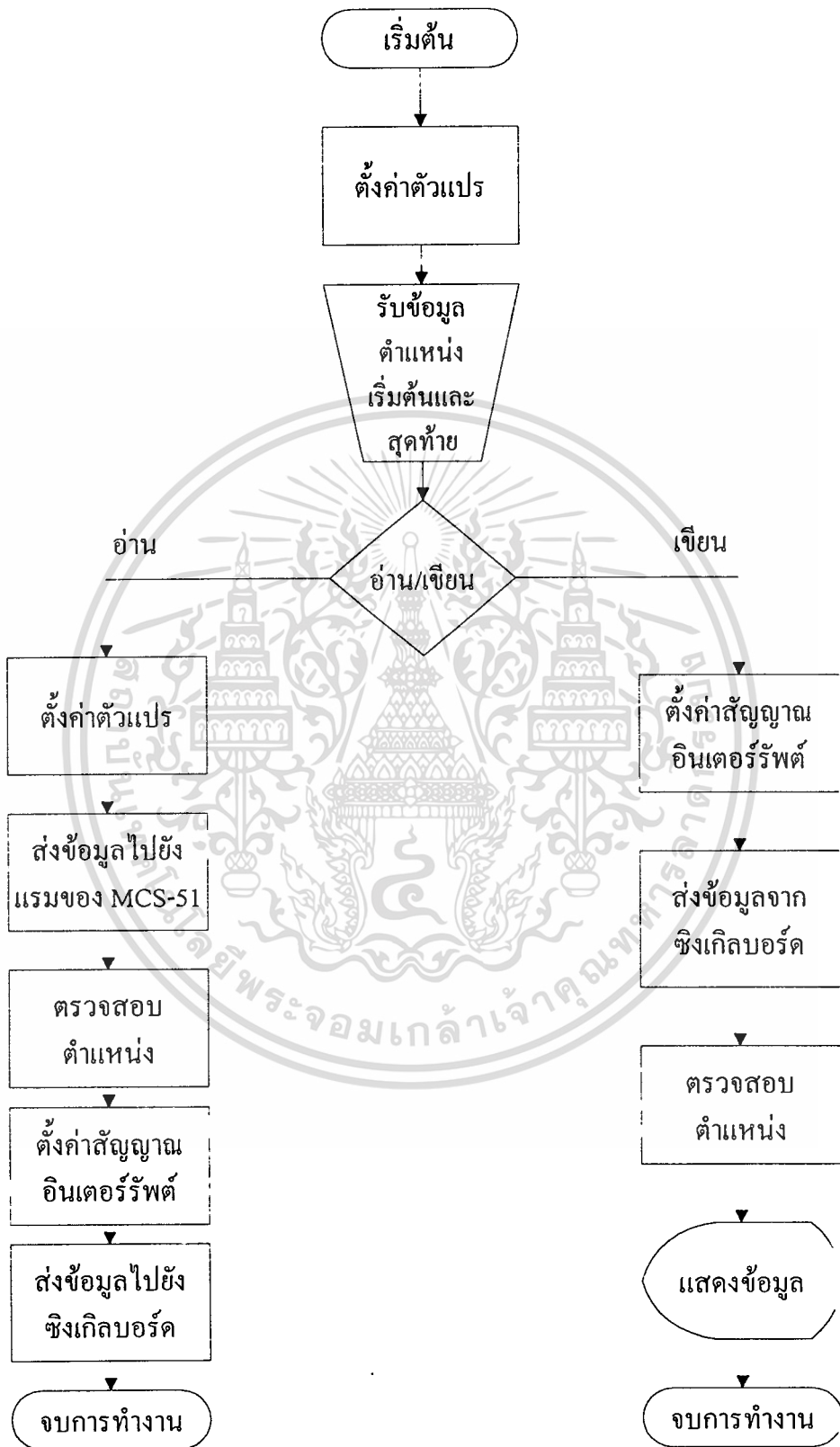
สามารถที่จะโอนย้ายข้อมูลจากตำแหน่งหน่วยความจำ F000H จาก Z-80 ไปยังตำแหน่ง 6000H ของ MCS-51 ได้ 255 ไบต์ โดยดูจาก Port Display



4.1.5 การทดลองโอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51

ตำแหน่งหน่วยความจำในการโอนย้ายจะถูกกำหนดโดยผู้ใช้ โดยป้อนตำแหน่งเริ่มต้นและตำแหน่งสุดท้าย ที่ตำแหน่งหน่วยความจำ F000H, F001H, F002H และ F003H

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดก็ตาม รูปที่ 4.9 แผนผังการทำงานของการ์โอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51 ที่มีการนำไปใช้

1) โปรแกรมที่ใช้ในการทดลอง

```

MODE1      EQU      80H
CONTROL    EQU      0C003H
PORTA      EQU      0C000H
PORTB      EQU      0C001H
PORTC      EQU      0C002H
ADDZ80HIGH EQU      20H
ADD51HIGH  EQU      21H
ADDZ80LOW  EQU      22H
ADD51LOW   EQU      23H
ADD51HR    EQU      24H
ADD51LR    EQU      25H
ADDZ80HW   EQU      26H
ADDZ80LW   EQU      27H
BYTEWRITE  EQU      32H
BYTEREAD   EQU      33H

ORG        0000H
MOV        DPTR, #CONTROL
MOV        A, #MODE1
MOVBX      @DPTR, A

;-----INITIAL PORT-----
MOV        A, #00H
MOV        DPTR, #PORTA
MOVBX      @DPTR, A
MOV        DPTR, #PORTB
MOVBX      @DPTR, A
MOV        DPTR, #PORTC
MOVBX      @DPTR, A
MOV        P1, #0FFH

;-----INITIAL DATA ADDRESS-----
MOV        ADD51HIGH, #60H
MOV        ADD51LOW, #00H
MOV        ADDZ80HIGH, #0F0H
MOV        ADDZ80LOW, #00H
MOV        ADD51HR, #80H
MOV        ADD51LR, #00H
MOV        ADDZ80HW, #0F8H
MOV        ADDZ80LW, #00H
MOV        BYTEREAD, #14H
MOV        BYTEWRITE, #0FFH
MOV        BYTEREADFILE, #0FH

;-----DETEC SIGNAL-----
SCANDATA:  MOV        P1, #0FFH
MOV        R0, P1
SCANREAD:  CJNE       R0, #11111011B, SCANWRITE
LJMP      MAINTO51

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นไว้สำหรับใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ทั้งนี้ทั้งนั้นหากมีข้อผิดพลาดประการใดขออภัยเป็นอย่างสูง และขอสงวนสิทธิ์ในเอกสารทุกครั้งที่มีการนำไปใช้

```

SCANWRITE:  LJMP      SCANDATA
             CJNE      R0,#11111101B,SCANDATA
             LJMP      MAINTOZ80
             LJMP      SCANDATA

;-----MAIN-----;
MAINTO51:   LCALL     WRITETOZ80
             LCALL     DELAYTIME
             LCALL     TRANSFERTO51      ; TO SET AT F000H
             SJMP      MAINTO51
             RET

MAINTOZ80:  LCALL     WRITETO51
             LCALL     DELAYTIME
             LCALL     TRANSFER51        ; TO SET AT F000H
             SJMP      MAINTOZ80
             RET

;-----WRITE-----;
WRITETO51:  CLR       P1.0
             LCALL     DELAY
             MOV       DPTR,#0F700H
             LCALL     WRITE
             MOV       DPTR,#0F800H
             LCALL     WRITE
             RET

WRITE:      MOV       R3,BYTEWRITE
             MOV       R4,#02H
             MOV       R4,#01H

LOOPW:      MOV       A,14H
             MOVX      @DPTR,A
             INC       DPL
             INC       R4
             DEC       R3
             CJNE      R3,#00H,LOOPW
             INC       DPL
             DJNZ      R4,LOOPW
             RET

WRITETOZ80: SETB      P1.0
             LCALL     DELAY
             MOV       DPTR,#ADDREAD51

WRITE1:     MOV       R3,BYTEWRITE
             MOV       R5,#00H
             MOV       R4,#02H

LOOPW1:     MOV       A,R5
             MOVX      @DPTR,A
             INC       DPL
             INC       R5
             DEC       R4

```

```

CJNE      R3,#00H,LOOPW1
INC       DPL
DJNZ      R4,LOOPW1
MOV       DPL,#00H
INC       DPH
MOV       R7,DPH
CJNE      R7,#82H,WRITE1
LCALL     DELAY
RET

```

;-----TRANSFER DATA RAM TO MCS-51-----

```

TRANSFER51:CLR      P1.0
LCALL     DELAY
MOV       R4,BYTEREAD      ; BYTE FILE 14H
DEC       R4
MOV       DPH,ADD51HIGH    ; 60H RECEIVER
MOV       DPL,ADD51LOW     ; 00H RECEIVER
PUSH      DPH
PUSH      DPL
MOV       DPH,ADDZ80HIGH   ; 0F0H TRANSFER
MOV       DPL,ADDZ80LOW    ; 00H TRANSFER
MOVX      A,@DPTR
MOV       10H,A
SETB      P1.0
LCALL     DELAY
POP       DPL
POP       DPH
MOV       A,10H
MOVX      @DPTR,A
CLR       P1.0
LCALL     DELAY
PUSH      DPH
PUSH      DPL
LOOPRR51: POP       DPL
POP       DPH
INC       DPL
PUSH      DPL
MOV       DPH,ADDZ80HIGH   ; ADDRESS Z-80 SEND
MOVX      A,@DPTR
SETB      P1.0
MOV       DPH,ADD51HIGH
MOVX      @DPTR,A
MOVX      A,@DPTR
MOV       R5,A
POP       DPL
PUSH      DPH
PUSH      DPL
CLR       P1.0
LCALL     DELAY
DJNZ      R4,LOOPRR51
PUSH      DPH
PUSH      DPL
LJMP     WRITEFILE

```

;-----TRANSFER DATA RAM TO MCS-51-----

TRANSFERTO51:

```

CLR          P1.0
LCALL        DELAY
MOV          ADD51HR,#80H      ; 80H
MOV          ADD51LR,#00H      ; 00H
MOV          R4,BYTE READ     ; BYTE FILE 14H
DEC          R4
MOV          DPH,ADD51HIGH     ; 60H RECEIVER
MOV          DPL,ADD51LOW      ; 00H RECEIVER
PUSH         DPH
PUSH         DPL
MOV          DPH,ADDZ80HIGH    ; 0F0H TRANSFER
MOV          DPL,ADDZ80LOW     ; 00H TRANSFER
MOVX         A,@DPTR
MOV          10H,A
SETB         P1.0
LCALL        DELAY
POP          DPL
POP          DPH
MOV          A,10H
MOVX         @DPTR,A
CLR          P1.0
LCALL        DELAY
PUSH         DPH
PUSH         DPL
LOOPPRTO51: POP          DPL
POP          DPH
INC          DPL
PUSH         DPL
MOV          DPH,ADDZ80HIGH    ; ADDRESS Z-80 SEND
MOVX         A,@DPTR          ; TRANSFER DATA SEND
SETB         P1.0
MOV          DPH,ADD51HIGH     ; TRANSFER DATA RECEIVER
MOVX         @DPTR,A
MOVX         A,@DPTR
MOV          R5,A
POP          DPL
PUSH         DPH
PUSH         DPL
CLR          P1.0
LCALL        DELAY
DJNZ         R4,LOOPPRTO51
LJMP         READFILE

```

;-----TRANSFER FOR FILE-----;

```

READFILE:  SETB         P1.0
           LCALL        DELAY
           MOV          DPTR,#6000H
           MOVX         A,@DPTR
           MOV          ADDZ80HW,A
           MOV          10H,A
           MOV          DPTR,#6001H
           MOVX         A,@DPTR

```

เอกสารนี้เป็นเอกสารที่รวบรวมไว้สำหรับใช้ศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้นผู้จัดทำห้ามมิให้ออกไปเผยแพร่และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      ADDZ80LW,A
MOV      11H,A
;-----FINE BYTE FROM END ADDRESS-----
MOV      DPTR,#6002H
MOVX     A,@DPTR
SUBB     A,10H
MOV      15H,A
SUB:     MOV      A,15H
        CJNE     A,#00H,LOAD
        MOV      DPTR,#6003H
        MOVX     A,@DPTR
        SUBB     A,11H
        MOV      BYTEREAD,A
        MOV      R4,BYTEREAD
        MOV      DPH,ADDZ80HW      ; FROM 6000H
        MOV      DPL,ADDZ80LW     ; FROM 6001H
        PUSH     DPH
        PUSH     DPL
        MOV      DPH,ADD51HR      ; 80H
        MOV      DPL,ADD51LR      ; 00H
        MOVX     A,@DPTR
        MOV      R7,A
        CLR      P1.0
        LCALL    DELAY
        POP      DPL
        POP      DPH
        MOV      A,R7
        MOVX     @DPTR,A
        SETB     P1.0
        PUSH     DPH
        PUSH     DPL
LOOPRDFILE: POP      DPL
        POP      DPH
        INC      DPL
        PUSH     DPL
        MOV      DPH,ADD51HR
        MOVX     A,@DPTR
        CLR      P1.0
        LCALL    DELAY
        MOV      DPH,ADDZ80HW
        MOVX     @DPTR,A
        SETB     P1.0
        MOV      R5,A
        POP      DPL
        PUSH     DPH
        PUSH     DPL
        LCALL    DISPLAY
        LCALL    DELAY
        DJNZ     R4,LOOPRDFILE
LOAD:    LJMP     SCANDATA
        MOV      R4,#0FEH
        MOV      DPH,ADDZ80HW      ; FROM 6000H
        MOV      DPL,ADDZ80LW     ; FROM 6001H

```

```

PUSH      DPH
PUSH      DPL
MOV       DPH,ADD51HR      ; 80H
MOV       DPL,ADD51LR      ; 00H
MOVX      A,@DPTR
MOV       10H,A
CLR       P1.0
LCALL     DELAY
POP       DPL
POP       DPH
MOV       A,10H
MOVX      @DPTR,A
SETB      P1.0
PUSH      DPH
PUSH      DPL
LCHKFILE: POP      DPL
POP       DPH
INC       DPL
PUSH      DPL
MOV       DPH,ADD51HR
MOVX      A,@DPTR
CLR       P1.0
LCALL     DELAY
MOV       DPH,ADDZ80HW
MOVX      @DPTR,A
SETB      P1.0
MOV       R5,A
POP       DPL
PUSH      DPH
PUSH      DPL
LCALL     DISPLAY
LCALL     DELAY
DJNZ      R4,LCHKFILE
MOV       DPH,ADD51HR
INC       DPH
MOV       ADD51HR,DPH
MOV       ADD51LR,#00H
MOV       DPH,ADDZ80HW
INC       DPH
MOV       ADDZ80HW,DPH
MOV       ADDZ80LW,#00H
DEC       15H
LJMP      SUB
;-----WRITE FILE-----;
WRITEFILE: POP      DP
POP       DPH
SETB      P1.0
MOV       DPTR,#6000H
MOVX      A,@DPTR
MOV       ADDZ80HW,A
MOV       12H,A
MOV       DPTR,#6001H
MOVX      A,@DPTR

```

เอกสารนี้เป็นเอกสารที่มอบไว้สำหรับ ADDZ80HW, A ศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ออกไปเผยแพร่ภายนอกและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      ADDZ80LW,A
MOV      13H,A
;-----FINE BYTE FROM END ADDRESS-----
MOV      DPTR,#6002H
MOVX     A,@DPTR
SUBB     A,12H
MOV      16H,A
LOOP1:   MOV      A,16H
         CJNE     A,#00H,LOOP2
         LJMP     LOAD1
LOOP2:   MOV      A,#0FFH
         SJMP     LOAD3
LOAD1:   MOV      DPTR,#6003H
MOVX     A,@DPTR
SUBB     A,13H
LOAD3:   MOV      BYTEREAD,A
MOV      R4,BYTEREAD
MOV      DPH,ADD51HR      ; 80H
MOV      DPL,ADD51LR      ; 00H
PUSH     DPH
PUSH     DPL
CLR      P1.0
LCALL    DELAY
MOV      DPH,ADDZ80HW      ; FROM 6000H
MOV      DPL,ADDZ80LW      ; FROM 6001H
MOVX     A,@DPTR
MOV      R7,A
SETB     P1.0
POP      DPL
POP      DPH
MOV      A,R7
MOVX     @DPTR,A
CLR      P1.0      ; IMPORTANT
LCALL    DELAY
PUSH     DPH
PUSH     DPL
LOOPWDFILE: POP    DPL
           POP     DPH
           INC     DPL
           PUSH    DPL
           MOV     DPH,ADDZ80HW      ; ADDRESS Z-80 SEND
           MOVX    A,@DPTR
           SETB    P1.0
           MOV     DPH,ADD51HR
           MOVX    @DPTR,A
           MOVX    A,@DPTR
           MOV     R5,A
           POP     DPL
           PUSH    DPH
           PUSH    DPL
           MOV     R5,A
           LCALL   DISPLAY

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้นหากทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        LCALL    DELAY
        CLR      P1.0
        LCALL    DELAY
        DJNZ     R4, LOOPWDFILE
        MOV      A, 16H
        CJNE     A, #00H, LOOPP
        LJMP     SCANDATA

LOOPP:
MOV      DPH, ADD51HR
        NC       DPH
        MOV      ADD51HR, DPH
        MOV      ADD51LR, #00H
        MOV      DPH, ADDZ80HW
        INC      DPH
        MOV      ADDZ80HW, DPH
        MOV      ADDZ80LW, #00H
        DEC      16H
        LJMP     LOOP1
;-----

DISPLAY: MOV      A, R5
        MOV      DPTR, #PORTC
        MOVX     @DPTR, A
        RET
;-----

DELAY:   MOV      R1, #28H
        MOV      R1, #35H           ; FOR WRITE

DELAY1:  MOV      R2, #0FFH
        DJNZ     R2, $
        DJNZ     R1, DELAY1
        RET

DELAYTIME: MOV     R1, #25H
DELAYTIME1: MOV     R2, #0FFH
        DJNZ     R2, $
        DJNZ     R1, DELAYTIME1
        RET

END1:   SETB     P1.0
        CLR      P1.7
        SJMP     END1
        END

```

รูปที่ 4.10 โปรแกรมการโอนย้ายข้อมูลระหว่าง Z-80 และ MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1) ส่วนของการอ่านข้อมูล สามารถอ่านข้อมูลได้ตามตำแหน่งที่กำหนด คือ ตำแหน่ง F000H การเปลี่ยนตำแหน่งการป้อนที่จะกำหนดตำแหน่งเริ่มต้นและตำแหน่งสุดท้าย กำหนดจาก โปรแกรมจากค่า ADDZ80HIGH และ ADDZ80LOW ซึ่งจะมีการกำหนดชื่อไฟล์ด้วยในตำแหน่ง หน่วยความจำ ซึ่งก็เป็นข้อมูลตัวหนึ่งนั่นเอง

<M> F000H : F1 00 F1 FF

; กำหนดตำแหน่งเริ่มต้น และสุดท้าย

<M> F100H : 00 01 02 03

<M> F1FFH : FC FD FE FF

<M> F000H : F8 00 F8 FF

; กำหนดตำแหน่งเริ่มต้น และสุดท้าย

<M> F800H : 00 01 02 03

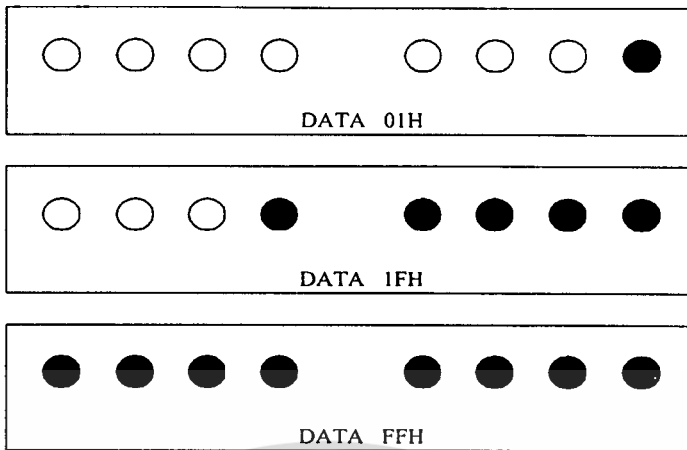
<M> F8FFH : FC FD FE FF

2.2) ส่วนของการเขียนข้อมูล สามารถนำข้อมูลในตำแหน่งที่กำหนดจากตำแหน่ง F000H เขียนให้แรมของ MCS-51 ได้ การเปลี่ยนตำแหน่งการป้อนที่จะกำหนดตำแหน่งเริ่มต้น และตำแหน่งสุดท้ายเป็นไปในลักษณะเดียวกันกับการอ่านข้อมูล

<M> F000H : F1 00 F1 FF

; กำหนดตำแหน่งเริ่มต้น และสุดท้าย

<M> F100H : 00 01 02 03



4.1.6 การป้อนค่าต่างๆ ในการโอนย้ายข้อมูล

การป้อนค่าต่างๆ ที่ใช้ในการโอนย้ายข้อมูลจะถูกกำหนดโดยคำสั่งและตัวชี้ในโปรแกรม Z-80 ซึ่งประกอบด้วย

1. ค่าตำแหน่งหน่วยความจำเริ่มต้น
2. ค่าตำแหน่งหน่วยความจำสุดท้าย
3. ชื่อไฟล์

โดยจะมีการใช้ตัวชี้เพื่อชี้ไปในตำแหน่งหน่วยความจำของ Z-80 ซึ่งจะเก็บชื่อไฟล์ ค่าตำแหน่งหน่วยความจำเริ่มต้นและค่าตำแหน่งสุดท้ายไว้ เมื่อ MSC-51 ได้รับคำสั่งจาก Z-80 จะปฏิบัติการโอนย้ายข้อมูลได้ ซึ่งนับสำคัญอยู่ที่ Z-80 ในการสั่งการทำงานให้มีการอ่านหรือเขียนดิสก์ โดยออกแบบให้สามารถขยายพอร์ตเพื่อให้ใช้กับ Z-80 รุ่นอื่นๆ ได้ ดังวงจรที่ได้ออกแบบในบทที่ 3

1) การทดลองป้อนตำแหน่งเริ่มต้น และตำแหน่งสุดท้ายในการโอนย้ายข้อมูล

ข้อมูลจะถูกอ่านและเขียนในตำแหน่งของหน่วยความจำที่กำหนดจาก Z-80 โดยมีโปรแกรมที่ต้องป้อนที่ Z-80 คือ

LD	A, 80H	
OUT	43H), A	
LD	A, 01H	; COMMAND ADDRESS START
OUT	40H), A	
LD	A, 00H	; INDEX ADDRESS BYTE LOW
OUT	41H), A	
LD	A, F1H	
OUT	42H), A	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้เผยแพร่โดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.11 โปรแกรมที่ป้อนให้ Z-80

การรันโปรแกรมในการป้อนค่าจะถูกกำหนดจาก COMMAND โดยมีรูปแบบคำสั่งดัง
ตารางที่ 4.1

ตารางที่ 4.1 รูปแบบคำสั่ง Command

Command	ความหมาย
01H	WRITE
02H	READ

1.1)โปรแกรมการทดลอง

MODE1	EQU	80H
CONTROL	EQU	0C003H
PORTA	EQU	0C000H
PORTB	EQU	0C001H
PORTC	EQU	0C002H
ADDWRITE51	EQU	6000H
ADDREAD51	EQU	7000H
ADDZ80HIGH	EQU	20H
ADD51HIGH	EQU	21H
ADDZ80LOW	EQU	22H
ADD51LOW	EQU	23H
ADD51HR	EQU	24H
ADD51LR	EQU	25H
ADDZ80HW	EQU	26H
ADDZ80LW	EQU	27H
BYTEL1	EQU	28H
BYTEL2	EQU	29H
BYTEH	EQU	30H
BYTEWRITE	EQU	32H
BYTEREAD	EQU	33H00
BYTEREADHIGH		
	EQU	34H
BYTEWRITEHIGH		
	EQU	35H
BYTEREADFILE		
	EQU	36H
PORTSET	EQU	P1.7
	ORG	0000H
	SETB	PORTSET
	MOV	DPTR, #CONTROL
	MOV	A, #MODE1
	MOVX	@DPTR, A

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับใช้ภายในมหาวิทยาลัยเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ที่ฝ่าฝืนทั้งห้ามีโทษจำคุกไม่เกินห้าปีหรือปรับไม่เกินห้าหมื่นบาทและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
;-----INITIAL PORT-----
```

```
MOV      A,#00H
MOV      DPTR,#PORTA
MOVX     @DPTR,A
MOV      DPTR,#PORTB
MOVX     @DPTR,A
MOV      DPTR,#PORTC
MOVX     @DPTR,A
MOV      P1,#0FFH
```

```
;-----INITIAL DATA ADDRESS-----
```

```
MOV      ADD51HIGH,#60H
MOV      ADD51LOW,#00H
MOV      ADDZ80HIGH,#0F0H
MOV      ADDZ80LOW,#10H
MOV      ADD51HR,#70H
MOV      ADD51LR,#00H
MOV      ADDZ80HW,#0F8H
MOV      ADDZ80LW,#00H
MOV      BYTEREAD,#14H
MOV      BYTEWRITE,#0FFH
MOV      BYTEREADFILE,#0FH
```

```
KEYINDEX: SETB      P1.7
          LCALL     DELAY
          MOV       P1,#8FH
          MOV       A,P1
          ANL       A,#0FH
          CJNE      A,#01H,KEYINDEX1
          LCALL     DELAY
          LCALL     DELAY
          MOV       DPTR,#0E000H
          MOVX      A,@DPTR
          MOV       ADDZ80HIGH,A
          LCALL     DELAY
          LCALL     DELAY
          MOV       DPTR,#0E001H
          MOVX      A,@DPTR
          MOV       ADDZ80LOW,A
          LCALL     DELAY
          LCALL     DELAY
          MOV       A,#0FH
          MOV       DPTR,#PORTC
          MOVX      @DPTR,A
          LJMP      MAINWRITE
KEYINDEX1: CJNE      R0,#02H,KEYINDEX
          LCALL     DELAY
          LCALL     DELAY
          MOV       DPTR,#0E000H
          MOVX      A,@DPTR
          MOV       ADDZ80HIGH,A
          LCALL     DELAY
          LCALL     DELAY
          MOV       DPTR,#0E001H
          MOVX      A,@DPTR
```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นไว้สำหรับ

ไม่ว่ากรณีใดๆ ทั้งสิ้นหากมีข้อผิดพลาดประการใดขออภัยเป็นอย่างสูงและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      ADDZ80LOW,A
LCALL    DELAY
LCALL    DELAY
MOV      A,#0F0H
MOV      DPTR,#PORTC
MOVX     @DPTR,A
LJMP     MAINREAD
;-----MAIN-----;
MAINREAD: LCALL    WRITETO51
          LCALL    DELAYTIME
          LJMP     TRNFERTOZ80
          LJMP     READFILE           ; TO SET AT F000H
MAINWRITE: LCALL    WRITETOZ80
          LCALL    DELAYTIME
          LJMP     TRNFERTO51
          LJMP     WRITEFILE         ; TO SET AT F000H
;-----WRITE-----;
WRITETOZ80: CLR     PORTSET
          LCALL    DELAY
          MOV      DPTR,#0F700H
          LCALL    WRITE
          MOV      DPTR,#0F800H
          LCALL    WRITE
          RET
WRITE:    MOV      R3,BYTEWRITE
          MOV      R4,#02H
          MOV      14H,#01H
LOOPW:    MOV      A,14H
          MOVX     @DPTR,A
          INC      DPL
          INC      14H
          DEC      R3
          CJNE     R3,#00H,LOOPW
          INC      DPL
          DJNZ     R4,LOOPW
          RET
WRITETO51: SETB     PORTSET
          LCALL    DELAY
          MOV      DPTR,#ADDREAD51
WRITE1:   MOV      R3,BYTEWRITE
          MOV      R4,#01H
          MOV      R5,#01H
LOOPW1:   MOV      A,R5
          MOV      A,#0FH
          MOVX     @DPTR,A
          INC      DPL
          DEC      R3
          CJNE     R3,#00H,LOOPW1
          INC      DPL
          DJNZ     R4,LOOPW1
          LCALL    DELAY
          RET
;-----TRANSFER FOR FILE-----;

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นไว้สำหรับการใช้งานเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น หากมีเหตุใดเปลี่ยนแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

READFILE:  SETB      PORTSET
           LCALL     DELAY
           MOV       DPTR,#6000H
           MOVBX     A,@DPTR
           MOV       ADDZ80HW,A
           MOV       10H,A
           MOV       DPTR,#6001H
           MOVBX     A,@DPTR
           MOV       ADDZ80LW,A
           MOV       11H,A
;-----FINE BYTE FROM END ADDRESS-----
           MOV       DPTR,#6002H
           MOVBX     A,@DPTR
           SUBB      A,10H
           MOV       15H,A
LOOPREAD:  MOV       A,15H
           CJNE      A,#00H,LOAD
           MOV       DPTR,#6003H
           MOVBX     A,@DPTR
           SUBB      A,11H
           MOV       BYTEREAD,A
           MOV       R4,BYTEREAD ; BYTE READ FROM
START-END MOV       DPH,ADDZ80HW ; FROM 6000H
           MOV       DPL,ADDZ80LW ; FROM 6001H
           PUSH      DPH
           PUSH      DPL
           MOV       DPH,ADD51HR ; 80H
           MOV       DPL,ADD51LR ; 00H
           MOVBX     A,@DPTR
           MOV       R7,A
           CLR       PORTSET
           LCALL     DELAY
           POP       DPL
           POP       DPH
           MOV       A,R7
           MOVBX     @DPTR,A
           SETB      PORTSET
           PUSH      DPH
           PUSH      DPL
LOOPRDFILE: POP       DPL
           POP       DPH
           INC       DPL
           INC       ADD51LR
           PUSH      DPL
           MOV       DPH,ADD51HR
           MOV       DPL,ADD51LR
           MOVBX     A,@DPTR
           CLR       PORTSET
           LCALL     DELAY
           MOV       DPH,ADDZ80HW ; ADDRESS Z-80 RECEIVER
           POP       DPL
           MOVX      @DPTR,A ; TRANSFER DATA RECEIVER
           SETB      PORTSET

```

เอกสารนี้เป็นเอกสารที่วางไว้สำหรับศึกษาเท่านั้น ADDRESS Z-80 RECEIVER

ไม่ว่ากรณีใดๆ ทั้งสิ้นทั้งห้ามมิให้นำไปลงนิตยสารและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีโอกาสนำไปใช้

```

MOV      R5,A
PUSH     DPH
PUSH     DPL
LCALL    DISPLAY
LCALL    DELAY
DJNZ     R4,LOOPRDFILE
LJMP     KEYLOOP
LOAD:    MOV      R4,#0FEH
MOV      DPH,ADDZ80HW      ; FROM 6000H
MOV      DPL,ADDZ80LW      ; FROM 6001H
PUSH     DPH
PUSH     DPL
MOV      DPH,ADD51HR      ; 80H
MOV      DPL,ADD51LR      ; 00H
MOVX     A,@DPTR
MOV      10H,A
CLR      PORTSET
LCALL    DELAY
POP      DPL
POP      DPH
MOV      A,10H
MOVX     @DPTR,A
SETB     PORTSET
PUSH     DPH
PUSH     DPL
LCHKFILE: POP      DPL
POP      DPH
INC      DPL
INC      ADD51LR
PUSH     DPL
MOV      DPH,ADD51HR
MOV      DPH,ADD51LR
MOVX     A,@DPTR
CLR      PORTSET
LCALL    DELAY
MOV      DPH,ADDZ80HW      ; ADDRESS Z-80 RECEIVER
POP      DPL
MOVX     @DPTR,A          ; TRANSFER DATA RECEIVER
SETB     PORTSET
MOV      R5,A
PUSH     DPH
PUSH     DPL
LCALL    DISPLAY
LCALL    DELAY
DJNZ     R4,LCHKFILE
MOV      DPH,ADD51HR
INC      DPH
MOV      ADD51HR,DPH
MOV      ADD51LR,#00H
MOV      DPH,ADDZ80HW
INC      DPH
MOV      ADDZ80HW,DPH

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ที่ผู้อื่นทำซ้ำโดยไม่ได้รับอนุญาตจะต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      ADDZ80LW,#00H
DEC      15H
LJMP     LOOPREAD
;-----WRITE FILE-----;
WRITEFILE: POP      DPL
          POP      DPH
          SETB     PORTSET
          MOV      DPTR,#6000H
          MOVX     A,@DPTR
          MOV      ADDZ80HW,A
          MOV      12H,A
          MOV      DPTR,#6001H
          MOVX     A,@DPTR
          MOV      ADDZ80LW,A
          MOV      13H,A
;-----FINE BYTE FROM END ADDRESS-----
MOV      DPTR,#6002H
MOVX     A,@DPTR
SUBB     A,12H
MOV      16H,A
WRITELOOP: MOV      A,16H
          CJNE     A,#00H,LOOP2
          LJMP     LOAD1
LOOP2:    MOV      A,#0FFH
          SJMP     LOAD3
LOAD1:    MOV      DPTR,#6003H
          MOVX     A,@DPTR
          SUBB     A,13H
LOAD3:    MOV      BYTEREAD,A
          MOV      R4,BYTEREAD
          MOV      DPH,ADD51HR      ; 80H
          MOV      DPL,ADD51LR      ; 00H
          PUSH     DPH
          PUSH     DPL
          CLR      PORTSET
          LCALL    DELAY
          MOV      DPH,ADDZ80HW      ; FROM 6000H
          MOV      DPL,ADDZ80LW      ; FROM 6001H
          MOVX     A,@DPTR
          MOV      R7,A
          SETB     PORTSET
          POP      DPL
          POP      DPH
          MOV      A,R7
          MOVX     @DPTR,A
          CLR      PORTSET          ; IMPORTANT
          LCALL    DELAY
          PUSH     DPH
          PUSH     DPL
          LOOPWDFILE: POP      DPL
          POP      DPH
          INC      DPL

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ที่ผู้อื่นทำผิดมีโทษตามกฎหมายและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

INC      ADDZ80LW
PUSH     DPL
MOV      DPH,ADDZ80HW          ; ADDRESS Z-80 SEND
MOV      DPL,ADDZ80LW
MOVX     A,@DPTR
SETB     PORTSET
MOV      DPH,ADD51HR
POP      DPL
MOVX     @DPTR,A
MOVX     A,@DPTR
MOV      R5,A
PUSH     DPH
PUSH     DPL
MOV      R5,A
LCALL    DISPLAY
LCALL    DELAY
LCALL    DELAY
CLR      PORTSET
LCALL    DELAY
DJNZ     R4,LOOPWDFILE
MOV      A,16H
CJNE     A,#00H,LOOPP
LJMP     KEYLOOP
LOOPP:   MOV      DPH,ADD51HR
INC      DPH
MOV      ADD51HR,DPH
MOV      ADD51LR,#00H
MOV      DPH,ADDZ80HW
INC      DPH
MOV      ADDZ80HW,DPH
MOV      ADDZ80LW,#00H
DEC      16H
LJMP     WRITELoop

```

LOOPP:

;-----TRANSFER DATA RAM TO MCS-51-----
 TRANFERTO51:

```

CLR      PORTSET
LCALL    DELAY
MOV      R4,BYTEREAD          ; BYTE FILE 14H
DEC      R4
MOV      DPH,ADD51HIGH        ; 60H RECEIVER
MOV      DPL,ADD51LOW         ; 00H RECEIVER
PUSH     DPH
PUSH     DPL
MOV      DPH,ADDZ80HIGH        ; 0F0H TRANSFER
MOV      DPL,ADDZ80LOW         ; 00H TRANSFER
MOVX     A,@DPTR
MOV      10H,A
SETB     PORTSET
LCALL    DELAY

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น หากมีข้อสงสัยหรือต้องการข้อมูลเพิ่มเติม กรุณาติดต่อเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

POP      DPL
POP      DPH
MOV      A,10H

```

```

MOVX      @DPTR,A
CLR       PORTSET
LCALL     DELAY
PUSH      DPH
PUSH      DPL
LOOPRR51: POP      DPL
          POP      DPH
          INC      DPL
          INC      ADDZ80LOW
          PUSH     DPL
          MOV      DPH,ADDZ80HIGH      ; ADDRESS Z-80 SEND
          MOV      DPL,ADDZ80LOW
          MOVX     A,@DPTR
          SETB     PORTSET
          MOV      DPH,ADD51HIGH
          POP      DPL
          MOVX     @DPTR,A
          MOVX     A,@DPTR
          MOV      R5,A
          PUSH     DPH
          PUSH     DPL
          CLR      PORTSET
          LCALL    DELAY
          DJNZ     R4,LOOPRR51
          PUSH     DPH
          PUSH     DPL
          LJMP     WRITEFILE
;-----TRANSFER DATA RAM TO MCS-51-----
TRANFERTOZ80:
          CLR      PORTSET
          LCALL    DELAY
          MOV      ADD51HR,#70H      ; 80H
          MOV      ADD51LR,#00H      ; 00H
          MOV      R4,BYTE READ      ; BYTE FILE 14H
          DEC      R4
          MOV      DPH,ADD51HIGH      ; 60H RECEIVER
          MOV      DPL,ADD51LOW      ; 00H RECEIVER
          PUSH     DPH
          PUSH     DPL
          MOV      DPH,ADDZ80HIGH      ; 0F0H TRANSFER
          MOV      DPL,ADDZ80LOW      ; 00H TRANSFER
          MOVX     A,@DPTR
          MOV      10H,A
          SETB     PORTSET
          LCALL    DELAY
          POP      DPL
          POP      DPH
          MOV      A,10H
          MOVX     @DPTR,A
          CLR      PORTSET
          LCALL    DELAY
          PUSH     DPH

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้นทางผู้จัดทำขอสงวนสิทธิ์ในเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

      PUSH      DPL
LOOPRRT051: POP      DPL
      POP      DPH
      INC      DPL
      INC      ADDZ80LOW
      PUSH     DPL
      MOV      DPH,ADDZ80HIGH      ; ADDRESS Z-80 SEND
      MOV      DPL,ADDZ80LOW
      MOVX     A,@DPTR
      SETB     PORTSET
      MOV      DPH,ADD51HIGH
      POP      DPL
      MOVX     @DPTR,A
      MOVX     A,@DPTR
      MOV      R5,A
      PUSH     DPH
      PUSH     DPL
      CLR      PORTSET
      LCALL    DELAY
      DJNZ     R4,LOOPRRT051
      LJMP     READFILE
DISPLAY: MOV      A,R5
      MOV      DPTR,#PORTC
      MOVX     @DPTR,A
      RET
DELAY:  MOV      R1,#28H
      MOV      R1,#35H      ; FOR WRITE
DELAY1: MOV      R2,#0FFH
      DJNZ     R2,$
      DJNZ     R1,DELAY1
      RET
DELAYTIME: MOV    R1,#25H
DELAYTIME1: MOV    R2,#0FFH
      DJNZ     R2,$
      DJNZ     R1,DELAYTIME1
      RET
END1:   SETB     PORTSET
      SJMP     END1
      END

```

รูปที่ 4.12 โปรแกรมการป้อนค่าตำแหน่งเริ่มต้นและตำแหน่งสุดท้ายในการโอนย้ายข้อมูล

1.2) ผลการทดลอง

LD	A, 80H	
OUT	(43H), A	
LD	A, 01H	; COMMAND ADDRESS START
OUT	(40H), A	
LD	A, 00H	; INDEX ADDRESS BYTE LOW

OUT	(41H), A
LD	A, F1H
OUT	(42H), A

รูปที่ 4.13 โปรแกรมที่ป้อนให้ Z-80

<M>	F100H : F8 00 F8 1F
-----	---------------------

; กำหนดตำแหน่งเริ่มต้น และสุดท้าย

<M>	F800H : 00 01 02 03
-----	---------------------

; ผลข้อมูลที่ตำแหน่ง F800H

4.2 การทดลองภาค FDC Processor

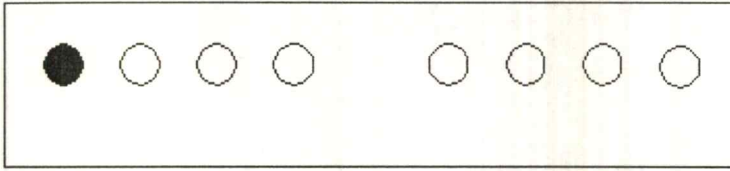
1.1) โปรแกรมการทดลอง

การทดลอง FDC โปรเซสเซอร์ที่สำคัญคือการอ่านค่าออกมาจาก Main Status Register ซึ่งจะบอกความพร้อมในการทำงานของ FDC โปรเซสเซอร์โดยทดลองป้อนคำสั่งการทำงานดังนี้คือ

	ORG	0000H
P_A	EQU	C000H
P_B	EQU	C001H
P_C	EQU	C002H
CNTR	EQU	C003H
STATYSR	EQU	AFF0H
DATAR	EQU	AFF1H
OUTPUTR	EQU	AFF3H
	MOV	A, 82H
	MOV	DPTR, #CNTR
	MOVX	@DPTR, A
	MOV	DPTR, #STATYSR
	MOVX	A, @DPTR
	MOV	DPTR, #P_A
	MOVX	@DPTR, A
	SJMP	\$
	END	

รูปที่ 4.14 โปรแกรมป้อนคำสั่งการทำงานของภาค FDC โปรเซสเซอร์

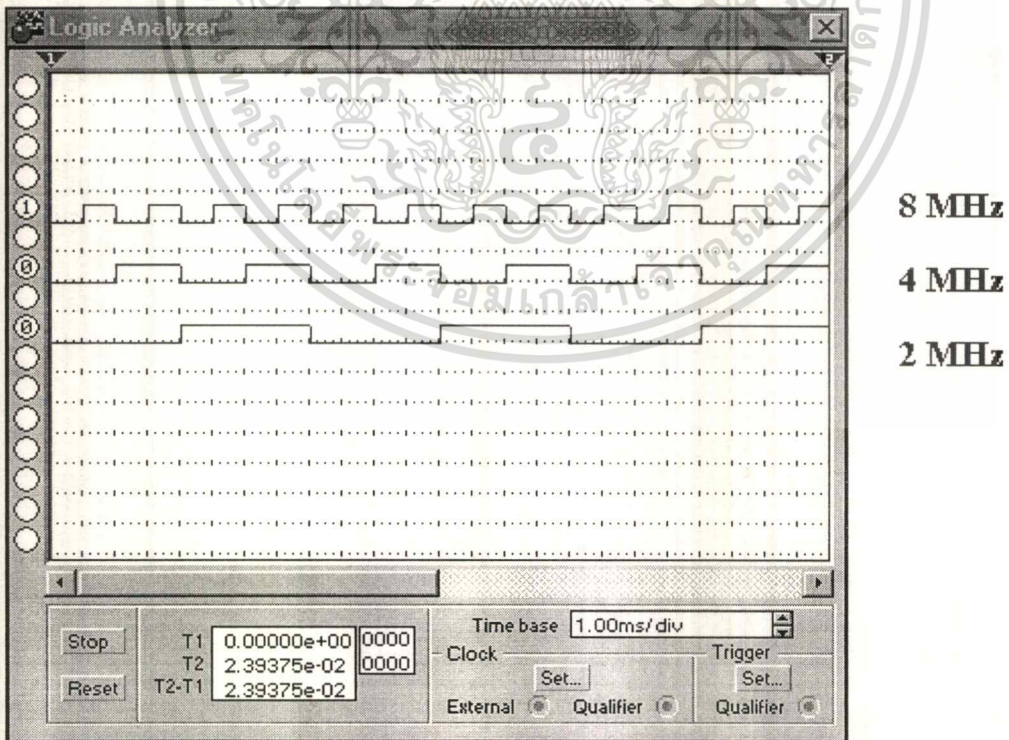
1.2) ผลการทดลอง



จากผลการทดลอง ค่าที่อ่านได้จากกริดเตอร์แสดงผล คือ 10000000B หรือ 80H แสดงว่าขณะนี้ FDC Processor พร้อมที่จะทำงานแล้ว

4.3 การทดลองภาคกำเนิดสัญญาณพิก้า

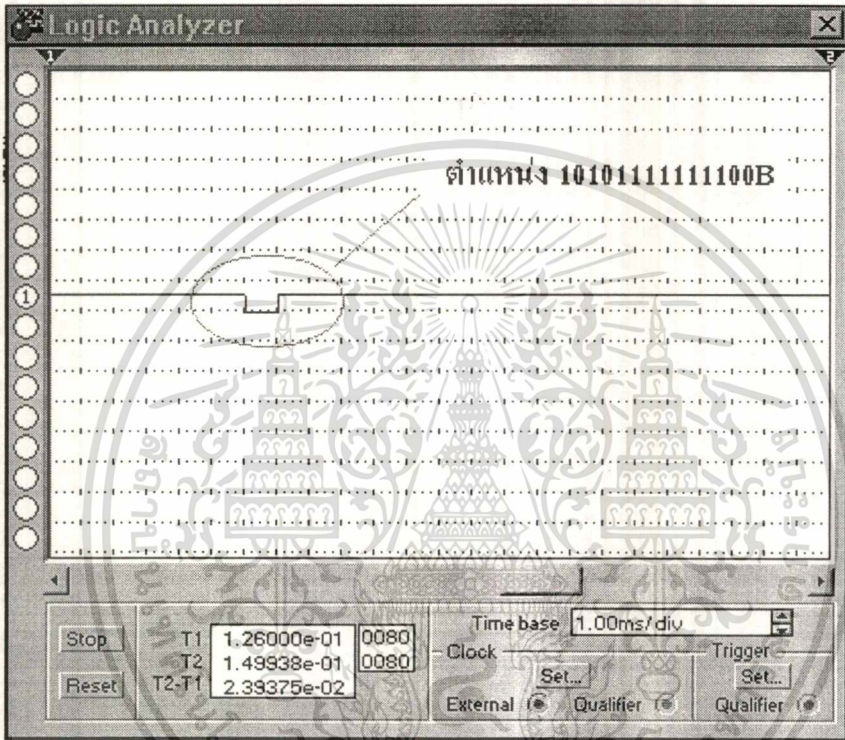
จากบทที่ 3 จะเห็นว่าสัญญาณพิก้าที่ใช้ในวงจรนี้ทั้งหมด 3 ความถี่คือ 8MHz, 4MHz และ 2MHz โดยสามารถตรวจวัดได้ที่ขา 12 ของ U7, ขา 12 ของ U6 และ ขา 9 ของ U6 ตามลำดับ ซึ่งได้ผลดังนี้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
รูปที่ 4.15 ความถี่ 8 MHz, 4 MHz และ 2 MHz ที่ได้จากวงจรกำเนิดสัญญาณพิก้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามเผยแพร่และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 การทดลองภาค Decoder

จากการทำงานภาค Decoder จะให้สภาวะลอจิก “0” เมื่ออินพุตเป็น 1010111111100B ซึ่ง Input Address 1 จะต่อเข้ากับ 5 – Address 2 ตามลำดับ ทดลองนำวงจร Simulate ผ่านทางโปรแกรม Electronic Workbench ได้ผลดังนี้



รูปที่ 4.16 ผลจากการทำงานของภาค Decoder

จากรูป จะเห็นว่าส่วนที่เกิดการแอกทีฟจะเป็นส่วนที่มีข้อมูลอินพุตเป็น 10101111111B ซึ่งหมายถึง วงจรได้ออกแบบถูกต้องแล้ว

4.5 การทดลองภาคเอาต์พุตริจิสเตอร์

ภาคเอาต์พุตริจิสเตอร์ ประกอบด้วยบิตต่างๆ ดังนี้คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.2 บิตต่างๆ ที่ได้จากภาคเอาต์พุตรีจิสเตอร์

บิต	D7	D6	D5	D4	D3	D2	D1	D0
หน้าที่	NC	NC	MT1	MT0	DRQ	RESET	NC	FDSEL

```
ORG 0000H
P_A EQU C000H
P_B EQU C001H
P_C EQU C002H
CNTR EQU C003H
OUTPUTR EQU AFF3H
TEST EQU 00000000B
; Initial 8255
MOV A, #85H
MOV DPTR, #CNTR
MOVX @DPTR, A
; Send Data to Output Register
MOV A, #TEST
MOV DPTR, #OUTPUTR
MOVX @DPTR, A
SJMP $
END
```

รูปที่ 4.17 โปรแกรมการส่งค่าคงที่ให้กับเอาต์พุตรีจิสเตอร์

จากโปรแกรมเป็นการส่งค่าคงที่ให้กับเอาต์พุตรีจิสเตอร์ โดยได้ผลการทดลองดังนี้

ตารางที่ 4.3 ผลการทดลองภาคเอาต์พุตรีจิสเตอร์

ค่าคงที่	ผลที่ได้
00000000B	FDC ไม่ทำงานอยู่ในสภาวะรีเซต
00000100B	FDC พร้อมทำงาน ขารีเซตไม่แอกทีฟ
00100101B	ดิสก์ไดรฟ์หมุนและแสดงไฟการทำงาน
00000101B	ดิสก์ไดรฟ์หยุดการทำงาน ไฟแสดงผลดับ
00001101B	สัญญาณ DRQ, DACK และ TC สามารถแสดงผลได้
00000101B	สัญญาณ DRQ, DACK และถูกยกเลิก มีสถานะลอจิก “0”

บทที่ 5

บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา

5.1 บทสรุป

ในวงจรดิจิทัล 3.5 นิ้ว สำหรับซิงเกิลบอร์ด เป็นวงจรที่พัฒนาให้ซิงเกิลบอร์ดมีการเก็บข้อมูลที่ใช้น้อยเก็บข้อมูลเป็นลักษณะของฮาร์ดแวร์ที่มีความคงทนในการเก็บข้อมูลมากกว่าที่เป็นอยู่ในปัจจุบัน ซึ่งในที่นี้เราจะหมายถึงดิจิทัล 3.5 นิ้ว หมายถึงเราจะต้องมีวงจรที่ออกแบบให้เลียนแบบการทำงานของคอมพิวเตอร์ที่ใช้ในปัจจุบัน การสร้างสัญญาณควบคุมต่างๆ เลียนแบบคอมพิวเตอร์นั้นไม่ค่อยเป็นปัญหามาก เนื่องจากการออกแบบวงจรใช้ไมโครคอนโทรลเลอร์ในการควบคุมการทำงานของวงจรอยู่เป็นเดิมที หมายถึงว่าสัญญาณต่างๆ ที่มีอยู่ในบัสต่างๆ มีลักษณะที่ใกล้เคียงกัน

5.2 ปัญหา และแนวทางแก้ไข

ปัญหาที่เกิดขึ้นในการจัดทำโครงการ และแนวทางการแก้ไขสามารถแบ่งได้ตามหัวข้อดังต่อไปนี้

5.2.1 แหล่งอ้างอิงข้อมูล

1) ปัญหา ในการ จัดหาข้อมูลเกี่ยวกับดิจิทัล 3.5 นิ้ว และวงจร FDC คอนโทรลเลอร์เนื่องจากในหนังสือคอมพิวเตอร์โดยปกติแล้ว การที่จะอ้างอิงการทำงานของฮาร์ดแวร์ ในส่วนของการควบคุมดิจิทัล 3.5 นิ้วโดยตรงนั้นมีไม่มากนัก หากพูดถึงเฉพาะภาษาไทยแล้ว หนังสือที่เรียกได้ว่าเจาะข้อมูลค่อนข้างลึกมากที่สุดในการทำงานของฮาร์ดแวร์ของคอมพิวเตอร์แล้ว คงจะเป็นหนังสือการเขียนโปรแกรมภาษาแอสเซมบลี เนื่องจากหนังสือประเภทนี้เป็นการเขียนควบคุมฮาร์ดแวร์โดยตรงว่าการเขียนควบคุมโดยภาษาอื่น ทั้งนี้การเขียนโปรแกรมผ่านทางภาษาแอสเซมบลียังมีข้อเสียในการอ้างอิงข้อมูล นั่นคือ การเขียนโปรแกรมเหล่านี้เป็นการเขียนโดยอ้างผ่านรีจิสเตอร์ของคอมพิวเตอร์และใช้การบริการอินเตอร์พดท์เข้าช่วย ดังนั้นการใช้หนังสือที่อ้างอิงเหล่านี้ดูเหมือนจะมีประโยชน์ค่อนข้างน้อย ในส่วนของหนังสือที่อ้างอิงส่วนฮาร์ดแวร์โดยตรงนั้น เป็นข้อมูลที่ค่อนข้างเก่ามาก กล่าวคือ การเขียนอ้างอิงส่วนใหญ่จะเป็นระบบที่ใช้กับโปรเซสเซอร์ประมาณรุ่น 80286-80386 ซึ่งค่อนข้างจะเก่ามาก หากเทียบกับโปรเซสเซอร์ที่ใช้ในปัจจุบัน (อ้างอิงปี พ.ศ. 2542) แต่กระนั้นอาจยังไม่ใช่ปัญหาหลัก เพราะการใช้ข้อมูลเก่าเป็นปัญหาที่เราไม่สามารถหาข้อมูลที่พัฒนาบ่อยๆ เช่น

ข้อมูลจากอินเทอร์เน็ต หรือ จากหนังสือภาษาอังกฤษใหม่ๆ แต่ปัญหาของหนังสือประเภทนี้มีการนำเสนอข้อมูลที่ผิดพลาด จากที่ได้ ทราบกันมาแล้วว่า FDC โปรเซสเซอร์ที่เลือกใช้ในวงจรนี้เป็นเบอร์ μ PD765 (ของ NEC) หรือ 8272A (ของ Intel) FDC โปรเซสเซอร์ซึ่งเบอร์ดังกล่าวเป็นโปรเซสเซอร์ที่ใช้ในบอร์ดแม่ของซีพียูรุ่น 80286–80386 การแสดงข้อมูลเมื่อนำมาเทียบกับทฤษฎีที่ได้จากหนังสือยังมีข้อผิดพลาดหลายจุด ตั้งแต่การจัดตั้งสัญญาณเบื้องต้นให้กับ FDC โปรเซสเซอร์หรือการจัดอุปกรณ์เพื่อการมอดูเลต อุปกรณ์เหล่านี้มีข้อผิดพลาดเกือบทั้งสิ้น

ในส่วนของหนังสือภาษาอังกฤษนั้น จากการค้นหาข้อมูลจากอินเทอร์เน็ต มีการแนะนำรายชื่อหนังสือที่สามารถอ้างอิงข้อมูลเกี่ยวกับ FDC ได้จากหนังสือทั้งสิ้น 16 เล่มดังนี้ คือ

1. The Users Guide for the Floppy Disk Controller Plus
2. The 1989 National Semiconductor Mass Storage Hand Book
3. The NEC 1981 Catalog for Microcomputer
4. The NEC 1987 Microcomputer Products Databook
5. The IBM Technical Reference Manual for PC
6. The IBM Technical Reference; Personal Computer AT
7. The Western Digital Databook
8. The 1983 INTEL Microprocessor and Peripheral Handbook
9. The 1985 INTEL Microsystems Component Handbook Volume II
10. The 1987 INTEL 80286 and 80287 Programmer's Reference Manual
11. The 1984 ROCKWELL Databook
12. The IBM PC Troubleshooting & Repair
13. The IBM PC Advance Troubleshooting & Repair
14. The IBM Personal Computer Troubleshooting & Repair
15. 8088 Assembly Language Programming
16. IBM ROM BIOS: Programmer's Quick Reference Series

จากรายการหนังสือทั้งหมดที่แสดงมานี้ มีเพียงเล่มเดียวที่สามารถหาได้ นั่นคือ รายการหนังสือลำดับที่ 13 คือ The IBM PC Advance Troubleshooting & Repair ซึ่งเป็นหนังสือที่เก่าพอสมควร เหตุที่ทางคณะผู้จัดทำใช้ FDC โปรเซสเซอร์ที่เก่าขนาดนั้น เนื่องจาก FDC โปรเซสเซอร์ที่เป็นเบอร์ใหม่นั้นมีการจัดทำลงในรูปแบบของชิป จึงหมายถึงว่า มีการรวมเอาฟังก์ชันการทำงานของ I/O ส่วนอื่นเข้ามารวมกับชิปด้วย อาทิเช่น วงจรพอร์ตขนาน, พอร์ตอนุกรม และ IDE เป็นต้น การเข้ามารวมในชิปเพียงตัวเดียว แต่ที่สำคัญที่สุดนั่นคือการที่ FDC โปรเซสเซอร์ที่เป็นลักษณะนี้ไม่มี

จำหน่ายในท้องตลาด จากที่สำรวจตลาดพบว่า FDC โปรเซสเซอร์ที่มีขายนั้นมีเพียง 8272A และ 8473 (ของ National Semiconductor ซึ่งทางคณะผู้จัดทำทราบถึงแหล่งจำหน่ายภายหลังจากการเลือกใช้เบอร์ 8272A ไปแล้วเป็นเวลานานพอควร)

นอกจากสื่อที่เป็นหนังสือแล้ว ได้มีการจัดหาข้อมูลจากทางอินเทอร์เน็ต ซึ่งเป็นข้อมูลที่ค่อนข้างจะเป็นจุดหลักมากที่สุดในการอ้างอิง แต่ 8272A และ μ PD765 นั้นเป็นโปรเซสเซอร์ที่ผลิตขึ้นมานานแล้ว (ประมาณ 12 ปี) Database ของบริษัทผู้ผลิตจึงหาได้น้อย โดยเฉพาะอย่างยิ่งข้อมูลประเภท Datasheet นั้น ไม่มีเลยทีเดียวทั้งจากการค้นหาจากบริษัทผู้ผลิตโดยตรงเอง หรือจากการหาข้อมูลผ่านทาง Web Site ที่ใช้ค้นหาข้อมูล (เช่น Yahoo, Thai Find, Hot Search ฯลฯ) ข้อมูลที่หาได้ค่อนข้างมากที่สุดจะเป็นคำสั่งการเขียนโปรแกรมค่อนข้างมาก โดยเฉพาะจากบริษัท SMSC (<http://www.smsc.com>) จะมีโปรแกรมทดสอบ ให้มาอ้างอิงบ้าง แต่จะมีข้อเสียที่ข้อมูลเหล่านี้ไม่ได้อ้างการเขียนโปรแกรมเพื่อควบคุมคิสก์ไครฟ์ขนาด 5 นิ้ว และ 8 นิ้ว ซึ่งใช้ในระบบ คอมพิวเตอร์รุ่น 80286

2) แนวทางแก้ไข

การหาแหล่งข้อมูลดังที่ได้กล่าวมาแล้วว่า ใช้การค้นหาข้อมูลจากอินเทอร์เน็ตเป็นตัวหลัก เนื่องจากสามารถค้นหาข้อมูลได้มากกว่าการหาข้อมูลจากหนังสือ และในส่วนของการอ้างอิงการเขียนโปรแกรมเป็นการอ้างโปรแกรมทดสอบจากทางบริษัท SMSC นั้น เป็นคิสก์ไครฟ์ ขนาด 5 นิ้ว และ 8 นิ้ว การแก้ปัญหาทั้งทางด้านข้อมูลและตัวอย่างทำได้โดยการหาข้อมูลอ้างอิงจาก FDC โปรเซสเซอร์ที่รองรับ (Compatible) โปรแกรมที่ใช้กับ μ PD765 แทน ซึ่งสามารถใช้รองรับกันได้พอสมควร

5.2.2 อุปกรณ์ใช้งาน

1) ปัญหา อุปกรณ์ที่ใช้งานในวงจร FDC นั้นเรียกได้ว่าไม่ค่อยจะเป็นที่สนใจมากนักในบรรดานักเขียนโปรแกรมของเมืองไทย เนื่องจาก การเขียนโปรแกรมจัดการระบบนั้นจะใช้ผ่านช่องบริการ (Interrupt Service) ของคอมพิวเตอร์ อีกทั้งในกรณีที่การ์ด FDC ในคอมพิวเตอร์เกิดเสียหาย จะทำการซื้อการ์ดเปลี่ยนมากกว่าการซ่อมแซมการ์ด อุปกรณ์ที่ใช้ประกอบวงจร FDC จึงมีจำกัด เช่น ไอซีที่ทำหน้าที่เป็น Data Separator ไม่สามารถหาซื้อได้ หรือ FDC โปรเซสเซอร์ที่เป็นรุ่นใหม่ๆ ไม่มีจำหน่าย

2) แนวทางแก้ไข

ในกรณีของไอซี Data Separator นั้นได้ทำการเปลี่ยนวงจรในส่วนนี้ จากไอซีตัวเดียวเป็นวงจรที่ประกอบจากไอซีที่ทำได้จากท้องตลาด นอกจากนั้นในส่วนของ FDC โปรเซสเซอร์ได้ใช้ของเดิมคือ 8272A แทน โดยพยายามหาข้อมูลอ้างอิงที่สามารถทดแทนกันได้ให้ได้มากที่สุด

5.2.3 จุดบกพร่องจากการทำงาน

1) ปัญหา วิธีการแก้ไขจุดบกพร่องจากการทำงานเป็นจุดด้อยที่เป็นผลพวงจากแหล่งข้อมูล เนื่องจากการทำงานทั้งหมดเกิดจาก FDC โปรเซสเซอร์เป็นหลัก ดังนั้นในกรณีที่เกิดความผิดพลาดจากตัว FDC โปรเซสเซอร์จะหาจุดบกพร่องได้ยาก เช่น กรณีการเกิด Equipment Check ขึ้นในขั้น Result Phase ของการทำงาน ซึ่งหมายถึงสัญญาณ Ready ของ FDD เกิดการเปลี่ยนแปลงสถานะต่างๆ ที่ขา Ready ถูกต่อเข้ากับลอจิก “1” ตลอดเพื่อให้สถานะ Ready Active ตลอดการทำงาน

2) แนวทางแก้ไข

ในกรณีที่เกิดการ Equipment Check ขึ้น และตรวจสอบฮาร์ดแวร์แล้วว่าไม่มีข้อผิดพลาด จะทำการสั่งรีเซ็ต FDC ใหม่อีกครั้งก็สามารถที่จะแก้ไขปัญหาดังกล่าวได้ในกรณีของการผิดพลาดทาง FDC โปรเซสเซอร์ก็จะใช้วิธีการเดียวกัน โดยจะทำการเรียกคำสั่งนั้นซ้ำอีกครั้งหนึ่ง

5.2.4 หัวอ่าน/เขียน ไม่เคลื่อนที่จากคำสั่ง

1) ปัญหา

ปกติการทำให้หัวอ่านหรือเขียนของฟลอปปีดิสก์เคลื่อนที่ได้มันจะต้องสร้างสัญญาณพัลส์ป้อนให้กับฟลอปปีดิสก์ไครฟ์ทางขา Step Pulse (ขา 20) ซึ่งสัญญาณดังกล่าวได้มาจากการคำนวณของ FDC โปรเซสเซอร์

2) การแก้ไขปัญหา

สาเหตุที่ FDC โปรเซสเซอร์ไม่สร้างสัญญาณพัลส์ให้กับฟลอปปีดิสก์ไครฟ์เนื่องจาก ขา Ready ของ FDC โปรเซสเซอร์ไม่แอคทีฟ High แก้ไขโดยการป้อนสัญญาณลอจิก “1” ให้

5.2.5 เกิดสัญญาณอินเตอร์รัพต์ในการเคลื่อนหัวเขียน/อ่าน

1) ปัญหา

หลังจากทำคำสั่งในการเคลื่อนหัวอ่าน เช่น Recalibrate, Seek หรือ Scan จะเกิดสัญญาณอินเตอร์รัพต์ขึ้น พร้อมทั้งแจ้งสถานะทางแฟลกของรีจิสเตอร์แสดงสถานะหลักว่า FDD Busy ซึ่งทำให้ไม่มีการเขียนหรืออ่าน

2) การแก้ไขปัญหา

ส่งคำสั่ง Sense Interrupt Status เข้าไปเคลียร์สัญญาณอินเตอร์รัพต์ ซึ่งจะส่งผลให้แฟลก FDD Busy ถูกเคลียร์ไปด้วย

5.2.6 ความเร็วในการส่งผ่านข้อมูลไม่เพียงพอ

1) ปัญหา

เป็นปัญหาในส่วนของการโอนย้ายข้อมูลระหว่างไมโครคอนโทรลเลอร์กับวงจรภาค FDC เนื่องจากความเร็วในการส่งผ่านข้อมูลที่ใช้ในการส่งถ่ายข้อมูลจะต้องมีความเร็วอย่างต่ำ 1Mbps แต่การทำงานของ MCS-51 ตอบสนองไม่เพียงพอ เนื่องจากใน 1 คำสั่งของ MCS-51 ใช้สัญญาณนาฬิกาอย่างต่ำ 12 ลูกหรือเท่ากับ Clock (Hz) / 12 ต่อ 1 คำสั่ง ซึ่งความเร็วในการทำงานของ MCS-51 สามารถใช้สัญญาณนาฬิกาความถี่สูงสุดคือ 40 MHz แต่ก็ยังไม่สามารถให้บริการได้ทัน หากนับจำนวนคำสั่งและนับจำนวนสัญญาณนาฬิกาที่ใช้แล้ว

2) การแก้ปัญหา

ทดลองเปลี่ยนเบอร์ของ MCS-51 จากเดิม 8031 เป็น DS80C320 แทน ซึ่งเบอร์ดังกล่าว ใช้สัญญาณนาฬิกาน้อยกว่า คือ 4 ลูกต่อคำสั่ง ซึ่งน้อยกว่า 8031 ถึง 3 เท่า แต่มีปัญหาเรื่องการออกคำสั่งควบคุม FDC Processor เนื่องจากคาบเวลาเร็วมากเกินไปกว่าการรับคำสั่งและประมวลผลของ FDC ซึ่งได้กล่าวไว้ในทฤษฎีแล้วว่า การทำงานของ FDC Processor จะแบ่งเฟสการทำงานออกเป็น 3 เฟสคือ Command Phase, Execution Phase และ Result Phase ตามลำดับ ซึ่งในแต่ละลำดับจะมีลำดับเวลาที่ไม่เท่ากัน โดยทั้ง 3 เฟสใช้การทำงานผ่านรีจิสเตอร์ตัวเดียวกันนั่นคือ Data Register ซึ่งทำงานเรียงลำดับกันไป โดยปกติการโอนย้ายข้อมูลในคำสั่ง Write Command หรือ Read Command ปกติจะใช้การส่งข้อมูลโดยใช้ DMA เข้าช่วย เนื่องจากมีความเร็วในการโอนย้ายข้อมูลที่ค่อนข้างเร็วมาก ซึ่งถูกใช้ในการส่งข้อมูลของคอมพิวเตอร์ การแก้ปัญหานี้ทางที่นำจะได้ผลคือการทำงานแบบ DMA (ปกติใช้แบบ Non-DMA) จึงเท่ากับว่าเป็นการเลียนแบบการทำงานของคอมพิวเตอร์ทั้งภาคมาใช้นั่นเอง

5.3 แนวทางการพัฒนา

การทำงานของดิสก์ไดรฟ์ 3.5" สำหรับซิงเกิลบอร์ด โดยปกติสามารถนำไปประยุกต์ใช้เป็นองค์ประกอบของงานที่ใหญ่ขึ้นได้มาก เช่นการบันทึกข้อมูลต่างๆ โดยเก็บข้อมูลจากตัวไมโครโปรเซสเซอร์ที่ใช้ควบคุมงานอุตสาหกรรม งานวิจัยต่างๆ เป็นต้น แนวทางการพัฒนาการทำงานให้มีประสิทธิภาพในการทำงานที่สูงขึ้นมีดังนี้

1) เปลี่ยนเบอร์ FDC โปรเซสเซอร์ ที่มีประสิทธิภาพการทำงานที่สูงกว่า ปัจจุบันมี FDC โปรเซสเซอร์หลายเบอร์ที่มีการพัฒนาการทำงานดีกว่าเบอร์ 8272A หรือ μ PD765 ซึ่งเป็น FDC โปรเซสเซอร์รุ่นเก่า โดยแนะนำเบอร์ DP8473 เนื่องจากมีจำหน่ายในท้องตลาด และเป็นเบอร์ที่รองรับทางด้านซอร์ฟแวร์กับ μ PD765 และ 8272A โดยข้อดีของเบอร์ดังกล่าว จะเป็นตัวที่รวมเอาวงจรที่สำคัญ

ทั้งหมดลงไว้ โดยเฉพาะอย่างยิ่งสามารถต่อโดยตรงเข้ากับ FDD ได้โดยที่ไม่จำเป็นต้องผ่านเกตหรือไอซีต่างๆ สามารถค้นหาข้อมูลได้จาก Web Site ของ National Semiconductor สามารถอ้างอิง Datasheet และ Application Note ได้มากกว่าเบอร์ μ PD765 หรือ 8272A

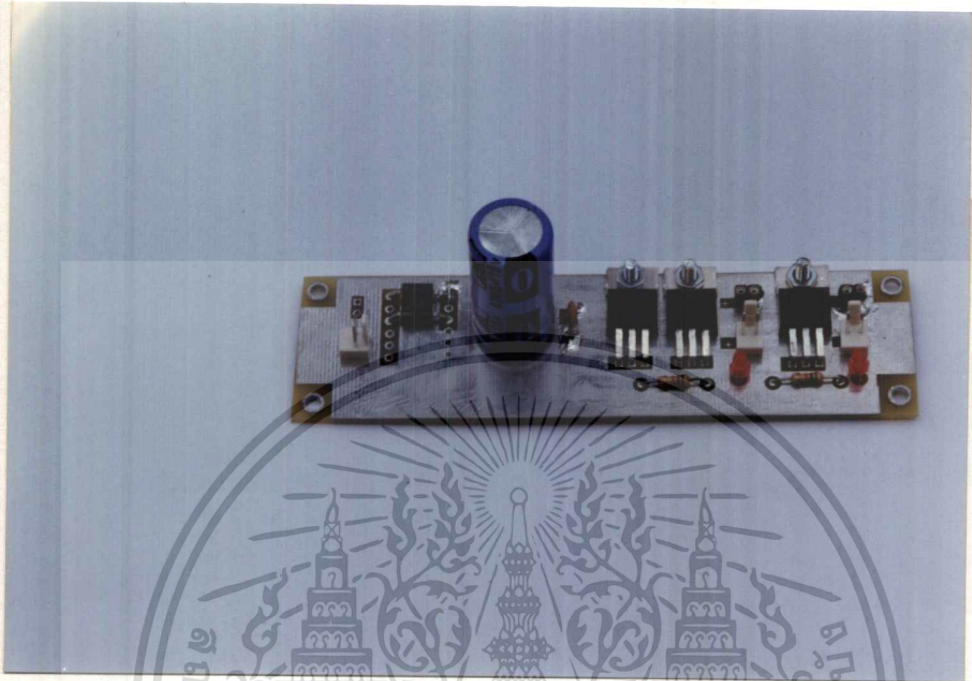
2) เพิ่มการทำงานแบบ DMA จากการแจ้งปัญหาให้ทราบว่า การโอนย้ายข้อมูลไม่ทันเนื่องจากการทำงานแบบ Non-DMA หากต้องการพัฒนากับไมโครคอนโทรลเลอร์ควรใช้การส่งข้อมูลแบบ DMA เข้าไปแทนที่ โดยหากต้องการหลีกเลี่ยงการใช้ไอซีแบบชิป ขอแนะนำเบอร์ 8237 ซึ่งเป็นตัวถังแบบ 40 ขา ซึ่งใช้ใน Main board รุ่นประมาณ 80286 – 80386 ซึ่งมีข้อมูลที่สามารถอ้างอิงทั้งที่เป็นทั้งภาษาไทยและภาษาอังกฤษอยู่ค่อนข้างมาก

3) เปลี่ยนการควบคุมวงจรจากที่ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นการควบคุมโดยไมโครโปรเซสเซอร์ เช่น Z-80 หรืออื่นๆ เนื่องจากการทำงานแบบ DMA จะมีการใช้สัญญาณพิเศษหลายสัญญาณที่ไมโครคอนโทรลเลอร์ไม่มีการสร้าง เช่น BUSRQ BUSACK IRQ เป็นต้น การใช้ไมโครโปรเซสเซอร์จะได้เปรียบกว่าการใช้ไมโครคอนโทรลเลอร์ที่สามารถเลียนแบบการทำงานได้ใกล้เคียงกับคอมพิวเตอร์ได้ค่อนข้างสูง เนื่องจากมีสัญญาณที่ใช้แบบเดียวกัน

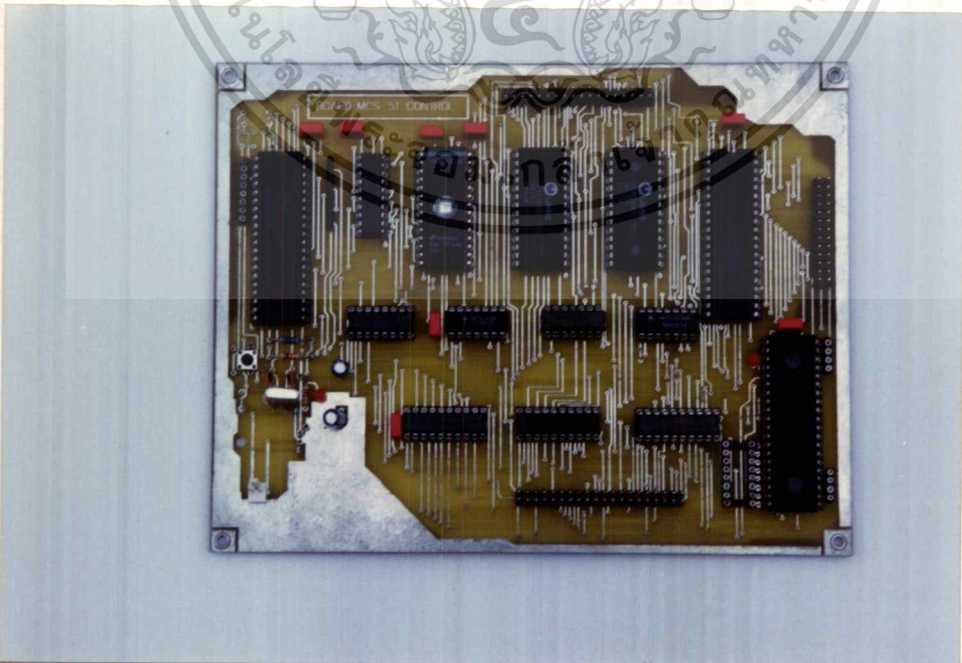
ในส่วนของการประยุกต์ใช้งานสามารถดัดแปลงได้หลายรูปแบบ เช่น ทำเครื่อง Format Floppy Disk, เครื่อง Pre-Format Floppy Disk เนื่องจากการทดลองพบว่าการเขียนหรืออ่านโดยการควบคุมผ่านทาง FDC โปรเซสเซอร์สามารถเขียนข้อมูลได้ตั้งแต่เซกเตอร์แรกของแผ่นเป็นต้นไป ดังนั้นเพียงกำหนดรูปแบบของข้อมูลไว้แล้วเขียนทับลงบนแผ่นที่ FAT หรือ Boot Sector ที่มีปัญหา ก็สามารถกระทำได้ การบันทึกข้อมูลจากอุปกรณ์ต่างๆ เป็นรูปแบบของฟลอปปี้ดิสก์ โดยเปลี่ยนแปลงจากการรับข้อมูลที่ได้จากการโอนย้ายข้อมูลเป็นการรับข้อมูลในส่วนที่ต้องการแทน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

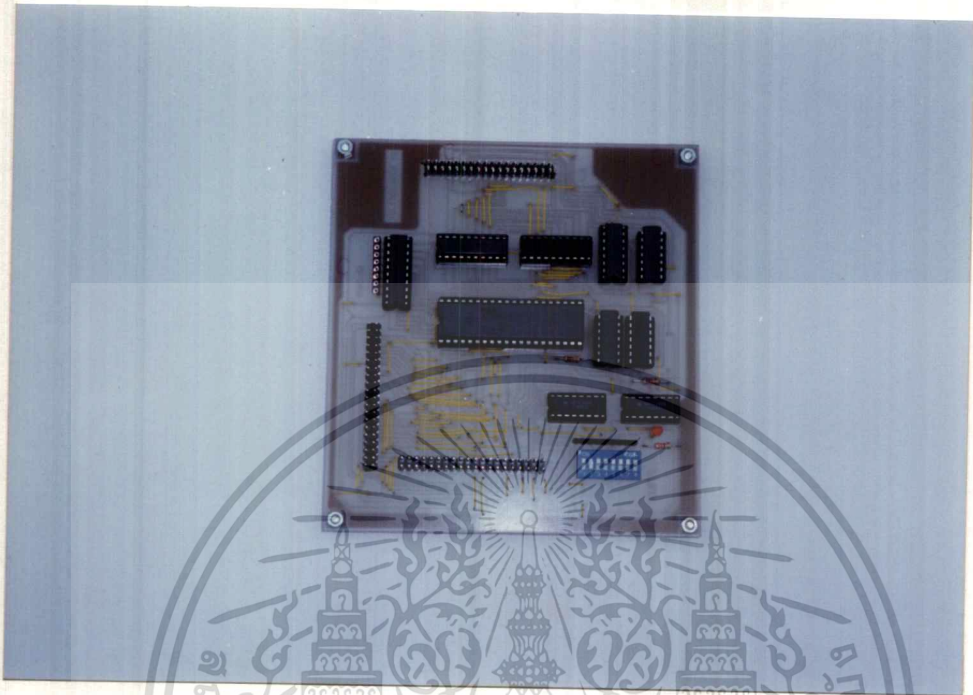


รูปที่ ก.1 ภาคจ่ายกำลังไฟฟ้า

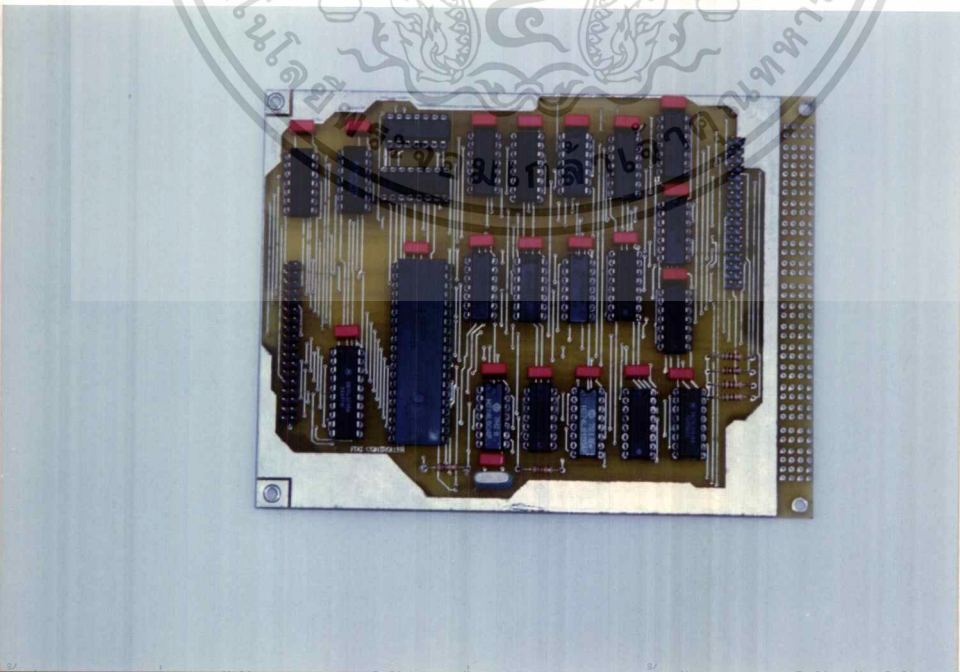


รูป ก.2 วงจรภาคควบคุมการโอนย้ายข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและตัดอำนาจของเอกสารทุกครั้งที่มีการนำไปใช้



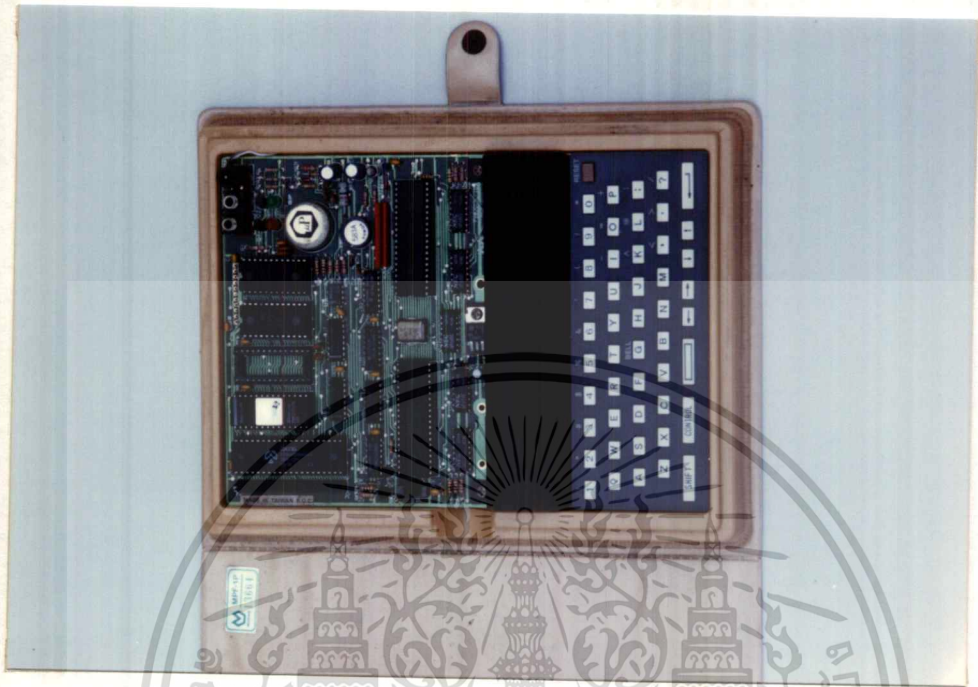
รูป ก.3 ภาขยายพอร์ต Z-80



รูปที่ ก.4 ภา FDC Controller

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

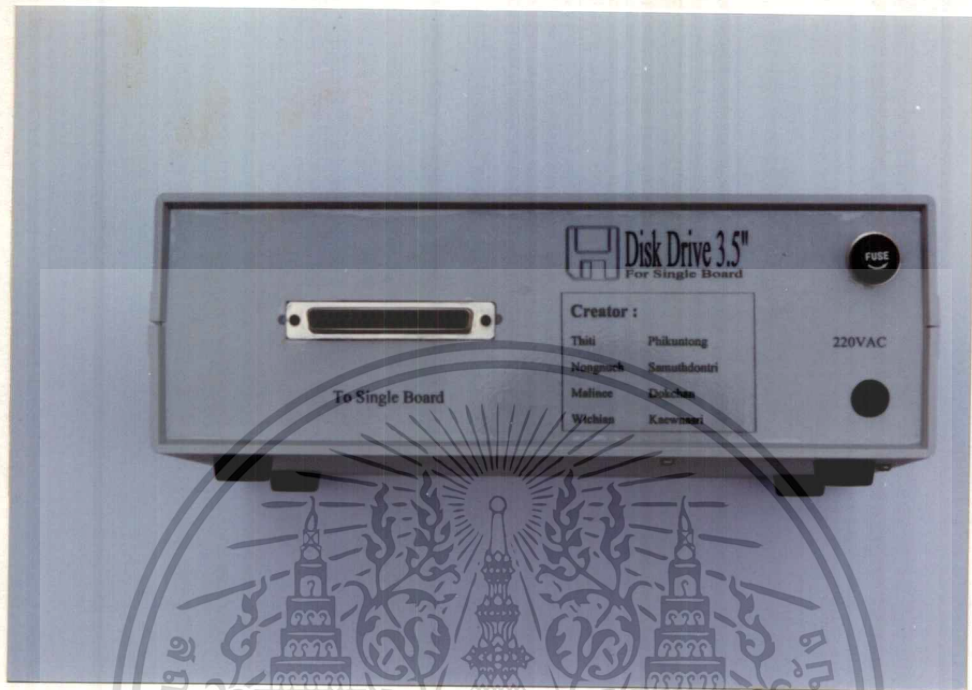


รูปที่ ก.5 ชิงเกิลบอร์ด MPF-1P



รูปที่ ก.6 กล้อง(ด้านหน้า)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



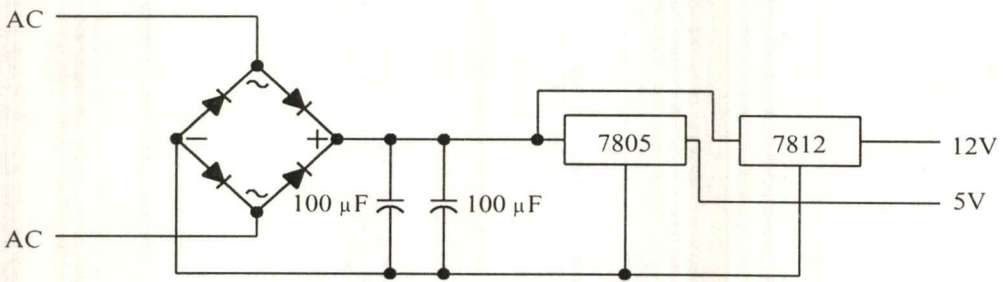
รูปที่ ก.7 กล่อง(ด้านหลัง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

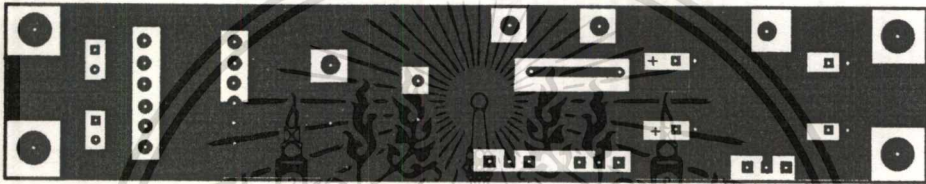


ภาคผนวก ข
วงจรรและแผ่นวงจรรพิมพ์

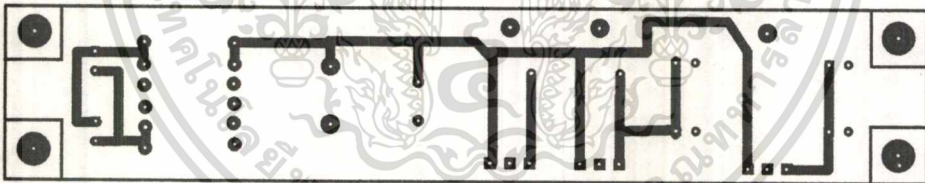
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



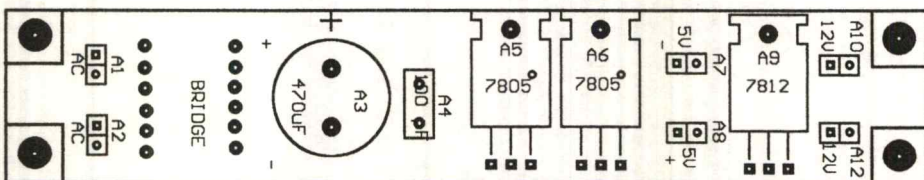
รูปที่ ข.1 วงจรภาคจ่ายกำลังไฟฟ้า



รูปที่ ข.2 ลายวงจรพิมพ์ภาคจ่ายกำลังไฟฟ้า (ด้านบน)

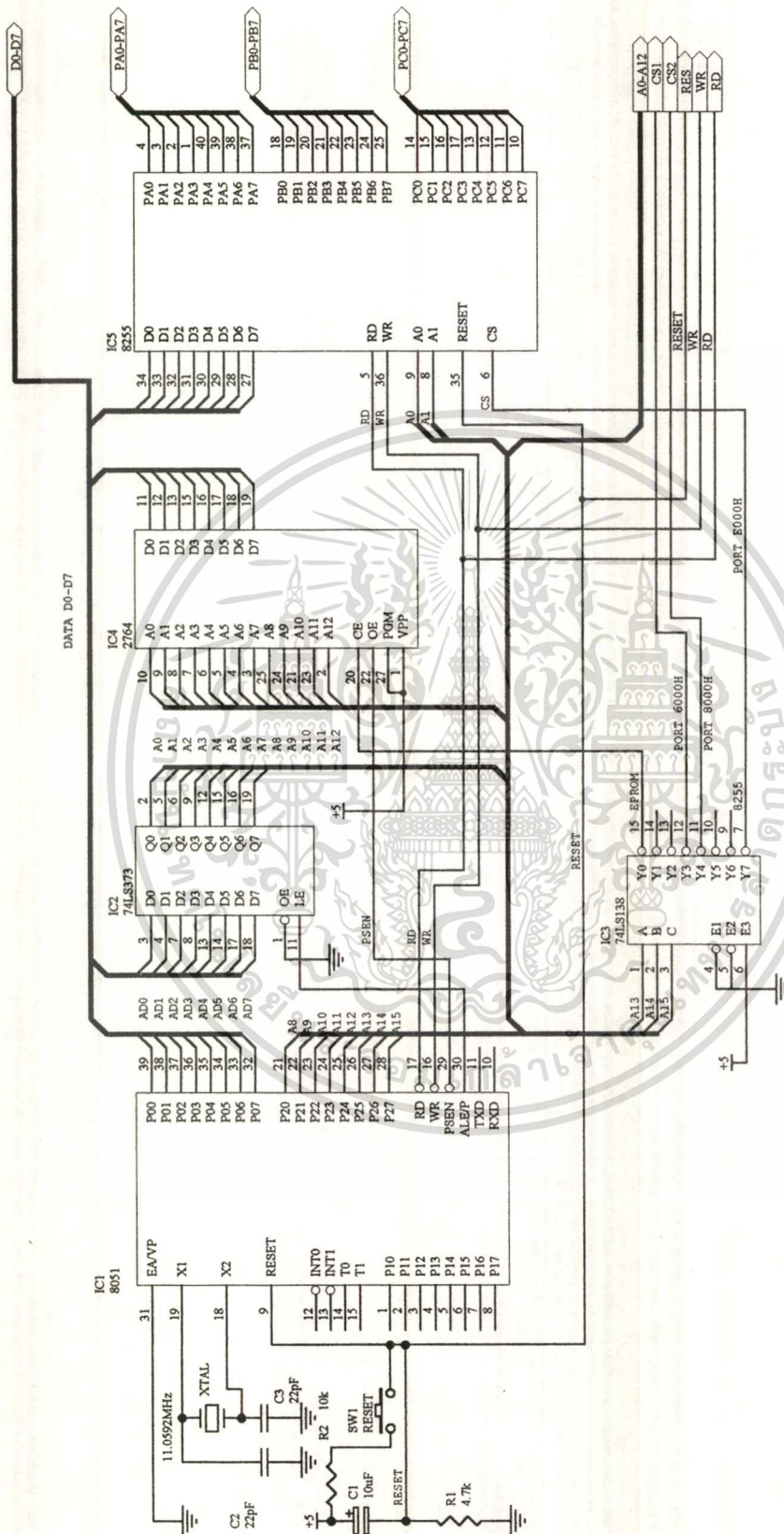


รูปที่ ข.3 ลายวงจรพิมพ์ภาคจ่ายกำลังไฟฟ้า (ด้านล่าง)

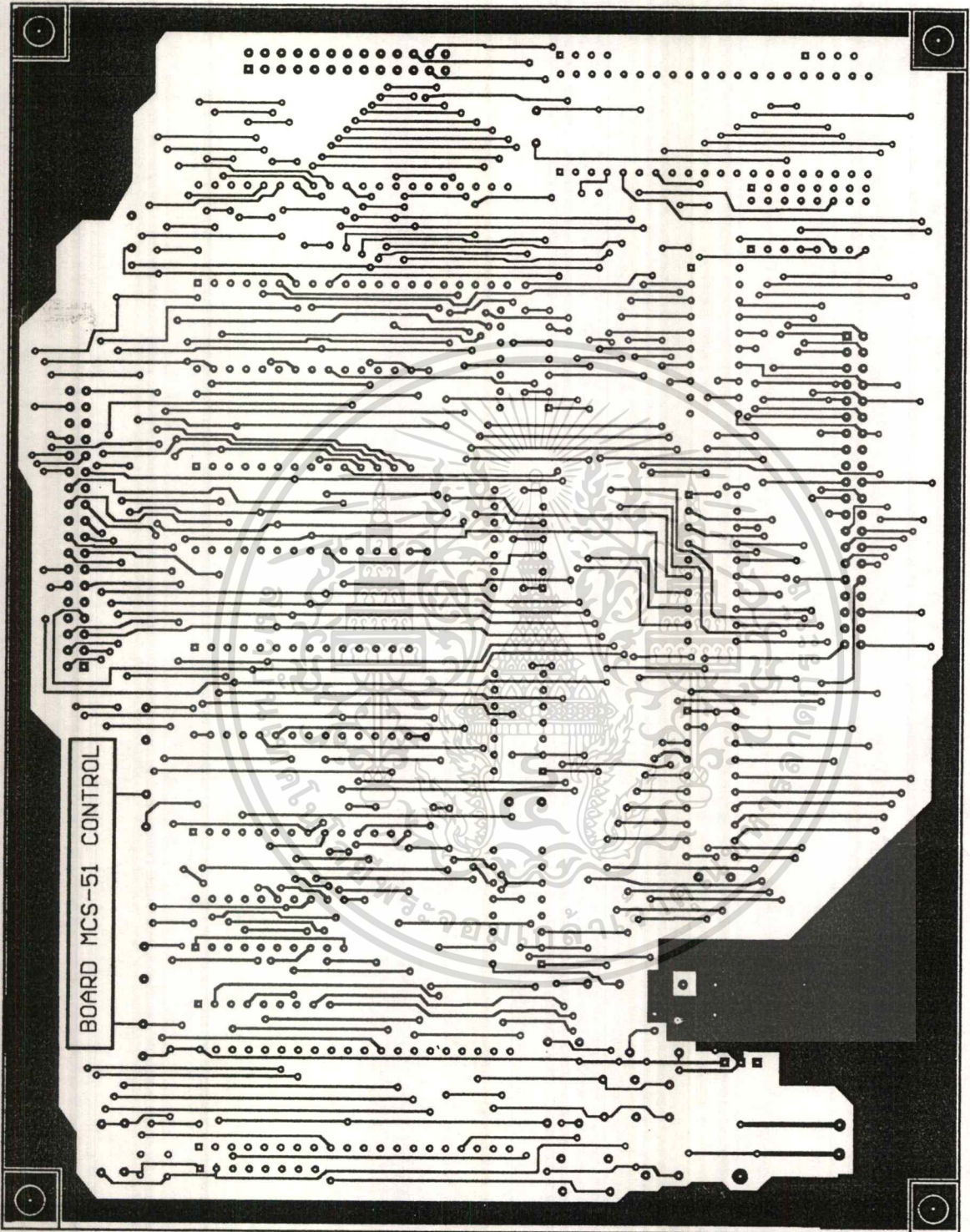


รูปที่ ข.4 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาคจ่ายกำลังไฟฟ้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

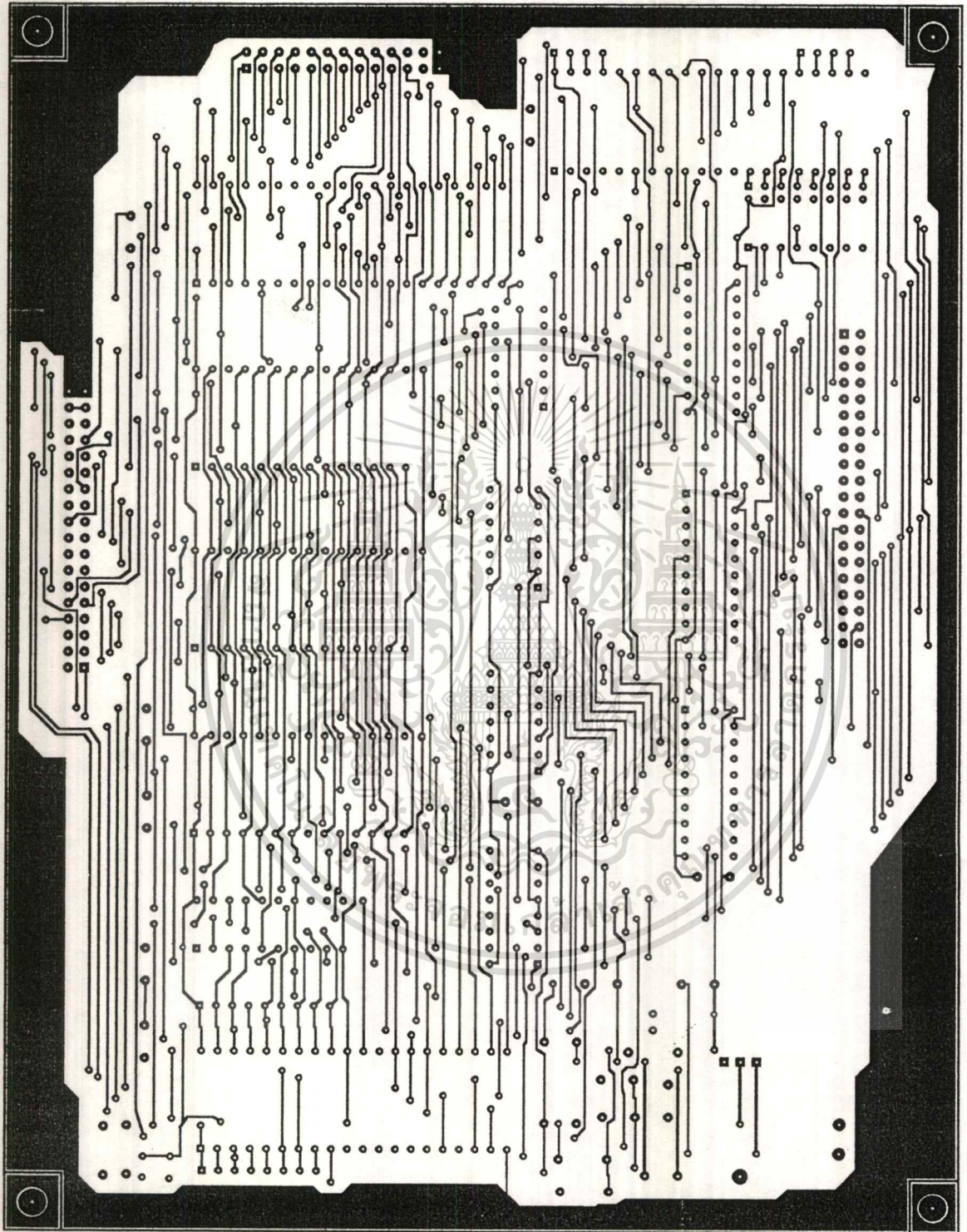


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งรูปที่ ๖.5 จรรยาบรรณควบคุมการโอนย้ายข้อมูลของเอกสารทุกครั้งที่มีการนำไปใช้



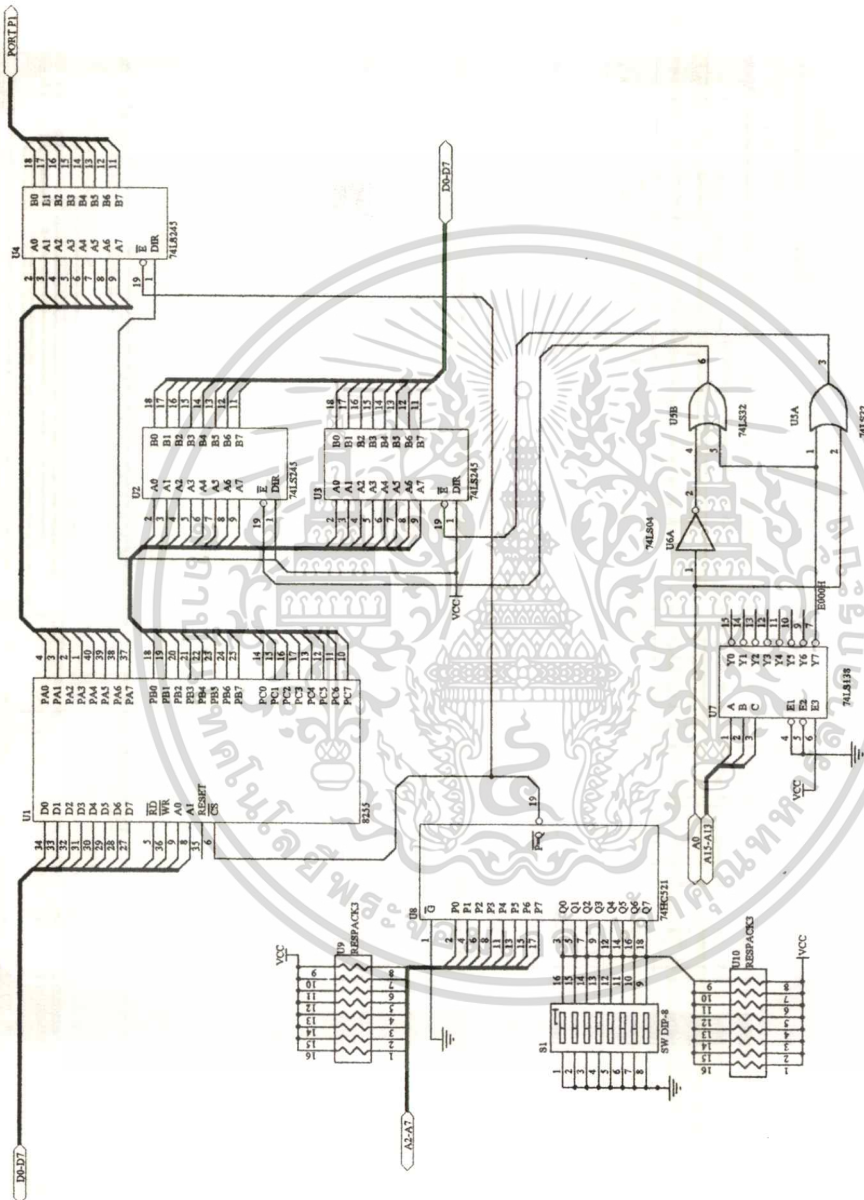
รูปที่ ข.6 ลายวงจรพิมพ์ภาคควบคุมการโอนย้ายข้อมูล (ด้านบน)

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับการใช้งานเฉพาะเท่านั้น ไม่ให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

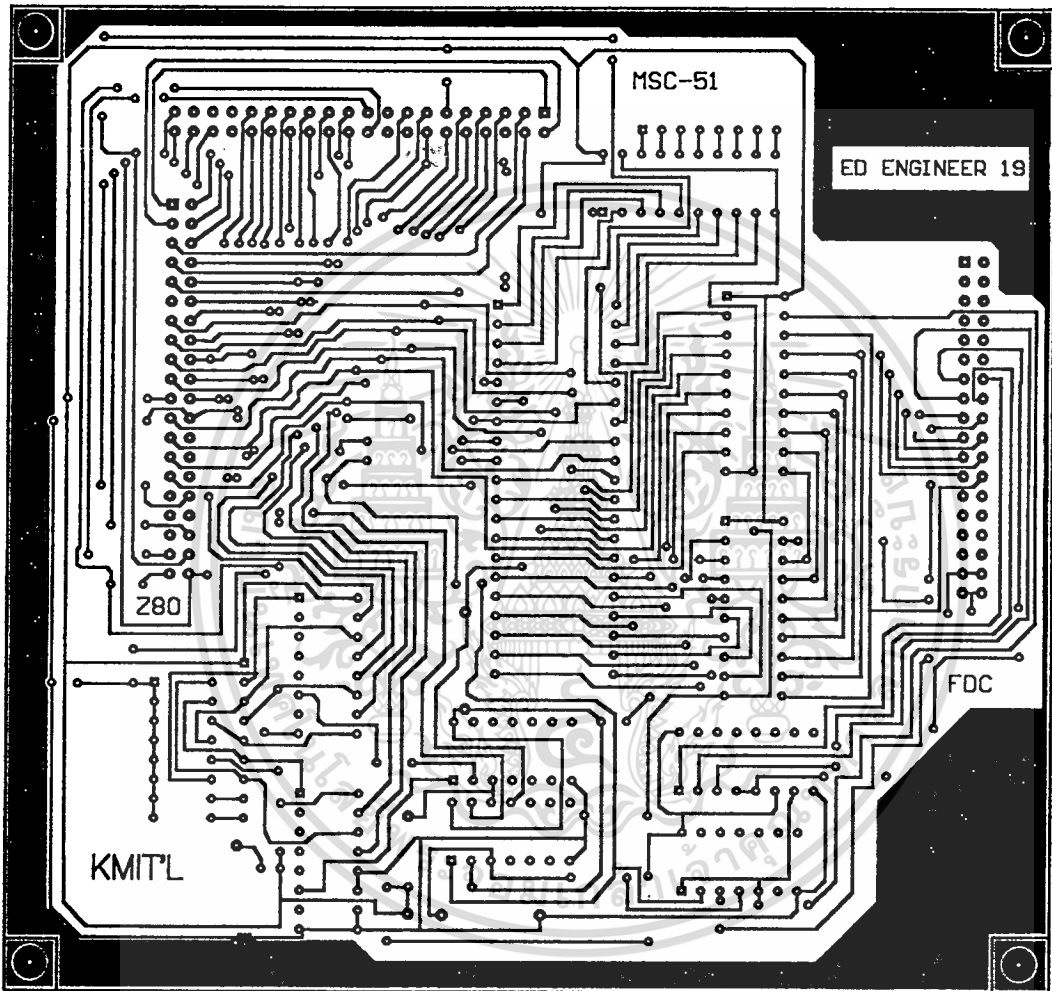


เอกสารนี้เป็นรูปที่ ๗.7 ลายวงจรพิมพ์ที่กล่าวถึงความคุ้มครองการโอนย้ายข้อมูล (ด้านล่าง) นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้**รูปที่ข.8** การวางอุปกรณ์บนแผงวงจรพิมพ์ภาคควบคุมการโอนย้ายข้อมูลไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

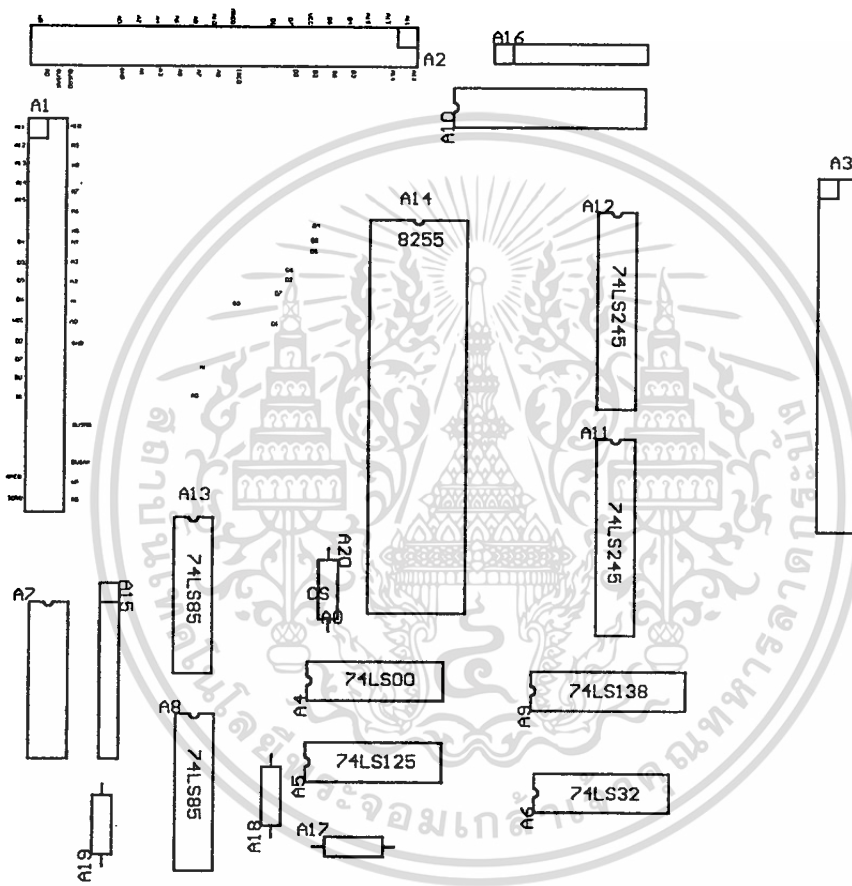


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้าม**รูปที่ ๗.9 วงจรภาคขยายพอร์ต Z-80**เจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



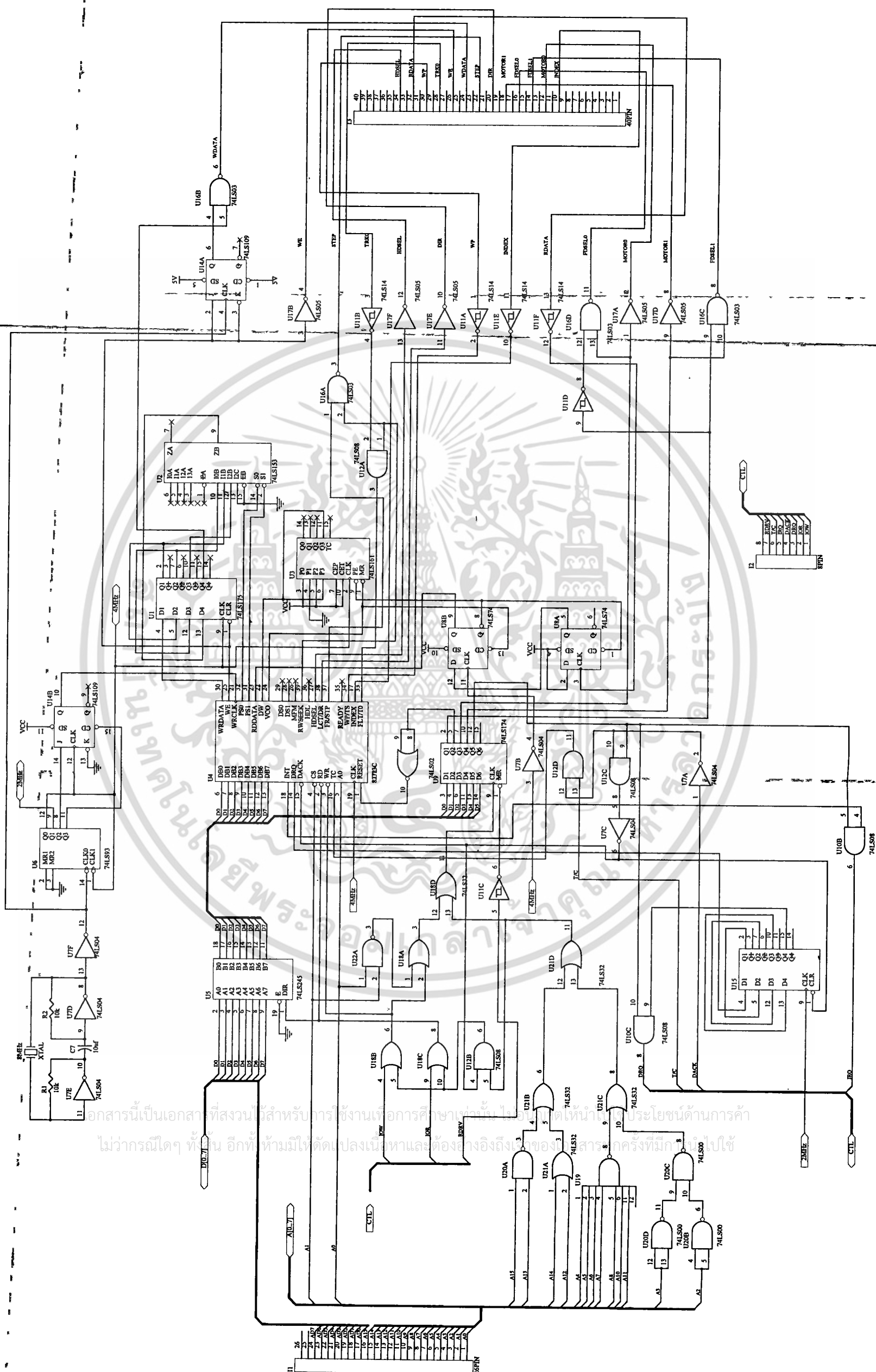
รูปที่ ข.10 ลายวงจรพิมพ์ภาคขยายพอร์ต Z-80

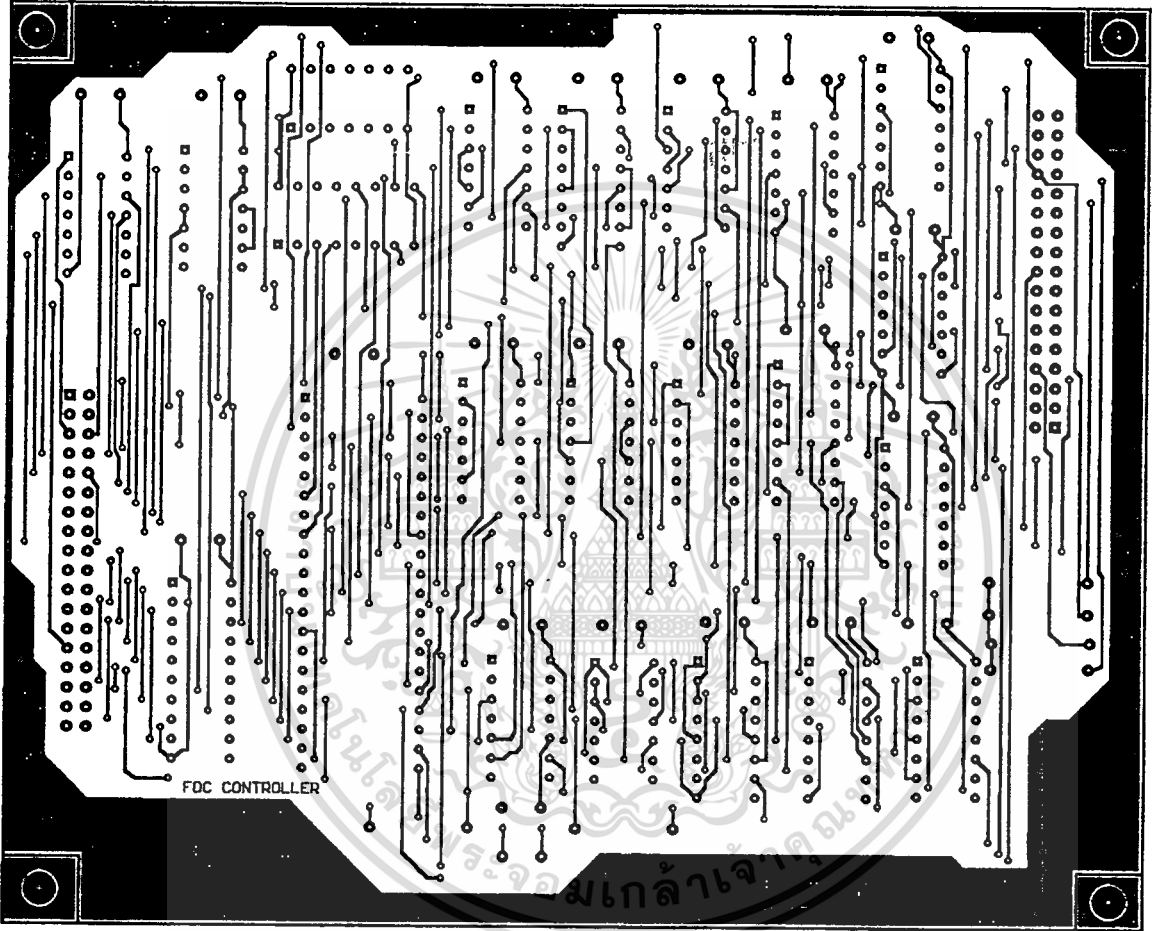
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.11 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาคขยายพอร์ต Z-80

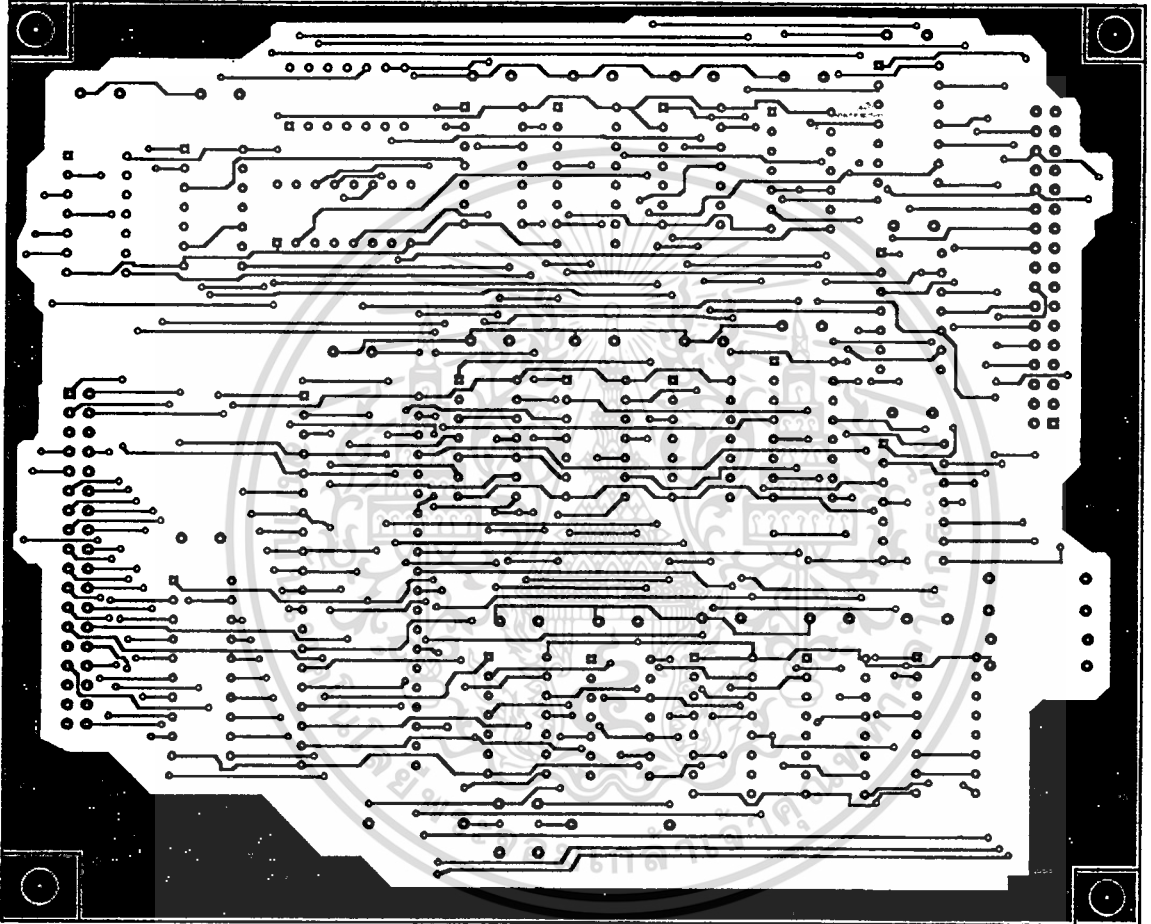
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้





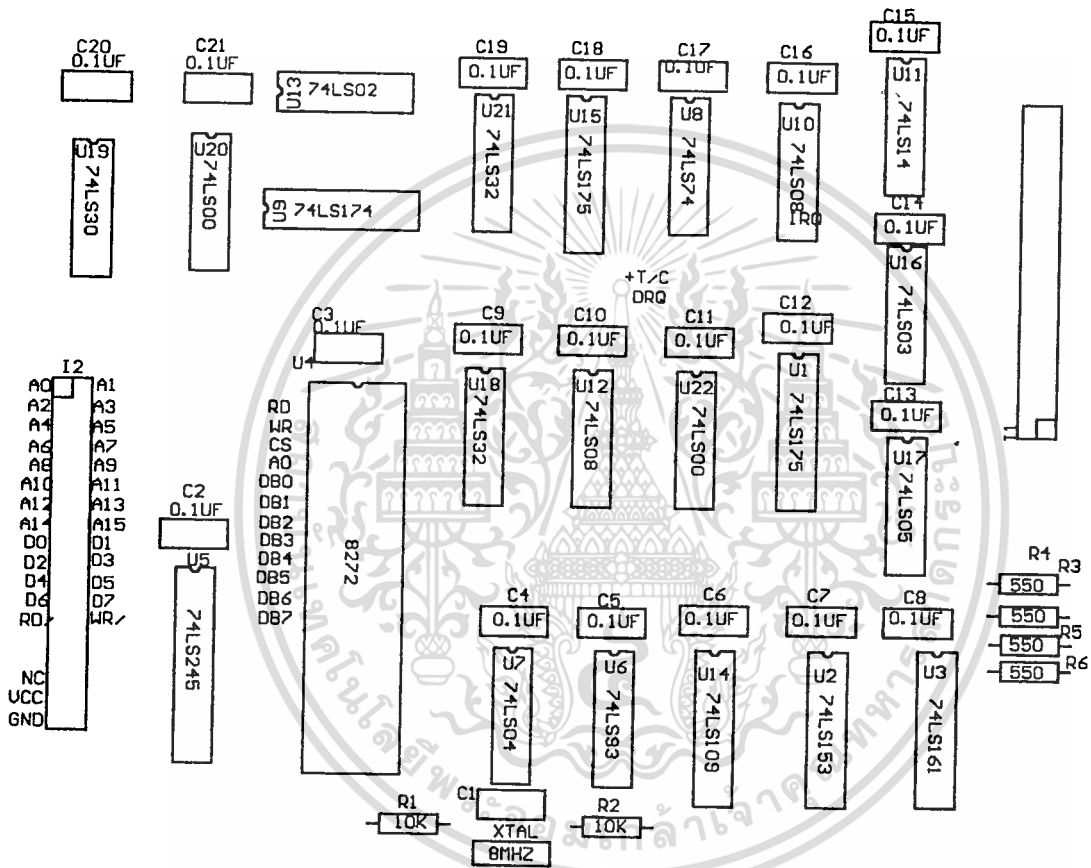
รูปที่ ข.13 ลายวงจรพิมพ์ภาค FDC Controller (ด้านบน)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข.14 ลายวงจรพิมพ์ภาค FDC Controller (ด้านล่าง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

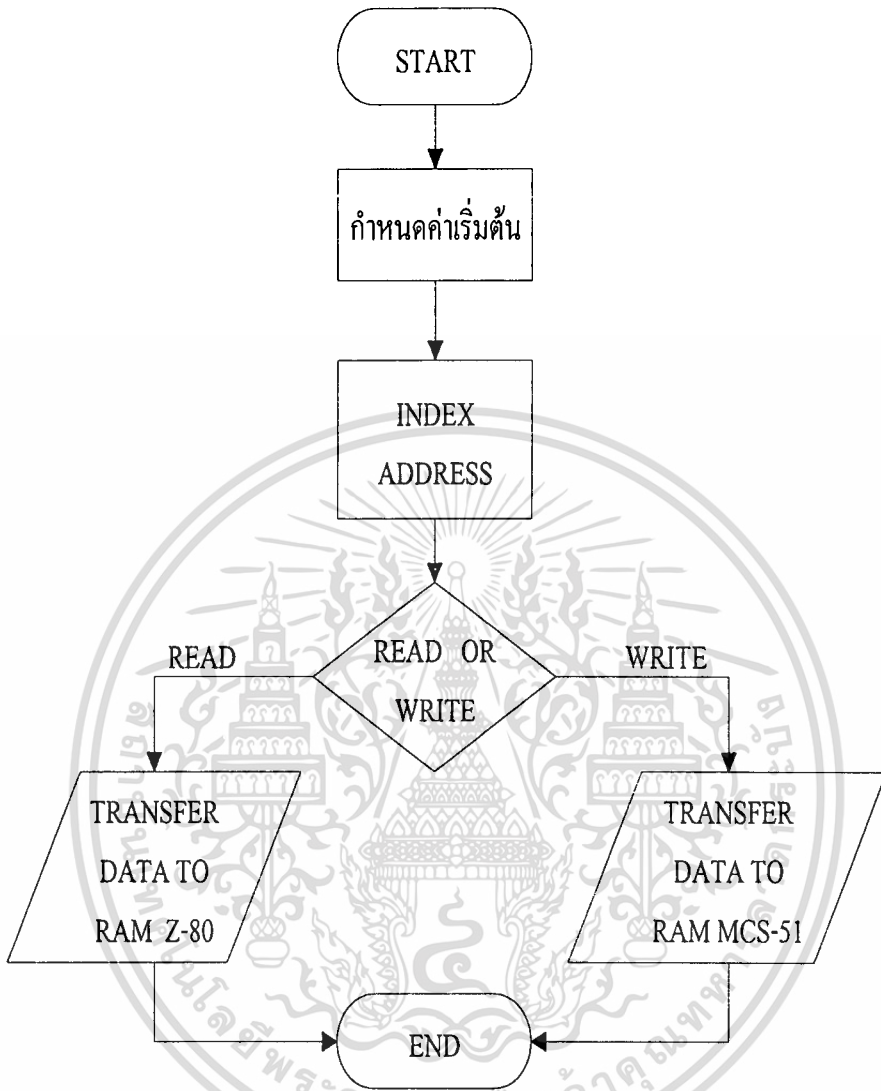


รูปที่ ข.15 การวางอุปกรณ์บนแผ่นวงจรพิมพ์ภาค FDC Controller



ภาคผนวก ค ผังการทำงานและโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ค.1 ฟังก์ชันของโปรแกรม

```

                ORG          0000H
                LJMP        INIT
PROG            EQU          0500H
MODEL          EQU          80H
CONTROL        EQU          0C003H
PORTA          EQU          0C000H
PORTB          EQU          0C001H
PORTC          EQU          0C002H
ADDWRITE51    EQU          6000H
ADDREAD51     EQU          7000H
ADDZ80HIGH    EQU          20H
ADD51HIGH     EQU          21H
ADDZ80LOW     EQU          22H
ADD51LOW      EQU          23H
ADD51HR       EQU          24H
ADD51LR       EQU          25H
ADDZ80HW      EQU          26H
ADDZ80LW      EQU          27H
BYTEL1        EQU          28H
BYTEL2        EQU          29H
BYTEH         EQU          30H
BYTEWRITE     EQU          32H
BYTEREAD      EQU          33H
BYTEREADHIGH  EQU          34H
BYTEWRITEHIGH EQU          35H
BYTEREADFILE  EQU          36H
PORTSET       EQU          P1.7
; Fdc Address
STATUSR       EQU          0AFF0H
DATAR         EQU          0AFF1H
OUTPTR        EQU          0AFF3H
; 8255 Address
P_A           EQU          0C000H
P_B           EQU          0C001H
P_C           EQU          0C002H
CNTR          EQU          0C003H
; RAM Address
RAM1          EQU          7000H
RAM2          EQU          8000H
; Memory MAP Variable
MEM           EQU          40H
HD            EQU          MEM
CYN           EQU          MEM+01H
H             EQU          MEM+02H
R             EQU          MEM+03H
N             EQU          MEM+04H
EOT           EQU          MEM+05H
GPL           EQU          MEM+06H
DTL           EQU          MEM+07H

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานในลักษณะที่ปรากฏเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีให้ดัดแปลงต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    ST0      EQU      MEM+08H
    ST1      EQU      MEM+09H
    ST2      EQU      MEM+0AH
    ST3      EQU      MEM+0BH
    INDEX    EQU      MEM+0CH
    CMD      EQU      MEM+0DH
    SRTHUT   EQU      MEM+0EH
    HLTDMA   EQU      MEM+0FH
                ORG      PROG

; Initial
INIT:;        MOV      A,#02H
;            MOV      DPTR,P_A
;            MOV      @DPTR,A
                LCALL   START_COMMAND
                MOV      A,#01H
                MOV      DPTR,#P_A
                MOVX     @DPTR,A
                LJMP     MAINPROGRAM
RETRN:        LCALL   START_COMMAND
                LCALL   INIT_MEM
                LCALL   RESET_FDC
                MOV      DPTR,#6004H
                MOVX     A,@DPTR
                MOV      CYN,A
                MOV      DPTR,#6005H
                MOVX     A,@DPTR
                MOV      H,A
                MOV      DPTR,#6006H
                MOVX     A,@DPTR
                MOV      R,A
                MOV      A,#03H
                MOV      DPTR,#P_A
                MOVX     @DPTR,A
                LCALL   WRITE_CMD
                SETB     PORTSET
                SETB     P1.6
LOOP_ERROR:   MOV      A,#08H
                MOV      DPTR,#P_A
                MOVX     @DPTR,A
                MOV      A,ST0
                ANL      A,#0C0H
                CJNE     A,#00H,LOOP_ERROR
                MOV      A,#01H
                MOV      DPTR,#P_A
                MOVX     @DPTR,A
                LJMP     $

```

Call Before Uses Program

; Procedure = 1. Start_Command

; 2. Write_CMD

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

; Defind Port A : Output

; Port B : Output

; Port C : Input,Output

MAINPROGRAM: SETB PORTSET

```

                SETB      P1.6
;-----INITIAL DATA ADDRESS-----
                MOV       ADD51HIGH,#60H
                MOV       ADD51LOW,#00H
                MOV       ADDZ80HIGH,#0F0H
                MOV       ADDZ80LOW,#10H
                MOV       ADD51HR,#70H
                MOV       ADD51LR,#00H
                MOV       ADDZ80HW,#0F8H
                MOV       ADDZ80LW,#00H
                MOV       BYTEREAD,#14H
                MOV       BYTEWRITE,#0FFH
                MOV       BYTEREADFILE,#0FH
KEYADDSTART:   MOV       P1,#0FFH
                SETB      P1.7
                LCALL     DELAY
                MOV       P1,#0CFH
                MOV       A,P1
                ANL       A,#0FH
                MOV       R0,A
;
;
;
                MOV       A,#01H
                MOV       DPTR,#P_A
                MOVB      @DPTR,A
                CJNE      R0,#01H,KEYREAD
                LCALL     DELAY
                LCALL     DELAY
                MOV       DPTR,#0E000H
                MOVB      A,@DPTR
                MOV       ADDZ80LOW,A
                LCALL     DELAY
                LCALL     DELAY
                MOV       DPTR,#0E001H
                MOVB      A,@DPTR
                MOV       ADDZ80HIGH,A
                LCALL     DELAY
                LCALL     DELAY
                LJMP      MAINWRITE
KEYREAD:       CJNE      R0,#02H,KEYADDSTART
                LCALL     DELAY
                LCALL     DELAY
                MOV       DPTR,#0E000H
                MOVB      A,@DPTR
                MOV       ADDZ80LOW,A
                LCALL     DELAY
                LCALL     DELAY
                MOV       DPTR,#0E001H
                MOVB      A,@DPTR
                MOV       ADDZ80HIGH,A
                LCALL     DELAY
                LCALL     DELAY
                MOV       DPTR,#0E001H
                MOVB      A,@DPTR
                MOV       ADDZ80HIGH,A
                LCALL     DELAY
                LCALL     DELAY
                LJMP      MAINREAD
;-----MAIN-----;

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์การใช้งานโดยไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังให้ตัดแปลงหรือทำซ้ำโดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MAINREAD:      MOV      A, #05H
                MOV      DPTR, #P_A
                MOVX     @DPTR, A
                LCALL    WRITETO51
                LCALL    DELAYTIME
                LJMP     TRNFERTOZ80
                LJMP     READFILE
MAINWRITE:     LCALL    WRITETOZ80
;              LCALL    DELAYTIME
                LJMP     TRNFERTO51
                LJMP     WRITEFILE
;-----WRITE-----;
WRITETOZ80:    CLR      P1.6
                LCALL    DELAY
                CLR      PORTSET
                LCALL    DELAY
                MOV      DPTR, #0F700H
                LCALL    WRITE
                LCALL    DELAY
                MOV      DPTR, #0F800H
                LCALL    WRITE
                RET
WRITE:         MOV      R3, BYTEWRITE
                MOV      R4, #02H
                MOV      14H, #01H
LOOPW:         MOV      A, 14H
                MOVX     @DPTR, A
                INC      DPL
                INC      14H
                DEC      R3
                CJNE     R3, #00H, LOOPW
                INC      DPL
                DJNZ     R4, LOOPW
                RET
WRITETO51:     SETB     PORTSET
                LCALL    DELAY
                MOV      DPTR, #ADDRREAD51
WRITE1:        MOV      R3, BYTEWRITE
                MOV      R4, #02H
                MOV      R5, #01H
LOOPW1:        MOV      A, R5
                MOV      A, #41H
                MOVX     @DPTR, A
                INC      DPL
                DEC      R3
                CJNE     R3, #00H, LOOPW1
                INC      DPL
                DJNZ     R4, LOOPW1
                LCALL    DELAY
                RET
;-----TRANSFER FOR FILE-----;
READFILE:     SETB     PORTSET

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ การใช้งานโดยไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น ถือว่าสงวนลิขสิทธิ์ไว้และต้องอ้างถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        LCALL    DELAY
        MOV      DPTR,#6000H
        MOVBX    A,@DPTR
        MOV      ADDZ80HW,A
        MOV      10H,A
        MOV      DPTR,#6001H
        MOVBX    A,@DPTR
        MOV      ADDZ80LW,A
        MOV      11H,A
;-----FINE BYTE FROM END ADDRESS-----
        MOV      DPTR,#6002H
        MOVBX    A,@DPTR
        SUBB     A,10H
LOOPREAD:  MOV      15H,A
        MOV      A,15H
        CJNE     A,#00H,LOAD
        MOV      DPTR,#6003H
        MOVBX    A,@DPTR
        SUBB     A,11H
        MOV      BYTEREAD,A
        MOV      R4,BYTEREAD
        MOV      DPH,ADDZ80HW
        MOV      DPL,ADDZ80LW
        PUSH     DPH
        PUSH     DPL
        MOV      DPH,ADD51HR
        MOV      DPL,ADD51LR
        MOVBX    A,@DPTR
        MOV      R7,A
        CLR      PORTSET
        LCALL    DELAY
        POP      DPL
        POP      DPH
        MOV      A,R7
        MOVBX    @DPTR,A
        SETB     PORTSET
        PUSH     DPH
        PUSH     DPL
LOOPRDFILE:  POP      DPL
        POP      DPH
        INC      DPL
        INC      ADD51LR
        PUSH     DPL
        MOV      DPH,ADD51HR
        MOV      DPL,ADD51LR
        MOVBX    A,@DPTR
        CLR      PORTSET
        LCALL    DELAY
        MOV      DPH,ADDZ80HW
        POP      DPL
        MOVBX    @DPTR,A
        SETB     PORTSET

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังให้ตัดแปลงเอกสารต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      R5,A
PUSH     DPH
PUSH     DPL
;
LCALL    DISPLAY
LCALL    DELAY
DJNZ     R4,LOOPRDFILE
MOV      A,#01H
MOV      DPTR,#P_A
MOVX     @DPTR,A
LJMP     END1
LJMP     KEYADDSTART
LOAD:    MOV      R4,#0FEH
MOV      DPH,ADDZ80HW
MOV      DPL,ADDZ80LW
PUSH     DPH
PUSH     DPL
MOV      DPH,ADD51HR
MOV      DPL,ADD51LR
MOVX     A,@DPTR
MOV      10H,A
CLR      PORTSET
LCALL    DELAY
POP      DPL
POP      DPH
MOV      A,10H
MOVX     @DPTR,A
SETB     PORTSET
PUSH     DPH
PUSH     DPL
LCHKFILE: POP      DPL
POP      DPH
INC      DPL
INC      ADD51LR
PUSH     DPL
MOV      DPH,ADD51HR
MOV      DPL,ADD51LR
MOVX     A,@DPTR
CLR      PORTSET
LCALL    DELAY
MOV      DPH,ADDZ80HW
POP      DPL
MOVX     @DPTR,A
SETB     PORTSET
MOV      R5,A
PUSH     DPH
PUSH     DPL
;
LCALL    DISPLAY
LCALL    DELAY
DJNZ     R4,LCHKFILE
MOV      DPH,ADD51HR
INC      DPH
MOV      ADD51HR,DPH

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MOV      ADD51LR, #00H
MOV      DPH, ADDZ80HW
INC      DPH
MOV      ADDZ80HW, DPH
MOV      ADDZ80LW, #00H
DEC      15H
LJMP     LOCPREAD
;-----WRITE FILE-----;
WRITEFILE: POP      DPL
           POP      DPH
           SETB     PORTSET
           MOV      DPTR, #6000H
           MOVX     A, @DPTR
           MOV      ADDZ80HW, A
           MOV      12H, A
           MOV      DPTR, #6001H
           MOVX     A, @DPTR
           MOV      ADDZ80LW, A
           MOV      13H, A
-----FINE BYTE FROM END ADDRESS-----
           MOV      DPTR, #6002H
           MOVX     A, @DPTR
           SUBB     A, 12H
           MOV      16H, A
WRITELOOP: MOV      A, 16H
           CJNE     A, #00H, LOOP2
           LJMP     LOAD1
LOOP2:    MOV      A, #0FFH
           SJMP     LOAD3
LOAD1:    MOV      DPTR, #6003H
           MOVX     A, @DPTR
           SUBB     A, 13H
LOAD3:    MOV      BYTEREAD, A
           MOV      R4, BYTEREAD
           MOV      DPH, ADD51HR
           MOV      DPL, ADD51LR           ; 00H
           PUSH     DPH
           PUSH     DPL
           CLR      PORTSET
           LCALL     DELAY
           MOV      DPH, ADDZ80HW
           MOV      DPL, ADDZ80LW
           MOVX     A, @DPTR
           MOV      R7, A
           SETB     PORTSET
           POP      DPL
           POP      DPH
           MOV      A, R7
           MOVX     A, @DPTR, A
           CLR      PORTSET
           LCALL     DELAY
           PUSH     DPH

```

```

LOOPWDFILE:  PUSH      DPL
              POP       DPL
              POP       DPH
              INC       DPL
              INC       ADDZ80LW
              PUSH      DPL
              MOV       DPH,ADDZ80HW
              MOV       DPL,ADDZ80LW
              MOVX      A,@DPTR
              SETB      PORTSET
              MOV       DPH,ADD51HR
              POP       DPL
              MOVX      @DPTR,A
              MOVX      A,@DPTR
              MOV       R5,A
              PUSH      DPH
              PUSH      DPL
              LCALL     DELAY
              LCALL     DELAY
              CLR       PORTSET
              LCALL     DELAY
              DJNZ      R4,LOOPWDFILE
              MOV       A,16H
              POP       DPL
              POP       DPH
              CJNE      A,#00H,LOOPP
              LJMP      RETRN
LOOPP:       MOV       DPH,ADD51HR
              INC       DPH
              MOV       ADD51HR,DPH
              MOV       ADD51LR,#00H
              MOV       DPH,ADDZ80HW
              INC       DPH
              MOV       ADDZ80HW,DPH
              MOV       ADDZ80LW,#00H
              DEC       16H
              LJMP      WRITELOOP
;-----TRANSFER DATA RAM TO MCS-51-----
TRANFERTO51: CLR       P1.6
              LCALL     DELAY
              CLR       PORTSET
              LCALL     DELAY
              MOV       R4,BYTEREAD
              DEC       R4
              MOV       DPH,ADD51HIGH
              MOV       DPL,ADD51LOW
              PUSH      DPH
              PUSH      DPL
              MOV       DPH,ADDZ80HIGH
              MOV       DPL,ADDZ80LOW
              MOVX      A,@DPTR
              MOV       0H,A

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังให้ตัดแปลงแก้ไขได้อย่างอิสระถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SETB      P1.6
SETB      PORTSET
LCALL     DELAY
POP       DPL
POP       DPH
MOV       A,10H
MOVX      @DPTR,A
CLR       P1.6
CLR       PORTSET
LCALL     DELAY
PUSH      DPH
PUSH      DPL
LOOPRR51: POP      DPL
POP       DPH
INC       DPL
INC       ADDZ80LOW
PUSH      DPL
MOV       DPH,ADDZ80HIGH
MOV       DPL,ADDZ80LOW
MOVX      A,@DPTR
SETB      PORTSET
MOV       DPH,ADD51HIGH
POP       DPL
MOVX      @DPTR,A
MOVX      A,@DPTR
MOV       R5,A
PUSH      DPH
PUSH      DPL
CLR       PORTSET
LCALL     DELAY
DJNZ      R4,LOOPRR51
PUSH      DPH
PUSH      DPL
LJMP      WRITEFILE
;-----TRANSFER DATA RAM TO MCS-51-----
TRANSFERTOZ80: CLR      P1.6
CLR       PORTSET
LCALL     DELAY
MOV       ADD51HR,#70H
MOV       ADD51LR,#00H
MOV       R4,BYTEREAD
DEC       R4
MOV       DPH,ADD51HIGH
MOV       DPL,ADD51LOW
PUSH      DPH
PUSH      DPL
MOV       DPH,ADDZ80HIGH
MOV       DPL,ADDZ80LOW
MOVX      A,@DPTR
MOV       A,10H,A
SETB      P1.6
SETB      PORTSET

```

```

                LCALL    DELAY
                POP      DPL
                POP      DPH
                MOV      A,10H
                MOVX     @DPTR,A
                CLR      P1.6
                CLR      PORTSET
                LCALL    DELAY
                PUSH     DPH
                PUSH     DPL
LOOPRRT051:    POP      DPL
                POP      DPH
                INC      DPL
                INC      ADDZ80LOW
                PUSH     DPL
                MOV      DPH,ADDZ80HIGH
                MOV      DPL,ADDZ80LOW
                MOVX     A,@DPTR
                SETB     P1.6
                SETB     PORTSET
                MOV      DPH,ADD51HIGH
                POP      DPL
                MOVX     @DPTR,A
                MOVX     A,@DPTR
                MOV      R5,A
                PUSH     DPH
                PUSH     DPL
                CLR      P1.6
                CLR      PORTSET
                LCALL    DELAY
                DJNZ     R4,LOOPRRT051
                LJMP     READFILE

; Command Zone
; Start Comand to Set FDD
START_COMMAND: LCALL    INIT8255
                LCALL    INIT_MEM
                LCALL    RESET_FDC
                LCALL    SPECIFY_CMD
                LCALL    RECALIBRATE_CMD
                LCALL    LONG_DELAY
                RET

; Specify Command
SPECIFY_CMD:   MOV      INDEX,#03H
                MOV      CMD,#00000011B
                MOV      SRTHUT,#11010000B
                MOV      HLTDMA, #00000001B
                MOV      DPTR,#SPECIFY
                LCALL    COMMAND
                LCALL    FDC_PROCESS
                LCALL    SLEEP
                RET

; Recalibrate

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังห้ามให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

RECALIBRATE_CMD:
; If EC Set Recalibrate again
RE_AGIAN:      MOV      INDEX, #02H
               MOV      CMD,#00000111B
               MOV      HD,#00000001B
               MOV      DPTR,#RECALIBRATE
               LCALL    ON_DRIVE
               LCALL    COMMAND
               LCALL    FDC_PROCESS
               LCALL    OFF_DRIVE
               LCALL    INT_STATUS_CMD
               MOV      A,ST0
               ANL      A,#10H
               CJNE     A,#10H,END_RECA
               LCALL    RESET_FDC
               LCALL    SPECIFY_CMD
               SJMP     RE_AGIAN
END_RECA:      LCALL    SLEEP
               RET

; Sense Interrupt Status
INT_STATUS_CMD:
; Check Seek Signal Busy Before Sense Command
               MOV      R5,#0AH
TRY_CHECK_INT: MOV      DPTR,#STATUSR
               MOVX     A,@DPTR
               ANL      A,#02H
               CJNE     A,#02H,CHECK_TRY_NOW
               SJMP     INT_STATUS_TRY
CHECK_TRY_NOW: DJNZ     R5,TRY_CHECK_INT
               SJMP     END_INT_CMD
INT_STATUS_TRY: MOV      INDEX,#01H
               MOV      CMD,#00001000B
               MOV      DPTR,#INT_STATUS_CM
               LCALL    COMMAND
               LCALL    FDC_PROCESS
               MOV      INDEX,#02H
               MOV      DPTR,#INT_STATUS_RE
               LCALL    RESULT
               MOV      DPTR,#STATUSR
               MOVX     A,@DPTR
               ANL      A,#02H
               CJNE     A,#00H,INT_STATUS_TRY
END_INT_CMD:   LCALL    SLEEP
               RET

SEEK_CMD:
; Head & Drive, Cylinder Define Before USES
               MOV      INDEX,#03H
               MOV      CMD,#00001111B
               MOV      DPTR,#SEEK
               LCALL    COMMAND
               LCALL    ON_DRIVE
               LCALL    FDC_PROCESS

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังห้ามนำไปคัดลอกหรือเผยแพร่โดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

                LCALL    OFF_DRIVE
                LCALL    INT_STATUS_CMD
                LCALL    SLEEP
                RET

; Write
; HD,CYN,H,R,DTL Define Before Uses
WRITE_CMD:      MOV      A,H
                RL        A
                RL        A
                ORL       A,#01H
                MOV       HD,A
                LCALL    SEEK_CMD
                LCALL    SLEEP
                LCALL    TC_C
                MOV       INDEX,#09H
                MOV       CMD,#01000101B
                MOV       DPTR,#WRITE_CM
                LCALL    ON_DRIVE
                LCALL    COMMAND
                LCALL    TRANSFER_DATA
                LCALL    FDC_PROCESS
                LCALL    OFF_DRIVE
                MOV       INDEX,#07H
                MOV       DPTR,#WRITE_RE
                LCALL    RESULT
                RET

; SUB Program
; Data Transfer From Write Command
TRANSFER_DATA:  LCALL    SLEEP
; Load DTL to R4 For Check Data Length
                MOV       R4,DTL
                MOV       R5,#00H

LOOP_TRANSFER:
; Check EXM BIT From FDC
INT_CHECK_WR:   MOV       DPTR,#STATUSR
                MOVX      A,@DPTR
                ANL       A,#20H
                CJNE      A,#20H,INT_CHECK_WR

LOOP_SEND:      MOV       DPTR,#RAM1
                MOV       A,DPL
                ADD        A,R5
                MOV       DPL,A
                MOVX      A,@DPTR
                MOV       DPTR,#DATAR
                MOVX      @DPTR,A
                INC        R5
                DJNZ       R4,LOOP_SEND
                RET

; Command Operation
; Const DPTR to Command table
COMMAND:        MOV       R6,#0H
                MOV       A,R6

```

เอกสารนี้เป็นเอกสารสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

; Const DPTR to Command table
COMMAND: MOV R6,#0H และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MOV A,R6

```

                MOVC      A,@A+DPTR
                MOV       R0,A
                MOV       A,@R0
                MOV       R7,A

; Loop Start
LOOP_CMD:      LCALL     FDC_BUSYW
                INC      R6
                MOV      A,R6
                MOVC     A,@A+DPTR
                MOV      R0,A
                MOV      A,@R0
                PUSH     DPH
                PUSH     DPL
                MOV      DPTR,#DATAR
                MOVX     @DPTR,A
                POP      DPL
                POP      DPH
                DJNZ     R7,LOOP_CMD
;
                LCALL     FDC_PROCESS
                RET

; Get Result Operation
RESULT:        MOV      R6,#0H
                MOV      A,R6
                MOVC     A,@A+DPTR
                MOV      R0,A
                MOV      A,@R0
                MOV      R7,A

; Loop Start
LOOP_RET:      LCALL     FDC_BUSYR
                INC      R6
                MOV      A,R6

; insert Indication
                PUSH     DPH
                PUSH     DPL
                MOV      DPTR,#P_B
                MOVX     @DPTR,A
                POP      DPL
                POP      DPH
                MOVC     A,@A+DPTR
                MOV      R0,A
                MOV      R0,A
                PUSH     DPH
                PUSH     DPL
                MOV      DPTR,#DATAR
                MOVX     A,@DPTR
                POP      DPL
                POP      DPH
                MOV      @R0,A
                DJNZ     R7,LOOP_RET
                RET

; On,Off Motor And Select Drive B
ON_DRIVE:      PUSH     DPH

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์การใช้งานที่ R7, LOOP_RET นั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น ผู้ใช้ต้องรับผิดชอบต่อเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        PUSH        DPL
        MOV         DPTR,#OUTPUTR
        MOV         A,#00100101B
        MOVX        @DPTR, A
        POP         DPL
        POP         DPH
        RET
OFF_DRIVE:
        PUSH        DPH
        PUSH        DPL
        MOV         DPTR,#OUTPUTR
        MOV         A,#00000101B
        MOVX        @DPTR, A
        POP         DPL
        POP         DPH
        RET
; Fdc Process
FDC_PROCESS:
        PUSH        DPH
        PUSH        DPL
        PUSH        07H
        PUSH        06H
LOOP_STATUS_PROCESS:
        MOV         R7,#0AFH
LOOP_BUSYP:
        MOV         R6,#0FFH
        DJNZ        R6,$
        DJNZ        R7,LOOP_BUSYP
;LOOP_STATUS_PROCESS:
        MOV         DPTR,#STATUSR
        MOVX        A,@DPTR
        ANL         A,#10100000B
        CJNE        A,#80H,LOOP_STATUS_PROCESS
        POP         06H
        POP         07H
        POP         DPL
        POP         DPH
        RET
; Fdc Busy SUB
; Check Ready Write Data
FDC_BUSYW:
        PUSH        DPH
        PUSH        DPL
        PUSH        07H
        PUSH        06H
        MOV         R7,#0AFH
LOOP_BUSYW:
        MOV         R6,#0FFH
        DJNZ        R6,$
        DJNZ        R7,LOOP_BUSYW
        MOV         DPTR,#STATUSR
LOOP_STATUS_WRITE:
        MOVX        A,@DPTR
        ANL         A,#0C0H
        CJNE        A,#80H,LOOP_STATUS_WRITE
        POP         06H
        POP         07H

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้ภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีให้ดัดแปลงแก้ไขและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        POP          DPL
        POP          DPH
        RET

; Check Ready Read Data
FDC_BUSYR:  PUSH     DPH
            PUSH     DPL
            PUSH     07H
            PUSH     06H
            MOV      R7,#0AFH
LOOP_BUSYR:  MOV      R6,#0FFH
            DJNZ     R6,$
            DJNZ     R7,LOOP_BUSYR

LOOP_STATUS_READ:
            MOV      DPTR,#STATUSR
            MOVX     A,@DPTR
            ANL      A,#0C0H
            CJNE     A,#0C0H,LOOP_STATUS_READ
            POP      06H
            POP      07H
            POP      DPL
            POP      DPH
            RET

; Terminal Counter
; Terminal
TC_T:       MOV      A,#01001111B
            MOV      DPTR,#P_C
            MOVX     @DPTR,A
            LCALL    SLEEP
            MOV      A,#10001111B
            MOV      DPTR,#P_C
            MOVX     @DPTR,A
            RET

; Counter
TC_C:       MOV      A,#10001111B
            MOV      DPTR,#P_C
            MOVX     @DPTR,A
            RET

; Long Delay Time
LONG_DELAY: MOV      R7,#0FFH
INNER_LONG_DELAY:
            MOV      R6,#0FFH
            DJNZ     R6,$
            DJNZ     R7,INNER_LONG_DELAY
            RET

; Sleep When Finish Command
SLEEP:      MOV      R7,#0AFH
INNER_SLEEP: MOV     R6,#0FFH
            DJNZ     R6,$
            DJNZ     R7,INNER_SLEEP
            RET

; Initial Output Register
RESET_FDC:  PUSH     DPH

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่อนุญาตให้ผู้อื่นนำข้อมูลไปเผยแพร่หรือดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PUSH      DPL
MOV       A,#080H
MOV       P1,A
MOV       A,#00000000B
MOV       DPTR,#OUTPUTR
MOVBX     @DPTR, A
MOV       A,#00000101B
MOVBX     @DPTR, A
POP       DPL
POP       DPH
RET

```

; Initial Define Data Starting

```

INIT_MEM:  MOV      HD,#01H
           MOV      CYN,#00H
           MOV      H,#00H
           MOV      R,#00H
           MOV      N,#02H
           MOV      EOT,#12H
           MOV      GPL,#6CH
           MOV      DTL,#0AH
           MOV      ST0,#00H
           MOV      ST1,#00H
           MOV      ST2,#00H
           MOV      ST3,#00H
           RET

```

; Initial 8255 to Set Port A,B,C to O O I,O

```

INIT8255:  MOV      A,#81H
           MOV      DPTR,#CNTR
           MOVBX     @DPTR, A
           MOV      A,#10001111B
           MOV      DPTR,#P_C
           MOVBX     @DPTR, A
           RET

```

;Command Code

```

SPECIFY:   DB      INDEX,CMD,SRTHUT,HLTDMA
RECALIBRATE: DB      INDEX,CMD,HD
SEEK:      DB      INDEX,CMD,HD,CYN
WRITE_CM:  DB      INDEX,CMD,HD,CYN,H,R,N,EOT,GPL,DTL
WRITE_RE:  DB      INDEX,ST0,ST1,ST2,CYN,H,R,N
INT_STATUS_CM: DB      INDEX,CMD
INT_STATUS_RE: DB      INDEX,ST0,CYN

```

; End Program File

```

DISPLAY:   MOV      A,R5
           MOV      DPTR,#PORTA
           MOVBX     @DPTR,A
           RET

```

```

DELAY: ;    MOV      R1,#28H

```

```

           MOV      R1,#35H

```

```

DELAY1:    MOV      R2,#0FFH

```

```

           DJNZ     R2,$

```

```

           DJNZ     R1,DELAY1

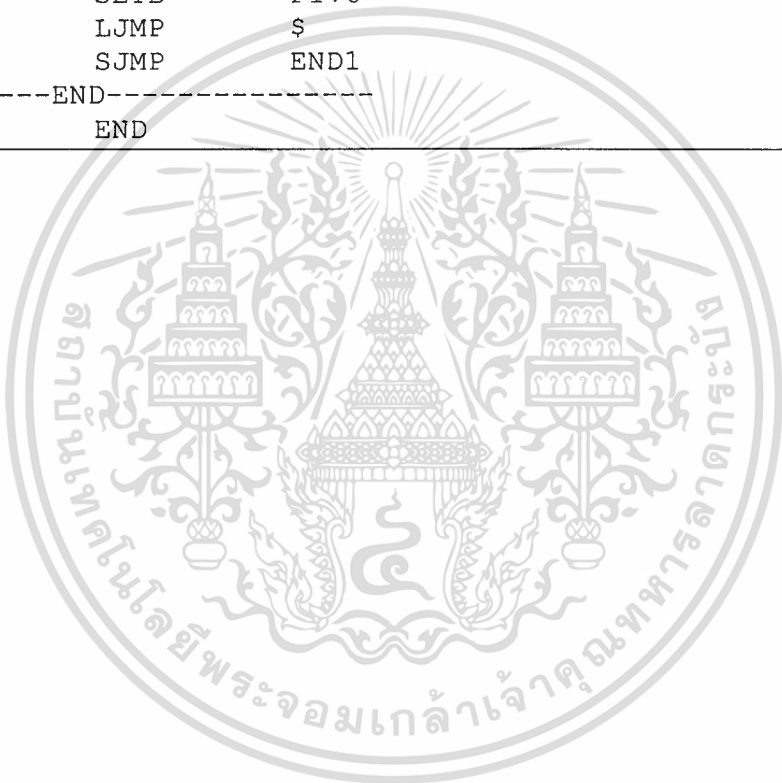
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้ภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่าในรูปแบบใดๆ ทั้งสิ้น อีกทั้งยังมิให้คัดลอกหรือเผยแพร่ข้อมูลนี้และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

                                RET
DELAYTIME:    MOV             R1,#25H
DELAYTIME1:   MOV             R2,#0FFH
                                DJNZ          R2,$
                                DJNZ          R1,DELAYTIME1
                                RET
END1:         MOV             CYN,#01H
                                MOV             H,#01H
                                MOV             R,#10H
                                LCALL          WRITE_CMD
                                SETB          PORTSET
                                SETB          P1.6
                                LJMP          $
;                                SJMP          END1
;-----END-----
                                END

```



ภาคผนวก ง
รายละเอียดและคุณสมบัติของอุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายการอุปกรณ์

วงจรภาค FDC (Floppy Disk Controller)

Comment	Pattern	Quality	Component
	IDC34	1	I1
	IDC36	1	I2
	RAD0.2	1	C1
0.1 µF	RAD0.2	20	C2, C3, C4, C5, C6 ,C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19. C20, C21
100pF	RAD0.2	1	A4
10K	AXIAL0.5	2	R1, R2
12V	SIP2	1	A12
12V	SIP2	1	A10
470µF	RB.3/6	1	A3
550	AXIAL0.5	4	R3, R4, R5, R6
5V	SIP2	2	A7, A8
74LS00	DIP14	2	U20, U22
74LS02	DIP14	1	U13
74LS03	DIP14	1	U16
74LS04	DIP14	1	U7
74LS05	DIP14	1	U17
74LS08	DIP14	2	U10, U12
74LS14	DIP14	1	U11
74LS30	DIP20	1	U19
74LS32	DIP14	2	U18, U21
74LS74	DIP14	1	U8
74LS93	DIP14	1	U6
74LS109	DIP16	1	U14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
โดยไม่ได้รับอนุญาตใดๆ ทั้งสิ้น และห้ามมิให้ดัดแปลงเนื้อหาและข้อมูลอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

74LS153	DIP16	1	U2
74LS161	DIP16	1	U3
74LS175	DIP16	2	U1, U15
74LS245	DIP20	1	U5
7805	TO-220	2	A5, A6
7812	TO-222	1	A9
8272	DIP40	1	U4
AC	SIP2	2	A1, A2
XTAL	XTAL1	1	8MHz

วงจรภาคโอนย้ายข้อมูล

Comment	Pattern	Quality	Component
	IDC36	1	FDD
	SIP14	2	A1, S3
0.1F	RAD0.2	1	C6
0.1F	RAD0.2	7	C7, C8, C9, C10, C11, C12, C13
10K	SIP9	1	S1
10F	RAD0.2	1	C5
10F	RB.2/.4	1	C4
10K	AXIAL0.5	1	R2
10F	RB.2/.4	1	C1
11.0592 MHz	XTAL1	1	XTAL
1K	AXIAL0.3	3	R3, R4, R5
22Pf	RAD0.2	2	C2, C3
26PIN	IDC26	1	I1
2764	DIP28	1	IC4
4.7K	AXIAL0.5	1	R1
40PIN	IDC40	1	I3
6264	DIP28	2	IC5, IC6
7408	DIP14	1	IC8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

74432	DIP14	1	IC9
74ALS245	DIP20	3	B1, B2, B3
74LS125	DIP14	1	IC10
74LS138	DIP16	2	IC3, IC12
74LS373	DIP20	1	IC2
7805	TO-220	1	P1
8051	DIP40	1	IC1
8255	DIP40	2	IC7, IC11
P1	SIP8	1	S2
RESET	RAD0.3	1	SW1

วงจรรักษาขั้วพอร์ตการทำงาน

Comment	Pattern	Quantity	Components
	DIP16	1	IC1
	SIP8	1	PORT P1
	SIP9	1	A4
1K	AXIAL0.5	4	R1, R2, R3, R4
74LS00	DIP14	1	IC6
74LS125	DIP14	1	IC7
74LS138	DIP16	1	IC10
74LS245	DIP20	3	IC4, IC8, IC9
74LS32	DIP14	1	IC11
74LS85	DIP16	2	IC2, IC3
8255	DIP40	1	IC5
PORT FDC	IDC36	1	I3
PORT MCS-51	IDC40	1	I2
PORT Z80	IDC40	1	I1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IS80C51/31

ISSI®

CMOS SINGLE CHIP 8-BIT MICROCONTROLLER

 PRELIMINARY
JUNE 1997

FEATURES

- 4K x 8 ROM (IS80C51 only)
- 128 x 8 RAM
- Two 16-bit timers/counters
- Full duplex serial channel
- Boolean processor
- Four 8-bit I/O ports, 32 I/O lines
- External memory expandable to 128K
- CMOS and TTL compatible
- Maximum speed range: 40 MHz @ $V_{CC} = 5V$
- Packages available:
 - 40-pin DIP
 - 44-pin PLCC
 - 44-pin PQFP

GENERAL DESCRIPTION

The ISSI IS80C51/31 is a high-performance microcontroller fabricated using high-density CMOS technology. The CMOS IS80C51/31 is functionally compatible with the NMOS Intel 8051/31 microcontroller.

The IS80C51 is designed with 4K x 8 ROM; 128 x 8 RAM; 32 I/O lines for either multiprocessor communications, I/O expansion, or full duplex UART; two 16-bit timers/counters; a five-source, two-priority-level, nested interrupt structure; and an on-chip oscillator and clock circuit. The IS80C51 can be expanded using standard TTL compatible memory.

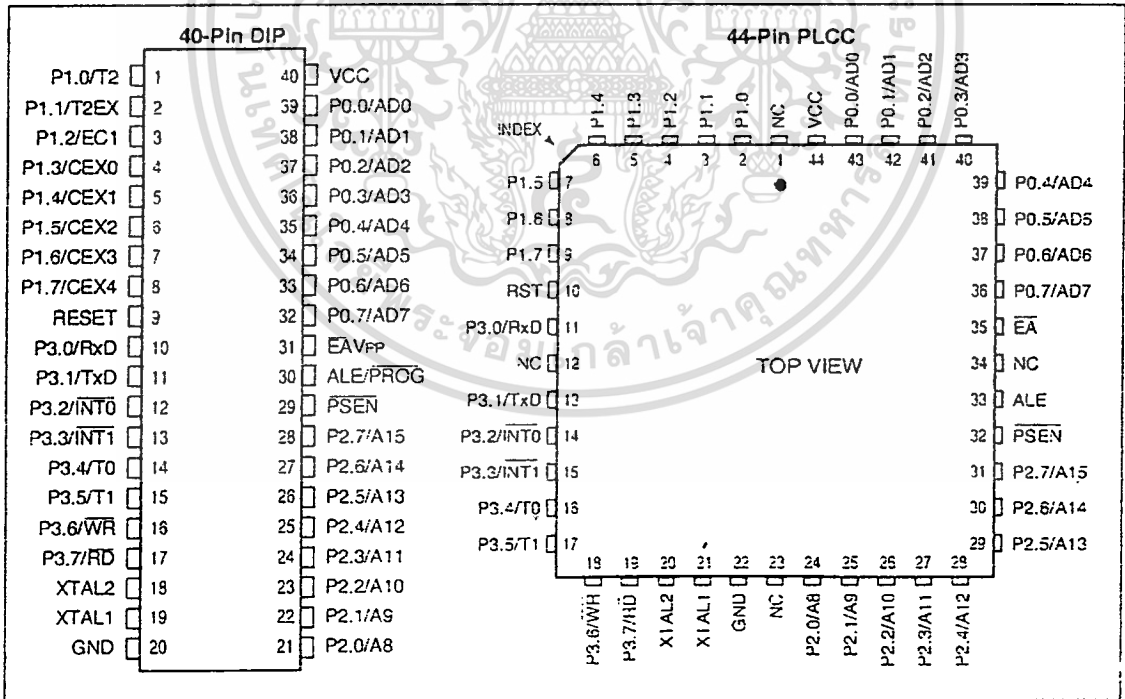


Figure 1. IS80C51/31 Pin Configurations

This document contains PRELIMINARY data. ISSI reserves the right to make changes to its products at any time without notice in order to improve design and supply the best possible product. We assume no responsibility for any errors which may appear in this publication. © Copyright 1997, Integrated Silicon Solution, Inc.

Integrated Silicon Solution, Inc.

PRELIMINARY MC003-18 สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

5/10/97

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

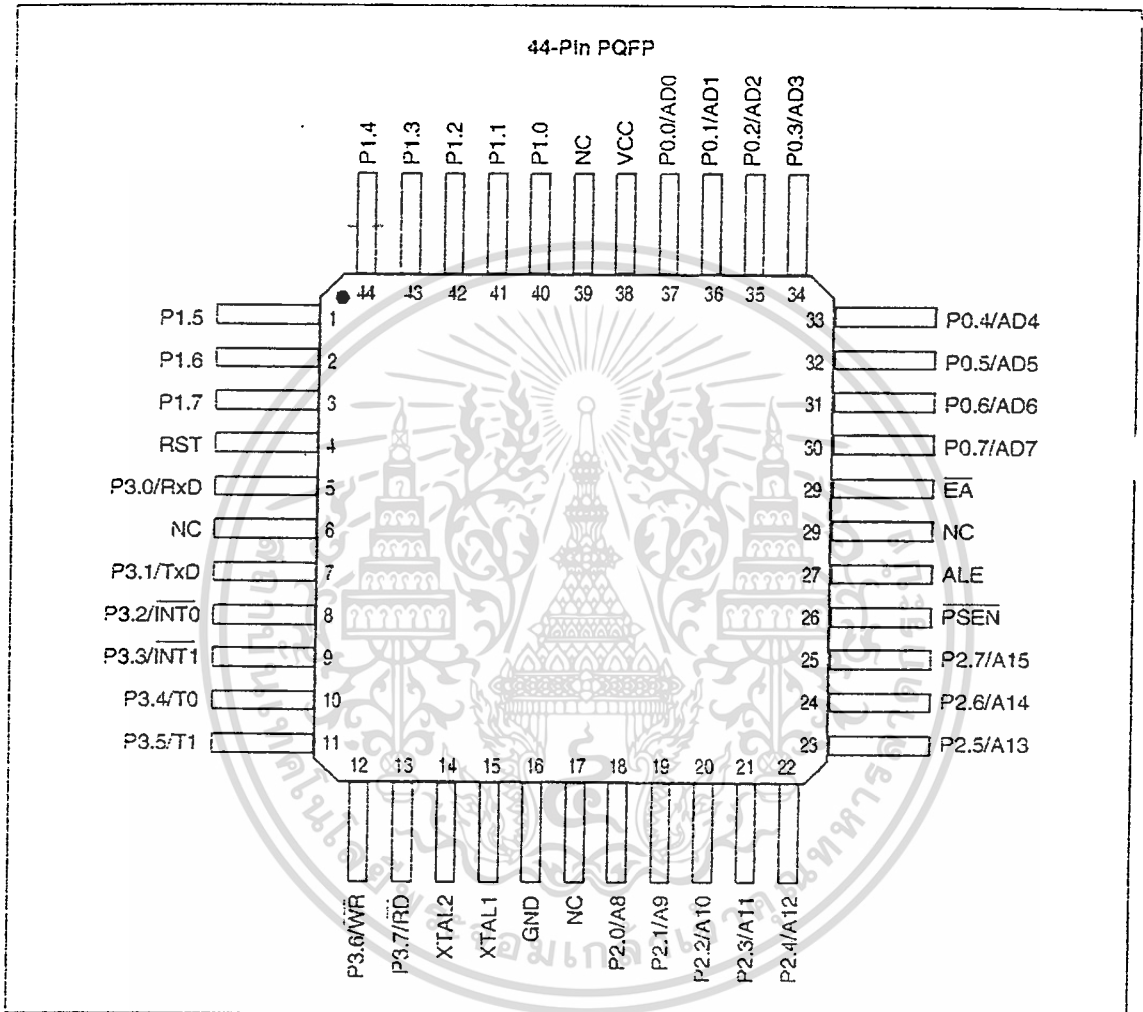


Figure 1. IS80C51/31 Pin Configurations (continued)

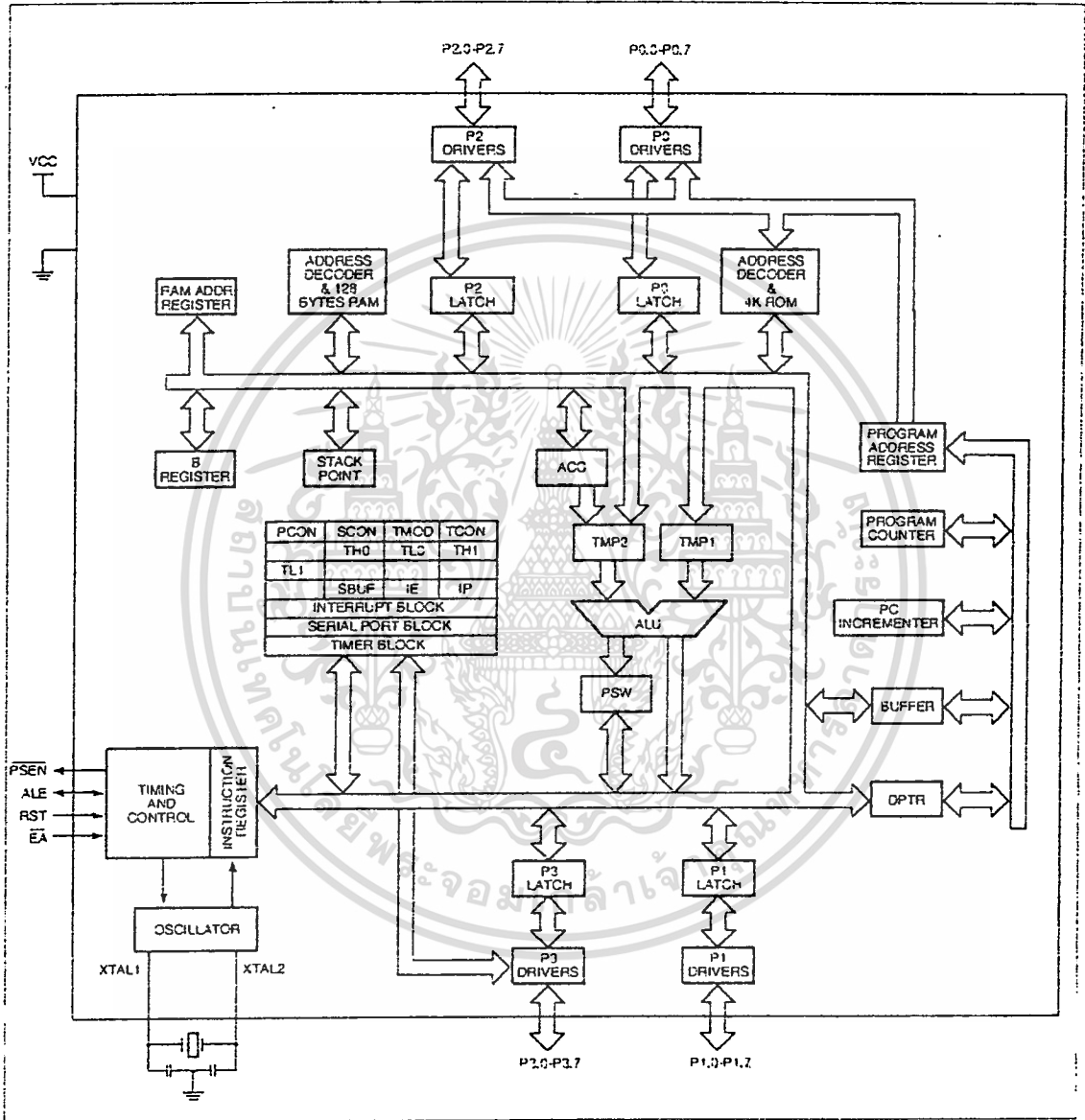


Figure 2. IS80C51 Block Diagram

PIN DESCRIPTION

Symbol	DIP	PLCC	QFP	I/O	Name and Function
P1.0-P1.7	1-8	2-9	40-44	I/O	Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. Port 1 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally pulled low will source current because of the internal pullups.
RST	9	10	4	I	Reset: A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to GND permits a power-on reset using only an external capacitor. A small internal resistor permits power-on reset using only a capacitor connected to VCC.
P3.0-P3.7	10-17	11, 13-19	5, 7-13	I/O	Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current because of the internal pullups. Port 3 also serves the special features of the IS80C51, as listed below:
	10	11	5	I	RxD (P3.0): Serial input port
	11	13	7	O	TxD (P3.1): Serial output port
	12	14	8	I	INT0 (P3.2): External interrupt
	13	15	9	I	INT1 (P3.3): External interrupt
	14	16	10	I	T0 (P3.4): Timer 0 external input
	15	17	11	I	T1 (P3.5): Timer 1 external input
	16	18	12	O	WR (P3.6): External data memory write strobe
	17	19	13	O	RD (P3.7): External data memory read strobe
XTAL 2	18	20	14	O	Crystal 2: Output from the inverting oscillator amplifier.
XTAL 1	19	21	15	I	Crystal 1: Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
GND	20	22	16	I	Ground: 0V reference.
P2.0-P2.7	21-28	24-31	18-25	I/O	Port 2: Port 2 is an 8-bit bi-directional I/O port with internal pullups. Port 2 pins that have 1s written to them are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally pulled low will source current because of the internal pullups. Port 2 emits the high order address byte during fetches from external program memory and during accesses to external data memory that used 16-bit addresses. In this application, it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses, Port 2 emits the contents of the P2 special function register.

(continued)

IS80C51/31

ISSI

PIN DESCRIPTION (continued)

Pin Symbol	DIP	PLCC	QFP	I/O	Name and Function
PSEN	29	32	26	O	Program Store Enable: The read strobe to external program memory. When the device is executing code from the external program memory, PSEN is activated twice each machine cycle except that two PSEN activations are skipped during each access to external data memory. PSEN is not activated during fetches from internal program memory.
ALE	30	33	27	I/O	Address Latch Enable: Output pulse for latching the low byte of the address during an address to the external memory. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency, and be used for external timing or clocking. Note that one ALE pulse is skipped during each access external data memory.
$\overline{EA}$	31	35	29	I	External Access enable: $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to 0FFFFH. If $\overline{EA}$ is held high, the device executes from internal program memory unless the program counter contains an address greater than 0FFFFH.
P0.0-P0.7	39-32	43-36	30-37	I/O	Port 0: Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pullups when emitting 1s.
Vcc	40	44	38	I	Power Supply: This is the power supply voltage for operation.

OPERATING DESCRIPTION

The detail description of the IS80C51 included in this description are:

- Memory map and registers
- Timers/counters
- Serial interface
- Interrupt system
- Instruction
- Other information

Memory Map and Registers

Memory

The IS80C51 has separate address spaces for program and data memory. The program memory can be up 64K bytes long. The lower 4K can reside on-chip. Figure 3 shows a map of the IS80C51 program memory.

The IS80C51 has 128 bytes of on-chip RAM, plus a number of special function registers. The lower 128 bytes can be accessed either by direct addressing or by indirect addressing. Figure 4 shows data memory origination.

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into three segments as listed below and shown in Figure 5.

1. **Register Banks 0-3:** locations 0 through 1FH (32 bytes). The device after reset defaults to register bank 0. To use the other register banks, the user must select them in software. Each register bank contains eight 1-byte registers 0-7. Reset initializes the stack point to location 07H, and is incremented once to start from 08H, which is the first register of the second bank.

2. **Bit Addressable Area:** 16 bytes have been assigned for this segment 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH). The bits can be referred to in two ways, both of which are acceptable by most assemblers. One way is referencing to bytes 20H-2FH. Thus, bits 0-7 can also be referred to as bits 20.0-20.7, and bits 8-15 are the same as 21.0-21.7, and so on. Each of the 16 bytes in this segment can also be addressed as a byte.

3. **Scratch Pad Area:** 30-7FH are available to the user as data RAM. However, if the data pointer has been initialized to this area, enough bytes should be left aside to prevent SP data destruction.

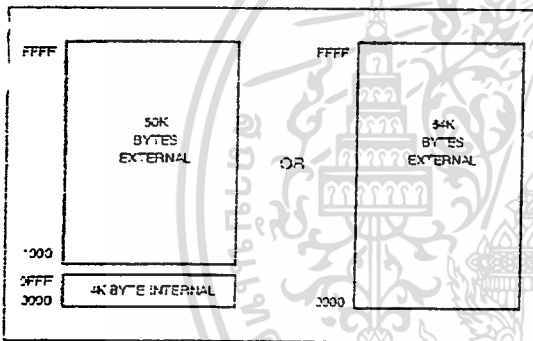


Figure 3. Program Memory Access Range

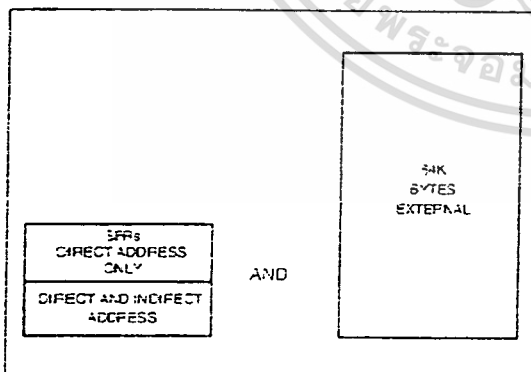


Figure 4. Data Memory Access Range

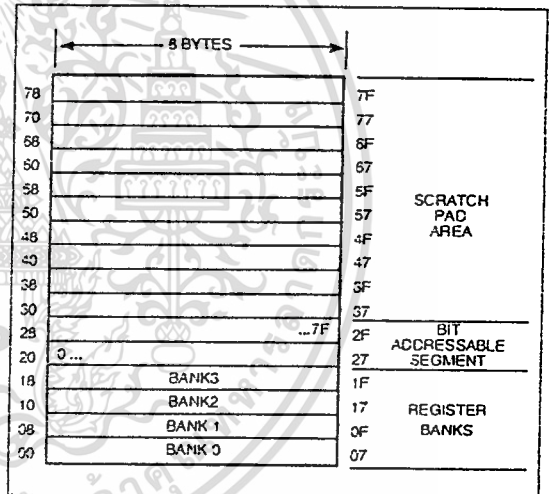


Figure 5. 128 Bytes of RAM Direct and Indirect Addressable

SPECIAL FUNCTION REGISTER

Symbol	Description	Direct Address	Bit Address, Symbol, or Alternative Port Function								Reset Value
ACC ⁽¹⁾	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B ⁽¹⁾	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
IE ⁽¹⁾	Interrupt enable	A8H	AF	AE	AD	AC	AB	AA	A9	A8	0X000000B
			EA	—	—	ES	ET1	EX1	ET0	EX0	
IP ⁽¹⁾	Interrupt priority	B8H	BF	BE	BD	BC	BB	BA	B9	B8	XXX00000B
			—	—	—	PS	PT1	PX1	PT0	PX0	
P0 ⁽¹⁾	Port 0	80H	87	86	85	84	83	82	81	80	FFH
			P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
P1 ⁽¹⁾	Port 1	90H	97	96	95	94	93	92	91	90	FFH
			P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
P2 ⁽¹⁾	Port 2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	FFH
			P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
P3 ⁽¹⁾	Port 3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	FFH
			P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	
PCON	Power control	87H	SMOD	—	—	—	GF1	GF0	PD	IDL	0XXX0000B
PSW ⁽¹⁾	Program status word	D0H	D7	D6	D5	D4	D3	D2	D1	D0	00H
			CY	AC	F0	RS1	RS0	OV	—	P	
SBUF	Serial data buffer	99H									XXXXXXXXB
SCON ⁽¹⁾	Serial controller	98H	9F	9E	9D	9C	9B	9A	99	98	00H
			SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
SP	Stack pointer	81H									07H
TCON ⁽¹⁾	Timer control	88H	8F	8E	8D	8C	8B	8A	89	88	
			TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H

Note:

1. Denotes bit addressable.

The detail description of each bit is as follows:

PSW

CY	AC	F0	RS1	RS0	OV	—	P
CY	PSW.7	Carry flag					
AC	PSW.6	Auxiliary carry flag					
F0	PSW.5	Flag 0 available to the user for general purpose					
RS1	PSW.4	Register bank selector bit 1					
RS0	PSW.3	Register bank selector bit 0 ⁽¹⁾					
OV	PSW.2	Overflow flag					
—	PSW.1	Usable as a general purpose flag					
P	PSW.0	Parity flag. Set/clear by hardware each instruction cycle to indicate an odd/even number of "1" bus in the accumulator					

Note:

1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

PCON

SMOD	—	—	—	GF1	GF0	PD	IDL
SMOD							
—							
—							
—							
GF1							
GF0							
PD							
IDL							

IE

EA	—	—	ES	ET1	EX1	ET0	EX0
EA	IE.7						
—	IE.6						
—	IE.5						
ES	IE.4						
ET1	IE.3						
EX1	IE.2						
ET0	IE.1						
EX0	IE.0						

IP

—	—	—	PS	PT1	PX1	PT0	PX0
—	IP.7						
—	IP.6						
—	IP.5						
PS	IP.4						
PT1	IP.3						
PX1	IP.2						
PT0	IP.1						
PX0	IP.0						

TCON

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the timer/counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.					
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 ON/OFF.					
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the timer/counter 0 overflows. Cleared by hardware as processor vectors to the interrupt service routine.					
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 ON/OFF.					
IE1	TCON.3	External interrupt 1 edge flag. Set by software when external interrupt edge is detected. Cleared by hardware when interrupt is processed.					
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software specify falling edge/low level triggered external interrupt.					
IE0	TCON.1	External interrupt 0 edge flag. Set by software when external interrupt edge is detected. Cleared by hardware when interrupt is processed.					
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software specify falling edge/low level triggered external interrupt.					

TMOD

Timer 1				Timer 0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
GATE When TRx (in TCON) is set and GATE=1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE=0, TIMER/COUNTERx will run only while TRx=1 (software control).							
C/T Timer or counter selector. Cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).							
M1 Mode selector bit. ⁽²⁾							
M0 Mode selector bit. ⁽²⁾							

Note 2:

M1	M0	Operating mode
0	0	Mode 0. (13-bit timer)
0	1	Mode 1. (16-bit timer/counter)
1	0	Mode 2. (8-bit auto-load timer/counter)
1	1	Mode 3. (TL0 is an 8-bit timer/counter controller by the standard timer 0 control bits. TH0 is an 8-bit timer and is controlled by timer 1 controller bits.)
1	1	Mode 3. (Timer/counter 1 stopped).

SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SM0	SCON.7	Serial port mode specifier. ⁽³⁾					
SM1	SCON.6	Serial port mode specifier. ⁽³⁾					
SM2	SCON.5	Enable the multiprocessor communication feature in mode 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 valid stop bit was not receive. In mode 0, SM2 should be 0.					
REN	SCON.4	Set/cleared by software to enable/disable reception.					
TB8	SCON.3	The 9th bit that will be transmitted in mode 2 and 3. Set/cleared by software.					
RB8	SCON.2	In modes 2 and 3, RB8 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	SCON.1	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. Must be cleared by software.					
RI	SCON.0	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 (except see SM2). Must be cleared by software.					

Note 3:

SM0	SM1	MODE	Description	Baud rate
0	0	0	Shift register	Fosc/12
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fosc/64 or Fosc/32
1	1	3	9-bit UART	Variable

Timers/counters

The IS80C51 contains two 16-bit counters for measuring time intervals, measuring pulse widths, counting events and generating precise, periodic interrupt request.

The operating mode is listed below.

Timer 1/Counter 1:

Timer 1/Counter 1 can be configured in one of four modes:

Mode 0: Provides an 8-bit counter with a divide-by-32 prescaler or an 8-bit timer with a divide-by-32 prescaler. A read/write of TH1 accesses counter 1's bits 12-5. A read/write of TL1 accesses counter 1's bits 7-0. TL1 bits 4-0 are the prescaler (counter 1's 4-0) while bits 7-5 are indeterminate and should be ignored. The programmer should clear the prescaler (counter 1's bits 4-0) before setting the run flag.

Mode 1: Configures counter 1 as a 16-bit timer/counter.

Mode 2: Configures counter 1 as an 8-bit auto-reload value. TH1 holds the reload value. TL1 is incremented. The value in TH1 is reload onto TL1 when TL1 overflows from all ones.

Mode 3: When counter 1's mode is repro-grammed to mode 3 (from mode 0, 1 or 2), it disables the incrementing of the counter. This mode is provided as an alternative to use TR1 bit (TCON.6) to start and stop counter 1.

The serial port receives a pulse each time that counter 1 overflows. The standard UART mode divides this pulse rate to generate the transmission rate.

Counter 0/ Timer 0:

Counter 0 can also be configured in one of four modes:

- Mode 0-2: Mode 0-2 are the same as for counter 1.
- Mode 3: In mode 3, the configure of TH0 is not affected by the bits in TMOD or TCON. It is configured solely as an 8-bit timer that is enabled for incrementing by TCON's TR1 bit. Upon TH0's overflow the TF1 flag gets set. Thus, neither TR1 nor TF1 is available to counter 1 when counter 0 is in mode 3. The function of

TR1 can be done by placing counter 1 in mode 3, so only the function of TF1 is actually given up by counter 1. In mode 3, TL0 is configured as an 8-bit timer/counter and is controlled, as usual, by the GATE (TMOD. 3), C/ (TMOD.2), TR0 (TCON.4) and TF0 (TCON.5) control bits.

The use of the timers/counters is determined by two 8-bit registers, TMOD and TCON is shown in SFR. The counter input circuit is shown in Figures 6 and 7.

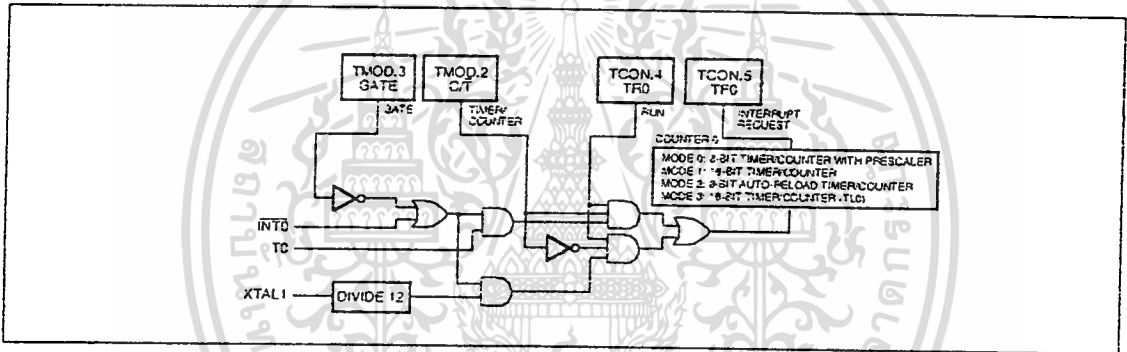


Figure 6. Timer/event Counter 0 Control and Status Flag Circuit

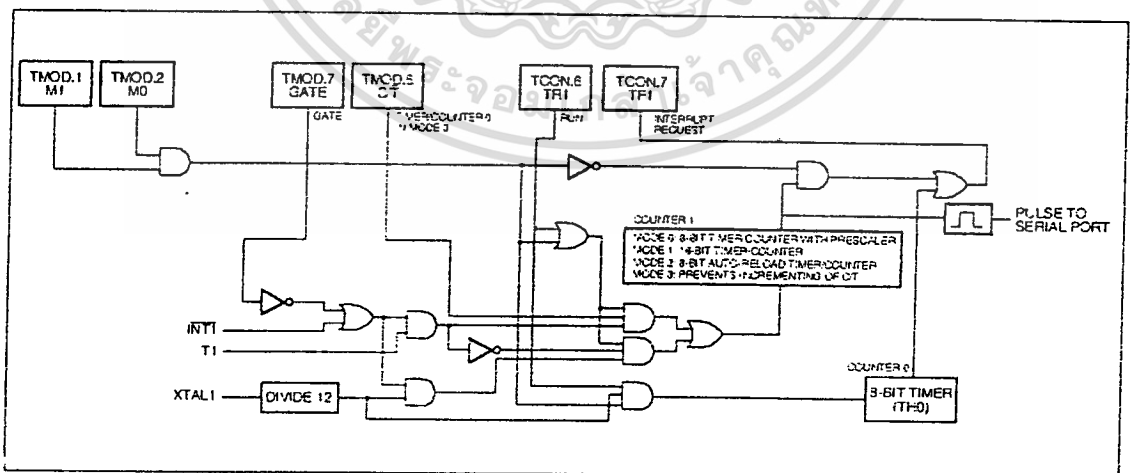


Figure 7. Timer/event Counter 1 Control and Status Flag Circuit

Serial Interface

The IS80C51 has a serial I/O port that is useful for serial linking peripheral devices as well as multiple IS80C51s through standard asynchronous protocol with full-duplex operations. The data for transmission to and from reception reside in the serial port buffer register (SBUF). The serial port control and the monitoring of its status is provided by the serial port control register (SCON). The contents of the 8-bit SCON register are shown in SFR.

The IS80C51 has a serial channel useful for serially linking UART (universal asynchronous receiver/transmitter) devices and for expanding I/O. The full-

duplex serial I/O port can be programmed to function in one of four operating modes:

- Mode 0: Synchronous I/O expansion using TTL or CMOS shift registers.
- Mode 1: UART interface with 10-bit frame and variable transmission rate.
- Mode 2: UART interface with 11-bit frame and fixed transmission rate.
- Mode 3: UART interface with 11-bit frame and variable transmission rate.

The serial interface circuit is shown in Figures 8 and 9. The use of the serial interface is determined by TCON and PCON registers is shown in SFR.

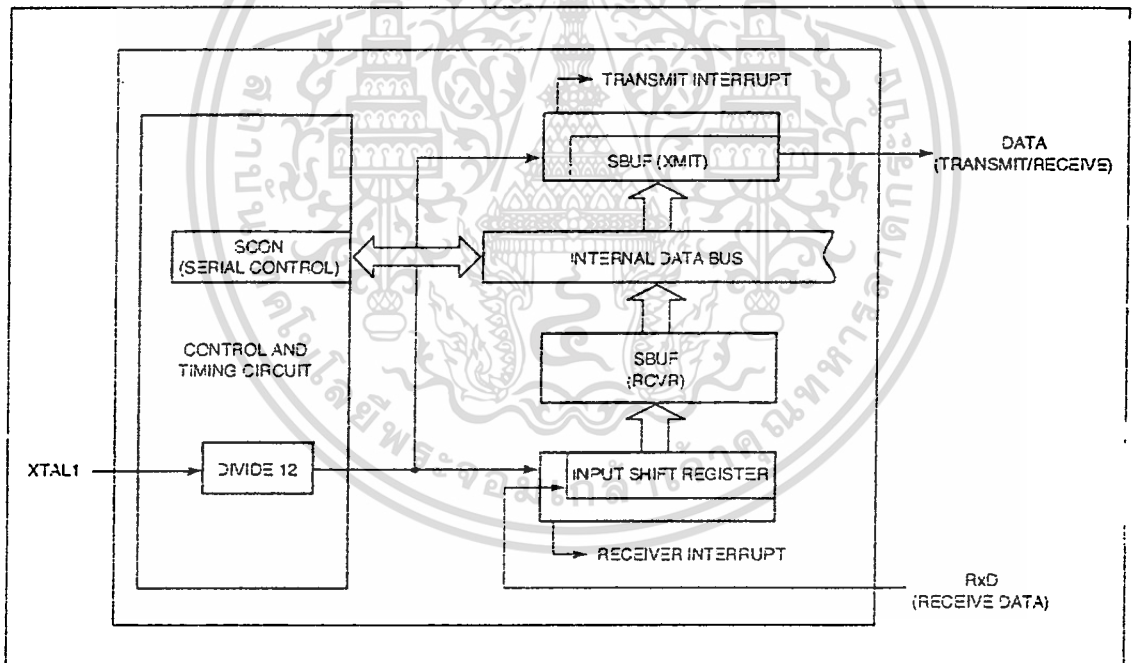


Figure 8. Serial Port: Synchronous Mode 0

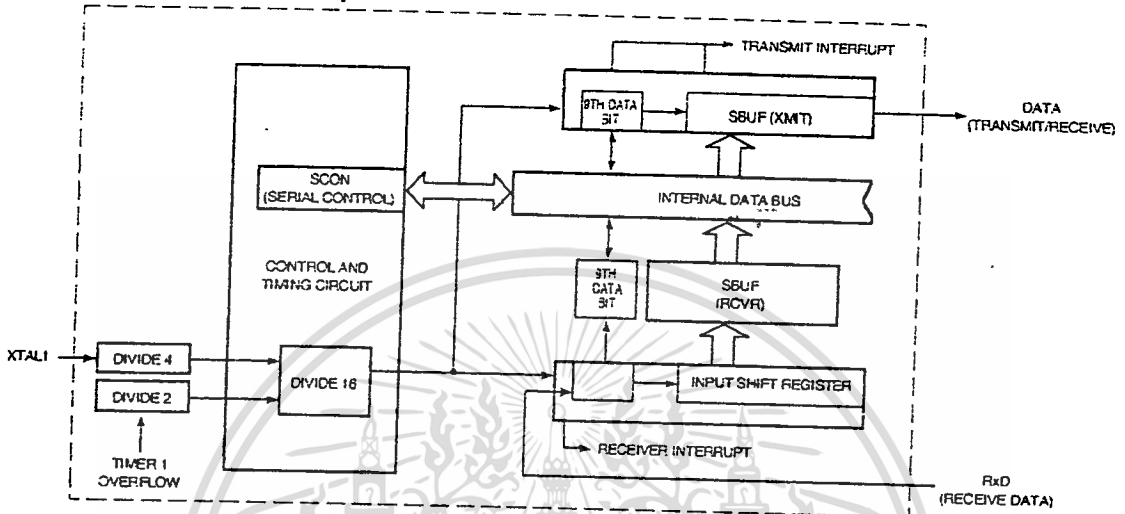


Figure 9. Serial Port: UART Mode 1, 2, and 3

Interrupt System

External events and the real-time driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section code. To tie the asynchronous activities of these functions to normal program execution, a sophisticated multiple-

source, two-priority-level, nested interrupt system is provided. The interrupt system is shown in Figure 10. The interrupt request flag and program memory location of interrupt service program is shown in the table below:

Interrupt Source	Request Flag	Bit Location	Start Address
External Request 0	IE0	TCON.1	3 (0003H)
Internal Timer 0/Counter 0	TF0	TCON.5	11 (000BH)
External Request 1	IE1	TCON.3	19 (0013H)
Internal Timer 1/Counter 1	TF1	TCON.7	27 (001BH)
Internal Serial Port (XMIT)	TI	SCON.1	35 (0023H)
Internal Serial Port (RCVR)	RI	SCON.0	35 (0023H)

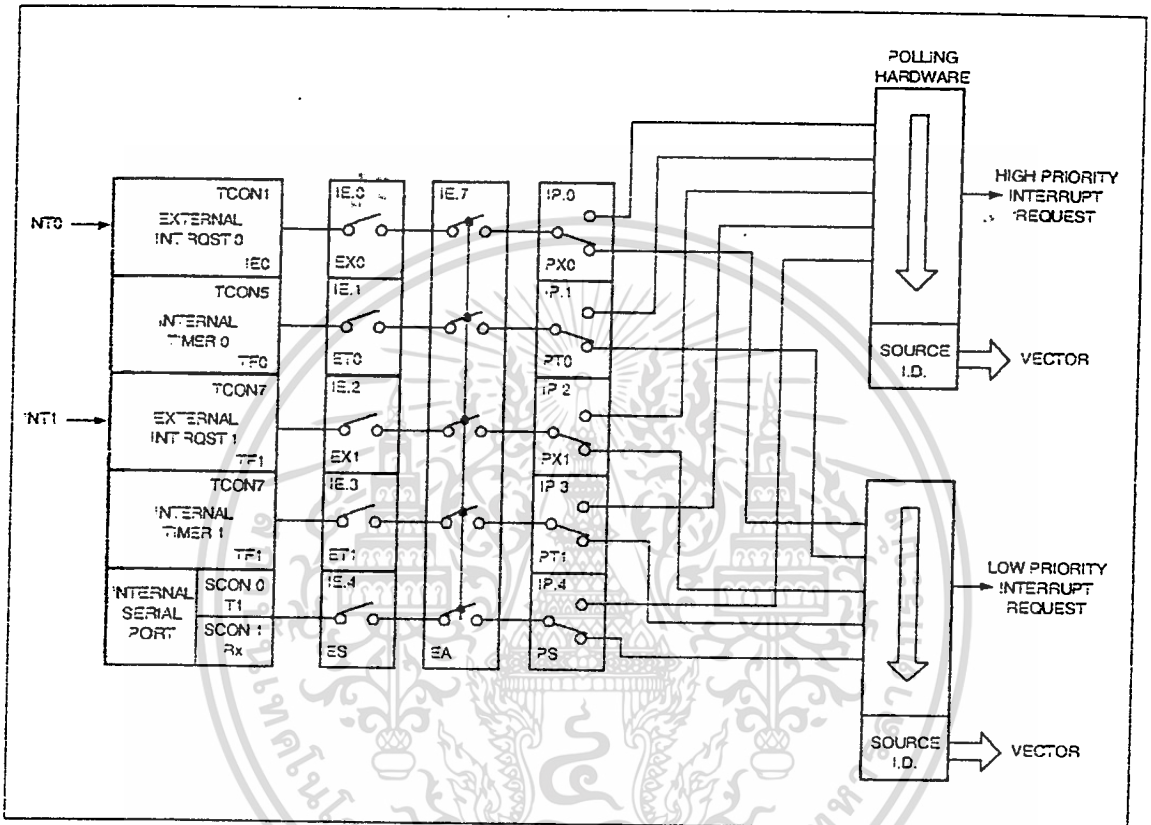


Figure 10. Interrupt System

ADDITIONAL INFORMATION

Idle mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the Special Function Register remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin, when idle is terminated by reset, the instruction following the one that invokes idle should not be one that writes to a port pin or to external memory. The status of external pins is shown in the following table.

Power-down mode

In the power-down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and special function register retain their values until the power down mode is terminated.

The only exit from power-down mode is a hardware reset. Reset redefines the SFR but does not change the on-chip RAM. 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. The status of the external pins during idle and power-down mode is shown in the following table.

Status of the External Pins during Idle and Power-down modes.

Mode	Memory	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

ROM Verification

The address of the program memory location to be read is applied to Port 1 and pins P2.3-P2.0. The other pins should be held at the "Verify" level are indicated in Figure 11. The contents of the addressed locations exits on Port 0. External pullups are required on Port 0 for this operation. Figure 11 shows the setup to verify the program memory.

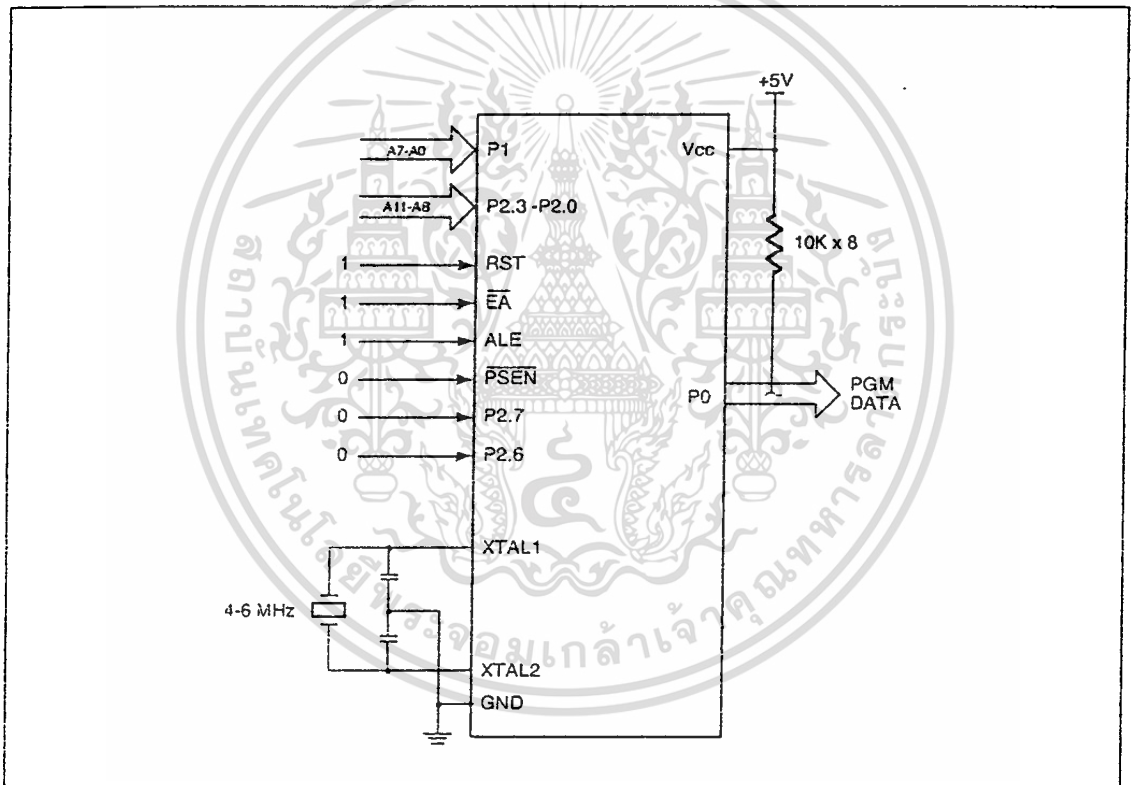


Figure 11. ROM Verification

ABSOLUTE MAXIMUM RATINGS⁽¹⁾

Symbol	Parameter	Value	Unit
V _{TERM}	Terminal Voltage with Respect to GND ⁽²⁾	-2.0 to +7.0	V
T _{BIAS}	Temperature Under Bias ⁽³⁾	0 to +70	°C
T _{STG}	Storage Temperature	-65 to +125	°C
P _T	Power Dissipation	1.5	W

Note:

1. Stress greater than those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.
2. Minimum DC input voltage is -0.5V. During transitions, inputs may undershoot to -2.0V for periods less than 20 ns. Maximum DC voltage on output pins is V_{CC} + 0.5V which may overshoot to V_{CC} + 2.0V for periods less than 20 ns.
3. Operating temperature is for commercial products only defined by this specification.

OPERATING RANGE⁽¹⁾

Range	Ambient Temperature	V _{CC}	Oscillator Frequency
Commercial	0°C to +70°C	5V ± 10%	3.5 to 40 MHz

Note:

1. Operating ranges define those limits between which the functionality of the device is guaranteed.

IS80C51/31



DC CHARACTERISTICS

(TA = 0°C to 70°C; Vcc = 5V ± 10%; GND = 0V)

Symbol	Parameter	Test conditions	Min	Max	Unit
V _{IL}	Input low voltage		-0.5	0.2V _{cc} - 0.1	V
V _{IL1}	Input low voltage		-0.5	0.2V _{cc} - 0.3	V
V _{IH}	Input high voltage (All except XTAL1, RST)		0.3V _{cc} + 0.9	V _{cc} + 0.5	V
V _{IH1}	Input high voltage (XTAL1, EA)		0.6V _{cc}	V _{cc} + 0.5	V
V _{SCH+}	RST positive schmitt-trigger ¹ threshold voltage		0.7V _{cc}	V _{cc} + 0.5	V
V _{SCH-}	RST negative schmitt-trigger threshold voltage		0	0.2V _{cc}	V
V _{OL} ⁽¹⁾	Output low voltage (Ports 1, 2, and 3)	I _{OL} = 100 µA	—	0.3	V
		I _{OL} = 1.6 mA	—	0.45	V
		I _{OL} = 3.5 mA	—	1.0	V
V _{OL1} ⁽¹⁾	Output low voltage (Port 0, ALE, PSEN)	I _{OL} = 200 µA	—	0.3	V
		I _{OL} = 3.2 mA	—	0.45	V
		I _{OL} = 7.0 mA	—	1.0	V
V _{OH}	Output high voltage (Port 1, 2, 3, ALE, PSEN)	I _{OH} = -10 µA V _{cc} = 4.5V-5.5V	0.9V _{cc}	—	V
		I _{OL} = -25 µA	0.75V _{cc}	—	V
		I _{OL} = -60 µA	2.4	—	V
V _{OH1}	Output high voltage (Port 0, ALE, PSEN)	I _{OH} = -80 µA V _{cc} = 4.5V-5.5V	0.9V _{cc}	—	V
		I _{OH} = -300 µA	0.75V _{cc}	—	V
		I _{OH} = -800 µA	2.4	—	V
I _{IL}	Logical 0 input current (Port 1, 2, 3)	V _{IN} = 0.45V	—	-50	µA
I _{L1}	Input leakage current (Port 0)	0.45V < V _{IN} < V _{cc}	-10	+10	µA
I _{TL}	Logical 1-to-0 transition current (Port 1, 2, 3)	V _{IN} = 2.0V	—	-650	µA
RRST	RST pulldown resistor		50	300	Kohm

Note:

1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:Maximum I_{OL} per port pin: 10 mAMaximum I_{OL} per 8-bit port

Port 0: 26 mA

Port 1, 2, 3: 15 mA

Maximum total I_{OL} for all output pins: 71 mAif I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink greater than the listed test conditions.

IS80C51/31

ISSI

POWER SUPPLY CHARACTERISTICS

Symbol	Parameter	Test conditions	Min	Max	Unit
I_{CC}	Power supply current ⁽¹⁾	Active mode	—	20	mA
	Idle mode	12 MHz	—	5	mA
	Power-down mode	40 MHz	—	15	mA

Note:

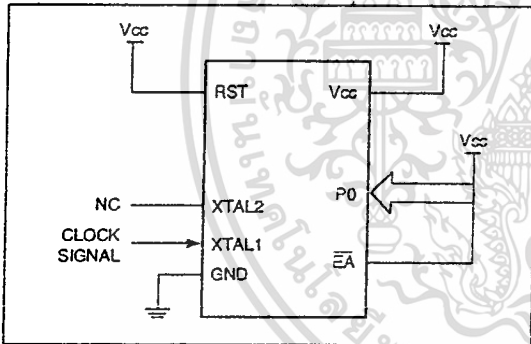
1. See Figures 12, 13, 14, and 15 for I_{CC} test conditions.

Figure 12. Active Mode

IS80C51/31 Active Current

Clock	Vcc = 4.5V	Vcc = 5.0V	Vcc = 5.5V
2 MHz	0.71 mA	0.81 mA	0.94 mA
4 MHz	1.38 mA	1.59 mA	1.83 mA
6 MHz	2.05 mA	2.37 mA	2.71 mA
8 MHz	2.65 mA	3.08 mA	3.49 mA
10 MHz	3.69 mA	4.23 mA	4.81 mA
12 MHz	4.00 mA	4.62 mA	5.24 mA

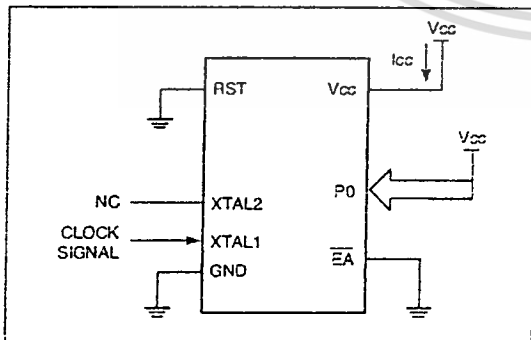


Figure 13. Idle Mode

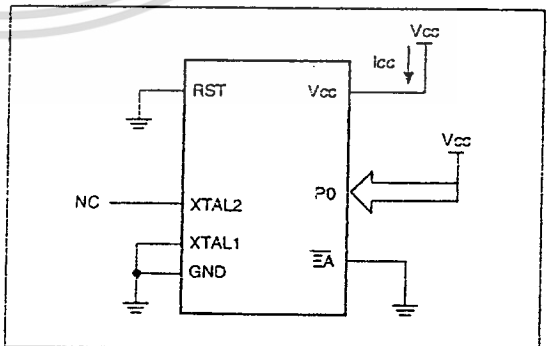


Figure 14. Power-down Mode

Integrated Silicon Solution, Inc.

PRELIMINARY MC003-1B

5/10/97

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

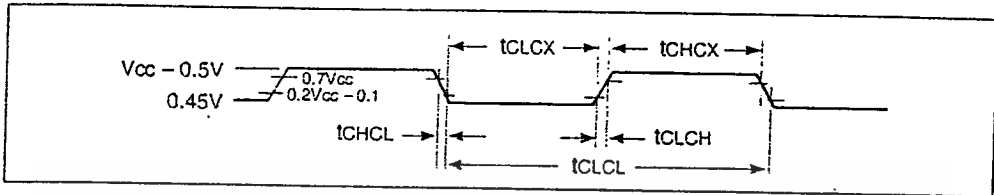


Figure 15. Icc Test Conditions

Clock signal waveform for Icc tests in active and idle mode (tCLCH = tCHCL = 5 ns)

AC CHARACTERISTICS

(TA = 0°C to 70°C; Vcc = 5V ± 10%; GND = 0V; CL for Port 0, ALE and PSEN; Outputs = 100 pF; CL for other outputs = 80 pF)

External Memory Characteristics

Symbol	Parameter	24 MHz Clock		40 MHz Clock		Variable Oscillator		Unit
		Min	Max	Min	Max	Min	Max	
1/tCLCL	Oscillator frequency	—	—	—	—	3.5	40	MHz
tLHLL	ALE pulse width	43	—	10	—	2tCLCL-40	—	ns
tAVLL	Address valid to ALE low	2	—	-15	—	tCLCL-40	—	ns
tLLAX	Address hold after ALE low	7	—	-10	—	tCLCL-35	—	ns
tLLIV	ALE low to valid instr in	—	105	—	55	—	3tCLCL-20	ns
tLLPL	ALE low to PSEN low	2	—	-15	—	tCLCL-40	—	ns
tPLPH	PSEN pulse width	80	—	30	—	3tCLCL-45	—	ns
tPLIV	PSEN low to valid instr in	—	73	—	40	—	2tCLCL-10	ns
tpXIX	Input instr hold after PSEN	0	—	0	—	0	—	ns
tpXIZ	Input instr float after PSEN	—	73	—	40	—	2tCLCL-10	ns
tAVIV	Address to valid instr in	—	147	—	80	—	4tCLCL-20	ns
tPLAZ	PSEN low to address float	—	10	—	10	—	10	ns
tRLRH	RD pulse width	150	—	50	—	6tCLCL-100	—	ns
tRLWH	WR pulse width	150	—	50	—	6tCLCL-100	—	ns
tALDV	RD low to valid data in	—	114	—	30	—	5tCLCL-95	ns
tRHOX	Data hold after RD	0	—	0	—	0	—	ns
tRHOZ	Data float after RD	—	63	—	30	—	2tCLCL-70	ns
tLDOV	ALE low to valid data in	—	244	—	110	—	8tCLCL-90	ns
tAVDV	Address to valid data in	—	285	—	135	—	9tCLCL-90	ns
tLLWL	ALE low to RD or WR low	75	175	25	125	3tCLCL-50	3tCLCL+50	ns
tAVWL	Address to RD or WR low	77	—	10	—	4tCLCL-90	—	ns
tOVVIX	Data valid to WR transition	2	—	-1.5	—	tCLCL-40	—	ns
tVHOX	Data hold after WR	2	—	-1.5	—	tCLCL-40	—	ns
tOVVWH	Data valid to WR high	219	—	105	—	7tCLCL-70	—	ns
tRLAZ	RD low to address float	—	63	—	30	—	2tCLCL-20	ns
tVHWH	RD or WR high to ALE high	2	82	-15	65	tCLCL-40	tCLCL+40	ns

Serial Port Timing: Shift Register Mode

Symbol	Parameter	24 MHz Clock		40 MHz Clock		Variable Oscillator		Unit
		Min	Max	Min	Max	Min	Max	
txLXL	Serial port clock cycle time	500	—	—	—	12tCLCL	—	ns
tdvXH	Output data setup to clock rising edge	284	—	117	—	10tCLCL-133	—	ns
tdHdX	Output data hold after clock rising edge	33	—	0	—	2tCLCL-50	—	ns
tdHdX	Input data hold after clock rising edge	0	—	0	—	0	—	ns
tdHdV	Clock rising edge to input data valid	—	284	—	117	—	10tCLCL-133	ns

External Clock Drive

Symbol	Parameter	Min	Max	Unit
1/tCLCL	Oscillator Frequency	3.5	40	MHz
tCHCX	High time	10	—	ns
tCLCX	Low time	10	—	ns
tCLCH	Rise time	—	10	ns
tCHCL	Fall time	—	10	ns

ROM Verification Characteristics

Symbol	Parameter	Min	Max	Unit
1/tCLCL	Oscillator Frequency	2.5	40	MHz
tAVGV	Address to data valid	—	48tCLCL	
tELOV	ENABLE low to data valid	—	48tCLCL	
tEHOZ	Data float after ENABLE	0	48tCLCL	

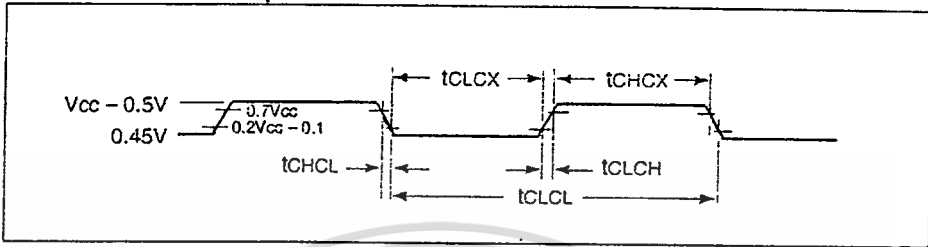


Figure 20. External Clock Drive Waveform

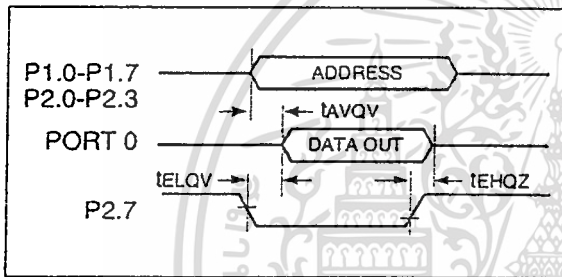


Figure 21. ROM Verification Waveforms

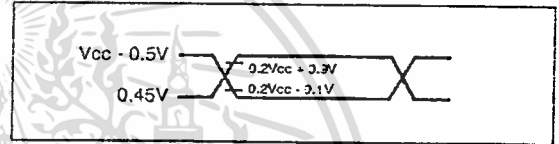


Figure 22. AC Test Point

Note:

1. AC inputs during testing are driven at $V_{CC} - 0.5V$ for logic "1" and $0.45V$ for logic "0". Timing measurements are made at V_{IH} min for logic "1" and max for logic "0".

IS80C51/31

ISSI

ORDERING INFORMATION

Speed	Order Part Number	Package
12 MHz	IS80C51-12PL	PLCC - Plastic Leaded Chip Carrier
	IS80C51-12PQ	PQFP - Plastic Quad Flat Pack
	IS80C51-12W	600-mil Plastic DIP
20 MHz	IS80C51-20PL	PLCC - Plastic Leaded Chip Carrier
	IS30C51-20PQ	PQFP - Plastic Quad Flat Pack
	IS80C51-20W	600-mil Plastic DIP
24 MHz	IS80C51-24PL	PLCC - Plastic Leaded Chip Carrier
	IS80C51-24PQ	PQFP - Plastic Quad Flat Pack
	IS80C51-24W	600-mil Plastic DIP
40 MHz	IS80C51-40PL	PLCC - Plastic Leaded Chip Carrier
	IS80C51-40PQ	PQFP - Plastic Quad Flat Pack
	IS80C51-40W	600-mil Plastic DIP
12 MHz	IS80C31-12PL	PLCC - Plastic Leaded Chip Carrier
	IS80C31-12PQ	PQFP - Plastic Quad Flat Pack
	IS80C31-12W	600-mil Plastic DIP
20 MHz	IS80C31-20PL	PLCC - Plastic Leaded Chip Carrier
	IS80C31-20PQ	PQFP - Plastic Quad Flat Pack
	IS80C31-20W	600-mil Plastic DIP
24 MHz	IS80C31-24PL	PLCC - Plastic Leaded Chip Carrier
	IS80C31-24PQ	PQFP - Plastic Quad Flat Pack
	IS80C31-24W	600-mil Plastic DIP
40 MHz	IS80C31-40PL	PLCC - Plastic Leaded Chip Carrier
	IS80C31-40PQ	PQFP - Plastic Quad Flat Pack
	IS80C31-40W	600-mil Plastic DIP

ISSI

Integrated Silicon Solution, Inc.

2231 Lawson Lane

Santa Clara, CA 95054

Fax: (408) 588-0806

Toll Free: 1-800-379-4774

<http://www.issiusa.com>**Integrated Silicon Solution, Inc.**

PRELIMINARY MCC03-16

6/10/97

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



TECHNICAL NOTE TN 6-6

Programming the FDC765A, FDC9266 and FDC9267

by David Cragg

Introduction

This technical note describes how to program the Standard Microsystems line of industry-standard floppy disk controllers based on the industry-standard FDC765A. Devices covered by this technical note include:

FDC765A	Industry-Standard Floppy Disk Controller
FDC9266	Enhanced Floppy Disk Controller and Data Separator
FDC9267	Enhanced Floppy Disk Controller and High-Performance Analog Data Separator

Because all of the above devices share a common command syntax, the system designer and software engineer are given the flexibility of selecting the level of performance and integration required.

This manual describes, in detail, each of the 15 commands the FDC is capable of executing. A simple flow chart details the steps a programmer must follow to insure reliable operation of the device. In addition, a narrative description of each command and a "command map" (similar to a register file) are included for each command.

The FDC765A and its derivatives all follow a consistent command syntax. Each command can be best thought of as having three distinct "phases". It is important to remember that each phase must be executed entirely. The three phases and their functions are listed below:

COMMAND PHASE Receive command byte and parameters from host microprocessor

EXECUTION PHASE FDC performs requested command

RESULT PHASE Host microprocessor **MUST** read out ALL status bytes from the FDC

The consistent structure of the commands allows system programmers to easily implement the FDC into a wide variety of popular operating systems, including UNIX, MS-DOS®, and CP/M™.

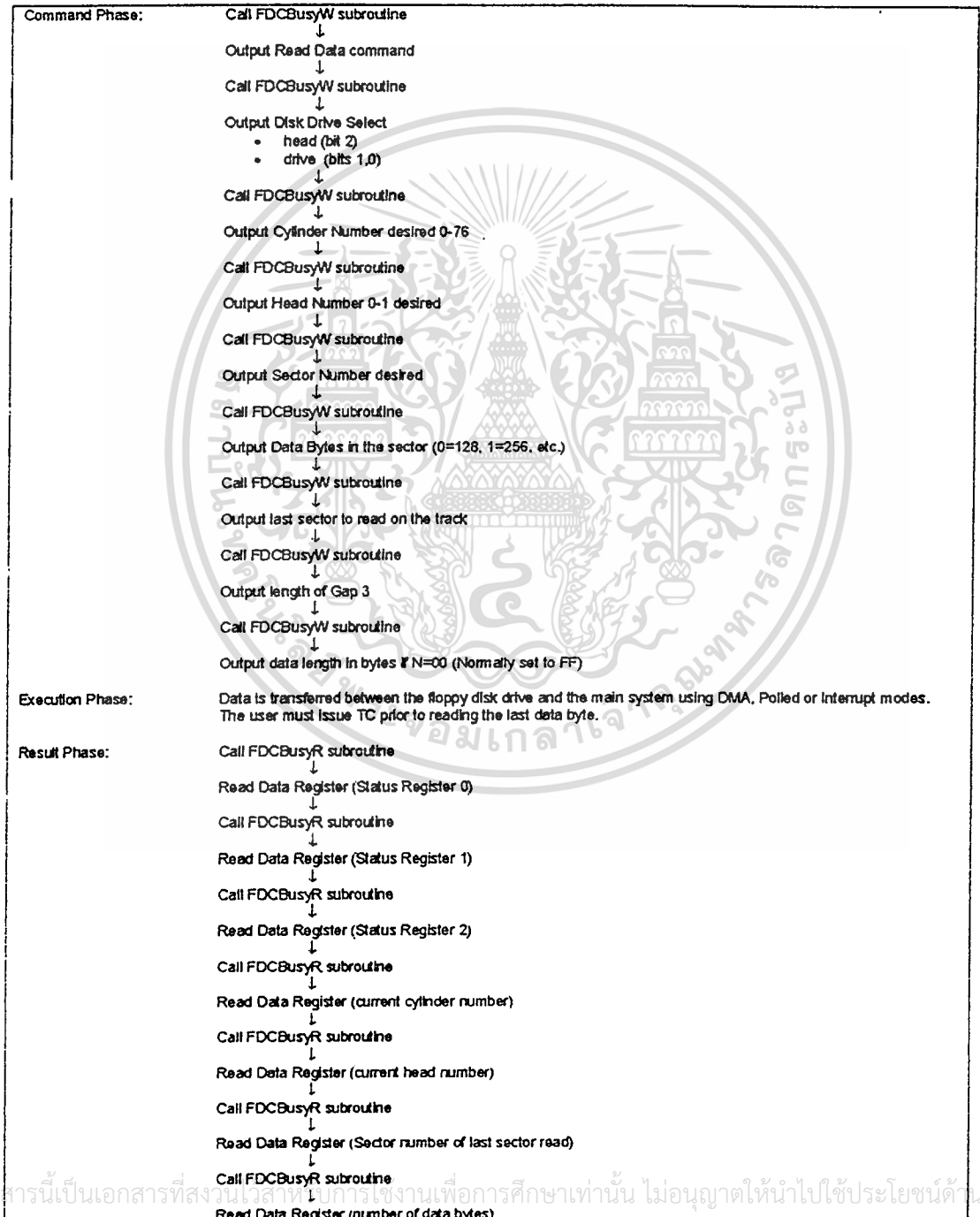
If you need additional information or technical assistance, please contact your local SMSC representative or field applications engineer.

Standard Microsystems Corporation
30 Arkay Drive
Hauppauge, NY 11788
516-433-6000
516-231-6004 FAX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ DATA

The FDC loads the head, waits the specified head setting time, and begins reading ID Address Marks and ID fields. When the current sector number ("R") stored in the internal ID Register (IDR) matches the sector number read off the diskette, the FDC outputs data (from the data field) byte-by-byte to the main system via the data bus.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ DATA

Command Phase: Read Data Word

MT	MF	SK	0	0	1	1	0
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Data length in bytes if N=00 (Normally set to FF)							(DTL)

Execution Phase: Data is transferred between the floppy disk drive and the main system using DMA, Polled or Interrupt modes. The user must issue TC prior to reading the last data byte.

Result Phase:

Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head Number)	(H)
Read Data Register (Sector Number)	(R)
Read Data Register (Number of data bytes written)	(N)

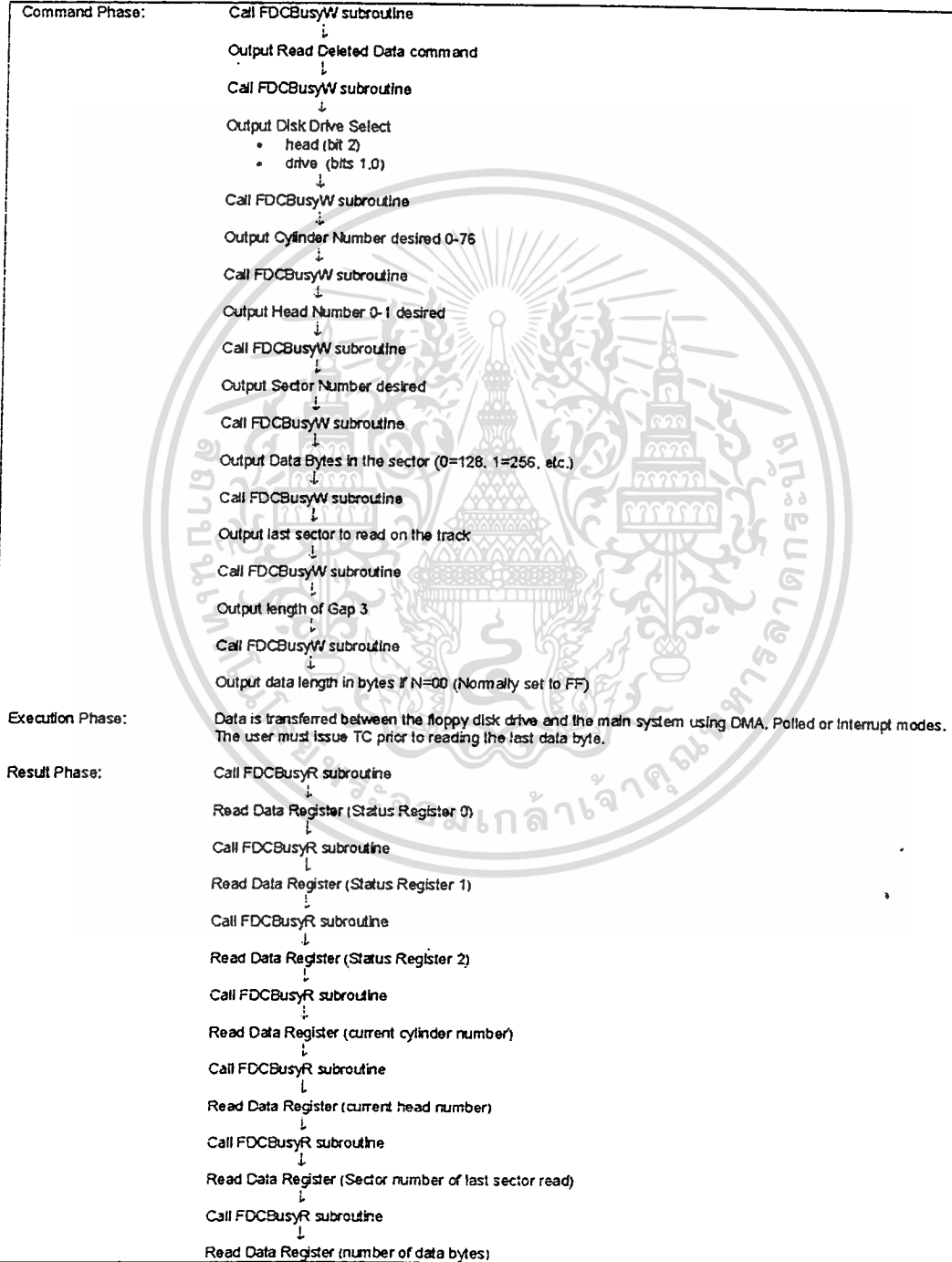
Command Word Abbreviations:

- MT Multi-Track. If MT is high, a multi-track operation is to be performed.
 MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
 SK Skip. SK stands for Skip Deleted Data Mark.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ DELETED DATA

This command is the same as the Read Data Command except that when the FDC detects a Data Address Mark at the beginning of a Data Field and SK=0 (low), it will read all the data in the sector and set the CM flag in Status Register 2 to a 1 (high) and then terminate the command. If SK=1, then the FDC skips the sector with the Data Address Mark and reads the next sector.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ DELETED DATA

Command Phase: Read Deleted Data Word

MT	MF	SK	0	1	1	0	0
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head Number 0 or 1 as specified in ID field							(H)
Sector Number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Data length in bytes if N=00 (Normally set to FF)							(DTL)

Execution Phase: Data is transferred between the floppy disk drive and the main system using DMA, Polled or Interrupt modes. The user must issue TC prior to reading the last data byte.

Result Phase:

Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head Number)	(H)
Read Data Register (Sector Number)	(R)
Read Data Register (Number of data bytes written)	(N)

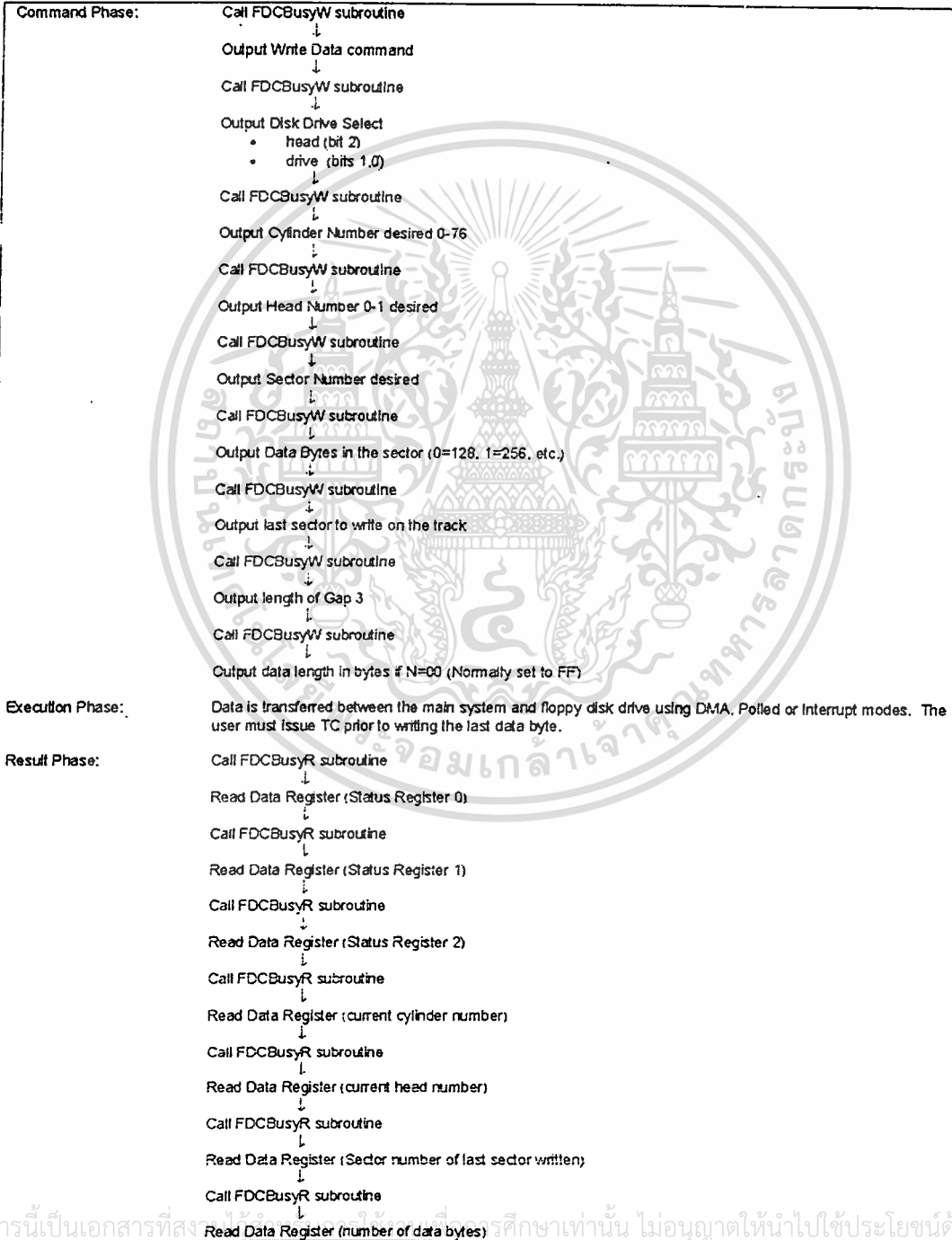
Command Word Abbreviations:

- MT Multi-Track. If MT is high, a multi-track operation is to be performed.
 MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
 SK Skip. SK stands for Skip Deleted Data Mark.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WRITE DATA

The FDC loads the head (if it is in the unloaded state), waits the specified Head Setting Time (defined in the Specify Command), and begins reading ID fields. When all four bytes loaded during the command (Cylinder Number, Head Address, Sector Number, Number of Data Bytes written in a sector) match the four bytes of the ID field from the diskette, the FDC takes data from the processor (or DMA controller) byte-by-byte via the data bus and outputs it to the Floppy Disk Drive.



เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WRITE DATA

Command Phase: Write Data Word

MT	MF	0	0	0	1	0	1
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Data length in bytes if N=00 (Normally set to FF)							(DTL)

Execution Phase: Data is transferred between the main system and floppy disk drive using DMA, Polled or Interrupt modes. The user must issue TC prior to writing the last data byte.

Result Phase:

Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

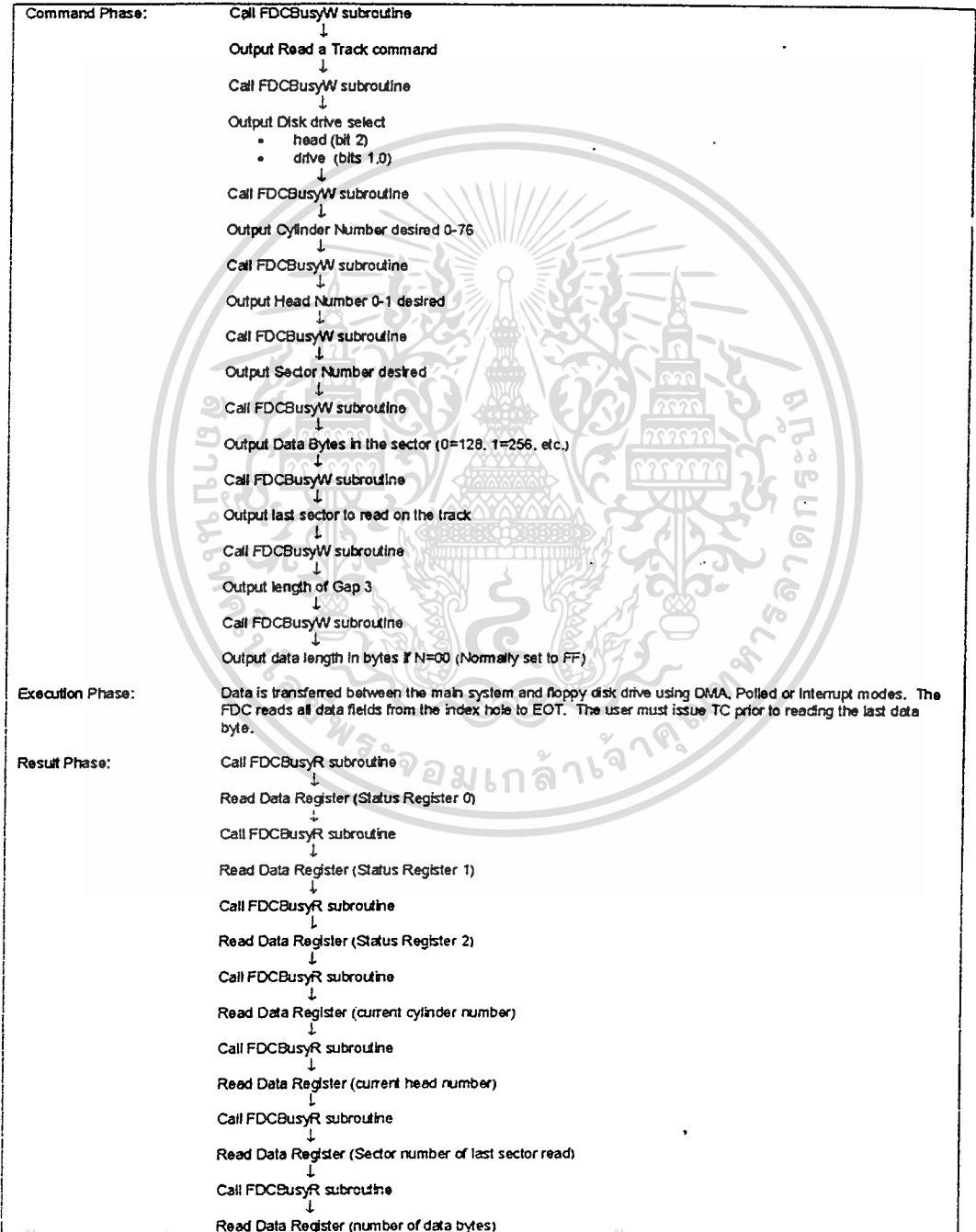
Command Word Abbreviations:

- MT Multi-Track. If MT is high, a multi-track operation is to be performed.
 MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
 SK Skip. SK stands for Skip Deleted Data Mark.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ A TRACK

This command allows the FDC to read physically consecutive sectors from the disk. Immediately after encountering the Index Hole, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or Data CRC check bytes, it continues to read data from the track. This command terminates when the number of sectors read equals EOT.



READ A TRACK

Command Phase: Read A Track Word

0	MF	SK	0	0	0	1	0
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Data length in bytes if N=00 (Normally set to FF)							(DTL)

Execution Phase: Data is transferred between the main system and floppy disk drive using DMA, Polled or Interrupt modes. The FDC reads all data fields from the index hole to EOT. The user must issue TC prior to reading the last data byte.

Result Phase:

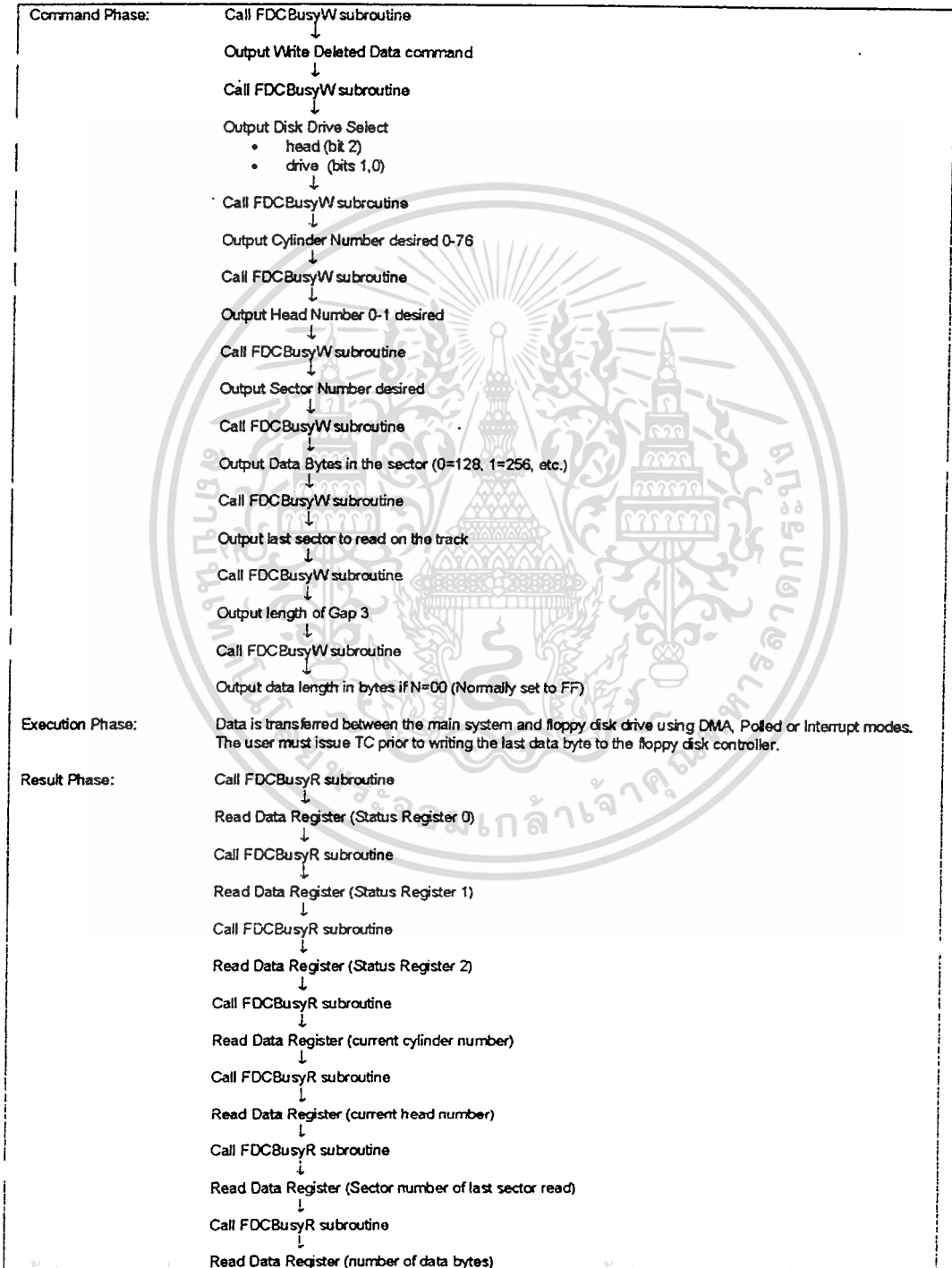
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviations:

- MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
- SK Skip Deleted Address Mark.

WRITE DELETED DATA

This command is the same as the Write Data Command except a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark.



WRITE DELETED DATA

Command Phase: Write Deleted Data Word

MT	MF	0	0	1	0	0	1
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Data length in bytes if N=00 (Normally set to FF)							(DTL)

Execution Phase: Data is transferred between the main system and floppy disk drive using DMA, Polled or Interrupt modes. The user must issue TC prior to writing the last data byte to the floppy disk controller.

Result Phase:

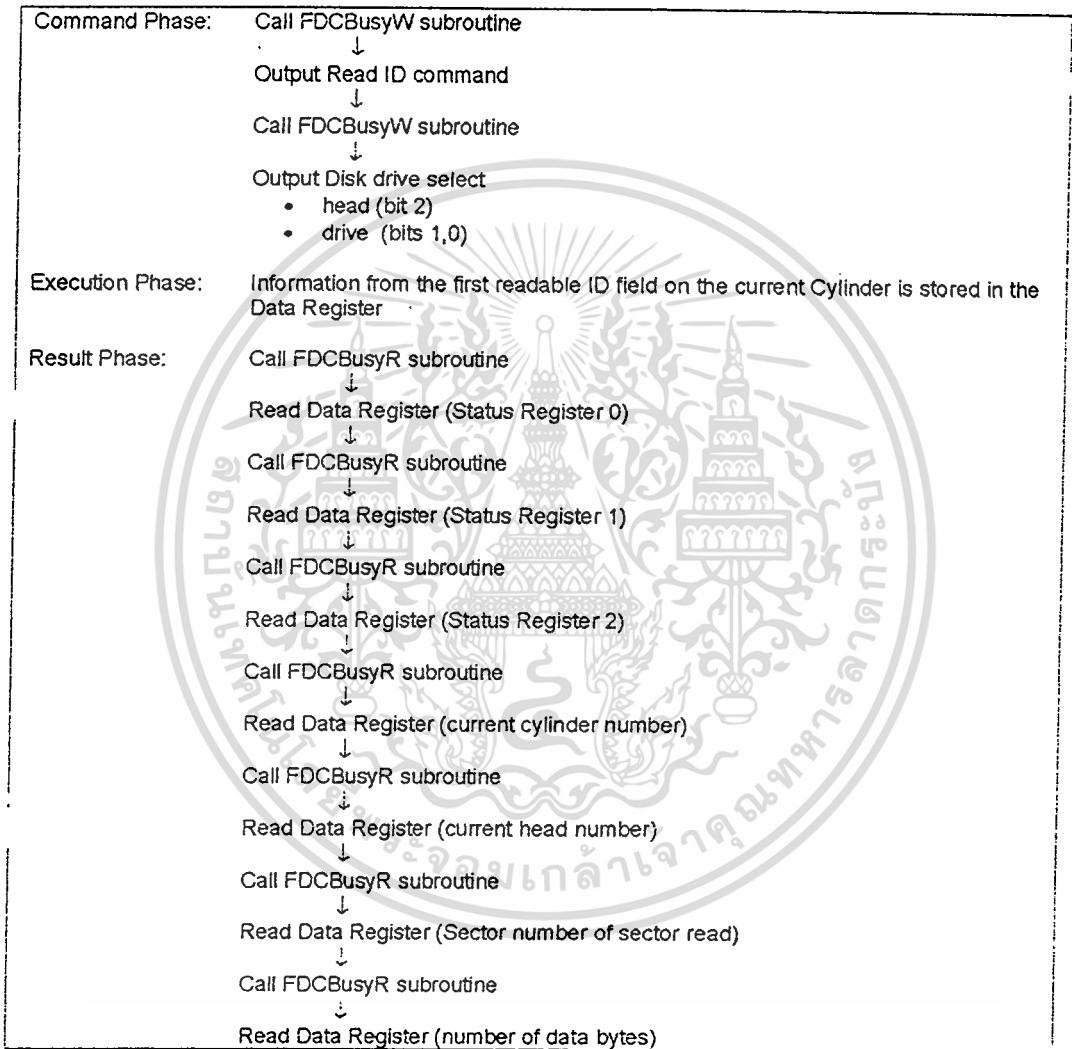
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviations:

MT Multi-Track. If MT is high, a multi-track operation is to be performed.
 MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.

READ ID

The Read ID command is used to give the present position of the selected drive's read/write head. The FDC stores the values from the first ID field it is able to read. During this command there is no data transferred between the FDC and the CPU except during the result phase.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

READ ID

Command Phase: Read ID Word

0	MF	0	0	1	0	1	0
X	X	X	X	X	Head	Drive Unit Select	

Execution Phase: The first correct ID information on the Cylinder is stored in the Data Register.

Result Phase:

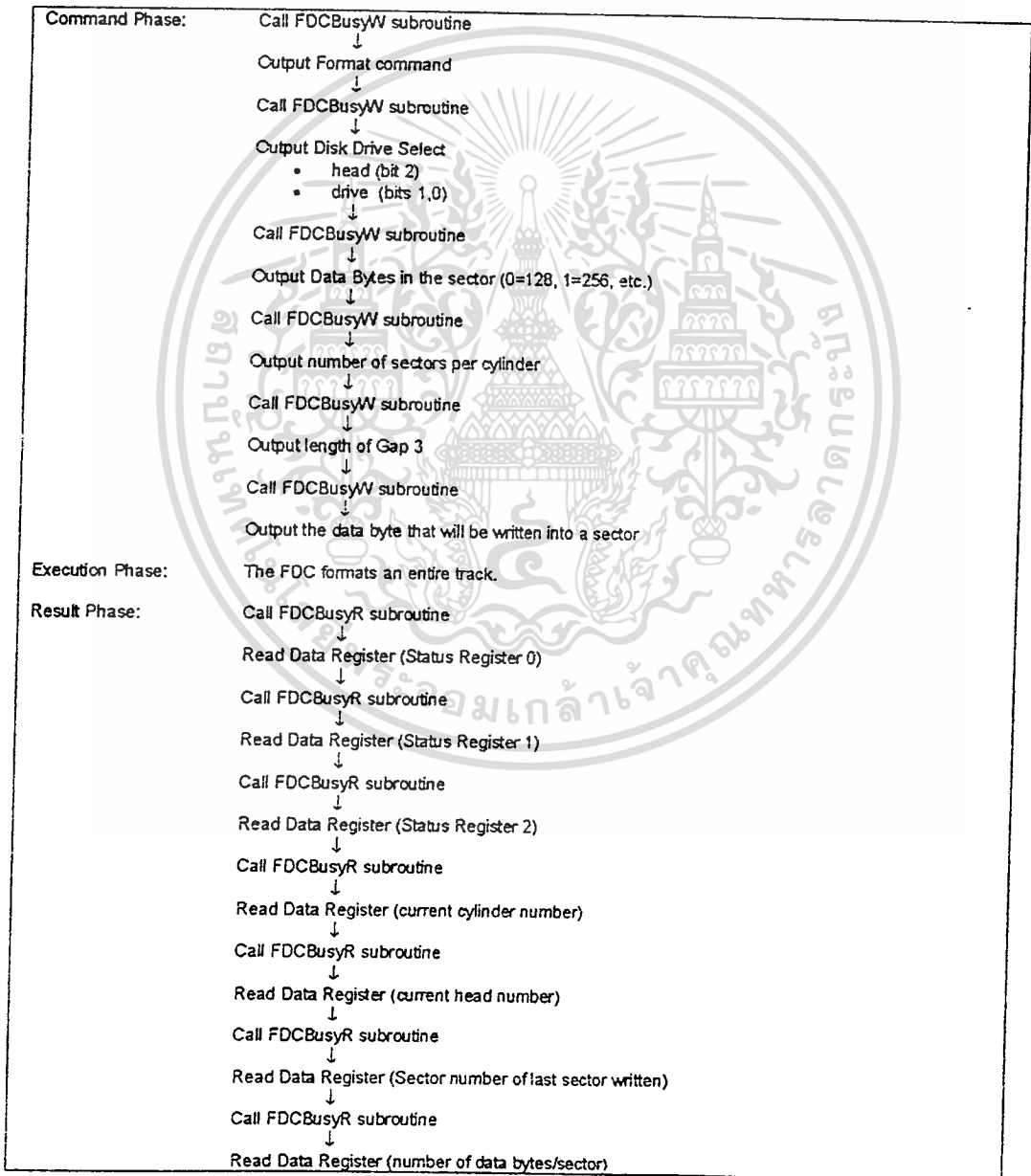
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviations:

MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.

FORMAT A TRACK

This command formats an entire track. After the Index Hole is detected, Gaps, Address Marks, ID fields and Data fields are written on the diskette. The particular format which will be written is controlled by the values programmed into N (number of bytes/sector), SC (sectors/cylinder), GPL (Gap Length), and D (Data Pattern) which are supplied by the processor during the Command Phase. The data field is filled with the byte of data stored in D. The FDC will request 4 bytes from the host for each sector formatted. This transfer may be done via DMA, Interrupt or Polled mode. It is important to activate TC prior to the transfer of the last byte for each sector. The information transferred (for each sector) includes Cylinder, Head, Sector and number of bytes per sector. This information is written to the ID field of the sector to be formatted.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FORMAT A TRACK

Command Phase: Format A Track Word

0	MF	0	0	1	1	0	1
X	X	X	X	X	Head	Drive Unit Select	
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The number of sectors per cylinder							(SC)
Gap 3 Length							(GPL)
Data Pattern to be written into sector							(D)

Execution Phase: FDC formats an entire track.

Result Phase:

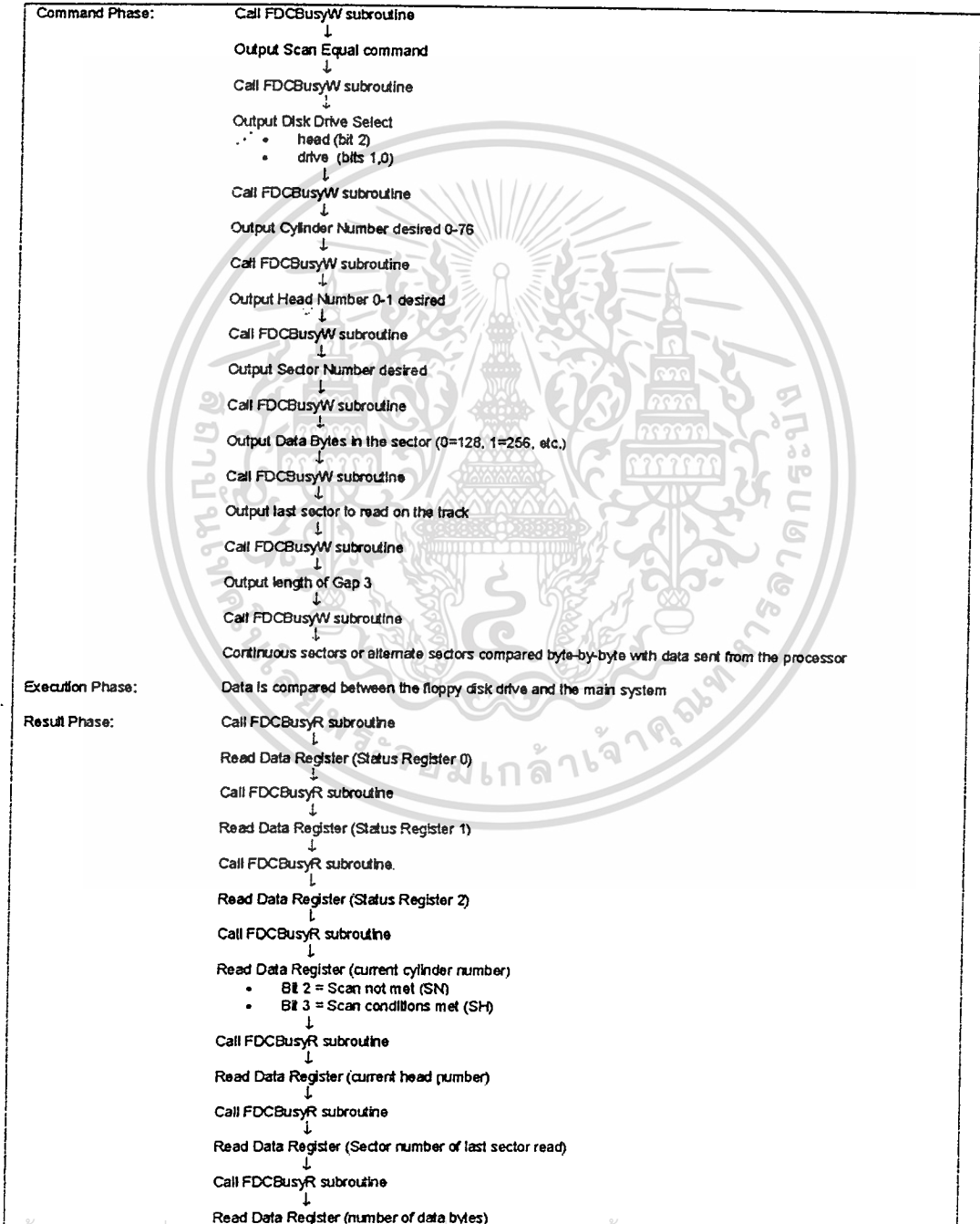
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviation:

MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.

SCAN EQUAL

This command allows data which is being read from the diskette to be compared against data which is being supplied from the main system. The FDC compares the data on a byte-by-byte basis and looks for a sector of data which meets the condition of $DATA_{\text{FDD}} = DATA_{\text{processor}}$. An FF_{HEX} from either the disk drive or system processor will always meet the conditions of a match.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SCAN EQUAL

Command Phase:

MT	MF	SK	1	0	0	0	1
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Contiguous sectors or alternate sectors compared							(STP)

Execution Phase: Data compared between the main system and the floppy disk drive

Result Phase:

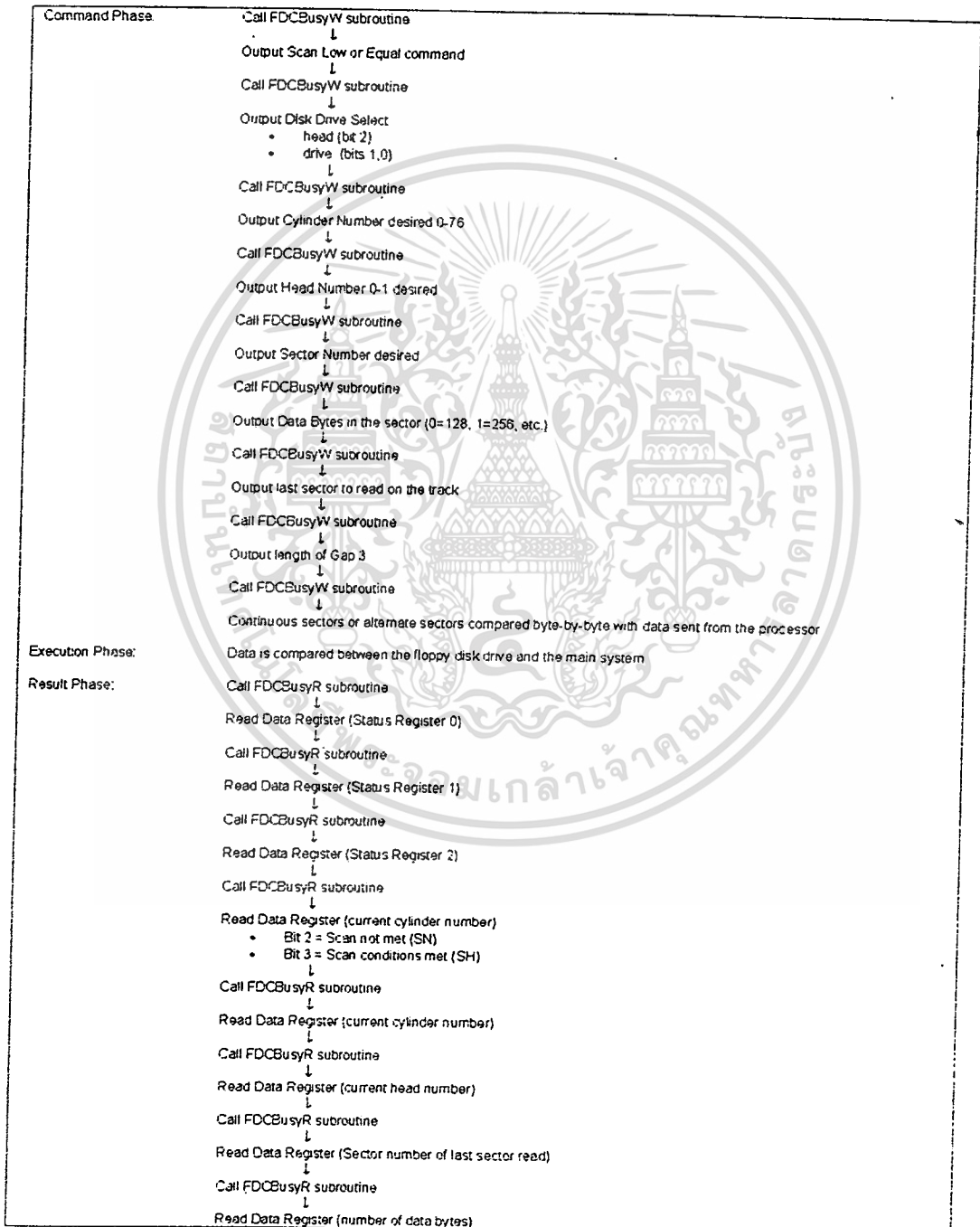
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviations:

- MT Multi-Track. If MT is high, both sides (of the current cylinder) will be read.
 MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
 SK Skip. SK stands for Skip Deleted Data Mark. If SK=0 and the FDC finds a sector with a deleted data mark, then the operation will stop after the current sector is read and the Control Mark flag (Bit 6, Status Register 2) will be set.

SCAN LOW OR EQUAL

This command allows data which is being read from the diskette to be compared against data which is being supplied from the main system. The FDC compares the data on a byte-by-byte basis and looks for a sector of data which meets the condition of $DATA_{\text{disk}} = DATA_{\text{processor}}$. An FF_{HEX} from either the disk drive or system processor will always meet the conditions of a search.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SCAN LOW OR EQUAL

Command Phase:

MT	MF	SK	1	1	0	0	1
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Contiguous sectors or alternate sectors compared							(STP)

Execution Phase: Data compared between the main system and the floppy disk drive

Result Phase:

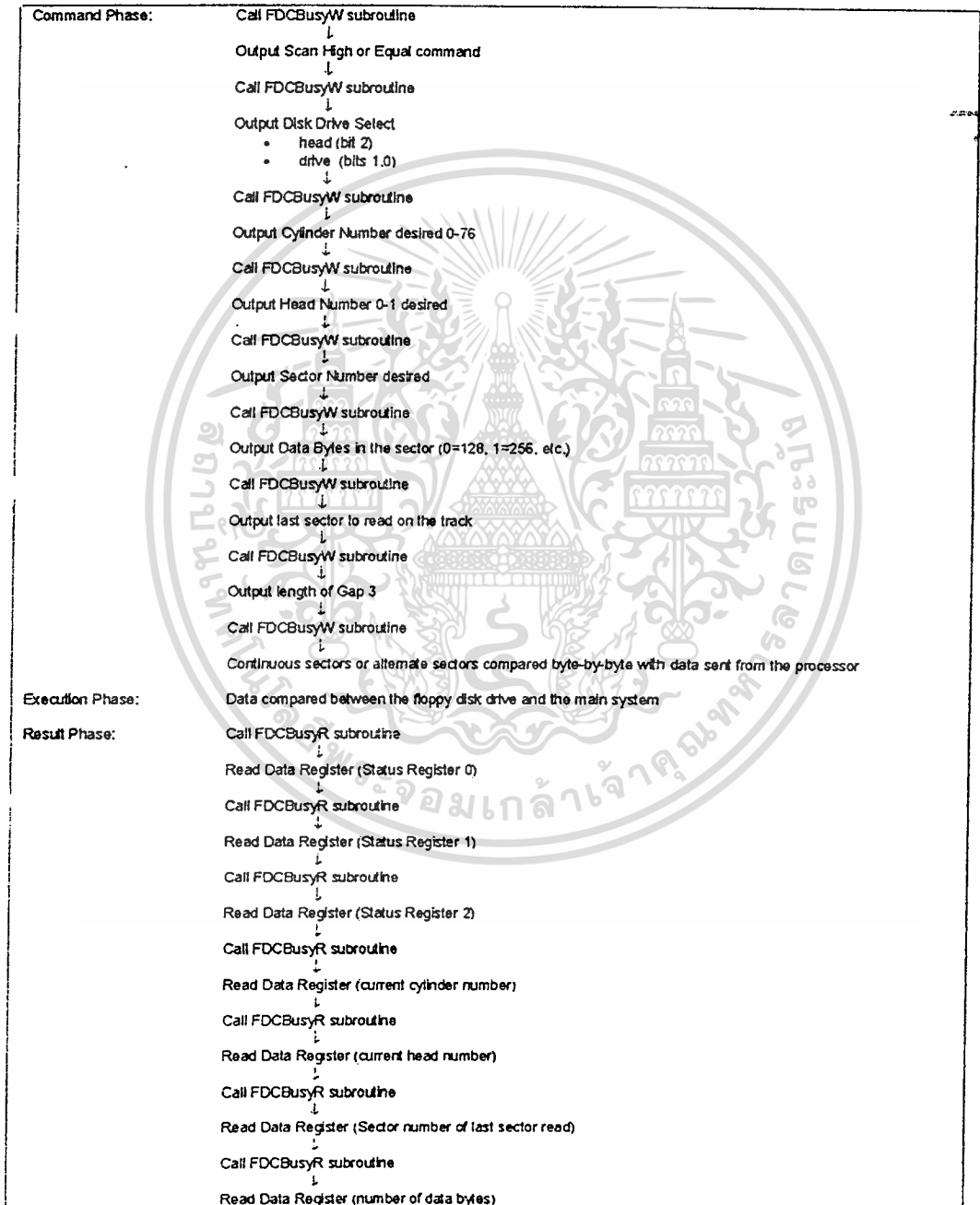
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

Command Word Abbreviations:

- MT Multi-Track. If MT is high, both sides (of the current cylinder) will be read.
- MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
- SK Skip. SK stands for Skip Deleted Data Mark. If SK=0 and the FDC finds a sector with a deleted data mark, then the operation will stop after the current sector is read and the Control Mark flag (Bit 6, Status Register 2) will be set.

SCAN HIGH OR EQUAL

This command allows data which is being read from the diskette to be compared against data which is being supplied from the main system. The FDC compares the data on a byte-by-byte basis and looks for a sector of data which meets the condition of $DATA_{add} = DATA_{processor}$. An FF_(HEX) from either the disk drive or system processor will always meet the conditions of a search.



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SCAN HIGH OR EQUAL

Command Phase:

MT	MF	SK	1	1	1	0	1
X	X	X	X	X	Head	Drive Unit Select	
Cylinder Number 0-76							(C)
Head number 0 or 1 as specified in ID field							(H)
Sector number							(R)
Number of data bytes in sector (0=128, 1=256, etc.)							(N)
The last sector number to be read							(EOT)
Gap 3 Length							(GPL)
Contiguous sectors or alternate sectors compared							(STP)

Execution Phase: Data compared between the main system and the floppy disk drive

Result Phase:

Read Data Register (Status Register 0)	(ST 0)
Read Data Register (Status Register 1)	(ST 1)
Read Data Register (Status Register 2)	(ST 2)
Read Data Register (New Cylinder Number)	(C)
Read Data Register (Head number)	(H)
Read Data Register (Sector number)	(R)
Read Data Register (Number of data bytes written)	(N)

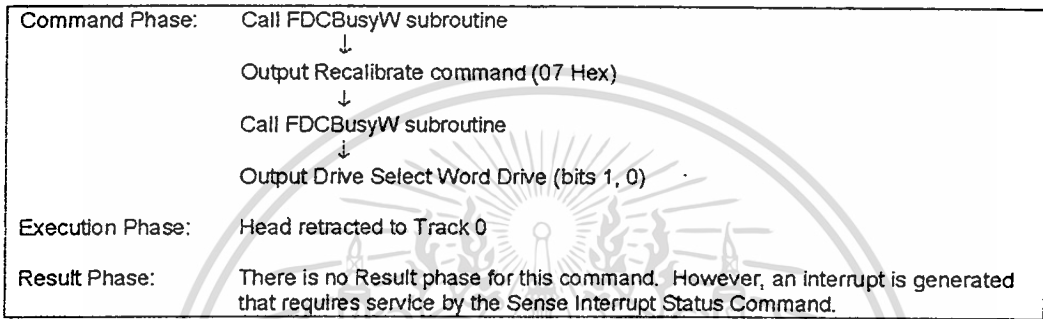
Command Word Abbreviations:

- MT Multi-Track. If MT is high, both sides (of the current cylinder) will be read.
- MF FM or MFM Mode. If MF is low, FM mode is selected, and if high MFM mode is selected.
- SK Skip. SK stands for Skip Deleted Data Mark. If SK=0 and the FDC finds a sector with a deleted data mark, then the operation will stop after the current sector is read and the Control Mark flag (Bit 6, Status Register 2) will be set.

RECALIBRATE (07 HEX)

This command attempts to retract the head to Track 00. The TR00 input is sampled. If TR00 is active low, indicating the Read/Write head is positioned over track 0, the command terminates. If TR00 is not active low, stepping pulses are issued until the TR00 input is activated. A maximum of 77 step pulses are issued. If TR00 is not true after 77 step pulses, the command terminates with the Seek End and Equipment Check flags (in status register 0) both set high.

If the drive selected has more than 77 tracks, it is usually easier to first Seek to Track 1, then issue the Recalibrate command. This avoids having to handle error conditions.

**RECALIBRATE**

Command Phase:

0	0	0	0	0	1	1	1
X	X	X	X	X	0	Drive Unit Select	

Execution Phase: Head retracted to Track 0.

Result Phase: None

SENSE INTERRUPT STATUS (08 HEX)

The Sense Interrupt command, when issued, resets the interrupt signal and by means of bits 5, 6 and 7 of Status Register 0 (returned during the result phase) identifies the cause of the interrupt. This command is used to identify the completion of Seek or Recalibrate commands and the ready signal from one of the disk drives changing state.

Command Phase:	Call FDCBusyW subroutine ↓ Output Sense Interrupt Status Command (08 Hex)
Execution Phase:	Resets the interrupt signal and via bits 5, 6 and 7 of Status Register 0 identifies the cause of the interrupt
Result Phase:	Call FDCBusyR subroutine ↓ Read Data Register (Status Register 0) bits 5, 6 and 7 to determine cause of interrupt ↓ Call FDCBusyR subroutine ↓ Read Data Register for the current cylinder address

SENSE INTERRUPT STATUS

Command Phase:

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

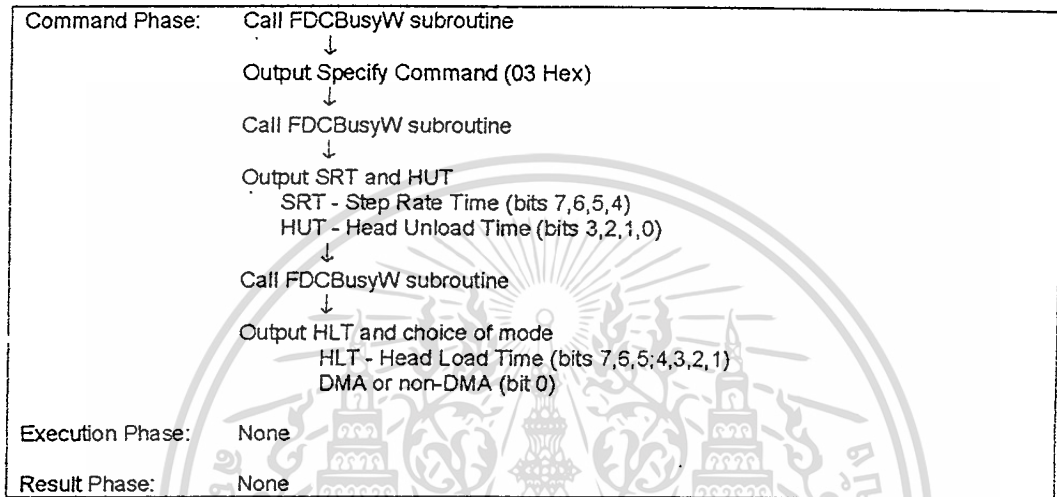
Execution Phase: Resets the interrupt signal and via bits 5, 6 and 7 of Status Register 0 identifies the cause of the interrupt

Result Phase:

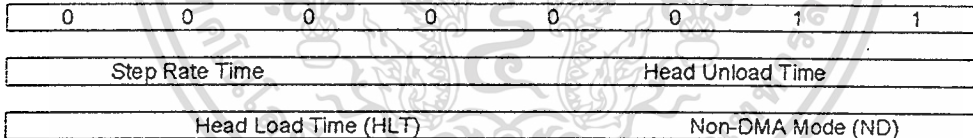
Read Data Register (Status Register 0)	(ST 0)
Read Data Register (current cylinder address)	(PCN)

SPECIFY (03 HEX)

The Specify command is used prior to performing any disk operation to define drive/FDC operating characteristics. The Specify command parameters set the values for the Head Load Time (HLT), Head Unload Time (HUT) and Step Rate Time (SRT). The Specify command also indicates the choice of DMA or non-DMA operation.

**SPECIFY**

Command Phase:



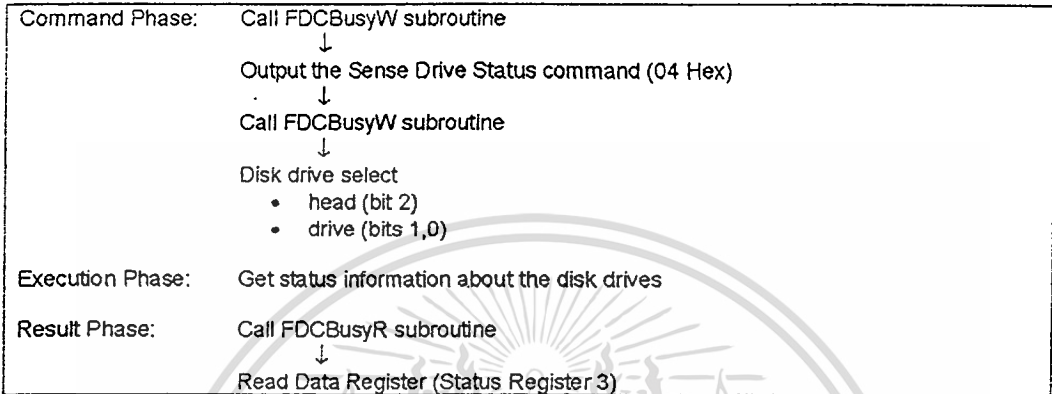
Execution Phase: None

Result Phase: None

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SENSE DRIVE STATUS (04 HEX)

This command is used by the processor whenever it wishes to obtain the status of the floppy disk drives. Status Register 3 contains the Drive Status information.

**SENSE DRIVE STATUS**

Command Phase:

0	0	0	0	0	1	0	0
X	X	X	X	X	Head Drive Unit Select		

Execution Phase: Get status information about the disk drives

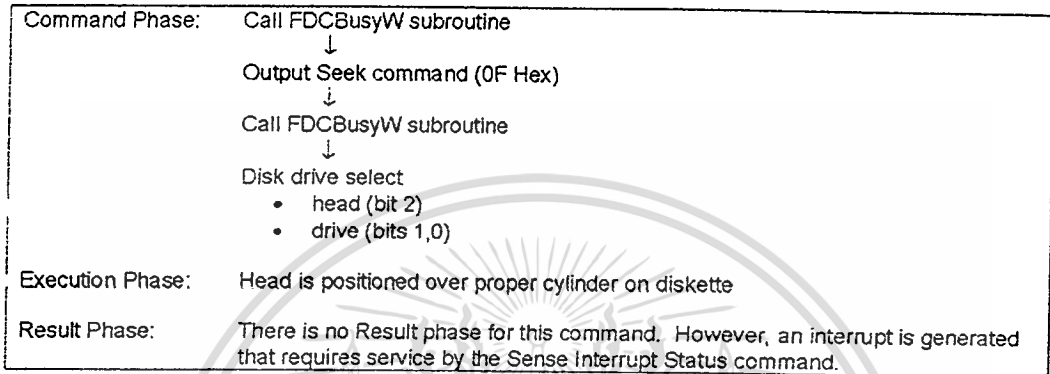
Result Phase:

Read Data Register (Status Register 3)	(ST 3)
--	--------

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SEEK (0F HEX)

The disk controller will update the contents of the internal Track Register (the current head location) and issue stepping pulses in the appropriate direction until the Track Register is equal to the Data Register (the desired track location).

**SEEK**

Command Phase:

0	0	0	0	1	1	1	1
X	X	X	X	X	Head Drive Unit Select		
New Cylinder Number							(C)

Execution Phase: Head is positioned over proper cylinder on diskette

Result Phase: None

FDCBusyW and FDCBusyR Subroutines

These subroutines wait a period of time (12 s or 24 s, depending on device mode) and check the main status register of the FDC, looking at the Request for Master bit (RQM, bit 7) and Direction Input/Output bit (DIO, bit 6). When these bits are properly set, the next byte may be output to or input from the FDC.

While it is important to wait the period of time specified, a timing loop in the subroutine is not always needed. In many instances, the time required to "push" the processor registers onto the stack is greater than the minimum period. Alternatively, the operating system may have a "timer" function call which may be used.

```

PROCEDURE FDCBusyW/FDCBusyR (Drive : Byte) ;
    {Drive=1 for 8" drive
     Drive=0 for 5.25" drive}

TYPE uSeconds = integer;

VAR   MainStatusReg : byte; {result of the Main Status Register}
      Mask           : byte; {used to mask out bits 5-0}
      Time           : uSeconds;

BEGIN
    Mask := 192;
    repeat
        for Time := 1 to (12 * Drive) do NoOp;           {NoOp = No Operation}
        read (MainStatusReg);                             {read Main Status Register}
        MainStatusReg := MainStatusReg AND Mask;          {Mask out bits 5-0}
    until MainStatusReg = 128;                             {Data I/O = 0
                                                            Request for Master = 1}

END.
```

RECOMMENDED FORMAT PARAMETERS

FORMAT	SECTOR SIZE	N	SC	GPL ¹	GPL ^{2,3}
8" Standard Floppy					
FM Mode	128 bytes/sector	00	1A	07	1B
	256	01	0F	0E	2A
	512	02	08	1B	3A
	1024	03	04	47	8A
	2048	04	02	C8	FF
	4096	05	01	C8	FF
MFM Mode ⁴	256	01	1A	0E	36
	512	02	0F	1B	54
	1024	03	08	35	74
	2048	04	04	99	FF
	4096	05	02	C8	FF
	8192	06	01	C8	FF
5 1/4" Minifloppy					
FM Mode	128 bytes/sector	00	12	07	09
	123	00	10	10	19
	256	01	08	18	30
	512	02	04	46	87
	1024	03	02	C8	FF
	2048	04	01	C8	FF
MFM Mode ⁴	256	01	12	0A	0C
	256	01	10	20	32
	512	02	08	2A	50
	1024	03	04	80	F0
	2048	04	02	C8	FF
	4096	05	01	C8	FF
3 1/2" Sony Micro Floppydisk					
FM Mode	128 bytes/sector	0	0F	07	1B
	256	1	09	0E	2A
	512	2	05	1B	3A
MFM Mode ⁴	256	1	0F	0E	36
	512	2	09	1B	54
	1024	3	05	35	74

Notes:

1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sections.
2. Suggested values of GPL in format command.
3. All values except sector size and hexadecimal.
4. In MFM mode FDC cannot perform a Read/Write/format operation with 128 bytes sector (N=00).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MAIN STATUS REGISTER BIT MAP

BIT NUMBER	NAME	SYMBOL	DESCRIPTION
DB ₀	FDD 0 Busy	D ₀ B	FDD number 0 is in the Seek mode. If any of the bits is set FDC will not accept read or write command.
DB ₁	FDD 1 Busy	D ₁ B	FDD number 1 is in the Seek mode. If any of the bits is set FDC will not accept read or write command.
DB ₂	FDD 2 Busy	D ₂ B	FDD number 2 is in the Seek Mode. If any of the bits is set FDC will not accept read or write command.
DB ₃	FDD 3 Busy	D ₃ B	FDD number 3 is in the Seek mode. If any of the bits is set FDC will not accept read or write command.
DB ₄	FDC Busy	CB	A read or write command is in process. FDC will not accept any other command.
DB ₅	Execution Mode	EXM	This bit is set only during execution phase in non-DMA mode. When DB ₅ goes low, execution phase has ended and result phase was started. It operates only during NON-DMA mode of operation.
DB ₆	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. IF DIO="1" then transfer is from Data Register to the Processor. If DIO="0" then transfer is from the Processor to Data Register.
DB ₇	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the Processor. Both bits DIO and RQM should be used to perform the hand-shaking functions of "ready" and "direction" to the processor.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คู่มือการใช้งาน

1. ป้อนตำแหน่งหน่วยความจำเริ่มต้นและสุดท้ายที่จะเขียนหรืออ่านดิสก์ ที่ตำแหน่ง INDEX ADDRESS ที่ซึ่งเกิดบอร์ค ตามด้วยชื่อไฟล์

เช่น ตำแหน่ง INDEX ADDRESS อยู่ที่ F010H ก็จะป้อน

<M>F010 : F7 00 F7 0F D I S K

2. ป้อนโปรแกรมและรันโปรแกรมที่ซินเกิลบอร์ค Z-80 ในการเขียนหรืออ่านดิสก์

LD	A, 80H	
OUT	(43H), A	
LD	A, 01H	; COMMAND ADDRESS START
OUT	(40H), A	
LD	A, 10H	; INDEX ADDRESS BYTE LOW
OUT	(41H), A	
LD	A, F0H	; INDEX ADDRESS BYTE HIGH
OUT	(42H), A	

รันโปรแกรม

3. ป้อนคำสั่งซึ่งจะถูกกำหนดจาก COMMAND โดยมีรูปแบบคำสั่งคือ

Command	ความหมาย
01H	WRITE
02H	READ

ข้อมูลจะถูกเขียนหรืออ่านไปยังดิสก์

บรรณานุกรม

یین ภู่วรรณ. เทคโนโลยีฮาร์ดแวร์ IBM PC. กรุงเทพฯ : เอช-เอน การพิมพ์. 2523

รัชชัย อินทุโส และไทรภพ อินทุโส. ไมโครคอนโทรลเลอร์. กรุงเทพฯ : ฟิสิกส์เซ็นเตอร์. 2523

วิบูลย์ ชื่นแขก. ไมโครโปรเซสเซอร์. พิมพ์ครั้งที่ 2. กรุงเทพฯ : สถาบันเทคโนโลยีพระจอมเกล้า
พระนครเหนือ. 2532

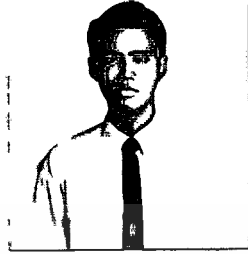
สมบัติ จุณณะปิยะ. การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51. กรุงเทพฯ :

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2541



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์

นายฐิติ พิกุลทอง

วันเดือนปีเกิด

23 มิถุนายน 2520

สถานที่เกิด

จังหวัดพังงา

ภูมิลำเนาเดิม

22/7 หมู่ 1 ต.บางม่วง อ.ตะกั่วป่า จ. พังงา 82110

ที่อยู่ปัจจุบัน

22/7 หมู่ 1 ต.บางม่วง อ.ตะกั่วป่า จ. พังงา 82110

โทรศัพท์

(076) 422024

ประวัติการศึกษา

ประถมศึกษา

โรงเรียนสุทธรินอนุสรณ์

มัธยมศึกษาตอนต้น

โรงเรียนตะกั่วป่า “เสนานุกูล”

ประกาศนียบัตรวิชาชีพ (ปวช.)

วิทยาลัยเทคนิคภูเก็ต

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

วิทยาลัยเทคนิคภูเก็ต

ปริญญาตรี

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

ผลงานที่ได้รับรางวัล

- รองชนะเลิศอันดับ 1 การแข่งขันเขียนโปรแกรมภาษาปาสคาล ระดับภาคใต้
- รางวัลนักศึกษากิจกรรมดีเด่น ปีการศึกษา 2539

ทุนการศึกษา

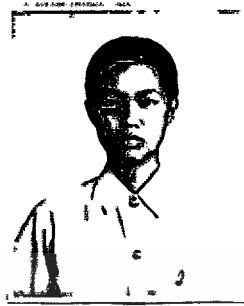
-

คติพจน์

อย่าหวังว่าต้นไม้ของตัวเองจะคงงามหากไม่รดน้ำให้มัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์

นางสาวนงคันธ์ สมุทรคนตรี

วันเดือนปีเกิด

31 พฤษภาคม 2520

สถานที่เกิด

จังหวัดลพบุรี

ภูมิลำเนาเดิม

54 หมู่ 1 ต.บางจา อ.ท่าม่วง จ.ลพบุรี 15150

ที่อยู่ปัจจุบัน

115 หมู่ 11 ต.ท่าม่วง อ.ท่าม่วง จ.ลพบุรี 15150

โทรศัพท์

(036) 622253

ประวัติการศึกษา

ประถมศึกษา

โรงเรียนวัดเกตุ

มัธยมศึกษาตอนต้น

โรงเรียนสิงห์บุรี

ประกาศนียบัตรวิชาชีพ (ปวช.)

วิทยาลัยเทคนิคลพบุรี

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

วิทยาลัยเทคนิคลพบุรี

ปริญญาตรี

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาวิศวกรรมโทรคมนาคม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่ได้รับรางวัล

-

ทุนการศึกษา

-

คติพจน์

แม้วันนี้ล้มลุกอย่าทุกข์ท้อ

จงก้าวต่อพอสร้างความหวังใหม่

แม้วันนี้ไม่มีสิทธิ์พิชิตชัย

วันหน้ายังมีใหม่ให้ท้าทาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์

นางสาวมาลินี คอกจันทร์

วันเดือนปีเกิด

7 กุมภาพันธ์ 2520

สถานที่เกิด

จังหวัดสระบุรี

ภูมิลำเนาเดิม

26 หมู่ 3 ต.ธารเกษม อ.พระพุทธบาท จ.สระบุรี 18120

ที่อยู่ปัจจุบัน

26 หมู่ 3 ต.ธารเกษม อ.พระพุทธบาท จ.สระบุรี 18120

โทรศัพท์

-

ประวัติการศึกษา

ประถมศึกษา

โรงเรียนประชาสงเคราะห์รังสรรค์

มัธยมศึกษาตอนต้น

โรงเรียนสุวีวิทยา

ประกาศนียบัตรวิชาชีพ (ปวช.)

วิทยาลัยเทคนิคพนบุรี

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

วิทยาลัยเทคนิคพนบุรี

ปริญญาตรี

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่ได้รับรางวัล

-

ทุนการศึกษา

-

คติพจน์

คิดก่อนทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์

นายวิเชียร แก้วณะศรี

วันเดือนปีเกิด

14 ธันวาคม 2519

สถานที่เกิด

จังหวัดเชียงใหม่

ภูมิลำเนาเดิม

65 หมู่ 9 บ้านป่าเป้า ต. มะเขือแจ้ อ. เมือง จ. ลำพูน
51000

ที่อยู่ปัจจุบัน

65 หมู่ 9 บ้านป่าเป้า ต. มะเขือแจ้ อ. เมือง จ. ลำพูน
51000

โทรศัพท์

(053) 503089, (01) 9553709

ประวัติการศึกษา

ประถมศึกษา

โรงเรียนมงคลวิทยา ลำพูน

มัธยมศึกษาตอนต้น

โรงเรียนมงคลวิทยา ลำพูน

ประกาศนียบัตรวิชาชีพ (ปวช.)

วิทยาลัยเทคนิคลำพูน

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

สถาบันเทคโนโลยีราชมงคลวิทยาเขตพายัพ เชียงใหม่

ปริญญาตรี

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาวิศวกรรมวิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่ได้รับรางวัล

- รางวัลชนะเลิศการแข่งขันทักษะทางวิชาการ ช่างอิเล็กทรอนิกส์ สถาบันเทคโนโลยีราชมงคล ระดับประเทศ
- รางวัลนักศึกษาผลการเรียนยอดเยี่ยม ปีการศึกษา 2539

ทุนการศึกษา

ทุนงบประมาณของสถาบัน

คติพจน์

ทำวันนี้ให้ดีที่สุด, อย่าเก็บความไม่รู้ไว้ในตัวเรา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาหรือข้อมูลใดๆ ของเอกสารนี้เพื่อการนำออกไปใช้