

การออกแบบและพัฒนาบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง
Design and Development of Microcontroller Board for Experiment



โดย

นาย คำเพ็ชร์ บุนนาค 42015722

นาย บุนนาค พิมสีประสม 42015726

เลขหมู่.....
เลขทะเบียน 42833
วัน, เดือน, ปี 10 ส.ย. 2545

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

การออกแบบและพัฒนาบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง
Design and Development of Microcontroller Board for Experiment

โดย

นาย คำเพ็ช บุนนະดี 42015722
นาย บุนอ้อม พิมสีประสม 42015726



อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง : การออกแบบและพัฒนาบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง

ผู้จัดทำ นาย คำเพ็ช นุนะดี 42015722

นาย นุนอ้อม พิมสีประสม 42015726



..... อาจารย์ที่ปรึกษา
(รศ. สมศักดิ์ มิตะถา)

การออกแบบและพัฒนาบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง

นาย คำเพ็ด บุนนะดี 42015722

นาย บุนอ้อม พิมพ์ประสม 42015726

รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา

ปีการศึกษา 2544

บทคัดย่อ

ในปัจจุบันได้มีบริษัทต่างๆที่พัฒนาและผลิตบอร์ดที่ใช้ในการทดลอง (Experiment board), ซึ่ง-
เกล็บอร์ด (Single Board) รุ่นต่างๆออกมาสู่ท้องตลาดมากมาย ซึ่งแต่ละรุ่น ของแต่ละบริษัทที่ผลิตออกมา
นั้น ต่างก็มีจุดเด่นจุดด้อยต่างกันไปและราคาก็ค่อนข้างสูง โครงการนี้เป็นโครงการที่เกิดขึ้นเนื่องจากว่า
มหาวิทยาลัยได้มองเห็นถึงความขาดแคลนทางด้านอุปกรณ์การทดลอง โดยเฉพาะอุปกรณ์การทดลองทาง
ด้านคอมพิวเตอร์ ที่มหาวิทยาลัยแห่งชาติลาว (National University Of Laos) โดยโครงการนี้ส่วนหนึ่งได้
คัดเลือกเอาจุดเด่นของหลายๆบอร์ดเข้ามารวมกัน เพื่อให้ได้บอร์ดตามที่ต้องการ และเหมาะสมกับการใช้
ในการทดลอง ซึ่งโครงการนี้มีจุดประสงค์สำคัญคือ สร้างบอร์ดเพื่อเป็นต้นแบบของบอร์ดทดลองที่จะ
สร้างให้กับมหาวิทยาลัยแห่งชาติลาว (NUOL) เพื่อให้มหาวิทยาลัยแห่งชาติลาว มีบอร์ดทดลองที่ใช้เข้าใน
การเรียนการสอนและลดต้นทุนในการซื้อบอร์ดทดลองอื่นๆ

ทั้งเพื่อให้บอร์ดที่สร้างขึ้นมานำไปใช้ประกอบการสอนได้และเพื่อเป็นการตรวจสอบ
การทำงานของบอร์ดทั้งหมด จึงได้จัดทำใบงานทดลองขึ้นมาเพื่อตรวจสอบการทำงานทั้งด้าน ฮาร์ดแวร์,
ซอฟต์แวร์ และเพื่อให้ผู้ใช้สามารถเรียนรู้การทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 และ
สามารถที่จะนำไปประยุกต์ใช้งานได้จริง

Design and Development of Micro controller Board for Experiment

Khampheth Bounnady

Boun-om Phimsipasom

Somsak Mittatha Advisor

2001

Abstract

At the present time, There are a lot of firms which develop many generation of Experiment board, single board. However each generations of each companys have different strong point and weak point and price is expensive. This project is generated because we knew and we found at National University Of Laos(NUOL) have not had board for experiment in the computer branch. Some part of this project is glean a strong point of another board for becoming for experiment user and save money for buy board.

Purpose of this project is to design and implement prototype board for NUOL. The main purpose of this board use for teaching and studying in NUOL. The result is NUOL can save budget for buying experiment and development board. For efficient use and test in all functions of our board, we wrote experiment direction for guide user how to test hardware and software, addition understand MCS51 microcontroller working mechanic and can apply for their work.

กิตติกรรมประกาศ

การที่โครงการนี้สำเร็จลุล่วงไปได้ด้วยดีนั้น เป็นผลสืบเนื่องมาจากความรู้พื้นฐานหลาย ๆ แขนง ที่ได้เรียนมา คำแนะนำต่าง ๆ ที่ได้รับจากอาจารย์ในภาควิชาคอมพิวเตอร์ทุก ๆ ท่าน โดยเฉพาะอาจารย์ที่ปรึกษา คือ รศ. สมศักดิ์ มิตะธา ที่ได้เอาใจใส่และทุ่มเทในโครงการนี้ คอยให้คำปรึกษาด้วยดีมาตลอด และคอยชี้แนะสิ่งที่เป็นประโยชน์ต่อโครงการนี้เป็นอย่างมาก และขอขอบคุณเพื่อน ๆ พี่ทุกคน โดยเฉพาะพี่เจริญ วงษ์ห่มเย็น และ พี่ นเรศ มาลัย ที่ได้ช่วยเหลือแนะนำสิ่งต่างๆ ที่ติดขัด และคอยให้กำลังใจจนงานสำเร็จลุล่วงไปได้

และขอขอบพระคุณองค์การ Japan International Cooperation Agency (JICA) ซึ่งเป็นผู้ที่ให้ทุนการศึกษาในโครงการนี้ เป็นผู้ประสานงานในโครงการร่วมมือสามฝ่ายคือ National University of Laos, King Mongkut's Institute of Technology Ladkrabang และ Japan International Cooperation Agency ขอขอบพระคุณอาจารย์ผู้ควบคุมสอบทุกท่านที่ได้สละเวลาอันเป็นประโยชน์ของท่านมาในการสอบโปรเจกต์ให้ผ่านลุล่วงไปด้วยดีครั้งนี้



สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	X
สารบัญรูปภาพ	VII
บทที่ 1 บทนำ	1
1.1 จุดประสงค์ของโครงการ	1
1.2 ขอบเขตของโครงการ	
1.3 ขั้นตอนการดำเนินงาน	2
1.4 รายละเอียดของโครงการ	2
บทที่ 2 ไมโครคอนโทรลเลอร์ MCS-51	3
2.1 ความหมายของไมโครคอนโทรลเลอร์	3
2.2 ความแตกต่างของไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์	3
2.3 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ตระกูล MCS-51	6
2.4 โครงสร้างภายในของ 8051	6
2.5 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51	10
2.6 ฐานเวลาในการทำงานของซีพียูภายใน 8051	14
2.7 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ของ 8051	15
2.7.1 หน่วยความจำโปรแกรม (Code Memory)	15
2.7.2 หน่วยความจำโปรแกรมภายใน (Internal Code Memory)	16
2.7.3 หน่วยความจำโปรแกรมภายนอก (External Code Memory)	16
2.8 หน่วยความจำข้อมูล (Data Memory)	17
2.8.1 หน่วยความจำข้อมูลภายใน (Internal Data Memory)	17
2.8.2 หน่วยความจำขนาด 128 ไบต์แรก	18
2.8.3 หน่วยความจำขนาด 128 ไบต์ถัดไป	20
2.8.4 รีจิสเตอร์หน้าที่พิเศษ (Special Function Register)	20
2.8.5 หน่วยความจำข้อมูลภายนอก (External Data Memory)	21
2.9 การอินเตอร์รัปต์ (Interrupt)	22
2.9.1 ประเภทของการอินเตอร์รัปต์ (Interrupt Type)	23
2.9.2 สัญญาณอินเตอร์รัปต์จากภายนอก (External Interrupt)	23

สารบัญ (ต่อ)

	หน้า
2.9.3 สัญญาณอินเทอร์รัปต์ภายใน (Internal Interrupt)	24
2.9.4 การควบคุมอินเทอร์รัปต์ (Interrupt Control)	24
2.9.5 อินเทอร์รัปต์ไพริอิตี รีจิสเตอร์ (Interrupt Priority Register)	25
2.9.6 การจัดการอินเทอร์รัปต์	25
2.9.7 สัญญาณอินเทอร์รัปต์จากภายนอก (External Interrupt)	
2.9.8 วงจรรีเซ็ต/จับเวลา	28
2.9.9 การอินเทอร์รัปต์ของวงจรรีเซ็ต/จับเวลา (Timer/Counter Interrupt)	29
2.10 การทำงานเป็นตัวจับเวลา (Timer)	31
2.10.1 การจับเวลาในโหมด 0	31
2.10.2 การจับเวลาในโหมด 1	32
2.10.3 การจับเวลาในโหมด 2	32
2.10.4 การจับเวลาในโหมด 3	33
2.11 การอินเทอร์รัปต์จากการสื่อสารข้อมูลอนุกรม	34
2.11.1 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม	35
2.11.2 รูปแบบของข้อมูลอนุกรม	36
2.12 ชุดคำสั่งของ MCS-51	36
2.12.1 รูปแบบการอ้างอิงตำแหน่งแอดเดรสของ 8051	37
บทที่ 3 การออกแบบฮาร์ดแวร์ (Hardware)	38
3.1 บอร์ดไมโครคอนโทรลเลอร์ 8051 (Microcontroller Board 8051)	38
3.1.1 โมดูลหน่วยประมวลผลกลาง (CPU Module)	39
3.1.1.1 วงจรที่สร้างสัญญาณนาฬิกาให้แก่อุปกรณ์	39
3.1.1.2 วงจรรีเซ็ต Reset เป็นวงจรที่ต่อให้กับซีพียู	39
3.1.2 โมดูลหน่วยความจำ (Memory Module)	40
3.1.3 โมดูลพอร์ต (Port Module)	41
3.1.3.1 พอร์ตใช้งาน 8255 (User port)	41
3.1.3.2 พอร์ตของระบบ (System port)	45
3.1.3.3 โมดูลพอร์ตสื่อสารข้อมูลอนุกรม (Serial Port)	46
3.1.4 โมดูลนับจับเวลา (Module Timer/Counter)	48
3.1.5 โมดูลสวิตช์อินเทอร์รัปต์ (Switch Interrupt Module)	49
3.2 บอร์ดทดลองอินพุตเอาต์พุต (Experiment input output board)	50
3.2.1 โมดูลที่เป็นอินพุตประกอบด้วยโมดูลต่างๆดังนี้	52

สารบัญ (ต่อ)

หน้า

3.2.1.1	โมดูล คีย์บอร์ด (Key board Module)	52
3.2.1.2	โมดูลคิพสวิตช์ (Dip Switch Module)	53
3.2.1.3	โมดูลการแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Module ADC)	54
3.2.1.4	ชุดตรวจวัดอุณหภูมิ (TEMPERATURE SENSOR)	56
3.2.1.5	โมดูลวงจรตรวจจับความเร็ว (SPEED SENSOR CIRCUIT MODULE)	57
3.2.1.6	โมดูลชุดควบคุมความร้อน (Heater Control Module)	58
3.2.2	โมดูลที่เป็นเอาต์พุต	59
3.2.2.1	โมดูล Port monitor	59
3.2.2.2	โมดูลชุดแสดงผลแบบ LED 7 ส่วน (7 Segment LED Display Module)	59
3.2.2.3	โมดูลแสดงผลแบบผลึกเหลว (Liquid Crystal Display Module)	62
3.2.2.4	โมดูลบี๊เซอร์ (Buzzer)	66
3.2.2.5	โมดูล Magnetic Relay	67
3.2.2.6	โมดูลโซลิดสเตตรีเลย์ (Solid State Relay)	69
3.2.2.7	โมดูลการแปลงสัญญาณดิจิตอลเป็นอนาล็อก (Digital to Analog Convertor)	70
3.2.2.8	โมดูล สเต็ปปีงมอเตอร์ (STEPPING MOTOR)	72
3.2.2.9	โมดูลการควบคุมมอเตอร์ (Motor control Module)	74
3.2.2.10	ชุดแสดงผลข้อความ (DOTMATRIX DISPLAY)	75
บทที่ 4	การใช้งานซอฟต์แวร์ (Software)	77
4.1	ขั้นตอนการใช้งาน	77
4.2	รายละเอียดทั่วไปของ SXA51 ASSEMBLE	80
4.2.1	การใช้งาน SXA51 ASSEMBLER	80
4.2.2	คำแนะนำในการใช้ SXA51	85
4.3	การใช้งานบอร์ดทดลองอินพุตเอาต์พุต	86
4.3.1	การใช้งาน โมดูลที่เป็นอินพุตและเอาต์พุต (Input/Output Module)	86
4.3.2	โมดูลพอร์ตมอนิเตอร์ (Port Monitor Module)	88
4.3.3	โมดูลชุดแสดงผลแบบ LED 7 ส่วน (7 Segment Display Module)	89

สารบัญ (ต่อ)

		หน้า
	4.3.4 โมดูลแสดงผลแบบผลึกเหลว(Liquid Crystal Display Module)	90
	4.3.5 โมดูลบี๊เซอร์ (Buzzer Module)	92
	4.3.6 โมดูลแมกเนติกรีเลย์ (Magnetic Relay Module)	93
	4.3.7 โมดูลโซลิดสเตตรีเลย์ (Solid State Relay Module)	93
	4.3.8 โมดูลการแปลงสัญญาณดิจิตอลเป็นอนาล็อก (DAC Module)	94
	4.3.9 โมดูลสเต็ปปีงมอเตอร์ (Stepping Motor Module)	95
	4.3.10 โมดูลการควบคุมมอเตอร์ (Motor control Module)	96
	4.3.11 ชุดแสดงผลข้อความ (Dot Matrix Display)	97
	4.4 โมดูลที่เป็นอินพุต (Input Modules)	98
	4.4.1 โมดูลคีย์บอร์ด (Key board Module)	98
	4.4.2 โมดูลดิพสวิตช์ (Dip Switch Module)	99
	4.4.3 โมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC Module)	100
	4.4.4 โมดูลชุดตรวจวัดอุณหภูมิ (Temperature Sensor)	101
	4.4.5 โมดูลวงจรตรวจจับความเร็ว (Speed Sensor Module)	103
	4.4.6 โมดูลชุดควบคุมความร้อน (Heater Control Module)	104
บทที่ 5	การทดสอบการทำงานของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง	105
5.1	บอร์ดไมโครคอนโทรลเลอร์ 8051	105
	5.1.1 ทดสอบโมดูล CPU	105
	5.1.2 ทดสอบโมดูลหน่วยความจำ	105
	5.1.3 ทดสอบโมดูลอินเตอร์รัปต์ (Interrupt Module Test)	106
	5.1.4 ทดสอบโมดูลวงจรมับ/จับเวลา (Timer/Counter Test)	106
	5.1.5 ทดสอบพอร์ตใช้งานทั่วไป 8255	107
	5.1.6 ทดสอบโมดูลพอร์ตอนุกรม (Serial Port RS232 Test)	107
	5.1.7 ทดสอบการ Interface ของบอร์ดไมโครคอนโทรลเลอร์ 8051	107
5.2	การทดสอบบอร์ดทดลองอินพุต/เอาต์พุต	108
	5.2.1 ทดสอบโมดูลพอร์ตคอมอนิเตอร์ (Port Monitor)	108
	5.2.2 ทดสอบโมดูลชุดแสดงผลแบบ LED 7 ส่วน (7 Segment Display Module)	108
	5.2.3 ทดสอบโมดูลสเต็ปปีงมอเตอร์ (Stepping Motor Module)	109
	5.2.4 ทดสอบชุดควบคุมความร้อน (Heater Control Module)	109

สารบัญ (ต่อ)

หน้า

5.2.5	ทดสอบโมดูลการควบคุมมอเตอร์ (Motor Control Module)	110
5.2.6	ทดสอบโมดูลบัสเซอร์ (Buzzer Module)	110
5.2.7	ทดสอบโมดูลแสดงผลแบบผลึกเหลว(LCD)	110
5.2.8	ทดสอบโมดูลแสดงผลข้อความ (Dot Matrix Display)	111
5.2.9	ทดสอบโมดูลโซลิดสเตตรีเลย์ (Solid State Relay Module)	129
5.2.10	ทดสอบโมดูลแม่เหล็กไฟฟ้ารีเลย์ (Magnetic Relay Module)	112
5.2.11	ทดสอบโมดูลคิพสวิตช์ (Dip Switch Module)	112
5.2.12	ทดสอบโมดูล คีย์บอร์ด(Key board Module)	112
5.2.13	ทดสอบโมดูลการแปลงสัญญาณดิจิตอลเป็นอนาล็อก (DAC Module)	113
5.2.14	ทดสอบโมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC Module)	113
5.2.15	ทดสอบโมดูลวงจรตรวจจับความเร็ว (Speed Sensor Module)	113
5.2.16	ทดสอบโมดูลชุดตรวจวัดอุณหภูมิ (Temperature Sensor Module)	114
5.2.17	การทดสอบการ Interface ของบอร์ดทดลองอินพุตเอาต์พุต	114
บทที่ 6	ใบงานการทดลอง	115
6.1	การทดลองที่ 1 การโปรแกรม 8255 และการแสดงผลข้อมูลขนาด 8 บิต	115
6.1.1	การโปรแกรม 8255 (Programmable peripheral interface)	115
6.1.2	การแสดงผลข้อมูลขนาด 8 บิต	117
6.2	การทดลองที่2 การอ่านข้อมูลขนาด 8 บิตจากสวิตช์อินพุต และการควบคุม BUZZER	122
6.2.1	การอ่านข้อมูลขนาด 8 บิต	122
6.2.2	การทดลองเกี่ยวกับ Buzzer	123
6.3	การทดลองที่3 การควบคุมการทำงานของ Magnetic Relay และ Solid State Relay	125
6.3.1	การควบคุมการทำงานของ Magnetic Relay	125
6.3.2	การควบคุมการทำงานของ Solid State Relay	127
6.4	การทดลองที่ 4 การแสดงผลข้อมูล 7 ส่วน (7 Segment display)	129
6.5	การทดลองที่ 5 การรับข้อมูลจากคีย์บอร์ด (Scan Key board)	132
6.6	การทดลองที่ 6 การแสดงผลข้อมูลด้วยคอคอดเมตริก (Dot matrix display)	135
6.7	การทดลองที่ 7 การแสดงผลข้อมูลด้วยตัวแสดงผลแบบผลึกเหลว (LCD)	138

สารบัญ (ต่อ)

หน้า

6.8	การทดลองที่ 8 การควบคุมการทำงานของมอเตอร์ (Motor control) และ ของสเต็ปป์มอเตอร์ (Stepping Motor)	144
6.8.1	การควบคุมการทำงานของมอเตอร์	144
6.8.2	การควบคุมการทำงานของสเต็ปป์มอเตอร์ (Stepping Motor)	145
6.9	การทดลองที่ 9 การแปลงสัญญาณดิจิทัลเป็นสัญญาณอนาล็อก	148
6.10	การทดลองที่ 10 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล	151
6.11	การทดลองที่ 11 การควบคุมการทำงานของตัวทำความร้อน(HEATER) และ การควบคุมการทำงานของอุปกรณ์ตรวจวัดอุณหภูมิ(Temperature Sensor)	154
6.11.1	การควบคุมการทำงานของตัวทำความร้อน(HEATER)	154
6.11.2	การควบคุมการทำงานของอุปกรณ์ตรวจวัดอุณหภูมิ(Temperature Sensor)	156
6.12	การทดลองที่ 12 การวัดค่าความเร็วของมอเตอร์ (Speed Sensor)	158
	คำตอบของการทดลอง	160
6.13	การทดลองที่ 1 การโปรแกรม 8255 และการแสดงผลข้อมูลขนาด 8 บิต	160
6.13.1	การแสดงผลข้อมูลขนาด 8 บิต	160
6.14	การทดลองที่ 2 การอ่านข้อมูลขนาด 8 บิตจากสวิตช์อินพุต และการควบคุม BUZZER	164
6.14.1	การอ่านข้อมูลขนาด 8 บิต	164
6.14.2	การทดลองเกี่ยวกับ Buzzer	167
6.15	การทดลองที่ 3 การควบคุมการทำงานของ Magnetic Relay และ Solid State Relay	169
6.15.1	การควบคุมการทำงานของ Magnetic Relay	169
6.15.2	การควบคุมการทำงานของ Solid State Relay	171
6.16	การทดลองที่ 4 การแสดงผลข้อมูล 7 ส่วน (7 Segment display)	173
6.17	การทดลองที่ 5 การรับข้อมูลจากคีย์บอร์ด (Scan Key board)	176
6.18	การทดลองที่ 6 การแสดงผลข้อมูลด้วยดอตเมตริก (Dot matrix display)	182
6.19	การทดลองที่ 7 การแสดงผลข้อมูลด้วย LCD Display	188
6.20	การทดลองที่ 8 การควบคุมการทำงานของมอเตอร์ (Motor control) และ ของสเต็ปป์มอเตอร์ (Stepping Motor)	195
6.20.1	การควบคุมการทำงานของมอเตอร์ (Motor Control)	195
6.20.2	สเต็ปป์มอเตอร์ (Stepping Motor)	198
6.21	การทดลองที่ 9 การแปลงสัญญาณดิจิทัลเป็นสัญญาณอนาล็อก	201
6.22	การทดลองที่ 10 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล	204

6.23	การทดลองที่ 11 การควบคุมการทำงานของ Heater และ Temperature Sensor	205
6.23.1	การควบคุมการทำงานของ Heater	205
6.24	การทดลองที่ 12 การวัดค่าความเร็วของมอเตอร์ (Speed sensor)	206
บทที่ 7	สรุปและวิจารณ์	208
7.1	ปัญหาที่พบและวิธีการแก้ไข	208
7.2	สรุปโครงงานที่ทำมา	209
7.3	คำวิจารณ์	209
7.4	แนวทางการพัฒนาต่อไป	209
ภาคผนวก .		
ชุดคำสั่งของ 8051 โดยสังเขป		
วงจรวงจรรวม		
วงจร PCB		
เอกสารอ้างอิง		



สารบัญตาราง

	หน้า
ตารางที่ 2.1 เปรียบเทียบคุณสมบัติของไมโครคอนโทรลเลอร์และไมโครโปรเซสเซอร์	5
ตารางที่ 2.2 แสดงรายละเอียดบางส่วน of ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของบริษัทอินเทล	8
ตารางที่ 2.3 แสดงรายละเอียดบางส่วน of ไมโครคอนโทรลเลอร์ ตระกูล MCS-51 ของบริษัทแอตเมล (Atmel)	9
ตารางที่ 2.4 ตำแหน่งของรีจิสเตอร์แบ่งกั	18
ตารางที่ 2.5 แสดงการเลือกใช้งานรีจิสเตอร์แบ่งกัแต่ละแบ่งกั	19
ตารางที่ 2.6 แสดงรีจิสเตอร์ใช้งานพิเศษ (Special Function Register หรือ SFR)	21
ตารางที่ 2.7 แสดงชื่อสัญญาณอินเตอร์รัปต์ (TF2+EXF2 มีเฉพาะ 8052 ขึ้นไป)	22
ตารางที่ 2.8 บิตต่างๆภายในรีจิสเตอร์ IE (Interrupt Enable)	24
ตารางที่ 2.9 บิตต่างๆภายในรีจิสเตอร์ IP (Interrupt Priority)	25
ตารางที่ 2.10 แสดงแอดเดรสของอินเตอร์รัปต์เวกเตอร์	26
ตารางที่ 2.11 บิตต่างๆภายในรีจิสเตอร์ TCON ที่เกี่ยวข้องกับการอินเตอร์รัปต์ภายนอก	28
ตารางที่ 2.12 บิตต่างๆภายในรีจิสเตอร์ TMOD (Timer/Count Mode Control)	29
ตารางที่ 2.13 บิตต่างๆภายในรีจิสเตอร์ TCON	30
ตารางที่ 2.14 การกำหนดสถานะในการทำงาน	31
ตารางที่ 3.1 แสดงโมดูลต่างๆที่มีในบอร์ดทดลองอินพุตเอาต์พุต	52
ตารางที่ 3.2 ตารางแสดงหน้าที่แต่ละขาของ LCD	62
ตารางที่ 3.3 แสดงชุดคำสั่งของ LCD	64
ตารางที่ 3.4 ตารางการกระตุ้นเฟสแต่ละแบบของสเต็ปมอเตอร์	74
ตารางที่ 4.1 แสดงการการกระตุ้นให้สเต็ปมอเตอร์หมุน	96
ตาราง 4.2 แสดงค่าที่ได้จากการแปลงสัญญาณ อานาล็อกเป็นสัญญาณดิจิตอล	101
ตาราง 4.3 แสดงค่าที่ได้จากชุดตรวจวัดอุณหภูมิ	102
ตารางที่ 6.1 ตารางบันทึกผลการทดลอง	118
ตารางที่ 6.2 ตารางบันทึกผลการทดลองของ Solid State Relay	126
ตารางที่ 6.3 ตารางบันทึกผลการทดลองของ Solid State Relay	128
ตารางที่ 6.4 ตารางบันทึกผลการทดลองของ 7 Segment display	130
ตารางที่ 6.5 ตารางบันทึก รหัส 7 Segment display จากการทดลอง	130
ตารางที่ 6.6 ตารางบันทึกผลของการทดลอง Dot matrix display	136

สารบัญตาราง(ต่อ)

	หน้า
ตารางที่ 6.7 ตารางแสดงคำสั่งควบคุมการทำงานของ LCD	139
ตารางที่ 6.8 ตารางการกระตุ้นเฟสแต่ละแบบของ Stepping Motor	146
ตารางที่ 6.9 ตารางบันทึกผลของการทดลอง Stepping Motor	149
ตารางที่ 6.10 ตารางบันทึกผลของการทดลองของ ADC	153
ตารางที่ 6.11 ตารางบันทึกผลการทดลอง Port monitor (LED)	161
ตารางที่ 6.12 ตารางบันทึกผลการทดลองของ Solid State Relay	169
ตารางที่ 6.13 ตารางบันทึกผลการทดลองของ Solid State Relay	171
ตารางที่ 6.14 ตารางบันทึกผลการทดลองของ 7 Segment display	173
ตารางที่ 6.15 ตารางบันทึกรหัส 7 Segment display จากการทดลอง	173
ตารางที่ 6.16 ตารางบันทึกผลของการทดลอง Dot matrix display	182
ตารางที่ 6.17 ตารางบันทึกผลของการทดลอง DAC	200
ตารางที่ 6.18 ตารางบันทึกผลของการทดลองของ ADC	204

สารบัญรูปภาพ

	หน้า
รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครโปรเซสเซอร์	4
รูปที่ 2.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์	4
รูปที่ 2.3 บล็อกไดอะแกรมโครงสร้างของไมโครคอนโทรลเลอร์ MCS-51	7
รูปที่ 2.4 สถาปัตยกรรมของ MCS-51	7
รูปที่ 2.5 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51	10
รูปที่ 2.6 แสดงโครงสร้างพอร์ต 0 ของไมโครคอนโทรลเลอร์ MCS-51	11
รูปที่ 2.7 แสดงโครงสร้างพอร์ต 1 ของไมโครคอนโทรลเลอร์ MCS-51	11
รูปที่ 2.8 แสดงโครงสร้างพอร์ต 2 ของไมโครคอนโทรลเลอร์ MCS-51	12
รูปที่ 2.9 แสดงโครงสร้างพอร์ต 3 ของไมโครคอนโทรลเลอร์ MCS-51	12
รูปที่ 2.10 แสดงการใช้คริสตัลภายนอกต่อเข้ากับวงจรออสซิลเลเตอร์ภายในไมโครคอนโทรลเลอร์	14
รูปที่ 2.11 แสดงลำดับการทำคำสั่งขนาด 1 ไบต์และใช้เวลาประมาณผลนาน 1 เมกซีเอ็นไอเซล	15
รูปที่ 2.12 การจัดพื้นที่หน่วยความจำโปรแกรมสำหรับไมโครคอนโทรลเลอร์	16
รูปที่ 2.13 การจัดพื้นที่หน่วยความจำข้อมูลสำหรับไมโครคอนโทรลเลอร์ 8051	17
รูปที่ 2.14 การจัดพื้นที่หน่วยความจำข้อมูลภายใน	18
รูป 2.15 หน่วยความจำข้อมูลภายในบริเวณที่เข้าถึงได้แบบบิต	20
รูปที่ 2.16 แสดงแหล่งที่มาและแผนภาพโครงสร้างของระบบการอินเตอร์รัปต์ของ 8051	23
รูปที่ 2.17 แสดงให้เห็นแผนภาพการทำงานของโปรแกรมเมื่อมี Interrupt เข้ามา	26
รูปที่ 2.18 การอินเตอร์รัปต์ทำงานที่ระดับสัญญาณขอบขาของสัญญาณอินเตอร์รัปต์	27
รูปที่ 2.19. การอินเตอร์รัปต์ทำงานที่ระดับสัญญาณลอจิกต่ำ	27
รูปที่ 2.20 ส่วนควบคุมการทำงานของวงจรมับ/จับเวลาซึ่งประกอบด้วยส่วนของ การกำหนดที่มาของสัญญาณ(Timer หรือ Counter) และบิตหรือขาสัญญาณ สำหรับการหยุดหรือทำงานของวงจรมับ	30
รูปที่ 2.21 การทำงาน Timer/Counter 0 หรือ 1 ในโหมด 0	31
รูปที่ 2.22 การทำงาน ของTimer/Counter 0 หรือ 1 ในโหมด 1	32
รูปที่ 2.23 การทำงาน ของTimer/Counter 0 หรือ 1 ในโหมด 2	33
รูปที่ 2.24 วงจรมับ/จับเวลา 0 เมื่อทำงาน โหมด 3	34
รูปที่ 2.25 แสดงการสื่อสารแบบอนุกรม	34
รูปที่ 2.26 แสดงการสื่อสารแบบขนาน	35

สารบัญรูปภาพ(ต่อ)

	หน้า
รูปที่ 2.27 บิตข้อมูลสื่อสารแบบขนานภายในไบต์ข้อมูลที่จะถูกส่งออกมา ในลักษณะแบบอนุกรม	35
รูปที่ 2.28 แสดงสัญญาณไฟฟ้าของข้อมูลในรูปที่ 2.26 ซึ่งเป็นการส่งข้อมูล แบบอนุกรมด้วยอัตราเร็ว 24000 บิต/วินาที	36
รูปที่ 3.1 Block diagram ของบอร์ดไมโครคอนโทรลเลอร์ 8051	38
รูปที่ 3.2 แสดงโมดูลซีพียู	39
รูปที่ 3.3 การ Charge และ Recharge ของ C ที่ขั้วรีเซ็ตของไมโครคอนโทรลเลอร์ และทำให้เกิดการรีเซ็ต	40
รูปที่ 3.4 แสดงโมดูล Memory	41
รูปที่ 3.5 แสดงขาสัญญาณของไอซีเบอร์ 8255	42
รูปที่ 3.6 แสดงบล็อกภายในของไอซีเบอร์ 8255	43
รูปที่ 3.7 แสดง Mode การทำงานของ 8255	44
รูปที่ 3.8 แสดง Command Word ของ 8255	45
รูปที่ 3.9 แสดงโมดูล Timer/Counter	46
รูปที่ 3.10 แสดงโมดูลอินเตอร์รัปต์	47
รูปที่ 3.11 แสดงคอนเนกเตอร์ของ System port	48
รูปที่ 3.12 แสดงรูปไอซี MAX232 และโครงสร้างภายในของไอซี MAX232	49
รูปที่ 3.13 แสดงโมดูล Serial Port	50
รูปที่ 3.14 แสดงการต่อพอร์ตการใช้งานของแต่ละ โมดูล	51
รูปที่ 3.15 แสดงการนำเอา Key board มาต่อเพื่อใช้งาน	53
รูปที่ 3.16 แสดงการต่อโมดูล ดิพสวิทช์	54
รูปที่ 3.17 แสดงความไม่ละเอียดของการแปลงสัญญาณของ A/D	55
รูปที่ 3.18 แสดงโมดูล ADC	56
รูปที่ 3.19 แสดงวงจรชุดตรวจวัดอุณหภูมิ	56
รูปที่ 3.20 แสดงเอาต์พุตของวงจรตรวจจับความเร็วเมื่อมอเตอร์หมุน 1 รอบ	57
รูปที่ 3.21 แสดงวงจร Speed Sensor	58
รูปที่ 3.22 วงจรชุดควบคุมความร้อน	58
รูปที่ 3.23 แสดงการต่อโมดูล LED กับ 8255	59
รูปที่ 3.24 แสดงโครงสร้างภายในของ 7-Segment ทั้ง 2 แบบ	60
รูปที่ 3.25 แสดงตำแหน่งขาของ 7-Segment Display	60
รูปที่ 3.26 แสดงการต่อโมดูล 7 Segment	61

สารบัญรูปภาพ(ต่อ)

	หน้า
รูปที่ 3.27 แสดงวงจรควบคุม LCD	63
รูปที่ 3.28 แสดงการต่อ Buzzer กับพอร์ต PB ของ 8255	67
รูปที่ 3.29 แสดงสัญลักษณ์ของ Magnetic Relay	68
รูปที่ 3.30 วงจร Magnetic Relay ในหนึ่ง Channel	68
รูปที่ 3.31 แสดงโมดูล Solid state relay ในหนึ่ง Channel	69
รูปที่ 3.32 แสดงโมดูล Digital to Analog Converter	70
รูปที่ 3.33 แสดงวงจร R-2R Ladder	71
รูปที่ 3.34 วงจรควบคุมสตีปีงมอเตอร์	73
รูปที่ 3.35 แสดงการจ่ายแรงดันให้กับมอเตอร์เป็นช่วงๆ	74
รูปที่ 3.36 วงจรควบคุม DC MOTOR	75
รูปที่ 3.37 วงจรชุดแสดงผลข้อความ	76
รูปที่ 4.1 แสดงการต่อบอร์ดไมโครคอนโทรลเลอร์และการต่อสาย Serial แบบ Null modem สำหรับการทดลองกับ PC	77
รูปที่ 4.2 การกำหนดพอร์ตอนุกรม COM1	
รูปที่ 4.3 แสดงอุปกรณ์ช่วยในการดาวน์โหลดโปรแกรมจาก PC ไปที่บอร์ดไมโครคอนโทรลเลอร์	78
รูปที่ 4.4 โฟลว์ชาร์ตการ RUN และดาวน์โหลดโปรแกรม	79
รูปที่ 4.5 แสดง Block diagram ของบอร์ดทดลองอินพุตเอาต์พุต	87
รูปที่ 4.6 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน LED Module	88
รูปที่ 4.7 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน 7-Segment Module	89
รูปที่ 4.8 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน LCD	91
รูปที่ 4.9 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน BUZZER	92
รูปที่ 4.10 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Magnetic Relay	93
รูปที่ 4.11 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Solid state Relay	94
รูปที่ 4.12 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน DAC	95
รูปที่ 4.13 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Stepping Motor	96
รูปที่ 4.14 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Motor control	97
รูปที่ 4.15 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Dot matrix Led Display Module	98
รูปที่ 4.16 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Key Board	99
รูปที่ 4.17 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Dip-Switch	100

สารบัญรูปภาพ(ต่อ)

	หน้า
รูปที่ 4.18 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน ADC	101
รูปที่ 4.19 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Temperature Sensor	102
รูปที่ 4.20 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Speed sensor	103
รูปที่ 4.21 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Heater	104
รูปที่ 6.1 Control Word 8255	116
รูปที่ 6.2 ไฟว์แบบที่1	119
รูปที่ 6.3 ไฟว์แบบที่2	120
รูปที่ 6.4 ไฟว์แบบที่3	120
รูปที่ 6.5 ไฟว์แบบที่4	121
รูปที่ 6.6 แสดงความไม่ระเอียดของการแปลงสัญญาณของ A/D	151



บทที่ 1

บทนำ

เนื่องจากความก้าวหน้าทางเทคโนโลยีในปัจจุบัน เป็นผลให้มีการนำไมโครโปรเซสเซอร์ (Microprocessor) มาใช้ในชีวิตประจำวันมากขึ้น ไม่ว่าจะเป็นเครื่องใช้ไฟฟ้าทั่วไป, ระบบควบคุม, ระบบสื่อสารข้อมูล, เครื่องมือวัดทางไฟฟ้า, ของเด็กเล่น, ตลอดจนใช้ในการควบคุมแต่ละจุดการทำงาน ในกระบวนการผลิตของโรงงานอุตสาหกรรม ล้วนแล้วแต่มีไมโครโปรเซสเซอร์เป็นตัวควบคุมการทำงานทั้งสิ้น

แต่จากการพัฒนาที่มีมากขึ้นเป็นลำดับ จึงได้มีการนำวงจรพื้นฐานที่จำเป็นรวมเข้าไปในไมโครโปรเซสเซอร์ซึ่งเรียกรวมกันว่าไมโครคอนโทรลเลอร์ (Microcontroller) ทำให้ไมโครคอนโทรลเลอร์มีความสามารถมากขึ้น ในขณะที่ขนาดของวงจรและจำนวนชิป (Chip) ที่ใช้ใน วงจรมีน้อยลง เหตุที่นำไมโครคอนโทรลเลอร์ 8051 มาประยุกต์ใช้งานเนื่องจากมีขีดความสามารถสูงทำงานได้รวดเร็วมีขนาดเล็กการเปลี่ยนแปลงฟังก์ชันในการทำงานเพียงแค่เปลี่ยนโปรแกรมเท่านั้น โดยไม่ต้องเปลี่ยนแปลงฮาร์ดแวร์เลย และเป็นที่นิยมสำหรับงานที่ไม่ซับซ้อนมากนัก

1.1 ความมุ่งหมายและวัตถุประสงค์ของโครงการ

วัตถุประสงค์สำคัญของโครงการนี้คือ ออกแบบและสร้างบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองซึ่งประกอบไปด้วยบอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองทดลองอินพุตเอาต์พุต เพื่อเป็นต้นแบบของบอร์ดทดลอง ที่จะสร้างให้กับ มหาวิทยาลัยแห่งชาติลาว (NUOL) และเพื่อเป็นการตรวจสอบและทดสอบการทำงานทั้งหมดทางด้านฮาร์ดแวร์และซอฟต์แวร์ของบอร์ดไมโครคอนโทรลเลอร์ สำหรับการทดลองที่ได้ออกแบบและสร้างไว้แล้ว ได้มีการจัดทำใบงานทดลองขึ้นมาเพื่อใช้ในการทดสอบการทำงานทั้งหมดเพื่อศึกษาหาความรู้ทั่วไปและการใช้งานของ MCS-51 ซึ่งเป็นชิปของบอร์ดไมโครคอนโทรลเลอร์ 8051 ซึ่งเป็นส่วนหนึ่งของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง ทั้งหมด เพื่อใช้เป็นประโยชน์สำหรับนักศึกษาและผู้ที่มีความสนใจ ต้องการศึกษการใช้งานของ MCS-51 เพื่อนำไปใช้และพัฒนางานที่เป็นประโยชน์ต่อไป

1.2 ขอบเขตของโครงการ

1. ศึกษา ออกแบบ และสร้างบอร์ดสำหรับการทดลอง
2. ทดสอบการทำงานทั้งหมดของบอร์ด ให้ได้บอร์ดที่พร้อมจะใช้งาน
3. เขียนคู่มือการใช้งานบอร์ดทั้งหมด
4. ออกแบบใบงานสำหรับการทดลองพร้อมคำเฉลย
5. เขียนบทวิทยานิพนธ์

1.3 ขั้นตอนการดำเนินงาน

1. โครงการนี้เริ่มจากการศึกษาเกี่ยวกับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 และศึกษาบอร์ดต่างๆที่มีขายตามท้องตลาด
2. ออกแบบ และ สร้างบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง ให้ใช้งานได้จริง
3. ตรวจสอบการทำงานทั้งหมดของ บอร์ดไมโครคอนโทรลเลอร์ที่ได้ออกแบบและสร้างไว้
4. จัดทำใบงานการทดลองทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์เพื่อทดสอบชุดคำสั่งและการติดต่อกับอุปกรณ์ภายนอก

1.4 รายละเอียดของโครงการ

เนื้อหาในปฏิญานพนธ์นี้แบ่งออกเป็นหลายๆส่วนในแต่ละบทโดยมีรายละเอียดของบทต่างๆทั่วไปดังนี้คือ

- บทที่ 1 กล่าวถึงบทนำ
- บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ในตระกูล MCS-51 และ ชุดคำสั่งของ MCS-51
- บทที่ 3 กล่าวถึงการออกแบบและการพัฒนาทั้งสองส่วนคือบอร์ด I/O และบอร์ด Microcontroller
- บทที่ 4 กล่าวถึงการนำใช้ซอฟต์แวร์
- บทที่ 5 กล่าวถึงการทดสอบบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง
- บทที่ 6 กล่าวถึงใบงานการทดลองในแต่ละส่วนพร้อมเฉลย
- บทที่ 7 กล่าวถึงบทสรุปและวิจารณ์ผลการทดลอง
- ภาคผนวก
- ชุดคำสั่งของ MCS-51
- วงจรรวมและ PCB

บทที่ 2

ไมโครคอนโทรลเลอร์ MCS-51

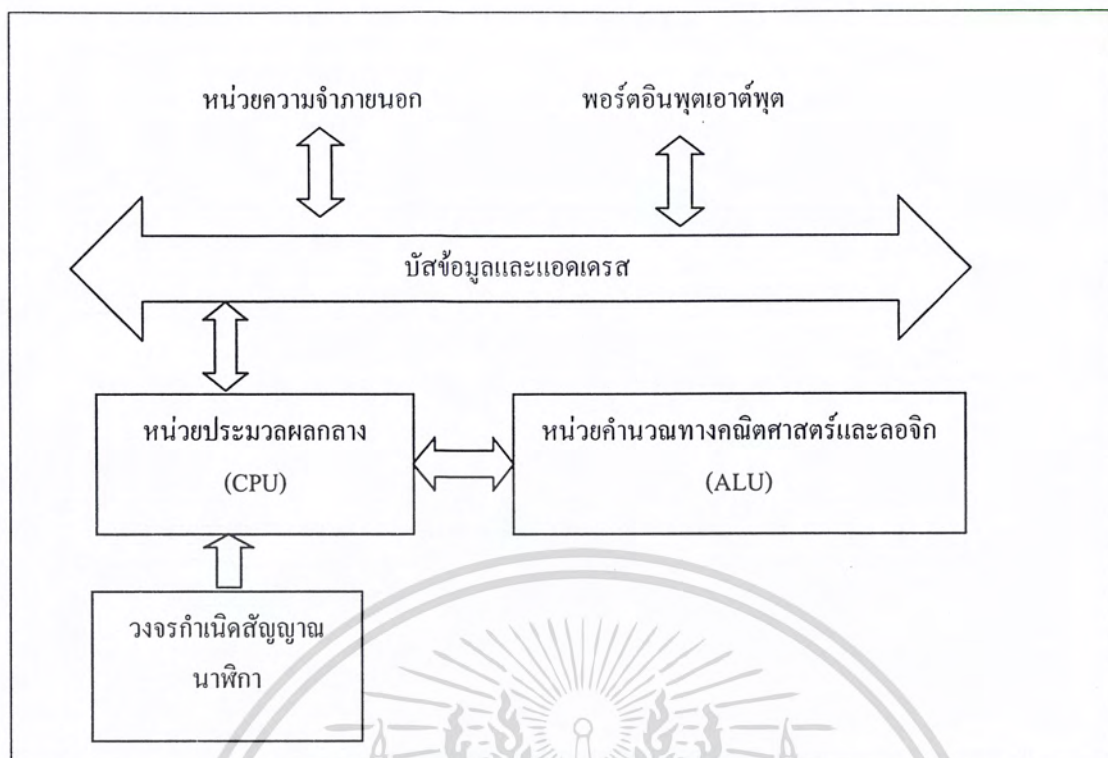
2.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่ง ที่บรรจุความสามารถมากมายไม่ว่าจะเป็นหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์ และ ลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณออกทางเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกา ทำให้ไมโครคอนโทรลเลอร์ สามารถนำไปประยุกต์ใช้งานแทนวงจรอิเล็กทรอนิกส์ ที่ซับซ้อนได้เป็นอย่างดี โดยช่วยลดจำนวนอุปกรณ์ และขนาดของระบบลง ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

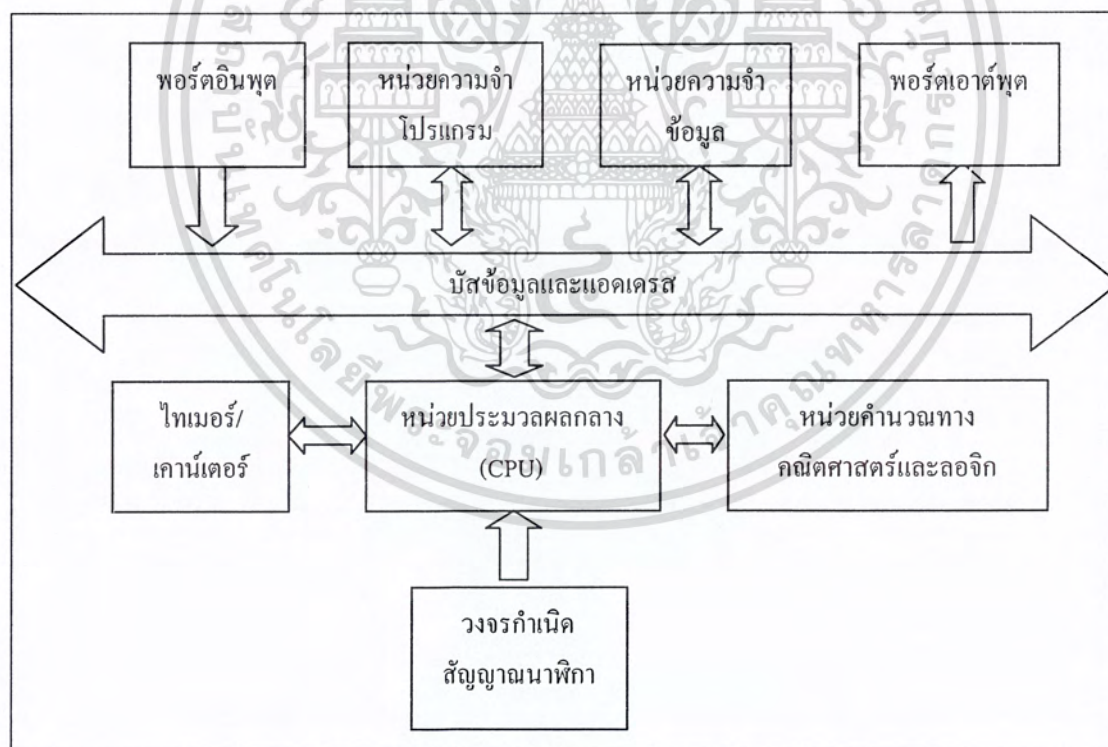
ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันคือ “ไมโคร” (Micro) ซึ่งหมายถึงไมโครโพรเซสเซอร์ (Microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลขนาดเล็ก ภายในประกอบด้วย หน่วยประมวลผลกลางหรือ ซีพียู (CPU: Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์ และลอจิก (ALU : Arithmetic-Logic Unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรเชื่อมต่อสัญญาณนาฬิกา อีกคำหนึ่งคือ “คอนโทรลเลอร์” (Controller) หมายถึงอุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรม เพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

2.2 ความแตกต่างของไมโครคอนโทรลเลอร์กับไมโครโพรเซสเซอร์

ในรูปที่ 2.1 แสดงโครงสร้างของไมโครโพรเซสเซอร์ ซึ่งประกอบด้วยหน่วยประมวลผลกลาง หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูลและแอดเดรส สำหรับติดต่อกับหน่วยความจำภายนอก และวงจรกำเนิดสัญญาณนาฬิกาซึ่งหมายความว่า การใช้งานไมโครโพรเซสเซอร์นั้น จะต้องเชื่อมต่อกับหน่วยความจำโปรแกรมภายนอก และถ้าหากมีความต้องการ เชื่อมต่อกับอุปกรณ์อินพุต และเอาต์พุต ต้องอาศัยอุปกรณ์ที่เรียกว่า ไอซีขยายพอร์ต (Port Expander) ทำให้การใช้งานระบบควบคุมไมโครโพรเซสเซอร์ มีความต้องการอุปกรณ์จำนวนมาก ส่งผลให้ขนาดของระบบใหญ่พอสมควร



รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครโปรเซสเซอร์



รูปที่ 2.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

ในรูปที่ 2.2 แสดงโครงสร้างของไมโครคอนโทรลเลอร์ จะเห็นได้อย่างชัดเจนว่าภายในไมโครคอนโทรลเลอร์มีอุปกรณ์พื้นฐาน เหมือนกับไมโครโพรเซสเซอร์ หากแต่จะบรรจุ หน่วยความจำ โปรแกรม หน่วยความจำข้อมูล และพอร์ตอินพุตเอาต์พุตไว้ภายในพร้อมสรรพ ผู้ใช้ งาน จึงเพียงแค่เขียนโปรแกรมควบคุมลงบนตัวไมโครคอนโทรลเลอร์ แล้วต่ออุปกรณ์ที่ใช้ ในการสร้างวงจรกำเนิดสัญญาณนาฬิกาอาทิ คริสตัล ตัวเก็บประจุเป็นต้น สุดท้ายเชื่อมต่อ อุปกรณ์อินพุตเอาต์พุตที่ต้องการเข้ากับขาพอร์ตของไมโครคอนโทรลเลอร์เพียงเท่านี้ก็สามารถใช้งานไมโครคอนโทรลเลอร์ได้แล้ว ส่งผลให้ขนาดและราคาของระบบลดลงอย่างมาก

คุณสมบัติ	ไมโครโพรเซสเซอร์	ไมโครคอนโทรลเลอร์
ขนาดของหน่วยประมวลผลกลาง	ไม่น้อยกว่า 8 บิต	ส่วนใหญ่จะมีขนาด 8 บิต
หน่วยคำนวณทางคณิตศาสตร์และลอจิก	มีอยู่ภายใน	มีอยู่ภายใน
วงจรกำเนิดสัญญาณนาฬิกา	มีอยู่ภายใน	มีอยู่ภายใน
การเชื่อมต่อกับหน่วยความจำภายใน	เชื่อมต่อกับภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับหน่วยความจำข้อมูล	เชื่อมต่อกับภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับพอร์ตอินพุตเอาต์พุต	เชื่อมต่อกับภายนอกเท่านั้น	มีอยู่ภายในและสามารถขยายได้
ไทมเมอร์/เคาน์เตอร์	ไม่มีในชิพยูนิตขนาดเล็ก	มีอย่างน้อย 1 ตัว ขนาด 8 – 16 บิต
วอตช์ด็อกไทมเมอร์	ไม่มีในชิพยูนิตขนาดเล็ก	มีอย่างน้อย 1 ตัวในเบอร์ AT89S8252
จำนวนขาต่อใช้งาน	ไม่น้อยกว่า 40 ขา	มีตั้งแต่ 8 ขาขึ้นไป

ตารางที่ 2.1 เปรียบเทียบคุณสมบัติของไมโครคอนโทรลเลอร์และไมโครโพรเซสเซอร์

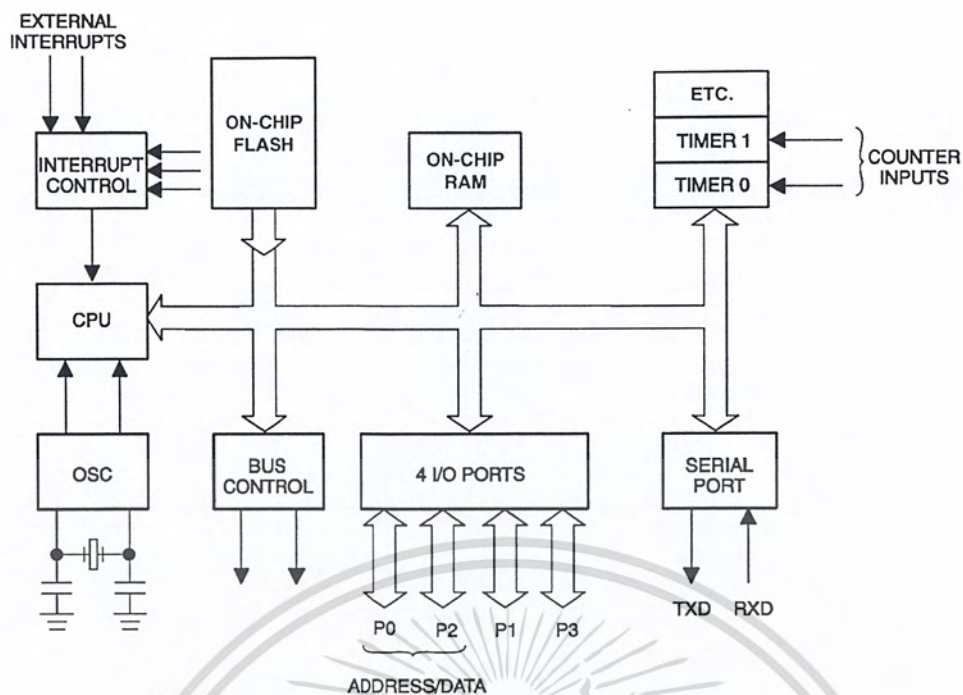
อย่างไรก็ตาม ไมโครคอนโทรลเลอร์ ก็สามารถเชื่อมต่อกับหน่วยความจำภายนอกได้เช่น กัน โดยพิจารณาให้หน่วยความจำภายนอกนั้นเป็นอุปกรณ์อินพุตเอาต์พุตตัวหนึ่ง แล้วใช้ขา พอร์ตที่มีอยู่ทำการติดต่อในตารางที่ 2.1 เป็นตารางสรุปความแตกต่างที่สำคัญระหว่างไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์

2.3 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ตระกูล MCS-51

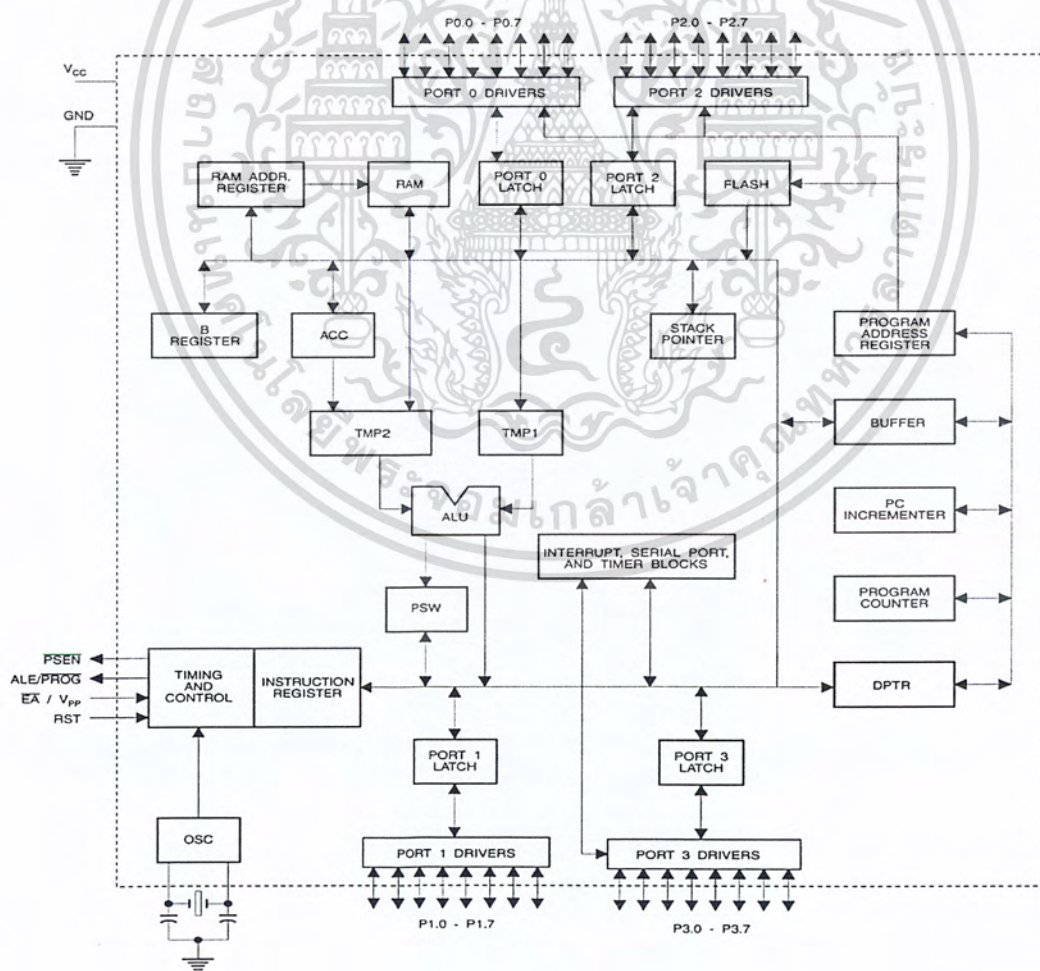
- ต้องการแหล่งจ่ายไฟ +5V ชุดเดียว
- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรมภายใน 128 ไบต์ ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- หน่วยความจำโปรแกรมภายใน 4 กิโลไบต์
- สามารถขยายหน่วยความจำโปรแกรมภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- สามารถขยายหน่วยความจำข้อมูลภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- ขาพอร์ตเป็นแบบสองทิศทาง 32 เส้นสามารถใช้งานได้เป็นทั้งอินพุตและเอาต์พุต และสามารถแยกใช้งานได้อย่างอิสระ
- ไทมเมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัว ทำงานได้ 4 โหมด
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท พร้อมการกำหนดลำดับ ความสำคัญ ได้สองระดับ
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายใน
- มีวงจรสื่อสารอนุกรม (UART) 2 พอร์ต Full Duplex เลือกรูปแบบการสื่อสารได้ 4 โหมด

2.4 โครงสร้างภายในของ 8051

ในรูปที่ 2.1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51. MCS-51 ใช้เทคโนโลยีการผลิตเป็นแบบ NMOS และ CMOS ในเบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวกสำหรับการโปรแกรมเมอร์ที่จะเขียนโปรแกรมในรูปที่ 2.4 เป็น Architectural ของไมโครคอนโทรลเลอร์



รูปที่ 2.3 บล็อกไดอะแกรมโครงสร้างของไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 2.4 สถาปัตยกรรมของ MCS-51

ไมโครคอนโทรลเลอร์หลายรุ่นที่มีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาดหรือ จำนวน หน่วยงานภายในที่แตกต่างกันออกไปเพื่อความเหมาะสมในงานประยุกต์ต่างๆตาม ความต้องการ ดัง แสดงในตารางที่ 2.2 ซึ่งแสดงรายละเอียดบางส่วน of ไมโครคอนโทรลเลอร์ ตระกูล MCS-51ของบริษัท อินเทล และตารางที่ 2.3 เป็นของบริษัท แอตเมล (Atmel)

Feature	8051	8031	8751	80C51	80C31	87C51	8052	8032	8752	8044H
	AH	AH	H	BH	BH		AH	AH		
ROM (Byte)	4K	-	4K	4K	-	4K	8K	-	8K	4K
RAM (Byte)	128	128	128	128	128	128	256	256	256	192
External ROM (Byte)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
External Ram (Byte)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
Max Clock (MHz)	12	12	12	16	16	16	16	12	12	12
Typical instruction time(uS)	1	1	1	0.75	0.75	0.75	1	1	1	1
16 Bit time/Count	2	2	2	2	2	2	3	3	3	2
Serial Communication	Synchronous Mode, Asynchronous Mode , 9 or 10 Bit Programmable								HDLC/SDLC	
NO. of I/O Line	32	16	32	32	16	32	32	16	23	32
Interrupt Source	5	5	5	5	5	5	6	6	6	5
(Two Priory Levels) Power Requirements 125	125	250	24	24	29	175	175	175	200	
(ICC Max mA) programmable power Mode	-	-	-	4.0 mA	4.0 mA	4.0 mA	-	-	-	-
Idle Power down				50uA	51mA	52uA				30mA

ตารางที่ 2.2 แสดงรายละเอียดบางส่วน of ไมโครคอนโทรลเลอร์ตระกูล MCS-51ของบริษัทอินเทล

FEATURE	AT89C 1051	AT8C 2051	AT89C 31	AT89 LV51	AT89C 52	AT89 LV52	AT89 C55	AT89 S8252
FLASH (Byte)	1K	2K	4K	4K	8K	8K	20	8K
RAM (Byte)	64	128	128	128	256	256	256	256
ON-CHIP EEPROM	0	0	0	0	0	0	0	2K
SPINTERFACE	NO	NO	NO	NO	NO	NO	NO	YES
IN-SYSTEM PROGRAMMING	YES	YES	YES	YES	YES	YES	YES	YES
TIMER/COUNTERS	1	2	2	2	3	3	3	4
SERIAL DOWN- LOADING	NO	NO	NO	NO	NO	NO	NO	YES
DATA POINTER	1	1	1	1	1	1	1	2
SECURITY LOCK BITS	2	2	3	3	3	3	3	3
WATCHDOG TIMER	NO	NO	NO	NO	NO	NO	NO	YES
SERIAL UART	NO	YES	YES	YES	YES	YES	YES	YES
INTERRUPT RECOVERY FROM POWER DOWN	NO	NO	NO	NO	NO	NO	NO	YES
ANALOGUE OMPARATOR	YES	YES	NO	NO	NO	NO	NO	NO
I/O PINS	15	15	32	32	32	32	32	32
INTERRUPT SOURCES	3	6	6	6	8	8	8	9
EXTERNAL ADDRESS/DATA BUS	NO	NO	YES	YES	YES	YES	YES	YES
POWER DOWN&IDLEMODE	YES	YES	YES	YES	YES	YES	YES	YES
POWER SUPPLY	2.7-6.0	2.7-6.0	4.0-6.0	2.7-0	4.0-6.0	2.7-0	2.7-0	2.7-0
FREQUENCY	0-24	0-24	0-24	0-24	0-24	0-24	0-33	0-33
PIN COUNT	20	20	40/44	40/44	40/44	40/44	40/44	40/44
I/O CURRENT SINK (Ma PER PIN)	20	20	10	10	10	10	10	10
MAX TOTAL I/O CURRENT SINK(mA)	80	80	71	71	71	71	71	71

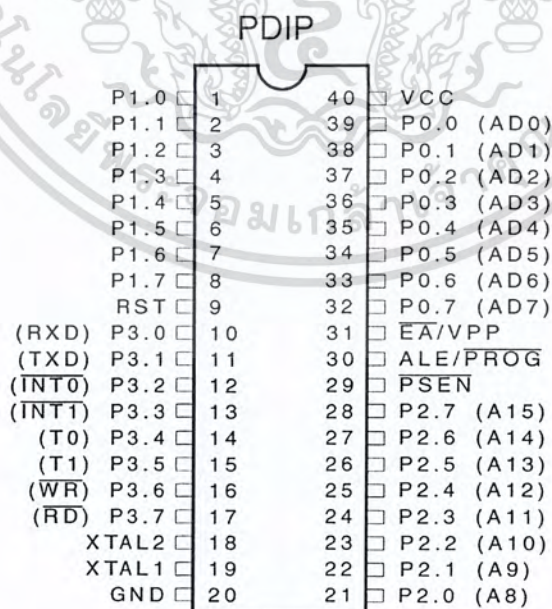
ตารางที่ 2.3 ซึ่งแสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ ตระกูล MCS-51ของบริษัท
แอตเมด (Atmel)

ในตารางที่ 2.2 และ 2.3 ในแถว (Column) แรกจะเป็น Feature ของไมโครคอนโทรลเลอร์ส่วนแถวต่อไปจะเป็นเบอร์ของไมโครคอนโทรลเลอร์ จากการเปรียบเทียบลักษณะสำคัญของไมโครคอนโทรลเลอร์ที่ผลิตโดย บริษัทแอตเมล (Atmel) และของบริษัทอินเทล มีส่วนที่แตกต่างบางส่วนเช่น

- หน่วยความจำโปรแกรมในบริษัทอินเทลจะเป็น ROM ในขณะที่ของบริษัทแอตเมลจะเป็นหน่วยความจำแบบ FLASH ซึ่งสามารถลบและเขียนใหม่ได้เป็นพันครั้ง
- ไมโครคอนโทรลเลอร์ของบริษัทแอตเมลในบางเบอร์สามารถที่จะโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่หรือที่เรียกว่าการโปรแกรมในวงจร หรือในระบบ (In system programming)
- ไมโครคอนโทรลเลอร์ของบริษัทแอตเมลในบางเบอร์จะมีลักษณะการติดต่อแบบ SPI (Serial-Peripheral Interface) ทำให้การพัฒนาหรือการซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัปเดตข้อมูลในหน่วยความจำโปรแกรมทำได้ง่าย
- ไมโครคอนโทรลเลอร์ของบริษัทแอตเมลในบางเบอร์จะมี ON CHIP EEPROM เช่นเบอร์ AT89S8252 ทำให้มีความสะดวกต่อการใช้งาน ไม่ต้องต่อ EEPROM จากภายนอก
- ไมโครคอนโทรลเลอร์ของบริษัทแอตเมลในบางเบอร์ยังมี SERIAL DOWN LOADING ซึ่งเราสามารถที่จะโหลดโปรแกรมที่เขียนเสร็จแล้วลงไปใช้งานได้เลย

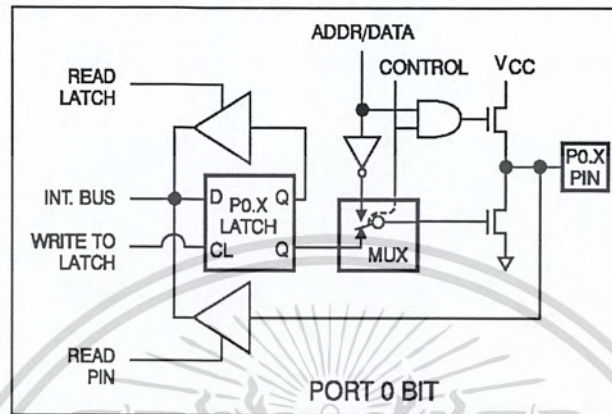
2.5 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกๆเบอร์ จะมีสถาปัตยกรรมและขาใช้งานพื้นฐาน เหมือนกัน ดังแสดงในรูปที่ 2.5 โดยมีรายละเอียดขั้นต้นดังนี้ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V ขา GND เป็นขากราวด์ สำหรับต่อกราวด์ของระบบ



รูปที่ 2.5 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51

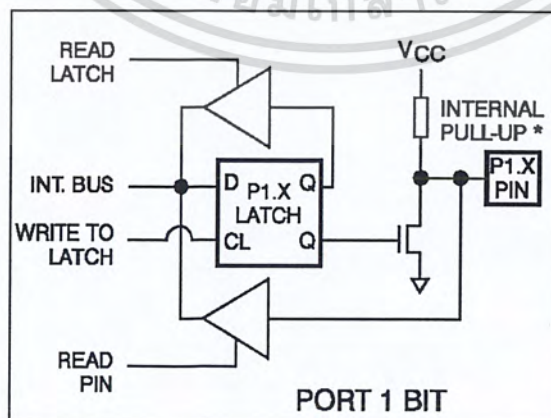
ขาพอร์ต 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (P0.0-P0.7) แต่ละขาสามารถกำหนดให้เป็น ได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ /Q ของ D-FF เป็น “0” ทำให้ FET ตัวล่างมีสถานะ OFF สัญญาณที่ใช้อ่านพอร์ตแลทช์ โดยส่งสัญญาณ READ LATCH ไปกระตุ้นที่ Tri-State Buffer ตัวบนและการ อ่าน Port (Pin) จะใช้สัญญาณ Read (pin)



รูปที่ 2.6 แสดงโครงสร้างพอร์ต 0 ของไมโครคอนโทรลเลอร์ MCS-51

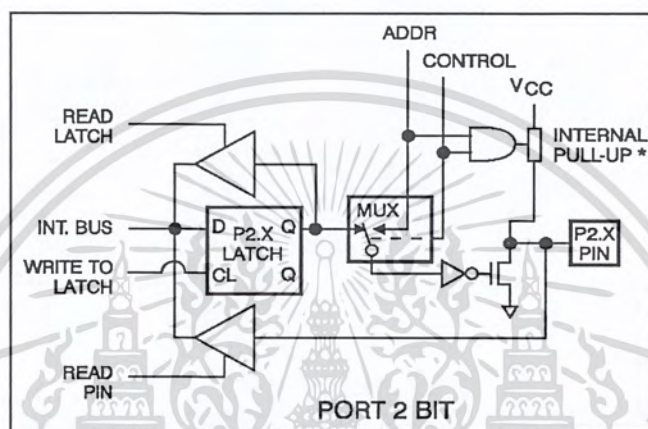
นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับแอดเดรสไบต์ต่ำของหน่วยความจำ ภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานให้เป็นไปได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต 1 (ขา 1-8) มีทั้งหมด 8 บิต คือ (P1.0-P1.7) แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 1 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย โดยการทำงานและโครงสร้างของพอร์ตนี้จะคล้ายกับพอร์ต 0 แต่จะต่างกันที่พอร์ตนี้มีความต้านทานภายในพูลอัพแทน Internal Pull-up Register



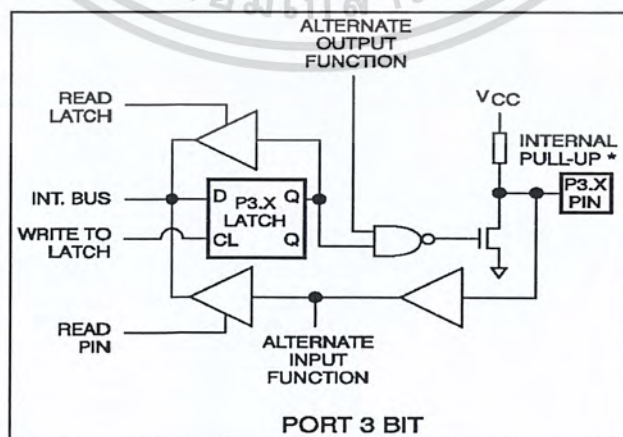
รูปที่ 2.7 แสดงโครงสร้างพอร์ต 1 ของไมโครคอนโทรลเลอร์ MCS-51

ขาพอร์ต 2 (ขา 21-28) มีทั้งหมด 8 บิต คือ (P2.0-P2.7) มีโครงสร้างคล้ายกับพอร์ต 0 โดยมี FET ตัวกลางตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน (Internal Pull up) พอร์ตนี้ ขาแต่ละขา สามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการ กำหนดให้ขาพอร์ต2 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยัง แต่ละ บิตของพอร์ตที่ต้องการติดต่อด้วย เพื่อบังคับให้ FET อยู่ในสภาวะ OFF ส่งผลให้พอร์ต นั้นมี สถานะปล่อยลอยจึงมีอินพุตอิมพีแดนสูง สามารถใช้งานเป็นพอร์ตขาอินพุตได้ นอกจาก นั้นขา พอร์ตนี้ยังถูกใช้งานในการติดต่อกับแอดแดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)



รูปที่ 2.8 แสดงโครงสร้างพอร์ต 2 ของไมโครคอนโทรลเลอร์ MCS-51

พอร์ต 3 (ขา 10-17) มีทั้งหมด 8 บิต คือ (P3.0-P3.7) แต่ละขาสามารถกำหนดให้เป็น ได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้พอร์ตนั้นมีสถานะปล่อยลอยจึงมีอินพุตอิมพีแดนสูง สามารถใช้งานเป็นพอร์ตขาอินพุตได้ นอกจากนั้นขาของพอร์ต 3 ยังเป็นขาที่มีการใช้งานพิเศษ ดังรายละเอียดต่อไปนี้



รูปที่ 2.9 แสดงโครงสร้างพอร์ต 3 ของไมโครคอนโทรลเลอร์ MCS-51

- P3.0 ใช้เป็นขาอินพุต สำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD (UART)
- P3.1 ใช้เป็นขาเอาต์พุต สำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD (UART)
- P3.2 ใช้เป็นขาอินพุต สำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 0 หรือขา /INT0 (External Interrupt 0)
- P3.3 ใช้เป็นขาอินพุต สำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 1 หรือขา /INT1 (External Interrupt 1)
- P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณ Counter จากภายนอกช่องที่ 0 หรือขา T0
- P3.5 ใช้เป็นขาอินพุต สำหรับรับสัญญาณ Counter จากภายนอกช่องที่ 1 หรือขา T1
- P3.6 ใช้เป็นขาสัญญาณ /WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก (External Data-memory Write Strobe)
- P3.7 ใช้เป็นขาสัญญาณ /RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก (External Data-memory Read Strobe)

ขารีเซต (RST ขา 9) ใช้ในการรีเซ็ตการทำงานของไมโครคอนโทรลเลอร์โดยในการป้อน สัญญาณในการรีเซต สถานะที่ขาขึ้นอยู่กับระดับรีเซตอย่างน้อย 2 แมกซีนไซเคิล โดยที่วงจรกำเนิด สัญญาณนาฬิกา ยังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE/PROG ขาที่ 30 (Address Latch Enable/program pulse input) เป็นขาที่ใช้ในการ ควบคุมการแลตช์ของพอร์ต 0 เมื่อมีการใช้หน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขา สำหรับรับพัลส์ของการ โปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีอีพรอม

ขา /PSEN ขาที่ 29 (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อ กับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ต้องส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละ ไมซีนไซเคิล แต่หาก ติดต่อ กับ หน่วยความจำ ข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา

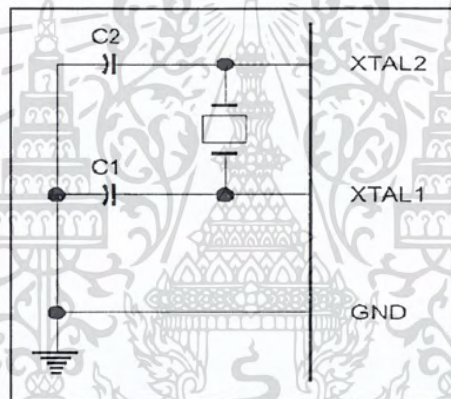
ขา EA/Vpp (External Acces enable/Programming voltage input) ขาที่ 29 ใช้สำหรับเลือก การติดต่อหน่วยความจำโปรแกรมจากภายนอก หรือภายในไมโครคอนโทรลเลอร์ ถ้าหากขานี้ เป็น “0” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ หากขานี้เป็น “1” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ ติดต่อกับหน่วยความจำ ภายในตัวไมโครคอนโทรลเลอร์เอง นอกจากนี้ขานี้ ยังใช้เป็นขาอินพุต สำหรับรับแรงดันไฟสูงสำหรับการ โปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์

ขา XTAL1 ขาที่ 19 และ XTAL2 ขาที่ 18 เป็นขาสำหรับต่อคริสตัลเพื่อสร้างสัญญาณ นาฬิกา ในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.6 ฐานเวลาในการทำงานของซีพียูภายใน 8051

ไมโครคอนโทรลเลอร์ มีวงจรออสซิลเลเตอร์อยู่ภายใน สำหรับการสร้างพัลส์ ของ สัญญาณนาฬิกา ซึ่งจะนำไปเป็นฐานเวลา หรือการกำหนดจังหวะของการทำงาน ทั้งหมดให้สอดคล้องกัน (Synchronization) โดยปกติแล้วก็มักจะทำได้โดยใช้คริสตอล เชื่อมต่อเข้ากับขาสัญญาณ XTAL1 และ XTAL2 พร้อมกับตัวเก็บประจุ ดังในรูปที่ 2.10 หรืออาจจะเป็น สัญญาณนาฬิกาจากภายนอกก็ได้

พัลส์ความถี่ของสัญญาณนาฬิกาจะเรียกว่า Pulse และคาบสัญญาณนาฬิกา นี้ เรียกว่า คาบเวลาออสซิลเลเตอร์ (Oscillator Period) คาบเวลาออสซิลเลเตอร์จำนวน 2 คาบ เรียกว่า State ซึ่งจะนำไปเป็นช่วงเวลาพื้นฐานการทำงานย่อย ของไมโครคอนโทรลเลอร์ เช่น การนำคำสั่ง Fetch มาถอดความหมาย (Decode) การประมวลผล (Execute) และการเขียนข้อมูล (Write) เป็นต้น ดังแสดงในรูปที่ 2.11 ช่วงเวลาของ State จำนวน 6 ครั้ง จะเรียกว่า แมชชีนไซเคิล (Machine Cycle) ดังนั้นค่าหนึ่ง แมชชีนไซเคิลจะใช้เวลา 12 คาบเวลาออสซิลเลเตอร์ ค่าของแมชชีนไซเคิลนี้ จัดว่าเป็นเวลาที่น้อยที่สุด ในการทำคำสั่งใดคำสั่งหนึ่ง ซึ่งหากว่าเป็นคำสั่งที่ซับซ้อนมากก็จะต้องใช้เวลาานสองแมชชีนไซเคิลหรือมากกว่า



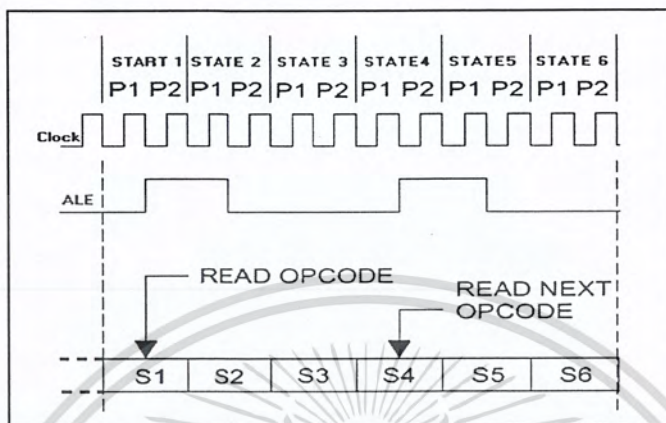
รูปที่ 2.10 แสดงการใช้คริสตอลภายนอกต่อเข้ากับวงจรออสซิลเลเตอร์ภายในไมโครคอนโทรลเลอร์

การคำนวณหาค่าเวลาที่ใช้ในการทำคำสั่งใดคำสั่งหนึ่งจนเสร็จสิ้น จะต้องดูว่าคำสั่งนั้นใช้จำนวนแมชชีนไซเคิลเป็นเท่าไรในการประมวลผล เวลาที่ใช้จะคำนวณตามสูตร

$$T = (C \times 12) / (\text{Crystal Frequency})$$

โดย	C	เป็นค่าแมชชีนไซเคิลของคำสั่ง
	Crystal Frequency	เป็นค่าความถี่ของคริสตอลที่ใช้กับไมโครคอนโทรลเลอร์
	T	เวลาที่ใช้ในการทำคำสั่งใดคำสั่งหนึ่ง

อย่างไรก็ตามในบางครั้งอาจพบเห็นการใช้ค่าคริสตอลเป็น 11.0592 เมกะเฮิร์ตซ์ ทั้งนี้ เหตุผลเนื่องจาก สามารถนำค่าความถี่ที่ได้นี้ ไปใช้ในการเป็นฐานสำหรับการสร้างความถี่ ในการรับ/ส่ง ข้อมูลอนุกรมซึ่งเป็นหน่วยการทำงานหนึ่งในภายในไมโครคอนโทรลเลอร์เอง โดยจะทำให้ได้ค่าที่ใกล้เคียงกับค่ามาตรฐานคือ 19200, 9600, 4800, 2400, 1200, และ 300 บิต/วินาที



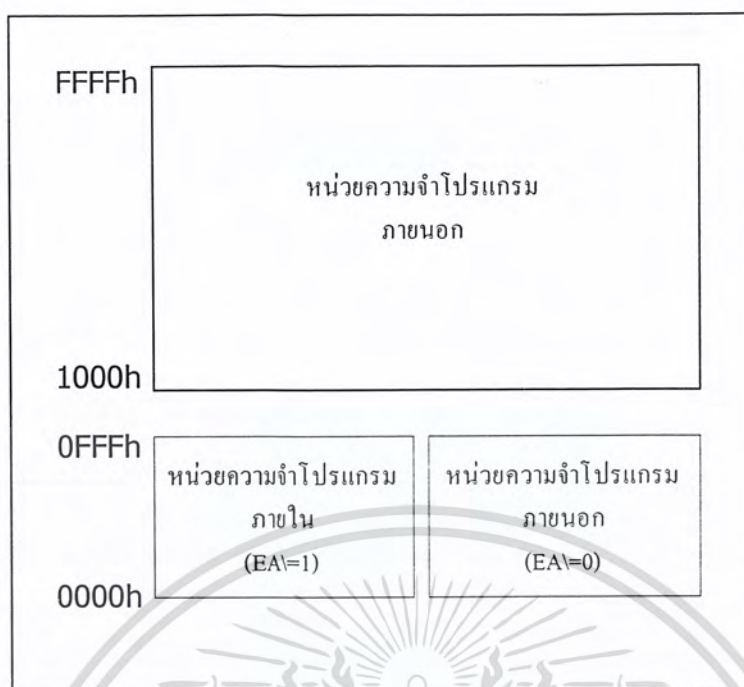
รูปที่ 2.11 แสดงลำดับการทำคำสั่งขนาด 1 ไบต์และใช้เวลาประมวลผลนาน 1 แมกซีนไซเคิล

2.7 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ของ 8051

ในไมโครคอนโทรลเลอร์ 8051 มีหน่วยความจำภายในหลัก อยู่ 2 ส่วนคือหน่วยความจำโปรแกรม(Program Memory) และหน่วยความจำข้อมูล (Data memory) ซึ่งก็มีขนาดและการจัดสรรตำแหน่งแตกต่างกันไปในแต่ละเบอร์

2.7.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของ 8051 เป็นบริเวณหน่วยความจำ สำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบ ข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหาย โครงสร้างของหน่วยความจำโปรแกรมนี้อาจมีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไอซี หน่วยความจำประเภทต่างๆ เช่น หน่วยความจำแบบ ROM (Read Only Memory) หรือ EPROM (Erasable Programmable Memory) เป็นต้นดูรูปที่ 2.12



รูปที่ 2.12 การจัดพื้นที่หน่วยความจำโปรแกรมสำหรับไมโครคอนโทรลเลอร์

8051 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมได้สูงสุดไม่เกิน 64 กิโลไบต์และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (Internal Program Memory) ซึ่งเป็นหน่วยความจำ ROM หรือ EPROM ที่อยู่ภายในตัวไอซีไมโครคอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก (External Program Memory) ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ

2.7.2 หน่วยความจำโปรแกรมภายใน

ไมโครคอนโทรลเลอร์เบอร์ต่างๆ ที่จัดอยู่ในตระกูล 8051 นี้ มีขนาดของหน่วยความจำโปรแกรมภายในแตกต่างกันออกไป เช่น 8051 และ 8052 มีหน่วยความจำแบบ ROM ขนาด 4 และ 8 กิโลไบต์ ตามลำดับประกอบอยู่ในไอซีและมีความเหมาะสมกับการนำไปใช้ในวงจรทางอุตสาหกรรมที่มีจำนวนการผลิตมาก เนื่องจากจะมีผลทำให้ต้นทุนค่าใช้จ่ายการผลิตต่อหน่วยลดลงได้มาก

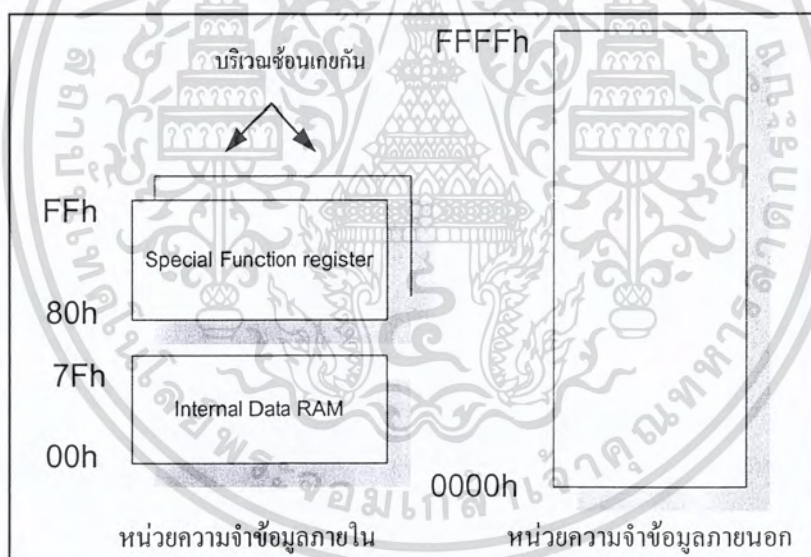
2.7.3 หน่วยความจำโปรแกรมภายนอก

หน่วยความจำโปรแกรมภายนอกเป็นการใช้หน่วยความจำ EPROM (หรือ ROM) เชื่อมต่อกับระบบของ 8051 โดยอาจจะมีสาเหตุได้หลายประการ เช่น เป็นการทดลองทำระบบต้นแบบจำนวนน้อยหรืออาจต้องการลดต้นทุนการผลิต เพราะราคาของไมโครคอนโทรลเลอร์ แบบที่ไม่มีหน่วยความจำโปรแกรมภายในราคาต่ำกว่าแบบที่มีหน่วยความจำโปรแกรมภายในมาก ในไมโครคอนโทรลเลอร์เบอร์ต่างๆ ของตระกูล 8051 นี้ สามารถขยายให้ใช้งานหน่วยความจำภายนอกได้ทั้งหมด โดยกรณีที่หน่วย

ความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้น จะต้องทำการพิจารณาระดับลอจิกของสัญญาณ EA ในขณะนั้นด้วย

2.8 หน่วยความจำข้อมูล (Data Memory)

โดยพื้นฐานแล้วเป็นหน่วยความจำ RAM ทำให้สามารถเขียนหรืออ่านข้อมูลได้ (Read or Write Memory) หน่วยความจำข้อมูลมีหน้าที่สำหรับเก็บข้อมูล หรือตัวแปรที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว โดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำ RAM แบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบ ก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญหายไป พื้นที่ของหน่วยความจำข้อมูลของ 8051 สามารถมีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ดังลักษณะของแผนภาพในรูป 2.13 คือ หน่วยความจำโปรแกรมภายใน (Internal Data Memory) ซึ่งเป็น RAM ที่อยู่ในตัวไอซีไมโครคอนโทรลเลอร์เอง และหน่วยความจำของข้อมูลภายนอก (External Data Memory) ซึ่งเป็นการใช้ไอซีหน่วยความจำ RAM มาเพิ่มเข้าไปในวงจร ลักษณะเกี่ยวกับการนำไอซี EPROM มาใช้งานเป็นหน่วยความจำโปรแกรมนั้นเอง



รูปที่ 2.13 การจัดพื้นที่หน่วยความจำข้อมูลสำหรับไมโครคอนโทรลเลอร์ 8051

2.8.1 หน่วยความจำข้อมูลภายใน

หน่วยความจำข้อมูลภายในของ 8051 มีจำนวนทั้งหมด 256 ไบต์ โดยจำแนกออกได้เป็นสองลักษณะ คือ พื้นที่เฉพาะสำหรับตัวประมวลผลกลาง (CPU) ใช้งานเท่านั้น ซึ่งเรามักจะเรียกกันในชื่อหนึ่งว่า รีจิสเตอร์ และพื้นที่ใช้งานทั่วไปสำหรับโปรแกรมใช้งานที่ผู้ใช้สร้างขึ้นมา รูปที่ 2.14 แสดงให้เห็นถึงการจัดสรรพื้นที่ของหน่วยความจำของข้อมูลภายในของ 8051 ซึ่งจำแนกออกเป็นสองส่วน ดังนี้

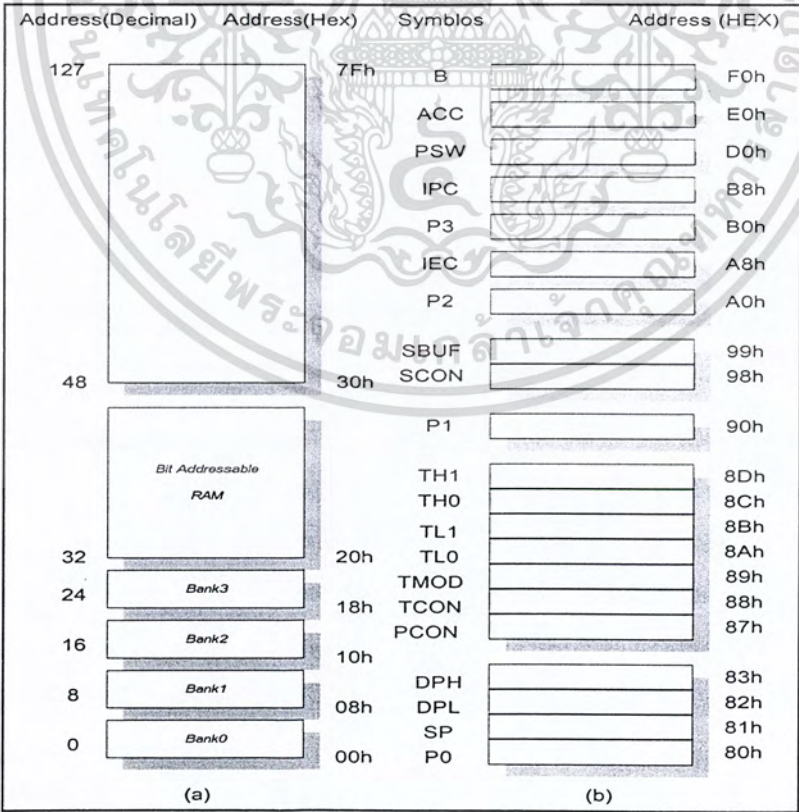
2.8.2 หน่วยความจำขนาด 128 ไบต์แรก

บริเวณนี้จะมีตำแหน่งแอดเดรสอยู่ในช่วง 00H-7FH ซึ่งยังได้มีการจำแนกย่อยออกไปอีกเป็นสามส่วนตามประเภทของการใช้งาน ดังนี้ บริเวณแอดเดรส 00H-1FH จำนวน 32 ไบต์ จำแนกออกเป็นกลุ่ม (หรือ แบงก์ (Bank)) ข้อมูลจำนวน 8 ไบต์ รวมทั้งหมดสี่กลุ่ม พื้นที่ข้อมูลในแต่ละกลุ่มจะถูกใช้งานในฐานะของรีจิสเตอร์ใช้งานทั่วไป ซึ่งมีชื่อเรียกว่า รีจิสเตอร์ R0-R7 ดังตารางต่อไปนี้

แอดเดรส	รีจิสเตอร์แบงก์	ชื่อรีจิสเตอร์ใช้งาน
00H – 07H	0	R0-R7
08H – 0FH	1	R0-R7
10H – 17H	2	R0-R7
18H – 1FH	3	R0-R7

ตารางที่ 2.4 ตำแหน่งของรีจิสเตอร์แบงก์

จะเห็นว่าชื่อของรีจิสเตอร์ไม่ว่าจะอยู่ในรีจิสเตอร์แบงก์ใด ก็จะมีชื่อ R0-R7 เหมือนกันทั้งสิ้น (ดูรูปที่ 2.14)



รูปที่ 2.14 การจัดพื้นที่หน่วยความจำข้อมูลภายใน

รูป (a) ช่วงตั้งแต่แอดเดรส 00-7FH และ รูป (b) ช่วงแอดเดรส 80-FFH ซึ่งกำหนดให้เป็นพื้นที่ของรีจิสเตอร์ใช้งานพิเศษ (Special-function registers) ดังนั้นในการใช้งานผู้ใช้จะต้องให้ความระมัดระวังว่าต้องการรีจิสเตอร์นั้นๆ จากแบงก์ใดก็สวิตช์เลือกแต่ละกลุ่มของรีจิสเตอร์นี้ก็ทำได้ง่าย เพียงการกำหนดค่าของบิตที่อยู่ภายในรีจิสเตอร์ PSW เท่านั้นตามตารางต่อไปนี้

รีจิสเตอร์	บิต RS0	บิต RS1	ตำแหน่งหน่วยความจำ
แบงก์ 0	0	0	0000h-0007h
แบงก์ 1	0	1	0008h-000Fh
แบงก์ 2	1	0	0010h-0017h
แบงก์ 3	1	1	0018h-001Fh

ตารางที่ 2.5 แสดงการเลือกใช้งานรีจิสเตอร์แบงก์แต่ละแบงก์

โดยทั่วไปก็มักจะมีการใช้งานรีจิสเตอร์ R0-R7 เฉพาะในแบงก์ 0 เท่านั้น ดังนั้นพื้นที่ของแบงก์อื่นๆที่เหลือก็สามารถนำมาใช้ในลักษณะของหน่วยความจำข้อมูลภายในปกติด้วยการอ้างถึงหมายเลขของแอดเดรสนั้นๆโดยตรง

บริเวณแอดเดรส 20H-2FH จำนวน 16 ไบต์ บริเวณพื้นที่เป็นส่วนสำหรับผู้ใช้ซึ่งจะมีความพิเศษต่างไปจากหน่วยความจำส่วนอื่นๆ เนื่องจากผู้ใช้ อาจสามารถอ้างถึงหน่วยความจำข้อมูลภายในปกติด้วยการอ้างถึงหน่วยความจำบริเวณนี้ได้ทั้งลักษณะของ ไบต์ข้อมูล เช่นปกติ หรืออาจจะเป็นบิตข้อมูล ได้โดยตรง ดังนั้นหากเรามองในลักษณะบิตข้อมูลแล้ว ก็จะมีพื้นที่ตัวแปรแบบบิตให้ใช้งานได้มากถึง 128 บิต โดยตำแหน่งแรกของบิตจะเป็นบิตซึ่งเริ่มต้นจากบิตนัยสำคัญต่ำสุด (LSB) ของแอดเดรส 20H เรื่อยไปจนกระทั่งถึงบิตที่ 127 ซึ่งเป็นบิตนัยสำคัญสูงสุด (MSB) ของแอดเดรส 2FH (ดูรูปที่ 2.15)

ความสามารถในการใช้งานพื้นที่ส่วนนี้แบบบิตข้อมูลโดยตรงนี้นับว่าน่าสนใจมาก และถือเป็นการใช้งาน 8051 อย่างเต็มประสิทธิภาพทีเดียว เนื่องจากว่า 8051 ได้รับการออกแบบมาสำหรับงานควบคุมเป็นพื้นฐานอยู่แล้ว ซึ่งส่วนมากงานลักษณะเช่นนี้หากเป็นการนำเข้าข้อมูลก็มักจะเป็นเพียงการอ่านค่าสถานะลอจิกของเส้นสัญญาณหรือการส่งออกข้อมูลก็จะเป็นการกำหนดภาวะลอจิกให้กับวงจรภายนอกผ่านทางบิตใดหนึ่งอยู่แล้ว ดังนั้นหากว่ามีการกำหนดบิตหรืออ่านค่าของบิตมาโดยตรง แทนที่จะต้องทำลอจิกขึ้นต้นกับข้อมูลทั้งไบต์เพื่อต้องการทราบผลเพียงหนึ่งบิตเช่นที่กระทำกันในโปรเซสเซอร์โดยทั่วไป ก็จะเพิ่มความสะดวกและรวดเร็วในการเขียนโปรแกรมควบคุมมาก รายละเอียดในส่วนนี้จะได้กล่าวถึงอีกครั้งหนึ่งเมื่อศึกษาถึงการใช้งานพอร์ตอินพุต/เอาต์พุตต่อไปบริเวณแอดเดรส 30H-7FH เป็นบริเวณที่สามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถอ้างถึงได้เฉพาะในลักษณะของไบต์ข้อมูลตามปกติเท่านั้น

(LSM)	(MSB)	แอดเดรสของ RAM
78 79 7A 7B 7C 7D 7E 7F	2Fh	←
70 71 72 73 74 75 76 77	2Eh	
68 69 6A 6B 6C 6D 6E 6F	2Dh	
60 61 62 63 64 65 66 67	2Ch	
58 59 5A 5B 5C 5D 5E 5F	2Bh	
50 51 52 53 54 55 56 57	2Ah	
48 49 4A 4B 4C 4D 4E 4F	29h	
40 41 42 43 44 45 46 47	28h	
38 39 3A 3B 3C 3D 3E 3F	27h	
30 31 32 33 34 35 36 37	26h	
28 29 2A 2B 2C 2D 2E 2F	25h	
20 21 22 23 24 25 26 27	24h	
18 19 1A 1B 1C 1D 1E 1F	23h	
10 11 12 13 14 15 16 17	22h	
08 09 0A 0B 0C 0D 0E 0F	21h	
00 01 02 03 04 05 06 07	20h	← ตำแหน่งบิต

รูป 2.15 หน่วยความจำข้อมูลภายในบริเวณที่อ้างถึงได้แบบบิต

2.8.3 หน่วยความจำขนาด 128 ไบต์ถัดไป

พื้นที่ตั้งแต่แอดเดรส 80H-FFH เป็นบริเวณของหน่วยความจำที่มีการใช้งานเฉพาะจาก 8051 เท่านั้น โดยจะนำมาใช้เป็นตำแหน่งของรีจิสเตอร์หน้าที่พิเศษ (Special Function Register หรือ SFR) สำหรับไมโครคอนโทรลเลอร์เบอร์ 8051 จะมีหน่วยความจำข้อมูลภายในสำหรับการใช้งานเพิ่มมากขึ้นกว่าเบอร์อื่นๆ เช่น 8031 หรือ 8751 อีก 128 ไบต์ โดยจะอยู่บริเวณช่วงแอดเดรส 80H ถึง FFH เช่นกัน ซึ่งแม้ว่าจะเป็นพื้นที่ที่มีหมายเลขแอดเดรสเดียวกับส่วนของรีจิสเตอร์หน้าที่พิเศษแต่ในความเป็นจริงแล้วจะเป็นพื้นที่หน่วยความจำอีกบริเวณหนึ่ง ซึ่งมีการซ้อนทับ (Overlap) กันให้อยู่ในบริเวณแอดเดรสส่วนนี้ ซึ่งหากผู้ใช้งานต้องการจะเก็บข้อมูลในพื้นที่หน่วยบริเวณนี้ก็จะต้องใช้การอ้างถึงหน่วยความจำโดยอ้อม (Indirect Addressing) เท่านั้น ดูรูปที่ 2.13

2.8.4 รีจิสเตอร์หน้าที่พิเศษ

รีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่และการทำงานของอุปกรณ์หรือพอร์ตของ 8051 ทั้งหมด โดยมีตำแหน่งแอดเดรสอยู่ในบริเวณแอดเดรส 80H - FFH การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้สามารถทำได้ทั้งการระบุถึงชื่อของรีจิสเตอร์หรือตำแหน่งของแอดเดรสที่เป็นที่อยู่ของรีจิสเตอร์นั้นก็ได้

ตารางต่อไปนี้แสดงให้เห็นลักษณะการจัดพื้นที่หน่วยความจำสำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดสที่เป็นจำนวนทวีคูณของค่า 8 จะสามารถอ้างถึงระดับบิตได้ด้วย (นั่นคือแอดเดส 80H, 88H, 90H, 98H, A0H, A8H, B0H, B8H, D0H, E0H และ F0H)

ชื่อรีจิสเตอร์	คำจำกัดความ	ความสามารถการอ้างถึงแบบบิต
ACC	Acculator	ได้
B	B Register	ได้
PSW	Program status word	ได้
SP	Stack pointer	ได้
DPTR	Data pointer (DPH & DPL)	ได้
P0	Port 0	ได้
P1	Port 1	ได้
P2	Port 2	ได้
P3	Port 3	ได้
IP	Interrupt Priority	ได้
IE	Interrupt Enable	ได้
TMOD	Timer/Counter mode	ไม่ได้
TCON	Timer / Counter control	ได้
TH0	Timer / Counter 0 (High)	ไม่ได้
TL0	Timer / Counter 0 (Low)	ไม่ได้
TH1	Timer / Counter 1 (High)	ไม่ได้
TL1	Timer / Counter 1 (Low)	ไม่ได้
SCON	Serial control	ไม่ได้
SBUF	Serial data control	ไม่ได้
PCON	Power control	ไม่ได้

ตารางที่ 2.6 แสดงรีจิสเตอร์ใช้งานพิเศษ (Special Function Register หรือ SFR)

2.8.5 หน่วยความจำข้อมูลภายนอก

การใช้หน่วยความจำข้อมูลภายนอกเป็นวิธีการแก้ปัญหาอย่างหนึ่ง ในกรณีที่มีความต้องการหน่วยความจำสำหรับการเก็บข้อมูลชั่วคราว หรือตัวแปรของโปรแกรมมากเกินไปจนกว่าขนาดของหน่วยความจำข้อมูลภายในซึ่งมีขนาดเพียง 128 หรือ 256 ไบต์เท่านั้น บางครั้งการใช้หน่วยความจำเก็บข้อมูลบางอย่างไว้ไม่ให้สูญหายแม้ว่าจะไม่มีการจ่ายไฟให้กับระบบ ก็สามารถทำได้โดยการใช้ไอซีหน่วยความจำ RAM พร้อมแบตเตอรี่สำรองประเภทลิเทียมหรืออินทิเกรต-แคดเมียมเป็นตัวเก็บข้อมูลเหล่านี้ไว้แทน

2.9 การอินเทอร์รัปต์

การติดต่อระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก มักจะทำการตรวจสอบสถานะของสัญญาณติดต่อระหว่างกัน การอินเทอร์รัปต์เป็นวิธีการหนึ่งที่นิยมนำมาใช้กับไมโครคอนโทรลเลอร์เพื่อสามารถจัดการตอบรับหรือบริการกับอุปกรณ์ต่างๆ ให้เป็นไปได้อย่างรวดเร็ว ความสามารถในการดำเนินการจัดการสัญญาณอินเทอร์รัปต์จากแหล่งกำเนิดสัญญาณหลายประเภทของ 8051 ถือได้ว่าเป็นลักษณะเด่นประการหนึ่ง ซึ่งหากว่าได้มีการนำมาใช้ในการออกแบบแล้ว ก็จะส่งผลให้ระบบสามารถตอบสนองต่อเหตุการณ์ภายนอกที่เกิดขึ้นได้ดียิ่งขึ้น

สำหรับการตรวจสอบสัญญาณการร้องขออินเทอร์รัปต์จะตรวจสอบที่ตำแหน่ง State5P2 ของทุกๆเมกซีนไซเคิลเมื่อพบแล้วในช่วงเมกซีนไซเคิลที่ 2 จะเป็นการตรวจสอบว่าเป็นของอุปกรณ์ใด เมกซีนไซเคิลที่ 3 จะกระโดดไปทำบริการอินเทอร์รัปต์ (Interrupt Service Routine; ISR) ดังแสดงค่า Vector ดังตารางที่ 2.7 อินเทอร์รัปต์ของ MCS-51 ได้มาจาก 6 แหล่งและมีเพียง 5 Vector สำหรับ 8031,8051 สำหรับ 8032,8052 ขึ้นไปจะมี 8 แหล่ง 6 Vector (เพิ่ม Timer/Count 2 เข้ามา) รูปที่ 2.16 แสดงแหล่งที่มาของอินเทอร์รัปต์ จากอุปกรณ์ต่างๆซึ่งมีทั้งภายใน ชิปและภายนอกชิป

ลำดับ	ชื่อสัญญาณอินเทอร์รัปต์	Vector Address	Prioritty
1	IE0 (จากขา /INT0)	0003H	Highest ↑ ↓ Lowest
2	TF0 (Timer / Counter0)	000BH	
3	IE1(จากขา /INT1)	0013H	
4	TF1 (Timer / Counter1)	001BH	
5	TI+RI(จาก Serial Port)	0023H	
6	TF2+EXF2 (จากTimer / Count2)	002BH	

ตารางที่ 2.7 แสดงชื่อสัญญาณอินเทอร์รัปต์ (TF2+EXF2 มีเฉพาะ 8052 ขึ้นไป)

- INT0

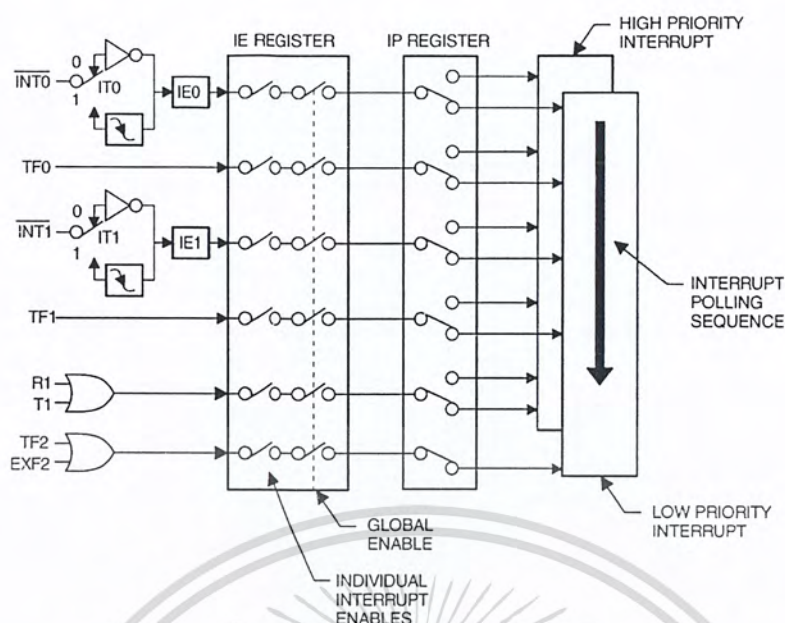
สัญญาณอินเทอร์รัปต์จากภายนอกทางขาสัญญาณ P3.2
- INT1

สัญญาณอินเทอร์รัปต์จากภายนอกทางขาสัญญาณ P3.3
- Timer0

สัญญาณการเกิด โอเวอร์โฟลว์ของ Timer0
- Timer1

สัญญาณการเกิด โอเวอร์โฟลว์ของ Timer1
- Serial Port

การเกิดอินเทอร์รัปต์ที่เกิดขึ้นจากการรับ/ส่งข้อมูลอนุกรมทำให้มีผลต่อแฟล็กอินเทอร์รัปต์ RI และ TI ตามลำดับ



รูปที่ 2.16 แสดงแหล่งที่มาและแผนภาพโครงสร้างของระบบการอินเทอร์รัปต์ของ 8051

ลักษณะการอินเทอร์รัปต์เป็นการขัดจังหวะการทำงานอย่างใดอย่างหนึ่งซึ่งกำลังดำเนินอยู่เหตุการณ์คล้ายกับการที่เรากำลังคุยโทรศัพท์กับเพื่อนของเราอยู่ แล้วมีเสียงกริ่งโทรศัพท์ดังขึ้นมาซึ่งจะมีผลให้เกิดการหันเหความสนใจของเราในการสนทนาไปยังเครื่องโทรศัพท์แทน การจัดการต่อเหตุการณ์นี้จะทำได้หลายลักษณะ เช่น

1. ไม่สนใจกับเสียงกริ่งโทรศัพท์ที่เกิดขึ้น และยังคงดำเนินการสนทนาต่อไปเช่นเดิม
2. หยุดพักการสนทนากับเพื่อนไว้ชั่วขณะ และยกหูโทรศัพท์ขึ้นสนทนา
3. ขอบการสนทนากับเพื่อนโดยทันที และยกหูโทรศัพท์ขึ้นสนทนา

ตัวอย่างข้างต้นแสดงให้เห็นถึงลักษณะของการอินเทอร์รัปต์ได้เป็นอย่างดีโดยเสียงกริ่งโทรศัพท์ทำหน้าที่เป็นสัญญาณอินเทอร์รัปต์ติดต่อเข้ามาหาเรา ซึ่งกำลังดำเนินงานอื่นอยู่และแม้ว่าเราจะจัดการตอบสนองการอินเทอร์รัปต์ นั้นในลักษณะไหนก็ตามจนเสร็จแล้ว ก็จะต้องกลับมาดำเนินงานที่ค้างต่อไปเช่นเดิม

2.9.1 ประเภทของการอินเทอร์รัปต์

ไมโครคอนโทรลเลอร์ 8051 สามารถเกิดการอินเทอร์รัปต์ โดยจำแนกตามแหล่งที่มาของสัญญาณ (Signal Source) ของสัญญาณอินเทอร์รัปต์นั้นๆ ได้แก่

2.9.2 สัญญาณอินเทอร์รัปต์จากภายนอก (External Interrupt)

8051 สามารถตอบสนองสัญญาณอินเทอร์รัปต์จากภายนอกได้สองสัญญาณ ทางขาสัญญาณ INT1 และ INT2 การตรวจสอบสัญญาณที่เข้ามาอินเทอร์รัปต์นี้ จะสามารถกำหนดให้มีการตรวจสอบใน

ลักษณะเมื่อได้มีการเปลี่ยนแปลงระดับสัญญาณ (Level-sensitive) ไปแล้ว หรือในช่วงเวลาขณะเริ่มมีการเปลี่ยนแปลงสัญญาณจากลอจิกสูงไปเป็นลอจิกต่ำ(Edge-sensitive)

2.9.3 สัญญาณอินเตอร์รัปต์ภายใน (Internal Interrupt)

แหล่งกำเนิดสัญญาณนี้จะป็นวงจรภายใน เช่น วงจรนับ/จับเวลา วงจรเชื่อมต่อสัญญาณอนุกรม เป็นต้น

ในการกำหนดให้ 8051 สามารถรับอินเตอร์รัปต์แต่ละประเภท ทำได้โดยการกำหนดบิตของข้อมูลที่เกี่ยวข้อง ซึ่งอยู่ภายในรีจิสเตอร์ TCON และ SCON หากว่าได้มีการกำหนดค่าของบิตซึ่งอยู่ภายในรีจิสเตอร์ IE (Interrupt Enable) ด้วยแล้ว ก็จะสามารถตอบรับการอินเตอร์รัปต์นั้นๆ ได้นอกจากนี้แล้วในรูปที่ 2.16 แสดงให้เห็นสัญญาณอินเตอร์รัปต์แต่ละประเภท

2.9.4 การควบคุมอินเตอร์รัปต์

ตามโครงสร้างด้านการจัดการอินเตอร์รัปต์ของ 8051 สามารถกำหนดเลือกเพื่อยินยอมหรือไม่ยินยอม (Enable/Disable) ให้มีการอินเตอร์รัปต์ของแต่ละสัญญาณได้ โดยใช้วิธีการกำหนดค่าของบิตภายในรีจิสเตอร์ IE ซึ่งจะมีทั้งแบบที่ระบุถึงอินเตอร์รัปต์โดยรวมทั้งหมด (บิตที่ 7) และอินเตอร์รัปต์แต่ละประเภทได้ ในกรณีที่กำหนดค่าข้อมูลเป็น 1 ให้กับบิตจะมีความหมายถึงการยินยอมให้มีการอินเตอร์รัปต์แต่ละประเภทได้ และจะเป็นกรณีตรงข้ามกันสำหรับการกำหนดค่าข้อมูลที่ป็น 0 ในการกำหนดต้องทำการกำหนดให้ยินยอมการอินเตอร์รัปต์ทั้งหมดให้เกิดขึ้นก่อน จึงจะมีผลทำให้การกำหนดบิตเพื่อยินยอมของแต่ละอินเตอร์รัปต์มีผลขึ้นได้

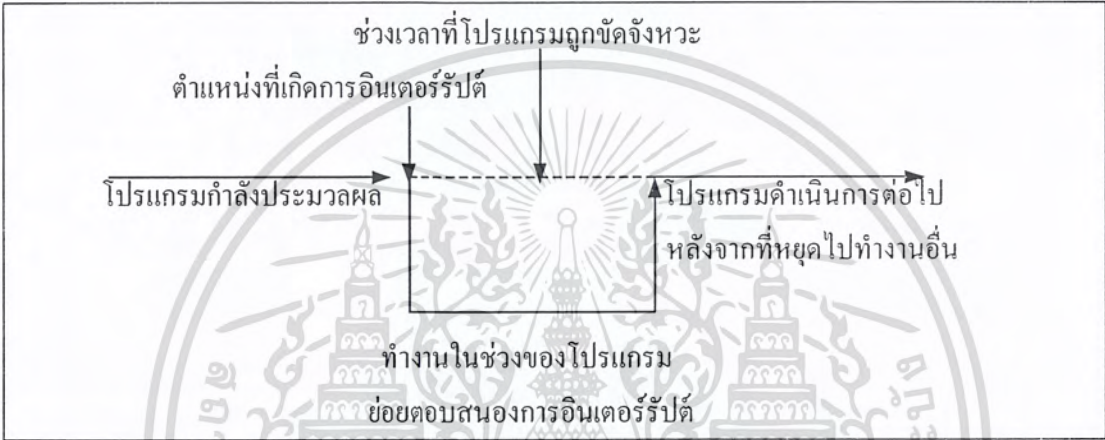
ชื่อบิต: IE		ตำแหน่ง A8h		ค่ากำหนดเริ่มต้น 0X00 0000			
EA	-	ET2	ES	ET1	EX1	ET0	EX0

ชื่อบิต	ตำแหน่ง	ความหมาย
EA	IE.7	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์โดยรวม
-	IE.6	
ET2	IE.5	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer 2
ES	IE.4	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์พอร์ตอนุกรม
ET1	IE.3	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer1
EX1	IE.2	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INT1
ET0	IE.1	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer 0
EX0	IE.0	อินาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INT0

ตารางที่ 2.8 บิตต่างๆภายในรีจิสเตอร์ IE (Interrupt Enable)

แหล่งกำเนิดสัญญาณ	สัญญาณ	ตำแหน่งแอดเดรส (HEX)
IE0	อินเทอร์รัปต์ภายนอก 0	0003H
TF0	วงจรรนับ/จับเวลา 0	000BH
IE1	อินเทอร์รัปต์ภายนอก 1	0013H
TF1	วงจรรนับ/จับเวลา 1	001BH
RI หรือ TI	วงจรรับ/ส่งข้อมูลอนุกรม	0023H

ตารางที่ 2.10 แสดงแอดเดรสของอินเทอร์รัปต์เวกเตอร์

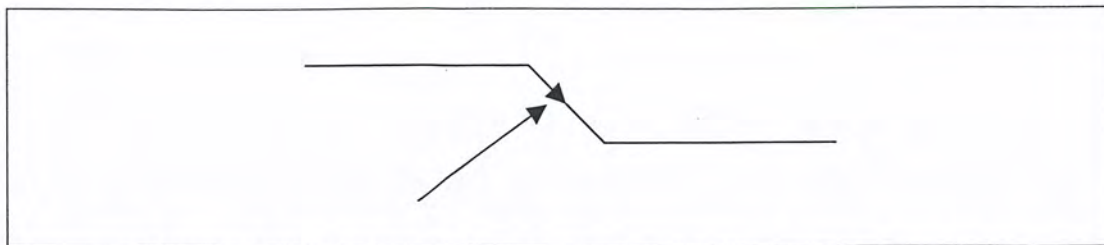


รูปที่ 2.17 แสดงให้เห็นแผนภาพการทำงานของโปรแกรมเมื่อมี Interrupt เข้ามา

ซึ่งโดยปกติจะทำงานอย่างใดอย่างหนึ่งอยู่ เมื่อมีการอินเทอร์รัปต์เกิดขึ้น โปรแกรมส่วนนี้จะหยุดค้างอยู่ เนื่องจากโปรเซสเซอร์จะทำการกระโดดไปทำงานในส่วนของโปรแกรมน้อยบริการอินเทอร์รัปต์ เมื่อเสร็จงานในโปรแกรมน้อยนี้แล้วจึงจะได้กลับไปทำงานเก่าที่ค้างอยู่ต่อไป

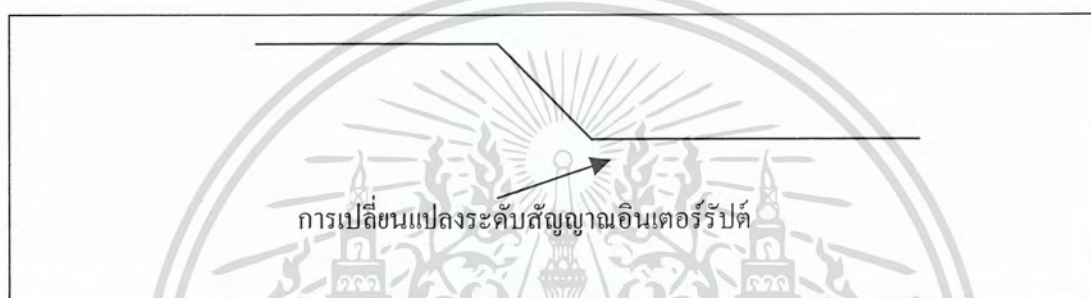
2.9.7 สัญญาณอินเทอร์รัปต์จากภายนอก (External Interrupt)

8051 สามารถตอบสนองสัญญาณอินเทอร์รัปต์จากภายนอกได้สองสัญญาณ ทางขาสัญญาณ INT0 และ INT1 โดยการรับรู้ถึงสัญญาณอินเทอร์รัปต์สามารถเลือกโปรแกรมให้เกิดขึ้นที่บริเวณช่วงขอบของสัญญาณที่เปลี่ยนแปลงจากระดับลอจิกสูงไปเป็นลอจิกต่ำ (high-to-low transition) หรือ ขณะที่ระดับของสัญญาณได้เปลี่ยนแปลงเป็นระดับลอจิกต่ำ (Low level interrupt) ดังแสดงในรูปที่ 2.18 และ รูปที่ 2.19



รูปที่ 2.18 การอินเทอร์รัปต์ทำงานที่ระดับสัญญาณขอบขาของสัญญาณอินเทอร์รัปต์

รูปที่ 2.18. การอินเทอร์รัปต์ทำงานที่ขอบสัญญาณจากลอจิกสูงเป็นลอจิกต่ำโดยจะต้องค้างสัญญาณที่ระดับลอจิกต่ำนี้เป็นเวลานานมากกว่า 12 คาบออสซิลเลเตอร์ และโปรเซสเซอร์จะต้องตรวจพบสัญญาณนี้ในช่วงเวลาใกล้กับขอบขาของสัญญาณ ALE



รูปที่ 2.19. การอินเทอร์รัปต์ทำงานที่ระดับสัญญาณลอจิกต่ำ

การอินเทอร์รัปต์ทำงานที่ระดับสัญญาณลอจิกต่ำ โปรเซสเซอร์จะต้องตรวจพบสัญญาณนี้ในช่วงเวลาใกล้กับขอบขาของสัญญาณ ALE อย่างไรก็ตามสัญญาณนี้จะถูกลบทิ้งก่อนที่เสร็จสิ้นการประมวลผลในโปรแกรมย่อยอินเทอร์รัปต์ ทั้งนี้เพื่อหลีกเลี่ยงการเกิดอินเทอร์รัปต์ซ้ำขึ้น

การเลือกประเภทของสัญญาณอินเทอร์รัปต์ภายนอก INT1 ใช้การกำหนดค่าข้อมูลที่ตำแหน่งบิตที่ 2 ภายในรีจิสเตอร์ TCON ซึ่งเป็นตำแหน่งบิต IT1 โดยทำหน้าที่เป็นตัวกำหนดประเภทของสัญญาณอินเทอร์รัปต์ของ INT1 ถ้ามีค่าเป็น 0 จะเป็นลักษณะแบบการทำงานที่ระดับสัญญาณในกรณีที่บิตนี้ได้รับการกำหนดให้มีค่าเป็น 1 ซึ่งเป็นลักษณะของการทำงานที่ขอบขาของสัญญาณจะมีแฟล็กสถานะ IE1 (interrupt-edge flag1) ซึ่งเป็นบิตภายในรีจิสเตอร์ TCON เข้ามาเกี่ยวข้องด้วย กล่าวคือเมื่อใดก็ตามที่สัญญาณ INT1 มีการเปลี่ยนแปลงระดับสัญญาณจากลอจิกสูงเป็นลอจิกต่ำแล้วแฟล็กสถานะ IE1 นี้ก็จะถูกกำหนดให้มีค่าเป็น 1 เช่นกัน

สำหรับการทำงานของสัญญาณอินเทอร์รัปต์ INT0 ก็มีลักษณะเดียวกับสัญญาณอินเทอร์รัปต์ INT1 ข้างต้นเช่นกัน โดยบิต 0 ภายในรีจิสเตอร์ TCON เป็นตำแหน่งของบิต IT0 และตำแหน่งของแฟล็กสถานะ IE0 จะอยู่ที่ตำแหน่งบิต 1 ภายในรีจิสเตอร์ TCON

ชื่อบิต : TCON ตำแหน่ง 88h ค่าเริ่มต้น: 0000 0000

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

ชื่อบิต	ตำแหน่ง	ความหมาย
TF1	TCON.7	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 1
TR1	TCON.6	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 1
TF0	TCON.5	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 0
TR0	TCON.4	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 0
IE1	TCON.3	แฟล็กแสดงการอินเตอร์รัปต์ของ INT1
IT1	TCON.2	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT1
IE0	TCON.1	แฟล็กแสดงการอินเตอร์รัปต์ของ INT0
IT0	TCON.0	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT0

ตารางที่ 2.11 บิตต่างๆภายในรีจิสเตอร์ TCON ที่เกี่ยวข้องกับการอินเตอร์รัปต์ภายนอก

2.9.8 วงจรนับ/จับเวลา

8051 ประกอบด้วยรีจิสเตอร์ขนาด 16 บิตจำนวนสองตัว คือ T0 (Timer 0) และ T1 (Timer1) ซึ่งสามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถควบคุมให้ทำหน้าที่เป็นตัวนับ/จับเวลา(Timer)เพื่อนับจำนวนพัลส์สัญญาณนาฬิกาภายในหรือควบคุมให้ทำหน้าที่เป็นตัวนับ (Counter) เพื่อนับจำนวนของพัลส์ของระบบได้

ภายในรีจิสเตอร์แต่ละตัวยังสามารถแยกออกได้เป็นรีจิสเตอร์ขนาด 8 บิตคือ TH0 กับ TL0 สำหรับรีจิสเตอร์ T0 และ TH1 และ TL1 สำหรับรีจิสเตอร์ T1 โดยการทำงานของรีจิสเตอร์ทั้งสองตัวนี้มีผลมาจากการกำหนดค่าของบิตที่อยู่ภายในรีจิสเตอร์ TMOD (Timer Mode Control Register) และรีจิสเตอร์ TCON (Timer/Counter Register)

ตามโครงสร้างภายในรีจิสเตอร์ TMOD ตารางที่ 2.12 จะเห็นได้ว่าการจัดแบ่งออกเป็นสองส่วนอย่างชัดเจน โดยบิตจำนวน 4 บิตทางด้านบนจะเป็นของรีจิสเตอร์ T1 และ 4 บิตที่เหลือทางด้านล่างจะเป็นของรีจิสเตอร์ T0 การกำหนดประเภทการทำงานทำได้โดยการกำหนดค่าภายในบิต C/T ที่ตำแหน่งบิต 6 และ 2 โดยหากเป็นข้อมูลที่มีค่า 0 จะทำหน้าที่เป็นตัวจับเวลาและหากมีค่าเป็น 1 จะทำหน้าที่เป็นตัวนับสัญญาณ

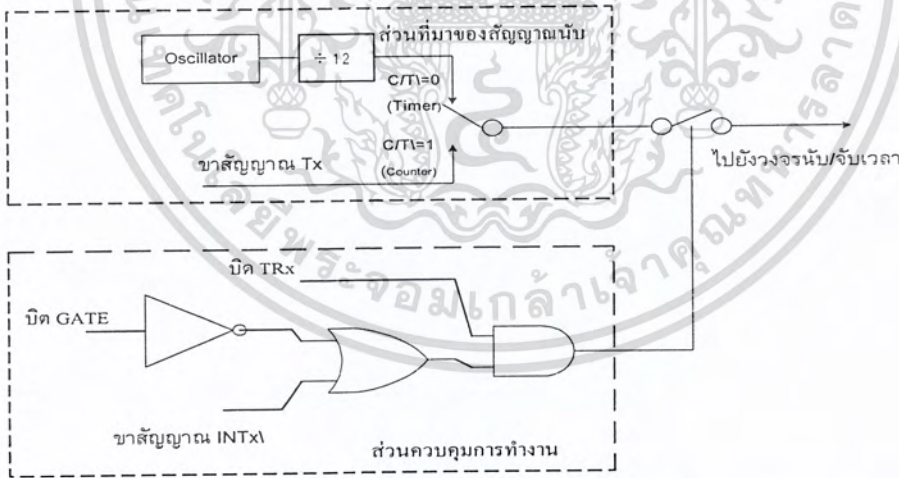
การกำหนดค่าของแฟล็กเพื่อแจ้งให้ซีพียูได้รับทราบ ดังนั้นโปรแกรมทั่วไปจึงมักจะใช้สถานะของแฟล็กนี้ (TF0 และ TF1) ซึ่งเป็นบิตอยู่ภายในรีจิสเตอร์ TCON (ดูตาราง 2.13) เพื่อตรวจสอบว่ากระบวนการได้นับเสร็จสิ้นลงแล้ว หรือใช้เพื่อทำการอินเตอร์รัปต์โปรแกรมต่อไป

ชื่อ TCON ตำแหน่ง : 88h ค่าเริ่มต้น 0000 0000

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

ชื่อบิต	ตำแหน่ง	ความหมาย
TF1	TCON.7	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 1
TR1	TCON.6	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 1
TF0	TCON.5	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 0
TR0	TCON.4	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 0
IE1	TCON.3	แฟล็กแสดงการอินเตอร์รัปต์ของ INT1
IT1	TCON.2	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT1
IE0	TCON.1	แฟล็กแสดงการอินเตอร์รัปต์ของ INT0
IT0	TCON.0	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT0

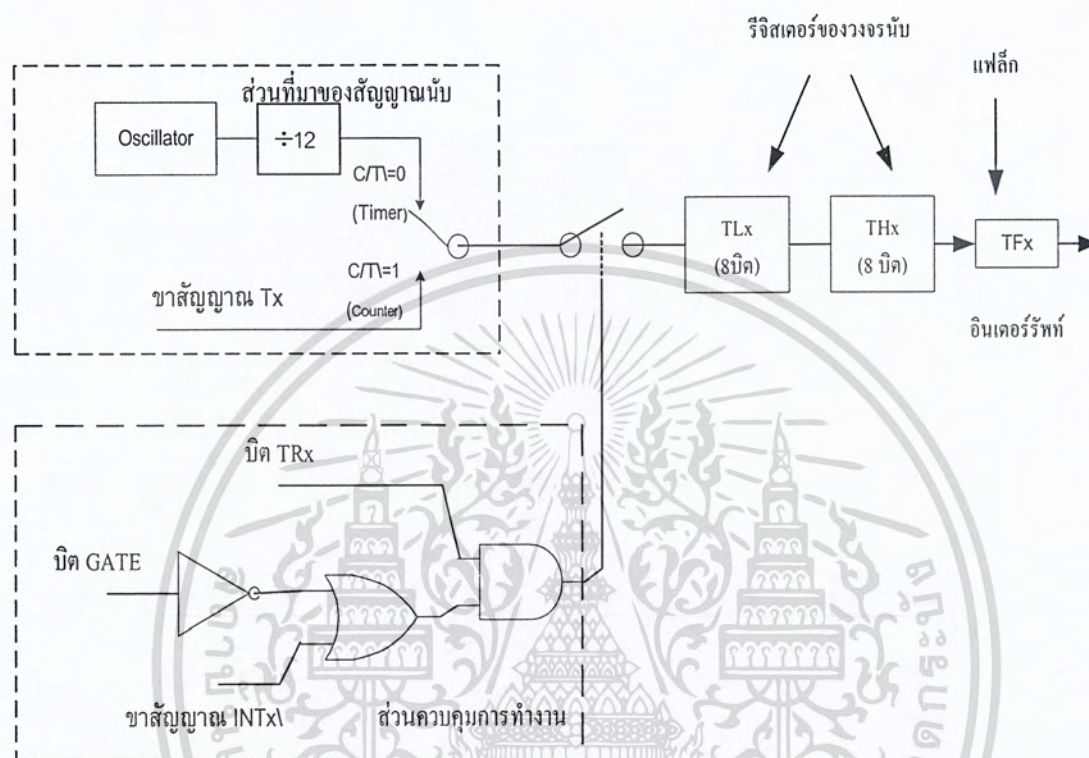
ตารางที่ 2.13 บิตต่างๆภายในรีจิสเตอร์ TCON



รูปที่ 2.20 ส่วนควบคุมการทำงานของวงจรนับ/จับเวลาซึ่งประกอบด้วยส่วนของการกำหนดที่มาของสัญญาณ(Timer หรือ Counter) และบิตหรือขาสัญญาณสำหรับการหยุดหรือทำงานของวงจรนับ

2.10.2 การจับเวลาในโหมด 1

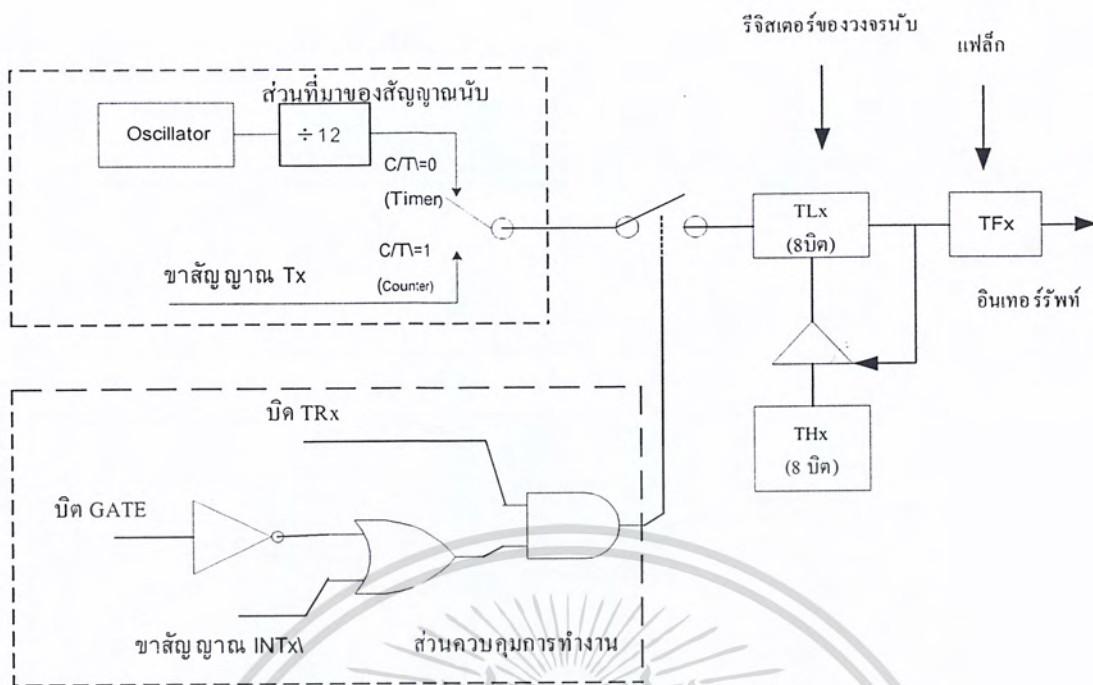
การทำงานในโหมด 1 มีความคล้ายคลึงกับโหมด 0 มาก เพียงแต่ต่างกันเฉพาะจำนวนบิตการนับเท่านั้น ซึ่งรีจิสเตอร์ของวงจรรนับ/จับเวลาจะถูกใช้เต็มทุกบิตในลักษณะของการนับแบบ 16 บิต ส่วน โดยค่าภายในรีจิสเตอร์ TH0 (หรือ TH1) จะเก็บค่าไบต์บน (High-order byte) ของตัวเลขตามแผนภาพการทำงานในรูปที่ 2.22



รูปที่ 2.22 การทำงาน ของ Timer/Counter 0 หรือ 1 ในโหมด 1

2.10.3 การจับเวลาในโหมด 2

การทำงานของโหมด 2 ในการนับ/จับเวลาจะมีความพิเศษต่างออกไป กล่าวคือจะมีเพียงการใช้รีจิสเตอร์ TL0 (หรือ TL1) เป็นตัวนับขนาด 8 บิตเท่านั้น ส่วนรีจิสเตอร์ TH0 (หรือ TH1) ใช้สำหรับทำหน้าที่เก็บค่าเริ่มต้นของการนับไว้ เมื่อรีจิสเตอร์ TL0 (หรือ TL1) เกิดการโอเวอร์โฟลว์จากค่า 0FFH เป็น 00H ระบบจะทำการนำค่าจากรีจิสเตอร์ TH0 (หรือ TH1) กลับมาใส่ให้โดยอัตโนมัติ (Automatic reload) ดังแสดงในรูปที่ 2.23

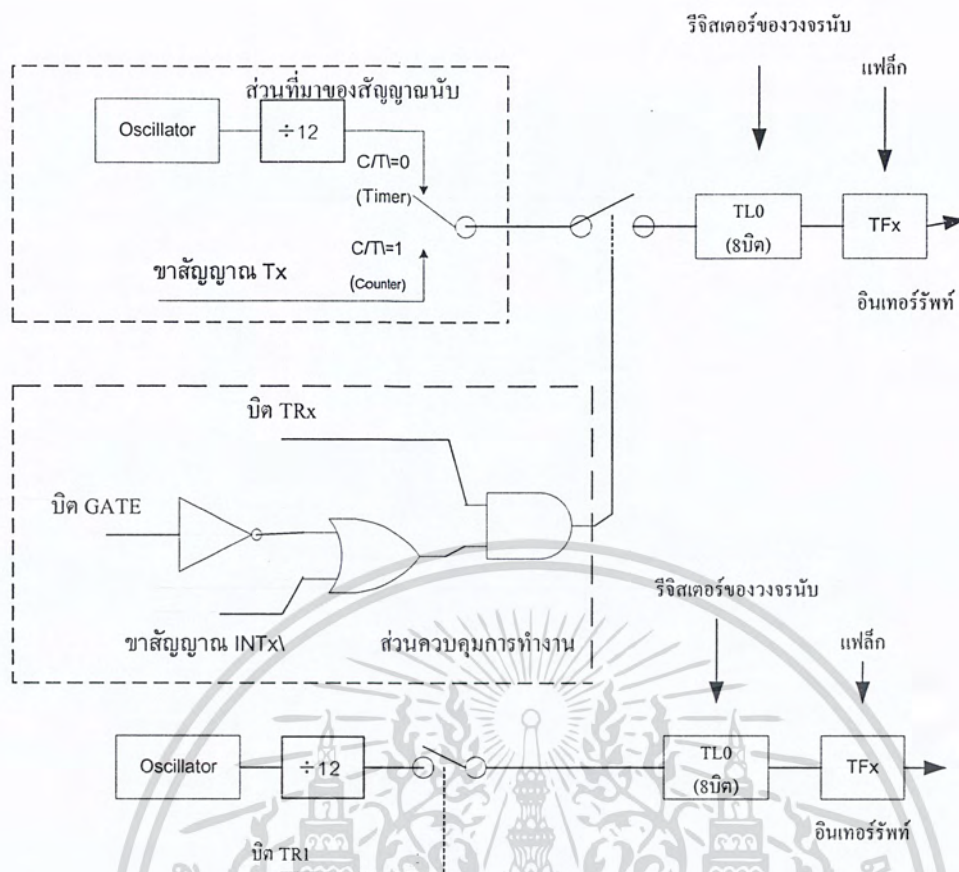


รูปที่ 2.23 การทำงาน ของ Timer/Counter 0 หรือ 1 ในโหมด 2

2.10.4 การจับเวลาในโหมด 3

การทำงานในโหมด 3 จะใช้ได้เฉพาะกับ Timer 0 เท่านั้น หากว่านำไปกำหนดให้ Timer 1 จะทำให้หยุดการทำงานไป เมื่อ Timer 0 ได้รับการกำหนดทำงานในโหมด 3 จะมีผลทำให้รีจิสเตอร์ของมันแยกกันทำงานเป็นอิสระ โดยรีจิสเตอร์ TL0 จะถูกควบคุมจากบิตภายในรีจิสเตอร์ TCON และขาสัญญาณอินเตอร์รัปต์ INT0 ดังแสดง ในรูปที่ 2.24 และเมื่อมีการโอเวอร์โฟลว์เกิดขึ้นจากค่า 0FFh เป็น 00h ก็จะมีผลให้แฟล็ก TF0 มีการเปลี่ยนแปลงเกิดขึ้น สำหรับรีจิสเตอร์ TH0 จะถูกกำหนดให้ทำงานในแบบของตัวจับเวลา ภายใต้การควบคุมของบิต TR1 ในรีจิสเตอร์ TCON เท่านั้น และหากเกิดการโอเวอร์โฟลว์ขึ้นก็จะมีผลเฉพาะต่อแฟล็ก TF1

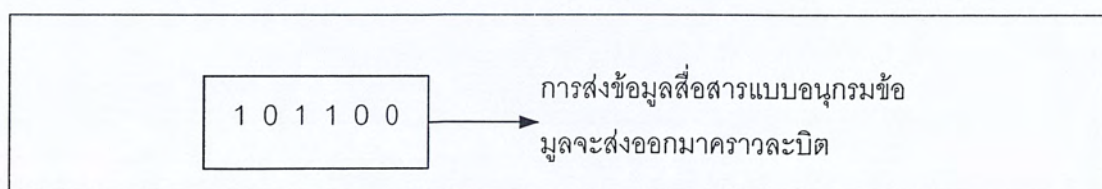
ในส่วนของ Timer 1 ขณะเมื่อ Time 0 ถูกให้ทำงานในโหมด 3 ก็ยังสามารถทำงานในโหมดอื่นๆที่ไม่ใช่โหมด 3 ได้เช่นเดิม ยกเว้นจะไม่มีการอินเตอร์รัปต์เกิดขึ้นเท่านั้น (เนื่องจากแฟล็ก TF1 ถูกใช้โดย Timer 0 ไปแล้ว)



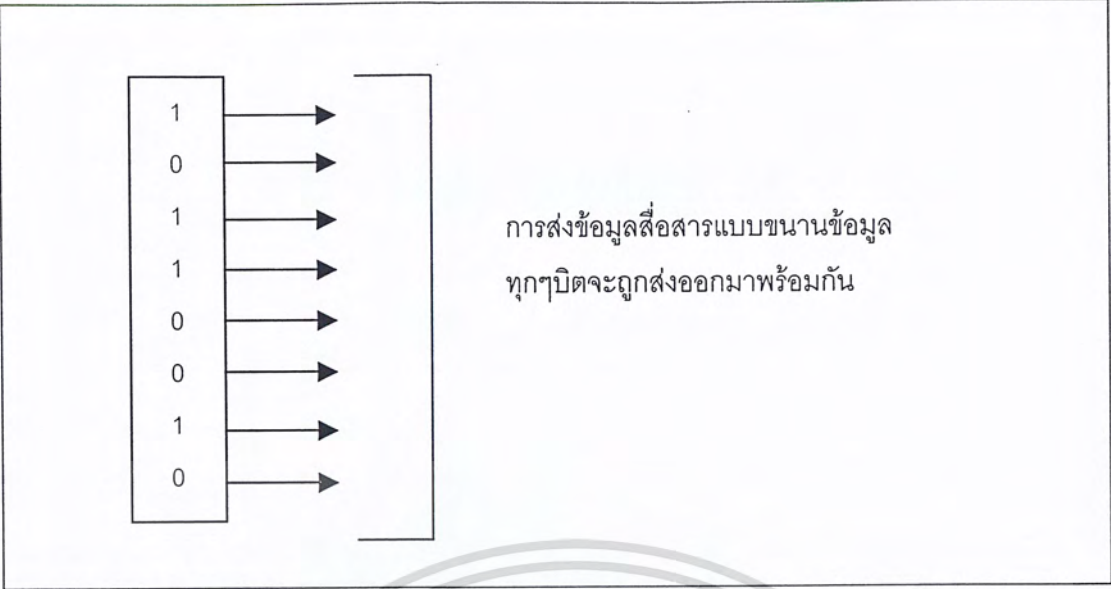
รูปที่ 2.24 วงจรนับ/จับเวลา 0 เมื่อทำงานโหมด 3

2.11 การอินเทอร์รัพท์จากการสื่อสารข้อมูลอนุกรม

การสื่อสารอนุกรมเป็นการรับหรือส่งข้อมูลในลักษณะของบิตหรือกลุ่มของบิตคราวละหนึ่งบิตเป็นลำดับเรื่อยไปจนสิ้นสุด การสื่อสารแบบนี้จะมีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมากเนื่องจากข้อมูลมีการโอนย้ายมาพร้อมกันจึงมีความจำเป็นต้องใช้เส้นสัญญาณมากขึ้นตามจำนวนบิตของข้อมูลด้วย ในขณะที่การสื่อสารแบบอนุกรมนั้นต้องการเส้นสัญญาณเพียงสองเส้นเท่านั้น ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะสมในการสื่อสารกับอุปกรณ์ภายนอกเป็นระยะทางไกลๆ เพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก ลองพิจารณาเปรียบเทียบการสื่อสารทั้งสองประเภทได้จาก รูปที่ 2.25 และรูปที่ 2.26



รูปที่ 2.25 แสดงการสื่อสารแบบอนุกรม



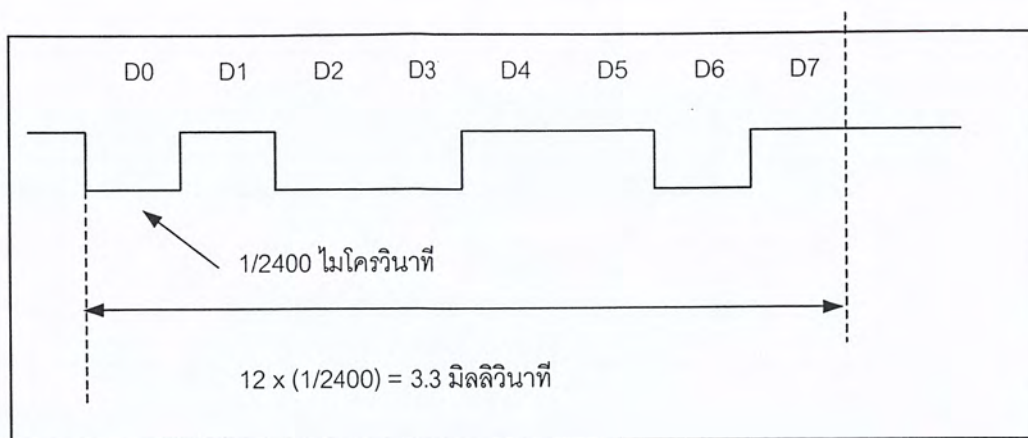
รูปที่ 2.26 แสดงการสื่อสารแบบขนาน

2.11.1 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม

เนื่องจากการสื่อสารข้อมูลแบบอนุกรมเป็นการรับ/ส่งข้อมูล ในลักษณะกลุ่มของบิตข้อมูล (Bit Stream) ดังนั้นจึงต้องให้ความสนใจในการพิจารณาถึงเรื่องของอัตราความเร็วในการรับ/ส่งบิตเหล่านี้เป็นลำดับแรก โดยทั่วไปมักจะระบุกันในหน่วยของจำนวนบิตข้อมูลภายในเวลาหนึ่งวินาทีเรียกว่า อัตราบอ์ดเรท (Baud Rate) ตามค่ามาตรฐานเหล่านี้ ได้แก่ 110, 150, 300, 1200, 2400, 4800 และ 9600 จากรูปที่ 2.27 แสดงให้เห็นลักษณะของรูปแบบสัญญาณข้อมูลอนุกรมที่ปรากฏในสายสัญญาณ ข้อมูลทั้ง 8 บิตนี้หากว่าถูกส่งออกมาด้วยอัตราบอ์ดเรท 2400 จะใช้เวลาในการส่งข้อมูล 1 บิตมีค่าเท่ากับ 1/2400 หรือ 416 ไมโครวินาที และเวลาในการส่งข้อมูลทั้ง 8 บิตมีค่าเท่ากับ (8 x 416) หรือ 3.328 ไมโครวินาที

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

รูปที่ 2.27 บิตข้อมูลสื่อสารแบบขนานภายในไบนารีข้อมูลที่จะถูกส่งออกมาในลักษณะแบบอนุกรม



รูปที่ 2.28 แสดงสัญญาณไฟฟ้าของข้อมูลในรูปที่ 2.26 ซึ่งเป็นการส่งข้อมูลแบบอนุกรมด้วยอัตราเร็ว 24000 บิต/วินาที

2.11.2 รูปแบบของข้อมูลอนุกรม

วิธีการที่จะทำให้ข้อมูลสื่อสารอนุกรมมีความถูกต้องมากยิ่งขึ้น จะใช้การเพิ่มเติมข้อมูลบางอย่าง ร่วมไปกับการส่งข้อมูลจริง ได้แก่

1. บิตเริ่มต้น (Start Bit) มีหน้าที่สำหรับการบ่งบอกให้วงจรฮาร์ดแวร์ทางด้านรับทราบถึง ตำแหน่งจุดเริ่มต้นของบิตข้อมูลกลุ่มใหม่เพื่อที่จะทำการปรับจังหวะของสัญญาณการรับข้อมูลให้ตรงกัน ดังนั้นบิตเริ่มต้นจึงจะถูกเพิ่มเข้าไปก่อนมีการส่งข้อมูลจริง ตามปกติแล้วค่าของบิตเริ่มต้นมักจะเป็นระดับลอจิกที่ตรงกันข้ามกับระดับลอจิกสถานะของสายสื่อสาร ขณะเมื่อไม่มีการส่งข้อมูล (Idle State) เช่น หากสถานะของสายเมื่อไม่มีข้อมูลเป็นลอจิกสูง บิตเริ่มต้นก็จะเป็นระดับลอจิกต่ำ
2. บิตแสดงสถานะความเป็นเลขคู่หรือเลขคี่ (Parity Bit) บิตนี้มีหน้าที่เพื่อ การตรวจสอบความถูกต้องของข้อมูล โดยทั่วไปมักเรียกว่า บิตพาริตี และจะนำไปแทรกต่อท้ายข้อมูล ค่าของบิตนี้ขึ้นอยู่กับค่าของบิตข้อมูลที่เป็น 1 ซึ่งจะเป็นได้สองลักษณะ คือ พาริตีคู่ (Even Parity) หรือ พาริตีคี่ (Odd Parity)

2.12 ชุดคำสั่งของ MCS-51

ไมโครคอนโทรลเลอร์ 8051 ประกอบด้วยคำสั่งทั้งหมดจำนวนมาก ซึ่งสามารถจะจัดกลุ่มคำสั่งเหล่านั้นตามลักษณะหน้าที่การทำงานที่คล้ายคลึงกัน เพื่อความสะดวกต่อการศึกษา ทำความเข้าใจและใช้งาน ดังนี้

1. กลุ่มคำสั่งในการโอนย้ายข้อมูล ทำหน้าที่ในการเคลื่อนย้ายข้อมูลระหว่างรีจิสเตอร์ หรือหน่วยความจำ
2. กลุ่มคำสั่งดำเนินการแบบลอจิก ทำหน้าที่เกี่ยวกับการประมวลผลแบบลอจิกต่างๆ เช่น การ AND OR หรือ EX-OR ระหว่างข้อมูลในรีจิสเตอร์ A กับรีจิสเตอร์ทั่วไป และการทำคำสั่งเกี่ยวกับการเลื่อนหรือหมุนบิตข้อมูลภายในรีจิสเตอร์ A เป็นต้น
3. กลุ่มคำสั่งการกระโดดไปยังตำแหน่งต่างๆ ภายในโปรแกรม

4. กลุ่มคำสั่งการคำนวณทางคณิตศาสตร์ เช่น การบวก ลบ คูณ และหารข้อมูลภายในรีจิสเตอร์ต่างๆ
5. กลุ่มคำสั่งการประมวลผลแบบบิต ซึ่งมีชุดคำสั่งที่จัดการโดยตรง

หากพิจารณาถึงชุดคำสั่งที่เกี่ยวข้องกับการโอนย้ายข้อมูลทั้งหมด สามารถจัดแบ่งได้เป็นสามกลุ่มหลัก ได้แก่ กลุ่มชุดคำสั่ง MOV กลุ่มชุดคำสั่ง PUSH กับ POP และกลุ่มคำสั่ง XCH อย่างไรก็ตามก่อนที่จะได้กล่าวลงไปในเรื่องละเอียดมากขึ้น ขออธิบายถึงลักษณะการบอกตำแหน่งของ 8051 เป็นลำดับแรกก่อน ทั้งนี้เพื่อที่จะได้สามารถนำข้อมูลหรือจัดเก็บข้อมูลได้อย่างมีประสิทธิภาพ

2.12.1 รูปแบบการอ้างตำแหน่งแอดเดรสของ 8051

การอ้างตำแหน่งของแอดเดรส (Addressing Model) เป็นวิธีการที่ไมโครคอนโทรลเลอร์จะสามารถระบุตำแหน่งของข้อมูลที่จะดำเนินการ เมื่อมีการใช้ชุดคำสั่งต่างๆ ได้ (โอเปอเรนด์ต้นทางและปลายทาง) ลักษณะการอ้างแอดเดรสของ 8051 มีทั้งหมด 5 ลักษณะ คือ

1. การอ้างแอดเดรสจากค่าตำแหน่งทันที (Immediate Addressing)
2. การอ้างแอดเดรสจากรีจิสเตอร์ (Register Addressing)
3. การอ้างแอดเดรสจากหน่วยความจำโดยตรง (Direct Addressing)
4. การอ้างแอดเดรสโดยอ้อมผ่านรีจิสเตอร์ (Register-indirect Addressing)
5. การอ้างแอดเดรสโดยอ้อมผ่านค่าฐานภายในรีจิสเตอร์รวมกับค่าดัชนีในอีกรีจิสเตอร์ (Base-register Plus index register Addressing)

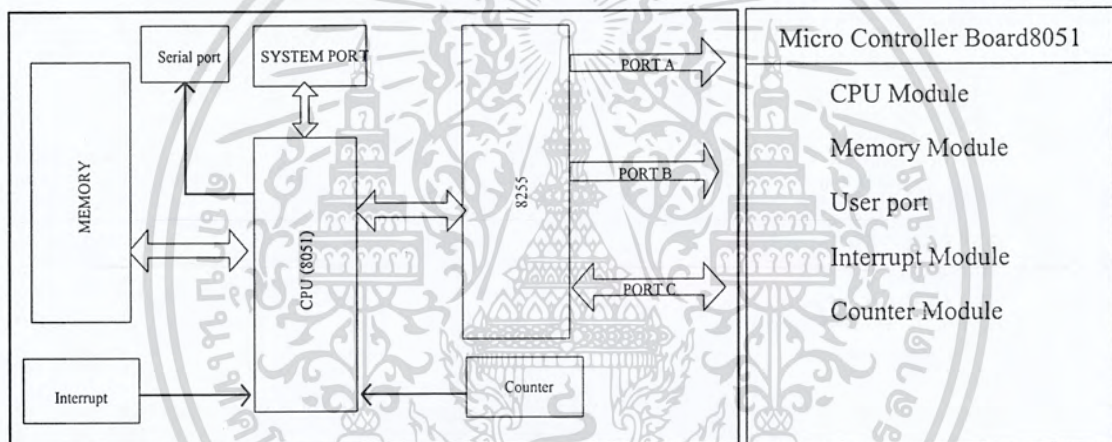
สำหรับชุดคำสั่งแบบต่างๆ ของ 8051 นั้นให้อ่านในภาคผนวก ก

บทที่ 3

การออกแบบฮาร์ดแวร์ (Hardware)

สำหรับการออกแบบฮาร์ดแวร์ของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองจะประกอบด้วย 2 บอร์ดคือ บอร์ดไมโครคอนโทรลเลอร์ 8051 (Microcontroller 8051 Board) และ บอร์ดทดลองอินพุตเอาต์พุต (Experiment Input / Output Board) ซึ่งแต่ละบอร์ดจะมีหน้าที่การทำงานที่แตกต่างกัน คือ บอร์ดไมโครคอนโทรลเลอร์ 8051 จะเป็นบอร์ดที่ใช้อินเตอร์เฟซ (Interface) กับบอร์ดทดลองอินพุตเอาต์พุต ซึ่งบอร์ดนี้จะมีไมโครคอนโทรลเลอร์ที่ใช้ไว้ในการประมวลผลต่าง ใช้ควบคุมอุปกรณ์อินพุตเอาต์พุตต่างๆ ซึ่งรายละเอียดของแต่ละบอร์ดมีดังนี้

3.1 บอร์ดไมโครคอนโทรลเลอร์ 8051 (Microcontroller Board 8051)

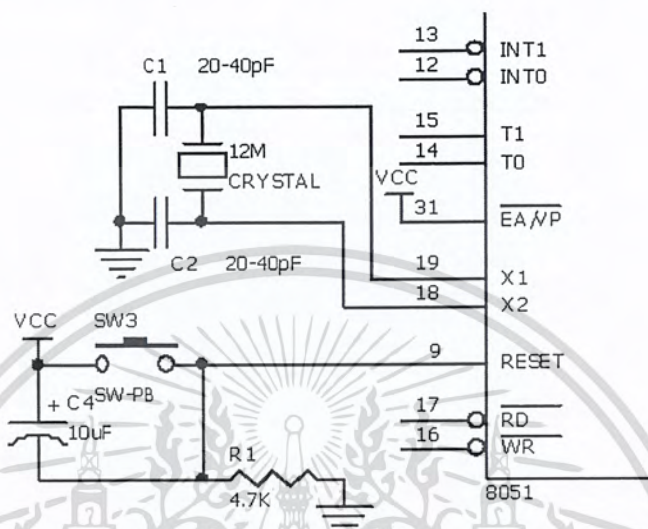


รูปที่ 3.1 Block diagram ของบอร์ดไมโครคอนโทรลเลอร์ 8051

ในบอร์ดไมโครคอนโทรลเลอร์ 8051 จะประกอบด้วยโมดูลต่างดังนี้ โมดูล ซีพียู ในนี้ไมโครคอนโทรลเลอร์จะทำหน้าที่เป็นซีพียู ในการประมวลผลต่างๆ และควบคุมอุปกรณ์ อินพุตเอาต์พุตต่างๆ ที่มีในบอร์ด ไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตเอาต์พุต, โมดูลหน่วยความจำ (MEMORY) เป็นโมดูลที่ใช้ไว้เก็บโปรแกรมในการทำงานและข้อมูลต่าง, โมดูลพอร์ตใช้งาน (USER PORT) เป็นโมดูลไว้ต่อใช้งาน และในที่นี้จะพอร์ตใช้งานจะต่อออกมาจาก 8255 สายเหตุที่ต่อมาจาก 8255 ก็เพราะว่าใช้ 8255 เป็นตัวที่ขยายพอร์ตใช้งานเนื่องจากว่าไมโครคอนโทรลเลอร์มีพอร์ตใช้งานไม่เพียงพอกับความต้องการ, System port เป็น โมดูลที่สร้างไว้ในกรณีที่ใช้ต้องการใช้พอร์ตจาก ระบบ, โมดูลพอร์ตสื่อสารข้อมูลอนุกรม (Serial port) เป็นโมดูลที่ใช้ไว้ในการกระโดดโปรแกรมจาก PC ลงที่บอร์ดไมโครคอนโทรลเลอร์, โมดูลอินเทอร์รัพท์ เป็นโมดูลที่มีไว้เพื่อสร้างอินเทอร์รัพท์จากภายนอก, โมดูลนับ/จับเวลา (Timer/Counter) เป็นโมดูลที่มีไว้เพื่อสร้างเป็นตัวนับ/จับเวลาจากภายนอก ซึ่งรายละเอียดของแต่ละโมดูลมีดังนี้

3.1.1 โมดูลซีพียู (CPU Module)

โมดูลซีพียูเป็นโมดูลหลักที่นำเอาไมโครคอนโทรลเลอร์ 8051 มาเป็น CPU ในการประมวลผลการทำงานของบอร์ด ซึ่งในโมดูลนี้จะประกอบด้วยวงจร ที่สร้างสัญญาณนาฬิกาเพื่อให้ซีพียูทำงาน เพราะซีพียูจะไม่ทำงานหากไม่มีสัญญาณนาฬิกา, และวงจร Reset ของตัวซีพียูเอง ดังรูปที่ 3.2



รูปที่ 3.2 แสดงโมดูลซีพียู

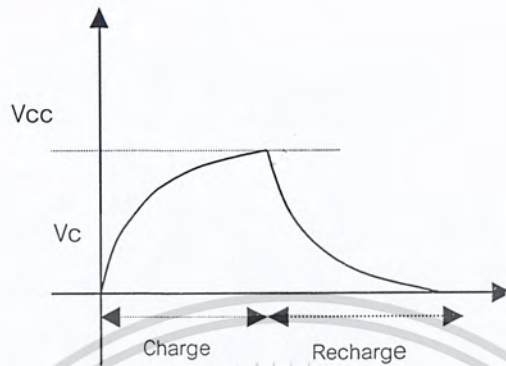
3.1.1.1 วงจรที่สร้างสัญญาณนาฬิกาให้แก่ซีพียู

เพื่อให้ซีพียูทำงานได้นั้นในบอร์ดนี้จำเป็นต้องมีการสร้างสัญญาณนาฬิกาให้กับซีพียูในการสร้างสัญญาณนาฬิกาให้กับซีพียูนั้นเราใช้ คริสตอลเป็นตัวกำเนิดสัญญาณตัวที่สร้างสัญญาณนาฬิกาออกมาให้กับซีพียูและใช้ คริสตอลที่มีความถี่ 11.0592 เมกะเฮิร์ตซ์ สาเหตุที่ใช้คริสตอลที่มีความถี่ 11.0592 เมกะเฮิร์ตซ์ ก็เพราะว่าสามารถนำค่าความถี่ที่ได้นี้ ไปใช้เป็นฐานเวลาสำหรับการสร้างความถี่ในการ รับ/ส่งข้อมูลอนุกรม โดยจะทำให้ได้ค่าที่ใกล้เคียงกับค่ามาตรฐานคือ 19200, 9600, 4800, 2400, และ 300 บิต/วินาที และในการที่จะให้ คริสตอลสร้างสัญญาณนาฬิกาออกมานั้นจะต้องต่อตัวเก็บประจุ C1 และ C2 ดังรูปที่ 3.24 ข้างบน โดย C1,C2จะมีค่าอยู่ระหว่าง 20 – 40 พิกโคฟาราด (pF) ดังรูปที่ 3.2

3.1.1.2 วงจรรีเซ็ต Reset เป็นวงจรที่ต่อให้กับซีพียู

ไมโครคอนโทรลเลอร์ 8051 ขา Reset จะ Active Hight, เพื่อทำการรีเซ็ตให้แก่ตัวซีพียูเองในเวลาที่เริ่มทำงานครั้งแรกนั้นจะต้องวงจร R และ C เข้าที่ขารีเซ็ตของไมโครคอนโทรลเลอร์ เพื่อทำให้ไมโครคอนโทรลเลอร์มีการรีเซ็ตตัวมันเองในครั้งแรกที่มีไฟเลี้ยงเข้าไป คือเมื่อเราเปิดสวิตช์ให้ไฟเลี้ยงแก่ไมโครคอนโทรลเลอร์จะมีไฟไหลเขาไปที่ผ่านตัวเก็บประจุและไหลเขาไปที่ขารีเซ็ตของไมโครคอนโทรลเลอร์ทำให้ไมโครคอนโทรลเลอร์รีเซ็ตตัวมันเอง และในเวลานี้ตัวเก็บประจุจะ Charge เมื่อตัวเก็บประจุ Charge เต็มมันก็จะทำการ Recharge ผ่านตัวความต้านทานลงกราวด์ ทำให้ไมโครคอนโทรล

เลอร์ หยุดการรีเซ็ตตัวมันเอง แต่ถ้าต้องการทำการรีเซ็ตให้แก่อุปกรณ์เพื่อเริ่มการทำงานใหม่หรือในเวลาที่กำลังทำงานอยู่แล้วเกิดการติดขัด หรือทำงานผิดพลาดก็สามารถที่จะทำการรีเซ็ตให้อุปกรณ์เริ่มการทำงานใหม่ได้โดยการกดสวิทช์รีเซ็ต ซึ่งจะทำให้มีไฟเลี้ยงที่มีค่าลอจิกเป็น 1 เข้าไปที่ขารีเซ็ตทำให้ไมโครคอนโทรลเลอร์รีเซ็ตตัวมันเอง รูปที่ 3.2



รูปที่ 3.3 การ Charge และ Recharge ของ C ที่ขารีเซ็ตของไมโครคอนโทรลเลอร์และทำให้เกิดการรีเซ็ต

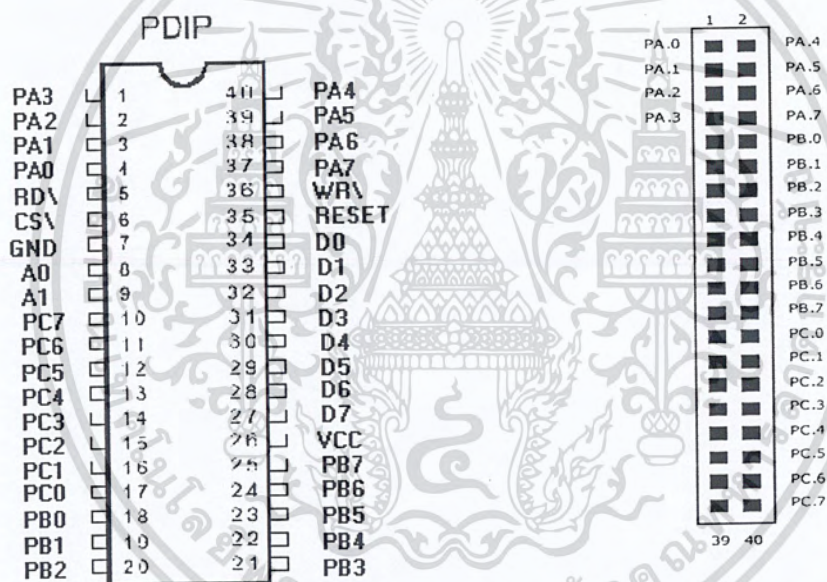
3.1.2 โมดูลหน่วยความจำ (MEMORY)

8051 สามารถที่จะอ้างอิงหน่วยความจำภายนอกที่ใช้เก็บโปรแกรมได้ 64 กิโลไบต์ และสามารถอ้างอิงหน่วยความจำภายนอกที่เก็บข้อมูลได้ 64 กิโลไบต์ หน่วยความจำที่ใช้เก็บข้อมูลซึ่งเป็นหน่วยความจำประเภทที่สามารถอ่านและเขียนได้ (RAM) หน่วยความจำข้อมูลจะแบ่งออกเป็น 2 ชนิดคือ Static RAM และ Dynamic RAM, Static RAM จะเป็นหน่วยความจำชนิดหนึ่งที่ย่อยต่อการใช้งานเป็นหน่วยความจำที่ไม่ต้องทำการ Refresh ตลอดเวลาแต่ราคาจะสูงเมื่อเทียบกับ Dynamic RAM และความจุจะต่ำ ส่วน Dynamic RAM นั้นราคาจะถูกกว่าจะมีความจุมากกว่า Static RAM แต่จะยุ่งยากกว่าในการนำไปใช้งานคือจะต้องได้ทำการ Refresh หน่วยความจำตลอดเวลา

ในบอร์ดไมโครคอนโทรลเลอร์ 8051 นี้จะใช้หน่วยความจำเบอร์ 62256 ซึ่งเป็นหน่วยความจำ RAM ขนาด 32 กิโลไบต์ เนื่องจากว่าบอร์ดทดลองของเรามีขนาดใหญ่และโปรแกรมที่ใช้ในการทดลองก็ไม่ใหญ่เท่าไรเพราะฉะนั้นเราจึงใช้ หน่วยความจำขนาด 32 กิโลไบต์ก็เพียงพอ และภายใน 32 กิโลไบต์นี้เราจะแบ่งเป็นสองส่วนคือส่วนที่เป็นหน่วยความจำสำหรับการเก็บโปรแกรมและส่วนที่เป็นหน่วยความจำที่ใช้เก็บข้อมูล โดยในการแบ่งหน่วยความจำนั้นได้ออกแบบไว้ให้ผู้ใช้งานเป็นคนแบ่งเอาเองว่าจะให้หน่วยความจำที่เก็บโปรแกรมนั้นอยู่ในช่วงไหนหาช่วงไหน และส่วนที่เหลือก็จะเป็นหน่วยความจำที่เก็บข้อมูล โดยที่หน่วยความจำนั้นจะเริ่มจากหลัง 4 กิโลไบต์เป็นต้นไปจนถึง 32 กิโลไบต์ หรือเริ่มจากตำแหน่งที่ 0800H ไปจนถึงตำแหน่ง 7FFFH รูปที่ 3.4

จากว่า 8255 เป็นไอซีที่ได้รับการออกแบบมาเพื่อทำหน้าที่เป็นพอร์ตสำหรับการรับ/ส่งข้อมูลแบบขนานระหว่างอุปกรณ์ภายนอกกับซีพียูหรือไมโครคอนโทรเลอร์

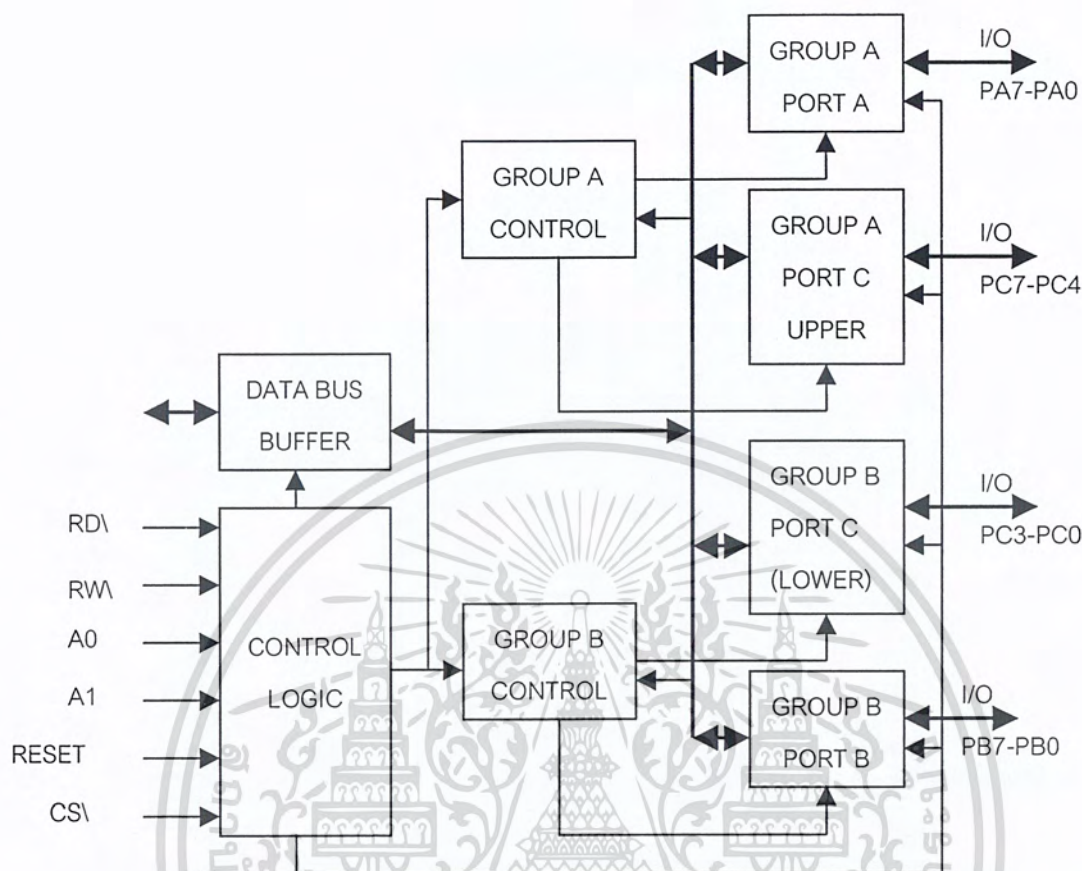
ข้อดีของการนำ 8255 มาใช้งาน คือมีพอร์ตใช้งานได้มากขึ้นและสามารถเปลี่ยนแปลงลักษณะการทำงานของพอร์ตให้เป็นการเอาต์พุตหรืออินพุตได้ โดยการโปรแกรมด้วยซอฟต์แวร์ทำให้มีความอ่อนตัวในการนำไปใช้งานตามความประสงค์มากขึ้น เนื่องจากความสะดวกเพียงการส่งข้อมูลควบคุม (Control Word) จากไมโครคอนโทรเลอร์ไปที่ 8255 ก่อนที่จะเริ่มการใช้งานเท่านั้น ความสามารถเช่นนี้เรียกว่า Programmable ก็สามารถโปรแกรมการทำงานได้ 8255 จะประกอบด้วยพอร์ตทั้งหมด 4 พอร์ตซึ่งจะเป็นพอร์ตอินพุต/เอาต์พุต 3 พอร์ตและ พอร์ตคอนโทรลอีก 1 พอร์ต สำหรับพอร์ตอินพุต/เอาต์พุตจะประกอบด้วยพอร์ต A (PA), B (PB) และ C (PC) อีก 1 พอร์ตเป็นพอร์ตคอนโทรล ที่ทำหน้าที่กำหนดการทำงานของพอร์ตทั้งสามว่า จะให้พอร์ตไหนทำงาน และทำงานอย่างไร ทำงานเป็นอินพุตหรือเอาต์พุตตาม Control Word และในการเลือกจะให้พอร์ตไหนทำงานนั้นจะใช้ขาแอดเดส A0,A1 ของไมโคร-คอนโทรเลอร์เป็นตัวเลือก รูปที่ 3.5 และ รูปที่ 3.6



ขาสัญญาณของไอซีเบอร์ 8255

แสดงตำแหน่งขาของคอนเน็กเตอร์ User port

รูปที่ 3.5 แสดงขาสัญญาณของไอซีเบอร์ 8255



รูปที่ 3.6 แสดงบล็อกภายในของไอซีเบอร์ 8255

8255 ได้จัดกลุ่มของพอร์ตออกเป็นสองกลุ่มคือกลุ่ม A จะประกอบด้วยพอร์ต PA จำนวน 8 บิต และพอร์ต PC จำนวน 4 บิต (เฉพาะ 4 บิตบนของพอร์ต) กลุ่ม B จะประกอบด้วยพอร์ต PB จำนวน 8 บิต และพอร์ต PC จำนวน 4 บิต (เฉพาะ 4 บิตล่างของพอร์ต)

สำหรับการโปรแกรม 8255 เพื่อใช้งานตามความต้องการนั้นเราสามารถที่จะโปรแกรมให้ 8255 ทำงานได้ 3 แบบ เนื่องจากว่า 8255 จะแบ่งการทำงานออกเป็น 3 โหมดคือ

1. โหมด 0 เป็นอินพุต/เอาต์พุตเบื้องต้น (Basic Input /Out put) ไม่มี Handshake
2. โหมด 1 โหมดนี้ใช้พอร์ต PA, PB ในการรับหรือส่งข้อมูล และใช้พอร์ต PC ในการตรวจสอบสัญญาณ Handshake
3. โหมด 2 โหมดนี้ใช้พอร์ต PA ในการรับส่งข้อมูล 2 ทิศทางและพอร์ต PB ในการรับหรือส่งข้อมูลและใช้พอร์ต PC บิต 0, 1, 2 ในการรับส่งข้อมูลบิตและบิต 3, 4, 5, 6, 7 เป็นสัญญาณ Handshake

โดยปกติเราจะใช้งานกันที่โหมด 0 ดังแสดงในรูปที่ 3.7

Mode Definition Summary

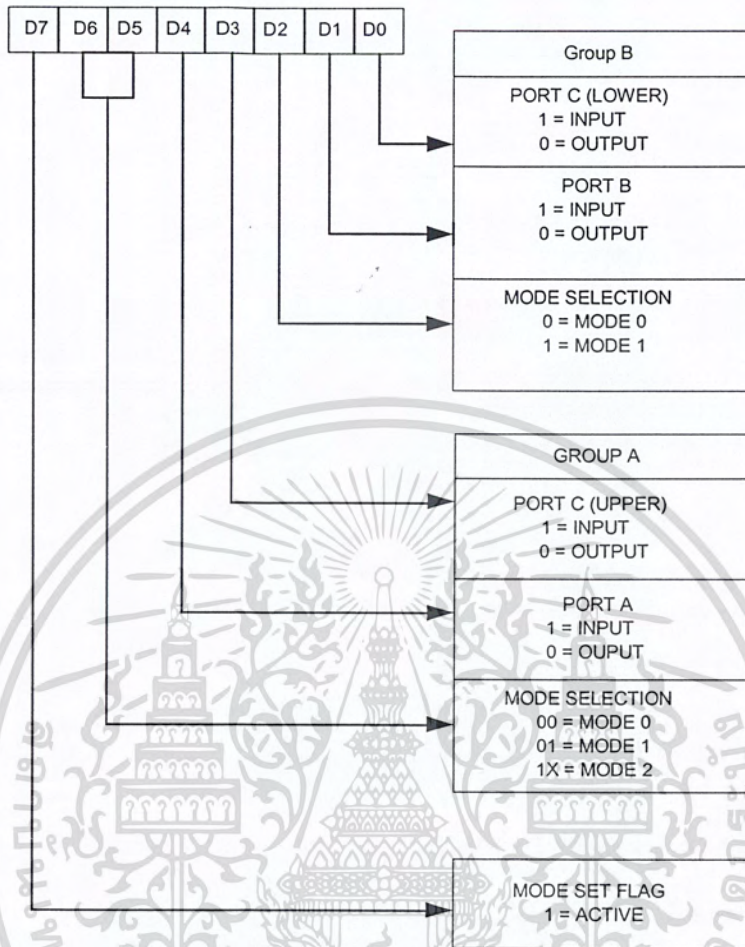
	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA ₀	IN	OUT	IN	OUT	↔
PA ₁	IN	OUT	IN	OUT	↔
PA ₂	IN	OUT	IN	OUT	↔
PA ₃	IN	OUT	IN	OUT	↔
PA ₄	IN	OUT	IN	OUT	↔
PA ₅	IN	OUT	IN	OUT	↔
PA ₆	IN	OUT	IN	OUT	↔
PA ₇	IN	OUT	IN	OUT	↔
PB ₀	IN	OUT	IN	OUT	—
PB ₁	IN	OUT	IN	OUT	—
PB ₂	IN	OUT	IN	OUT	—
PB ₃	IN	OUT	IN	OUT	—
PB ₄	IN	OUT	IN	OUT	—
PB ₅	IN	OUT	IN	OUT	—
PB ₆	IN	OUT	IN	OUT	—
PB ₇	IN	OUT	IN	OUT	—
PC ₀	IN	OUT	INTR _B	INTR _B	I/O
PC ₁	IN	OUT	IBF _B	OBFA	I/O
PC ₂	IN	OUT	STB _B	ACK _B	I/O
PC ₃	IN	OUT	INTR _A	INTR _A	INTR _A
PC ₄	IN	OUT	STB _A	I/O	STB _A
PC ₅	IN	OUT	IBF _A	I/O	IBF _A
PC ₆	IN	OUT	I/O	ACK _A	ACK _A
PC ₇	IN	OUT	I/O	OBFA	OBFA

MODE 0
OR MODE 1
ONLY

รูปที่ 3.7 แสดง Mode การทำงานของ 8255

สำหรับ CONTROL WORD ของ 8255 แสดงในรูปที่ 3.8 ในบอร์ดไมโครคอนโทรลเลอร์จะใช้ 8255 มาเพื่อทำการขยายพอร์ต 1 ตัว เนื่องจากว่าในบอร์ดทดลองอินพุตเอาต์พุตมีหลายโมดูลและเพื่อให้มีพอร์ตเพียงพอกับการใช้งานกับทุกๆโมดูล เพื่อให้ทุกๆโมดูลสามารถใช้งานได้โดยไม่สับสนเราจึงใช้ 8255 เพียง 1 ตัวโดยจะทำงานที่ตำแหน่งแอดเดสที่ 8000H เป็นต้นไป ในการ Decode Address เพื่อ 8255 ทำงานนั้นเราจะใช้ การโปรแกรม PAL เพื่อให้ PAL เป็นตัว Decode Address ให้ 8255 ทำงาน, ในการที่จะต่อพอร์ตของ 8255 ที่มีในบอร์ดไมโครคอนโทรลเลอร์ 8051 ไปใช้งานนั้นก็ สามารถที่จะออกไปได้จาก คอนเน็คเตอร์ 40 ขาที่เป็น USER PORT ของบอร์ดไมโครคอนโทรลเลอร์ ที่มีตำแหน่งพอร์ตที่ต่อกับขาต่างๆของคอนเน็คเตอร์ดังแสดงในรูปที่ 3.5

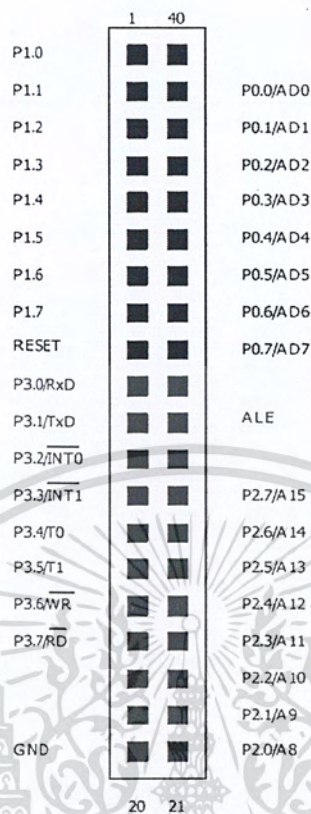
CONTROL WORD



รูปที่ 3.8 แสดง Command Word ของ 8255

3.1.3.2 พอร์ตของระบบ (System port)

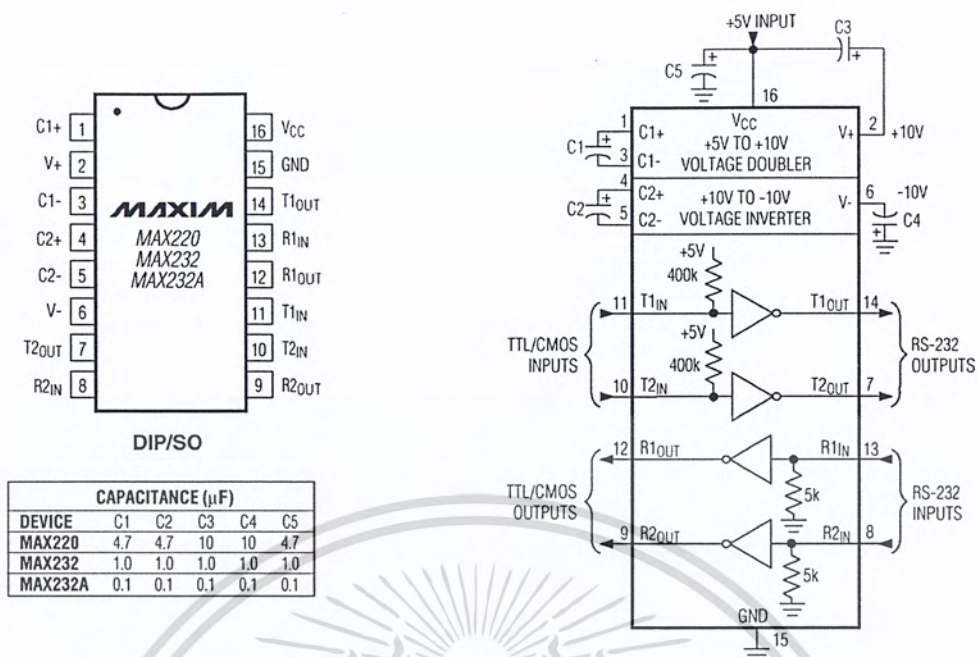
System port เป็นโมดูลที่ใช้ไว้เพื่อใช้งาน โดยที่โมดูลนี้จะต่อกับพอร์ตต่างที่มีในตัวไมโครคอนโทรลเลอร์โดยตรงออกมาเพื่อไว้ใช้งาน ก็จะมีพอร์ต P0, พอร์ตP1, พอร์ตP2 และ P3 โดยที่การต่อออกมาใช้งานนั้นจะต่อผ่านคอนเน็กเตอร์ที่มี 40 ขา และได้จัดตำแหน่งของขาเหล่านั้นให้ตรงกับตำแหน่งขาของไมโครคอนโทรลเลอร์ 8051. สำหรับการใช้งานในโมดูลนี้ให้เป็นพอร์อินพุตเอาต์พุตนั้นจะมีจุดคือสามารถที่จะอ้างอิงเป็นบิตได้ ดูรูปที่ 3.9



รูปที่ 3.9 แสดงคอนเนกเตอร์ของ System port

3.1.3.3 โมดูลพอร์ตสื่อสารข้อมูลอนุกรม (Serial Port)

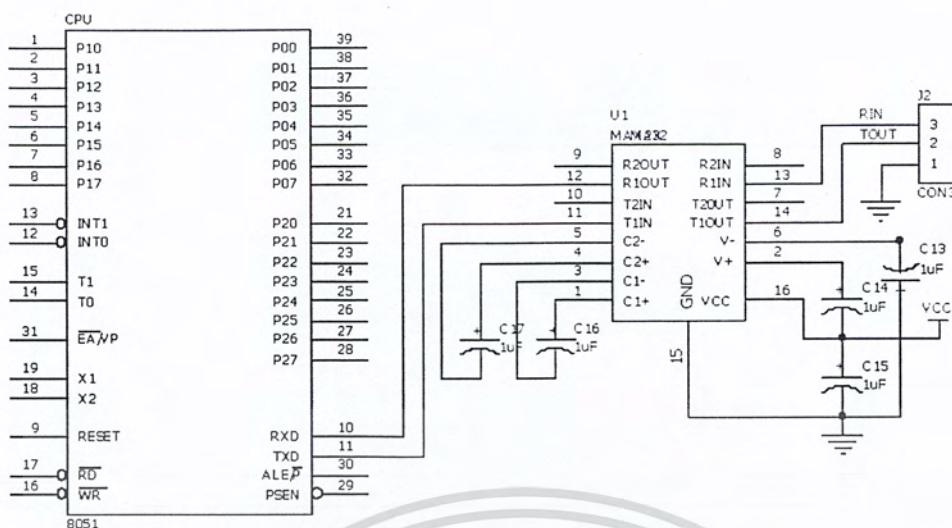
เป็นโมดูลที่ใช้ในการติดต่อระหว่างบอร์ดไมโครคอนโทรลเลอร์ 8051 และ PC เช่นในการ DowLoad Program หรืออื่นๆ ในการทำเป็นโมดูล Serial Port นี้มีไอซีหลายตัวที่เราสามารถนำมาทำเป็นโมดูลนี้ได้เช่น MAX223/230/232/ 234/236/237/238/240/241 ซึ่งเป็นไอซี Drivers/Receivers, เนื่องจากว่าระหว่าง PC และ ไมโครคอนโทรลเลอร์ใช้ไฟเลี้ยงที่มีระดับต่างกันดังนั้นเราจำเป็นต้องได้ทำการแปลงแรงไฟเหล่านั้นในการติดต่อกันระหว่าง PC และ ไมโครคอนโทรลเลอร์ มันจึงสามารถที่จะติดต่อกันได้ในบอร์ดไมโครคอนโทรลเลอร์ ในโมดูลนี้เราจะใช้ MAX232 ซึ่งเป็นไอซีที่มี 16 ขา มี Packet แบบ DIP ซึ่งมีชุดการทำงานอยู่ภายในสองชุดดังแสดงในรูปที่ 3.10



รูปที่ 3.10 แสดงรูปไอซี MAX232 และโครงสร้างภายในของไอซี MAX232

ไอซีตัวนี้จะทำหน้าที่ในการรับและส่งข้อมูลระหว่าง บอร์ดไมโครคอนโทรลเลอร์กับ PC และจะเป็นตัวที่เปลี่ยนแปลงแรงดันจาก 5 โวลต์ที่ถูกส่งออกมาจากบอร์ดไมโครคอนโทรลเลอร์คือขา TX ของ 8051 ให้เป็น 10 โวลต์เพื่อส่งให้ขา RX ของ PC และแปลงแรงดัน 10 โวลต์ที่ส่งมาจาก TX ของ PC ให้เป็น 5 โวลต์เพื่อส่งไปที่ขา RX ของบอร์ดไมโครคอนโทรลเลอร์ 8051 โดยที่อยู่ภายในตัว MAX232 นี้จะมีวงจรที่ทำการเปลี่ยนแรงดันจาก 5 โวลต์ เป็น 10 โวลต์ (Voltage Double) และวงจรกับค่าแรงดัน (Voltage inverter)

สำหรับไอซี MAX232 ในเวลาที่นำไปใช้งานเราจะต้องใช้ Capacitor มีค่า 1μF ตามรูปที่ 3.34 เพื่อให้ได้ค่าแรงดันที่เปลี่ยนแปลงในเวลา 8051 ส่งข้อมูลมาที่ PC จาก 5 V เป็น 10V และจาก PC มาที่ไมโครคอนโทรลเลอร์ 8051 ที่มีค่าแรงดัน 10 โวลต์ ให้เป็น 5 โวลต์. ในการนำมาใช้งานในบอร์ดไมโครคอนโทรลเลอร์ ได้แสดงในรูปที่ 3.11



รูปที่ 3.11 แสดงโมดูล Serial Port

3.1.4 โมดูลนับ/จับเวลา (Module Timer/Counter)

8051 มีขาจับ Timer/Counter จากภายนอก 2 ขาคือ T0 และ T1 ขาที่ 14 และ 15 เพื่อให้ 8051 ทำงานเป็นตัวนับสัญญาณที่เข้ามาจากภายนอกนั้น ในที่นี้เราจะใช้ไอซี 555 เพื่อเป็นตัวสร้าง Pulse ออกมา ไอซี 555 เป็น ไอซีตั้งเวลาที่นิยมใช้กันมาก และยังประยุกต์ใช้งานได้หลายแบบ ในการสร้างให้เป็นโมดูล Counter นั้นเราจะต้องจงรทำให้ 555 สร้าง Pulse ออกมาที่ขา 3 แล้วต่อเข้ากับขา T0 ของ 8051 สำหรับ IC 555 เราจะใช้ R และ C เป็นตัวที่ทำให้ 555 สร้างสัญญาณ Pulse ออกมา โดยความถี่ที่ 555 สร้างขึ้นมานี้เราสามารถคำนวณได้ตามสมการดังนี้

$$f = \frac{1.44}{(R_1 + 2R_2) \cdot C_1}$$

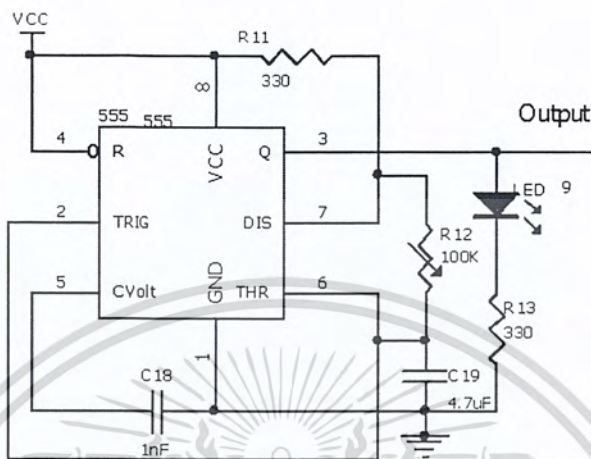
f ความถี่

R_1 เท่ากับ R_{11} ในวงจรที่อยู่ในรูปที่ 3.12

R_2 เท่ากับ R_{12} ในวงจรที่อยู่ในรูปที่ 3.12

C_1 เท่ากับ C_3 ในวงจรที่อยู่ในรูปที่ 3.12

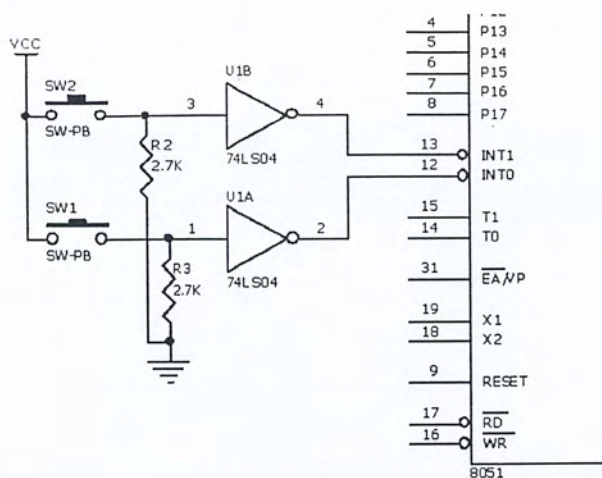
โดยที่ความถี่ที่สร้างขึ้นมานี้จะเปลี่ยนแปลงไปตามค่าความต้านทาน R12 ที่เป็น R ที่สามารถปรับค่าได้, ที่ขาเอาต์พุตของ 555 จะมี LED ต่ออยู่เราสามารถดูที่ LED นี้ได้ว่า 555 สร้าง Pulse ออกมาให้เราหรือไม่ถ้ามี Pulse ออกมา LED จะติดดับตามสัญญาณความถี่ของ Pulse ถ้าไม่มีสัญญาณออกมา LED จะไม่ติด



รูปที่ 3.12 แสดงโมดูลนับ/จับเวลา (Timer/Counter)

3.1.5 โมดูลสวิตช์อินเทอร์รัพท์ (Switch Interrupt Module)

การอินเทอร์รัพท์ (Interrupt) เป็นกระบวนการที่เข้ามาขัดจังหวะการทำงานโดยปกติของไมโครคอนโทรลเลอร์ ในไมโครคอนโทรลเลอร์ MCS-51 สามารถตอบสนองการอินเทอร์รัพท์ที่เกิดขึ้นได้ 5 แหล่งกำเนิดประกอบด้วย การรับสัญญาณอินเทอร์รัพท์จากภายนอกผ่านขา INT0 และ INT1, สัญญาณอินเทอร์รัพท์จากไทมเมอร์/คาน์เตอร์ T0 และ T1 และ สัญญาณอินเทอร์รัพท์จากพอร์ตอนุกรมภายใน ไมโครคอนโทรลเลอร์ ในบอร์ดไมโครคอนโทรลเลอร์ 8051 เราจะใช้สวิตช์ 2 ตัว ดังแสดงในรูปที่ 3.31เป็นตัวกำหนดการอินเทอร์รัพท์ผ่านทางขาอินเทอร์รัพท์ขา INT0 และINT1(ขาที่ 12 และขาที่ 13) ในการทำอินเทอร์รัพท์ “0” และอินเทอร์รัพท์ “1” เพื่อให้เกิดมี Pule ออกมาหนึ่งลูก เมื่อมีการอินเทอร์รัพท์ในไมโครคอนโทรลเลอร์ MCS-51 เกิดขึ้นและมีการอินเเปิดการตอบสนองการอินเทอร์รัพท์ไว้ จากนั้นซีพียูจะทำการกระโดดไปยังแอดเดรสในหน่วยความจำที่กำหนดไว้เรียกตำแหน่งแอดเดรสนี้ว่าแอดเดรสอินเทอร์รัพท์เวกเตอร์ (Interrupt vector address) ดังนั้นจึงมีการเขียนโปรแกรมย่อยการบริการอินเทอร์รัพท์ไว้ที่แอดเดรสนี้

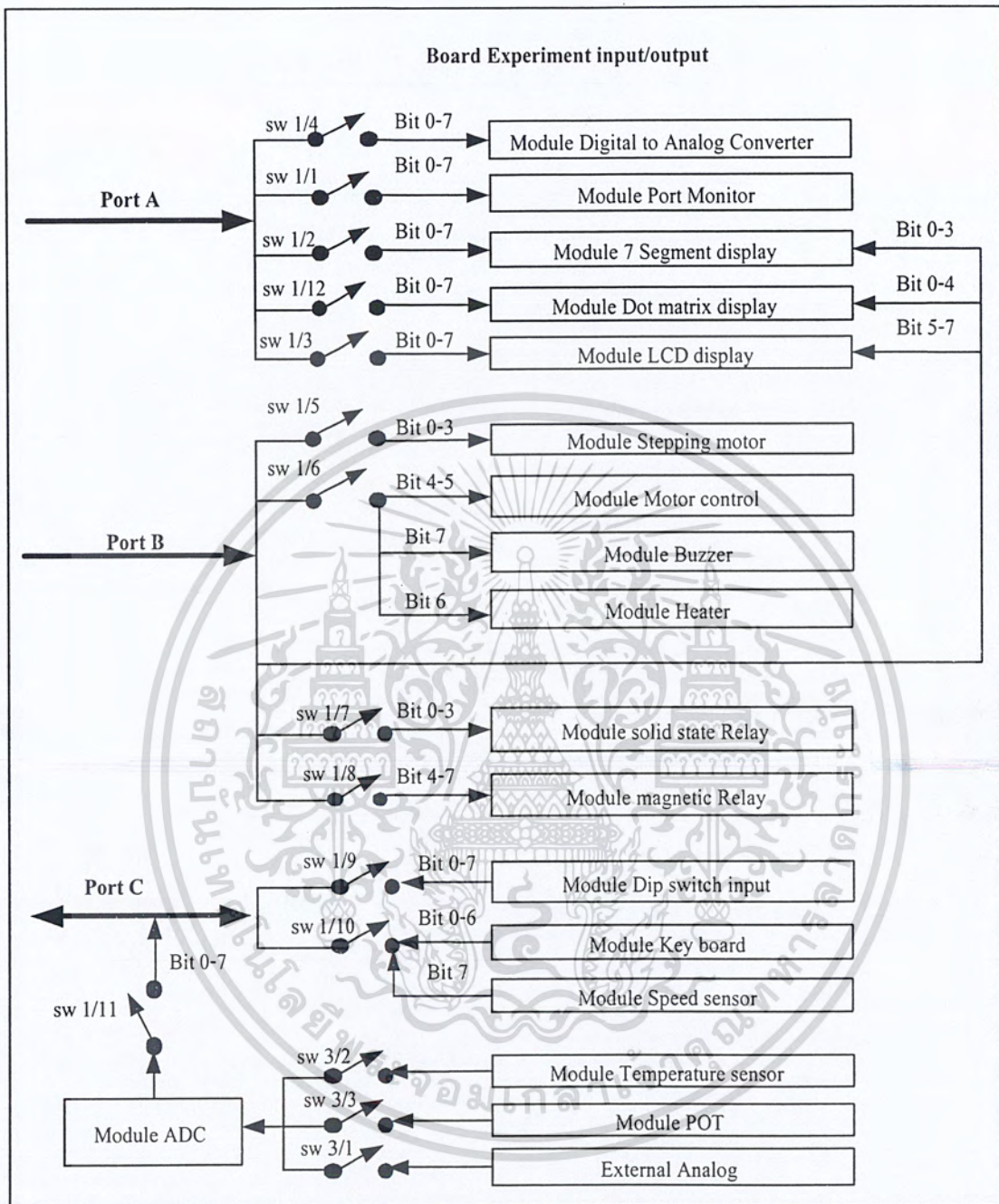


รูปที่ 3.13 แสดงโมดูลอินเทอร์เฟซ

3.2 บอร์ดทดลองอินพุตเอาต์พุต (Experiment input output board)

ส่วนบอร์ดทดลองอินพุตเอาต์พุตนั้นจะประกอบด้วยโมดูลต่างๆที่เป็นอินพุตและเอาต์พุต มีโมดูลหลักๆและ โมดูลย่อยต่างๆที่แบ่งตามลักษณะของการใช้งาน บอร์ดทดลองอินพุตเอาต์พุตนี้ ถูกออกแบบมาให้มีหลายโมดูลเพื่อที่จะเอาไปใช้ได้หลากหลายรูปแบบ เพราะ จุดประสงค์ของการสร้างบอร์ดนี้ขึ้นมา ก็เพื่อที่จะให้เป็นบอร์ดที่สามารถใช้เป็นส่วนหนึ่งของการทำการทดลอง (LAB) ในการศึกษาในระดับ ปวช และปวส หรือ ในระดับปริญญาตรี สำหรับการทำงานของแต่ละ โมดูลที่มีในบอร์ดสามารถทำงานได้อย่างอิสระโดยไม่ขึ้นต่อกันคือ สามารถที่จะ ใช้เป็นโมดูลเดี่ยวๆได้ และสามารถที่จะใช้โมดูลต่างๆร่วมกันได้.

บอร์ดทดลองอินพุตเอาต์พุตจะประกอบด้วยหลายโมดูล ในขณะที่พอร์ตการใช้งานนั้นมีเพียงสามพอร์ตเท่านั้นดังนั้นจึงมีหลายโมดูลที่ต่อพอร์ตใช้งานร่วมกัน ซึ่งมันได้ทำให้โมดูลเหล่านั้น (โมดูลที่ต่อพอร์ตใช้งานร่วมกัน) ไม่สามารถที่จะทำงานพร้อมกันได้เนื่องจากว่าพอร์ตมันจะชนกัน และเพื่อให้แต่ละโมดูลที่ต่อพอร์ตใช้งานร่วมกันทำงานได้อย่างเป็นอิสระโดยไม่ผลกระทบต่โมดูลอื่นๆนั้น จะใช้บัฟเฟอร์เป็นตัวที่แยกแต่ละโมดูลออกจากกัน และจะใช้สวิตช์เป็นตัวเลือกว่าจะให้บัฟเฟอร์ตัวไหน โมดูลไหนทำงาน ด้วยการต่อขาเลือกของบัฟเฟอร์เข้ากับสวิตช์ดังนั้นจึงทำให้แต่ละ โมดูลทำงาน ได้อย่างไม่มีปัญหา ส่วน โมดูลไหนที่ไม่ได้ต่อกับพอร์ตใช้งานพอร์ตเดียวกันนั้นก็สามารที่จะทำงานร่วมกับได้. ส่วนของพอร์ตใช้งานทั้งสามพอร์ตนั้นจะต่อเข้ากับคอนเน็คเตอร์ 40 ขา และมีตำแหน่งของขาตรงกับตำแหน่งของขาคอนเน็คเตอร์ 40 ขา ที่เป็น User port ที่มีในบอร์ดไมครคอนโทรลเลอร์ 8051.



รูปที่ 3.14 แสดงการต่อพอร์ตการใช้งานของแต่ละโมดูล

จากรูป ที่ 3.14 จะเห็นมีหลายโมดูลที่ใช้งานร่วมกันไม่ได้เช่น โมดูล Port monitor และ โมดูล 7 Segment หรือโมดูล 7 Segment และ โมดูล Stepping motor หรือโมดูลอื่นๆ ที่ต่อพอร์ตใช้งานร่วมกัน และก็มีหลายโมดูลที่ต่อใช้งานร่วมกันเช่นโมดูล Temperature sensor กับ โมดูล ADC และ Heater หรือโมดูล Speed sensor กับโมดูล Motor control และยังมีอีกหลายๆ โมดูลที่ใช้งานร่วมกันได้

บอร์ดทดลองอินพุต เอาต์พุต (Experiment Input/Output Board)	
ส่วนที่เป็นเอาต์พุต	ส่วนที่เป็นอินพุต
โมดูล Port monitor	โมดูล Key Board
โมดูล 7- Segment	โมดูล Dip Switch
โมดูล LCD	โมดูล ADC Module
โมดูล Buzzer	โมดูล Temperature
โมดูล Magnetic Relay	โมดูล Speed sensor
โมดูล Solid State Relay	โมดูล Heater
โมดูล DAC	
โมดูล Stepping motor	
โมดูล Motor control	
โมดูล Dot matrix display	

ตารางที่ 3.1 แสดงโมดูลต่างๆที่มีในบอร์ดทดลองอินพุตเอาต์พุต

สำหรับแต่ละโมดูลที่มีในบอร์ดทดลองอินพุตเอาต์พุตจะต่ออยู่กับพอร์ตใช้งานสามพอร์ตคือ พอร์ต PA, PB, PC และแต่ละโมดูลได้แยกจากกันอย่างเป็นอิสระ โดยการใช้บัฟเฟอร์เป็นตัวแยกแต่ละโมดูลออกจากกัน สำหรับการใช้งานแต่ละโมดูลนั้นเราสามารถควบคุมได้โดยการใช้สวิทช์เป็นควบคุมให้แต่ละโมดูลทำงาน ซึ่งการต่อแต่ละโมดูลเข้ากับแต่ละพอร์ตมีดังแสดงในรูปที่ 3.14 ส่วนรายละเอียดของแต่ละโมดูลนี้นั้นมีดังนี้

3.2.1 โมดูลที่เป็นอินพุตประกอบด้วยโมดูลต่างดังนี้

3.2.1.1 โมดูล คีย์บอร์ด (Key board)

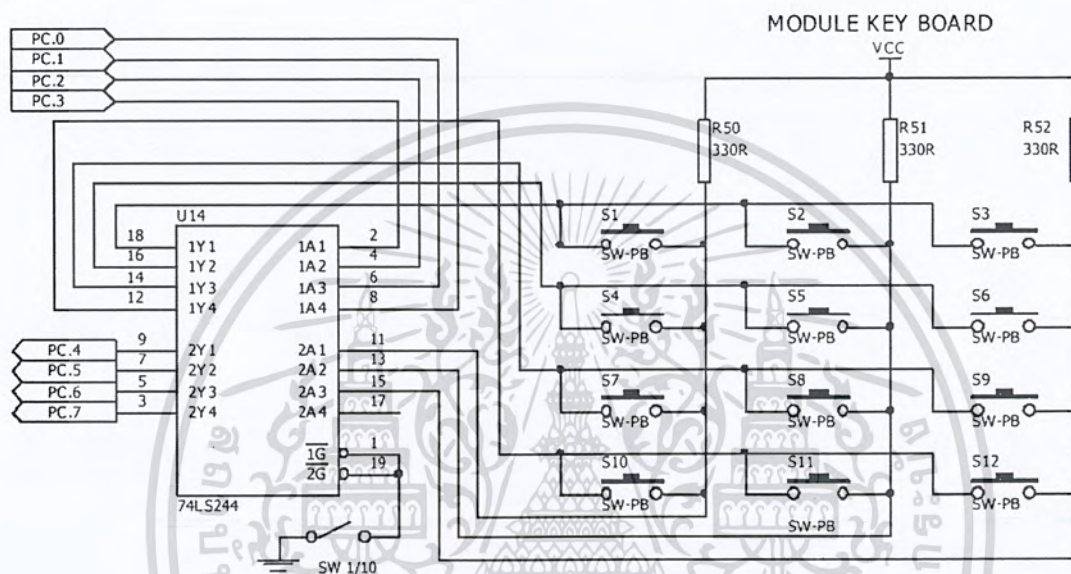
เพื่อเป็นการศึกษาการทำงานของคีย์บอร์ด ให้เราดูและเข้าถึงการทำงานของคีย์บอร์ด ดังนั้นในบอร์ดทดลองอินพุตเอาต์พุตจึงสร้างให้มีโมดูลคีย์บอร์ดโดยที่คีย์บอร์ดที่มีใน บอร์ดทดลองทดลองอินพุตเอาต์พุตเป็นตัวอย่างแสดงถึงหลักการทำงานของคีย์บอร์ดในระบบไมโครคอมพิวเตอร์ซึ่งจะเป็นพื้นฐานในการนำไปประยุกต์ใช้งานอุปกรณ์อื่นๆได้

คีย์บอร์ดที่ใช้จะเป็นคีย์บอร์ดขนาด 3 x 4 โดยจะมี 3 คอลัมน์ (Columns) และจะมี 4 แถว (Rows) จะมีปุ่มกดทั้งหมด 12 ปุ่ม เริ่มจากปุ่มกดที่มีหมายเลข 1 – 9 และอีก 3 ปุ่มคือปุ่มหมายเลข 0, ปุ่ม *, ปุ่ม # ดังแสดงในรูปที่ 3.1

จากรูปที่ 3.1 เราจะเห็นว่าคีย์บอร์ดจะมีตัวด้านทาน 330โอห์ม ต่อเข้ากับคอลัมน์ คือ C1, C2, C3 (C: Column) แล้วต่อเข้ากับ V_{cc} เพื่อทำให้คอลัมน์มีสถานะลอจิกเป็น “1” (High) โดยที่คอลัมน์ C1, C2, C3

จะต่อเข้ากับพอร์ตอินพุตเพื่อตรวจสอบว่าคีย์ใดถูกกด โดยที่จะต่อเข้ากับพอร์ต PC ของ 8255 ตัวที่ 1 บิตที่ PC4, PC5 และ PC6 ส่วนแถวคือ R1, R2, R3 และ R4 (R:Row) จะถูกต่อเข้ากับพอร์ตเอาพุตโดยสัญญาณของเอาพุตแต่ละแถว จะมีลอจิกเป็น “0” (Low) ที่แถว เมื่อแถวใดเป็น “0” แถวที่เหลือก็จะเป็น “1” โดยจะวนรอบอยู่อย่างนี้

โดยที่แต่ละแถวจะต่อเข้ากับพอร์ต PC ของ 8255 บิตที่ PC0, PC1, PC2 และ PC3 และแต่ละ Column ต่อที่ PC4, PC5, PC6 และ PC7 โดยผ่านบัฟเฟอร์(buffer) 74LS244 ดังแสดงในรูปที่ 3.15 การเลือกให้โมดูลคีย์บอร์ดทำงาน ต้องทำการเลือกสวิตช์ 1/10 ไปที่ตำแหน่ง “On”



รูปที่ 3.15 แสดงการนำเอา Key board มาต่อเพื่อใช้งาน

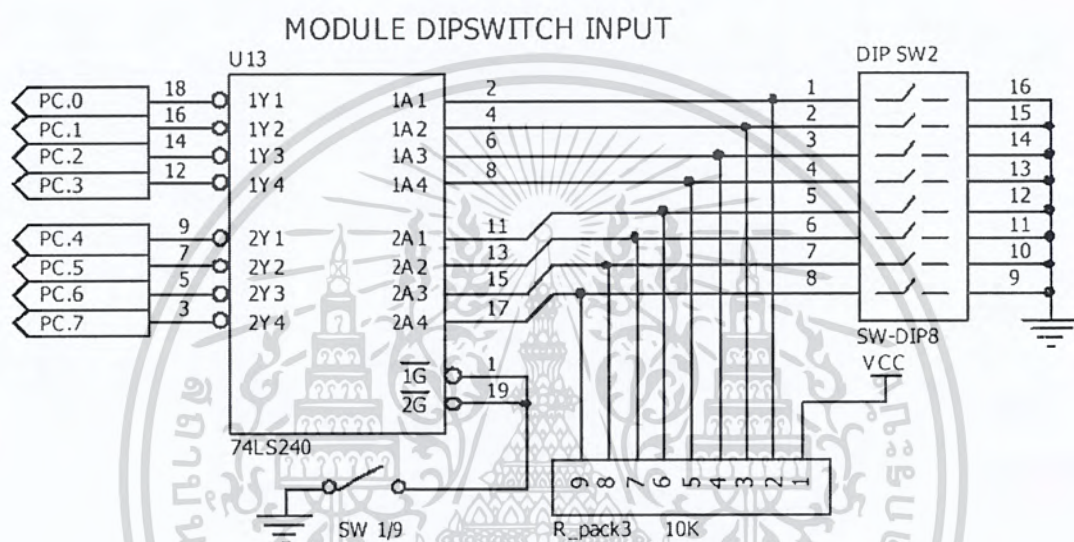
ในการทำงานเมื่อมีการกดคีย์ใดหนึ่งสัญญาณของอินพุตของคอลัมน์ C1, C2, C3 ที่เป็นลอจิก “1” ทั้งหมด 1 1 1 ในขณะที่แถวมีค่าเป็น “0” อยู่แถวหนึ่งแล้วส่วนแถวที่เหลือมีค่าเป็น “1” ทั้งหมดและวนรอบไปเรื่อยๆ นั้น เมื่อมีการกดคีย์ใดคีย์หนึ่งจะทำให้คอลัมน์ที่ถูกกดที่มีค่าเป็น “1” นั้นมีค่าเป็น “0” เช่น มีการกดที่คีย์หมายเลข 4 คอลัมน์ C1, C2, C3 ที่มีค่าเป็น 1 1 1 ทั้งหมด ก็จะกลายเป็น 0 1 1 เพราะที่ C1 จะกลายเป็น “0” ในจังหวะที่แถว R2 มีสถานะเป็น “0” เมื่อนำผลของการอินพุตและเอาต์พุตมารวมด้วยกันก็สามารถบอกตำแหน่งของคีย์บอร์ดที่ถูกกดได้

3.2.1.2 โมดูลสวิตช์อินพุต (คิฟสวิตช์)

โมดูลนี้จะเป็นการจะศึกษาการอ่านข้อมูลขนาด 8 บิต ที่พอร์ต PC ของ 8255 โดยผ่าน บัฟเฟอร์ (buffer) 74LS240 Switch จะเป็นตัวสร้างสถานะลอจิกสูง “HIGH” หรือค่า “LOW” ดังแสดงในรูปที่ 3.2 โมดูลสวิตช์อินพุตจะเป็นสวิตช์จำนวน 8 ตัวคือ SW1-SW8 ซึ่งสวิตช์อินพุตนี้จะใช้คิฟสวิตช์ (คิฟสวิตช์) เป็นตัวที่ทำหน้าที่ในการสวิตช์เพื่อให้มีสัญญาณอินพุตเข้าไป ในที่นี้คิฟสวิตช์ข้างที่ต่อเข้ากับอินพุตของบัฟเฟอร์ 74LS240 จะมี ความต้านทาน 10 กิโลโอห์ม ต่ออยู่แล้วต่อเข้ากับ V_{CC} เรียกว่าการพูลอัพ (Pull

up) และอีกข้างหนึ่งของคิพสวิตช์จะต่อรวมกันแล้วต่อลงกราวด์ การทำงานของ คิพสวิตช์ เมื่อ คิพสวิตช์อยู่ที่ตำแหน่ง "OFF" ในเวลานี้จะทำให้พอร์ต PC ของ 8255 มีสถานะลอจิกเป็น "0" เนื่องจากว่าได้รับแรงไฟจากการ पुल้อบผ่านตัวต้านทานโดยผ่านบัฟเฟอร์ 74LS240 เข้าไปที่พอร์ต PC และในเวลาทีคิพสวิตช์อยู่ที่ตำแหน่ง "ON" ในเวลานี้อินพุตของพอร์ต PC ของ 8255 จะมีสถานะ ลอจิกเป็น "1" เนื่องจากว่า คิพสวิตช์ "ON" ทำให้ตัวต้านทาน ถูกต่อลงกราวด์ทำให้กระแสไหล ลง กราวด์

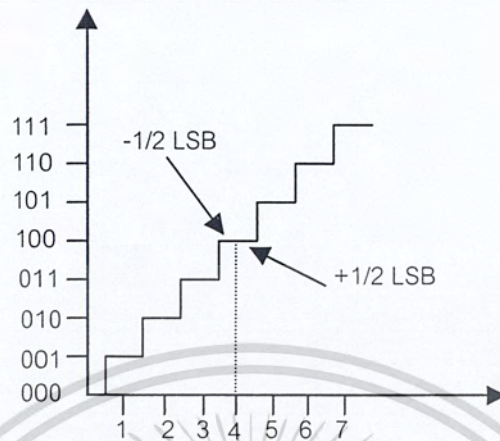
ขณะที่อินพุตของ บัฟเฟอร์ 74LS240 จะมีลอจิกเป็น "0" ดังนั้นที่พอร์ต PC ของ 8255 จึงมีลอจิกเป็น "1" ในการเลือกให้โมดูล คิพสวิตช์ ทำงาน ต้องทำการเลือกสวิตช์ 1/9 ไปที่ตำแหน่ง "On"



รูปที่ 3.16 แสดงการต่อโมดูล คิพสวิตช์

3.2.1.3 โมดูลการแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Module ADC)

เป็นโมดูลที่ทำการแปลงสัญญาณอนาล็อกสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง เช่นระดับแรงดันไฟหรือ กระแส ให้กลายเป็นสัญญาณดิจิตอล ที่อยู่ในรูปฐานสอง "0" กับ "1" ในการแปลงสัญญาณอนาล็อกให้เป็นดิจิตอลนั้นจะต้องคำนึงถึงความละเอียดของ ADC ซึ่งความละเอียดนี้จะขึ้นอยู่กับจำนวนบิตของเอาต์พุต ยิ่งมีจำนวนบิตของเอาต์พุตมากเท่าไร ความละเอียดของ A/D ก็ยิ่งมากขึ้น ทั้งนี้เพราะว่าแรงดันอินพุตที่ป้อนเข้ามาถ้าหากมีการเปลี่ยนแปลงอยู่ในช่วง $+1/2$ LSB ถึง $-1/2$ LSB จะไม่มีผลต่อเอาต์พุตของ A/D เลข (LSB บิตที่มีความสำคัญต่ำสุด Least Significant Bit) (MSB บิตที่มีความสำคัญสูงสุด Most Significant Bit) ตัวอย่างการแปลงสัญญาณไฟฟ้าให้เป็นสัญญาณดิจิตอลขนาด 3 บิตเอาต์พุตจะประกบด้วย D0,D1 และ D2 ในทางอุดมคติของการแปลงสัญญาณอนาล็อกให้เป็นดิจิตอล ขนาด 3 บิตจะให้ 1LSB เทียบเท่าแรงดัน 1 โวลต์ ดังนั้นเมื่อป้อนอินพุตเป็น 0,1,2 เอาต์พุตจะเป็น 000 001 010 แต่ถ้าหากอินพุตที่เข้ามาหากเป็น 1.25 โวลต์ เอาต์พุตที่ได้ก็ยังคงเป็น 001 เพราะแรงดันเอาต์พุตที่ป้อนเข้ามามีค่าอยู่ในช่วง $+1/2$ LSB ถึง $-1/2$ LSB จะไม่มีผลต่อเอาต์พุต

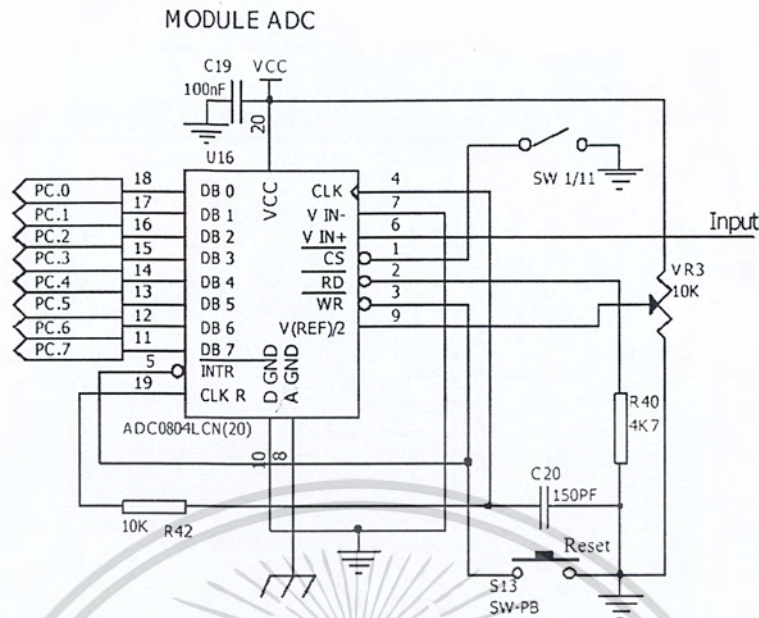


รูปที่ 3.17 แสดงความไม่ละเอียดของการแปลงสัญญาณของ A/D

เนื่องจาก A/D ไม่สามารถแยกความแตกต่างได้ถ้าแรงดันอินพุตมีค่าความแตกต่างของแรงดันน้อยกว่า 1 LSB แต่เพื่อที่จะทำให้ A/D แยกความแตกต่างของแรงดันอินพุตได้ดีขึ้นจะต้องทำให้ 1 LSB มีค่าน้อยที่สุด โดยการเพิ่มจำนวนบิตเอาต์พุตของ A/D ให้มากขึ้น เราสามารถคำนวณขนาดของ LSB ได้จากสมการ

$$1 \text{ LSB} = \frac{\text{แรงดันสูงสุดของอินพุต}}{(2^{\text{จำนวนบิตเอาต์พุต}}) - 1}$$

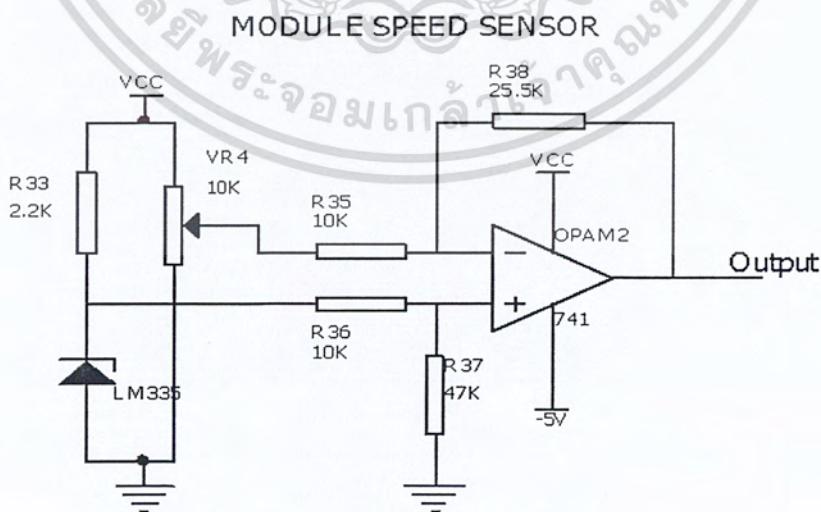
ในบอร์ดนี้จะใช้ A/D เบอร์ 0804 ซึ่งเป็น A/D ขนาดเอาต์พุต 8 บิต แล้วหลังจากนั้นก็ทำการต่อสายสัญญาณ AD0-AD7 เข้ากับ A/D ส่วนสัญญาณอินพุตที่จะต่อเข้ามานั้นสามารถที่จะต่อมาจากภายนอก และในเวลาที่กำลังทำงานอยู่แล้วเกิดขัดข้องขึ้นหรือทำงานไม่เป็นปกติ ก็สามารถทำการ Reset ได้ด้วยการกด SW Reset ดังแสดงในรูปที่ 3.18 เมื่อต้องการให้ไมโครนี้ทำงาน ต้องเลือกสวิตช์ที่ 1/11 ไปที่ตำแหน่ง “On”



รูปที่ 3.18 แสดงโมดูล ADC

3.2.1.4 ชุดตรวจวัดอุณหภูมิ (TEMPERATURE SENSOR)

เป็น โมดูลที่ศึกษาการรับข้อมูลทางอุณหภูมิในบอร์ดทดลองอินพุตเอาต์พุต ตัวตรวจวัดอุณหภูมิที่ใช้คือ LM335 ซึ่งมีการทำงานคล้าย กับซีเนอร์ไดโอด (Zener Diode) จะมีจุด Break Down Voltage ที่เป็นสัดส่วนกับค่าสัมบูรณ์ของอุณหภูมิตัว LM335 จะมีอัตราการเปลี่ยนแปลงของแรงดันเมื่ออุณหภูมิเปลี่ยนไปเท่ากับ 10 มิลลิโวลต์ต่อองศาเคลวิน ชุดตรวจวัดอุณหภูมิแสดงในรูปที่ 3.5เมื่อจะทำการใช้งานได้ต้องทำการเลือกสวิตช์ที่ 1/11 และสวิตช์ 3/2 ไปที่ตำแหน่ง “On”



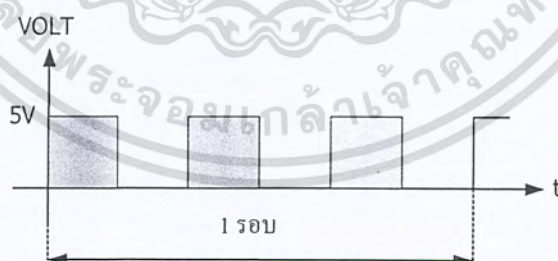
รูปที่ 3.19 แสดงวงจรชุดตรวจวัดอุณหภูมิ

ในบอร์ดทดลองอินพุตเอาต์พุต ชุดตรวจวัดอุณหภูมิต่อเข้ากับ PC0-PC7 โดยผ่านตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC0804LCN) LM 335 จะถูกวางไว้ใกล้ๆ กับตัวทำความร้อน (Heater) เพื่อให้สามารถทำการทดลองแบบ Close Loop ได้ เมื่ออุณหภูมิ มีการเปลี่ยนแปลงในช่วง 0-100 องศาเซลเซียส เมื่ออุณหภูมิเพิ่มขึ้นแรงดันก็จะเพิ่มตามด้วยอัตราส่วนที่คงที่ แรงดันที่ได้จะส่งเป็นอินพุตของออปแอมป์ 741 ซึ่งต่อเป็นวงจรดิฟเฟอเรนเชียล (Differential Amplifier) ทำให้ได้เอาต์พุต V_0 เป็นแรงดันที่เปลี่ยนอยู่ในช่วง 0-2.55 โวลต์ซึ่งแรงดัน V_0 นี้จะต่อเข้ากับวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล เพื่อเปลี่ยนแรงดัน 0-2.55 โวลต์ ให้เป็นสัญญาณดิจิตอลขนาด 8 บิต LM335 จะติดอยู่กับตัวฮีตเตอร์ เมื่ออุณหภูมิของฮีตเตอร์เพิ่มขึ้น แรงดันตกคร่อม LM 335 ก็จะมากขึ้น ทำให้ผลต่างแรงดันที่ขา 2 และ ขา 3 ของ 741 มีมากขึ้น แรงดันเอาต์พุต ของวงจรก็จะเพิ่มขึ้นจนถึง 2.55v เมื่ออุณหภูมิที่ตัวฮีตเตอร์ขึ้นถึง 100 เซลเซียส

3.2.1.5 โมดูลวงจรตรวจจับความเร็ว (SPEED SENSOR CIRCUIT MODULE)

เป็นส่วนที่ทำการศึกษารองรับความเร็วรอบของ มอเตอร์ โดยจะอาศัยใบพัดของมอเตอร์ที่หมุนตัดผ่านแสงอินฟราเรด (Infrared) ที่ส่องมายังตัวโฟโตทรานซิสเตอร์ (Photo Transistor) เมื่อลำแสงอินฟราเรดถูกบังโดยใบพัดของมอเตอร์ วงจรตรวจจับความเร็วก็จะส่ง เอาต์พุตออกมาเป็น “1” เนื่องจากว่าโฟโตทรานซิสเตอร์ไม่ได้รับแสงอินฟราเรดก็จะไม่นำกระแสแสดงกราวด์ เมื่อใบพัดมอเตอร์เลื่อนผ่านไป โฟโตทรานซิสเตอร์ก็สามารถรับแสงจากหลอดอินฟราเรดได้โฟโตทรานซิสเตอร์ก็จะนำกระแสแสดงกราวด์วงจรก็จะให้เอาต์พุตออกมาเป็น “0”

ใบพัดของมอเตอร์เป็นแบบ 3 ใบพัด ดังนั้นเมื่อมอเตอร์หมุนไป 1 รอบ ใบพัดก็จะไปตัดแสงอินฟราเรด เอาต์พุตของวงจรตรวจจับความเร็วจะมีสภาวะเป็น “1” และ “0” อย่างละ 3 ครั้งสลับกันจะเร็วหรือช้า ขึ้นอยู่กับความเร็วของมอเตอร์



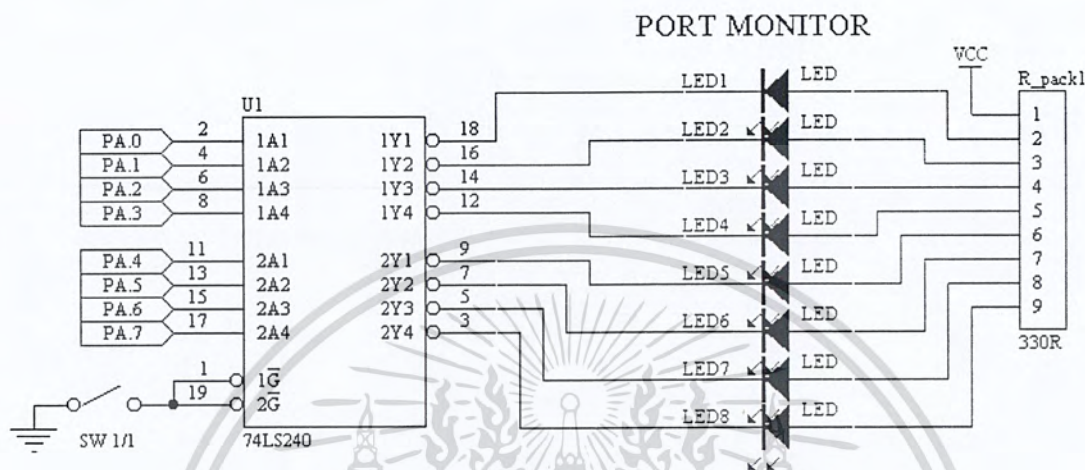
รูปที่ 3.20 แสดงเอาต์พุตของวงจรตรวจจับความเร็วเมื่อมอเตอร์หมุน 1 รอบ

หลอดอินฟราเรดจะส่งแสงอินฟราเรดออกมาอยู่ตลอดเวลา โดยมีตัวต้านทาน 220โอห์ม เป็นตัวจำกัดกระแส ให้หลอดอินฟราเรด ส่วนโฟโตทรานซิสเตอร์จะต่อเข้ากับความต้าน 470 โอห์ม ที่ขั้วเอาต์พุตของวงจรจะมีแรงดันประมาณ 5 โวลต์ ในบอร์ดทดลองอินพุตเอาต์พุตนี้ วงจรตรวจจับความเร็วจะต่อเข้ากับ

3.2.2 โมดูลที่เป็นเอาต์พุต

3.2.2.1 โมดูล Port monitor

เป็นโมดูลที่แสดงผลถึงระดับบิตโดยใช้ LED (Light Emitting Diode) เป็นตัวแสดงผล Light Emitting Diode จะทำหน้าที่เป็นตัวแสดงผลสถานะลอจิกในแต่ละบิต(Bit) ของเอาต์พุต ซึ่งจะประกอบด้วยหลอด LED จำนวน 8 หลอดและจะต่อเข้ากับพอร์ต PA ของ 8255 ดังแสดงในรูปที่ 3.9



รูปที่ 3.23 แสดงการต่อโมดูล LED

โดยที่ LED ทั้ง 8 หลอดจะต่อเข้ากับพอร์ต PA ของ 8255 เริ่มจากบิต PA1.0-PA1.7 ของ 8255 โดยใช้ บัฟเฟอร์ (buffer) 74LS240 เป็นตัวกั้นระหว่าง IC8255 และ LED ในโมดูลนี้เราจะให้ Vcc เป็นตัวที่จ่ายกระแสให้กับ LED สายเหตุที่มีการต่อวงจรแบบนี้ก็เพราะว่า LED จะได้รับกระแสเต็มที่และจะทำให้ LED สว่างมากกว่าการที่จะให้ 8051 เป็นตัวที่จ่ายกระแสให้ LED และ LED ก็จะไม่ดึงกระแสจาก 8051 ซึ่งเป็นผลดีและไม่มีความเสี่ยงต่อ 8051

โมดูลนี้จะทำงานเมื่อสวิตช์ 1/1 อยู่ที่ตำแหน่ง “On” และมีเอาต์พุตลอจิกเป็น “1” จากพอร์ต PA ของ 8255 มาที่บัฟเฟอร์ (buffer) 74LS240 ทำงานให้ข้อมูลผ่านเมื่อเกิดความต่างศักย์ของแรงดันระหว่างแหล่งจ่าย +5 โวลต์กับเอาต์พุตของพอร์ต PA ของ 8255 เป็นผลให้มีกระแสจำนวนหนึ่งไหลผ่าน LED และตัวต้านทาน 330 โอห์ม ทำให้ LED ติดสว่างในที่นี้ ถ้าเอาต์พุตพอร์ต PA ของ 8255 มีลอจิกเป็น “0” ทำให้ไม่เกิดความต่างศักย์ขึ้นระหว่างแหล่งจ่าย +5 โวลต์ กับเอาต์พุตทำให้ไม่มีกระแสไหลผ่าน LED และตัวต้านทานทำให้ หลอด LED ไม่ติดสว่าง

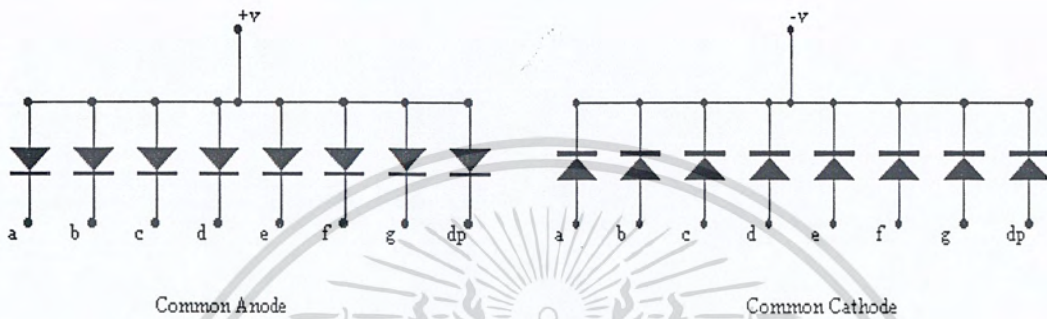
3.2.2.2 โมดูล 7 Segment

ชุดแสดงผลแบบ LED 7 ส่วน (7-SEGMENT LED DISPLAY) เป็นโมดูลที่แสดงผลเป็นตัวเลขและตัวอักษรบางตัว 7-Segment display เป็นตัวแสดงผลที่นำหลอด LED มาต่อรวมกันเป็นชุดอยู่ในตัวถังเดียวกันภายในประกอบด้วยหลอด LED ต่อเรียงกันอยู่ดังในรูปที่ 3.24 และรูปที่ 3.25 เพื่อให้สามารถ

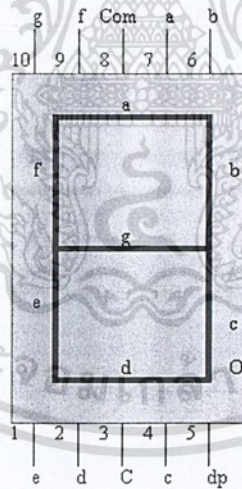
แสดงผลออกมาเป็นตัวเลข 0-9 หรือเป็นตัวอักษร A,B,C,D,E,F,H, I,J,L,O,P,S,U และเครื่องหมาย - หรือ จุดทศนิยม(.)

LEDแต่ละดวงจะต่อรวมกันโดยจะมีอยู่หาหนึ่งที่ต่อรวมกัน อาจจะเป็นขาคาโทด(Cathode) หรือ อาโนด (Anode) ดังนั้นเราจึงแบ่ง 7-Segment display ออกเป็น 2 แบบคือ

1. แบบอาโนดร่วม(Common Anode)
2. แบบคาโทดร่วม(Common Cathode)



รูปที่ 3.24 แสดงโครงสร้างภายในของ 7-Segment ทั้ง 2 แบบ



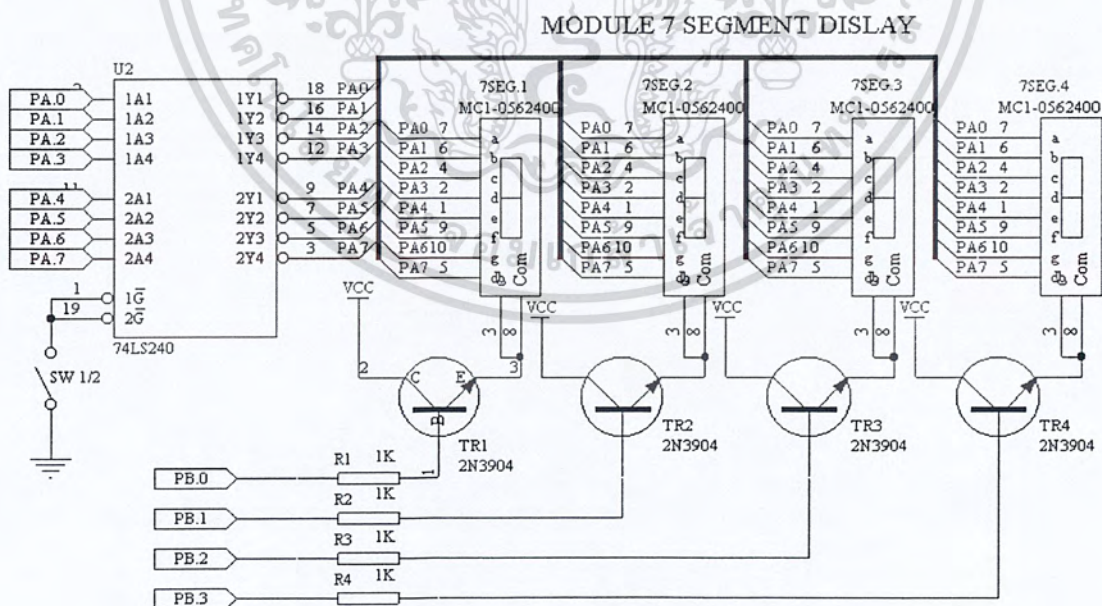
รูปที่ 3.25 แสดงตำแหน่งขาของ 7-Segment Display

การแสดงผลแบบ Multiplex โดยใช้ 7-Segment แบบ Common Anode การแสดงผลแบบนี้อาศัยวิธีการใช้ LED 7-Segment ติดทีละหลักโดยอาศัยช่วงเวลาต่างกัน 1/24 วินาที ทำให้ตาเราจับไม่ทัน จากรูปที่ 3.26 แสดงถึงการต่อ LED 7-Segment จำนวน 4 หลัก โดยเชื่อมโยงกับพอร์ต 8255 โดยผ่าน buffer เป็นตัวกันดังนี้

- พอร์ต PA ถูกต่อกับ 7-Segment ตั้งแต่ Segment a จนถึง Segment g และ dot รวมเป็น 8 Segment โดย

PA0 ต่อกับ	Segment a
PA1 ต่อกับ	Segment b
PA2 ต่อกับ	Segment c
PA3 ต่อกับ	Segment d
PA4 ต่อกับ	Segment e
PA5 ต่อกับ	Segment f
PA6 ต่อกับ	Segment g
PA7 ต่อกับ	Segment dot

- พอร์ต PB0-PB3 ต่อกับขา Common Anode ของแต่ละหลักโดยผ่านทรานซิสเตอร์ Vcc จะจ่ายไฟให้แก่ขาร่วมของ LED ตัวเลข 7 ส่วนที่ละหลักไล่ไปตามลำดับด้วยความเร็วสูงโดยผ่านทรานซิสเตอร์ ทั้งนี้การต่อทรานซิสเตอร์เพื่อขับขาคอมมอนนั้นยังช่วยให้ภาระในการจ่ายกระแสของไมโครคอนโทรลเลอร์ลดลงด้วย ในบอร์ดที่สร้างขึ้นนี้คือโมดูล 7 Segment ส่วนขาของแต่ละเซกเมนต์จะถูกต่อเข้ากับพอร์ต PA0-PA7 โมดูลนี้จะทำงานเมื่อสวิตช์ 1/2 อยู่ที่ตำแหน่ง “ON” กระบวนการเริ่มต้นโดยไมโครคอนโทรลเลอร์ MSC-51 ส่งข้อมูลออกไปยัง 8255 ต่ออยู่กับเซกเมนต์ a-g และ dp ดังแสดงในรูปที่ 3.26 สำหรับความต้าน 1 กิโลโอห์มเป็นตัวป้องกันไม่ให้ทรานซิสเตอร์ได้รับความเสียหาย



รูปที่ 3.26 แสดงการต่อโมดูล 7 Segment

3.2.2.3 โมดูล LCD

การใช้งานอุปกรณ์แสดงผลแบบผลึกเหลว LCD (Liquid Crystal Display) เป็นส่วนที่ศึกษาส่วนที่แสดงผลเป็นแบบตัวอักษรหรือตัวเลข ตัวแสดงผล LCD เป็นอุปกรณ์แสดงผลแบบหนึ่งที่มีนิยมใช้มาก และพบเห็นกันบ่อยๆ LCD สามารถแบ่งประเภทตามลักษณะของการแสดงผลได้ 3 แบบคือ LCD แบบอักขระ (Character LCD Module), แบบกราฟิก (Graphic LCD Module) และLCD แบบเซกเมนต์ (Segment LCD Module)

LCDแบบอักขระเป็นโมดูลLCD ที่สามารถแสดงตัวอักษร,ตัวเลข,และเครื่องหมายต่างๆได้ โดยสร้างจากจุดเล็กๆหรือเรียกว่าคอตเมตริกซ์ซึ่งก็จะมีขนาดความกว้างและสูงของอักขระแต่ละตัว โดยทั่วไปมี 2 ขนาดคือ 5x7 จุด 5x10 จุด นอกจากนั้น LCD แบบนี้สามารถแสดงข้อความได้ “1” บรรทัด หรือมากกว่าก็ได้ ขึ้นอยู่กับรุ่นของ LCD นั้นๆ ในบอร์ดทดลองอินพุตเอาต์พุต ใช้ LCD แบบ Dot Matrix ขนาด 16 ตัวอักษร “2” บรรทัด โดยต่อเข้ากับพอร์ต PA และ PB ของ 8255 จากบอร์ดไมโครคอนโทรลเลอร์ พอร์ต PA0-PA7 ทำหน้าที่เป็นบัสข้อมูลขนาด 8 บิต พอร์ต PB5-PB7 ทำหน้าที่เป็นตัวควบคุม, LCD จะมีสัญญาณที่ทำหน้าที่ต่างกัน ดังแสดงในตารางข้างล่างนี้

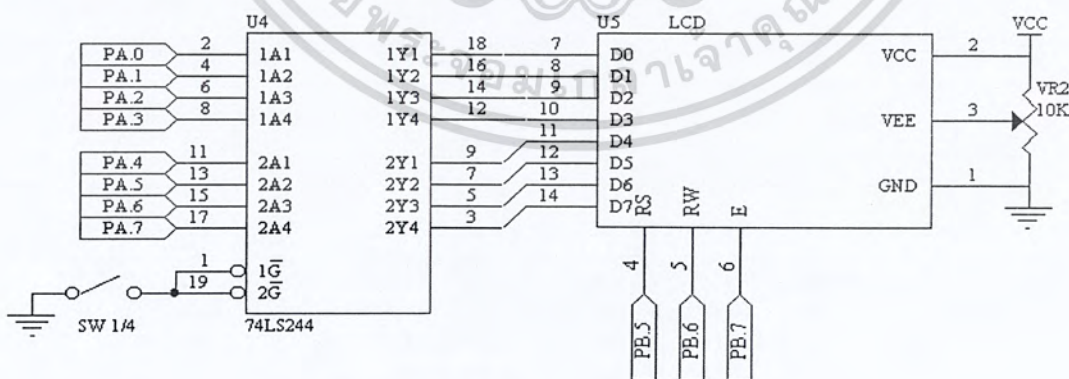
	สัญญาณ	หน้าที่
1	VSS	Ground (0 v)
2	VCC	+5V Power Supply
3	VEE	LCD drive Voltage input(+V)
4	RS	“1”:Data “0”:Instruction
5	R/W	“1”:Read “0”:Write
6	E	Enable Signal
7	DB0	Data bus bit 0
8	DB1	Data bus bit 1
9	DB2	Data bus bit 2
10	DB3	Data bus bit 3
11	DB4	Data bus bit 4
12	DB5	Data bus bit 5
13	DB6	Data bus bit 6
14	DB7	Data bus bit 7

ตารางที่ 3.2 ตารางแสดงหน้าที่แต่ละขาของ LCD

- ขา 1,2 ต่อเข้ากับแหล่งจ่ายไฟ +5v
- ขา 3 สำหรับปรับความเข้มของ LCD

- ขา 4(RS) เป็นขาสัญญาณสำหรับกำหนดว่าข้อมูลที่รับเข้ามาจาก DB0-DB7 เป็นอะไร
- ถ้าค่า RS = “1” แสดงว่าเป็นข้อมูลแสดงผล
- ถ้าค่า RS= “0” แสดงว่าเป็นชุดคำสั่งสำหรับควบคุมการทำงานของ LCD
- ขา 5 (R/W) เป็นขาสัญญาณสำหรับกำหนดว่าจะอ่านหรือเขียนข้อมูลกับตัว LCD
- ถ้าค่า R/W = “1” แสดงว่าตัวควบคุมจากภายนอก (8255 หรือ CPU) ต้องการข้อมูลที่แสดงสถานะของตัว LCD จากขาสัญญาณ DB0-DB7 (ตัว LCD สามารถที่จะแสดงสถานะของตัวเองให้ตัวควบคุมภายนอกได้)
- ถ้าค่า R/W= “0” แสดงว่าตัวควบคุมจากภายนอกต้องการส่งข้อมูลให้กับ LCD ผ่านทางขาสัญญาณ DB0-DB7 ซึ่งข้อมูลที่ส่งให้ LCD ก็จะขึ้นอยู่กับขา RS ด้วยว่าจะป็นชุดคำสั่งหรือเป็นข้อมูลสำหรับการแสดงผล
- ขา 6(E) เป็นขาสัญญาณที่กำหนดให้ LCD ตอบสนองกับตัวควบคุมจากภายนอกไม่ว่าจะมีการเขียนหรืออ่านข้อมูลกับ LCD ขา E จะต้องมีการเปลี่ยนแปลงของสัญญาณจาก “0” ไปเป็น “1” ทุกครั้ง (ทำงานที่ขอบสัญญาณขาขึ้น) ดังนั้นสัญญาณขา E จะต้องมีการเปลี่ยนแปลงทุกครั้งที่จะทำการติดต่อกับ LCD
- ขา DB0-DB7 เป็นขาสัญญาณสำหรับรับหรือส่งข้อมูลระหว่าง LCD กับตัวควบคุมภายนอก

ดังนั้นจะเห็นว่าการจะใช้งาน LCD จำเป็นจะต้องทราบถึงหน้าที่การทำงานของแต่ละขาสัญญาณ แล้วกำหนดการทำงานของตัวควบคุมจากภายนอกให้ถูกต้อง การใช้งาน LCD ก็จะไม่มีความยุ่งยากและซับซ้อนแต่ประการใด วงจรการทำงานของ LCD บนบอร์ดทดลองอินพุตเอาต์พุต มีการต่อสัญญาณข้อมูลผ่าน PA0-PA7 โดยผ่าน 74LS244 ซึ่งเป็นบัฟเฟอร์ ขา 1 และ ขา 19 ต่ออยู่กับสวิตช์ที่ 1/3 เมื่อสวิตช์ 1/3 อยู่ตำแหน่ง “ON” โมดูลนี้ก็จะทำงาน ที่พอร์ต PA ก็จะสามารถทำการติดต่อกับ LCD ได้ ส่วนขาควบคุม RS ,R/W และ E ต่ออยู่กับพอร์ต PB5-PB7 ตามลำดับ VR1 เป็นตัวปรับความเข้มของการแสดงผล ดังแสดงในรูปที่ 3.27



รูปที่ 3.27 แสดงวงจรควบคุม LCD

การเขียนคำสั่งควบคุม LCD

การเขียนหรืออ่านข้อมูลกับ LCD ก็คือการกำหนด LCD ทำตามเงื่อนไขต่างๆ ซึ่ง LCD จะมีชุดคำสั่งเฉพาะอยู่ หรือการสั่งให้ LCD แสดงข้อความหรือรูปภาพตามที่กำหนด ในการที่จะสั่งให้ LCD ทำงานตามที่เรานั้นอันดับแรกเราต้องส่งคำสั่ง Function set ออกไปก่อนและจะต้องหน่วงเวลาไว้ในช่วงระยะหนึ่งอย่างน้อยต้องมากกว่า 40 ไมโครเซกกัน (uS), หลังจากนั้นจึงส่งคำสั่ง Display ON/OFF ออกไป แล้วหน่วงเวลาไว้ในช่วงระยะหนึ่งมากกว่า 40 ไมโครเซกกัน (uS), แล้วส่งคำสั่ง Clear display ออกไป แล้วหน่วงเวลาไ้นานมากกว่า 1640 ไมโครเซกกัน (uS), แล้วจึงส่งคำสั่ง Cursor preset ออกไปหน่วงเวลาไ้นานมากกว่า 40 ไมโครเซกกัน (uS), หลังจากนั้นก็ส่งคำสั่ง Set ddram Addr ออกไป แล้วนงเวลาไว้ในช่วงระยะหนึ่งมากกว่า 40 ไมโครเซกกัน (uS), คำสั่งเหล่านี้เป็นคำสั่งที่ใช้ในการ Reset ตัว LCD เอง ก่อนการใช้งาน เมื่อทำการ Reset เสร็จแล้วก็สามารถที่จะส่งข้อมูลออกไปแสดงบน LCD ได้ ส่วนรายละเอียดของคำสั่งต่างนั้นได้แสดงข้างล่างนี้

Instruction		R/ W	E	DATA BIT								EXE TIM E(uS)
				7	6	5	4	3	2	1	0	
				PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	
CLEAR DISPLAY	0	0		0	0	0	0	0	0	0	1	1640
RETURN HOME	0	0		0	0	0	0	0	0	1	X	1640
40CURSOR PRESET	0	0		0	0	0	0	0	1	I/D	S	40
DISPLAY ON/OFF	0	0		0	0	0	0	1	D	C	B	40
DISPLAY SHIFT	0	0		0	0	0	1	S/C	R/L	X	X	40
FUNCTION SET	0	0		0	0	1	DL	N	F	X	X	40
SET CGRAM ADD	0	0		0	1	CGRAM ADDRESS						40
SET DDRAM ADD	0	0		1	DDRAM ADDRESS							40
BUSY,ADD,READ	0	0		BF	ADDRESS							0
CGRAM,DDRAM WR	0	0		WRITE DATA								40
READ DATA	0	0										40

ตารางที่ 3.3 แสดงชุดคำสั่งของ LCD

รายละเอียดของชุดคำสั่ง

1. CLEAR DISPLAY

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

เป็นคำสั่งลบข้อมูลต่างๆ ที่อยู่ใน DDRAM ให้เป็นข้อความที่ว่างเปล่า ทำให้ข้อความที่อยู่บนจอ LCD ถูกลบหายไป และส่งให้ CURSOR ไปอยู่ที่ตำแหน่งแรกสุดของจอ LCD (ตำแหน่งบนซ้ายสุด)

2. RETURN HOME

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	X

เป็นคำสั่งกำหนดให้ DDRAM ADDRESS เป็น 0 คือไปอยู่ที่ตำแหน่งแรกสุด และส่งให้ เคอร์เซอร์ไปอยู่ที่ตำแหน่งแรกสุดของจอ LCD

3. CURSOR PRESET

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

เป็นคำสั่งกำหนดตำแหน่งของ CURSOR ในขณะที่ LED แสดงข้อความการกำหนด I/D และ S ให้ทำก่อนที่จะมีการเขียนข้อมูลใน DDRAM และจะต้องไม่ใช่คำสั่ง CLEAR DISPLAY อีก.

I/D = “0” ทิศทางของ CURSOR และตำแหน่งของ DDRAM จะเป็นแบบเพิ่มขึ้น

I/D = “1” ทิศทางของ CURSOR และตำแหน่งของ DDRAM จะเป็นแบบลดลง

S = “0” CURSOR จะเลื่อนไปตามทิศทางของการกำหนดค่า I/D

S = “1” CURSOR จะอยู่กับที่ แต่ตัวอักษรจะเลื่อนไปตามทิศทางของการกำหนดค่า I/D

4. DISPLAY ON/OFF

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	D	C	B

D = “0” DISPLAY OFF

D = “1” DISPLAY ON

C = “0” CURSOR OFF

C = “1” CURSOR ON มีขีดใต้ตัวอักษร

B = “0” CUSOR ไม่มีการกระพริบ

B = “1” CUSOR มีการกระพริบเป็นรูปสี่เหลี่ยม

5. DISPLAY SHIF

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

- S/C = “0” เลื่อน CURSOR ไปตามทิศทางที่กำหนดด้วย R/L ไป 1 ตำแหน่ง
- S/C = “1” เลื่อนข้อความไปตามทิศทางที่กำหนดด้วย R/L ไป 1 CORUML ทุกบรรทัด
- R/L = “0” กำหนดทิศทางไปทางซ้าย
- R/L = “1” กำหนดทิศทางไปทางขวา

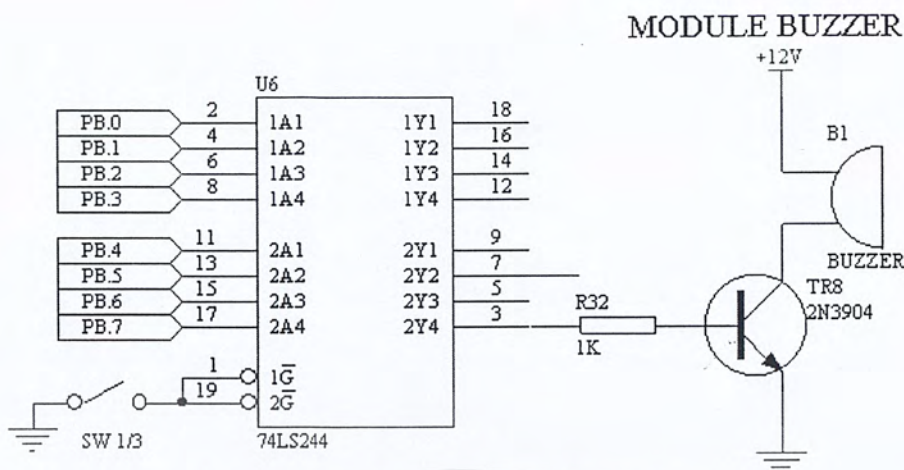
6. FUNCTION SET

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

- DL = “0” LCM ติดต่อกับตัวควบคุมภายนอกแบบ 4 bit
- DL = “1” LCM ติดต่อกับตัวควบคุมภายนอกแบบ 8 bit
- N = “0” กำหนดจำนวนบรรทัดแบบ 1/8 และ 1/11duty
- N = “1” กำหนดจำนวนบรรทัดแบบ 1/16 duty
- F = “0” แสดงอักษรแบบ 5 x 7 Dots
- F = “1” แสดงอักษรแบบ 5 x 7 Dots (ถ้า LCD) เป็นแบบ 5 x 7 Dots จะไม่มีผลต่อคำสั่งนี้

3.2.2.4 โมดูลบัชเซอร์ (Buzzer)

เพื่อเป็นการศึกษาการสร้างสัญญาณเสียงในโมดูลนี้เราจะใช้ บัชเซอร์ (Buzzer) เป็นตัวแสดงผลในบอร์ดทดลองอินพุตเอาต์พุต บัชเซอร์จะต่อเข้ากับ PB บิตที่ 7 ของ 8255 โดยผ่านบัฟเฟอร์ (buffer) 74LS244 ในโมดูลนี้เราจะใช้ Vcc เป็นตัวที่จ่ายกระแสให้กับบัชเซอร์ดูรูปที่ 3.28 จากรูปจะเห็นว่าจากพอร์ต PB บิตที่ 7 ของ 8255 จะมีความต้าน 1 กิโลโอห์ม ต่ออยู่ และต่อเข้ากับขา Base ของทรานซิสเตอร์ เพื่อป้องกันไม่ให้ทรานซิสเตอร์ 2N3904 ได้รับกระแสมากเกินไปเพราะถ้าได้รับกระแสมากเกินไปอาจเกิดความเสียหายได้และทรานซิสเตอร์จะทำหน้าที่เป็นเหมือน Switch on/off เพื่อให้ตัวบัชเซอร์ทำงานในการที่จะทำการทดลองต้องปรับสวิตช์ 1/6 อยู่ที่ตำแหน่ง “On”



รูปที่ 3.28 แสดงการต่อ Buzzer กับพอร์ต PB ของ 8255

ในการทำงานเมื่อมีสัญญาณลอจิก “1” มาจากพอร์ต PB บิตที่ 7 ของ 8255 ก็จะทำให้มีกระแสไหลเข้าที่ Base ของทรานซิสเตอร์ทำให้ทรานซิสเตอร์เกิดการนำกระแส และทำให้มีกระแสที่ไหลจาก Vcc ผ่าน บัซเซอร์ (Buzzer) ทำให้บัซเซอร์ส่งเสียงดังขึ้น และเมื่อพอร์ต PB บิตที่ 7 ของ 8255 มีลอจิกเป็น “0” ก็ไม่มีกระแสไหลเข้าไปที่ขา Base ของทรานซิสเตอร์ทำให้ไม่มีการนำกระแสเกิดขึ้นและบัซเซอร์ก็ไม่ดัง.

บัซเซอร์เป็นอุปกรณ์ที่สามารถสร้างสัญญาณเสียงออกมาได้ บัซเซอร์ที่ใช้ทั่วไปแบ่งออกเป็น 2 ชนิด คือ

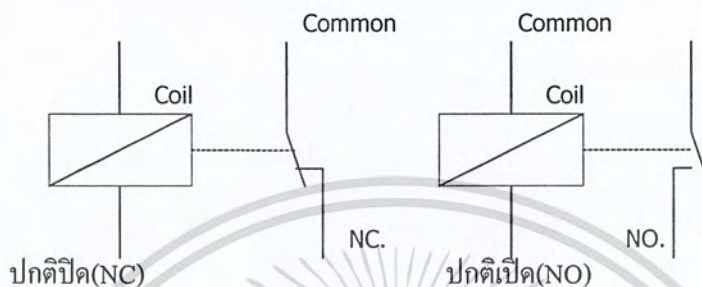
1. Type A เป็นแบบทำงานอิสระเมื่อป้อนแรงดันไฟฟ้าให้กับตัวมันก็จะสามารถสร้างสัญญาณเสียงออกมาได้สัญญาณเสียงที่ออกมาไม่สามารถเปลี่ยนแปลงได้
2. Type B เป็นแบบที่จะต้องสร้างสัญญาณพัลส์ (Pulse) ที่มีความถี่แตกต่างกัน ป้อนให้กับตัวบัซเซอร์จึงจะสามารถสร้างสัญญาณเสียงออกมาได้ตามความถี่ที่ป้อนให้ บัซเซอร์ชนิดนี้โดยทั่วไปจะเรียกว่า “เพียโซ” (Piezo) ในบอร์ดของเราใช้บัซเซอร์แบบ Type B เพราะฉะนั้นการที่จะสั่งให้ตัวบัซเซอร์สร้างสัญญาณเสียงออกมาก็ต้องสร้างสัญญาณพัลส์(Pulse)ที่มีความถี่แตกต่างกันป้อนให้กับบัซเซอร์นี้

3.2.2.5 โมดูล Magnetic Relay

Magnetic Relay เป็นอุปกรณ์ที่ทำหน้าที่เป็นสวิตช์ตัวเปิดหรือปิดวงจร โดยอาศัยแรงดันไฟฟ้าที่จ่ายให้กับขดลวดที่พันอยู่บนแกนเหล็ก เพื่อสร้างสนามแม่เหล็กไฟฟ้าเกิดแรงดูดทำให้น้ำสัมผัสของสวิตช์ที่วางอยู่ใกล้ๆ สัมผัสกัน เพื่อต่อวงจรหรือแยกวงจรออกจากกัน น้ำสัมผัสสวิตช์ของรีเลย์แบ่งออกเป็น 2 แบบคือ

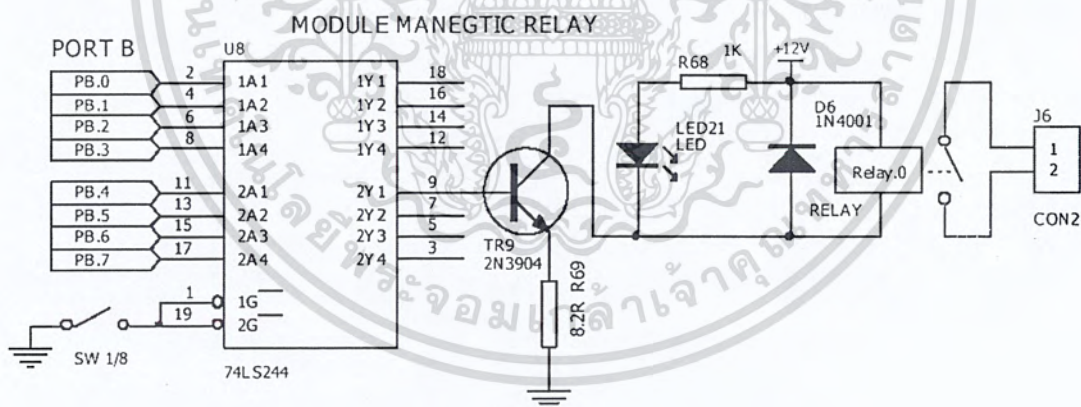
1. แบบปกติปิด (NC. Normal Close)
2. แบบปกติเปิด (NO. Normal Open)

แต่ละแบบจะทำงานไม่เหมือนกัน แบบปกติปิด หมายถึงขณะที่ยังไม่ได้ป้อนไฟให้กับขดลวดของ Relay หน้าสัมผัสของสวิตช์จะต่อสัมผัสกันอยู่ เมื่อขดลวดของรีเลย์ได้รับไฟเลี้ยงจะแยกออกจากกันทำให้วงจรเปิด แต่ถ้าเป็นแบบปกติเปิดก็จะทำงานตรงกันข้ามกับแบบปกติปิดดังแสดงในรูปที่ 3.29



รูปที่ 3.29 แสดงสัญลักษณ์ของ *Magnetic Relay*

บอร์ดทดลองอินพุตเอาต์พุต ส่วนของ Magnetic Relay จะใช้ Relay แบบปกติเปิด(N.O.) 4 ตัวต่อเข้ากับ พอร์ต PA4-PA7 โดยผ่าน บัฟเฟอร์ (buffer) 74 LS244 ในการที่จะทำการทดลองโมดูลนี้ต้องปรับสวิทช์ 1/8 ไปที่ตำแหน่ง “ON” ดังที่แสดงในรูปที่ 3.30



รูปที่ 3.30 วงจร *Magnetic Relay* ในหนึ่ง *Channel*

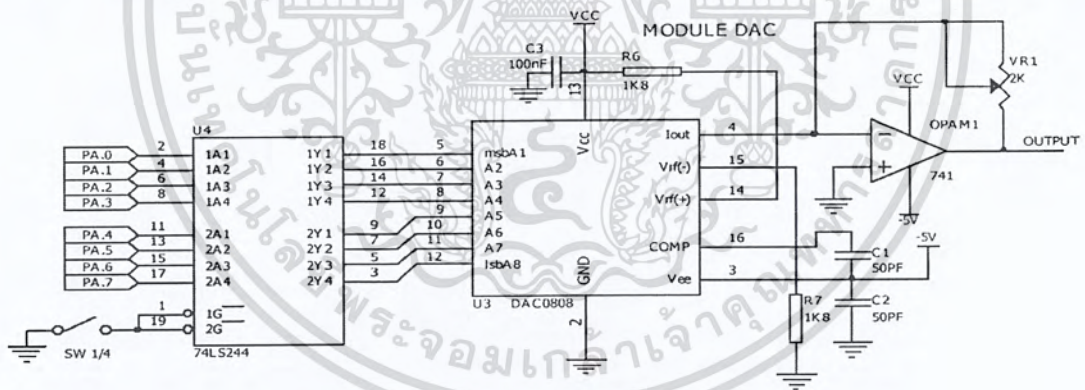
เมื่อเอาท์พุทจาก 8255 ออกมาเป็นค่า logic “1” ทำให้กระแสไฟฟ้าไหลผ่าน บัฟเฟอร์ (buffer) 74 LS244 ไปไบอัสทรานซิสเตอร์ 2N3904 ส่งผลให้รีเลย์ที่ต่อยุ่กับขาคอลเล็กเตอร์ของทรานซิสเตอร์ 2N3904 ครบวงจร กระแสไฟฟ้าไหลผ่านขดลวดของรีเลย์ ผ่านทรานซิสเตอร์ ผ่านตัวต้านทาน 8.2 โอห์ม ทำให้รีเลย์ทำงานดูดหน้าสัมผัสเข้าหากันต่อวงจร

ดังนั้นเมื่อต้องการให้รหัสตัวใดทำงาน จะต้องกำหนดให้บิตนั้นมีลอจิกเป็น “1” และจะหยุดทำงานเมื่อลอจิกนั้นเป็น “0” ในวงจรจะเห็นว่าไคโอดต่อคร่อมขดลวดของรีเลย์อยู่ ไคโอดตัวนี้เราเรียก

นี้เมื่อส่วนใดส่วนหนึ่งเกิดลัดวงจรหรือมีความเสียหายเกิดขึ้นจะไม่ส่งผลกระทบต่อส่วนอื่นและไม่มีจุดเชื่อมต่อกัน ในบอร์ดทดลองอินพุตเอาต์พุตโซลิดสเตตรีเลย์ ขั้วอินพุตจะต่อเข้ากับพอร์ต PB0-PB3 ของ 8255 โดยผ่าน buffer 74LS244 การทำงานคือเมื่อบิตนี้มีลอจิกเป็น“1”ทำให้อุปกรณ์เชื่อมโยงทางแสง MOC3040/MOC3041 ต่อวงจรทำให้ไครเอท 2N6073 สามารถนำกระแสได้ แต่ถ้าพอร์ต PB0-PB3 มีลอจิกเป็น“0” ไครเอทจะหยุดนำกระแสและแซลแนลนี้จะทนแรงดันได้ประมาณ $15V_{AC}$ และทนกระแสได้ประมาณ 1 แอมแปร์ในการทำงานของมันเราสามารถสังเกตได้ว่าทำงานหรือหยุดการทำงานโดยสังเกตที่ LED ที่ต่ออยู่ ถ้า LED ติดแสดงว่าเวลานี้มันทำงาน และถ้า LED ดับแสดงว่าเวลานี้มันหยุดการทำงาน ส่วนวงจรที่จะต่อเป็นเอาต์พุตหรือวงจรที่จะต่อเพื่อให้ Solid State ควมคุมนั้นเราสามารถนำมาต่อได้ที่ Con 2 วงจรที่จะนำมาต่อเป็นเอาต์พุตอาจจะเป็นวงจรที่ควบคุมการติด-ดับของหลอดไฟ หรือวงจรอื่นๆก็ได้

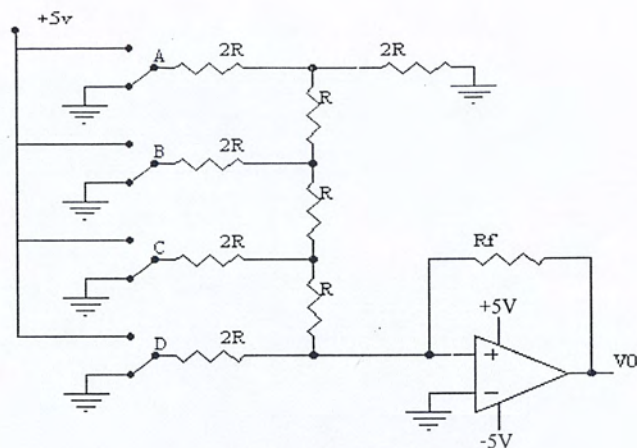
3.2.2.7 โมดูลการแปลงสัญญาณดิจิทัลเป็นอนาลอก (Digital to Analog Converter)

เพื่อเป็นการศึกษาและเรียนรู้การเปลี่ยนแปลงสัญญาณดิจิทัลเป็นอนาลอกอยู่ในบอร์ดทดลองอินพุตเอาต์พุตโดยใช้ DAC0808 เป็นตัวที่ทำการเปลี่ยนแปลงสัญญาณดิจิทัลเป็นอนาลอกการแปลงกลุ่มสัญญาณดิจิทัล (DIGITAL) ให้กลายเป็นปริมาณของแรงดันหรือกระแส ไฟฟ้าเราเรียกวิธีการนี้ว่า D/A (Digital to Analog Converse) ในบอร์ดทดลองอินพุตเอาต์พุต ถ้าทำการทดลองในโมดูลนี้จะปรับสวิตช์ 1/4 ไปที่ตำแหน่ง “ON”



รูปที่ 3.32 แสดงโมดูล Digital to Analog Converter

ในการแปลงสัญญาณดิจิทัลเป็นอนาลอกในที่นี้เราใช้ DAC0808 เป็นตัวที่ทำการเปลี่ยนแปลงสัญญาณดิจิทัลเป็นอนาลอกเมื่อแปลงสัญญาณออกมาเป็นอนาลอกแล้วสัญญาณที่ได้จะมีขนาดเล็กมากดังนั้นเราใช้ ออปแอมป์ (Op-amp) เป็นตัวขยายสัญญาณมากขึ้น การแปลงสัญญาณดิจิทัลให้เป็นอนาลอกจะใช้ ออปแอมป์เป็นตัวรวบรวมสัญญาณ ซึ่งภายในตัว DAC0808 การทำงานภายในของมันจะทำงานเหมือนวงจร R-2R Ladder ดังแสดงในรูปที่ 3.33



รูปที่ 3.33 วงจร R-2R Ladder

DAC0808 จะเป็นตัวทำการเปลี่ยนสัญญาณจาก Digital to Analog โดยที่เราสามารถคำนวณหาแรงดันที่เป็นเอาต์พุตออกมาได้ตามสมการของแรงเอาต์พุต

$$V_0 = (V_{ref}/R_f) \times R_0 \times D \text{ โวลต์}$$

เมื่อ $V_{ref} =$ แรงดันอ้างอิง 2.55 โวลต์

$D =$ เลขฐานสิบแปลงมาจากสัญญาณอินพุต

ในบอร์ดที่ทำขึ้นมานี้ในที่นี้ใช้ไอซี DAC0808 เป็นตัวแปลงสัญญาณดิจิทัล 8 บิตให้เป็นกระแสแล้วใช้ออปแอมป์ 741 แปลงกระแสที่ได้เป็นแรงดันให้อยู่ในช่วง 0-2.55V_{cc} จากวงจรสามารถที่จะคำนวณหาค่าแรงดัน V_0 ได้โดยอาศัยสมการ

$$V_0 = (V_{ref}/R_f) \times R_0 \times (D_7/2 + D_6/4 + D_5/8 + D_4/16 + D_3/32 + D_2/64 + D_1/128 + D_0/256)$$

เมื่อ $V_{ref} =$ แรงดันอ้างอิง (กำหนดให้ = 5 โวลต์)

$R_f =$ ตัวต้านทานที่ต่อระหว่างขา 13 และ 14 ของ DAC0808

$R_0 =$ ตัวต้านทานป้อนกลับของออปแอมป์

$D_0 =$ ลอจิกของสัญญาณดิจิทัล

ตัวอย่าง ป้อนสัญญาณดิจิทัลให้กับวงจรเป็น 10101010 จงคำนวณหาค่าแรงดัน V_0 คำนวณจากสมการ

$$V_0 = (V_{ref}/R_f) \times R_0 \times (D_7/2 + D_6/4 + D_5/8 + D_4/16 + D_3/32 + D_2/64 + D_1/128 + D_0/256)$$

$$V_{ref} = 5 \text{ โวลต์}$$

$$R_f = 1.8K$$

$$R_0 = 2K$$

$$V_0 = (5/1.8) \times 2 \times (1/2 + 0/4 + 1/8 + 0/16 + 1/32 + 0/64 + 1/128 + 0/256)$$

$$= 5.55 \times (0.5 + 0.125 + 0.03125 + 0.0078125)$$

$$= 5.55 \times 0.664$$

$$V_0 = 3.688 \text{ โวลต์}$$

3.2.2.8 โมดูล STEPPING MOTOR

สเต็ปป์มอเตอร์เป็นมอเตอร์ที่เหมาะสมสำหรับงานควบคุมการหมุน ที่ต้องการตำแหน่งที่แน่นอน สเต็ปป์มอเตอร์ต่างจาก DC มอเตอร์ตรงที่การหมุนไปที่ละสเต็ป อาจจะเป็นสเต็ปละ 0.9, 1.8, 5, 7.5, 15, หรือ 50 องศา ตามแต่ละชนิด ส่วน DC มอเตอร์จะหมุนแบบต่อเนื่อง ไม่สามารถหมุนเป็นสเต็ปได้ สเต็ปป์มอเตอร์จะแบ่งออกเป็นสามชนิดคือ

- Variable Reluctance (VR) เป็นแบบการหมุนได้อย่างอิสระในกรณีที่ไม่วางไฟให้โรเตอร์ โรเตอร์จะทำได้ด้วยเหล็กอ่อน มีลักษณะเป็นฟันเลื่อยรูปทรงกระบอก จะมีการหมุนสเต็ปหนึ่ง 7.5 องศาหรือ 15 องศา
- Permanent Magnet (PM) เป็นสเต็ปป์ที่มีโรเตอร์ทำได้ด้วยแม่เหล็กถาวร เป็นรูปทรงกระบอกฟันเลื่อยที่ตัวสเตเตอร์ (Stator) จะมีขดลวดพันอยู่ ตัวโรเตอร์จะหมุนไม่เป็นอิสระในกรณีที่ไม่ได้ป้อนไฟ เพราะแรงดูดจากแม่เหล็กถาวร
- Hybrid เป็นสเต็ปป์ที่นำเอาข้อดีของสเต็ปป์แบบ VR และ PM เข้ามาด้วยกันไว้ ซึ่งจะมีตัวสเตเตอร์คล้ายกับแบบ VR ที่โรเตอร์จะมีแม่เหล็กกำลังสูงติดอยู่ ทำให้แรงบิดสูง มีความแม่นยำในการหมุนสูง และมีขนาดกะทัดรัด ขนาดของสเต็ปป์ จะเป็น 0.9 องศา และ 1.8 องศา

สเต็ปเปอร์มอเตอร์ได้รับการพัฒนามากที่สุดและหาได้ง่ายคือ สเต็ปป์มอเตอร์แบบยูนิโพลาร์ (Uni polarsteppermotor) สเต็ปเปอร์มอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1, 2, 3 และ 4 มีการต่อสายออกมาจากขดลวด แต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปเปอร์มอเตอร์แบบนี้มีทั้งแบบ 5 สายและ 6 สายถ้าเป็นแบบ 5 สายจะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสอง มาต่อรวมกันเป็นสายเดียว สำหรับบอร์ดทดลองอินพุตเอาต์พุต จะมีวงจรควบคุมการจ่ายกระแสไฟไว้สำหรับสเต็ปป์มอเตอร์แบบ 2 Phase 6Wire วงจรควบคุมมอเตอร์จะต่อเข้ากับพอร์ต PB0-PB3 โดยผ่าน buffer 74LS244 ดังแสดงในรูปที่ 3.34 แต่ละเฟสจะมี LED แสดงสถานะของแต่ละเฟส เมื่อเอาต์พุตของพอร์ต PB มีค่าเป็น “1” ขดลวดสเตเตอร์ของสเต็ปป์มอเตอร์จะได้รับกระแสไฟฟ้าหลอด LED ก็จะติดสว่าง ในขณะนั้นสเต็ปป์มอเตอร์จะหมุนรอบตามการกระตุ้นในการเลือกใช้โมดูลสเต็ปป์มอเตอร์นี้จะต้องปรับสวิตช์ 1/5 ไปที่ตำแหน่ง “On”

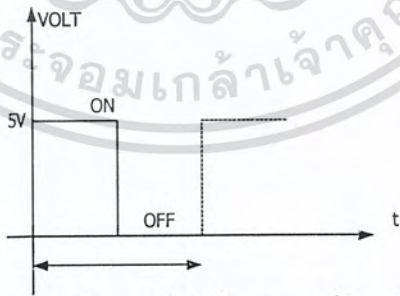
การควบคุมสตีปีงมอเตอร์ ก็คือการควบคุมกระแสไฟที่จ่ายให้กับขดลวดสเตเตอร์แต่ละขด

Phase Driving Method	Phase 1 Excitation				Phase 2 Excitation				Phase 1-2 Excitation							
	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8
A	1				1	1			1	1	1					
B		1				1	1				1	1	1			
/A			1				1	1					1	1	1	
/B				1	1			1	1						1	1

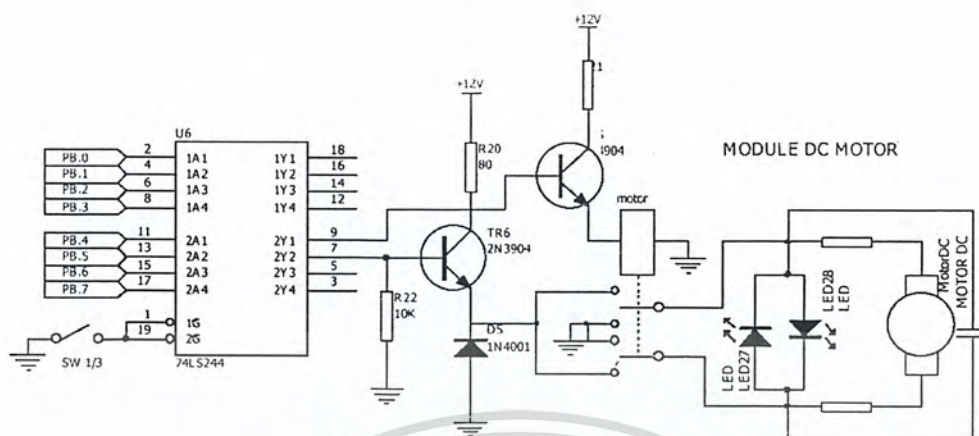
ตารางที่ 3.4 ตารางการกระตุ้นเฟสแต่ละแบบของสตีปีงมอเตอร์

3.2.2.9 โมดูลการควบคุมมอเตอร์ (Motor control Module)

Motor control ในการควบคุมความเร็วของมอเตอร์ให้ความเร็วคงที่ ตามที่เราต้องการเป็นเรื่องทำที่ค่อนข้างยาก ถ้าใช้การควบคุมแบบธรรมดาทั่วไป คือ การควบคุมแรงดันที่จ่ายให้กับมอเตอร์การจะควบคุมความเร็วของมอเตอร์ให้ได้ความเร็วที่คงที่ ต้องควบคุมกระแสที่จ่ายให้กับมอเตอร์ ซึ่งวิธีนี้เรียกว่า การควบคุมแบบ Pulse Width Modulation (PWM) เป็นการควบคุมกระแสที่จ่ายให้กับมอเตอร์ โดยการจ่ายแรงดันให้กับมอเตอร์เป็นช่วงๆ กำลังไฟฟ้าที่ได้รับจะเป็นค่าเฉลี่ยที่เกิดขึ้นในช่วง “ON” โดยความถี่ของการ ON และ OFF ลงที่ดังแสดงในรูปที่ 3.35 เมื่อต้องการโมดูลการควบคุมมอเตอร์ในการทำงานต้องปรับสวิตช์ 1/6 ไปที่ตำแหน่ง “On”



รูปที่ 3.35 แสดงการจ่ายแรงดันให้กับมอเตอร์เป็นช่วงๆ



รูปที่ 3.36 วงจรควบคุม Motor control

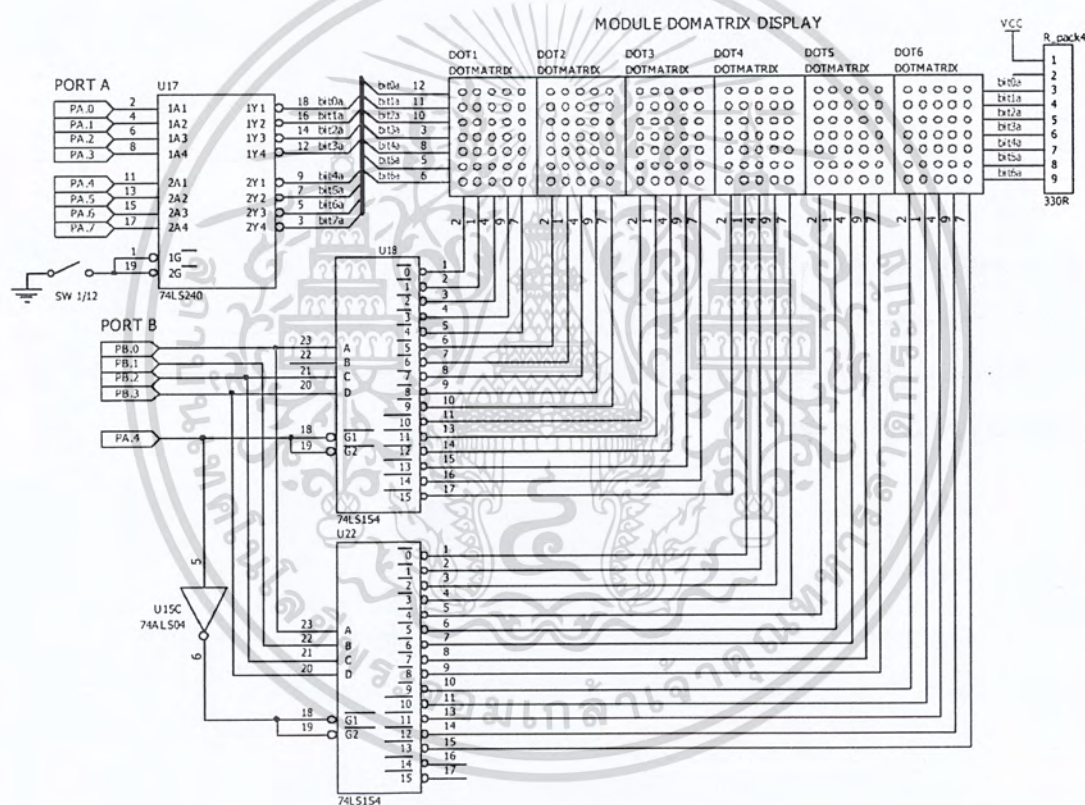
เมื่อความถี่ “ON” มีเวลาที่ยาวนานกว่าช่วง “OFF” กำลังไฟฟ้าเฉลี่ยที่มอเตอร์ได้รับก็จะมีมากขึ้น ความเร็วของมอเตอร์ ก็จะเพิ่มขึ้น และถ้าหากต้องการลดความเร็วรอบของมอเตอร์ลง ก็ให้ลดช่วง “ON” ให้มีเวลาน้อยลง

สำหรับบอร์ดทดลองอินพุตเอาต์พุต แสดงในรูปที่ 3.22 สัญญาณควบคุมจะต่อกับพอร์ต PB บิตที่ 4 จะเป็นบิตควบคุมทิศทางหมุนของมอเตอร์ โดยที่บิตนี้จะต่อเข้ากับขา Base ของ ทรานซิสเตอร์ 2N3904 เพื่อทำให้รีเลย์เปิดหรือปิด สำหรับ PB5 เป็นบิตควบคุมการทำงานของมอเตอร์ โดย โดยที่บิตนี้จะต่อเข้ากับขา Base ของทรานซิสเตอร์ BFY51 เพื่อขยายกระแสที่จะใช้ขับให้มอเตอร์หมุน เมื่อ สัญญาณที่จ่ายเข้ามามีสถานะเป็น “1” ทรานซิสเตอร์ BFY51 ก็จะต้องวงจรทำให้มอเตอร์หมุนและถ้า สัญญาณที่จ่ายเข้ามามีสถานะเป็น “0” ทรานซิสเตอร์ BFY51ก็จะเปิดวงจรทำให้มอเตอร์หยุดหมุน

3.2.2.10 ชุดแสดงผลข้อความ DOTMATRIX DISPLAY

เป็นส่วนที่ศึกษาการแสดงผลที่สร้างให้เกิดเป็นข้อความหรือรูปลักษณะต่างๆ ตามที่เราต้องการได้ Dot Matrix Display คือกลุ่มของหลอด LED มารวมกันให้เป็นกลุ่มในแนวนอนและแนวตั้ง รูปแบบของการแสดงผลก็จะเปลี่ยนไปตามการติด-ดับของหลอด LED ใน Dot Matrix Display มีด้วยกันหลายแบบ หลายขนาดเช่น ขนาด 4X4,5X7,5X8,7X5,8X8,16X16 มีทั้งแบบสีเดียว (Single Color) หรือแบบหลายสี (Multi Color) และยังแบ่งตามการใช้งานเป็นแบบ Anode Column, Cathode Row และ Cathode Column, Anode Row บอร์ดทดลองอินพุตเอาต์พุตนี้ใช้ Dot matrix ขนาด 5X7 แบบ Cathode Column, Anode Row จำนวน 6 ตัวดังแสดงในรูปที่ 3.37 และต่อ Dot matrix เข้ากับพอร์ต PA0-PA7 โดยผ่าน buffer 74LS240 เป็นบิตกำหนด Row และต่อกับตัวต้านทาน(R_pack4) เพื่อทำการ Pull up สำหรับ PB0-PB3 ต่อกับ Column โดยผ่าน DECODERS 74LS154 จำนวน 2 ตัว และใช้ PB4 เป็นตัวเลือกที่ Decoder 74LS154 ตัวที่ 2 นั้นจะต่อกับ Inverter เพื่อให้ทำงานสลับกัน ในการทำงานให้แสดงเป็นข้อความหรือเป็นรูปแบบต่างนั้น ข้อมูลที่จะแสดงนั้นแน่นอนจะมีจำนวนที่มากแต่ใน Dot matrix ที่มีในวงจรข้างล่างจะ

เห็นว่าข้อมูลจะต่อกันทุกตัว ดังนั้นเพื่อแสดงข้อมูลจำนวนมากเหล่านั้นให้เกิดเป็นข้อความหรือรูปแบบต่างๆตามที่ต้องการนั้น จะส่งข้อมูลจำนวนนั้นออกมาที่ขาข้อมูลของ Dot matrix ที่ละชุดคือที่ละ 8 บิต แล้วก็ Decode ตำแหน่ง Column ที่จะให้ติดนั้นออกมาโดย Decode ผ่าน Decoder 74LS154 ก็จะทำให้ Dot matrix ติดตามข้อมูลที่ส่งออกมานั้น ตำแหน่งที่ Decode ใวนั้น แล้วหน่วงเวลาไว้ช่วงเวลาหนึ่งเพื่อให้เรามองเห็น และข้อมูลชุดต่อไปก็ทำเช่นเดียวกับนี้ เมื่อทำอย่างนี้ตลอดไปเราก็สามารถที่จะส่งข้อมูลออกไปที่ Dot matrix ได้ตามที่เรต้องการ การที่เห็นข้อความปรากฏบน Dot matrix ก็เป็นการส่งข้อมูลออกมาแบบนี้ คือส่งข้อมูลออกมาชุดหนึ่งแล้ว Scan ไป Column หนึ่งแต่เนื่องจากว่าการทำงานนั้นเร็วมากจนกระทั่งสายตาของเราจับไม่ทันว่าติดที่ละ Column ดังนั้นเราจึงเห็นเกิดเป็นข้อความหรือรูปแบบต่างๆตามข้อมูลที่เรานั้นส่งออกมานั้น เมื่อต้องการให้โมดูล แสดงผลหรือต้องการทำการทดลองโมดูลนี้ต้องปรับ สวิตช์ 1/12 ไปที่ตำแหน่ง “On”



รูปที่ 3.37 วงจรชุดแสดงผลข้อความ

บทที่ 4

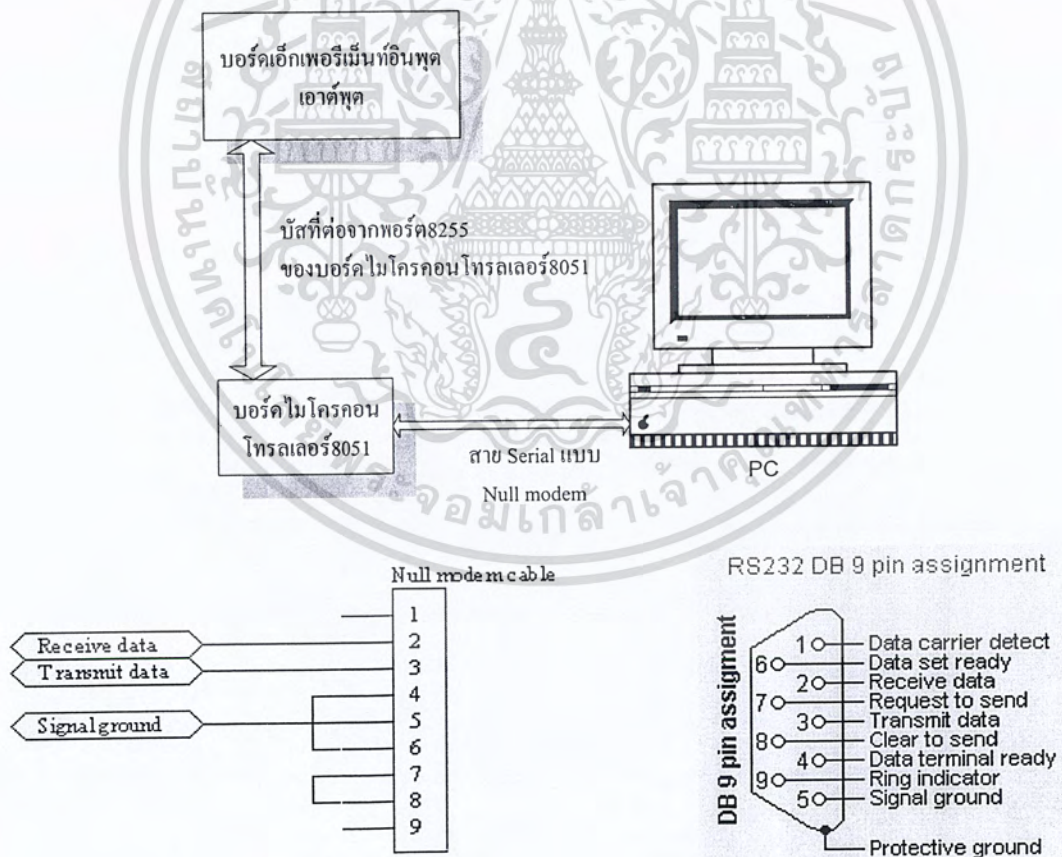
การใช้งานซอฟต์แวร์ (Software)

ในการใช้งานซอฟต์แวร์ (Software) ของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองนี้ จะมีการนำใช้ต่อไปนี้ สำหรับบอร์ดไมโครคอนโทรลเลอร์ 8051 เพื่อที่จะให้สามารถดาวน์โหลดโปรแกรมจาก PC ลงมาที่บอร์ดไมโครคอนโทรลเลอร์นั้นเราต้องได้ทำการลง Software บางตัวก่อน คือโปรแกรมการสื่อสาร XTALK (ที่มีมาใน Compact Disk :CD) ที่ให้มานี้

4.1 ขั้นตอนการใช้งาน

1. ต่อสาย Serial พอร์ตของบอร์ดไมโครคอนโทรลเลอร์ 8051 เข้ากับพอร์ตอนุกรม Com1 หรือ Com2 ตามที่เราต้องการ หลังจากนั้นให้ Copy Directory XTALK ที่มีใน CD(ใน CD ที่ประกอบมานี้มี Soft ware XTALK และ SXA51) ลงไปไว้ที่ Drive C ของเครื่อง

2. ต่อสายบัส (Buses) ของบอร์ดไมโครคอนโทรลเลอร์ 8051 เข้ากับ บอร์ดทดลองอินพุตเอาต์พุต แล้วต่อไฟเลี้ยงให้กับบอร์ด ดังรูปที่ 4.1.



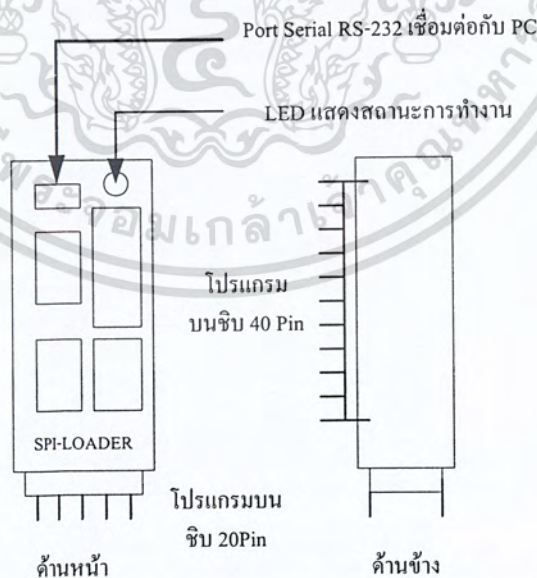
รูปที่ 4.1 แสดงการต่อบอร์ดไมโครคอนโทรลเลอร์และการต่อสาย Serial แบบ Null modem สำหรับการทดลองกับ PC

ความมุ่งหมายที่สำคัญในการซอร์ท(Sort) ขาที่ 4 กับ 6 และ 7 กับ 8 เพื่อกำหนดให้ซอร์ฟแวร์ (Software) เห็นว่ามีการทำแฮนด์เช็กกิ้ง (Hand Shacking) ได้ผลจาก เคเบิล (Cable)

3. ใช้อุปกรณ์ดาวน์โหลด SPI-LOADER ต่อกับ Socket 8051 ที่บอร์ดไมโครคอนโทรลเลอร์โดยพอร์ตที่มี 9Pin ต่อเข้ากับ Com1หรือ Com2 ของ PC ดังรูปที่4.2 จากนั้นไปที่ MS DOS แล้ว Set mode การทำงานของพอร์ตอนุกรม COM1 หรือ COM2 ตามที่เราต่อใช้งานอยู่นั้น ด้วยอัตรา BAUD RATE = 4800, DATA-BIT = 8, STOP-BIT =1, PARITY = NONE โดยพิมพ์ที่ MS DOS คือ Mode COM1 : 48,n,8,1,p สำหรับ COM1 หรือ Mode com2 : 48,n,8,1,p สำหรับ COM2 แล้ว Enter จะปรากฏเหมือนรูปที่ 4.2

```
C:\>
C:\>MODE COM1:48,N,8,1,P
Status for device COM1:
-----
Baud:          4800
Parity:        None
Data Bits:     8
Stop Bits:     1
Timeout:       OFF
XON/XOFF:      OFF
CTS handshaking: ON
DSR handshaking: ON
DSR sensitivity: OFF
DTR circuit:   HANDSHAKE
RTS circuit:   HANDSHAKE
```

รูปที่ 4.2 การกำหนดพอร์ตอนุกรม COM1



รูปที่4.3 แสดงอุปกรณ์ช่วยในการดาวน์โหลดโปรแกรมจาก PC ไปที่บอร์ดไมโครคอนโทรลเลอร์

4. เรียกใช้ Batch file ใน Directory XTALK หรือ S1 สำหรับการดาวน์โหลดผ่านพอร์ตอนุกรม COM1 และ S2 สำหรับผ่านพอร์ตอนุกรม COM2 โดยปกติแล้วเมื่อจ่ายไฟเข้าไปที่บอร์ดไมโครคอนโทรลเลอร์ บอร์ดจะทำงานใน Run Mode โดยที่ LED จะกะพริบ ช้าๆ ก่อนเสมอ เมื่อผู้ใช้เลือกใช้ S1.bat หรือ S2.bat แล้วกด Spacebar ที่ Key board PC หนึ่งครั้ง บนหน้าจอ PC จะแสดงข้อความที่แสดงถึงการติดต่อก็คือ

**** SPI-LOADOADER V2.0 EDditMODE Baudrate 4800.. SILA 2000. ****

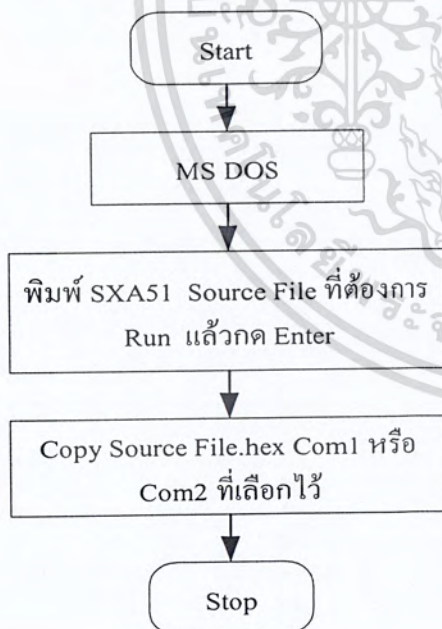
ซึ่งเป็นชื่อของอุปกรณ์ที่ช่วยในการดาวน์โหลดที่มีอัตรา Baudrate 4800

5. ไปที่ Edit Mode จะสังเกตเห็น LED ติดสว่างค้างอยู่ แสดงว่าพร้อมที่จะใช้งานแล้ว หลังจากนั้นให้ Key ตัว Q เข้าไป เพื่อเป็นการออกจาก Edit Mode และพร้อมที่จะรับ การดาวน์โหลดโปรแกรม (สำหรับ XTALK นี้ไม่สามารถใช้ได้กับ Window NT.)

6. ลงซอฟต์แวร์ SXA51 ASSEMBLER ใช้สำหรับการแปล Source Program ที่เป็น File.ASM ให้เป็น File.HEX โดยการ Copy เอา SXA51 ที่เป็น File.EXE ที่ให้มาใน CD แล้วไปวางที่ตำแหน่งที่ Save โปรแกรมไว้.

7. ที่ MS DOS Editor เขียนโปรแกรมที่ต้องการใช้เสร็จแล้วให้ Save เป็น File ที่มีนามสกุล เป็น .ASM จากนั้นไปที่ Menu File Exit เพื่อไปที่ MS DOS

8. ที่ MS DOS พิมพ์ SXA51 แล้วกด Space หนึ่งครั้งจากนั้นพิมพ์ SOURCE FILE ที่ต้องการตามโฟลว์ชาร์ตนี้



ให้สังเกต LED กระพริบตามข้อมูลที่โหลดเข้ามา เมื่อการโหลดโปรแกรมสิ้นสุด จะเปลี่ยนโหมดเข้าสู่การรันโปรแกรมนั้นโดยอัตโนมัติ (LED กระพริบตามปกติ)

รูปที่ 4.4 โฟลว์ชาร์ตการ RUN และดาวน์โหลดโปรแกรม

4.2 รายละเอียดทั่วไปของ SXA51 ASSEMBLER

SXA51 เป็นโปรแกรม 8051/51 ครอสแอสเซมเบลอร์ ใช้สำหรับทำการแปล SOURCE PROGRAM ซึ่งเขียนเป็นภาษาแอสเซมบลีของ MCS-51 (FILE.ASM) จะได้เอาท์พุทไฟล์เป็น INTEL HEX FILE (FILE.HEX) การเขียน SOURCE FILE ให้กระทำจากโปรแกรม EDITOR ใดๆ ก็ได้ จากนั้นจึงนำมาแปลด้วย SXA51 ซึ่งทำได้โดยการเรียกใช้โปรแกรม SXA51.EXE และตามด้วยชื่อ SOURCE PROGRAM ที่เขียนขึ้น การแปลของโปรแกรม SXA51 จะกระทำในแบบ 2 PASS ASSEMBLER ซึ่งหากมีข้อผิดพลาดใด ๆ SXA51 จะแสดงให้ทราบในขณะที่แปล ลักษณะของ SOURCE PROGRAM จะเป็นไปตามมาตรฐานของการเขียนโปรแกรม ASSEMBLY ทั่วไป

4.2.1 การใช้งาน SXA51 ASSEMBLER

โปรแกรมแอสเซมเบลอร์ที่ใช้งานคือ SXA51.EXE โดย ASSEMBLER จะทำการแปล SOURCE FILE ซึ่งถูกกำหนดไว้เป็น FILE.ASM เมื่อทำการแปลแล้ว จะได้ FILE OUTPUT เป็น INTEL-HEX FILE ซึ่งถูกกำหนดไว้เป็น FILE.HEX โดย FILE.HEX นี้เอง ที่ผู้ใช้สามารถนำไปโหลดเข้า บอร์ดต่างๆ หรืออัดเข้าตัว EPROM เพื่อนำไปทดสอบโปรแกรมตามที่เขียนได้ นอกจากนี้ SXA51 ยังสามารถสร้าง LISTING FILE ต่าง ๆ ได้ตามรูปแบบของ OPTION ดังนี้

A>SXA51 <OPTION> SOURCE FILE

โดย

SOURCE FILE : คือชื่อ FILE ของ โปรแกรม กรณีที่ไม่ใส่نامสกุล ของ FILE ตัว

SXA51 จะกำหนดให้เป็น .ASM โดยอัตโนมัติ

<OPTION> : เป็นส่วนของการกำหนด OPTION ต่างๆ ประกอบด้วยเครื่องหมาย

“-“ และตามด้วยตัวอักษรย่อเพียงตัวเดียว ซึ่งการกำหนด OPTION นี้

สามารถกำหนดได้มากกว่า 1 ตัวในการแปลแต่ละครั้ง รายละเอียด

ของ OPTION จะมีดังนี้

-L หมายถึงให้สร้าง FILE แสดงรายละเอียด (LISTING FILE)

โดย OUTPUT FILE ที่ได้คือ FILE.LST

-N หมายถึงให้ทำการคอมไพล์ โดยไม่สร้าง FILE OUTPUT

ใดๆ ทั้งนี้เพื่อการตรวจสอบความถูกต้องของโปรแกรมอย่างรวดเร็ว

-C หมายถึงให้สร้าง FILE แสดงรายละเอียด เช่นเดียวกับ

-L โดยเพิ่ม

symbol cross-reference ไว้ที่ท้ายของ FILE ทั้งนี้เพื่อความสะดวกใน

การค้นหาดำแหน่งต่าง ๆ ตามชื่อ LABEL

-D หมายถึง ให้แสดง PROCESS ขณะที่กำลังทำการ ASSEMBLER

โดยโปรแกรมจะแสดงเลขที่บรรทัดที่กำลังแปลอยู่

ตัวอย่างการใช้คำสั่ง

A>SXA51 J31EX4

หมายถึง SOURCE : J31EX4.ASM
LISTING : ไม่สร้าง
HEX : สร้าง HEX FILE ชื่อ J31EX4.HEX

A>SXA51 -L J31EX4.ASM

หมายถึง SOURCE : J31EX4.ASM
LISTING : สร้าง LISTING FILE ชื่อ J31EX4.LST
HEX : สร้าง HEX FILE ชื่อ J31EX4.HEX

A>SXA51 -N J31EX4.ASM

หมายถึง SOURCE : J31EX4.ASM
LISTING : ไม่สร้าง
HEX : ไม่สร้าง

ข้อกำหนดของ SOURCE FILE

1. ในแต่ละบรรทัดต้องมีคำสั่งในภาษาแอสเซมบลี หรือ ASSEMBLY DIRECTIVE เพียงหนึ่งคำสั่งเท่านั้น
2. LABEL DECLARATION ทุก LABEL ต้องปิดท้ายด้วย ":"
3. โปรแกรม SXA51 จะมองตัวอักษรตัวเล็กหรือตัวใหญ่เหมือนกัน เช่น hh: จะเหมือนกับ HH:
4. บรรทัดว่างในโปรแกรมต้นแบบจะยังคงปรากฏอยู่ใน FILE แสดงรายละเอียด (FILE.LST) แต่ใน FILE แสดงรหัสภาษาเครื่อง (FILE.HEX) จะถูกตัดทิ้งไป
5. SOURCE FILE ในแต่ละบรรทัดต้องประกอบด้วยตัวอักษรไม่เกิน 128 ตัวอักษร ซึ่งในแต่ละบรรทัดต้องมีรูปแบบดังนี้

[LABEL:] OPERATION [OPERAND1],[OPERAND2] [;COMMENT]

LABEL เป็นกลุ่มของตัวอักษรและตัวเลขที่จะใช้เป็นตัวแปร หรือตำแหน่งหน่วยความจำสำหรับอ้างอิงในโปรแกรม LABEL ต้องขึ้นต้นด้วยตัวอักษร A-Z หรือ "_" ":" "?" และต้องปิดท้ายด้วย ":" เสมอ

OPERATION ส่วนนี้จะเป็นคำสั่งมาตรฐานของ MCS51 เช่นคำสั่ง MOV INC DEC ADD

OPERAND	เป็นกลุ่มของตัวอักษรและตัวเลข ซึ่งอาจจะอยู่ในรูปของการกระทำทางคณิตศาสตร์ หรือการกระทำทางด้าน LOGIC
COMMENT	หมายถึงประโยคหรือข้อความที่ตามหลังเครื่องหมาย “;” โดยทั่วไปมักจะใช้อธิบายการทำงานของโปรแกรมสั้นๆ เพื่อให้ทำความเข้าใจได้ง่ายขึ้น เครื่องหมาย “;” จะอยู่ส่วนใดของโปรแกรมก็ได้ ซึ่งข้อความที่ตามหลังเครื่องหมาย “;” จะไม่มีผลต่อการแปลของ ASSEMBLER

ASSEMBLER DIRECTIVE

DB expression [,expression...]

DB (DEFINE BYTE) คำสั่งนี้ใช้กำหนดค่าข้อมูลเป็นไบต์ลงในหน่วยความจำโปรแกรม สามารถใส่ข้อความรหัสแอสกี(ASCII STRING) ลงตำแหน่งในหน่วยความจำได้ โดยข้อความนั้นต้องอยู่ในเครื่องหมาย QUOTES “” และยังกำหนดค่าข้อมูลที่เป็นตัวเลขและตัวอักษรในบรรทัดเดียวกันได้โดยการค้นด้วยเครื่องหมาย “,” เช่น

POINTER DB ‘This is a string’,0

DS expression

DS (DEFINE SPACE) ใช้สำหรับกำหนดที่ว่างเป็นจำนวนไบต์สำหรับเก็บค่าตัวแปรหรือใช้เป็นส่วนของ BUFFER

DW expression [,expression...]

DW (DEFINE WORD) ใช้ในการเขียนข้อมูล 16 บิต ของหน่วยความจำโปรแกรมโดยการกำหนดค่าข้อมูลไบต์ต่ำก่อนแล้วตามด้วยข้อมูลไบต์สูง โดยข้อมูลแต่ละ WORD ต้องค้นด้วยเครื่องหมาย “,” เช่นเดียวกับคำสั่ง DB

ORG expression

ORG (ORIGIN) ใช้สำหรับกำหนด ADDRESS เริ่มต้นของโปรแกรม (ORIGIN PROGRAM LOCATION)

END

END คำสั่งนี้มีไว้เพื่อบอกให้ ASSEMBLER ทราบถึงจุดสิ้นสุดของโปรแกรมว่าอยู่ที่ตำแหน่งใดคำสั่งนี้จำเป็นต้องใส่ไว้ท้ายโปรแกรมด้วย

SYMBOL_NAME EQU expression

SYMBOL_NAME BIT expression

สำหรับการกำหนดค่าให้กับ SYMBOL_NAME (หรือที่เรียกกันว่า LABEL) โดยสามารถใช้คำสั่ง EQU หรือคำสั่ง BIT ก็ได้ ซึ่งจะทำงานเหมือนกันทุกประการ

SYMBOL_NAME SET expression

คำสั่ง SET จะทำงานเหมือนกับ EQU ทุกประการ แต่จะสามารถกำหนดให้ SYMBOL_NAME มีค่าได้หลาย ๆ จุดตามที่ต้องการ

Expression

การกำหนดค่าใน Expression ของแต่ละคำสั่งในภาษา ASSEMBLER นั้น สามารถใส่ตัว OPERATOR ต่าง ๆ เพื่อความสะดวกในการใช้งานได้ รวมทั้งยังกำหนดเป็นเลขฐานต่างๆ ได้ด้วยสรุปได้ดังนี้

1. โดยปกติตัวเลขจะถือว่าเป็นเลขฐานสิบเสมอ แต่ถ้าต้องการกำหนดเป็นเลขฐานสิบหก (HEXADECIMAL) จะต้องเขียนนำหน้าด้วยตัวเลข และลงท้ายด้วยตัวอักษร H เช่น 12H, 0F7H และถ้าต้องการกำหนดเป็นเลขฐานแปด (OCTAL) ให้ลงท้ายด้วยตัวอักษร O หรือ Q และถ้าต้องการเป็นเลขฐานสอง (BINARY) ให้ลงท้ายด้วย B
2. สัญลักษณ์ S จะหมายถึงการอ้างตำแหน่ง ADDRESS ในบรรทัดนั้น ๆ
5. สามารถใช้คำว่า HIGH หรือ LOW เพื่อกำหนดค่าเฉพาะส่วน สูง หรือ ต่ำได้ เช่นถ้ามีการกำหนดค่าให้ชื่อ TEST EQU 1234H สามารถเรียกค่าส่วนสูงได้คือ MOV A,#HIGH TEST ซึ่งจะทำให้ได้ค่า 12H ตามต้องการ
6. สามารถกำหนด OPERATOR ในการคำนวณได้คือ + - * / ^ (exponentiation)
7. สามารถกระทำทางด้าน LOGIC ได้คือ MOD SHR SHL AND OR XOR

BIT OPERATION

SXA51 มีความสามารถอ้างตัวแปรในระดับ BIT ได้ โดยใช้หลักการ h.m ซึ่ง n คือค่า BYTE ADDRESS และ m คือ หมายเลข BIT ADDRESS เท่านั้น เช่น 22H.3 จะมีค่าเท่ากับ 13H รวมทั้งจะใช้กับชื่อมาตรฐานได้ด้วย เช่น ACC.7 P1.2

SXA51 ERROR MESSAGE

Byte

B Value out-of-bounds Must be between 0 and 255

Control

- C1 A primary control must be used before any code generation
- C2 Unrecognized control

Expression

- E1 Unbalanced parentheses
- E2 cannot translate variable name
- E2 Devide by zero
- E4 Incorrect bit address
- E5 Ganeral expression error
- E6 Radix error
- E7 Bit.n: n must be between 0-7

Include

- I Include nesting too deep

Multiply define

- M1 Label multiply defined
- M2 EQU or SET multiply defined

Offset error

- 01 ACALL or AMP must be inside 4096 byte syntex
- 02 SJMP must be within +127 or -128

Syntex

- S General
- S1 First char. on line bust be alphabetic
- S2 Label error: propably a reserved word
- S3 First char.(after label) must be alphabetic
- S4 Reseved eord in EQU or SET
- S5 Incorrect use of opcode
- S6 Cannot evaluate ORG expression
- S7 Incorrect register or data type
- S8 Cannot DEC DPTR

S9	Expected 'A'
S10	Expected comma
S11	Expected 'AB'
S12	Expected '@A+DPTR'
S13	Expected '@A+PC' or '@A+DPTR'
S14	Expected 'A,@DPTR','A,@R0' or 'A,@R1'
S15	Expected '@R0' or '@R1'
S16	SETB operand must be Bit or C
S17	String must be enclosed in quotes or is too long
Unrecognized	
U	General
U0	Opcode

4.2.2 ข้อแนะนำในการใช้ SXA51

1. BUG ของ SXA51 ที่ต้องหลีกเลี่ยง

ในกรณีที่ผู้ใช้ใช้คำสั่ง MOV BUFFER,#35H โดยชื่อ BUFFER คือส่วนหนึ่งในโปรแกรมที่ได้ทำการกำหนดเอาไว้ด้วยคำสั่งเทียบ DS หรือ คำสั่งเทียบ EQU ในกรณีนี้ถ้าการกำหนดชื่ออยู่ก่อนการใช้อ้างอิงก็จะมีปัญหาอะไร แต่ถ้าอยู่หลังการใช้อ้างอิงจะเกิดปัญหาการแปลไม่ถูกต้อง และที่สำคัญคือ ไม่มีการบอก ERROR ให้ผู้ใช้ได้ทราบ ดังนั้นขอให้จำหลักการง่าย ๆ ดังนี้คือ การกำหนดค่าให้กับ LABEL ไม่ว่าจะจากคำสั่งเทียบ DS หรือ EQU ก็ตาม ให้กระทำที่ต้นโปรแกรมเสมอ ซึ่งก็จะใช้งานไปได้อย่างดี

2. เทคนิคการใช้ตัวแปร S

คือตัวแปรที่ใช้แทนค่า ADDRESS ของบรรทัดนั้นๆ ซึ่งจะทำให้ผู้ใช้สามารถอ้างอิงถึง ADDRESS ในระยะใกล้ๆ ได้ โดยที่ไม่ต้องกำหนด LABEL ให้มากเกินไปจนความจำเป็น

ตัวอย่าง CJNE A,#1,\$+6 ;ถ้า A ไม่เท่ากับ 1 ให้ JUMP ไปอีก 6 BYTE คือไปทำคำสั่ง CJNE ต่อไป LJMP MAIN1 นั่นเอง

```

CJNE A,#2,$+6
LJMP MAIN2
CJNE A,#3,$+6
LJMP MAIN3
LJMP MAIN4

```


3. ความสามารถในการอ้าง LABEL เฉพาะส่วน

กรณีที่ผู้ใช้กำหนด LABEL ดังนี้ TEST EQU 147FH ชื่อ TEST จะมีค่าตัวแปรเป็น 16 BYTE และถ้าผู้ใช้ต้องการอ้างอิงแต่ละ BYTE ของชื่อนี้ก็สามารถทำได้ดังนี้

```
MOV R2,#HIGH TEST ;จะได้ค่าเท่ากับ 14H
```

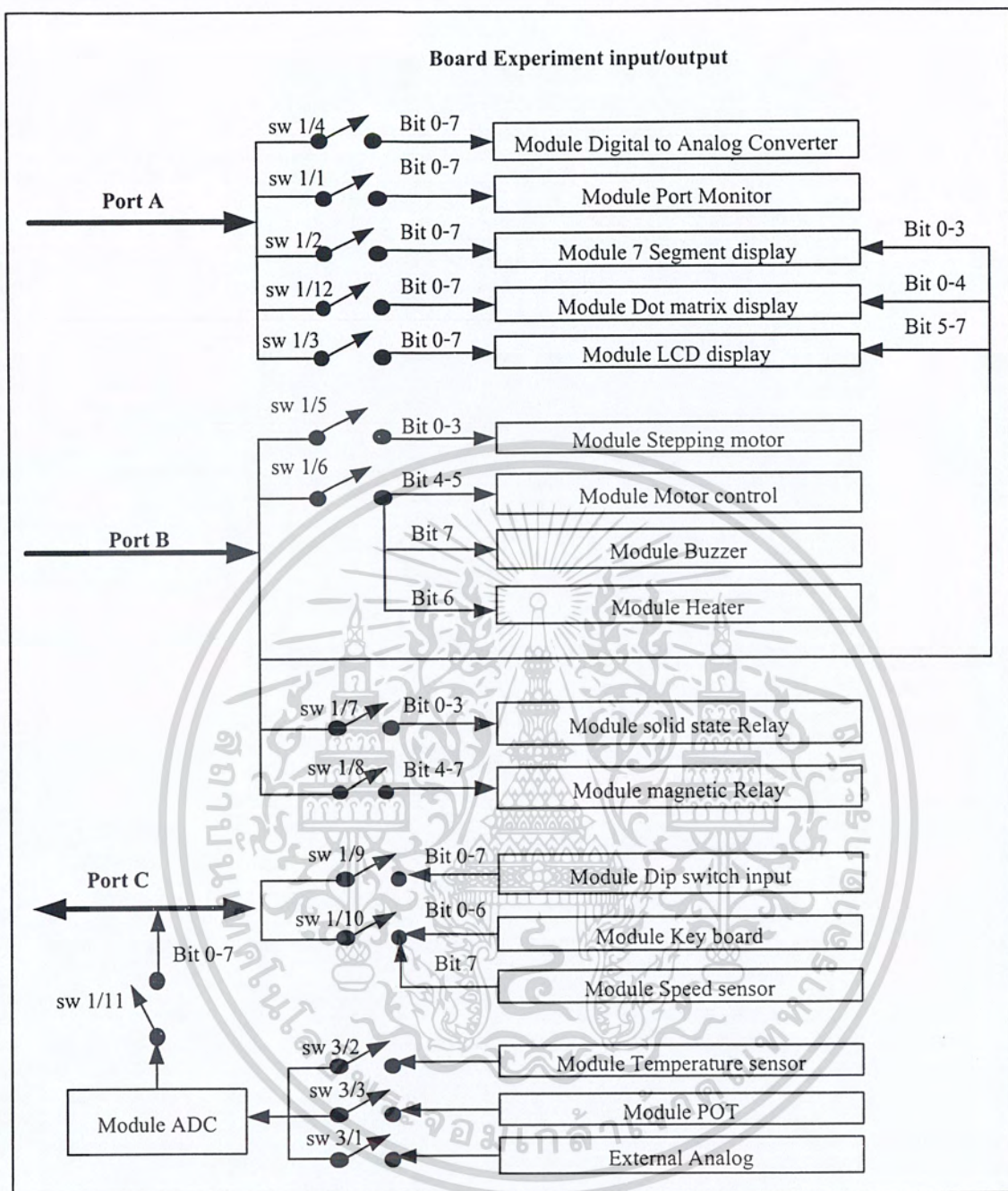
```
MOV R3,#LOW TEST ;จะได้ค่าเท่ากับ 7FH
```

4.3 การใช้งานบอร์ดทดลองอินพุตเอาต์พุต

ทำการต่อบอร์ดไมโครคอนโทรลเลอร์ 8051 เข้ากับบอร์ดทดลองอินพุตเอาต์พุตแล้วและต่อบอร์ดไมโครคอนโทรลเลอร์ 8051เข้ากับ PC จากนั้น Download โปรแกรมจาก PC ลงไปที่บอร์ดไมโครคอนโทรลเลอร์ 8051 เพื่อไปควบคุมบอร์ดทดลองอินพุตเอาต์พุต

4.3.1 การใช้งานโมดูลที่เป็นอินพุตและเอาต์พุต (Input/Output Module)

ในการใช้งานแต่ละส่วนจำเป็นต้องเลือก Dip-Switch ที่ต่อกับวงจรส่วนที่ต้องการใช้ซึ่งมีในวงจรซึ่งรายละเอียดรูปที่ 4.5

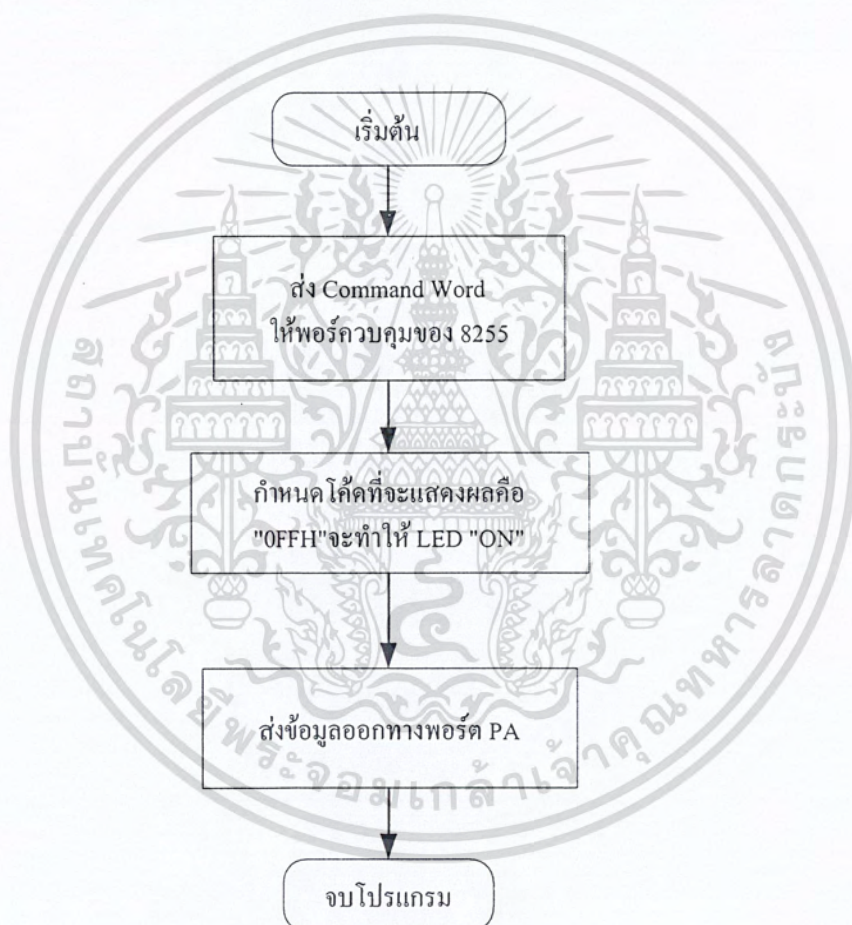


รูปที่ 4.5 แสดง Block diagram ของบอร์ดทดลองอินพุตเอาต์พุต

4.3.2 โมดูลพอร์ตมอนิเตอร์ (Port Monitor Module)

1. ต่อสายบัสน์เข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) แล้ว Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. คาวนโหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

เมื่อต้องการใช้งาน Module Led ซึ่งเป็นส่วนที่แสดงผลขนาด 8 บิต จะแสดงขึ้นตอนตามโฟลว์ชาร์ต รูปที่ 4.6

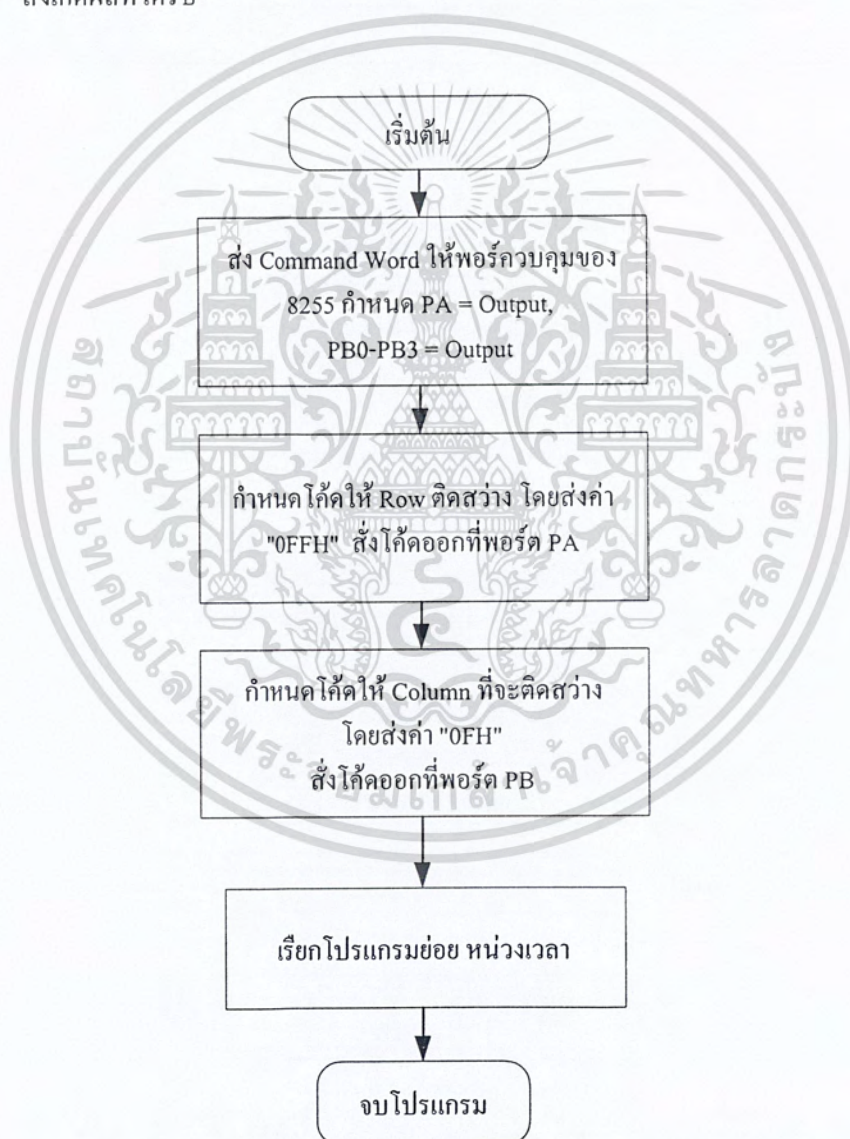


รูปที่ 4.6 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน LED Module

4.3.3 โมดูลชุดแสดงผลแบบ LED 7 ส่วน (7 Segment LED Display Module)

เป็นส่วนที่แสดงผลแบบตัวเลขและตัวอักษรบางตัวเมื่อต้องการใช้งานส่วนนี้ขั้นตอนตามโฟลว์ชาร์ต รูปที่ 4.7 และมีขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/2 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษาแอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. คาวนโหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



รูปที่ 4.7 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน 7-Segment Module

4.3.4 โมดูลแสดงผลแบบผลึกเหลว(Liquid Crystal Display Module)

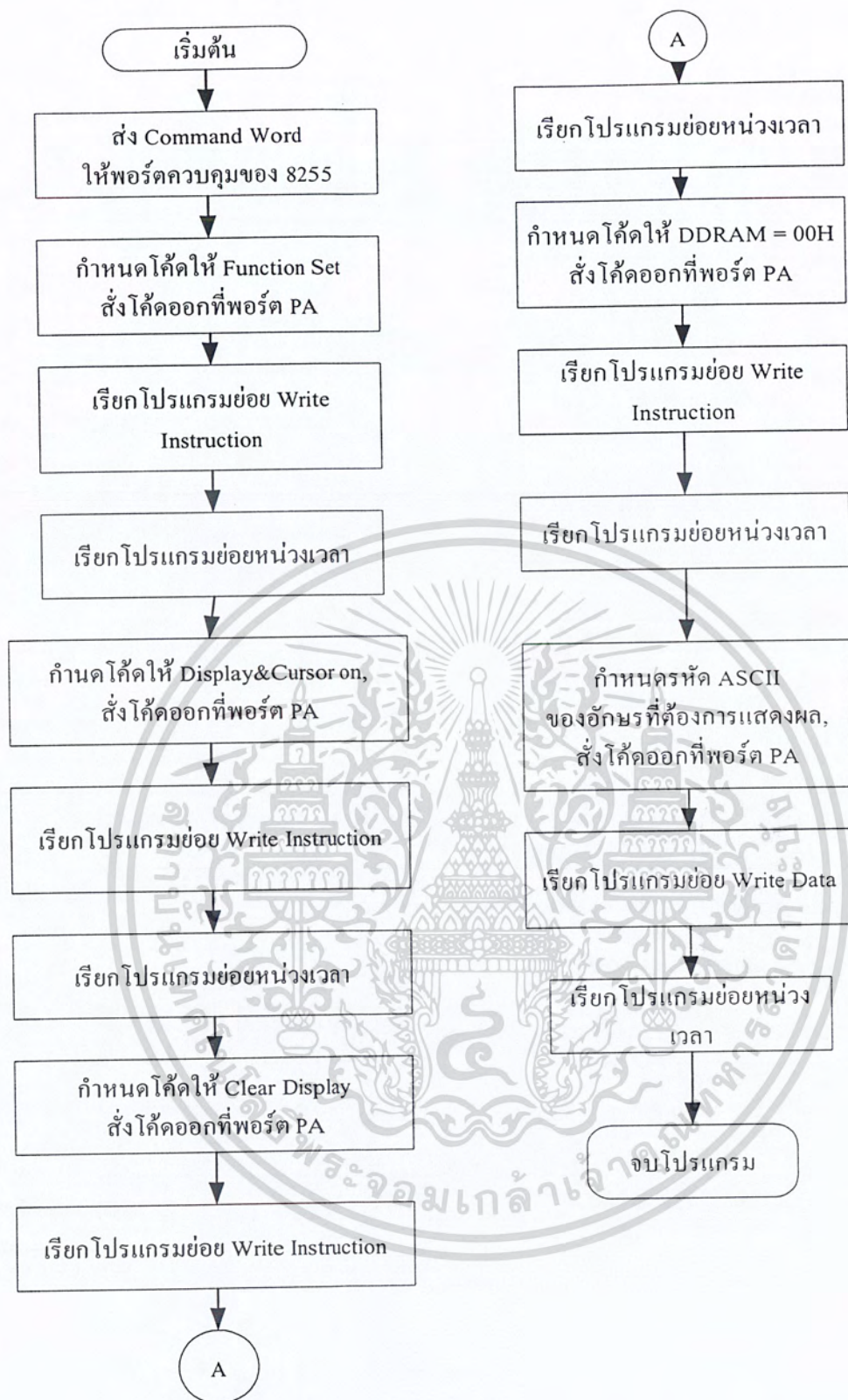
เป็นส่วนที่แสดงผลเป็นตัวอักษร ตัวเลข และรูปภาพแบบกราฟิกต่างๆ ในโมดูล LCD ประกอบหลักๆ 3 ส่วนดังนี้

ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงมีมุมมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษรหรือเลื่อนเคอร์เซอร์ เป็นต้น

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด การใช้งานส่วนนี้ขึ้นตอนตามโฟลว์ ชาร์ต รูปที่ 4.8 ตามขั้นตอนดังนี้

1. ต่อสายバスเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/3 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(Download) โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

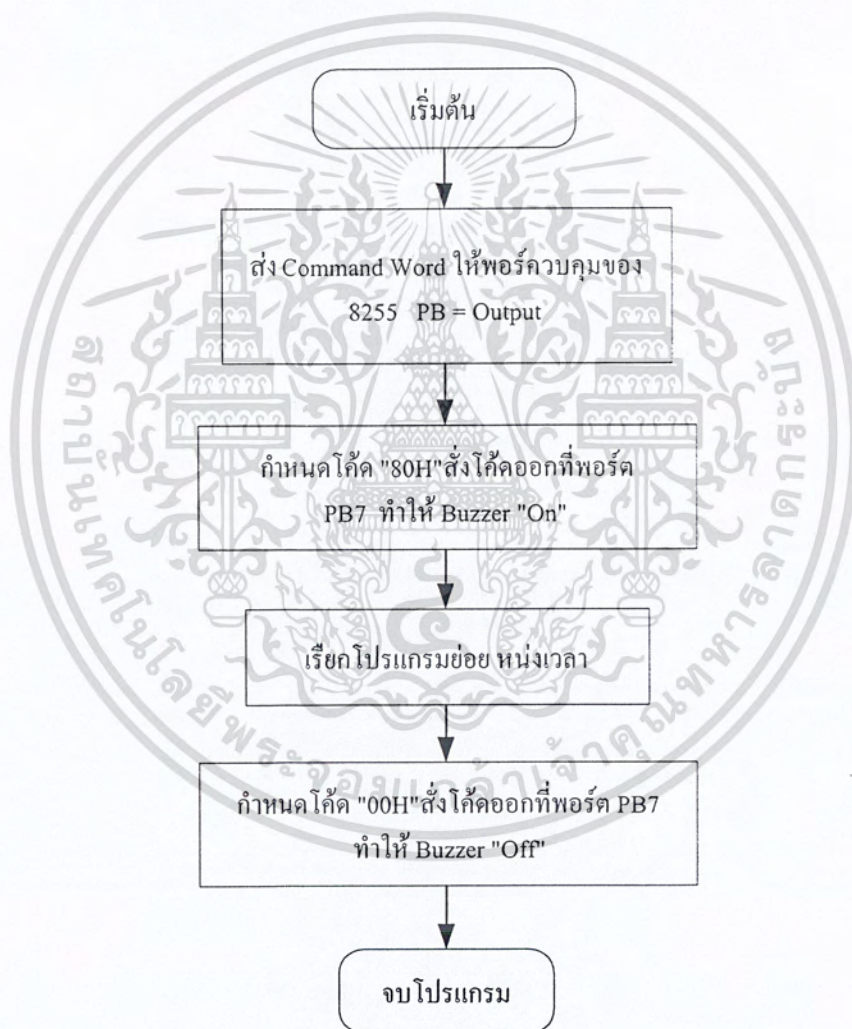


รูปที่ 4.8 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน LCD

4.3.5 โมดูลบีซเซอร์ (Buzzer Module)

เป็นส่วนแสดงผลที่มีเสียงเมื่อถูกใช้งานโดยส่งให้ข้อมูลออกที่ PB7 ตามโฟลว์ ชาร์ต รูปที่ 4.9 ถ้าต้องการให้มีการ “On” และ “Off” ในเวลาทำงานต้องส่งค่า “80H” และ “00H” ตามขั้นตอนดังนี้

1. ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
2. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
3. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

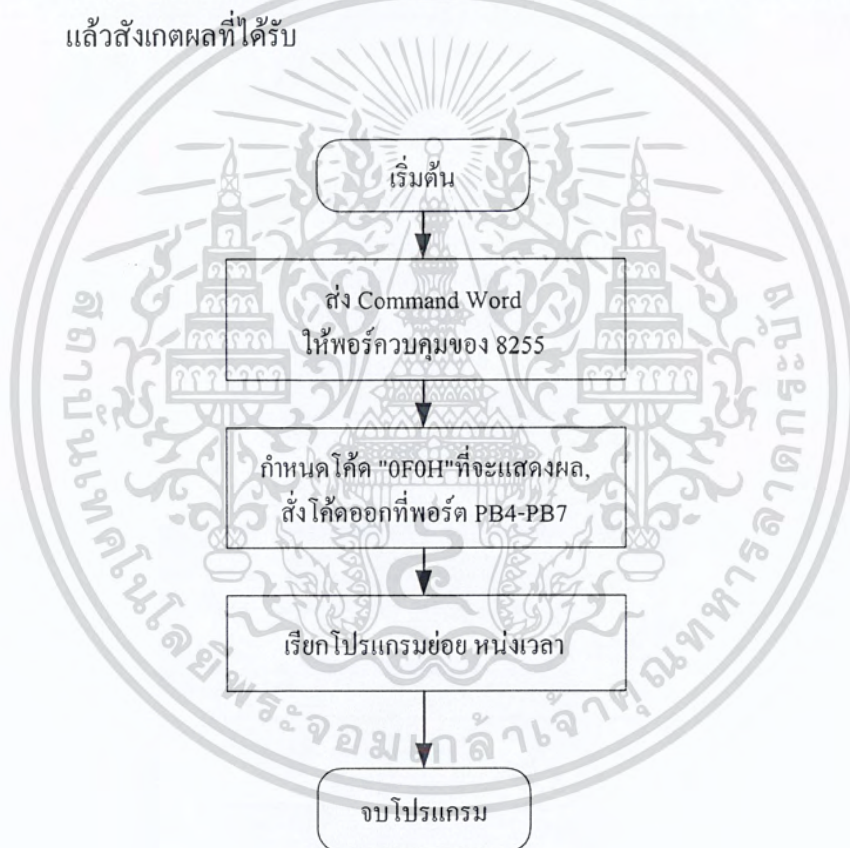


รูปที่ 4.9 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน BUZZER

4.3.6 โมดูลแม่เหล็กเหนี่ยวนำ(Magnetic Relay Module)

การใช้งาน โมดูลแม่เหล็กเหนี่ยวนำต้องส่งข้อมูลที่ต้องการออกไปที่ PB4-PB7 พอร์ตที่ต่อกับวงจร Magnetic Relay Module ถ้าต้องการให้ “On” ต้องส่งค่า “0F0H” และส่งค่า “00H” จะทำให้ “Off” ตามโฟลว์ชาร์ต รูปที่ 4.10 ตามขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดเอ็ดเพรีเม้นท์อินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/8 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download) โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงานแล้วสังเกตผลที่ได้รับ

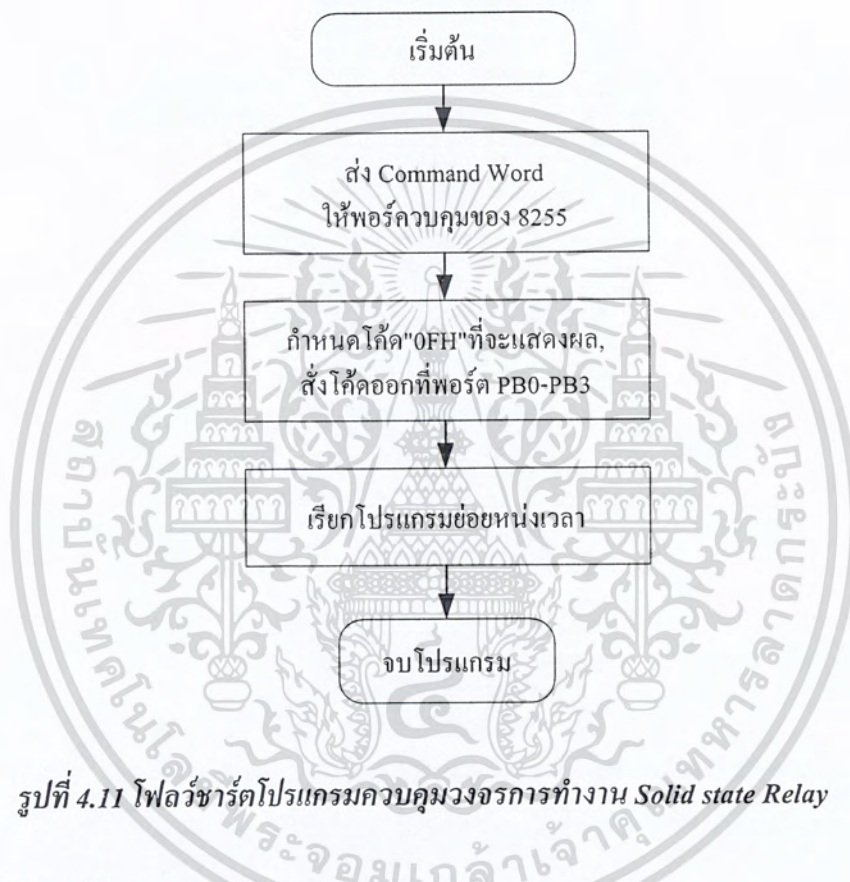


รูปที่ 4.10 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Magnetic Relay

4.3.7 โมดูลโซลิดสเตตรีเลย์ (Solid State Relay Module)

ส่วนการใช้งานโมดูลโซลิดสเตตรีเลย์ คล้ายกับการใช้โมดูลแม่เหล็กเหนี่ยวนำแต่ในการส่งข้อมูลออกต้องส่งออกไปที่ PB0-PB3 ถ้าต้องการให้ “On” ต้องส่งค่า “0FH” และส่งค่า “00H” จะทำให้ “Off” ดังแสดงตามโฟลว์ชาร์ต รูปที่ 4.11 ตามขั้นตอนดังนี้

1. ต่อสายบัสน์เข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุต เอาต์พุต
2. ปรับ สวิตช์ 1/7 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



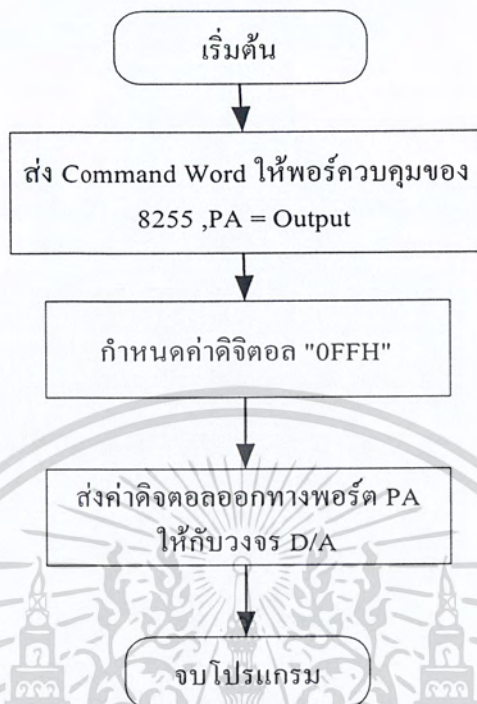
รูปที่ 4.11 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Solid state Relay

4.3.8 โมดูลการแปลงสัญญาณดิจิทัลเป็นอนาล็อก (DAC Module)

ตัวแปลงสัญญาณดิจิทัล 8 บิต ให้เป็นกระแสแล้วให้ออปแอมป์ 741 แปลงกระแสที่ได้ให้เป็นแรงดันโดยส่งค่า "0FFH" ออกที่ PA0-PA7 แล้วปรับค่า VR1 2K จะได้ค่าอยู่ในช่วง $0-2.55V_{DC}$ การนำใช้ Module นี้แสดงตามโฟลว์ ชาร์ต รูปที่ 4.12 ตามขั้นตอนดังนี้

1. ปรับ สวิตช์ 4/1 ไปที่ตำแหน่ง ON
2. ต่อสายบัสน์เข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุต เอาต์พุต
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51 Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน

5. สังเกตผลที่ได้รับโดยการใช้โวลท์มิเตอร์วัดที่จุด เอาต์พุตของสัญญาณอนาล็อกที่ Connector ที่ 1



รูปที่ 4.12 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน DAC

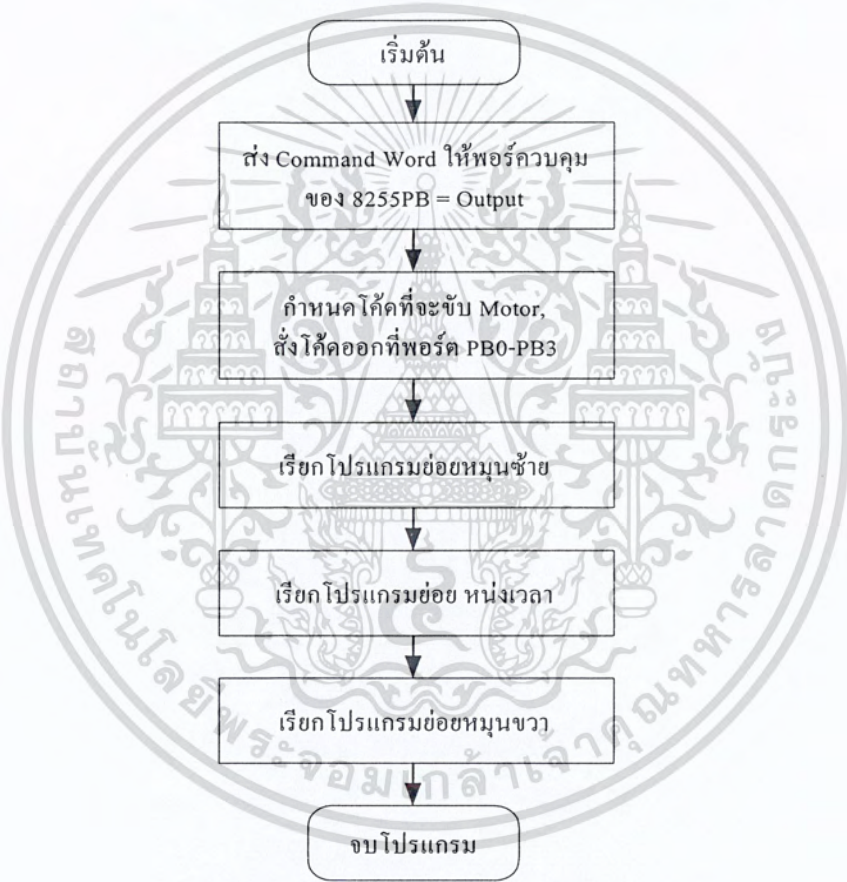
4.3.9 โมดูลสเต็ปมอเตอร์ (Stepping Motor Module)

เมื่อต้องการให้ สเต็ปเปอร์มอเตอร์หมุนให้ส่งข้อมูล “1” ไปเรียงไปตามลำดับจาก PB0-PB3 และขึ้นกับความต้องการที่จะให้สเต็ปเปอร์มอเตอร์ หมุนตามการกระตุ้นและควบคุมการหมุนอย่างไร เช่น แบบหนึ่งเฟส (full step), แบบ 2 เฟส (Two phase) และแบบครึ่งสเต็ป (Half step) ดังแสดงตามตารางที่ 4.1 โฟลว์ ชาร์ต รูปที่ 4.13 เมื่อต้องการให้สเต็ปเปอร์มอเตอร์หมุนไปซ้ายและขวาตามขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุต เอาต์พุต
2. ปรับ สวิตช์ 1/5 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51 Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

Phase Driving Method	Phase 1 Excitation				Phase 2 Excitation				Phase 1-2 Excitation							
	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8
	1				1	1			1	1	1					
A	1				1	1			1	1	1					
B		1				1	1				1	1	1			
/A			1				1	1					1	1	1	
/B				1	1			1	1						1	1

ตารางที่ 4.1 แสดงการกระตุ้นให้สเต็ปเปอร์มอเตอร์หมุน



รูปที่ 4.13 ฟลัวร์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Stepping Motor

4.3.10 โมดูลการควบคุมมอเตอร์ (Motor control Module)

การใช้งานโมดูลการควบคุมมอเตอร์ ก็คล้ายกับสเต็ปเปอร์มอเตอร์ เมื่อต้องการให้โมดูลการควบคุมมอเตอร์ ทำงานได้ต้องส่งสัญญาณควบคุมจาก Microcontroller 8051 ไปที่ PB5 ซึ่งเป็นบิตควบคุมการทำงานของมอเตอร์ ถ้าสัญญาณเป็น “1” จะทำให้มอเตอร์หมุน ถ้าสัญญาณเป็น “0” ทำให้มอเตอร์

หยุดหมุน เมื่อต้องการให้มอเตอร์เปลี่ยนทิศทางการหมุนต้องส่งสัญญาณควบคุม “1” ไปที่ PB4 ซึ่งทำหน้าที่เปลี่ยนทิศทางการหมุนของมอเตอร์ ถ้าส่งค่า “20H” มอเตอร์จะหมุนไปข้างหน้า (Forward) ถ้าส่งค่า “30H” มอเตอร์จะหมุนตรงกันข้าม (Reverse) ตามโฟลว์ชาร์ต รูปที่ 4.14 ซึ่งการทำงานตามสองทิศทาง ตามขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assmblly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

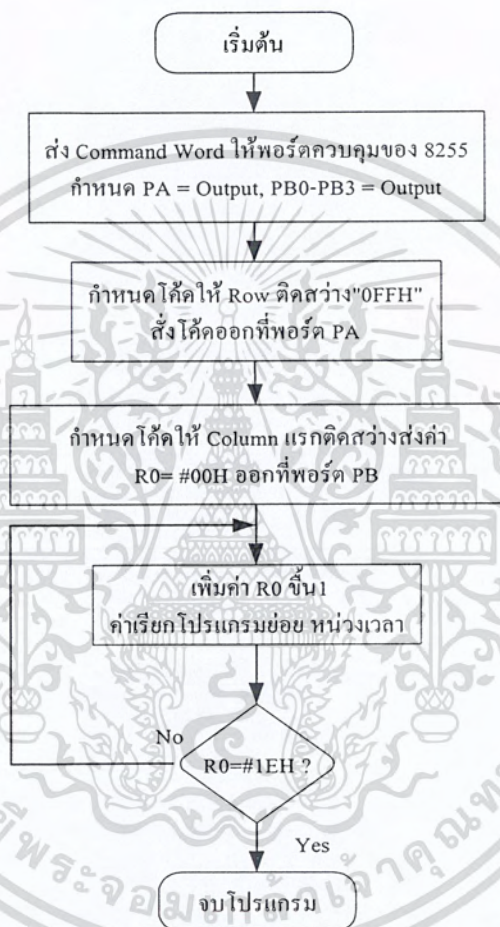


รูปที่ 4.14 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Motor control

4.3.11 ชุดแสดงผลข้อความ (Dot Matrix Display)

การทำให้หลอด LED ติดสว่างไม่สามารถที่จะสั่งให้ LED ทุกหลอดติดสว่างพร้อมๆ กันได้จะต้องอาศัยเทคนิคในการสั่งให้ติดทีละ Column ต่อเนื่องกันไปด้วยความเร็วที่มากกว่า 24 ครั้งต่อวินาทีจะทำให้ดูเหมือนว่าหลอด LED ทุกหลอดติดสว่างพร้อมกันหมด ถ้าต้องการให้ Column แรก “On” แล้วเลื่อนไปยัง Column สุดท้ายต้องส่งค่าตามโฟลว์ ชาร์ต รูปที่ 4.15 ทำตามขั้นตอนดังนี้

1. ต่อสายบัสน์เข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุต เอาต์พุต
2. ปรับ สวิตช์ 1/12 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



รูปที่ 4.15 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Dot matrix LED Display Module

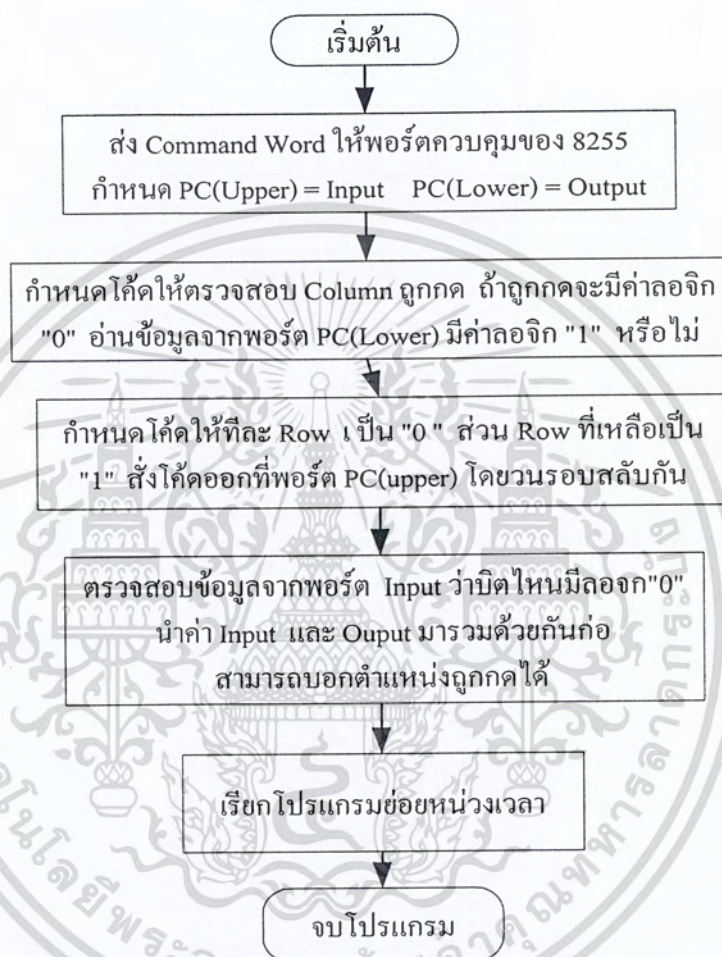
4.4 โมดูลที่เป็นอินพุต (Input Modules)

4.4.1 โมดูลคีย์บอร์ด (Key board Module)

การใช้งานคีย์บอร์ด จะทำการตรวจสอบทีละ Row และ Column ว่า Row และ Column ไหนมีการกด โดยอ่านค่าข้อมูล เข้ามาที่ PC(Lower), กำหนดโค้ดให้ทีละ Row มีค่าลอจิก "0" แล้วส่งข้อมูลออกที่ PC (upper) วนรอบสลับกันดูตามโฟลว์ชาร์ต รูปที่ 4.16 ทำตามขั้นตอนดังนี้

1. ต่อสายบัสน์เข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุต เอาต์พุต

2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. ปรับ สวิตช์ 1/10 ไปที่ตำแหน่ง ON
4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
5. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



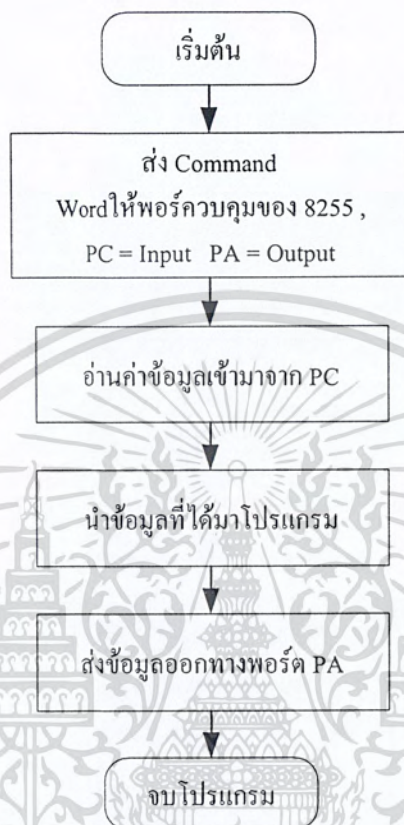
รูปที่ 4.16 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรถ่ายการทำงาน Key Board

4.4.2 โมดูลดิพสวิตช์ (Dip Switch Module)

การใช้งาน Dip-Switch สามารถเลือกใช้บิตไหนให้ทำงานและเมื่ออ่านข้อมูลเข้ามาทาง Dip-Switch แล้วจะนำค่านั้นไปโปรแกรมแล้วนำไปแสดงผลที่พอร์ต PA ดังแสดงโฟลว์ชาร์ต รูปที่ 4.17 ทำตามขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. ปรับ สวิตช์ 1/9 ไปที่ตำแหน่ง ON

4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
5. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



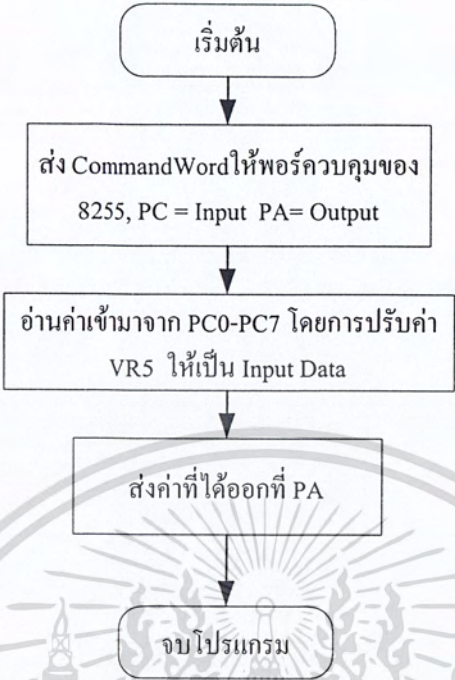
รูปที่ 4.17 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Dip-Switch

4.4.3 โมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC Module)

การใช้ ADC Module จะทำการอ่านค่าเข้ามาจาก PC ผ่าน A/D จะทำหน้าที่แปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลแล้วนำไปแสดงผลที่ PA โดยการนำใช้ดูตามโฟลว์ชาร์ต รูปที่ 4.18 ทำตามขั้นตอนนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดอินพุตเอาต์พุต
2. ใช้ Supply ต่อเข้าที่ Connector 1 ปรับไฟให้ได้ 1 โวลต์
3. ปรับ สวิตช์ 1/11 ไปที่ตำแหน่ง ON
4. ปรับ สวิตช์ 3/3 ไปที่ตำแหน่ง ON
5. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
6. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
7. ดาวน์โหลด(Download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน

8. สังเกตผลที่ได้รับที่ Port Monitor



รูปที่ 4.18 ไฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน ADC

การแปลงสัญญาณ อนาล็อกเป็นสัญญาณดิจิทัลจะได้ค่าตามตารางนี้

แรงดันอินพุต A/D	ข้อมูลที่ได้
0V	00H
0.5V	37H
1V	6BH
1.5V	A0H
2V	D5H
2.55V	FFH

ตาราง 4.2 แสดงค่าที่ได้จากการแปลงสัญญาณ อนาล็อกเป็นสัญญาณดิจิทัล

4.4.4 โมดูลชุดตรวจวัดอุณหภูมิ (Temperature Sensor)

การนำใช้ตัวตรวจจับอุณหภูมิต้องทำให้ตัว Heater ร้อนขึ้น เมื่ออุณหภูมิเพิ่มขึ้นแรงดันก็จะเพิ่มตามด้วยอัตราส่วนที่คงที่ แรงดันที่ได้จะส่งเป็นอินพุตของออปแอมป์ 741 ซึ่งต่อเป็นวงจรดิฟเฟอเรนเชียล (Differential Amplifier) ทำให้ได้เอาต์พุต V_0 ซึ่งแรงดัน V_0 นี้ต่อเข้ากับวงจรแปลงสัญญาณอนาล็อกเป็น

สัญญาณดิจิตอลเพื่อเปลี่ยนแรงดันให้เป็นสัญญาณดิจิตอลขนาด 8 บิต โดยการใช้งานดูตามโฟลว์ชาร์ตรูปที่ 4.19 ทำตามขั้นตอนดังนี้

- 1. ต่อสายบัสด้านระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
- 2. ปรับ สวิตช์ 1/11 ไปที่ตำแหน่ง ON
- 3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
- 4. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงานแล้วสังเกตผลที่ได้รับ



รูปที่ 4.19 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Temperature Sensor

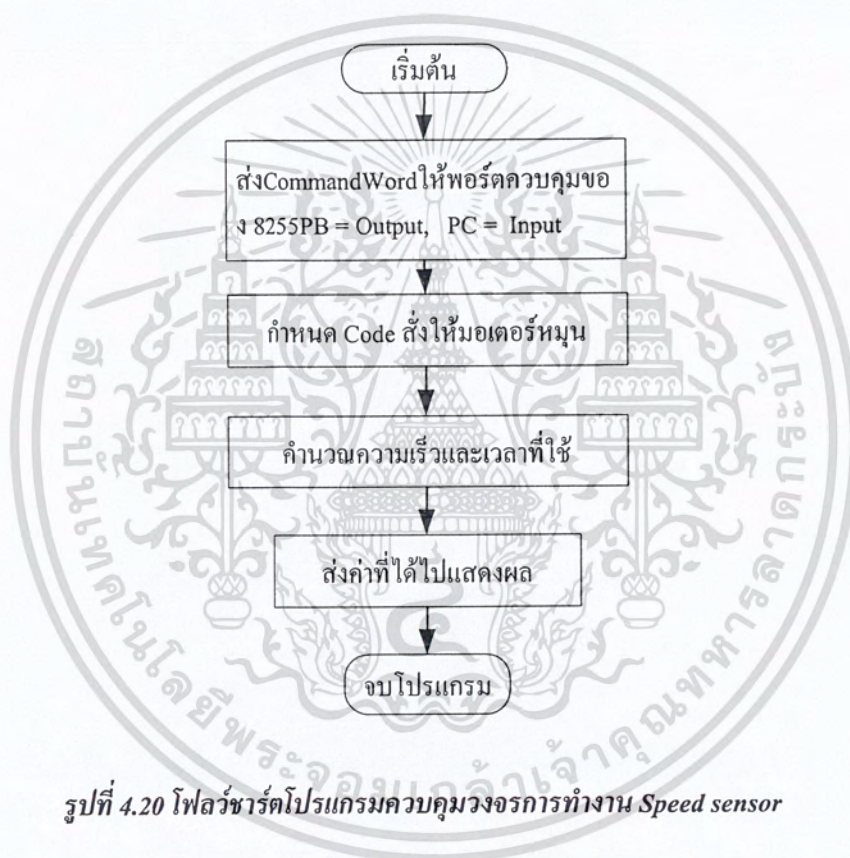
ค่าที่อ่านจาก ออปแอมป์ 741	อุณหภูมิที่ได้
2.73V	25°C
3.73V	100°C

ตาราง 4.3 แสดงค่าที่ได้จากชุดตรวจวัดอุณหภูมิ

4.4.5 โมดูลวงจรตรวจจับความเร็ว (Speed Sensor Module)

การใช้งานต้องทำให้แสงอินฟราเรดถูกบังโดยใบพัดของมอเตอร์วงจรตรวจจับความเร็วจะส่งสัญญาณออกมาเป็น “0” เมื่อใบพัดเลื่อนผ่านไปโฟโต้ทรานซิสเตอร์สามารถรับแสงจากหลอดอินฟราเรดได้ วงจรจะส่งเอาต์พุตออกมาเป็น “1” โดยการนำใช้ดูตามโฟลว์ ชาร์ต รูปที่ 4.20 ทำตามขั้นตอนนี้

1. ต่อสายบัสดักระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/10 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษาแอสเซมบลี 51 (Assumbly 51) Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

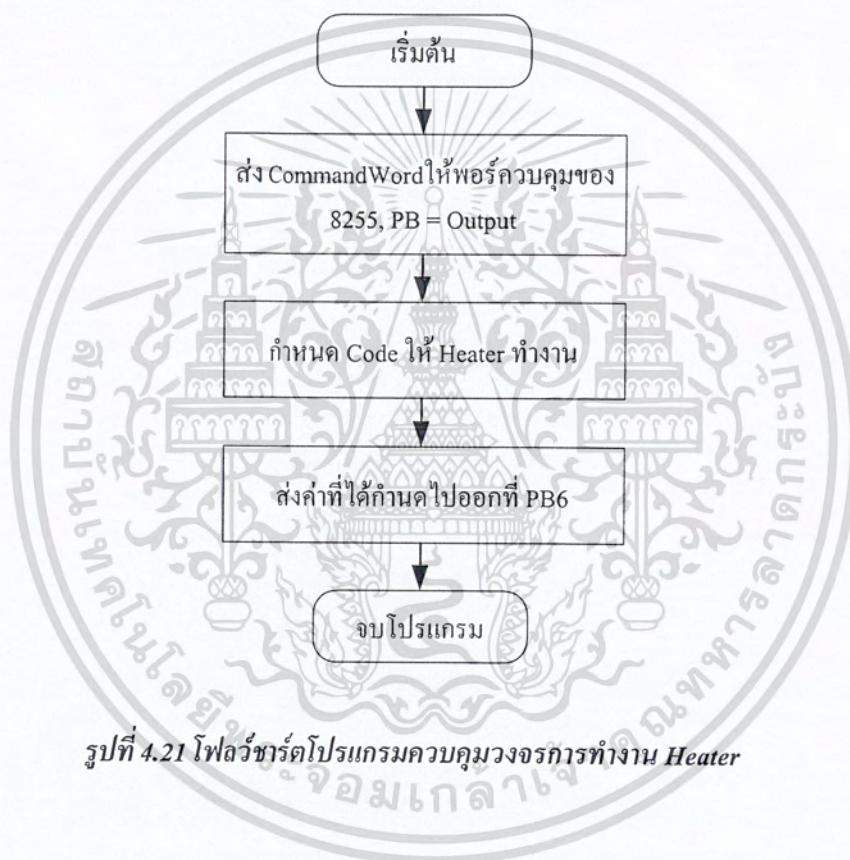


รูปที่ 4.20 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Speed sensor

4.4.6 โมดูลชุดควบคุมความร้อน (Heater Control Module)

เป็นตัวด้านทานเมื่อทำให้กระแสไหลผ่านตัวด้านทานนี้จะทำให้เกิดความร้อนขึ้นในขั้นตอนการนำใช้แสดงตามโฟลว์ ชาร์ต รูปที่ 4.21 ทำตามขั้นตอนดังนี้

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assumbly 51 Compile โดย SXA51 ASSUMBLER ให้เป็น File.Hex
4. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ



รูปที่ 4.21 โฟลว์ชาร์ตโปรแกรมควบคุมวงจรการทำงาน Heater

บทที่ 5

การทดสอบการทำงานของบอร์ดไมโครคอนโทรลเลอร์

สำหรับการทดลอง

การตรวจสอบและทดสอบการทำงานทั้งหมดทางด้านฮาร์ดแวร์และซอฟต์แวร์ ของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง ที่ได้ออกแบบและสร้างไว้แล้วนั้น ได้มีการทดสอบการทำงานในแต่ละโมดูลด้วยวิธีการต่างๆ เช่นสั่งให้โมดูลนั้นทำงานแล้วดูผลที่ได้รับถูกต้องหรือไม่ ซึ่งรายละเอียดของการทดสอบการทำงานในแต่ละโมดูลนั้นจะได้กล่าวต่อไป นอกจากการทดสอบแล้วยังได้จัดทำใบการทดลองขึ้นมาเพื่อใช้ในการทดสอบการทำงานทั้งหมดของบอร์ด

การทดสอบบอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตพอร์ตเอาต์พุตพอร์ตของบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง

5.1 บอร์ดไมโครคอนโทรลเลอร์ 8051

5.1.1 ทดสอบโมดูล CPU (Test CPU Module)

ขั้นตอนการทดลอง

- เขียนโปรแกรมส่งค่า #0ffh ออกที่พอร์ต 1 ของหน่วยประมวลผลกลาง และสังเกตผลโดยใช้ลอจิกโปรบวัดสัญญาณที่ขา 1-8 ของ (8051)
- ทดลองส่งค่าอื่นๆออกมาที่พอร์ต 1
- ส่วนพอร์ตอื่นๆก็ทำเช่นเดียวกัน

ผลการทดลอง

- สัญญาณที่พอร์ต 1 เป็นไปตามค่าที่ส่งออกมา
- สำหรับการทดลองพอร์ตอื่นก็ได้รับผลตามที่เราส่งค่าออกมา

สรุปผลการทดลอง

- หน่วยประมวลผลกลาง CPU บอร์ดสามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.1.2 ทดสอบโมดูลหน่วยความจำ (Test Memory Module)

ขั้นตอนการทดลอง

- เขียนโปรแกรมส่งค่า #0a0h ส่งไปเก็บที่หน่วยความจำตำแหน่งใดตำแหน่งหนึ่ง
- เขียนโปรแกรมไปอ่านค่ามาจากหน่วยความจำตำแหน่งที่เราเอาข้อมูลไปเก็บแล้วส่งออกที่พอร์ต 1 ของหน่วยประมวลผลกลาง CPU และสังเกตผลโดยใช้ลอจิกโปรบวัดสัญญาณที่ขา 1-8 ของ (8051)

- ทดลองส่งค่าอื่นๆไปเก็บที่หน่วยความจำแล้วเขียนโปรแกรมไปอ่านค่าที่หน่วยความจำแล้วส่งออกมาที่พอร์ต 1

ผลการทดลอง

- สัญญาณที่พอร์ต 1 เป็นไปตามค่าที่เราเอาไปเก็บที่หน่วยความจำ

สรุปผลการทดลอง

- โมดูลหน่วยความจำสามารถทำงานได้อย่างถูกต้อง

5.1.3 ทดสอบโมดูลอินเทอร์รัปต์ (Interrupt Module Test)

ขั้นตอนการทดลอง

- ใช้ลอจิกโพรบจับที่ขา Interrupt 0 ของ IC 8051 ขาที่ 12
- ทดลองกดสวิตช์ Interrupt 0 แล้วดูผลลัพธ์ที่โพรบ
- สำหรับการทดสอบ Interrupt 1ก็ทำเหมือนกันแต่ใช้โพรบจับที่ขา 13

ผลการทดลอง

- สัญญาณที่โพรบจะเปลี่ยนสถานะจาก High เป็น Low

สรุปผลการทดลอง

- โมดูล Interrupt ทำงานได้อย่างถูกต้อง

5.1.4 ทดสอบโมดูลวงจรรนับ/จับเวลา (Timer/Counter Test)

ขั้นตอนการทดลอง

- ให้โมดูล Counter ทำงาน
- ใช้ลอจิกโพรบจับที่ขา Timer/Counter 0 ของ IC 8051 ขาที่ 14

ผลการทดลอง

- สัญญาณที่โพรบจะกระพริบจาก High เป็น Low อย่างต่อเนื่อง

สรุปผลการทดลอง

- โมดูล Timer/Counter ทำงานได้อย่างถูกต้อง

5.1.5 ทดสอบพอร์ตใช้งานทั่วไป 8255

ขั้นตอนการทดลอง

- ทดลองเขียนโปรแกรมส่งค่า #80h ไปยังพอร์ต D ของ 8255 ที่ตำแหน่ง 8003h เพื่อให้ทุกๆ พอร์ตเป็นเอาต์พุตพอร์ต
- ส่งค่า #0a0h ไปยังพอร์ต A, ส่งค่า #0fh ไปยังพอร์ต B, ส่งค่า #80h ไปยังพอร์ต C ของ 8255 ที่ตำแหน่ง 8000h, 8001, 8002 ตามลำดับ
- ใช้ลอจิกโปรบวัดสัญญาณที่พอร์ต A, B, C

ผลการทดลอง

- พอร์ต A สามารถส่งค่าออกมาได้ตามโปรแกรมที่รัน
- พอร์ต B สามารถส่งค่าออกมาได้ตามโปรแกรมที่รัน
- พอร์ต C สามารถส่งค่าออกมาได้ตามโปรแกรมที่รัน

สรุปผลการทดลอง

- 8255 บนบอร์ดสามารถทำงานได้อย่างถูกต้อง

5.1.6 ทดสอบโมดูลพอร์ตอนุกรม (Serial Port RS232 Test)

ขั้นตอนการทดลอง

- ต่อสาย Serial เข้าระหว่าง PC และบอร์ดไมโครคอนโทรลเลอร์ 8051
- เขียนโปรแกรมติดต่อกับ Serial port ของ PC โดยรับค่ามาจาก PC แล้วเพิ่มค่าขึ้นหนึ่งค่าแล้วส่งกลับมาที่ PC
- ที่ PC เข้าไปที่ Hyper terminal ส่งค่า A ออกไป

ผลการทดลอง

- อยู่ที่ PC จะเห็น B ปรากฏขึ้น

สรุปผลการทดลอง

- โมดูล Serial port สามารถทำงานได้อย่างถูกต้อง

5.1.7 ทดสอบการ Interface ของบอร์ดไมโครคอนโทรลเลอร์ 8051

ขั้นตอนการทดลอง

- ต่อสาย Bus เข้าระหว่าง บอร์ด MI-1 ของ Ana และบอร์ดไมโครคอนโทรลเลอร์ 8051
- ปรับสวิตช์ของโมดูลที่จะทำการทดลองในบอร์ด Ana ให้ถูกต้อง
- เขียนโปรแกรมสั่งให้โมดูลทำการทดลองนั้นทำงาน

ผลการทดลอง

- ที่บอร์ดของ Ana ในโมดูลทำการทดลองสามารถทำงานได้อย่างถูกต้อง

สรุปผลการทดลอง

- การ Interface ของบอร์ดไมโครคอนโทรลเลอร์สามารถทำงานได้อย่างถูกต้อง

5.2 การทดสอบบอร์ดทดลองอินพุตเอาต์พุต (Experiment input/output board)

การทดสอบบอร์ดทดลองอินพุตโมดูลและเอาต์พุตโมดูล ของบอร์ดไมโครคอนโทรลเลอร์ สำหรับการทดลอง ในการทดสอบการทำงานของบอร์ดนี้อุปกรณ์ที่ใช้ประกอบมี

- บอร์ดไมโครคอนโทรลเลอร์ 8051 ที่เราได้ออกแบบและสร้างขึ้นมา
- สาย Bus ที่ได้จัดทำขึ้นมา และสายสำหรับดาวน์โหลดโปรแกรมลงในบอร์ด
- บอร์ดทดลองอินพุตเอาต์พุต

5.2.1 ทดสอบโมดูลพอร์ตมอนิเตอร์ (Port Monitor)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/1 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0ffh ออกที่พอร์ต A ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์และสังเกตผลที่ Port monitor จะเห็น LED ติดและดับตามข้อมูลที่ส่งออกมา
- ทดลองส่งค่าอื่นๆออกมาที่พอร์ต A ของ 8255

ผลการทดลอง

- ที่โมดูล Port monitor จะเห็น LED ติดและดับตามค่าข้อมูลที่เราส่งออกมาที่พอร์ต A ของ 8255

สรุปผลการทดลอง

- โมดูล Port monitor สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.2 ทดสอบโมดูลชุดแสดงผลแบบ LED 7 ส่วน (7 Segment Led Display Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/2 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0ffh ออกที่พอร์ต A และ ส่งค่า #0fh ไปที่พอร์ต B ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์และสังเกตผลที่ 7 Segment
- ทดลองเขียนโปรแกรมส่งให้ 7 Segment ติดทีละ Segment

ผลการทดลอง

- ที่โมดูล 7 Segment จะเห็น 7 Segment เป็นเลข 8 ทุกๆตัวและติดเป็น Segment ตามค่าที่เราส่งออกมา

สรุปผลการทดลอง

- โมดูล 7 Segment สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.3 ทดสอบโมดูลสเต็ปปีงมอเตอร์ (Stepping Motor Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/5 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0fh ออกที่พอร์ต B ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์และสังเกตผลที่โมดูล Stepping motor
- ทดลองส่งค่า #05h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ที่โมดูล Stepping motor จะเห็น Stepping motor หมุนเป็นแบบ Full duplex และ Haft duplex ตามข้อมูลที่เราส่งออกมา

สรุปผลการทดลอง

- โมดูล Stepping motor สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.4 ทดสอบชุดควบคุมความร้อน (Heater Control Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/6 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #40h ออกที่พอร์ต B ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์และสังเกตผลที่ Heater โมดูลโดยใช้มือสัมผัสที่ Heater
- ทดลองส่ง # 00h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ที่โมดูล Heater จะร้อนขึ้น และเย็นลงตามค่าข้อมูลที่เราส่งออกมา

สรุปผลการทดลอง

- โมดูล Heater สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.5 ทดสอบโมดูลการควบคุมมอเตอร์ (Motor Control Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/6 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #20h ออกที่พอร์ต B และสังเกตผลที่ DC motor 1 โมดูล
- เขียนโปรแกรมส่งค่า #30h ออกที่พอร์ต B และสังเกตผลที่ DC motor 1 โมดูล
- ทดลองส่งค่า #00h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ที่โมดูล Motor control จะเห็น Motor หมุนซ้าย, หมุนขวา และ หยุดหมุนตามข้อมูลที่เราส่งออกมา

สรุปผลการทดลอง

- โมดูล Motor control สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.6 ทดสอบโมดูลบัสเซอร์ (Buzzer Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/6 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #80h ออกที่พอร์ต B และสังเกตผลที่โมดูล Buzzer
- ทดลองส่งค่า #00h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ดังแล้ว,หยุดตามข้อมูลที่เราส่งออกมา

สรุปผลการทดลอง

- โมดูล Buzzer สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.7 ทดสอบโมดูลแสดงผลแบบผลึกเหลว(LCD)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/3 ไปที่ตำแหน่ง ON
- ทดลองเขียนโปรแกรมส่งคำสั่งต่อไปนี้ไปยังอุปกรณ์แสดงผลแบบผลึกเหลว LCD ที่พอร์ต A เพื่อกำหนดโหมดการทำงานให้ LCD

- #00111000b ;function set
- #00000110b ;entry mode set
- #00001111b ;display on,cursor on,blink on

- #00000001b ;clear screen
- #00000010b ;cursor goto home

- ให้โปรแกรมส่งรหัสแอสกีของ 'a' ไปที่พอร์ต A เพื่อส่งข้อมูลที่จะแสดงบนจอไปที่ LCD
- ให้โปรแกรมลองส่งตัวอักษรตัวอื่นตามไป

ผลการทดลอง

- บนจอ LCD สามารถแสดงตัวอักษรตามรหัสแอสกีที่ส่งไปให้ได้

สรุปผลการทดลอง

- LCD บนบอร์ดสามารถทำงานได้อย่างถูกต้อง

5.2.8 ทดสอบโมดูลแสดงผลข้อความ (Dot Matrix Display)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/12 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0ffh ออกที่พอร์ต A และ ส่งค่า 00h ออกไปที่พอร์ต B ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์และสังเกตผลที่ Dot matrix display
- ทดลองส่งค่าอื่นๆออกมาที่พอร์ต A ของ 8255

ผลการทดลอง

- ที่โมดูล Dot matrix จะเห็น Dot matrix ติดเป็นแนวตั้งตามค่าข้อมูลที่เราส่งออกมาที่พอร์ต A ของ 8255

สรุปผลการทดลอง

- โมดูล Dot matrix display สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.9 ทดสอบโมดูลโซลิดสเตทรีเลย์ (Solid State Relay Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/7 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0fh ออกที่พอร์ต B และสังเกตผลที่ Solid state relay โมดูล
- ทดลองส่งค่า #00h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ที่โมดูล Solid state relay จะเห็น รีเลย์ เปิดทำงาน, และ ปิดการทำงานตามข้อมูลที่เราส่งออกมาที่พอร์ต B

สรุปผลการทดลอง

- โมดูล Solid state relay สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.10 ทดสอบโมดูลแม่เหล็กนติกรีเลย์ (Magnetic Relay Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/8 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #0f0h ออกที่พอร์ต B และสังเกตผลที่ Magnetic relay โมดูล
- ทดลองส่งค่า #00h ออกมาที่พอร์ต B ของ 8255

ผลการทดลอง

- ที่โมดูล Magnetic relay จะเห็น Relay เปิดทำงาน, และ ปิดการทำงานตามข้อมูลที่เราส่งออกมาที่พอร์ต B

สรุปผลการทดลอง

- โมดูล Magnetic relay สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.11 ทดสอบโมดูลดิพสวิตช์ (Dip Switch Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/9 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมอ่านค่าเข้ามาจากพอร์ต C ของ 8255 แล้วส่งไปแสดงผลที่ Port monitor แล้วกดสวิตช์อินพุตบิตที่ 1
- ทดลองกดสวิตช์อินพุตแต่ละบิต

ผลการทดลอง

- ที่โมดูล Port monitor จะเห็น LED ตามสวิตช์อินพุตที่เรากด

สรุปผลการทดลอง

- โมดูล สวิตช์อินพุต สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.12 ทดสอบโมดูล คีย์บอร์ด (Key board Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/10 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมอ่านค่าเข้ามาจากคีย์บอร์ดที่ต่อกับพอร์ต C ของ 8255 แล้วส่งไปแสดงผลที่ Port monitor โดยที่กดคีย์หมายเลข 1 ให้แสดงผลเลข 1 ในเลขฐาน 2 (Binary) สังเกตผลที่ Port monitor
- ทดลองกดคีย์หมายเลขตัวอื่นๆ

ผลการทดลอง

- ที่โมดูล Port monitor จะเห็น LED ติดเป็นเลขฐาน 2 ในเมื่อกดคีย์หมายเลขใดหนึ่ง

สรุปผลการทดลอง

- โมดูล คีย์บอร์ด สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.13 ทดสอบโมดูลการแปลงสัญญาณดิจิตอลเป็นอนาล็อก (DAC Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/4 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งข้อมูล #0ffh ออกไปที่พอร์ต A สังเกตผลได้รับที่คอนเนกเตอร์ J0 โดยใช้ Meter วัดค่ามาวัด
- ทดลองส่งค่าข้อมูลอื่นออกไปที่พอร์ต A

ผลการทดลอง

- โมดูล DAC ที่คอนเนกเตอร์ J0 เมื่อวัดแล้วจะได้ค่าตามที่เรากำหนด

สรุปผลการทดลอง

- โมดูล DAC สามารถทำงานตามโปรแกรมของผู้ใช้ได้ถูกต้อง

5.2.14 ทดสอบโมดูลแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 3/3 ไปที่ตำแหน่ง ON
- ปรับสวิตช์ 1/11 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมอ่านค่าเข้ามาจากพอร์ต C ของ 8255 แล้วส่งไปแสดงผลที่ Port monitor แล้วหมุน Pot แล้วสังเกตผลที่ Port monitor
- หมุน POT ที่เป็นตัวต้านทานปรับค่าได้

ผลการทดลอง

- ที่โมดูล Port monitor จะเห็น LED มีค่าเพิ่มขึ้นลดลงเป็นเลขฐาน 2 ตามการหมุน POT ของเรา

สรุปผลการทดลอง

- โมดูล ADC สามารถทำงานตามโปรแกรมของผู้ใช้ได้ถูกต้อง

5.2.15 ทดสอบโมดูลวงจรตรวจจับความเร็ว (Speed Sensor Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/6 ไปที่ตำแหน่ง ON
- ปรับสวิตช์ 1/10 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมสั่งให้ Motor หมุนแล้วอ่านข้อมูลเข้ามาจากพอร์ต C บิตที่ 7 ของ 8255 แล้วเพิ่มค่าขึ้นทุกๆครั้งที่มีการเปลี่ยนแปลงสถานะ แล้วส่งไปแสดงผลที่ Port monitor

ผลการทดลอง

- ที่โมดูล Port monitor จะเห็น LED มีค่าเพิ่มขึ้น ลดลงเป็นเลขฐาน 2 ตามการหมุนของ DC Motor

สรุปผลการทดลอง

- โมดูล Speed sensor สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.16 ทดสอบโมดูลชุดตรวจวัดอุณหภูมิ (Temperature Sensor Module)

ขั้นตอนการทดลอง

- ปรับสวิตช์ 1/6, 1/11, 3/2 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมส่งค่า #40h ออกที่พอร์ต B ของ 8255 โหลดโปรแกรมลงในบอร์ดไมโครคอนโทรลเลอร์ 8051 อ่านค่าเข้ามาจากพอร์ต PC แล้วส่งออกไปที่ Port monitor

ผลการทดลอง

- ที่โมดูล Port monitor LED จะติดมีค่าเพิ่มขึ้น

สรุปผลการทดลอง

- โมดูล Temperature สามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

5.2.17 การทดสอบการ Interface ของบอร์ดทดลองอินพุตเอาต์พุต

ขั้นตอนการทดลอง

- ต่อ Single board ไมโครคอนโทรลเลอร์ 8031 ของ Ana เข้ากับบอร์ดทดลองอินพุตเอาต์พุต
- ปรับสวิตช์ให้โมดูล Port monitor ทำงาน หรือโมดูลอื่นๆก็ได้
- เขียนโปรแกรมอยู่ที่ Single Board ส่งให้โมดูลที่ปรับสวิตช์ให้ทำงานนั้นทำงาน
- ทดลองต่อกับ Single Board Z-80 ของ Ana แล้วทำการทดลองเหมือนกันกับข้างบน

ผลการทดลอง

- โมดูลที่ปรับสวิตช์ให้ทำงานนั้นทำงานตามโปรแกรมที่ส่งมา

สรุปผลการทดลอง

- การ Interface ของบอร์ดทดลองสามารถทำงานได้อย่างถูกต้อง

บทที่ 6

ใบงานการทดลอง (LAB)

6.1 การทดลองที่ 1 การโปรแกรม 8255 และการแสดงผลข้อมูลขนาด 8 บิต

6.1.1 การโปรแกรม 8255 (Programmable peripheral interface)

■ วัตถุประสงค์

1. เพื่อศึกษาการเขียน Command Word 8255
2. เพื่อที่จะกำหนด Command word ให้ถูกต้องกับการใช้งานของบอร์ดชุดทดลองพื้นฐานไมโครคอนโทรลเลอร์

■ ทฤษฎี

การติดต่อสั่งงานของ ไมโครคอนโทรลเลอร์ 8051 กับบอร์ดทดลองอินพุตเอาต์พุตของบอร์ดไมโครคอนโทรลเลอร์ สำหรับการทดลองนั้น จะส่งผ่านมาทาง 8255 ซึ่งทำหน้าที่เป็นพอร์ตอินพุตเอาต์พุต ซึ่งภายใน 8255 จะประกอบไปด้วยพอร์ตต่างๆ คือ พอร์ต PA, PB, PC ซึ่งเป็น พอร์ตขนาด 8 บิต สำหรับติดต่อกับอุปกรณ์ภายนอก และอีกพอร์ตหนึ่งเป็นพอร์ตควบคุม ซึ่งคำสั่งจากไมโครคอนโทรลเลอร์ เพื่อควบคุมให้ 8255 กำหนดรูปแบบการทำงานของพอร์ตต่างๆ

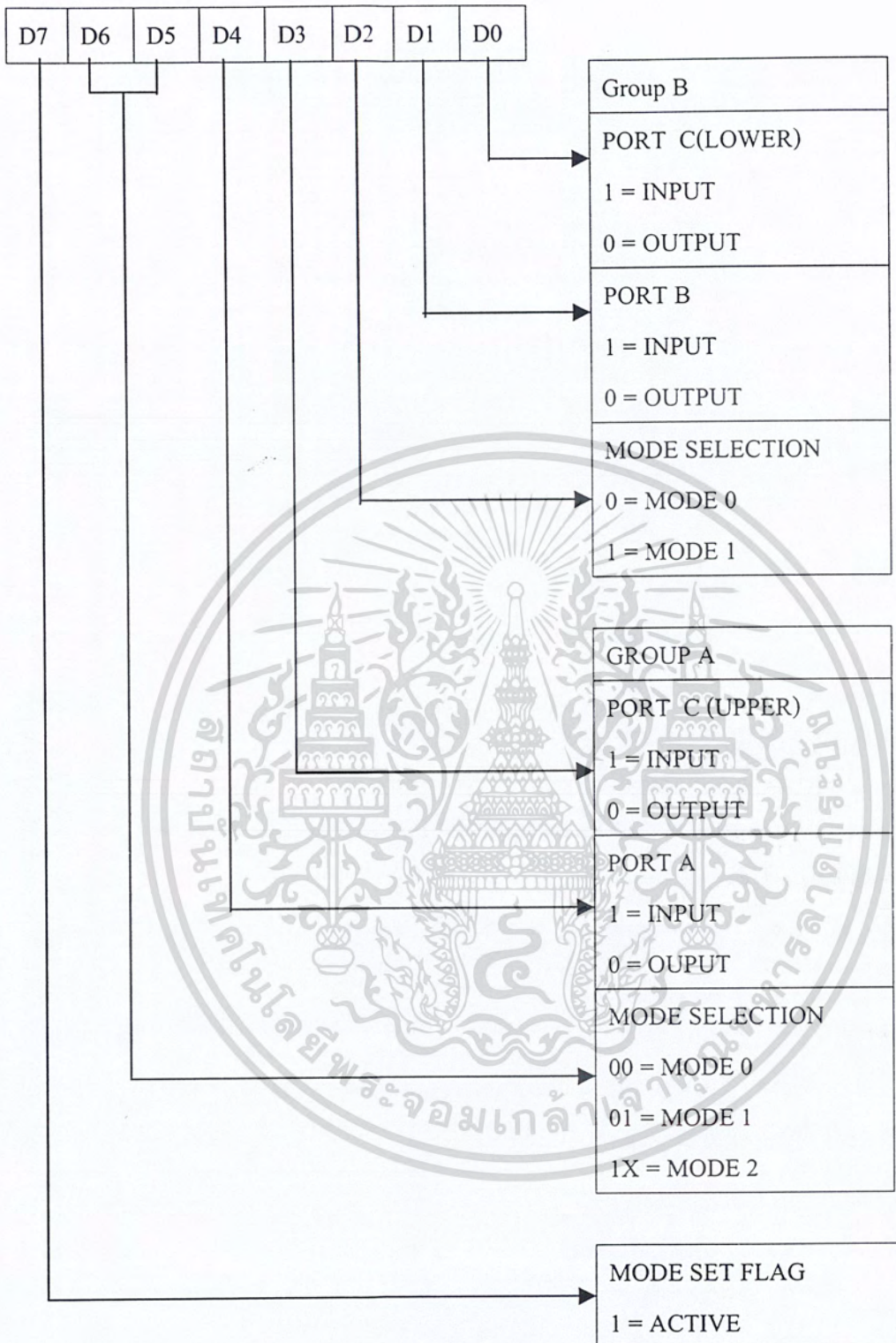
ดังนั้นการใช้งาน 8255 จึงจำเป็นจะต้องเขียนโปรแกรมคำสั่งควบคุม (Command Word) ส่งมาที่พอร์ตควบคุมทุกครั้งก่อนที่จะมีการใช้งาน 8255 ซึ่งคำสั่งนี้จะคงอยู่ตลอดไป จนกว่าจะมีการรีเซ็ต (Reset) เมื่อทำการส่งคำสั่งใช้งานให้กับ 8255 แล้ว ก็สามารถที่จะนำเอา พอร์ต PA, PB และ PC มาใช้เป็นอินพุตเอาต์พุตตามที่ต้องการได้

สำหรับบอร์ดทดลองไมโครคอนโทรลเลอร์ สำหรับการทดลองจะกำหนดการทำงานของพอร์ต PA ที่ตำแหน่ง 8000H, การทำงานของพอร์ต PB ที่ตำแหน่ง 8001H, การทำงานของพอร์ต PC ที่ตำแหน่ง 8002H, และการทำงานของพอร์ตคอนโทรล ที่ตำแหน่ง 8003H สำหรับการกำหนดการทำงานของพอร์ตใน 8255 จะแบ่งการทำงานออกเป็น 3 โหมดคือ

1. โหมด 0 เป็นอินพุต-เอาต์พุตเบื้องต้น (Basic Input/Output)
2. โหมด 1 เป็นอินพุต-เอาต์พุตแบบสโตรบ (Strobe Input/Output)
3. โหมด 2 เป็นบัสสองทิศทาง (Bidirection Bus)

โดยปกติเราจะใช้งานกันที่โหมด 0

CONTROL WORD



รูปที่ 6.1 Control word 8255

เพราะฉะนั้นทุกครั้งที่มีการเขียนโปรแกรมในการทำการทดลองส่วนหัวของโปรแกรมจะต้องกำหนด Command Word หรือเรียกว่า การ “Initial” ซึ่งการ Initial 8255 จะทำเพียงครั้งแรกของการทำงานของโปรแกรมเท่านั้น ถ้าโปรแกรมมีการทำงานซ้ำก็สามารถที่จะข้ามการ Initial ในตอนแรกไปได้

โปรแกรมที่ 1.1 การ Initial 8255

0000	1	ORG 8000H
0000 908003	2	MOV DPTR,#8003h ;init 5255
0003 7480	3	MOV A,#80H
0005 F0	4	MOVBX @DPTR,A

โปรแกรมที่ 1.1 เป็นการ Initial 8255 พอร์ต PA, พอร์ต PB, และพอร์ต PC เป็นเอาต์พุตพอร์ต

6.1.2 การแสดงผลข้อมูลขนาด 8 บิต

■ วัตถุประสงค์

1. เพื่อศึกษาการส่งข้อมูลขนาด 8 บิต ผ่านทางพอร์ต PA ของ 8255

■ ทฤษฎี

LED (Port monitor) ที่มีในบอร์ดไมโครคอนโทรลเลอร์ สำหรับการทดลองนั้นจะต่ออยู่กับพอร์ต PA ของ 8255 โดยที่จะมี บัฟเฟอร์ 74LS240 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PA ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS240 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขาเลือกของ 74LS240 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัวเลือกว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล LED ขาเลือกของ 74LS240 จะต่ออยู่กับ บิตที่ 1 ของ Dip switch ตัวที่ 1.

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดอินพุต/เอาต์พุต
2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) เช่น โปรแกรมที่ 1.2 แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 1.2 (สั่งให้ LED ดับ 4 หลอดและ ติด 4 หลอด)

```

0000          1      ORG  0000H
0000 908003    2      MOV  DPTR,#8003H  ;INITAIL 8255
0003 7480      3      MOV  A,#80H
0005 F0        4      MOVX  @DPTR,A
0006 908000    5      MOV  DPTR,#8000H  ;PORT PA 8255
0009 740F      6      MOV  A,#0FH      ;DATA FOR DISPLAY
000B F0        7      MOVX  @DPTR,A    ;SENT DATA TO PORT A
000C 80FE      8      JMP   S          ;STOP PROGRAM
0000 =         9      END

```

5. จากโปรแกรมที่ 2 ทดลองเปลี่ยนรหัสที่ตำแหน่ง Address 0009 จากค่า 0fh เป็น 0FFH, 0AAH, 0FEH, 01H แล้วบันทึกผลที่ได้รับใน table

รหัส	แสดงผล LED บิตที่							
	7	6	5	4	3	2	1	0
0FFH	○	○	○	○	○	○	○	○
0AAH	○	○	○	○	○	○	○	○
0FEH	○	○	○	○	○	○	○	○
01H	○	○	○	○	○	○	○	○

ตารางที่ 6.1 ตารางบันทึกผลการทดลอง

โปรแกรมที่ 1.3 (การสร้างไฟวิ่งบน LED)

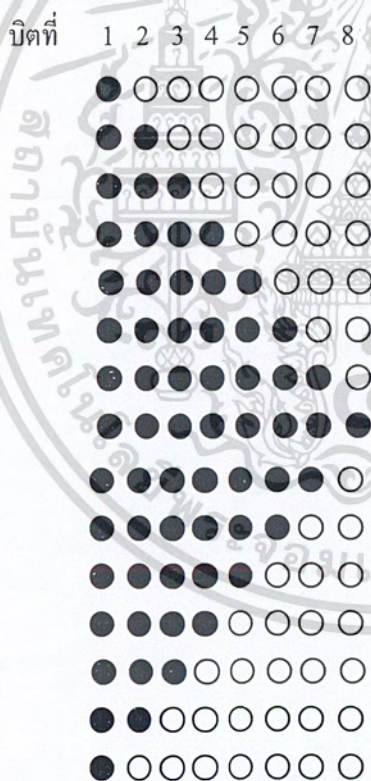
```

0000          1      ORG  0000H
0000 908003    2      MOV  DPTR,#8003H  ;INIT 8255
0003 7480      3      MOV  A,#80H
0005 F0        4      MOVX  @DPTR,A
0006 908000    5      MOV  DPTR,#8000H  ;เลือก P.A
0009 7401      6      START: MOV  A,#01H      ;DATA
000B F0        7      ST:    MOVX  @DPTR,A
000C 23        8      RL   A      ;ROTAGE DATA
000D 12001D    9      CALL  DELAY ;DELAY TIME
0010 B480F8   10     CJNE  A,#80H,ST;Check & Rotate

```

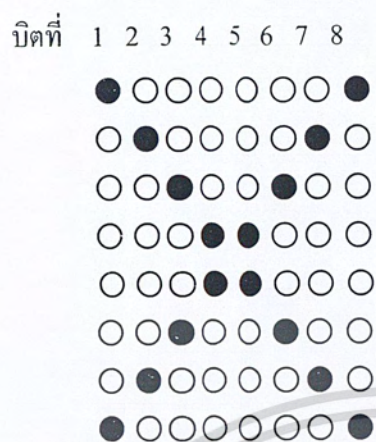
0013 F0	11	ST1:	MOVX @DPTR,A
0014 03	12		RR A
0015 12001D	13		CALL DELAY
0018 B401F8	14		CJNE A,#01H,ST1
001B 80EC	15		JMP START ;START AGAIN
001D 7800	16	DELAY:	MOV R0,#00H ;RELAY TIME
001F 7900	17	LOOP:	MOV R1,#00H
0021 D9FE	18		DJNZ R1,S
0023 D8FA	19		DJNZ R0,LOOP
0025 22	20		RET
0000=	21		END

6. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.2



รูปที่ 6.2

7. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.3



รูปที่ 6.3

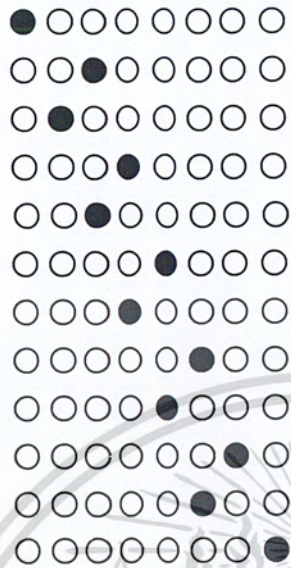
8. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.4



รูปที่ 6.4

9. ให้เขียนโปรแกรมสร้างไฟว้างดังรูปที่ 6.5

บิตที่ 1 2 3 4 5 6 7 8



รูปที่ 6.5



6.2 การทดลองที่2 การอ่านข้อมูลขนาด 8 บิตจากสวิตช์อินพุต และการควบคุม BUZZER

6.2.1 การอ่านข้อมูลขนาด 8 บิต

■ วัตถุประสงค์

1. เพื่อศึกษาการอ่านข้อมูลขนาด 8 บิต ผ่านทางพอร์ต PC ของ 8255

■ ทฤษฎี

สวิตช์อินพุตที่มีในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองนี้เราจะใช้ Dip switch เป็นสวิตช์อินพุต ซึ่ง Dip switch นี้จะต่ออยู่กับพอร์ต PC ของ 8255 โดยที่จะมี บัฟเฟอร์ 74LS240 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PC ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS240 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขาเลือกของ 74LS240 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 บิตที่ 9 เป็นตัว เลือก ให้โมดูล Dip switchทำงาน

■ ขั้นตอนของการทดลอง

1. ต่อสายบััสระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. ปรับ สวิตช์ 1/9ไปที่ตำแหน่ง ON
4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี51 (Assembly 51) อ่านข้อมูลมาจาก สวิตช์อินพุตแล้วส่งออกที่ Port Monitor ดังโปรแกรมที่ 2.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 Assembler ให้เป็น File.Hex
5. ดาวน์โหลด (Download) โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรม 2.1การอ่านค่ามาจากสวิตช์อินพุตแล้วส่งออกที่ Port monitor

0000	1	ORG	0000h
0000 908003	2	MOV	DPTR,#8003H ;INITIAL 8255
0003 7489	3	MOV	A,#89H
0005 F0	4	MOVX	@DPTR,A
0006 908002	5	START: MOV	DPTR,#8002h ;READ DATA
0009 E0	6	MOVX	A,@DPTR
000A 908000	7	MOV	DPTR,#8000h ;DISPLAY DATA
000D F0	8	MOVX	@DPTR,A
000E 120013	9	CALL	RELAY
0011 80F3	10	JMP	START

0013 7800	11	RELAY: MOV R0,#00H
0015 D8FE	12	DJNZ R0,\$
0017 22	13	RET
0000=	14	END

6. จากโปรแกรมไฟวิ่งทั้งหมดที่ได้สร้างไว้ก่อนหน้านี้ ให้เขียนโปรแกรมอ่านค่ามาจาก สวิตช์ อินพุตแล้วส่งไฟวิ่งตามแบบต่างๆดังต่อไปนี้

- เมื่อกดสวิตช์อินพุต 2/1 ส่งให้ไฟวิ่งแบบที่ 1
- เมื่อกดสวิตช์อินพุต 2/2 ส่งให้ไฟวิ่งแบบที่ 2
- เมื่อกดสวิตช์อินพุต 2/3 ส่งให้ไฟวิ่งแบบที่ 3
- เมื่อกดสวิตช์อินพุต 2/4 ส่งให้ไฟวิ่งแบบที่ 4
- เมื่อกดสวิตช์ใดๆก็ตามนอกเหนือจากข้างบนให้ไฟวิ่งดังโปรแกรมที่ 1.3

6.2.2 การทดลองเกี่ยวกับ Buzzer

■ วัตถุประสงค์

- เพื่อศึกษาวิธีการสร้างสัญญาณเสียงด้วย Buzzer
- เพื่อให้สามารถใช้ไมโครคอนโทรลเลอร์ควบคุม Buzzer ให้สร้างสัญญาณเสียงได้

■ ทฤษฎี

Buzzer ที่มีในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองนี้เราจะใช้ Buzzer ที่ใช้แรงดันไฟฟ้าขนาด 12 โวลต์ เป็นตัวกำเนิดสัญญาณเสียง โดย Buzzer นี้จะต่อกับพอร์ต PB บิตที่ 07 ของ 8255 โดยที่มี บัฟเฟอร์ 74LS244 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PB ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขาเลือกของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 บิตที่ 5 เป็นตัวเลือก ให้โมดูล Buzzer ทำงาน จาก บิตที่ 07 ของ พอร์ต PB ของ 8255 จะต่อเข้ากับ ทรานซิสเตอร์ โดยเราจะใช้ ทรานซิสเตอร์ เป็นตัวเปิดปิดการทำงานของ Buzzer.

■ ขั้นตอนของการทดลอง

- ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
- เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่ง Buzzer On และ Off ดังโปรแกรมที่ 2.2 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
- ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 2.2 Buzzer On/Off

0000	1	ORG 0000H
0000 7480	2	MOV A,#80H
0002 908003	3	MOV DPTR,#8003H ;INIT 8255
0005 F0	4	MOVX @DPTR,A
0006 908001	5	MOV DPTR,#8001H ;เลือก PORT B
0009 E580	6	START: MOV A,80H
000B F0	7	MOVX @DPTR,A ;ON BUZZER
000C 120017	8	CALL DELAY
000F 7400	9	MOV A,#00H
0011 F0	10	MOVX @DPTR,A ;OFF BUZZER
0012 120017	11	CALL DELAY
0015 80F2	12	JMP START
0017 7800	13	DELAY: MOV R0,#00H
0019 7900	14	LOOP: MOV R1,#00H
001B D9FE	15	DJNZ R1,S
001D D8FA	16	DJNZ R0,LOOP
001F 22	17	RET
0000=	18	END

4. ให้เขียนโปรแกรมกันขโมย โดยใช้ Dip switch เป็น Sensor ในการตรวจจับขโมย ถ้าเมื่อใดก็ตามที่มีการกด Dip switch ไม่ว่าจะเป็น Dip switch บิตที่เท่าใดก็ตามให้ Buzzer ส่งเสียงเตือนคือดังแล้วเงียบ ดังแล้วเงียบ.

6.3 การทดลองที่3 การควบคุมการทำงานของ Magnetic Relay และ Solid State Relay

6.3.1 การควบคุมการทำงานของ Magnetic Relay

■ วัตถุประสงค์

1. เพื่อศึกษาการควบคุม Magnetic relay โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม
2. เพื่อสามารถเขียนโปรแกรมควบคุมการทำงานของ Magnetic Relay ได้

■ ทฤษฎี

Magnetic Relay เป็นอุปกรณ์ที่ทำหน้าที่เป็นสวิตช์เปิดหรือปิดวงจร โดยอาศัยแรงดันไฟฟ้าที่จ่ายให้กับขดลวดที่พันอยู่บนแกนเหล็ก เพื่อสร้างสนามแม่เหล็กไฟฟ้าเกิดแรงดูดทำให้หน้าสัมผัสของสวิตช์ที่วางอยู่ใกล้ๆ สัมผัสกัน เพื่อต่อวงจรหรือแยกวงจรออกจากกัน, Magnetic relay จะมีสองแบบคือ แบบปกติปิด และแบบปกติเปิด. ในบอร์ดทดลองอินพุตเอาต์พุตเราใช้ Magnetic Relay แบบปกติเปิด 4 ตัวต่อเข้ากับพอร์ต PB บิตที่ 4-7 ของ 8255 โดยจะมี บัฟเฟอร์ 74LS244 เป็นตัวกั้นอยู่ เนื่องจากพอร์ต PB ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกั้นแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขา เลือกของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip-switch เป็นตัวเลือกว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Magnetic Relay ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 8 ของ Dip switch ตัวที่ 1 นี้.

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/8 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ Magnetic Relay ตัวที่ 1 ทำงาน ดังโปรแกรมที่ 3.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 3.1 สั่งให้ Megnetic Relay ตัวที่ 1 ทำงาน

```
0000      1      ORG    0000H
0000 7480    2      MOV   A,#80H
0002 908003  3      MOV   DPTR,#8003H ;INIT 8255
0005 F0      4      MOVX  @DPTR,A
0006 908001  5      MOV   DPTR,#8001H ;เลือก PORT B
0009 E510    6      MOV   A,10H
000B F0      7      MOVX  @DPTR,A ;MAGNETIC RELAY ON
000C 80FE    8      JMP    S
0000=       9      END
```

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง 000A H เป็นค่า 20H, 40H, 80H, 0FF และ 00H แล้วบันทึกผลลัพธ์ลงในตารางข้างล่างนี้

รหัสที่ส่งออกไป	การทำงานของ Magnetic relay
20H	
40H	
80H	
0FFH	
00H	

ตารางที่ 6.2 ตารางบันทึกผลการทดลองของ Solid State Relay

- 6. ให้เขียนโปรแกรมสั่งให้ Magnetic Relay ได้ทำงานทั้ง 4 ตัว โดยที่สั่งให้ติดครั้งละ 2 ตัว นานประมาณ 30 วินาที แล้ว 2 ตัวที่เหลือก็ติดนานประมาณ 1 นาที
- 7. ให้เขียนโปรแกรม สั่งให้ Magnetic Relay ทำงานเหมือนไฟวิ่งโดยทำงานทีละตัวและการทำงานของ Magnetic Relay แต่ละตัวนั้นนานประมาณ 30 วินาที

6.3.2 การควบคุมการทำงานของ Solid State Relay

■ วัตถุประสงค์

1. เพื่อศึกษาการทำงานของ Solid state relay
2. เพื่อศึกษาการควบคุม Solid state relay โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม

■ ทฤษฎี

Solid state relay เป็นอุปกรณ์ที่ทำหน้าที่คล้ายกับรีเลย์ คือเป็นอุปกรณ์ในการตัดต่อวงจรแตกต่างกันตรงที่รีเลย์อาศัยการทำงานของขดลวดสนามแม่เหล็กไฟฟ้าเพื่อคูดหน้าสัมผัสให้เกิดการเคลื่อนที่ในการตัด-ต่อวงจร แต่ Solid state relay จะทำงานโดยอาศัยอุปกรณ์ไทรแอสต์เป็นตัวตัด-ต่อวงจร Solid state relay จะมีข้อดีคือมีขนาดเล็กกว่า มีกำลังงานสูญเสียต่ำกว่า เพราะเพียงแต่จ่ายแรงดันไฟฟ้าให้กับตัวเชื่อมต่อโยทางแสงที่มีอยู่ข้างในเท่านั้น

ในบอร์ดทดลองอินพุตเอาต์พุตใช้ Solid state Relay 4 ตัวต่อเข้ากับพอร์ต PB บิตที่ 0-3 ของ 8255 โดยจะมี บัฟเฟอร์ 74LS244 เป็นตัวกันอยู่เนื่องจากว่าพอร์ต PB ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Solid state relay ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 7 ของ Dip switch ตัวที่ 1 นี้

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/7 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ Solid State Relay ตัวที่ 1 ทำงาน ดังโปรแกรมที่ 3.2 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 3.2 สั่งให้ Solid State Relay ตัวที่ 1 ทำงาน

0000	1	ORG	0000H
0000 7480	2	MOV	A,#80H
0002 908003	3	MOV	DPTR,#8003H ;INIT 8255
0005 F0	4	MOVX	@DPTR,A
0006 908001	5	MOV	DPTR,#8001H ;เลือก PORT B
0009 E501	6	MOV	A,01H


```
000B F0      7      MOVX  @DPTR,A    ;SOLID STATE RELAY ON
000C 80FE     8      JMP      $
0000=         9      END
```

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง 000A H เป็นค่า 02H, 04H, 08H, 0FF และ 00H แล้วบันทึกผลลัพธ์ลงในตารางข้างล่างนี้

รหัสที่ส่งออกไป	การทำงานของ Solid State Relay
02H	
04H	
08H	
0FFH	
00H	

ตารางที่ 6.3 ตารางบันทึกผลการทดลองของ Solid State Relay

- 6. ให้เขียนโปรแกรม สั่งให้ Solid State Relay ทำงานเหมือนไฟวิ่งโดยทำงานทีละตัวเริ่มจากตัวที่ 1 แล้วตัวที่ 3 แล้วกลับมาที่ตัวที่ 2 แล้วไปที่ตัวที่ 4 และการทำงานของ Solid State Relay แต่ละตัวนั้นนานประมาณ 30 วินาที
- 7. ให้เขียนโปรแกรม สั่งให้ Solid State Relay ทำงานตามการ เปิด, ปิด ของโมดูล Dip Switch บิตที่ 1-4 ของ Dip Switch ตัวที่ 2. ก่อนการทดลองควรปรับ สวิตช์ 1/9 ไปที่ตำแหน่ง ON ก่อน

6.4 การทดลองที่ 4 การแสดงผลข้อมูล 7 ส่วน (7 Segment display)

■ วัตถุประสงค์

1. เพื่อศึกษาการแสดงผลของ 7 Segment Display โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม
2. เพื่อสามารถควบคุมการแสดงผลของ 7 Segment Display ได้

■ ทฤษฎี

7 Segment Display เป็นตัวแสดงผลอีกแบบหนึ่งที่แสดงผลออกมาเป็นตัวเลข หรือตัวอักษรได้ บางตัว ภายใน 7 Segment จะประกอบไปด้วย LED 8 ดวงที่ต่อร่วมกับเป็น 7 Segment โดยจะมีทั้งแบบ Common Anode และ Cathode. สำหรับโมดูล 7 Segment ที่มีในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองนี้การทำงานจะเป็นแบบ Scan ซึ่งจะมี 7 Segment อยู่ 4 หลักโดยที่สัญญาณข้อมูลจะต่ออยู่กับพอร์ต PA ส่วนสัญญาณสำหรับ Scan นั้นจะต่ออยู่กับพอร์ต PB บิตที่ 0-3 สำหรับสัญญาณข้อมูลที่ต่อกับพอร์ต PA ของ 8255 นั้นจะมี บัฟเฟอร์ 74LS240 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PA ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS240 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขาเลือก ของ 74LS240 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล 7 Segment ขาเลือก ของ 74LS240 จะต่ออยู่กับ บิตที่ 2 ของ Dip switch ตัวที่ 1 นี้

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/2 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) ส่งข้อมูลออกมาสั่งให้ 7 Segment หลักที่ 1 ติดเป็นตัวเลข 0 ดังโปรแกรมที่ 4.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 4.1 สั่งให้ 7 Segment หลักที่ 1 ติดเป็นเลข 0

0000	1	ORG	0000H
0000 908003	2	MOV	DPTR,#8003H ;INITIAL 8255
0003 7480	3	MOV	A,#80H
0005 F0	4	MOVX	@DPTR,A
0006 908000	5	START: MOV	DPTR,#8000H ;SEND DATA TO PA
0009 74BF	6	MOV	A,#0BFH


```
000B F0      7      MOVX  @DPTR,A
000C 908001   8      MOV   DPTR,#8001H ;เลือก 7 SEGMENT
000F 7401     9      MOV   A,#01H
0011 F0      10     MOVX  @DPTR,A
0012 80F2     11     JMP   START
0000=        12     END
```

5. จากโปรแกรมที่ 3.1 ให้เปลี่ยนข้อมูลที่ตำแหน่ง 0009H เป็นค่าต่างๆดังตารางข้างล่างนี้แล้ว บันทึกผลที่ได้รับ

รหัส	7 Segment display
86H	
0CFH	
0FDH	
17H	
0EFH	

ตารางที่ 6.4 ตารางบันทึกผลการทดลองของ 7 Segment display

6. จากโปรแกรมที่ 3.1 จงสร้างตารางสำหรับ รหัส ของ 7 Segment ทุกๆตัวตั้งแต่ 0-9 แล้ว บันทึกลงในตารางข้างล่างนี้

รหัส	แสดงผล 7 Segment
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9

ตารางที่ 6.5 ตารางบันทึก รหัส 7 Segment display จากการทดลอง

7. จงเขียนโปรแกรมสั่งให้ 7 Segment นับตั้งแต่ 0-9 เริ่มจากติดเป็นเลข 0 ที่หลักที่ 1 และติดเป็นเลข 1 ที่หลักที่ 2 ต่อๆไป เมื่อถึงหลักที่ 4 แล้วให้เลขตัวถัดไปติดที่หลักที่ 1 และเลขต่อไปติดที่หลักต่อไป จนครบถึงเลข 9 แล้วให้กลับมาเริ่มต้นใหม่



6.5 การทดลองที่ 5 การรับข้อมูลจากคีย์บอร์ด (Scan Key board)

■ วัตถุประสงค์

1. เพื่อศึกษาวิธีการสแกนคีย์บอร์ด (Scan key board)
2. เพื่อให้สามารถนำคีย์บอร์ดไปประยุกต์ใช้งานได้

■ ทฤษฎี

คีย์บอร์ดเป็นอุปกรณ์อินพุต ชนิดหนึ่ง ในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองได้นำเอาคีย์บอร์ดขนาด 3 x 4 มาใช้ในการศึกษา. โดยคีย์บอร์ดนี้เราจะใช้ ไมโครสวิตช์แทนคีย์บอร์ดในแต่ละปุ่ม โมดูลคีย์บอร์ดจะต่ออยู่กับพอร์ต PC ของ 8255 โดยที่จะมี บัฟเฟอร์ 74LS244 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PC ได้ถูกใช้งานต่ออยู่กับหลายโมดูล ดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Key board ขาเลือกของ 74LS244 จะต่ออยู่กับ บิตที่ 10 ของ Dip switch ตัวที่ 1 นี้

ในการทำงานเมื่อมีการกดคีย์ใดหนึ่งสัญญาณของอินพุตของคอลัมน์ C1, C2, C3 ที่เป็นลอจิก "1" ทั้งหมด 1 1 1 ในขณะที่แถวมีค่าเป็น "0" อยู่แถวหนึ่งแล้วส่วนแถวที่เหลือมีค่าเป็น "1" ทั้งหมดและวนรอบไปเรื่อยๆนั้น เมื่อมีการกดคีย์ใดคีย์หนึ่งจะทำให้คอลัมน์ที่ถูกกดที่มีค่าเป็น "1" นั้นมีค่าเป็น "0" เช่น มีการกดที่คีย์หมายเลข 4 คอลัมน์ C1, C2, C3 ที่มีค่าเป็น 1 1 1 ทั้งหมด ก็จะกลายเป็น 0 1 1 เพราะที่ C1 จะกลายเป็น "0" ในจังหวะที่แถว R2 มีสถานะเป็น "0" เมื่อนำผลของการอินพุตและเอาพุตมารวมด้วยกันก็สามารถบอกตำแหน่งของคีย์บอร์ดที่ถูกกดได้

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
3. ปรับ สวิตช์ 1/10 ไปที่ตำแหน่ง ON
4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) อ่านค่ามาจากคีย์บอร์ดแล้วส่งผลไปแสดงที่โมดูล Port monitor ดังโปรแกรมที่ 4.2 เมื่อเขียนเสร็จ แล้ว Compile โดย SXAS1 ASSEMBLER ให้เป็น File.Hex
5. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 5.1 การหาค่าประจำคีย์แต่ละตัว

```

0000          1          ORG    0000H
0080=         2  PORTA    EQU    80H
0082=         3  PORTC    EQU    82H
0000 7488     4          MOV    A,#88H          ;INITIAL 8255
0002 908003   5          MOV    DPTR,#8003H
0005 F0       6          MOVX   @DPTR,A
0006 7900     7  MAIN:    MOV    R1,#00H
0008 12000F   8          CALL   SCAN
000B F580     9          MOV    PORTA,A          ;DISPLAY OUTPUT
000D 80F7    10         JMP     MAIN
000F 900026  11  SCAN:    MOV    DPTR,#TABLE          ;SCAN KEY
0012 E0      12         MOVX   A,@DPTR
0013 F8      13         MOV    R0,A
0014 F582    14         MOV    PORTC,A
0016 E582    15         MOV    A,PORTC
0018 5570    16         ANL    A,70H
001A B47007  17         CJNE   A,#70H,SCAN1
001D A3      18         INC    DPTR
001E 09      19         INC    R1
001F B905ED  20         CJNE   R1,#05H,SCAN
0022 80E2    21         JMP     MAIN
0024 28      22  SCAN1:  ADD    A,R0
0025 22      23         RET
0026 0E0D0B07 24  TABLE: DB    0EH,0DH,0BH,07H
0000=        25         END

```


6. เขียนโปรแกรมอ่านค่าจากคีย์บอร์ดแล้วส่งออกไปแสดงผลที่ Port monitor โดยที่ให้แสดงผลเป็นค่าเลขฐาน 2 ในเมื่อกดคีย์ใดๆ เช่นกดคีย์หมายเลข 1 ให้แสดงผลเป็นค่า 1 ในเลขฐาน 2, กดคีย์หมายเลข 4 ให้แสดงผลเป็นค่า 4 ในเลขฐาน 2
7. จงเขียนโปรแกรมอ่านค่าจากคีย์บอร์ดแล้วส่งไปแสดงผลที่ 7 Segment หลักที่ 2 โดยการใช้รหัส ของ 7 Segment ในตารางข้างบนที่ได้ทำการทดลองมาก่อนหน้านี้ ในการกำหนด รหัส ของ 7 Segment ในการแสดงผล. ก่อนการทดลองปรับ สวิตช์ 1/2 ไปที่ตำแหน่ง ON ก่อน



6.6 การทดลองที่ 6 การแสดงผลข้อมูลด้วยจอทเมตริก (Dot matrix display)

■ วัตถุประสงค์

1. เพื่อศึกษาการทำงานของ Dot matrix display
2. เพื่อศึกษาการควบคุมการทำงานของ Dot matrix display
3. เพื่อสามารถเขียนโปรแกรมควบคุมการทำงานของ Dot matrix display ได้

■ ทฤษฎี

Dot matrix Display เป็นอุปกรณ์แสดงผลอีกตัวแบบหนึ่ง ที่สามารถแสดงผลได้หลายอย่าง เพราะสามารถที่จะแสดงผลได้ทั้งตัวอักษรและ ภาพสัญลักษณ์ต่างๆ เท่าจำนวนจุดของ Dot matrix นั้น Dot matrix เกิดจากการเอา LED มาวางเรียงกันเป็นแถวในแนวตั้งและแนวนอน มีหลายขนาด มีทั้งที่เป็น Anode Column, Cathode Row และ Cathode Column, Anode Row. ในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองของเรานี้จะใช้ Dot matrix ขนาด 5x7 แบบ Anode Column, Cathode Row โดยโมดูลนี้สัญญาณที่เลือก แต่ละ Column จะต่อกับพอร์ต PB และต่อเข้ากับ Decoder 4 to 16 (74LS154) จำนวน 2 ตัว, สัญญาณ Data จะต่อกับพอร์ต PA ของ 8255 โดยจะมี บัฟเฟอร์ 74LS244 เป็นตัวกันอยู่ เนื่องจากว่าพอร์ต PA ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขาเลือก ของ 74LS240 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Dot matrix display ขาเลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 12 ของ Dip switch ตัวที่ 1 นี้.

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/12 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) ส่งข้อมูลออกมาสั่งให้ 7 Segment หลักที่ 1 ดิดเป็นตัวเลข 0 ดังโปรแกรมที่ 5.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 5.1 สั่งให้ Dot matrix Column ที่ 1 ติดหมด

```

0000          1      ORG   0000H
0000 7480      2      MOV   A,#80H           ;INITIAL 8255
0002 908003     3      MOV   DPTR,#8003H
0005 F0         4      MOVX  @DPTR,A
0006 74FF       5      MOV   A,#0FFH
0008 908000     6      MOV   DPTR,#8000H      ;SEND DATA DISPLAY
000B F0         7      MOVX  @DPTR,A
000C 908001     8      MOV   DPTR,#80001H     ;เลือก COLUMN
000F 7400       9      MOV   A,#00H
0011 F0        10     MOVX  @DPTR,A
0012 7804      11     MOV   R0,#04H
0014 7900      12 LOOP: MOV   R1,#00H
0016 D9FE      13     DJNZ  R1,$
0018 D8FA      14     DJNZ  R0,LOOP
0000=          15     END

```

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง Address 0007h จากค่า 0ffh เป็นค่า 0h, 02h, 04h, 08h, 10h, 20h, 40h, 80h หลังจากนั้นทดลองเปลี่ยนค่าข้อมูลที่ตำแหน่ง Address 0010h จากค่า 00h เป็น 01h, 02h, แล้วบันทึกผลลัพธ์ของแต่ละการทดลองลงในตารางข้างล่างนี้

ข้อมูลที่ Address 0007	ข้อมูลที่ Address 0010 เป็น : 01H	ข้อมูลที่ Address 0010 เป็น :	ข้อมูลที่ Address 0010 เป็น :
01H	Row : Column :	Row : Column :	Row : Column :
02H	Row : Column :	Row : Column :	Row : Column :
04H	Row : Column :	Row : Column :	Row : Column :
08H	Row : Column :	Row : Column :	Row : Column :
10H	Row : Column :	Row : Column :	Row : Column :
20H	Row : Column :	Row : Column :	Row : Column :
40H	Row : Column :	Row : Column :	Row : Column :
80H	Row : Column :	Row : Column :	Row : Column :

ตารางที่ 6.6 ตารางบันทึกผลของการทดลอง Dot matrix display

6. ให้เขียนโปรแกรม แสดงตัวอักษรตัว A ปรากฏขึ้นที่ Dot matrix display
7. ให้เขียนโปรแกรมแสดงชื่อของตัวเองเป็นภาษาลาวที่ dot matrix display
8. ให้เขียนโปรแกรมแสดงชื่อของตัวเองวิ่งอยู่บน Dot matrix display



6.7 การทดลองที่ 7 การแสดงผลข้อมูลด้วยตัวแสดงผลแบบผลึกเหลว (LCD)

■ วัตถุประสงค์

1. เพื่อศึกษาการทำงานของ LCD
2. เพื่อศึกษาการควบคุมการทำงานของ LCD
3. เพื่อสามารถเขียนโปรแกรมควบคุมการทำงานของ LCD ได้

■ ทฤษฎี

LCD เป็นอุปกรณ์แสดงผลอีกแบบหนึ่งที่นิยมใช้กันมากในปัจจุบัน เพราะใช้กระแสไฟต่ำ สามารถแสดงผลได้หลายอย่าง แสดงผลได้ทั้งตัวอักษร ตัวเลข และ รูปภาพ LCD ก็คือจุดแสดงผลที่มีวงจรควบคุมการแสดงผลต่ออยู่ภายใน ซึ่งจะช่วยให้การใช้งาน LCD ง่ายขึ้น

LCD มีอยู่ด้วยกันหลายแบบ ที่มีคุณสมบัติแตกต่างกันออกไป แต่สามารถที่จะแบ่งออกเป็น 2 แบบใหญ่คือ แบบที่เป็น Dot matrix ใช้แสดงตัวอักษรขนาด 5x8 จุด จะมีจำนวนตัวอักษรและจำนวนบรรทัดแตกต่างกันไปแต่ละรุ่น แบบที่เป็น Graphic สามารถแสดงผลแบบ Bit map คือสร้างเป็นภาพต่างๆได้ ในบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลองของเราจะใช้ LCD ขนาด 16 ตัวอักษร 2 บรรทัด แบบ Dot matrix โดยโมดูลนี้สัญญาณควบคุม จะต่อกับพอร์ต PB บิตที่ PB5-PB7 ของ 8255 สัญญาณ Data จะต่อกับพอร์ต PA ของ 8255 โดยจะมี บัฟเฟอร์ 74LS244 เป็นตัวกันอยู่เนื่องจากว่าพอร์ต PA ได้ถูกใช้งานต่ออยู่กับหลายโมดูลดังนั้นเราจึงต้องใช้ 74LS244 เป็นตัวกันแต่ละโมดูลเพื่อไม่ให้ไปรบกวนกัน สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ที่จะให้โมดูลไหนทำงาน สำหรับโมดูล LCD ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 3 ของ Dip switch ตัวที่ 1 นี้.

ก่อนการใช้งานของ LCD จะต้องมีการเขียนคำสั่งควบคุมต่างๆเพื่อกำหนดการทำงานของ LCD ให้ LCD ทำงานตามเงื่อนไขต่างๆ ซึ่ง LCD จะมีชุดคำสั่งเฉพาะอยู่ดังข้างล่างนี้

เป็นคำสั่งลบข้อมูลต่างๆ ที่อยู่ ใน DDRAM ให้เป็นข้อมูลที่ว่างเปล่า ทำให้ข้อมูลที่อยู่บนจอ LCD ถูกลบหายไป และสั่งให้ CURSOR ไปอยู่ที่ตำแหน่งแรกของจอ LCD (ตำแหน่งบนซ้ายสุด)

2. RETURN HOME

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	X

เป็นคำสั่งที่กำหนดให้ DDRAM ADDRESS เป็น ๐ คือ ไปอยู่ที่ตำแหน่งแรกสุด และสั่งให้ CURSOR ไปอยู่ที่ตำแหน่งแรกของจอ LCD

3. CURSOR PRESET

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	I/D	S

เป็นคำสั่งกำหนดตำแหน่งของ CURSOR ในขณะที่ LED แสดงข้อความกำหนด I/D และ S ให้ทำก่อนที่จะมีการเขียนข้อมูลใน DDRAM และจะต้องไม่ใช่คำสั่ง CLEAR DISPLAY อีก

- I/D = “0” ทิศทางของ CURSOR และตำแหน่งของ DDRAM จะเป็นแบบเพิ่มขึ้น
- I/D = “1” ทิศทางของ CURSOR และตำแหน่งของ DDRAM จะเป็นแบบลดลง
- S = “0” CURSOR จะเลื่อนไปตามทิศทางของการกำหนดค่า I/D
- S = “1” CURSOR จะอยู่กับที่แต่ตัวอักษรจะเลื่อนไปตามทิศทางของการกำหนดค่า I/D

4. DISPLAY ON/OFF

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	C	B

- D = “0” DISPLAY OFF
- D = “1” DISPLAY ON
- C = “0” CURSOR OFF
- C = “1” CURSOR ON มีขีดได้ตัวอักษร
- B = “0” CURSOR ไม่มีการกระพริบ
- B = “1” CURSOR มีการกระพริบเป็นรูปสี่เหลี่ยม

5. DISPLAY SHIF

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	S/C	P/L	X	X

S/C = "0" เลื่อน CURSOR ไปตามทิศทางที่กำหนดด้วย R/L ไป 1 ตำแหน่ง

S/C = "1" เลื่อนข้อความไปตามทิศทางที่กำหนดด้วย R/L ไป 1 COLUMN ทุกบรรทัด

R/L = "0" กำหนดทิศทางไปทางซ้าย

R/L = "1" กำหนดทิศทางไปทางขวา

6. FUNCTION SET

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	DL	N	F	X	X

DL = "0" LCD ติดต่อกับตัวควบคุมภายนอกแบบ 4 บิต

DL = "1" LCD ติดต่อกับตัวควบคุมภายนอกแบบ 8 บิต

N = "0" กำหนดจำนวนบรรทัดแบบ 1/8 และ 1/11 Duty

N = "1" กำหนดจำนวนบรรทัดแบบ 1/16 Duty

F = "0" แสดงอักษรแบบ 5x7 Dots

F = "1" แสดงอักษรแบบ 5x10 Dots (ถ้า LCD เป็นแบบ 5x7 dots จะไม่มีผลต่อคำสั่งนี้)

■ ขั้นตอนของการทดลอง

1. ต่อสายバスเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/3 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) ส่งข้อมูลออกมาสั่งให้ 7 Segment หลักที่ 1 ติดเป็นตัวเลข 0 ดังโปรแกรมที่ 6.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download) โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 6.1 ตั้งให้ LCD แสดงผลเป็น A ที่บนจอ LCD

0000	1	ORG 0000H
0000 7488	2	MOV A,#88H
0002 908003	3	MOV DPTR,#80003H
0005 F0	4	MOVX @DPTR,A
0006 7438	5	MOV A,#38H ;FUNCTION SET
0008 120029	6	CALL SEND_CMD
000B 740E	7	MOV A,#0EH ;DISPLAY ON/OFF
000D 120029	8	CALL SEND_CMD
0010 7401	9	MOV A,#01H ;CLEAR DISPLAY
0012 120029	10	CALL SEND_CMD
0015 7406	11	MOV A,#06H ;CURSOR PRESET
0017 120029	12	CALL SEND_CMD
001A 120060	13	CALL DELAY_
001D 7480	14	MOV A,#80H ;SET DDRAM ADDRESS
001F 120029	15	CALL SEND_CMD
0022 7441	16	MOV A,'#A'
0024 120040	17	CALL SEND_DATA ;SEND Data display
0027 80FE	18	JMP \$
0029 908000	19	SEND_CMD: MOV DPTR,#80000H
002C F0	20	MOVX @DPTR,A
002D 9080001	21	MOV DPTR,#80001H
0030 7400	22	MOV A,#00H
0032 F0	23	MOVX @DPTR,A
0033 7480	24	MOV A,#80H
0035 F0	25	MOVX @DPTR,A
0036 7400	26	MOV A,#00H
0038 F0	27	MOVX @DPTR,A
0039 120057	28	CALL DELAY
003C 120057	29	CALL DELAY
003F 22	30	RET

0040 908000	31	SEND_DATA: MOV DPTR,#80000H	
0043 F0	32	MOVX @DPTR,A	
0044 9080001	33	MOV DPTR,#80001H	
0047 7420	34	MOV A,#20H	
0049 F0	35	MOVX @DPTR,A	
004A 74A0	36	MOV A,#0A0H	
004C F0	37	MOVX @DPTR,A	
004D 7420	38	MOV A,#20H	
004F F0	39	MOVX @DPTR,A	
0050 120057	40	CALL DELAY	
0053 120057	41	CALL DELAY	
0056 22	42	RET	
0057 7910	43	DELAY: MOV R1,#10H	;DELAY 1
0059 7800	44	BACK1: MOV R0,#00H	
005B D8FE	45	DJNZ R0,S	
005D D9FA	46	DJNZ R1,BACK1	
005F 22	47	RET	
0060 7901	48	DELAY_: MOV R1,#01H	;DELAY 2
0062 7A10	49	BACK: MOV R2,#10H	
0064 7B00	50	LL: MOV R3,#00H	
0066 DBFE	51	DJNZ R3,S	
0068 DAFA	52	DJNZ R2,LL	
006A D9F6	53	DJNZ R1,BACK	
006C 22	54	RET	
0000=	55	END	

5. จงเขียนโปรแกรมแสดงชื่อของตัวเองบน LCD
6. จงเขียนโปรแกรมแสดงข้อความสองบรรทัดบน LCD

6.8 การทดลองที่ 8 การควบคุมการทำงานของมอเตอร์ (Motor control) และ ของสตีปปีงมอเตอร์ (Stepping Motor)

6.8.1 การควบคุมการทำงานของมอเตอร์

■ วัตถุประสงค์

1. เพื่อศึกษาการควบคุมมอเตอร์ โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม

■ ทฤษฎี

การควบคุมความเร็วของมอเตอร์ให้ความเร็วคงที่ตามที่เรต้องการเป็นเรื่องที่ทำได้ค่อนข้างยาก ถ้าใช้การควบคุมแบบธรรมดาทั่วไปคือ การควบคุมแรงดันที่จ่ายให้กับมอเตอร์ การที่จะควบคุมความเร็วของมอเตอร์ให้มีความเร็วที่คงที่นั้น ต้องควบคุมกระแสที่จ่ายให้กับมอเตอร์ โดยการจ่ายแรงดันให้กับมอเตอร์เป็นช่วงๆ กระแสไฟที่มอเตอร์ได้รับจะเป็นค่าเฉลี่ยที่เกิดขึ้นในช่วง ON โดยความถี่ของการ ON และ OFF คงที่ เมื่อความเร็วคงที่ที่เราสามารถที่จะเปลี่ยนความเร็วรอบของมอเตอร์ได้โดยทำให้ช่วง ON มีเวลาที่ยาวกว่าช่วง OFF กระแสเฉลี่ยที่มอเตอร์ได้รับจะมีค่ามากขึ้น ความเร็วรอบของมอเตอร์ก็จะเพิ่มขึ้น และถ้าหากต้องการลดความเร็วรอบของมอเตอร์ลง ก็ให้ลดช่วง ON ให้มีเวลาน้อยลง

ในบอร์ดทดลองอินพุตเอาต์พุตโมดูล Motor control ได้ต่อกับพอร์ต PB บิตที่ 4 และ 5 ของ 8255 โดยบิตที่ 5 จะเป็นบิตที่ควบคุมการทำงานของมอเตอร์ ส่วนบิตที่ 4 จะต่อเข้ากับรีเลย์ซึ่งทำหน้าที่เป็นตัวเปลี่ยนทิศทางการหมุนของมอเตอร์ แต่เนื่องจากว่าพอร์ต PB ได้ต่อกับหลายโมดูลดังนั้นเราจะใช้ บัฟเฟอร์ 74LS244 เป็นตัวกันเพื่อไม่ให้ไปรบกวนโมดูลอื่นๆ สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือกว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Motor control ขา เลือก ของ 74LS244 จะต่อกับ บิตที่ 6 ของ Dip switch ตัวที่ 1 นี้.

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ มอเตอร์ ทำงาน ดังโปรแกรมที่ 7.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. คัดลอกโปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 7.1 สั่งให้มอเตอร์ ทำงาน

0000	1	ORG	0000H	
0000 748A	2	MOV	A,#8AH	;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVX	@DPTR,A	
0006 7420	5	MOV	A,#20H	;ON MOTOR
0008 908001	6	MOV	DPTR,#80001H	
000B F0	7	MOVX	@DPTR,A	
000C 80FE	8	JMP	S	
0000=	9	END		

- 5. จงเขียนโปรแกรมสั่งให้มอเตอร์หมุนชั่วคราวหนึ่งแล้วหยุดหมุน
- 6. ให้เขียนโปรแกรมอ่านค่ามาจาก Dip Switch บิตที่ 4 และ บิตที่ 5 เพื่อสั่งให้มอเตอร์ทำงาน โดยที่บิตที่ 5 ให้เป็นการควบคุมการทำงานของมอเตอร์ บิตที่ 4 ให้เป็นการควบคุมทิศทางการหมุนของมอเตอร์ ก่อนการทดลองต้องปรับสวิตช์ 1/9 ไปที่ตำแหน่ง ON ก่อน
- 7. ให้เขียนโปรแกรมสั่งให้มอเตอร์หมุนไปทางหน้าแล้วหมุนกลับหลัง โดยการหมุนแต่ละอย่างนั้นนานประมาณ 5 วินาที

6.8.2 การควบคุมการทำงานของสเต็ปมิ่งมอเตอร์ (Stepping Motor)

■ วัตถุประสงค์

- 1. เพื่อศึกษาการกระตุ้นเฟสของสเต็ปมิ่งมอเตอร์
- 2. เพื่อศึกษาการควบคุมการทำงานของสเต็ปมิ่งมอเตอร์โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม

■ ทฤษฎี

สเต็ปมิ่งมอเตอร์เป็นมอเตอร์ที่เหมาะสมสำหรับงานควบคุมการหมุน ที่ต้องการตำแหน่งที่แน่นอน สเต็ปมิ่งมอเตอร์ต่างจากคีมอเตอร์ ตรงที่การหมุนไปที่ละสเต็ป อาจเป็นสเต็ปละ 0.9, 1.8, 5, 7.5, 15, หรือ 50 องศา ตามแต่ชนิด แต่คีมอเตอร์จะหมุนแบบต่อเนื่อง ไม่สามารถที่จะหมุนเป็นสเต็ปได้ ส่วนการกระตุ้นเฟส เพื่อให้สเต็ปมิ่งทำงานนั้นสามารถทำได้ 3 แบบคือ

- 1. กระตุ้นแบบเฟสเดียว (1 Phase Excitation)
- 2. กระตุ้นแบบสองเฟส (2 Phase Excitation)
- 3. กระตุ้นแบบหนึ่งเฟสและสองเฟส (1-2 Phase Excitation)

Phase Driving Method	Phase 1				Phase 2				Phase 1-2							
	Excitation				Excitation				Excitation							
	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8
A	1				1	1			1	1	1					
B		1				1	1				1	1	1			
/A			1				1	1					1	1	1	
/B				1	1			1	1						1	1

ตารางที่ 6.8 ตารางการกระตุ้นเฟสแต่ละแบบของ Stepping Motor

ในบอร์ดทดลองอินพุตเอาต์พุตโมดูล Stepping Motor เราจะใช้สเต็ปมอเตอร์ขนาด 12 โวลต์ ในการหมุน 1 สเต็ปเท่ากับ 7.5 องศา และโมดูลนี้ได้ต่ออยู่กับพอร์ต PB บิตที่ 0-3 ของ 8255 แต่เนื่องจากว่าพอร์ต PB ได้ต่ออยู่กับหลายโมดูลดังนั้นเราจะใช้ บัฟเฟอร์ 74LS244 เป็นตัวกันเพื่อไม่ให้ไปรบกวนโมดูลอื่นๆ สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็น ตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Magnetic Relay ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 5 ของ Dip switch ตัวที่ 1 นี้

- ขั้นตอนของการทดลอง
- 1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
- 2. ปรับ สวิตช์ 1/5 ไปที่ตำแหน่ง ON
- 3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) ตั้งให้ สเต็ปมอเตอร์ ทำงาน ดัง โปรแกรมที่ 8.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
- 4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hexลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกต ผลที่ได้รับ

โปรแกรมที่ 8.1 ตั้งให้ สเต็ปมอเตอร์ ทำงานโดยการกระตุ้นแบบเฟสเดียว

```
0000          1          ORG  0000H
0000 7480      2          MOV  A,#80H          ;INITIAL 8255
0002 908003    3          MOV  DPTR,#8003H
0005 F0        4          MOVX  @DPTR,A
0006 7F00      5  START:  MOV  R7,#00H
0008 90002B    6          MOV  DPTR,#TABLE      ;เลือก DATA
000B E0        7  NEXT:   MOVX  A,@DPTR
```

000C A3	8	INC	DPTR	
000D 0F	9	INC	R7	
000E C082	10	PUSH	82H	
0010 C083	11	PUSH	83H	
0012 908001	12	MOV	DPTR,#8001H	;STEPPING ON
0015 F0	13	MOVX	@DPTR,A	
0016 120022	14	CALL	DELAY	
0019 D083	15	POP	83H	
001B D082	16	POP	82H	
001D BF05EB	17	CJNE	R7,#05H,NEXT	
0020 80E4	18	JMP	START	
0022 780A	19	DELAY:	MOV R0,#0AH	;ROUTINE DELAY
0024 7900	20	LOOP1:	MOV R1,#00H	
0026 D9FE	21	DJNZ	R1,S	
0028 D8FA	22	DJNZ	R0,LOOP1	
002A 22	23	RET		
002B 10040208	24	TABLE:	DB 10H,04H,02H,08H	;DATA
0000=	25	END		

5. จงเขียน โปรแกรมสั่งให้สเต็ปปีงมอเตอร์ทำงานโดยการกระตุ้นแบบสองเฟส
6. จงเขียน โปรแกรมสั่งให้สเต็ปปีงมอเตอร์ทำงานโดยการกระตุ้นแบบหนึ่งเฟส และสองเฟส
7. จงเขียน โปรแกรมสั่งให้สเต็ปปีงมอเตอร์ทำงานโดยหมุนไป 90 องศา แล้วหยุด

6.9 การทดลองที่ 9 การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก

■ วัตถุประสงค์

1. เพื่อศึกษาการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก

■ ทฤษฎี

การแปลงกลุ่มสัญญาณดิจิตอล ให้กลายเป็นปริมาณของแรงดันหรือกระแส เราเรียกรวมกันว่า DAC (Digital to Analog Converter) วงจรพื้นฐานของการแปลงสัญญาณดิจิตอลให้เป็น อนาล็อกนี้คือวงจร R-2R Ladder แล้วหลังจากนั้นจะใช้ Op Amp เป็นตัวรวบรวม สัญญาณที่แปลงได้นั้น ในการแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลนั้นเราสามารถที่จะคำนวณได้ดังสมการข้างล่างนี้

$$V_o = (V_{ref}/R_f) \times R_o \times (D_7/2 + D_6/4 + D_5/8 + D_4/16 + D_3/32 + D_2/64 + D_1/128 + D_0/256)$$

$$V_{ref} = 5 \text{ V}$$

$$R_f = 4.7 \text{ K}$$

$$R_o = 2.35 \text{ K}$$

$$D_o = \text{ลอจิกของสัญญาณดิจิตอล}$$

ดังนั้นเราได้สมการ

$$V_o = 2.55 \times (D_7/2 + D_6/4 + D_5/8 + D_4/16 + D_3/32 + D_2/64 + D_1/128 + D_0/256)$$

ในบอร์ดทดลองอินพุตเอาต์พุตโมดูล DAC เราจะใช้ไอซี DAC 0808 เป็นตัวที่ทำหน้าที่ในการแปลงสัญญาณดิจิตอลให้เป็นอนาล็อก ซึ่งไอซีเบอร์นี้จะเป็นไอซีที่แปลงสัญญาณดิจิตอลขนาด 8 บิตให้เป็นกระแส ซึ่ง 8 บิตสามารถที่จะแปลงให้มีความละเอียดได้ 256 จุด และจากผลลัพธ์ที่ได้จากการแปลงจะใช้โอปแอมป์ 741 แปลงกระแสที่ได้ให้เป็นแรงดันที่อยู่ในช่วง 0-2.55 โวลต์ และโมดูลนี้ได้ต่อกับพอร์ต PA ของ 8255. แต่เนื่องจากว่าพอร์ต PA ได้ต่อกับหลายโมดูลดังนั้นเราจะใช้ บัฟเฟอร์ 74LS244 เป็นตัวกันเพื่อไม่ให้ไปรบกวน โมดูลอื่นๆ สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Digital to Analog converter นี้ขา เลือก ของ 74LS244 จะต่อกับ บิตที่ 4 ของ Dip switch ตัวที่ 1 นี้.

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/4 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก ดังโปรแกรมที่ 9.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน

5. สังเกตผลที่ได้รับจากการใช้โวลท์มิเตอร์วัดที่จุด เอาต์พุตของสัญญาณนาฬิกาที่ Connector ที่ 1

โปรแกรมที่ 9.1 การแปลงสัญญาณดิจิตอลเป็นสัญญาณรอนาล็อก

```
0000      1      ORG      0000H
0000 748A      2      MOV      A,#8AH      ;INITIAL 8255
0002 908003      3      MOV      DPTR,#8003H
0005 F0         4      MOVX     @DPTR,A
0006 74FF      5      MOV      A,#0FFH      ;DATA SEND TO D/A
0008 90FC00      6      MOV      DPTR,#0FC00H
000B F0         7      MOVX     @DPTR,A
000C 80FE      8      JMP      S
0000=          9      END
```

6. จากโปรแกรมที่ 9.1 จงเปลี่ยนค่าข้อมูลที่ตำแหน่ง Address 0007h ด้วยค่าต่างๆดังตารางข้างล่างนี้ แล้วบันทึกผลที่ได้รับ

ปริมาณดิจิตอล	แรงดันที่วัดได้ (V)
00H	
19H	
32H	
4BH	
64H	
7DH	
96H	
AFH	
C8H	
E1H	
FFH	

ตารางที่ 6.9 ตารางบันทึกผลของการทดลอง Stepping Motor

7. ให้นำเอาหลอดไฟต่อที่ Connector ที่ 1 แล้วจงเขียนโปรแกรมควบคุมความสว่างของหลอดไฟนั้นโดยให้สว่างมากแล้วค่อยๆสว่างน้อยลง

8. ให้เขียนโปรแกรมแปลงสัญญาณดิจิทัลเป็นสัญญาณรูปฟีนเลื้อย เพื่อผลที่ได้ให้ใช้ออสซิลอสโคปวัด
9. ให้เขียนโปรแกรมแปลงสัญญาณดิจิทัลเป็นสัญญาณรูปสี่เหลี่ยม เพื่อผลที่ได้ให้ใช้ออสซิลอสโคปวัด



6.10 การทดลองที่ 10 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล

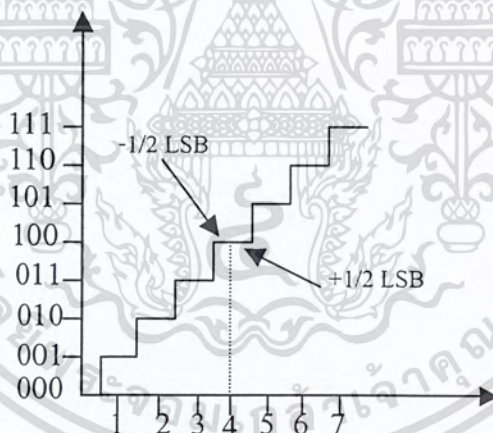
■ วัตถุประสงค์

1. เพื่อเป็นการศึกษาวงจรการแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล

■ ทฤษฎี

การแปลงสัญญาณอนาล็อกสัญญาณที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง เช่นระดับแรงดันไฟหรือกระแส ให้กลายเป็นสัญญาณดิจิทัล ที่อยู่ในรูปฐานสอง “0” กับ “1” ในการแปลงสัญญาณอนาล็อกให้เป็นดิจิทัลนั้นจะต้องคำนึงถึงความละเอียดของ A/D ซึ่งความละเอียดนี้จะขึ้นอยู่กับจำนวนบิตของเอาต์พุต ยังมีจำนวนบิตของเอาต์พุตมากเท่าไร ความละเอียดของ A/D ก็ยิ่งมากขึ้น ทั้งนี้เพราะว่าแรงดันอินพุตที่ป้อนเข้ามาถ้าหากมีการเปลี่ยนแปลงอยู่ในช่วง $+1/2$ LSB ถึง $-1/2$ LSB จะไม่มีผลต่อเอาต์พุตของ A/D เลข (LSB บิตที่มีความสำคัญต่ำสุด Least Significant Bit) (MSB บิตที่มีความสำคัญสูงสุด Most Significant Bit)

ตัวอย่างการแปลงสัญญาณไฟฟ้าให้เป็นสัญญาณดิจิทัลขนาด 3 บิตเอาต์พุตจะประกอบด้วย D0,D1 และ D2 ในทางอุดมคติของการแปลงสัญญาณอนาล็อกให้เป็นดิจิทัล ขนาด 3 บิตจะให้ 1LSB เทียบเท่าแรงดัน 1 โวลต์ ดังนั้นเมื่อป้อนอินพุตเป็น 0,1,2 เอาต์พุตจะเป็น 000 001 010 แต่ถ้าหากอินพุตที่เข้ามาหากเป็น 1.25 โวลต์ เอาต์พุตที่ได้ก็ยังคงเป็น 001 เพราะแรงดันเอาต์พุตที่ป้อนเข้ามามีค่าอยู่ในช่วง $+1/2$ LSB ถึง $-1/2$ LSB จะไม่มีผลต่อเอาต์พุต



รูปที่ 6.6 แสดงความไม่ละเอียดของการแปลงสัญญาณของ A/D

เนื่องจาก A/D ไม่สามารถแยกความแตกต่างได้ถ้าแรงดันอินพุตมีค่าความแตกต่างของแรงดันน้อยกว่า 1 LSB แต่เพื่อที่จะทำให้ A/D แยกความแตกต่างของแรงดันอินพุตได้ดียิ่งขึ้นจะต้องทำให้ 1 LSB มีค่าน้อยที่สุด โดยการเพิ่มจำนวนบิตเอาต์พุตของ A/D ให้มากขึ้น เราสามารถคำนวณขนาดของ LSB ได้จากสมการ

แรงดันสูงสุดของอิพุด

1 LSB =
$$\frac{\text{ (จำนวนบิตเอาต์พุต) } - 1}{\text{ (จำนวนบิตเอาต์พุต) } - 1}$$

ในบอร์ดทดลองอินพุตเอาต์พุตโมดูล ADC เราจะใช้ไอซี ADC 0804 เป็นตัวที่ทำหน้าที่ในการแปลงอนาล็อกให้เป็นสัญญาณดิจิทัลซึ่งไอซีเบอร์นี้จะเป็นไอซีที่แปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิทัลขนาด 8 บิต ซึ่ง 8 บิตสามารถที่จะแปลงให้มีความละเอียดได้ 256 ค่า โดยที่แรงดันอยู่ในช่วง 0-2.55 โวลต์ จะมีค่า 1 LSB = 10 มิลลิโวลต์ ในโมดูลนี้เราสามารถที่จะเลือกเอาสัญญาณอนาล็อกมาจากหลายส่วนได้โดยการ ON Dip Switch ที่ 3 เช่นบิตที่ 1 จะเป็น Potentiometer (POT), บิตที่ 2 จะเป็น Temperature และบิตที่ 3 จะเป็น External Analog ส่วนโมดูลนี้ได้ต่อกับพอร์ต PC ของ 8255.

■ ขั้นตอนของการทดลอง

- 1. ต่อสายบัดเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051และ บอร์ดทดลองอินพุตเอาต์พุต
- 2. ใช้ Supply ต่อเข้าที่ Connector 1 ปรับไฟให้ได้ 1 โวลต์
- 3. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
- 4. ปรับ สวิตช์ 3/3 ไปที่ตำแหน่ง ON
- 5. ปรับ สวิตช์ 1/1 ไปที่ตำแหน่ง ON
- 6. เขียนโปรแกรมด้วยภาษา แอสเซมบลี51 (Assembly 51) อ่านค่าสัญญาณอนาล็อกเข้ามาแปลงสัญญาณอนาล็อกนั้นให้เป็นสัญญาณดิจิทัลแล้วส่งไปแสดงผลที่ Port Monitor ดังโปรแกรมที่ 10.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
- 7. ดาวน์โหลด(download)โปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน
- 8. สังเกตผลที่ได้รับที่ Port Monitor

โปรแกรมที่ 10.1 การแปลงสัญญาณอนาล็อกเป็นดิจิทัล

0000	1	ORG	0000H	
0000 7489	2	MOV	A,#89H	;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVB	@DPTR,A	;READ ADC
0006 908002	5	MOV	DPTR,#8002H	
0009 E0	6	MOVB	A,@DPTR	
000A 908000	7	MOV	DPTR,#8000H	;DISPLAY OUTPUT
000D F0	8	MOVB	@DPTR,A	
000E 80FE	9	JMP	S	
0000=	10	END		

9. จากโปรแกรมที่ 10.1 จงเปลี่ยนค่าข้อมูลที่ป้อนให้ด้วยค่าต่างดังตารางข้างล่างนี้ แล้วบันทึกผลที่ได้รับ

แรงดันอินพุต A/D	PORT MONITOR	ข้อมูล
0V	○○○○○○○○○○	00H
0.5V	○○○○○○○○○○	
1 โวลต์	○○○○○○○○○○	
1.5V	○○○○○○○○○○	
2V	○○○○○○○○○○	
2.55 โวลต์	○○○○○○○○○○	

ตารางที่ 6.10 ตารางบันทึกผลของการทดลองของ ADC

10. ให้เขียนโปรแกรมอ่านค่าเข้ามาจาก POT แปลงให้เป็นสัญญาณดิจิตอลแล้วส่งไปแสดงผลที่ Port Monitor ก่อนการทดลองให้ปรับ สวิตช์ 3/3, สวิตช์ 1/11 และ สวิตช์ 1/1 ไปที่ตำแหน่ง ON ก่อน แล้วสังเกตผลที่ได้รับที่ Port Monitor ในเวลาที่เรารับ POT ขึ้นและลง
11. อธิบายการติด คับของหลอด LED หมายถึงอะไร

6.11 การทดลองที่ 11 การควบคุมการทำงานของตัวทำความร้อน(HEATER) และ การควบคุมการทำงานของอุปกรณ์ตรวจวัดอุณหภูมิ(Temperature sensor)

6.11.1 การควบคุมการทำงานของตัวทำความร้อน(HEATER)

■ วัตถุประสงค์

1. เพื่อศึกษาการควบคุม การทำงานของ HEATER โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุม

■ ทฤษฎี

Heater เป็นอุปกรณ์ที่ให้ความร้อนออกมา ในที่นี่เราจะใช้ตัวต้านทาน 68 โอห์ม 6 วัตต์เป็น Heater โดยโมดูล Heater นี้จะต่อกับพอร์ต PB บิตที่ 6 ของ 8255 แต่เนื่องจากว่าพอร์ต PB ได้ต่อกับหลายโมดูลดังนั้นเราจะใช้ บัฟเฟอร์ 74LS244 เป็นตัวกันเพื่อไม่ให้ไปรบกวนโมดูลอื่นๆ สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Heater ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 6 ของ Dip switch ตัวที่ 1 นี้

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/6 ไปที่ตำแหน่ง ON
3. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ Heater ทำงาน ดังโปรแกรมที่ 11.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
4. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 11.1 สั่งให้ Heater ทำงาน (ระวังความร้อนจากตัว Heater ถ้าเปิดไว้นานๆ)

0000	1	ORG	0000H	
0000 7480	2	MOV	A,#80H	;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVX	@DPTR,A	
0006 7440	5	MOV	A,#40H	;ON HEATOR
0008 908001	6	MOV	DPTR,#80001H	
000B F0	7	MOVX	@DPTR,A	
000C 80FE	8	JMP	S	
0000=	9	END		

5. จงเขียนโปรแกรมสั่งให้ Heater ยุคการทำงาน
6. ให้เขียนโปรแกรมอ่านค่ามาจาก Dip Switch ตัวที่ 2 บิตที่ 1 มาเป็นการควบคุมการทำงานของ Heater



6.11.2 การควบคุมการทำงานของอุปกรณ์ตรวจวัดอุณหภูมิ(Temperature sensor)

■ วัตถุประสงค์

1. เพื่อศึกษาการใช้ไมโครคอนโทรลเลอร์อ่านค่าอุณหภูมิจากตัววัดอุณหภูมิ
2. เพื่อศึกษาการใช้ไมโครคอนโทรลเลอร์ในการควบคุมอุณหภูมิ

■ ทฤษฎี

ตัววัดอุณหภูมิที่ใช้คือ LM335 ซึ่งมีการทำงานคล้ายกับ Zener diode จะมีจุด Break down voltage ที่มีค่าสัดส่วนกับค่าสัมบูรณ์ของอุณหภูมิ ตัว LM335 จะมีอัตราการเปลี่ยนแปลงของแรงดันเมื่ออุณหภูมิเปลี่ยนไป เท่ากับ 10 มิลลิโวลต์/ องศาเซลวิน

LM335 จะถูกวางไว้ใกล้ๆ กับตัวทำความร้อน Heater เพื่อให้สามารถที่จะทำงานทดลองแบบ Close loop ได้ เมื่ออุณหภูมิเปลี่ยนแปลงในช่วง 0 – 100 องศาเซลเซียส แรงดันที่ตกคร่อม LM335 จะอยู่ระหว่าง 2.73 โวลต์ – 3.73 โวลต์ เมื่ออุณหภูมิเพิ่มขึ้นแรงดันก็เพิ่มขึ้นด้วยอัตราส่วนที่คงที่ แรงดันที่ได้ก็จะส่งไปเป็นอินพุตของออปแอมป์ 741 ซึ่งต่อเป็นวงจรคิฟเฟอร์แมนเชี่ยล ทำให้ได้เอาต์พุต V_o เป็นแรงดันที่เปลี่ยนแปลงอยู่ในช่วง 0 – 2.55 โวลต์ ซึ่งแรงดัน V_o นี้จะต่อเข้ากับวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล เพื่อแปลงแรงดัน 0-2.55 V ให้เป็นสัญญาณดิจิตอลขนาด 8 บิต

การคำนวณหาอุณหภูมิ

$$\begin{aligned}
 V_z &= 10 \text{ mV} / ^\circ\text{K} \times T \\
 0^\circ\text{C} &= 273^\circ\text{K} \\
 V_z &= 2.73 + 10 \text{ mV} / ^\circ\text{K} \times T \\
 \text{เมื่อ } V_z &= \text{แรงดันที่ตกคร่อม LM335} \\
 T &= \text{อุณหภูมิที่มีหน่วยเป็น เซลเซียส } (^\circ\text{C})
 \end{aligned}$$

■ ขั้นตอนของการทดลอง

1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
2. ปรับ สวิตช์ 1/11 ไปที่ตำแหน่ง ON
3. ปรับ สวิตช์ 3/2 ไปที่ตำแหน่ง ON
4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ อ่านค่าอุณหภูมิเข้ามาแล้วแสดงผล ดังโปรแกรมที่ 11.2 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
5. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงานแล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 11.2 สั่งให้อ่านค่าอุณหภูมิเข้ามาแล้วแสดงผล

0000	1	ORG	0000H	
0000 7489	2	MOV	A,#89h	;Init 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVB	@DPTR,A	
0006 908000	5	MOV	DPTR,#8000H	;Off port monitor
0009 7400	6	MOV	A,#00H	
000B F0	7	MOVB	@DPTR,A	
000C 7440	8	MOV	A,#40H	;On heate
000E 908001	9	MOV	DPTR,#8001H	
0011 F0	10	MOVB	@DPTR,A	
0012 908002	11	ST: MOV	DPTR,#8002H	;Read temperature
0015 E0	12	MOVB	A,@DPTR	
0016 908000	13	MOV	DPTR,#8000H	;Diplay
0019 F0	14	MOVB	@DPTR,A	
001A 80F6	15	JMP	ST	
0000=	16	END		

6. ให้เขียนโปรแกรมควบคุมอุณหภูมิโดยการตั้งค่าอุณหภูมิไว้ค่าใดค่าหนึ่งแล้วอ่านค่าอุณหภูมิ
จึงเข้ามาเปรียบเทียบกับค่าที่ตั้งไว้ถ้าอุณหภูมิที่ได้จากการอ่านเข้ามานั้นสูงกว่าค่าที่ตั้งไว้ให้
พัดลมที่อยู่กับมอเตอร์ทำงานระบายความร้อนจนกว่าอุณหภูมิต่ำลงน้อยกว่าค่าที่ตั้งไว้หรือเท่า
กับค่าที่ตั้งไว้ จึงให้มอเตอร์หยุดทำงาน (ในการตั้งค่าอุณหภูมิไว้ นั้นต้องตั้งให้มีค่าสูงกว่าค่า
อุณหภูมิของห้องที่ทำการทดลอง)

6.12 การทดลองที่ 12 การวัดค่าความเร็วของมอเตอร์ (Speed Sensor)

■ วัตถุประสงค์

1. เพื่อศึกษาการวัดความเร็วรอบด้วยอินฟราเรด
2. เพื่อศึกษาการทำงานของ Speed sensor

■ ทฤษฎี

การตรวจจับความเร็วของมอเตอร์ จะอาศัยใบพัดของมอเตอร์ที่หมุนตัดผ่านแสงอินฟราเรด ที่ส่องมายังตัวโฟโต้ทรานซิสเตอร์ เมื่อลำแสงอินฟราเรดถูกบังโดยใบพัดของมอเตอร์ วงจรตรวจจับความเร็วก็จะให้เอาต์พุตออกมาเป็น 0 เมื่อใบพัดเคลื่อนผ่านไป โฟโต้ทรานซิสเตอร์ก็สามารถรับแสงจากหลอดอินฟราเรดได้วงจรก็จะให้เอาต์พุตออกมาเป็น 1 โดยที่หลอดอินฟราเรด MLED71 จะส่งแสงออกมาอยู่ตลอดเวลา และทำให้ได้แรงดันที่เอาต์พุตออกมาเป็น 5 โวลต์ โดยที่โมดูลนี้จะต่ออยู่กับพอร์ต PC ของ 8255 แต่เนื่องจากว่าพอร์ต PC นี้ได้ต่ออยู่กับหลายโมดูลดังนั้นเราจะใช้ บัฟเฟอร์ 74LS244 เป็นตัวกันเพื่อไม่ให้ไปรบกวนโมดูลอื่นๆ สำหรับขา เลือก ของ 74LS244 นั้นจะต่อเข้ากับ Dip Switch ตัวที่ 1 โดยเราจะใช้ Dip switch เป็นตัว เลือก ว่าจะให้โมดูลไหนทำงาน สำหรับโมดูล Heater ขา เลือก ของ 74LS244 จะต่ออยู่กับ บิตที่ 10 ของ Dip switch ตัวที่ 1 นี้

■ ขั้นตอนของการทดลอง

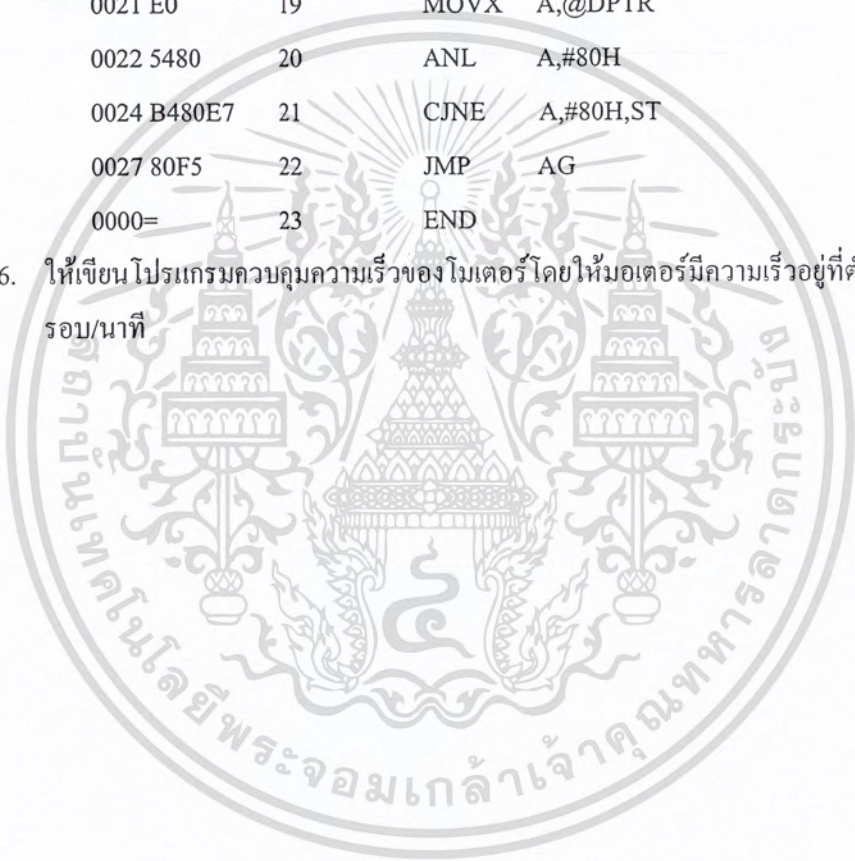
1. ต่อสายบัสเข้าระหว่าง บอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุตเอาต์พุต
3. ปรับ สวิตช์ 1/10 ไปที่ตำแหน่ง ON
4. เขียนโปรแกรมด้วยภาษา แอสเซมบลี 51 (Assembly 51) สั่งให้ DC motor ทำงานอ่านค่าความเร็วเข้ามาเพื่อแสดงผล ดังโปรแกรมที่ 12.1 เมื่อเขียนเสร็จ แล้ว Compile โดย SXA51 ASSEMBLER ให้เป็น File.Hex
5. คำนวณโหลดโปรแกรมที่เป็น File.Hex ลงในบอร์ดให้โปรแกรมทำงาน แล้วสังเกตผลที่ได้รับ

โปรแกรมที่ 12.1 อ่านค่าจาก Speed sensor แสดงผลที่ Port monitor

0000	1	ORG	0000H
0000 7800	2	MOV	R0,#00H
0002 7489	3	MOV	A,#89H
0004 908003	4	MOV	DPTR,#8003H
0007 F0	5	MOVB	@DPTR,A
0008 908000	6	MOV	DPTR,#8000H
000B 7400	7	MOV	A,#00H
000D F0	8	MOVB	@DPTR,A
000E 908002	9	ST: MOV	DPTR,#8002H

0011 E0	10	MOVX	A,@DPTR
0012 5480	11	ANL	A,#80H
0014 B480F7	12	CJNE	A,#80H,ST
0017 E8	13	MOV	A,R0
0018 04	14	INC	A
0019 908000	15	MOV	DPTR,#8000H
001C F0	16	MOVX	@DPTR,A
001D F8	17	MOV	R0,A
001E 908002	18	AG: MOV	DPTR,#8002H
0021 E0	19	MOVX	A,@DPTR
0022 5480	20	ANL	A,#80H
0024 B480E7	21	CJNE	A,#80H,ST
0027 80F5	22	JMP	AG
0000=	23	END	

6. ให้เขียนโปรแกรมควบคุมความเร็วของมอเตอร์โดยให้มอเตอร์มีความเร็วอยู่ที่ตำแหน่ง150 รอบ/นาที



คำตอบของการทดลอง

6.13 การทดลองที่ 1 การโปรแกรม 8255 และการแสดงผลข้อมูลขนาด 8 บิต

6.13.1 การแสดงผลข้อมูลขนาด 8 บิต

5. จากโปรแกรมที่ 2 ทดลองเปลี่ยนF ไลต์ที่ตำแหน่ง Address 0009 จากค่า 0fh เป็น 0FFH, 0AAH, 0FEH, 01H แล้วบันทึกผลที่ได้รับในตาราง

รหัส	แสดงผลบิตที่							
	7	6	5	4	3	2	1	0
0FFH	●	●	●	●	●	●	●	●
0AAH	●	○	●	○	●	○	●	○
0FEH	●	●	●	●	●	●	●	○
01H	○	○	○	○	○	○	○	●

ตารางที่ 6.11 ตารางบันทึกผลการทดลอง Port monitor (LED)

6. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.2

```

0000          1          ORG    0000H
0000 908003    2          MOV    DPTR,#8003H    ;INITIAL 8255
0003 7480      3          MOV    A,#80H
0005 F0        4          MOVX   @DPTR,A
0006 908000    5          MOV    DPTR,#8000H    ;SELECT PA
0009 7401      6  START:   MOV    A,#01H
000B F0        7  ST:      MOVX   @DPTR,A    ;DISPLAY DATA
000C 12001F    8          CALL   DELAY
000F D3        9          SETB   C
0010 33        10         RLC     A
0011 B4FFF7    11         CJNE   A,#0FFH,ST
0014 F0        12  ST1:    MOVX   @DPTR,A
0015 12001F    13         CALL   DELAY
0018 C3        14         CLR     C
0019 13        15         RRC     A
001A B400F7    16         CJNE   A,#00H,ST1
001D 80EA      17         JMP     START
  
```

001F 7800	18	DELAY:	MOV R0,#00H ;DELAY
0021 7900	19	LOOP:	MOV R1,#00H
0023 D9FE	20		DJNZ R1,\$
0025 D8FA	21		DJNZ R0,LOOP
0027 22	22		RET
0000=	23		END

7. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.3

0000	1		ORG 0000H
0000 908003	2		MOV DPTR,#8003H ;INITIAL 8255
0003 7480	3		MOV A,#80H
0005 F0	4		MOVX @DPTR,A
0006 7A01	5	START:	MOV R2,#01H
0008 7B80	6		MOV R3,#80H
000A EA	7	AGAIN:	MOV A,R2
000B 4B	8		ORL A,R3
000C 908000	9		MOV DPTR,#8000H ;SELECT PA
000F F0	10		MOVX @DPTR,A ;DISPLAY DATA
0010 12001E	11		CALL DELAY
0013 EA	12		MOV A,R2
0014 23	13		RL A
0015 FA	14		MOV R2,A
0016 EB	15		MOV A,R3
0017 03	16		RR A
0018 FB	17		MOV R3,A
0019 BB00EE	18		CJNE R3,#00H,AGAIN
001C 80E8	19		JMP START
001E 7800	20	DELAY:	MOV R0,#00H ;DELAY
0020 7900	21	LOOP:	MOV R1,#00H
0022 D9FE	22		DJNZ R1,\$
0024 D8FA	23		DJNZ R0,LOOP
0026 22	24		RET
0000=	25		END

8. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.4

0000	1	ORG	0000H
0000 908003	2	MOV	DPTR,#8003H ;INITIAL 8255
0003 7480	3	MOV	A,#80H
0005 F0	4	MOVX	@DPTR,A
0006 7AFE	5	START:	MOV R2,#0FEH
0008 7B7F	6	MOV	R3,#7FH
000A EA	7	AGAIN:	MOV A,R2
000B 5B	8	ANL	A,R3
000C 908000	9	MOV	DPTR,#8000H ;SELECT PA
000F F0	10	MOVX	@DPTR,A ;DISPLAY DATA
0010 12001E	11	CALL	DELAY
0013 EA	12	MOV	A,R2
0014 23	13	RL	A
0015 FA	14	MOV	R2,A
0016 EB	15	MOV	A,R3
0017 03	16	RR	A
0018 FB	17	MOV	R3,A
0019 BB7FEE	18	CJNE	R3,#7FH,AGAIN
001C 80E8	19	JMP	START
001E 7800	20	DELAY:	MOV R0,#00H ;DELAY
0020 7900	21	LOOP:	MOV R1,#00H
0022 D9FE	22	DJNZ	R1,\$
0024 D8FA	23	DJNZ	R0,LOOP
0026 22	24	RET	
0000=	25	END	

9. ให้เขียนโปรแกรมสร้างไฟวิ่งดังรูปที่ 6.5

0000	1	ORG	0000H
0000 908003	2	MOV	DPTR,#8003H ;INITIAL 8255
0003 7480	3	MOV	A,#80H
0005 F0	4	MOVX	@DPTR,A
0006 7401	5	START:	MOV A,#01H
0008 120019	6	CALL	DISPLAY

000B 23	7 AGAIN:	RL	A
000C 23	8	RL	A
000D 120019	9	CALL	DISPLAY
0010 03	10	RR	A
0011 120019	11	CALL	DISPLAY
0014 B401F4	12	CJNE	A,#01H,AGAIN
0017 80ED	13	JMP	START
0019 908000	14 DISPLAY:	MOV	DPTR,#8000H ;DISPLAY DATA
001C F0	15	MOVX	@DPTR,A
001D 7800	16	MOV	R0,#00H ;DELAY
001F 7900	17 LOOP:	MOV	R1,#00H
0021 D9FE	18	DJNZ	R1,S
0023 D8FA	19	DJNZ	R0,LOOP
0025 22	20	RET	
0000=	21	END	



6.14 การทดลองที่2 การอ่านข้อมูลขนาด 8 บิตจากสวิตช์อินพุต และการควบคุม BUZZER

6.14.1 การอ่านข้อมูลขนาด 8 บิต

6. จากโปรแกรมไฟว์ทั้งหมดที่ได้สร้างไว้ก่อนหน้านี้ ให้เขียนโปรแกรมอ่านค่ามาจากสวิตช์อินพุตแล้วส่งไฟว์ตามแบบต่างๆดังต่อไปนี้

- เมื่อกดสวิตช์อินพุต 2/1 ส่งให้ไฟว์แบบที่ 1
- เมื่อกดสวิตช์อินพุต 2/2 ส่งให้ไฟว์แบบที่ 2
- เมื่อกดสวิตช์อินพุต 2/3 ส่งให้ไฟว์แบบที่ 3
- เมื่อกดสวิตช์อินพุต 2/4 ส่งให้ไฟว์แบบที่ 4
- เมื่อกดสวิตช์ไหน ก็ตามนอกเหนือจากข้างบนให้ไฟว์ดังโปรแกรมที่ 2.1

```

0000          1          ORG  0000H
0000 908003    2          MOV  DPTR,#8003H ;init 8255
0003 7489      3          MOV  A,#89H
0005 F0        4          MOVX  @DPTR,A
0006 908002    5 START:  MOV  DPTR,#8002H ;READ DIP SW
0009 E0        6          MOVX  A,@DPTR
000A F520      7          MOV  20H,A
000C 300105    8          JNB   01H,BB
000F 120031    9          CALL  LED1
0012 80F2     10          JMP   START
0014 300205   11 BB:     JNB   02H,CC
0017 120049   12          CALL  LED2
001A 80EA     13          JMP   START
001C 300305   14 CC:     JNB   03H,DD
001F 120060   15          CALL  LED3
0022 80E2     16          JMP   START
0024 300405   17 DD:     JNB   04H,EE
0027 120077   18          CALL  LED4
002A 80DA     19          JMP   START
002C 120091   20 EE:     CALL  ANOTHER
002F 80D5     21          JMP   START

```

=====ไฟวิ่งแบบที่ 1=====

0031 908000	23	LED1:	MOV DPTR,#8000H ;SELECT PA
0034 7401	24		MOV A,#01H
0036 F0	25	ST:	MOVX @DPTR,A ;DISPLAY DATA
0037 1200A7	26		CALL DELAY
003A D3	27		SETB C
003B 33	28		RLC A
003C B4FFF7	29		CJNE A,#0FFH,ST
003F F0	30	ST1:	MOVX @DPTR,A
0040 1200A7	31		CALL DELAY
0043 C3	32		CLR C
0044 13	33		RRC A
0045 B400F7	34		CJNE A,#00H,ST1
0048 22	35		RET

=====ไฟวิ่งแบบที่ 2=====

0049 7A01	37	LED2:	MOV R2,#01H
004B 7B80	38		MOV R3,#80H
004D EA	39	AGAIN:	MOV A,R2
004E 4B	40		ORL A,R3
004F 908000	41		MOV DPTR,#8000H ;SELECT PA
0052 F0	42		MOVX @DPTR,A ;DISPLAY DATA
0053 1200A7	43		CALL DELAY
0056 EA	44		MOV A,R2
0057 23	45		RL A
0058 FA	46		MOV R2,A
0059 EB	47		MOV A,R3
005A 03	48		RR A
005B FB	49		MOV R3,A
005C BB00EE	50		CJNE R3,#00H,AGAIN
005F 22	51		RET

=====ไฟวิ่งแบบที่ 2=====

0060 7AFE	54 LED3:	MOV R2,#0FEH
0062 7B7F	55	MOV R3,#7FH
0064 EA	56 AGAIN1:	MOV A,R2
0065 5B	57	ANL A,R3
0066 908000	58	MOV DPTR,#8000H ;SELECT PA
0069 F0	59	MOVX @DPTR,A ;DISPLAY DATA
006A 1200A7	60	CALL DELAY
006D EA	61	MOV A,R2
006E 23	62	RL A
006F FA	63	MOV R2,A
0070 EB	64	MOV A,R3
0071 03	65	RR A
0072 FB	66	MOV R3,A
0073 BB7FEE	67	CJNE R3,#7FH,AGAIN1
0076 22	68	RET

=====ไฟวิ่งแบบที่ 3=====

0077 7401	70 LED4:	MOV A,#01H
0079 120089	71	CALL SHOW
007C 23	72 AGAIN2:	RL A
007D 23	73	RL A
007E 120089	74	CALL SHOW
0081 03	75	RR A
0082 120089	76	CALL SHOW
0085 B401F4	77	CJNE A,#01H,AGAIN2
0088 22	78	RET
0089 908000	79 SHOW:	MOV DPTR,#8000H ;DISPLAY DATA
008C F0	80	MOVX @DPTR,A
008D 1200A7	81	CALL DELAY
0090 22	82	RET

```

=====ไฟวิ่ง 4=====
0091 908000      85 ANOTHER: MOV  DPTR,#8000H  ;SELECT P.A
0094 7401        86          MOV  A,#01H      ;DATA
0096 F0          87 ST2: MOVX  @DPTR,A
0097 23          88          RL   A          ;ROTAGE DATA
0098 1200A7      89          CALL DELAY ;DELAY TIME
009B B480F8      90          CJNE  A,#80H,ST2    ;CHECK & ROTAE
009E F0          91 ST3:   MOVX  @DPTR,A
009F 03          92          RR   A
00A0 1200A7      93          CALL DELAY
00A3 B401F8      94          CJNE  A,#01H,ST3
00A6 22          95          RET

```

```

=====การหน่วงเวลา=====
00A7 7900      98 DELAY:  MOV  R1,#00H
00A9 7A00      99 LOOP:   MOV  R2,#00H
00AB DAFE     100          DJNZ  R2,$
00AD D9FA     101          DJNZ  R1,LOOP
00AF 22       102          RET
0000=         103          END

```

6.14.2 การทดลองเกี่ยวกับ Buzzer

4. ให้เขียนโปรแกรมกันขโมย โดยใช้ Dip switch เป็น Sensor ในการตรวจจับขโมย ถ้าเมื่อใดก็ตามที่มีการกด Dip switch ไม่ว่าจะ เป็น Dip switch บิตที่เท่าใดก็ตามให้ Buzzer ส่งเสียงเตือนคือดังแล้วเงียบ ดังแล้วเงียบ.

```

0000          1          ORG  0000H
0000 908003    2          MOV  DPTR,#8003H  ;INITAIL 8255
0003 7489      3          MOV  A,#89H
0005 F0        4          MOVX  @DPTR,A
0006 908001    5          MOV  DPTR,#8001H  ;READ DIP SW
0009 E0        6 START:  MOVX  A,@DPTR
000A B40002    7          CJNE  A,#00H,WORK

```


000D 80FA	8	JMP	START	
000F 908001	9	WORK:	MOV	DPTR,#8001H
0012 7480	10	MOV	A,#80H	
0014 F0	11	MOVX	@DPTR,A	;BUZZER ON
0015 120020	12	CALL	DELAY	
0018 7400	13	MOV	A,#00H	
001A F0	14	MOVX	@DPTR,A	;BUZZER OFF
001B 120020	15	CALL	DELAY	
001E 80EF	16	JMP	WORK	
0020 7800	17	DELAY:	MOV	R0,#00H ;DELAY TIME
0022 7900	18	LOOP:	MOV	R1,#00H
0024 D9FE	19	DJNZ	R1,S	
0026 D8FA	20	DJNZ	R0,LOOP	
0028 22	21	RET		
0000=	22	END		



6.15 การทดลองที่ 3 การควบคุมการทำงานของ Magnetic Relay และ Solid State Relay

6.15.1 การควบคุมการทำงานของ Magnetic Relay

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง 000A H เป็นค่า 20H, 40H, 80H, 0FF และ 00H แล้วบันทึกผลลัพธ์ลงในตารางข้างล่างนี้

รหัสที่ส่งออกไป	การทำงานของ Magnetic Relay
20H	ตัวที่ 2 ทำงาน
40H	ตัวที่ 3 ทำงาน
80H	ตัวที่ 4 ทำงาน
0FFH	ทำงานทุกตัว
00H	ไม่มีตัวไหนทำงาน

ตารางที่ 6.12 ตารางบันทึกผลการทดลองของ Solid State Relay

6. ให้เขียนโปรแกรมสั่งให้ Magnetic Relay ได้ทำงานทั้ง 4 ตัว โดยที่สั่งให้ติดครั้งละ 2 ตัวนานประมาณ 30 วินาที แล้ว 2 ตัวที่เหลือก็ติดนานประมาณ 1 นาที

```
0000 1 ORG 0000H
0000 7480 2 MOV A,#80H
0002 908003 3 MOV DPTR,#8003H
0005 F0 4 MOVX @DPTR,A
0006 7430 5 MOV A,#30H
0008 908001 6 MOV DPTR,#8001H
000B F0 7 ST: MOVX @DPTR,A
000C 23 8 RL A
000D 23 9 RL A
000E 120017 10 CALL DELAY1
0011 F0 11 MOVX @DPTR,A
0012 120024 12 CALL DELAY2
0015 80F4 13 JMP ST
0017 7B04 14 DELAY1: MOV R3,#04H
0019 7800 15 LOOP1: MOV R0,#00H
001B 7900 16 LOOP: MOV R1,#00H
001D D9FE 17 DJNZ R1,$
001F D8FA 18 DJNZ R0,LOOP
```


0021 DBF6	19	DJNZ R3,LOOP1
0023 22	20	RET
0024 7B08	21 DELAY2:	MOV R3,#08H
0026 7800	22 LOOP3:	MOV R0,#00H
0028 7900	23 LOOP2:	MOV R1,#00H
002A D9FE	24	DJNZ R1, \$
002C D8FA	25	DJNZ R0,LOOP2
002E DBF6	26	DJNZ R3,LOOP3
0030 22	27	RET
0000=	28	END

7. ให้เขียนโปรแกรม สั่งให้ Magnetic Relay ทำงานเหมือนไฟวิ่งโดยทำงานทีละตัวและการทำงานของ Magnetic Relay แต่ละตัวนี้นานประมาณ 30 วินาที

0000	1	ORG 0000H
0000 7480	2	MOV A,#80H
0002 908003	3	MOV DPTR,#8003H
0005 F0	4	MOVX @DPTR,A
0006 7411	5	MOV A,#11H
0008 908001	6	MOV DPTR,#8001H
000B F0	7	ST: MOVX @DPTR,A
000C 23	8	RL A
000D 120012	9	CALL DELAY1
0010 80F9	10	JMP ST
0012 7B04	11 DELAY1:	MOV R3,#04H
0014 7800	12 LOOP1:	MOV R0,#00H
0016 7900	13 LOOP:	MOV R1,#00H
0018 D9FE	14	DJNZ R1,\$
001A D8FA	15	DJNZ R0,LOOP
001C DBF6	16	DJNZ R3,LOOP1
001E 22	17	RET
0000=	18	END

6.15.2 การควบคุมการทำงานของ Solid State Relay

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง 000A H เป็นค่า 02H, 04H, 08H, 0FF และ 00H แล้วบันทึกผลลัพธ์ลงในตารางข้างล่างนี้

รหัสที่ส่งออกไป	การทำงานของ Solid State Relay
02H	ตัวที่ 2 ทำงาน
04H	ตัวที่ 3 ทำงาน
08H	ตัวที่ 4 ทำงาน
0FFH	ทำงานทุกตัว
00H	ไม่มีตัวไหนทำงาน

ตารางที่ 6.13 ตารางบันทึกผลการทดลองของ Solid State Relay

6. ให้เขียนโปรแกรมสั่งให้ Solid State Relay ทำงานเหมือนไฟวิ่งโดยทำงานทีละตัวเริ่มจากตัวที่ 1 แล้วตัวที่ 3 แล้วกลับมาที่ตัวที่ 2 แล้วไปที่ตัวที่ 4 และการทำงานของ Solid State Relay แต่ละตัวนั้นนานประมาณ 30 วินาที

```
0000 1 ORG 0000H
0000 7480 2 MOV A,#80H
0002 908003 3 MOV DPTR,#8003H
0005 F0 4 MOVX @DPTR,A
0006 7411 5 MOV A,#11H
0008 908001 6 MOV DPTR,#8001H
000B F0 7 ST: MOVX @DPTR,A
000C 23 8 RL A
000D 23 9 RL A
000E 120018 10 CALL DELAY1
0011 F0 11 MOVX @DPTR,A
0012 120018 12 CALL DELAY1
0015 03 13 RR A
0016 80F3 14 JMP ST
0018 7B04 15 DELAY1: MOV R3,#04H
001A 7800 16 LOOP1: MOV R0,#00H
001C 7900 17 LOOP: MOV R1,#00H
001E D9FE 18 DJNZ R1,$
```


0020 D8FA	19	DJNZ R0,LOOP
0022 DBF6	20	DJNZ R3,LOOP1
0024 22	21	RET
0000=	22	END

7. ให้เขียนโปรแกรม สั่งให้ Solid State Relay ทำงานตามการ เปิด, ปิด ของโมดูล Dip Switch บิตที่ 1-4 ของ Dip Switch. ก่อนการทดลองควรปรับ สวิตช์ 1/9 ไปที่ตำแหน่ง ON ก่อน.

0000	1	ORG 0000H
0000 7489	2	MOV A,#89H
0002 908003	3	MOV DPTR,#8003H
0005 F0	4	MOVX @DPTR,A
0006 908002	5	ST: MOV DPTR,#8002H
0009 E0	6	MOVX A,@DPTR
000A 540F	7	ANL A,#0FH
000C 908001	8	MOV DPTR,#8001H
000F F0	9	MOVX @DPTR,A
0010 80F4	10	JMP ST
0000=	11	END

6.16 การทดลองที่ 4 การแสดงผลข้อมูล 7 ส่วน (7 Segment display)

5. จากโปรแกรมที่ 3.1 ให้เปลี่ยนข้อมูลที่ตำแหน่ง 0009H เป็นค่าต่างๆดังตารางข้างล่างนี้ แล้วบันทึกผลที่ได้รับ

รหัส	แสดงผล 7 Segment
6H	1
0CFH	3
0FDH	6
17H	7
0EFH	9

ตารางที่ 6.4 ตารางบันทึกผลการทดลองของ 7 Segment display

6. จากโปรแกรมที่ 3.1 จงสร้างตารางสำหรับรหัสของ 7 Segment ทุกๆตัวตั้งแต่ 0-9 แล้วบันทึกลงในตารางข้างล่างนี้

รหัส	แสดงผล 7 Segment
0BFH	0
86H	1
0DBH	2
0CFH	3
0E6H	4
0EDH	5
0FDH	6
87H	7
0FFH	8
0EFH	9

ตารางที่ 6.15 ตารางบันทึกรหัส 7 Segment display จากการทดลอง

7. จงเขียนโปรแกรมสั่งให้ 7 Segment นับตั้งแต่ 0-9 เริ่มจากติดเป็นเลข 0 ที่หลักที่ 1 และติดเป็นเลข 1 ที่หลักที่ 2 ค่อยๆไปเรื่อยๆ เมื่อถึงหลักที่ 4 แล้วให้เลขตัวต่อไปติดที่หลักที่ 1 และเลขต่อไปติดที่หลักต่อไปเรื่อยๆจนครบถึงเลข 9 แล้วให้กลับมาเริ่มต้นใหม่

0000	1	ORG 0000H
0000 7F01	2	START: MOV R7,#01H
0002 908003	3	MOV DPTR,#8003H
0005 7480	4	MOV A,#80h
0007 F0	5	MOVX @DPTR,A
0008 74BF	6	MOV A,#0BFH
000A 12003F	7	CALL DISPLAY
000D 7486	8	MOV A,#86H
000F 12003F	9	CALL DISPLAY
0012 74DB	10	MOV A,#0DBH
0014 12003F	11	CALL DISPLAY
0017 74CF	12	MOV A,#0CFH
0019 12003F	13	CALL DISPLAY
001C 74E6	14	MOV A,#0E6H
001E 12003F	15	CALL DISPLAY
0021 74ED	16	MOV A,#0EDH
0023 12003F	17	CALL DISPLAY
0026 74FD	18	MOV A,#0FDH
0028 12003F	19	CALL DISPLAY
002B 7487	20	MOV A,#87H
002D 12003F	21	CALL DISPLAY
0030 74FF	22	MOV A,#0FFH
0032 12003F	23	CALL DISPLAY
0035 74EF	24	MOV A,#0EFH
0037 12003F	25	CALL DISPLAY
003A 12005C	26	CALL RESTART
003D 80C1	27	JMP START
003F 908000	28	DISPLAY: MOV DPTR,#8000H
0042 F0	29	MOVX @DPTR,A
0043 EF	30	MOV A,R7
0044 908001	31	MOV DPTR,#8001H
0047 F0	32	MOVX @DPTR,A
0048 23	33	RL A
0049 FF	34	MOV R7,A

004A BF1002	35	CJNE R7,#10H,CONT
004D 7F01	36	MOV R7,#01H
004F 7A03	37	CONT: MOV R2,#03H
0051 7900	38	LOOP1: MOV R1,#00H
0053 7800	39	LOOP: MOV R0,#00H
0055 D8FE	40	DJNZ R0,\$
0057 D9FA	41	DJNZ R1,LOOP
0059 DAF6	42	DJNZ R2,LOOP1
005B 22	43	RET
005C 7A04	44	RESTART: MOV R2,#04H
005E 7900	45	LOOP3: MOV R1,#00H
0060 7800	46	LOOP2: MOV R0,#00H
0062 D8FE	47	DJNZ R0,\$
0064 D9FA	48	DJNZ R1,LOOP2
0066 DAF6	49	DJNZ R2,LOOP3
0068 22	50	RET
0000=	51	END

6.17 การทดลองที่ 5 การรับข้อมูลจากคีย์บอร์ด (Scan Key board)

6. จงเขียนโปรแกรมอ่านค่าจากคีย์บอร์ดแล้วส่งออกไปแสดงผลที่ Port monitor โดยที่ให้แสดงผลเป็นค่าเลขฐาน 2 ในเมื่อคีย์ใดๆ เช่นกดคีย์หมายเลข 1 ให้แสดงผลเป็นค่า 1 ในเลขฐาน 2, กดคีย์หมายเลข 4 ให้แสดงผลเป็นค่า 4 ในเลขฐาน 2

0000	1	ORG	0000H	
0000 7488	2	MOV	A,#88H	;Init 8255.1
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVB	@DPTR,A	
0006 7FFF	5	MOV	R7,#0FFH	
0008 908002	6	START: MOV	DPTR,#8002H	;select port c
000B 7400	7	WAITE: MOV	A,#00H	;for check push key or not
000D F0	8	MOVB	@DPTR,A	
000E E0	9	MOVB	A,@DPTR	
000F 5570	10	ANL	A,70H	
0011 B47005	11	CJNE	A,#70H,SCAN	
0014 1200B4	12	CALL	DISPLAY	
0017 80EF	13	JMP	START	
0019 908002	14	SCAN: MOV	DPTR,#8002H	;for SCAN key
001C 74FE	15	MOV	A,#0FEH	;for check row 1
001E F0	16	MOVB	@DPTR,A	
001F E0	17	MOVB	A,@DPTR	
0020 5470	18	ANL	A,#70H	
0022 F520	19	MOV	20H,A	
0024 200407	20	JB	04H,AA	;Check if not key one
0027 7F01	21	MOV	R7,#01H	;if key 1, get data
0029 1200B4	22	CALL	DISPLAY	
002C 80DA	23	JMP	START	
002E 200507	24	AA: JB	05H,BB	;Check if not key 2
0031 7F02	25	MOV	R7,#02H	;if key 2, get data
0033 1200B4	26	CALL	DISPLAY	
0036 80D0	27	JMP	START	
0038 200607	28	BB: JB	06H,CC	;Check if not key 3
003B 7F03	29	MOV	R7,#03H	;if key 3, get data

003D 1200B4	30	CALL	DISPLAY	
0040 80C6	31	JMP	START	
0042 74FD	32	CC: MOV	A,#0FDH	;for Check row 2
0044 F0	33	MOVSX	@DPTR,A	
0045 E0	34	MOVSX	A,@DPTR	
0046 5470	35	ANL	A,#70H	
0048 F520	36	MOV	20H,A	
004A 200407	37	JB	04H,D	;Check if not key 4
004D 7F04	38	MOV	R7,#04H	;if key 4, get data
004F 1200B4	39	CALL	DISPLAY	
0052 80B4	40	JMP	START	
0054 200507	41	D: JB	05H,E	;Check if not key 5
0057 7F05	42	MOV	R7,#05H	;if key 5, get data
0059 1200B4	43	CALL	DISPLAY	
005C 80AA	44	JMP	START	
005E 200607	45	E: JB	06H,F	;Check if not key 6
0061 7F06	46	MOV	R7,#06H	;if key 6, get data
0063 1200B4	47	CALL	DISPLAY	
0066 80A0	48	JMP	START	
0068 74FB	49	F: MOV	A,#0FBH	;for CCheck row 3
006A F0	50	MOVSX	@DPTR,A	
006B E0	51	MOVSX	A,@DPTR	
006C 5470	52	ANL	A,#70H	
006E F520	53	MOV	20H,A	
0070 200407	54	JB	04H,G	;Check if not key7
0073 7F07	55	MOV	R7,#07H	;if key7 ,get data
0075 1200B4	56	CALL	DISPLAY	
0078 808E	57	JMP	START	
007A 200507	58	G: JB	05H,H	;Check if not key8
007D 7F08	59	MOV	R7,#08H	;if key 8,get data
007F 1200B4	60	CALL	DISPLAY	
0082 8084	61	JMP	START	
0084 200607	62	H: JB	06H,I	;Check if not key 9
0087 7F09	63	MOV	R7,#09H	;if key 9, get data


```

0089 1200B4    64      CALL    DISPLAY
008C 0108      65      JMP     START
008E 74F7      66      I:  MOV    A,#0F7H      ;for Check row 4
0090 F0        67      MOVX   @DPTR,A
0091 E0        68      MOVX   A,@DPTR
0092 5470      69      ANL    A,#70H
0094 F520      70      MOV    20H,A
0096 200407    71      JB     04H,J      ;Check if not key *
0099 7FFF      72      MOV    R7,#0FFH    ;if yes key *
009B 1200B4    73      CALL    DISPLAY
009E 0108      74      JMP     START
00A0 200507    75      J:  JB     05H,K      ;Check if not 0
00A3 7F00      76      MOV    R7,#00H    ;if yes 0,get data
00A5 1200B4    77      CALL    DISPLAY
00A8 0108      78      JMP     START
00AA 200605    79      K:  JB     06H,S      ;if not key #
00AD 7FF0      80      MOV    R7,#0F0H    ;if yes #, get data
00AF 1200B4    81      CALL    DISPLAY
00B2 0108      82      S:  JMP     START
00B4           83DISPLAY::MOV    DPTR,#8000H ;display routine
00B4 EF        84      MOV    A,R7
00B5 F0        85      MOVX   @DPTR,A
00B6 1200BA    86      CALL    DELAYS1
00B9 22        87      RET
00BA 7802      88  DELAYS1:MOV    R0,#02H    ;delay routine
00BC D8FE      89      DJNZ   R0,S
00BE 22        90      RET
0000=          91      END

```

7. จงเขียนโปรแกรมอ่านค่าจากคีย์บอร์ดแล้วส่งไปแสดงผลที่ 7 Segment หลักที่ 2 โดยการใช้รหัส ของ 7 Segment ในตารางข้างบนที่ได้ทำการทดลองมาก่อนหน้านี้ ในการกำหนด รหัส ของ 7 Segment ในการแสดงผล. ก่อนการทดลองปรับ สวิตช์ 1/2 ไปที่ตำแหน่ง ON ก่อน.

```

0000          1      ORG     0000H
0000 7488      2      MOV    A,#88H      ;Init 8255.1

```

0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVX	@DPTR,A	
0006 7FFF	5	MOV	R7,#0FFH	
0008 908002	6	START: MOV	DPTR,#8002H	;select port c
000B 7400	7	WAITE: MOV	A,#00H	;for check push key or not
000D F0	8	MOVX	@DPTR,A	
000E E0	9	MOVX	A,@DPTR	
000F 5570	10	ANL	A,70H	
0011 B47005	11	CJNE	A,#70H,SCAN	
0014 1200B4	12	CALL	DISPLAY	
0017 80EF	13	JMP	START	
0019 908002	14	SCAN: MOV	DPTR,#8002H	;for SCAN key
001C 74FE	15	MOV	A,#0FEH	;for check row 1
001E F0	16	MOVX	@DPTR,A	
001F E0	17	MOVX	A,@DPTR	
0020 5470	18	ANL	A,#70H	
0022 F520	19	MOV	20H,A	
0024 200407	20	JB	04H,AA	;Check if not key one
0027 7F01	21	MOV	R7,# 86H	;if key 1, get data
0029 1200B4	22	CALL	DISPLAY	
002C 80DA	23	JMP	START	
002E 200507	24	AA: JB	05H,BB	;Check if not key 2
0031 7F02	25	MOV	R7,# #0DBH	;if key 2, get data
0033 1200B4	26	CALL	DISPLAY	
0036 80D0	27	JMP	START	
0038 200607	28	BB: JB	06H,CC	;Check if not key 3
003B 7F03	29	MOV	R7,# 0CFH	;if key 3, get data
003D 1200B4	30	CALL	DISPLAY	
0040 80C6	31	JMP	START	
0042 74FD	32	CC: MOV	A,#0FDH	;for Check row 2
0044 F0	33	MOVX	@DPTR,A	
0045 E0	34	MOVX	A,@DPTR	
0046 5470	35	ANL	A,#70H	
0048 F520	36	MOV	20H,A	

004A 200407	37		JB	04H,D	;Check if not key 4
004D 7F04	38		MOV	R7,# 0E6H	;if key 4, get data
004F 1200B4	39		CALL	DISPLAY	
0052 80B4	40		JMP	START	
0054 200507	41	D:	JB	05H,E	;Check if not key 5
0057 7F05	42		MOV	R7,# 0EDH	;if key 5, get data
0059 1200B4	43		CALL	DISPLAY	
005C 80AA	44		JMP	START	
005E 200607	45	E:	JB	06H,F	;Check if not key 6
0061 7F06	46		MOV	R7,# 0FDH	;if key 6, get data
0063 1200B4	47		CALL	DISPLAY	
0066 80A0	48		JMP	START	
0068 74FB	49	F:	MOV	A,#0FBH	;for Check row 3
006A F0	50		MOVX	@DPTR,A	
006B E0	51		MOVX	A,@DPTR	
006C 5470	52		ANL	A,#70H	
006E F520	53		MOV	20H,A	
0070 200407	54		JB	04H,G	;Check if not key7
0073 7F07	55		MOV	R7,# 87H	;if key7 ,get data
0075 1200B4	56		CALL	DISPLAY	
0078 808E	57		JMP	START	
007A 200507	58	G:	JB	05H,H	;Check if not key8
007D 7F08	59		MOV	R7,# 0FFH	;if key 8,get data
007F 1200B4	60		CALL	DISPLAY	
0082 8084	61		JMP	START	
0084 200607	62	H:	JB	06H,I	;Check if not key 9
0087 7F09	63		MOV	R7,#0EFH	;if key 9, get data
0089 1200B4	64		CALL	DISPLAY	
008C 0108	65		JMP	START	
008E 74F7	66	I:	MOV	A,#0F7H	;for Check row 4
0090 F0	67		MOVX	@DPTR,A	
0091 E0	68		MOVX	A,@DPTR	
0092 5470	69		ANL	A,#70H	
0094 F520	70		MOV	20H,A	

0096 200407	71		JB	04H,J	;Check if not key *
0099 7FFF	72		MOV	R7,# #40h	;if yes key *
009B 1200B4	73		CALL	DISPLAY	
009E 0108	74		JMP	START	
00A0 200507	75	J:	JB	05H,K	;Check if not 0
00A3 7F00	76		MOV	R7,# 0BFH	;if yes 0,get data
00A5 1200B4	77		CALL	DISPLAY	
00A8 0108	78		JMP	START	
00AA 200605	79	K:	JB	06H,S	;if not key #
00AD 7FF0	80		MOV	R7,# #80H	;if yes #, get data
00AF 1200B4	81		CALL	DISPLAY	
00B2 0108	82	S:	JMP	START	
00B2 908000	83	DISPLAY:	MOV	DPTR,#8000H	;display routine
00B5 EF	84		MOV	A,R7	
00B6 F0	85		MOVX	@DPTR,A	
00B7 908001	86		MOV	DPTR,#8001H	
00BA 7401	87		MOV	a,#01h	
00BC F0	88		MOVX	@DPTR,A	
00BD 1200C1	89		CALL	DELAYS1	
00C0 22	90		RET		
00BA 7802	91	DELAYS1:	MOV	R0,#02H	;delay routine
00BC D8FE	92		DJNZ	R0,S	
00BE 22	93		RET		
0000=	94		END		

6.18 การทดลองที่ 6 การแสดงผลข้อมูลด้วยดอตเมตริก (Dot matrix display)

5. ทดลองเปลี่ยนข้อมูลที่ตำแหน่ง Address 0007H จากค่า 0ffh เป็นค่า 00H, 02H, 04H, 08H, 10H, 20H, 40H, 80H หลังจากนั้นทดลองเปลี่ยนค่าข้อมูลที่ตำแหน่ง Address 0010H จากค่า 00H เป็น 01H, 02H, แล้วบันทึกผลลัพธ์ของแต่ละการทดลองลงในตารางข้างล่างนี้

ข้อมูลที่ Address 0007	ข้อมูลที่ Address 0010 เป็น : 01H	ข้อมูลที่ Address 0010 เป็น : 02	ข้อมูลที่ Address 0010 เป็น : 04
01H	Row : 1 Column : 2	Row : 1 Column : 3	Row : 1 Column : 5
02H	Row : 2 Column : 2	Row : 2 Column : 3	Row : 2 Column : 5
04H	Row : 3 Column : 2	Row : 3 Column : 3	Row : 3 Column : 5
08H	Row : 4 Column : 2	Row : 4 Column : 3	Row : 4 Column : 5
10H	Row : 5 Column : 2	Row : 5 Column : 3	Row : 5 Column : 5
20H	Row : 6 Column : 2	Row : 6 Column : 3	Row : 6 Column : 5
40H	Row : 7 Column : 2	Row : 7 Column : 3	Row : 7 Column : 5
80H	Row : 8 Column : 2	Row : 8 Column : 3	Row : 8 Column : 5

ตารางที่ 6.16 ตารางบันทึกผลของการทดลอง Dot matrix display

6. ให้เขียนโปรแกรมแสดงตัวอักษรตัว K ปรากฏขึ้นที่ Dot matrix display

```
0000      1      ORG      0000H
0000 7480      2      MOV      A,#80H      ;INITIAL 8255
0002 908003      3      MOV      DPTR,#8003H
0005 F0         4      MOVX     @DPTR,A
0006 7F00      5      ST:  MOV      R7,#00H
0008 743E      6      MOV      A,#3eh      ;Data display K
000A 120023      7      CALL     DISPLAY
000D 7408      8      MOV      A,#08h
000F 120023      9      CALL     DISPLAY
0012 7414     10      MOV      A,#14h
0014 120023     11      CALL     DISPLAY
0017 7422     12      MOV      A,#22h
0019 120023     13      CALL     DISPLAY
```

001C 7400	14	MOV	A,#00h
001E 120023	15	CALL	DISPLAY
0021 80E3	16	JMP	ST
0023 908000	17	DISPLAY: MOV	DPTR,#8000H
0026 F0	18	MOVX	@DPTR,A
0027 908001	19	MOV	DPTR,#8001H
002A EF	20	MOV	A,R7
002B F0	21	MOVX	@DPTR,A
002C 0F	22	INC	R7
002D 78FA	23	MOV	R0,#0FAH
002F D8FE	24	DJNZ	R0,S
0031 22	25	RET	
0000=	26	END	

7. ให้เขียนโปรแกรมแสดงชื่อของตัวเองเป็นภาษาลาวที่ dot matrix display

0000	1	ORG	0000H	
0000 7480	2	MOV	A,#80H	;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVX	@DPTR,A	
0006 7F00	5	ST: MOV	R7,#00H	
0008 7478	6	MOV	A,#78H	
000A 120087	7	CALL	DISPLAY	
000D 7464	8	MOV	A,#64H	
000F 120087	9	CALL	DISPLAY	
0012 7400	10	MOV	A,#00H	
0014 120087	11	CALL	DISPLAY	
0017 740F	12	MOV	A,#0FH	
0019 120087	13	CALL	DISPLAY	
001C 747F	14	MOV	A,#7FH	
001E 120087	15	CALL	DISPLAY	
0021 7422	16	MOV	A,#22H	
0023 120087	17	CALL	DISPLAY	
0026 7412	18	MOV	A,#12F	
0028 120087	19	CALL	DISPLAY	

002B 7422	20	MOV A,#22H
002D 120087	21	CALL DISPLAY
0030 747D	22	MOV A,#7DH
0032 120087	23	CALL DISPLAY
0035 7400	24	MOV A,#00H
0037 120087	25	CALL DISPLAY
003A 747C	26	MOV A,#7CH
003C 120087	27	CALL DISPLAY
003F 7464	28	MOV A,#64H
0041 120087	29	CALL DISPLAY
0044 7404	30	MOV A,#04H
0046 120087	31	CALL DISPLAY
0049 747C	32	MOV A,#7CH
004B 120087	33	CALL DISPLAY
004E 7400	34	MOV A,#00H
0050 120087	35	CALL DISPLAY
0053 7400	36	MOV A,#00H
0055 120087	37	CALL DISPLAY
0058 7400	38	MOV A,#00H
005A 120087	39	CALL DISPLAY
005D 7400	40	MOV A,#00H
005F 120087	41	CALL DISPLAY
0062 7400	42	MOV A,#00H
0064 120087	43	CALL DISPLAY
0067 7400	44	MOV A,#00H
0069 120087	45	CALL DISPLAY
006C 7400	46	MOV A,#00H
006E 120087	47	CALL DISPLAY
0071 7400	48	MOV A,#00H
0073 120087	49	CALL DISPLAY
0076 7400	50	MOV A,#00H
0078 120087	51	CALL DISPLAY
007B 7400	52	MOV A,#00H
007D 120087	53	CALL DISPLAY

0080 7400	54	MOV	A,#00H
0082 120087	55	CALL	DISPLAY
0085 0106	56	JMP	ST
0087 908000	57	DISPLAY:	MOV DPTR,#8000H
008A F0	58	MOVX	@DPTR,A
008B 908001	59	MOV	DPTR,#8001H
008E EF	60	MOV	A,R7
008F F0	61	MOVX	@DPTR,A
0090 0F	62	INC	R7
0091 78FA	63	MOV	R0,#0FAH
0093 D8FE	64	DJNZ	R0,S
0095 22	65	RET	
0000=	66	END	

8. ให้เขียนโปรแกรมแสดงชื่อของตัวเองวิ่งอยู่บน Dot matrix display

0000	1	ORG	0000H	
0000 7480	2	ST1: MOV	A,#80H	;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H	
0005 F0	4	MOVX	@DPTR,A	
0006 7D00	5	MOV	R5,#00H	
0008 7B28	6	MOV	R3,#28h	
000A 7F00	7	MOV	R7,#00h	;Select column
000C 90004B	8	MOV	DPTR,#TABLE	
000F E0	9	ST: MOVX	A,@DPTR	
0010 A3	10	INC	DPTR	
0011 C082	11	PUSH	82H	
0013 C083	12	PUSH	83H	
0015 12001E	13	CALL	DISPLAY	
0018 D083	14	POP	83H	
001A D082	15	POP	82H	
001C 80F1	16	JMP	ST	
001E 908000	17	DISPLAY:	MOV DPTR,#8000H	
0021 F0	18	MOVX	@DPTR,A	
0022 908001	19	MOV	DPTR,#8001H	

0025 EF	20		MOV	A,R7	
0026 F0	21		MOVX	@DPTR,A	
0027 0F	22		INC	R7	
0028 7E04	23		MOV	R6,#04H	
002A 7801	24	L:	MOV	R0,#01H	
002C 7900	25	LOOP:	MOV	R1,#00H	
002E D9FE	26		JNZ	R1,S	
0030 D8FA	27		JNZ	R0,LOOP	
0032 DEF6	28		DJNZ	R6,L	
0034 BF1E13	29		CJNE	R7,#1EH,RE	
0037 7F00	30		MOV	R7,#00H	
0039 D083	31		POP	83H	
003B D082	32		POP	82H	
003D 90004B	33		MOV	DPTR,#TABLE	
0040 0D	34		INC	R5	
0041 ED	35		MOV	A,R5	
0042 FC	36		MOV	R4,A	
0043 A3	37	LOOP1:	INC	DPTR	
0044 DCFD	38		DJNZ	R4,LOOP1	
0046 DBC7	39		DJNZ	R3,ST	
0048 80B6	40		JMP	ST1	
004A 22	41	RE:	RET		
004B 3E081422	42	TABLE:	DB	03EH,08H,14H,22H,00H	;K
004F 00					
0050 3E040804	43		DB	3EH,04H,08H,04H,3EH,00H	;M
0054 3E00					
0056 223E2200	44		DB	22H,3EH,22H,00H	;I
005A 02023E02	45		DB	02H,02H,3EH,02H,02H,00H	;T
005E 0200					
0060 3E202020	46		DB	3EH,20H,20H,20H,00H	;L
0064 00					
0065 00000000	47		DB	00H,00H,00H,00H,00H,00H	
0069 0000					
006B 00000000	48		DB	00H,00H,00H,00H,00H,00H	

006F 0000

0071 00000000	49	DB	00H,00H,00H,00H,00H,00H
---------------	----	----	-------------------------

0075 0000

0077 00000000	50	DB	00H,00H,00H,00H,00H,00H
---------------	----	----	-------------------------

007B 0000

0000=	51	END	
-------	----	-----	--



6.19 การทดลองที่ 7 การแสดงผลข้อมูลด้วย LCD Display

5. จงเขียนโปรแกรมแสดงชื่อของตัวเองบน LCD

0000	1	ORG	0000H
0000 7488	2	MOV	A,#88H
0002 908003	3	MOV	DPTR,#08003H
0005 F0	4	MOVBX	@DPTR,A
0006 7438	5	MOV	A,#38H ;FUNCTION SET
0008 12006F	6	CALL	SEND_CMD
000B 740E	7	MOV	A,#0EH ;DISPLAY ON/OFF
000D 12006F	8	CALL	SEND_CMD
0010 7401	9	MOV	A,#01H ;CLEAR DISPLAY
0012 12006F	10	CALL	SEND_CMD
0015 7406	11	MOV	A,#06H ;CURSOR PRESET
0017 12006F	12	CALL	SEND_CMD
001A 1200A6	13	CALL	DELAY_
001D 7480	14	MOV	A,#80H ;SET DDRAM ADDRESS
001F 12006F	15	CALL	SEND_CMD
0022 7441	16	MOV	A,#'A'
0024 120086	17	CALL	SEND_DATA ;SEND DATA DISPLAY
0027 7442	18	MOV	A,#'B'
0029 120086	19	CALL	SEND_DATA ;SEND DATA DISPLAY
002C 7443	20	MOV	A,#'C'
002E 120086	21	CALL	SEND_DATA ;SEND DATA DISPLAY
0031 7444	22	MOV	A,#'D'
0033 120086	23	CALL	SEND_DATA ;SEND DATA DISPLAY
0036 7445	24	MOV	A,#'E'
0038 120086	25	CALL	SEND_DATA ;SEND DATA DISPLAY
003B 7446	26	MOV	A,#'F'
003D 120086	27	CALL	SEND_DATA ;SEND DATA DISPLAY
0040 7447	28	MOV	A,#'G'
0042 120086	29	CALL	SEND_DATA ;SEND DATA DISPLAY
0045 7448	30	MOV	A,#'H'
0047 120086	31	CALL	SEND_DATA ;SEND DATA DISPLAY
004A 7449	32	MOV	A,#'I'

004C 120086	33	CALL SEND_DATA ;SEND DATA DISPLAY
004F 744A	34	MOV A,#'J'
0051 120086	35	CALL SEND_DATA ;SEND DATA DISPLAY
0054 744B	36	MOV A,#'K'
0056 120086	37	CALL SEND_DATA ;SEND DATA DISPLAY
0059 744C	38	MOV A,#'L'
005B 120086	39	CALL SEND_DATA ;SEND DATA DISPLAY
005E 742E	40	MOV A,#'.'
0060 120086	41	CALL SEND_DATA ;SEND DATA DISPLAY
0063 742E	42	MOV A,#'.'
0065 120086	43	CALL SEND_DATA ;SEND DATA DISPLAY
0068 742E	44	MOV A,#'.'
006A 120086	45	CALL SEND_DATA ;SEND DATA DISPLAY
006D 80FE	46	JMP \$
006F 908000	47	SEND_CMD: MOV DPTR,#08000H ;FUNC SEND CMD
0072 F0	48	MOVX @DPTR,A
0073 908001	49	MOV DPTR,#08001H
0076 7400	50	MOV A,#00H
0078 F0	51	MOVX @DPTR,A
0079 7480	52	MOV A,#80H
007B F0	53	MOVX @DPTR,A
007C 7400	54	MOV A,#00H
007E F0	55	MOVX @DPTR,A
007F 12009D	56	CALL DELAY
0082 12009D	57	CALL DELAY
0085 22	58	RET
0086 908000	59	SEND_DATA:MOV DPTR,#08000H ;FUN SEND DATA
0089 F0	60	MOVX @DPTR,A
008A 908001	61	MOV DPTR,#08001H
008D 7420	62	MOV A,#20H
008F F0	63	MOVX @DPTR,A
0090 74A0	64	MOV A,#0A0H
0092 F0	65	MOVX @DPTR,A
0093 7420	66	MOV A,#20H

0095 F0	67	MOVX @DPTR,A	
0096 12009D	68	CALL DELAY	
0099 12009D	69	CALL DELAY	
009C 22	70	RET	
009D 7910	71	DELAY: MOV R1,#10H	;DELAY 1
009F 7800	72	BACK1: MOV R0,#00H	
00A1 D8FE	73	DJNZ R0,\$	
00A3 D9FA	74	DJNZ R1,BACK1	
00A5 22	75	RET	
00A6 7901	76	DELAY_: MOV R1,#01H	;DELAY 2
00A8 7A10	77	BACK: MOV R2,#10H	
00AA 7B00	78	LL: MOV R3,#00H	
00AC DBFE	79	DJNZ R3,\$	
00AE DAFA	80	DJNZ R2,LL	
00B0 D9F6	81	DJNZ R1,BACK	
00B2 22	82	RET	
0000=	83	END	

6. จงเขียนโปรแกรมแสดงข้อความสองบรรทัดบน LCD

0000	1	ORG 0000H	
0000 7488	2	MOV A,#88H	
0002 908003	3	MOV DPTR,#8003H	
0005 F0	4	MOVX @DPTR,A	
0006 7438	5	MOV A,#38H	;FUNCTION SET
0008 1200BF	6	CALL SEND_CMD	
000B 740E	7	MOV A,#0EH	;DISPLAY ON/OFF
000D 1200BF	8	CALL SEND_CMD	
0010 7401	9	MOV A,#01H	;CLEAR DISPLAY
0012 1200BF	10	CALL SEND_CMD	
0015 7406	11	MOV A,#06H	;CURSOR PRESET
0017 1200BF	12	CALL SEND_CMD	
001A 1200F6	13	CALL DELAY_	
001D 7481	14	MOV A,#81H	;SET DDRAM ADDR
001F 1200BF	15	CALL SEND_CMD	

0022 7444	16	MOV A,#'D'
0024 1200D6	17	CALL SEND_DATA ;SEND DATA DISPLAY
0027 7465	18	MOV A,#'e'
0029 1200D6	19	CALL SEND_DATA ;SEND DATA DISPLAY
002C 7470	20	MOV A,#'p'
002E 1200D6	21	CALL SEND_DATA ;SEND DATA DISPLAY
0031 7474	22	MOV A,#'t'
0033 1200D6	23	CALL SEND_DATA ;SEND DATA DISPLAY
0036 742E	24	MOV A,#'.'
0038 1200D6	25	CALL SEND_DATA ;SEND DATA DISPLAY
003B 7420	26	MOV A,#' '
003D 1200D6	27	CALL SEND_DATA ;SEND DATA DISPLAY
0040 7443	28	MOV A,#'C'
0042 1200D6	29	CALL SEND_DATA ;SEND DATA DISPLAY
0045 746F	30	MOV A,#'o'
0047 1200D6	31	CALL SEND_DATA ;SEND DATA DISPLAY
004A 746D	32	MOV A,#'m'
004C 1200D6	33	CALL SEND_DATA ;SEND DATA DISPLAY
004F 7470	34	MOV A,#'p'
0051 1200D6	35	CALL SEND_DATA ;SEND DATA DISPLAY
0054 7475	36	MOV A,#'u'
0056 1200D6	37	CALL SEND_DATA ;SEND DATA DISPLAY
0059 7474	38	MOV A,#'t'
005B 1200D6	39	CALL SEND_DATA ;SEND DATA DISPLAY
005E 7465	40	MOV A,#'e'
0060 1200D6	41	CALL SEND_DATA ;SEND DATA DISPLAY
0063 7472	42	MOV A,#'r'
0065 1200D6	43	CALL SEND_DATA ;SEND DATA DISPLAY
0068 74C3	44	MOV A,#0c3H ;SET DDRAM ADDRESS
006A 1200BF	45	CALL SEND_CMD
006D 7445	46	MOV A,#'E'
006F 1200D6	47	CALL SEND_DATA ;SEND DATA DISPLAY
0072 746E	48	MOV A,#'n'
0074 1200D6	49	CALL SEND_DATA ;SEND DATA DISPLAY

0077 7467	50	MOV	A,#'g'
0079 1200D6	51	CALL	SEND_DATA ;SEND DATA DISPLAY
007C 7469	52	MOV	A,#'i'
007E 1200D6	53	CALL	SEND_DATA ;SEND DATA DISPLAY
0081 746E	54	MOV	A,#'n'
0083 1200D6	55	CALL	SEND_DATA ;SEND DATA DISPLAY
0086 7465	56	MOV	A,#'e'
0088 1200D6	57	CALL	SEND_DATA ;SEND DATA DISPLAY
008B 7465	58	MOV	A,#'e'



008D 1200D6	59	CALL SEND_DATA ;SEND DATA DISPLAY
0090 7472	60	MOV A,#'r'
0092 1200D6	61	CALL SEND_DATA ;SEND DATA DISPLAY
0095 7469	62	MOV A,#'i'
0097 1200D6	63	CALL SEND_DATA ;SEND DATA DISPLAY
009A 746E	64	MOV A,#'n'
009C 1200D6	65	CALL SEND_DATA ;SEND DATA DISPLAY
009F 7467	66	MOV A,#'g'
00A1 1200D6	67	CALL SEND_DATA ;SEND DATA DISPLAY
00A4 7420	68	MOV A,#' '
00A6 1200D6	69	CALL SEND_DATA ;SEND DATA DISPLAY
00A9 7420	70	MOV A,#' '
00AB 1200D6	71	CALL SEND_DATA ;SEND DATA DISPLAY
00AE 7420	72	MOV A,#' '
00B0 1200D6	73	CALL SEND_DATA ;SEND DATA DISPLAY
00B3 7420	74	MOV A,#' '
00B5 1200D6	75	CALL SEND_DATA ;SEND DATA DISPLAY
00B8 7420	76	MOV A,#' '
00BA 1200D6	77	CALL SEND_DATA ;SEND DATA DISPLAY
00BD 80FE	78	JMP \$
00BF 908000	79	SEND_CMD:MOV DPTR,#8000H ;Function send cmd
00C2 F0	80	MOVX @DPTR,A
00C3 908001	81	MOV DPTR,#8001H
00C6 7400	82	MOV A,#00H
00C8 F0	83	MOVX @DPTR,A
00C9 7480	84	MOV A,#80H
00CB F0	85	MOVX @DPTR,A
00CC 7400	86	MOV A,#00H
00CE F0	87	MOVX @DPTR,A
00CF 1200ED	88	CALL DELAY
00D2 1200ED	89	CALL DELAY
00D5 22	90	RET
00D6 908000	91	SEND_DATA: MOV DPTR,#8000H ;Function send data
00D9 F0	92	MOVX @DPTR,A

00DA 908001	93	MOV DPTR,#8001H	
00DD 7420	94	MOV A,#20H	
00DF F0	95	MOVBX @DPTR,A	
00E0 74A0	96	MOV A,#0A0H	
00E2 F0	97	MOVBX @DPTR,A	
00E3 7420	98	MOV A,#20H	
00E5 F0	99	MOVBX @DPTR,A	
00E6 1200ED	100	CALL DELAY	
00E9 1200ED	101	CALL DELAY	
00EC 22	102	RET	
00ED 7910	103	DELAY: MOV R1,#10H	;DELAY 1
00EF 7800	104	BACK1: MOV R0,#00H	
00F1 D8FE	105	DJNZ R0,\$	
00F3 D9FA	106	DJNZ R1,BACK1	
00F5 22	107	RET	
00F6 7901	108	DELAY: MOV R1,#01H	;DELAY 2
00F8 7A10	109	BACK: MOV R2,#10H	
00FA 7B00	110	LL: MOV R3,#00H	
00FC DBFE	111	DJNZ R3,\$	
00FE DAFA	112	DJNZ R2,LL	
0100 D9F6	113	DJNZ R1,BACK	
0102 22	114	RET	
0000=	115	END	

6.20 การทดลองที่ 8 การควบคุมการทำงานของมอเตอร์ (Motor control)

และ ของสตีปิ้งมอเตอร์ (Stepping Motor)

6.20.1 การควบคุมการทำงานของมอเตอร์ (Motor control)

5. จงเขียนโปรแกรมสั่งให้มอเตอร์หมุนในช่วงเวลาหนึ่งแล้วหยุดหมุน

0000	1	ORG 0000H	
0000 908003	2	MOV DPTR,#8003H	;init 8255
0003 7480	3	MOV A,#80H	
0005 F0	4	MOVX @DPTR,A	
0006 908001	5	MOV DPTR,#8001H	;DC MOTRO ON
0009 7420	6	START: MOV A,#20H	
000B F0	7	MOVX @DPTR,A	
000C 120017	8	CALL DELAY1	
000F 7400	9	MOV A,#00H	;DC MOTOR OFF
0011 F0	10	MOVX @DPTR,A	
0012 120020	11	CALL DELAY2	
0015 80F2	12	JMP START	
0017 7800	13	DELAY1: MOV R0,#00H	;DELAY TIME 1
0019 7900	14	LOOP: MOV R1,#00H	
001B D9FE	15	DJNZ R1,\$	
001D D8FA	16	DJNZ R0,LOOP	
001F 22	17	RET	
0020 7803	18	DELAY2: MOV R0,#03H	;DELAY TIME 2
0022 7900	19	LOOP2: MOV R1,#00H	
0024 7A00	20	LOOP1: MOV R2,#00H	
0026 DAFE	21	DJNZ R2,\$	
0028 D9FA	22	DJNZ R1,LOOP1	
002A D8F6	23	DJNZ R0,LOOP2	
002C 22	24	RET	
0000=	25	END	

6. ให้เขียนโปรแกรมอ่านค่ามาจาก Dip Switch บิตที่ 4 และ บิตที่ 5 เพื่อสั่งให้มอเตอร์ทำงาน โดยที่บิตที่ 5 ให้เป็นการควบคุมการทำงานของมอเตอร์ บิตที่ 4 ให้เป็นการควบคุมทิศทางหมุนของมอเตอร์ ก่อนการทดลองต้องปรับสวิตช์ 1/7 ไปที่ตำแหน่ง ON ก่อน

```

0000          1          ORG  0000H
0000 908003    2          MOV  DPTR,#8003H  ;init 8255
0003 7489      3          MOV  A,#89H
0005 F0        4          MOVX  @DPTR,A
0006 908002    5  START:  MOV  DPTR,#8002H  ;READ DIP SW
0009 E0        6          MOVX  A,@DPTR
000A 5430      7          ANL   A,#30H
000C B4300B    8          CJNE  A,#30H,NEXT1
000F 7430      9          MOV  A,#30H      ;MOVE BACKWARD
0011 908001   10          MOV  DPTR,#8001H
0014 F0       11          MOVX  @DPTR,A
0015 120028   12          CALL  DELAY
0018 80EC     13          JMP   START
001A B42009   14  NEXT1:  CJNE  A,#20H,NEXT2
001D 7420     15          MOV  A,#20H      ;MOVE FORWARD
001F 908001   16          MOV  DPTR,#8001H
0022 F0       17          MOVX  @DPTR,A
0023 120028   18          CALL  DELAY
0026 80DE     19  NEXT2:  JMP   START
0028 7800     20  DELAY:  MOV  R0,#00H      ;DELAY TIME
002A 7900     21  LOOP:   MOV  R1,#00H
002C D9FE     22          DJNZ  R1,$
002E D8FA     23          DJNZ  R0,LOOP
0030 22       24          RET
0000=         25          END

```

2. ให้เขียนโปรแกรมสั่งให้มอเตอร์หมุนไปทางหน้าแล้วหมุนกลับหลัง โดยการหมุนแต่ละอย่างนั้นนานประมาณ 5 วินาที

0000	1	ORG 0000H	
0000 908003	2	MOV DPTR,#8003H	;init 8255
0003 7480	3	MOV A,#80H	
0005 F0	4	MOVX @DPTR,A	
0006 908001	5	MOV DPTR,#8001H	
0009 7420	6	START: MOV A,#20H	;FORWARD
000B F0	7	MOVX @DPTR,A	
000C 120026	8	CALL DELAY2	
000F 7400	9	MOV A,#00H	;DC MOTOR OFF
0011 F0	10	MOVX @DPTR,A	
0012 12001D	11	CALL DELAY1	
0015 7430	12	MOV A,#30H	;BACKWARD
0017 F0	13	MOVX @DPTR,A	
0018 120026	14	CALL DELAY2	
001B 80EC	15	JMP START	
001D 7800	16	DELAY1: MOV R0,#00H	;DELAY TIME 1
001F 7900	17	LOOP: MOV R1,#00H	
0021 D9FE	18	DJNZ R1,\$	
0023 D8FA	19	DJNZ R0,LOOP	
0025 22	20	RET	
0026 7805	21	DELAY2: MOV R0,#05H	;DELAY TIME 2
0028 7900	22	LOOP2: MOV R1,#00H	
002A 7A00	23	LOOP1: MOV R2,#00H	
002C DAFE	24	DJNZ R2,\$	
002E D9FA	25	DJNZ R1,LOOP1	
0030 D8F6	26	DJNZ R0,LOOP2	
0032 22	27	RET	
0000=	28	END	

6.20.2 การควบคุมการทำงานของ Stepping Motor

5. จงเขียนโปรแกรมสั่งให้สเต็ปมอเตอร์ทำงานโดยการกระตุ้นแบบสองเฟส

0000	1	ORG 0000H	
0000 7480	2	MOV A,#80H	;INITIAL 8255
0002 908003	3	MOV DPTR,#8003H	
0005 F0	4	MOVX @DPTR,A	
0006 7F00	5	START: MOV R7,#00H	
0008 90002B	6	MOV DPTR,#TABLE	;SELECT DATA
000B E0	7	NEXT: MOVX A,@DPTR	
000C A3	8	INC DPTR	
000D 0F	9	INC R7	
000E C082	10	PUSH 82H	
0010 C083	11	PUSH 83H	
0012 908001	12	MOV DPTR,#8001H	;STEPPING ON
0015 F0	13	MOVX @DPTR,A	
0016 120022	14	CALL DELAY	
0019 D083	15	POP 83H	
001B D082	16	POP 82H	
001D BF05EB	17	CJNE R7,#05H,NEXT	
0020 80E4	18	JMP START	
0022 780A	19	DELAY: MOV R0,#0AH	;ROUTINE DELAY
0024 7900	20	LOOP1: MOV R1,#00H	
0026 D9FE	21	DJNZ R1,\$	
0028 D8FA	22	DJNZ R0,LOOP1	
002A 22	23	RET	
002B 0905060A	24	TABLE: DB 09H,05H,06H,0AH	;DATA
0000=	25	END	

6. จงเขียนโปรแกรมสั่งให้สเต็ปมอเตอร์ทำงานโดยการกระตุ้นแบบหนึ่งเฟส และสองเฟส

0000	1	ORG 0000H	
0000 7480	2	MOV A,#80H	;INITIAL 8255
0002 908003	3	MOV DPTR,#8003H	

0005 F0	4	MOVX @DPTR,A
0006 7F00	5 START:	MOV R7,#00H
0008 90002B	6	MOV DPTR,#TABLE ;SELECT DATA
000B E0	7 NEXT:	MOVX A,@DPTR
000C A3	8	INC DPTR
000D 0F	9	INC R7
000E C082	10	PUSH 82H
0010 C083	11	PUSH 83H
0012 908001	12	MOV DPTR,#8001H ;STEPING ON
0015 F0	13	MOVX @DPTR,A
0016 120022	14	CALL DELAY
0019 D083	15	POP 83H
001B D082	16	POP 82H
001D BF09EB	17	CJNE R7,#09H,NEXT
0020 80E4	18	JMP START
0022 780A	19 DELAY:	MOV R0,#0AH ;ROUTINE DELAY
0024 7900	20 LOOP1:	MOV R1,#00H
0026 D9FE	21	DJNZ R1,\$
0028 D8FA	22	DJNZ R0,LOOP1
002A 22	23	RET
002B 09010504	24 TABLE:	DB 09H,01H,05H,04,06H,02H,0AH,08H
002F 06020A08		
0000=	25	END

7. จงเขียนโปรแกรมสั่งให้สแตมป์มอเตอร์ทำงานโดยหมุนไป 90 องศาแล้วหยุด

0000	1	ORG 0000H
0000 7480	2	MOV A,#80H ;INITIAL 8255
0002 908003	3	MOV DPTR,#8003H
0005 F0	4	MOVX @DPTR,A
0006 7E00	5	MOV R6,#00H
0008 7F00	6 START:	MOV R7,#00H
000A 0E7		INC R6
000B 900031	8	MOV DPTR,#TABLE ;SELECT DATA
000E E0	9 NEXT:	MOVX A,@DPTR

000F A3	10	INC	DPTR	
0010 0F	11	INC	R7	
0011 C082	12	PUSH	82H	
0013 C083	13	PUSH	83H	
0015 908001	14	MOV	DPTR,#8001H	;STEPPING ON
0018 F0	15	MOVX	@DPTR,A	
0019 120028	16	CALL	DELAY	
001C D083	17	POP	83H	
001E D082	18	POP	82H	
0020 BF05EB	19	CJNE	R7,#05H,NEXT	
0023 BE04E2	20	CJNE	R6,#04H,START	
0026 80FE	21	JMP	\$	
0028 780A	22	DELAY:	MOV R0,#0AH	;ROUTINE DELAY
002A 7900	23	LOOP1:	MOV R1,#00H	
002C D9FE	24		DJNZ R1,\$	
002E D8FA	25		DJNZ R0,LOOP1	
0030 22	26		RET	
0031 0905050A	27	TABLE:	DB 09H,05H,05H,0AH	
0000=	28		END	

6.21 การทดลองที่ 9 การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก

5. จากโปรแกรมที่ 9.1 จงเปลี่ยนค่าข้อมูลที่ตำแหน่ง Address 0007H ด้วยค่าดังตารางข้างล่างนี้ แล้วบันทึกผลที่ได้รับ

ปริมาณดิจิตอล	แรงดันที่วัดได้ (V)
00H	0
19H	1.52
32H	0.76
4BH	2.09
64H	3.08
7DH	1.98
96H	1.05
AFH	2.44
C8H	0.19
E1H	1.35
FFH	2.55

ตารางที่ 6.17 ตารางบันทึกผลของการทดลอง DAC

6. ให้นำเอาหลอดไฟต่อใส่ที่ Connector ที่ 1 แล้วจึงเขียนโปรแกรมควบคุมความสว่างของหลอดไฟนั้น โดยให้สว่างมากแล้วค่อยๆ สว่างน้อยลง

```
0000      1      ORG    0000H
0000 748A      2      MOV    A,#8AH ;INITIAL 8255
0002 908003      3      MOV    DPTR,#8003H
0005 F0        4      MOVX   @DPTR,A
0006 74FF      5  START:  MOV    A,#0FFH    ;DATA DISPLAY
0008 9080000      6      MOV    DPTR,#80000H
000B F0        7  AGAIN:  MOVX   @DPTR,A
000C 14        8      DEC    A
000D 12001D      9      CALL   DELAY
0010 B400F8     10      CJNE   A,#00H,AGAIN
0013 F0        11  NEXT:  MOVX   @DPTR,A
0014 12001D     12      CALL   DELAY
0017 04        13      INC    A
0018 B4FFF8     14      CJNE   A,#0FFH,NEXT
```


001B 80E9	15	JMP	START
001D 7800	16	DELAY:	MOV R0,#00H
001F 7900	17	LOOP:	MOV R1,#00H
0021 D9FE	18		DJNZ R1,\$
0023 D8FA	19		DJNZ R0,LOOP
0025 22	20		RET
0000=	21		END

7. ให้เขียนโปรแกรมแปลงสัญญาณดิจิตอลเป็นสัญญาณรูปฟันเลื่อยเพื่อดูผลที่ได้ให้ใช้ออสซิลโลสโคปวัด

0000	1	ORG	0000H
0000 748A	2	MOV	A,#8AH ;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H
0005 F0	4	MOVX	@DPTR,A
0006 9008000	5	MOV	DPTR,#8000H
0009 7400	6	START:	MOV A,#00H ;DATA DISPLAY
000B F0	7	NEXT:	MOVX @DPTR,A
000C 120015	8	CALL	DELAY
000F 04	9	INC	A
0010 B400F8	10	CJNE	A,#00H,NEXT
0013 80F4	11	JMP	START
0015 7800	12	DELAY:	MOV R0,#00H
0017 7900	13	LOOP:	MOV R1,#00H
0019 D9FE	14		DJNZ R1,\$
001B D8FA	15		DJNZ R0,LOOP
001D 22	16		RET
0000=	17		END

8. ให้เขียนโปรแกรมแปลงสัญญาณดิจิตอลเป็นสัญญาณรูปสี่เหลี่ยมเพื่อดูผลที่ได้ให้ใช้ออสซิลโลสโคปวัด

0000	1	ORG	0000H
0000 748A	2	MOV	A,#8AH ;INITIAL 8255
0002 908003	3	MOV	DPTR,#8003H
0005 F0	4	MOVX	@DPTR,A

0006 9008000	5	MOV DPTR,#8000H
0009 7400	6	MOV A,#00H ;DATA DISPLAY 00H,FFH
000B F0	7 NEXT:	MOVX @DPTR,A
000C 120012	8	CALL DELAY
000F F4	9	CPL A
0010 80F9	10	JMP NEXT
0012 7800	11 DELAY:	MOV R0,#00H
0014 7900	12 LOOP:	MOV R1,#00H
0016 D9FE	13	DJNZ R1,\$
0018 D8FA	14	DJNZ R0,LOOP
001A 22 15		RET
0000=	16	END



6.22 การทดลองที่ 10 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิทัล

9. จากโปรแกรมที่ 10.1 จงเปลี่ยนค่าข้อมูลที่ป้อนให้ด้วยค่าดังตารางข้างล่างนี้ แล้วบันทึกผลที่ได้รับ

แรงดันอินพุต A/D	PORT MONITOR	ข้อมูล
0V	○ ○ ○ ○ ○ ○ ○ ○	00H
0.5V	○ ○ ● ● ○ ● ● ●	37H
1V	○ ● ● ○ ● ○ ● ●	6BH
1.5V	● ○ ● ○ ○ ○ ○ ○	A0H
2V	● ● ○ ● ○ ● ○ ●	D5H
2.55V	● ● ● ● ● ● ● ●	FFH

ตารางที่ 6.18 ตารางบันทึกผลของการทดลองของ ADC

10. ให้เขียนโปรแกรมอ่านค่าเข้ามาจาก POT แปลงให้เป็นสัญญาณดิจิทัลแล้วส่งไปแสดงผลที่ Port Monitor ก่อนการทดลองให้ปรับ สวิตช์ 3/3, สวิตช์ 1/11 และ สวิตช์ 1/1 ไปที่ตำแหน่ง ON ก่อน แล้วสังเกตผลที่ได้รับที่ Port Monitor ในเวลาที่เราปรับ POT ขึ้นและลง

```
0000      1      ORG      0000H
0000 7489      2      MOV     A,#89H          ;INITIAL 8255
0002 908003     3      MOV     DPTR,#8003H
0005 F0         4      MOVX    @DPTR,A
0006 908002     5  START:  MOV     DPTR,#8002H      ;READ ADC
0009 E0         6      MOVX    A,@DPTR
000A 908000     7      MOV     DPTR,#8000H ;DISPLAY OUTPUT
000D F0         8      MOVX    @DPTR,A
000E 120013     9      CALL    DELAY
0011 80F3      10      JMP     START
0013 790A     11  DELAY:  MOV     R1,#0AH
0015 7800     12  LOOP:   MOV     R0,#00H
0017 A817     13      MOV     R0,$
0019 A915     14      MOV     R1,LOOP
001B 22       15      RET
0000=        16      END
```

11. อธิบายการติด ดับของหลอด LED หมายถึงอะไร

6.23 การทดลองที่ 11 การควบคุมการทำงานของ Heater และ Temperature Sensor

6.23.1 การควบคุมการทำงานของ Heater

5. จงเขียนโปรแกรมสั่งให้ Heater หยุดการทำงาน

0000	1	ORG	0000H
0000 7480	2	MOV	A,#80H
0002 908003	3	MOV	DPTR,#8003H
0005 F0	4	MOVX	@DPTR,A
0006 908001	5	MOV	DPTR,#8001H
0009 7400	6	MOV	A,#00H
000B F0	7	MOVX	@DPTR,A
000C 80FE	8	JMP	\$
0000=	9	END	

6. ให้เขียนโปรแกรมอ่านค่ามาจาก Dip Switch ตัวที่ 2 บิตที่ 1 มาเป็นการควบคุมการทำงานของ Heater

0000	1	ORG	0000H
0000 7480	2	MOV	A,#80H ;Init 8255
0002 908003	3	MOV	DPTR,#8003H
0005 F0	4	MOVX	@DPTR,A
0006 908002	5	ST: MOV	DPTR,#8002H ;Check SW
0009 E0	6	MOVX	A,@DPTR
000A 5401	7	ANL	A,#01H
000C B40A08	8	CJNE	A,#0AH,NEXT
000F 7440	9	MOV	A,#40H ;On heater
0011 908001	10	MOV	DPTR,#8001H
0014 F0	11	MOVX	@DPTR,A
0015 80EF	12	JMP	ST
0017 908001	13	NEXT: MOV	DPTR,#8001H ;Off heater
001A 7400	14	MOV	A,#00H
001C F0	15	MOVX	@DPTR,A
001D 80E7	16	JMP	ST
0000=	17	END	

6.24 การทดลองที่ 12 การวัดค่าความเร็วของมอเตอร์ (Speed sensor)

1. ให้เขียนโปรแกรมวัดความเร็วของมอเตอร์แล้วแสดงผลออกมา

0000	1	ORG	0000H
0000 02000E	2	JMP	START
000B	3	ORG	0BH
000B 02004A	4	JMP	INT
000E 7800	5	START:	MOV R0,#00H
0010 7900	6	MOV	R1,#00H
0012 908003	7	MOV	DPTR,#8003H
0015 7489	8	MOV	A,#89H
0017 F0	9	MOVX	@DPTR,A
0018 758901	10	MOV	TMOD,#01H
001B D2AF	11	SETB	EA
001D D2A9	12	SETB	ET0
001F 758A00	13	MOV	TL0,#00H
0022 758C00	14	MOV	TH0,#00H
0025 D28C	15	SETB	TR0
0027 908000	16	MOV	DPTR,#8000H
002A 7400	17	MOV	A,#00H
002C F0	18	MOVX	@DPTR,A
002D 908001	19	MOV	DPTR,#8001H
0030 7420	20	MOV	A,#20H
0032 F0	21	MOVX	@DPTR,A
0033 908002	22	ST:	MOV DPTR,#8002H
0036 E0	23	MOVX	A,@DPTR
0037 5480	24	ANL	A,#80H
0039 B480F7	25	CJNE	A,#80H,ST
003C E9	26	MOV	A,R1
003D 04	27	INC	A
003E F9	28	MOV	R1,A
003F 908001	29	AG:	MOV DPTR,#8002H
0042 E0	30	MOVX	A,@DPTR
0043 5480	31	ANL	A,#80H
0045 B480EB	32	CJNE	A,#80H,ST

0048 80F5	33		JMP	AG
004A C0E0	34	INT:	PUSH	ACC
004C C0D0	35		PUSH	PSW
004E E8	36		MOV	A,R0
004F 04	37		INC	A
0050 F8	38		MOV	R0,A
0051 B40E0E	39		CJNE	A,#0eH,S
0054 7800	40		MOV	R0,#00H
0056 E9	41		MOV	A,R1
0057 75F003	42		MOV	B,#03H
005A 84	43		DIV	AB
005B D4	44		DA	A
005C 908000	45		MOV	DPTR,#8000H
005F F0	46		MOVBX	@DPTR,A
0060 7900	47		MOV	R1,#00H
0062 758A00	48	S:	MOV	TL0,#00H
0065 758C00	49		MOV	TH0,#00H
0068 D0D0	50		POP	PSW
006A D0E0	51		POP	ACC
006C 32	52		RETI	
0000=	53		END	

บทที่ 7

สรุปและวิจารณ์

จากการออกแบบ และสร้างบอร์ดทดลองขึ้นมา โดยมีทั้งทางฮาร์ดแวร์และซอฟต์แวร์ ที่ได้กล่าวผ่านมาแล้วนั้น ทำให้สามารถที่นำบอร์ดทดลองในโครงการนี้ ไปใช้ประโยชน์ได้ในการเรียนการสอน โดยการใช้งานบอร์ดนี้ไม่ต้องอาศัยเครื่องมืออื่นๆมากนัก เพราะนอกจากจะมีเครื่องมือหรือชุดที่จำเป็นเหมือนบอร์ดทดลองอื่นๆที่มีทั่วไปในท้องตลาดแล้ว ยังได้มีการจัดทำบททดลอง พร้อมคำอธิบาย พร้อมคัมภีร์จึงสามารถที่จะนำไปใช้งานได้เลย แต่อย่างไรก็ตามในการทำโครงการนี้ก็พบปัญหาหลายอย่างเช่น มีเวลาจำกัด, และห่างไกลจาก Adviser ดังนั้นจึงมีปัญหาในการให้คำปรึกษา, ให้คำแนะนำ. ในขณะที่จำนวนงานที่ต้องทำนั้นก็ยังมีมาก ทั้งการออกแบบฮาร์ดแวร์ ของบอร์ดทดลอง, การสร้างบอร์ดทดลอง, การหาวิธีและทำการทดสอบบอร์ดทดลอง, การทดสอบฮาร์ดแวร์, การหาอุปกรณ์, การเขียนโปรแกรม การออกแบบการทำใบงานทดลองอื่นๆ, ซึ่งมักจะเกิดปัญหาที่คาดไม่ถึงถึงจะต้องใช้เวลาานกว่าจะพบและทำการแก้ไขข้อผิดพลาดได้ทั้งที่มีอยู่ในวงจรและตัวโปรแกรมซึ่งเป็นไปได้ที่อาจจะมีข้อผิดพลาดบางประการ ที่ทางผู้จัดทำยังทดสอบไปไม่ถึง ซึ่งทางผู้จัดทำต้องขออภัยล่วงหน้า และจะได้หาโอกาสดำเนินการแก้ไขต่อไป

7.1 ปัญหาที่พบและวิธีการแก้ไข

1. มีเวลาจำกัดในการทำโครงการนี้, และห่างไกลจาก Adviser ดังนั้นจึงมีปัญหาในการให้คำปรึกษา, ให้คำแนะนำ ในการแก้ไขได้ติดต่อกันทาง E-mail แต่ก็ทำให้มีความชักช้า และยุ่งยาก เนื่องจากว่าอยู่ที่ มหาวิทยาลัยแห่งชาติลาวมี Internet ให้ใช้ในเวลาจำกัด
2. เนื่องจากการออกแบบฮาร์ดแวร์เป็นวงจรรวมขนาดใหญ่และสลับซับซ้อนนั้นในการออกแบบต้องใช้เวลานาน ในการหาข้อมูลตัวอุปกรณ์ที่จะมาทำการออกแบบค่อนข้างหาได้ลำบาก เมื่อได้ออกแบบวงจรและสร้างเป็นบอร์ดครั้งแรก แต่ได้มีการปรับเปลี่ยนบางโมดูลเพื่อให้เหมาะสมในการนำไปใช้งานทำให้ต้องได้แก้ไขการออกแบบใหม่
3. ในการออกแบบนั้นเราไม่มีอุปกรณ์ และไม่มีคู่มือของอุปกรณ์ ทำให้โครงการที่ได้ออกแบบมานั้นมีความผิดพลาด ทางด้าน PCB เช่นตำแหน่งของ Foot print ผิดและขนาดของ Foot print ผิด
4. เมื่อได้บอร์ดมา แต่ไม่มีอุปกรณ์ที่จะเอามาประกอบ ทำให้เสียเวลาในการประกอบบอร์ด เพราะต้องรออุปกรณ์จาก Adviser
5. ในขั้นตอนการทดสอบจะพบปัญหาที่บางโมดูลไม่ทำงาน เนื่องจากมีความผิดพลาดเช่นขาของ Foot print ผิด วิธีการแก้ไขได้ตรวจสอบวงจรที่ออกแบบอีกครั้งว่าถูกต้องหรือไม่ ตรวจสอบตำแหน่งของขาอุปกรณ์ กับตำแหน่งขาของ Foot print ที่ได้ออกแบบไปนั้นว่าตรงกันหรือไม่, ใช้

อุปกรณ์ตรวจสอบหาสัญญาณและแก้ไขส่วนที่ผิดโดยใช้สายต่อแทนลาย Print ที่มีในบอร์ด ให้ถูกต้อง จากนั้นเขียน โปรแกรมทดสอบทีละ โมดูลจนกระทั่งทุก โมดูลทำงานถูกต้อง

7.2 สรุปโครงการที่ทำมา

1. จากโครงการที่ทำมาได้ ออกแบบบอร์ดไมโครคอนโทรลเลอร์สำหรับการทดลอง โดยประกอบมี สองบอร์ดคือ บอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตเอาต์พุต
2. ได้ประกอบ, ทำการทดสอบ และแก้ไขจุดที่ผิดพลาดของบอร์ดไมโครคอนโทรลเลอร์ 8051 และ บอร์ดทดลองอินพุต เอาต์พุต ทุกๆ โมดูลสามารถทำงานได้ถูกต้อง
3. ได้เขียนวิทยานิพนธ์, ใบสั่งงานสำหรับการทดลองพร้อมเฉลยของการทดลอง และคู่มือการใช้งานของทั้งสองบอร์ดคือ บอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตเอาต์พุต
4. ได้ทำการแก้ไขในจุดที่ผิดพลาดของบอร์ด ให้เป็นบอร์ดที่สมบูรณ์พร้อมที่จะเอาไปใช้งาน
5. ผลรับของโครงการที่ได้ทำผ่านมานี้เห็นว่าเป็นไปตามจุดประสงค์ที่ได้ตั้งไว้

7.3 คำวิจารณ์

บอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตเอาต์พุตที่ทำขึ้นนี้ เป็นสองบอร์ดที่แยกออกจากกัน ต้องใช้ไฟเลี้ยงสองชุด คือบอร์ดละชุด และในเวลาใช้งานต้องใช้ทั้งสองบอร์ดรวม, ใน บอร์ดทดลองอินพุตเอาต์พุต จะใช้ไฟทั้ง 5 โวลต์ และ 12 โวลต์ เพราะมีอุปกรณ์บางตัวใช้ไฟ 12 โวลต์, และโมดูลต่างที่มีในบอร์ดจะต่อพอร์ตใช้งานร่วมกันอยู่ ดังนั้นมีบางโมดูลที่ไม่สามารถที่จะทำงานร่วมกับได้, ในการที่จะทำการทดลองในแต่ละ โมดูลจะต้องได้ปรับสวิตช์เพื่อให้แต่ละ โมดูลนั้นทำงาน

7.4 แนวทางพัฒนาต่อไป

1. อาจจะสร้างให้เป็นบอร์ดทดลองที่มีเพียงบอร์ดเดียว คือรวมทั้งบอร์ดไมโครคอนโทรลเลอร์ 8051 และบอร์ดทดลองอินพุตเอาต์พุตเข้าด้วยกัน
2. ให้เป็นบอร์ดที่ใช้ไฟเลี้ยงเพียงชุดเดียว
3. พัฒนาให้เป็นบอร์ดที่ใช้งานได้ง่ายและสะดวก โดยไม่ต้องไปเช็ดสวิตช์ ใด
4. สร้างให้เป็นสามารถใช้งานทุกๆ โมดูลร่วมกันได้.
5. ทำให้บอร์ดมีขนาดเล็กลง

ภาคผนวก



ชุดคำสั่งของ 8051 โดยสังเขป

ตารางสัญลักษณ์ที่ใช้

สัญลักษณ์	ความหมาย
Rn	รีจิสเตอร์ภายใน R0-R7
Direct	รีจิสเตอร์ SFR และหน่วยความจำข้อมูลภายใน
@Ri	ค่าแอดเดรสหน่วยความจำภายใน อ้างแอดเดรส โดยอ้อมผ่านรีจิสเตอร์ R0 หรือ R1
#data	ค่าคงที่ขนาด 8 บิต (ค่าจาก 0-FF)
#data16	ค่าคงที่ขนาด 16 บิต (ค่าจาก 0-FFFF)
addr11	ค่าตำแหน่งแอดเดรสจำนวน 11 บิต สำหรับคำสั่ง ACALL หรือ AJUMP
addr16	ค่าตำแหน่งแอดเดรสจำนวน 16 บิต สำหรับคำสั่ง LCALL หรือ LJUMP
rel	ค่าออฟเซต (Offset) หรือค่าบวกความสัมพัทธ์ (ค่าจากจาก-17ถึง 18 ไบต์)
bit	ตำแหน่งบิตของหน่วยความจำภายในที่อ้างถึง ได้แบบบิต หรือ SFR
@dptr	แอดเดรสของหน่วยความจำภายนอก หรือพอยน์เตอร์ โดยอ้อมผ่านรีจิสเตอร์ DPTR

คำสั่งประมวลผลทางคณิตศาสตร์ (Arithmetic Operations)

รูปแบบคำสั่ง	ความหมาย	จำนวน ไบต์	จำนวนเมฆ ซินไซเกิล
ADD A,Rn	บวกค่า Rn กับ A	1	1
ADD A,direct	บวกค่าในหน่วยความจำ direct กับ A	2	1
ADD A,@Ri	บวกค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
ADD A,#data	บวกค่าคงที่ 8 บิตกับ A	2	1
ADDC A,Rn	บวกค่า Rn กับ A พร้อมแฟล็ก Carry	1	1
ADDC A,direct	บวกค่าในหน่วยความจำ direct กับ A พร้อมแฟล็ก Carry	2	1
ADDC A,@Ri	บวกค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A พร้อม แฟล็ก Carry	1	1
ADDC A,#data	บวกค่าคงที่ 8 บิตกับ A พร้อมแฟล็ก Carry	2	1
SUBB A,Rn	ลบค่า Rn กับ A พร้อมแฟล็ก Borrow	1	1
SUBB A,direct	ลบค่าในหน่วยความจำ direct กับ A พร้อม แฟล็ก Borrow	2	1
SUBB A,@Ri	ลบค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A พร้อม แฟล็ก Borrow	1	1
SUBB A,#data	ลบค่าคงที่ 8 บิตกับ A พร้อมแฟล็ก Borrow	2	1
INC A	เพิ่มค่าใน A	1	1
INC Rn	เพิ่มค่าใน Rn	1	1
INC direct	เพิ่มค่าในหน่วยความจำ direct	2	1
INC @Ri	เพิ่มค่าในหน่วยความจำที่เก็บอยู่ใน Ri	1	1
DEC A	ลดค่าใน A	1	1
DEC Rn	ลดค่าใน Rn	1	1
DEC direct	ลดค่าในหน่วยความจำ direct	2	1
DEC @Ri	ลดค่าในหน่วยความจำที่เก็บอยู่ใน Ri	1	1
INC DPTR	เพิ่มค่าใน DPTR	1	2
MUL AB	คูณ A กับ B แล้วเก็บค่าใน A	1	4
DIV AB	หาร A กับ B แล้วเก็บค่าใน A	1	4
DA A	ทำ decimal adjust ค่าใน A	1	1

คำสั่งทางตรรก (Logical Operation)

รูปแบบคำสั่ง	ความหมาย	จำนวนไบต์	จำนวนเมกซ์ ซินไซเคิล
ANL A,Rn	AND ค่า Rn กับ A	1	1
ANL A,direct	AND ค่าในหน่วยความจำ direct กับ A	2	1
ANL A,@Ri	AND ค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
ANL A,#data	AND ค่าคงที่ 8 บิตกับ A	2	1
ANL direct,A	AND ค่า A กับหน่วยความจำ direct	2	1
ANL direct,#data	AND ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
ORL A,Rn	OR ค่า Rn กับ A	1	1
ORL A,direct	OR ค่าในหน่วยความจำ direct กับ A	2	1
ORL A,@Ri	OR ค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
ORL A,#data	OR ค่าคงที่ 8 บิตกับ A	2	1
ORL direct,A	OR ค่า A กับหน่วยความจำ direct	2	1
ORL direct,#data	OR ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
XRL A,Rn	EX-OR ค่า Rn กับ A	1	1
XRL A,direct	EX-OR ค่าในหน่วยความจำ direct กับ A	2	1
XRL A,@Ri	EX-OR ค่าในหน่วยความจำที่เก็บอยู่ใน Ri กับ A	1	1
XRL A,#data	EX-OR ค่าคงที่ 8 บิตกับ A	2	1
XRL direct,A	EX-OR ค่า A กับหน่วยความจำ direct	2	1
XRL direct,#data	EX-OR ค่าคงที่ 8 บิตกับหน่วยความจำ direct	3	2
CLR A	ทำค่าใน A ให้เป็น 0	1	1
APL A	กลับค่าใน A เป็นตรงข้ามทุกบิต	1	1
RL A	หมุนบิตใน A ไปทางซ้าย 1 บิตและบิต 0 มีค่า 0	1	1
RLC A	หมุนบิตใน A ไปทางซ้าย 1 บิตและบิต 0 เป็นค่าบิต ที่อยู่ในแฟล็ก Carry	1	1
RR A	หมุนบิตใน A ไปทางขวา 1 บิตและบิต 0 เป็นค่าจาก บิต 7	1	1
RRC A	หมุนบิตใน A ไปทางขวา 1 บิตและค่าจาก บิต 0 นำ ไปเก็บในแฟล็ก Carry และบิตที่อยู่ใน แฟล็ก Carry เดิมจะย้ายมาเก็บในบิต 7	1	1
SWAP A	สลับค่าสลับบิตบนกับสลับด้านล่างภายใน A	1	1

คำสั่งเคลื่อนย้ายข้อมูล (Data Transfer)

รูปแบบคำสั่ง	ความหมาย	จำนวน ไบต์	จำนวนเมฆ ซินไซเคิล
MOV A,Rn	ย้ายข้อมูลจาก Rn ไป A	1	1
MOV A,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป A	2	1
MOV A,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไป A	1	1
MOV A,#data	ย้ายค่าคงที่ 8 บิตไป A	2	1
MOV Rn,A	ย้ายข้อมูลจาก A ไป Rn	1	1
MOV Rn,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไป Rn	2	2
MOV direct,A	ย้ายข้อมูลจาก A ไปหน่วยความจำ direct	2	1
MOV direct,Rn	ย้ายข้อมูลจาก Rn ไปหน่วยความจำ direct	2	2
MOV direct,direct	ย้ายข้อมูลระหว่างหน่วยความจำภายใน	3	2
MOV direct,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไปยัง หน่วยความจำ direct	2	2
MOV direct,#data	ย้ายค่าคงที่ 8 บิตไปยังหน่วยความจำ direct	3	2
MOV @Ri,A	ย้ายข้อมูลใน A ไปยังหน่วยความจำที่เก็บอยู่ใน Ri	1	1
MOV @Ri,direct	ย้ายข้อมูลจากหน่วยความจำ direct ไปยังหน่วย ความจำที่เก็บอยู่ใน Ri	2	2
MOV @Ri,#data	ย้ายค่าคงที่ 8 บิตไปยังหน่วยความจำที่เก็บอยู่ใน Ri	2	1
MOV DPTR,#data16	ย้ายค่าคงที่ 16 บิตไปยัง DPTR	3	2
MOVC A,@A+DPTR	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพันธ์กับ DPTR ไป A	1	2
MOVC A,@A,PC	ย้ายข้อมูลจากหน่วยความจำข้อมูลที่สัมพันธ์กับ PC ไป A	1	2
MOVX A,@Ri	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน Ri ไปยัง A	1	2
MOVX A,@DPTR	ย้ายข้อมูลจากหน่วยความจำที่เก็บอยู่ใน DPTR ไป ยัง A	1	2
MOVX @Ri,A	ย้ายข้อมูลใน A ไปยังหน่วยความจำที่เก็บอยู่ใน Ri	1	2
MOVX @DPTR,A	ย้ายข้อมูลใน A ไปยังหน่วยความจำที่เก็บอยู่ใน DPTR	1	
PUSH direct	ย้ายข้อมูลจากหน่วยความจำ direct ไปยัง Stack	2	2

คำสั่งเคลื่อนย้ายข้อมูล (Data Transfer) ต่อ

POP	direct	ย้ายข้อมูลจาก Stack ไปยังหน่วยความจำ direct	2	2
XCH	A,Rn	แลกเปลี่ยนข้อมูลระหว่าง A กับ Rn	1	1
XCH	A,direct	แลกเปลี่ยนข้อมูลระหว่างหน่วยความจำ direct กับ A	2	1
XCH	A,@Ri	แลกเปลี่ยนข้อมูลระหว่างหน่วยความจำที่เก็บอยู่ Ri กับ A	1	1
XCHD	A,@Ri	แลกเปลี่ยนข้อมูลสลับคู่จากหน่วยความจำที่เก็บอยู่ Ri กับ A	1	1

คำสั่งจัดการข้อมูลแบบบิต (Boolean Variable Manipulation)

รูปแบบคำสั่ง		ความหมาย	จำนวน ไบต์	จำนวนเมก ไซเคิล
CLR	C	ทำค่าแฟล็ก Carry ให้เป็น 0	1	1
CLR	bit	ทำค่า bit ให้เป็น 0	2	1
SETB	C	ทำค่าแฟล็ก Carry ให้เป็น 1	1	1
SETB	bit	ทำค่า bit ให้เป็น 1	2	1
CPL	C	กลับค่าแฟล็ก Carry ให้เป็นตรงข้าม	1	1
CPL	bit	กลับค่า bit ให้เป็นตรงข้าม	2	1
ANL	C,bit	AND ค่า bit กับแฟล็ก Carry	2	2
ANL	C,/bit	AND ค่าตรงข้ามของ bit กับแฟล็ก Carry	2	2
ORL	C,bit	ORL ค่า bit กับแฟล็ก Carry	2	2
ORL	C,/bit	ORL ค่าตรงข้ามของ bit กับแฟล็ก Carry	2	2
MOV	C,bit	ย้ายค่า bit มายังแฟล็ก Carry	2	1
MOV	bit,C	ย้ายค่าแฟล็ก Carry มายัง bit	2	2
JC	ret	กระโดด ถ้าค่าแฟล็ก Carry เป็น 1	2	2
JNC	ret	กระโดด ถ้าค่าแฟล็ก Carry เป็น 0	2	2
JB	bit,ret	กระโดด ถ้าค่า bit เป็น 1	3	2
JNB	bit,ret	กระโดด ถ้าค่า bit เป็น 0	3	2
JBC	bit,ret	กระโดด ถ้าค่า bit เป็น 1 และเปลี่ยนค่า bit เป็น 0	3	2

คำสั่งควบคุมการทำงานของโปรแกรม (Program and Machine Control)

รูปแบบคำสั่ง	ความหมาย	จำนวน ไบต์	จำนวนเมฆ ซินไซเกิล
CLR C	ทำค่าแฟล็ก Carry ให้เป็น 0	1	1
ACALL addr 11	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 11 บิต	2	2
ACALL addr 16	ไปทำโปรแกรมย่อยจากค่าแอดเดรส 16 บิต	3	2
RET	คำสั่งสิ้นสุดการทำงานของโปรแกรมย่อย	1	2
RETI	คำสั่งสิ้นสุดการทำงานของโปรแกรมย่อย อินเตอร์รัปต์	1	2
AJMP addr 11	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 11 บิต	2	2
LJMP addr 16	กระโดดไปยังตำแหน่งจากค่าแอดเดรส 16 บิต	3	2
SJMP ret	กระโดดไปยังตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบัน	2	2
JMP @A+DPTR	กระโดดไปยังตำแหน่งที่สัมพันธ์กับ DPTR	1	2
JZ ret	กระโดดไปยังตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบันถ้า หากค่า A เป็น 0	2	2
JNZ ret	กระโดดไปยังตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบันถ้า หากค่า A เป็น 1	2	2
CJNE A,direct,ret	เปรียบเทียบ A กับหน่วยความจำ direct และ กระโดดไป ยังตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบันถ้าค่าไม่เท่ากัน	3	2
CJNE A,#data,ret	เปรียบเทียบ A กับหน่วยความจำคงที่และ กระโดดไปยัง ตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบันถ้าค่าไม่เท่ากัน	3	2
CJNE Rn,#data,ret	เปรียบเทียบ Rn กับหน่วยความจำคงที่และ กระโดดไปยัง ตำแหน่งที่สัมพันธ์กับตำแหน่งปัจจุบันถ้าค่าไม่เท่ากัน	3	2
CJNE @Ri,#data,ret	เปรียบเทียบค่าในหน่วยความจำที่เก็บใน Ri กับค่า คงที่ และกระโดดไปยังตำแหน่งที่สัมพันธ์กับตำแหน่ง ปัจจุบันถ้าค่าไม่เท่ากัน	3	2
DJNZ Rn,ret	ลดค่าใน Rn และ กระโดดไปยังตำแหน่งที่สัมพันธ์กับ ตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0	2	2
DJNZ direct,ret	ลดค่าในหน่วยความจำ direct และ กระโดดไปยังตำแหน่ง ที่สัมพันธ์กับตำแหน่งปัจจุบัน ถ้าค่าไม่เป็น 0	3	2
NOP	ไม่มีการทำงานใดๆเกิดขึ้น	1	1

MODULE DOMATRIX DISPLAY

MODULE DAC

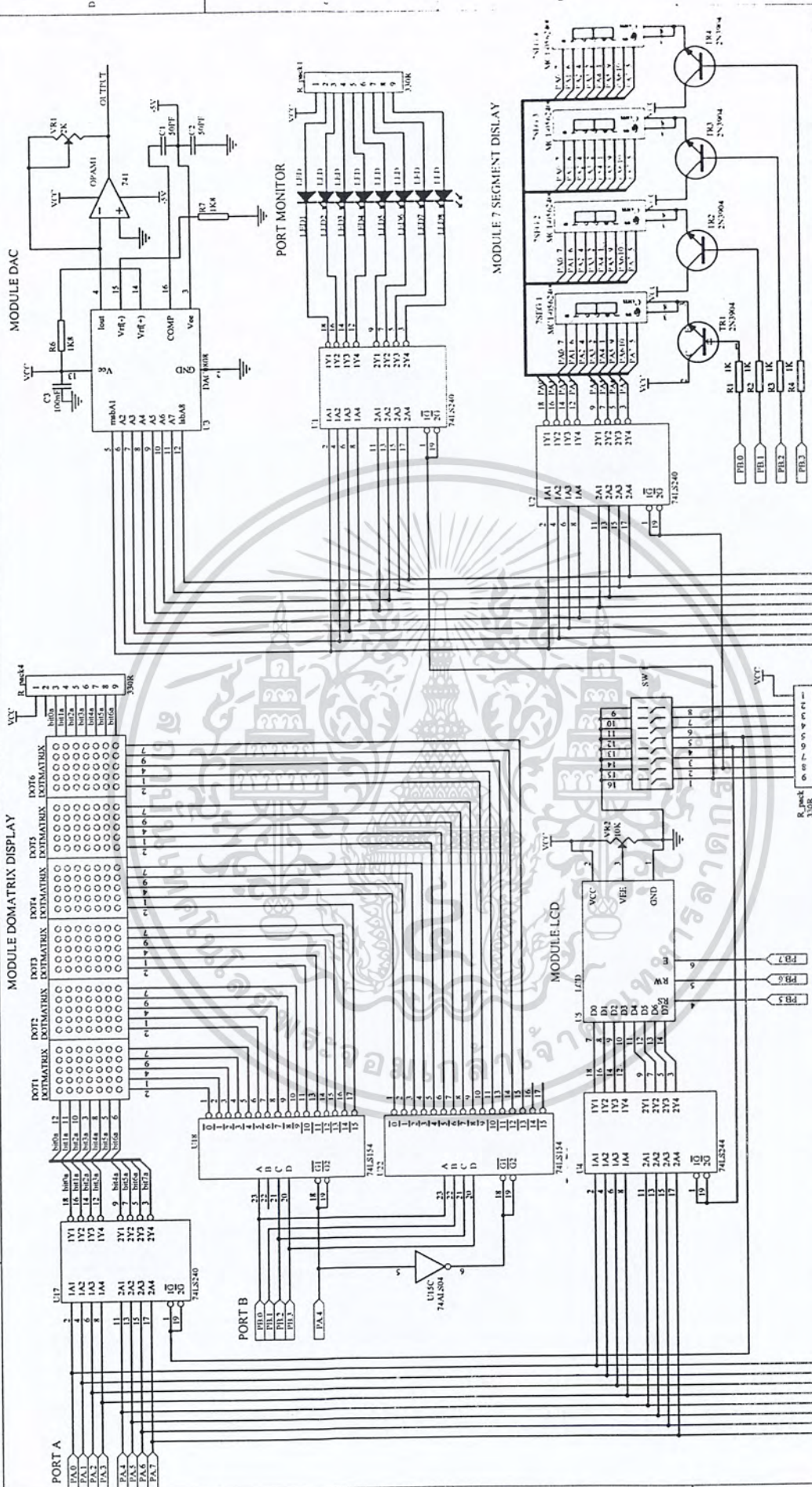
PORT MONITOR

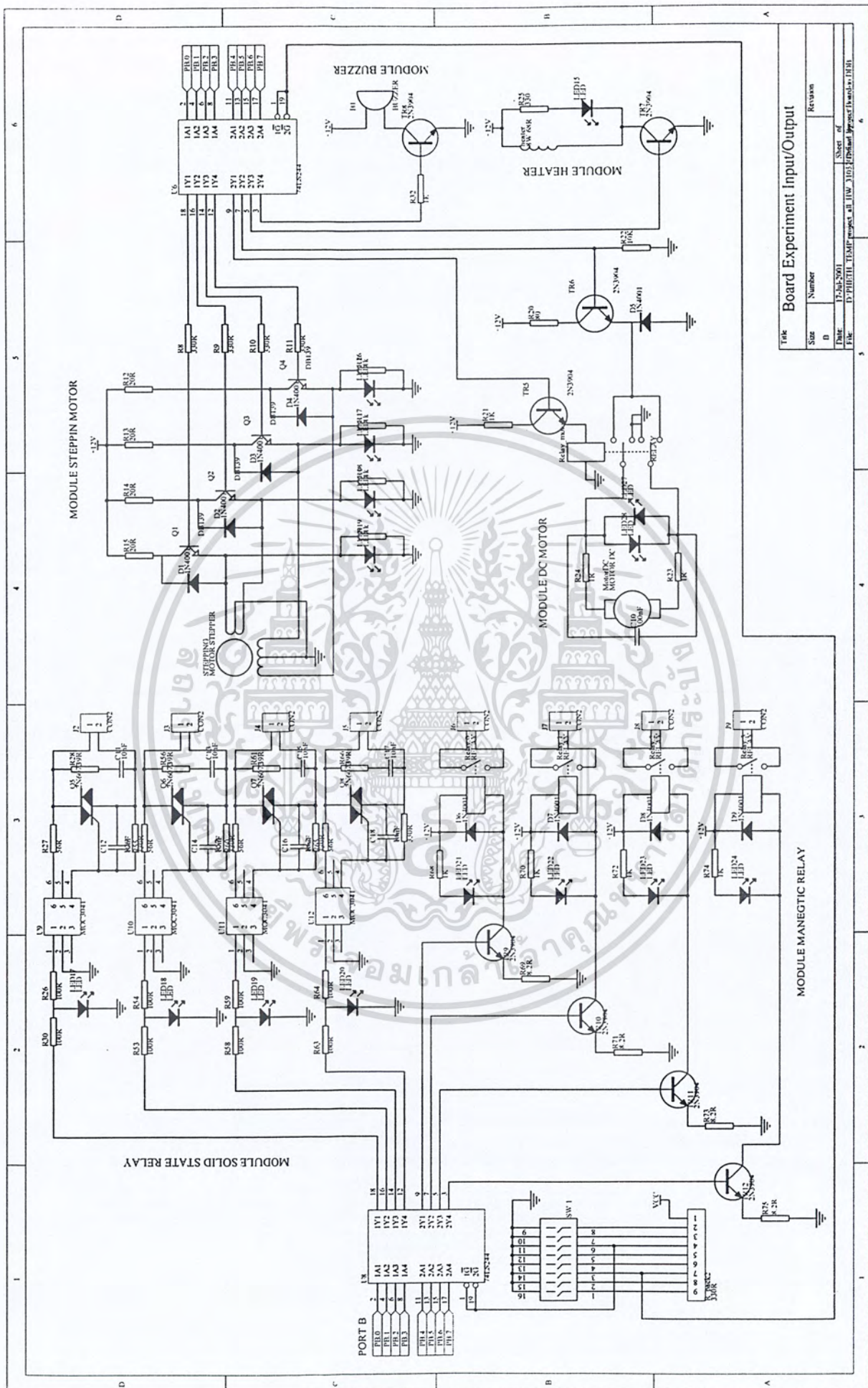
MODULE 7 SEGMENT DISPLAY

MODULE LCD

Board Experiment Input/Output

Title		Revision
Size	Number	Revision
Date	15-July-2001	Sheet of
Page	17/20	17/20

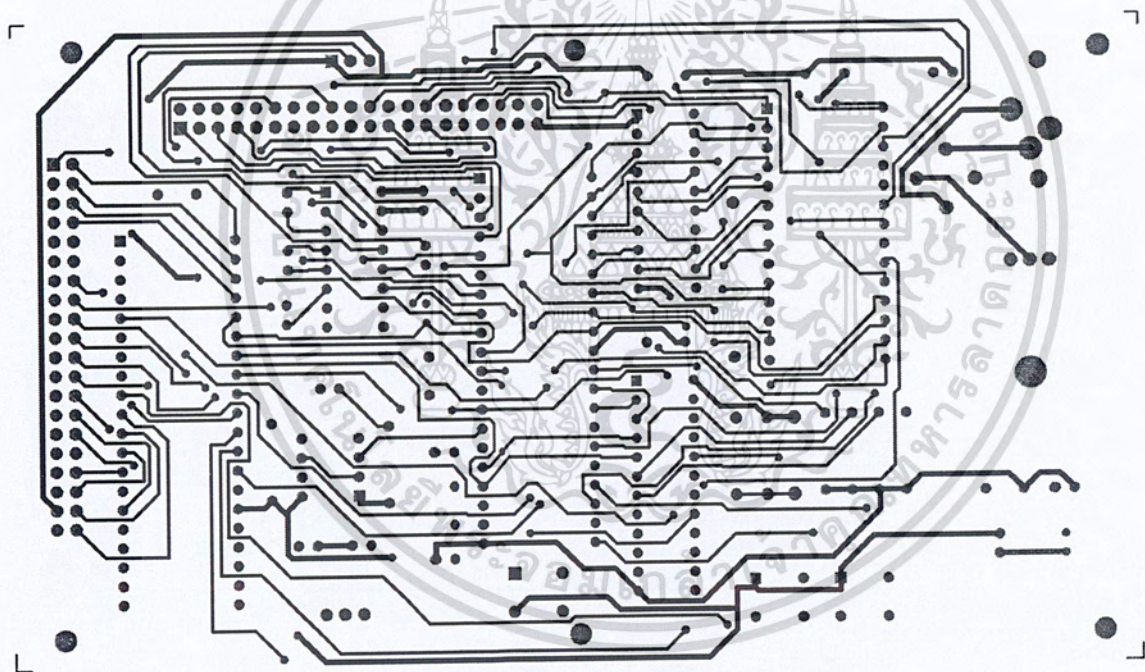




Task Board Experiment Input/Output			
Size	Number	Revision	
D			
Date	17-Jul-2001	Sheet of	4
File	D:\PHOTO\THER\project\input\board\input.hrb		

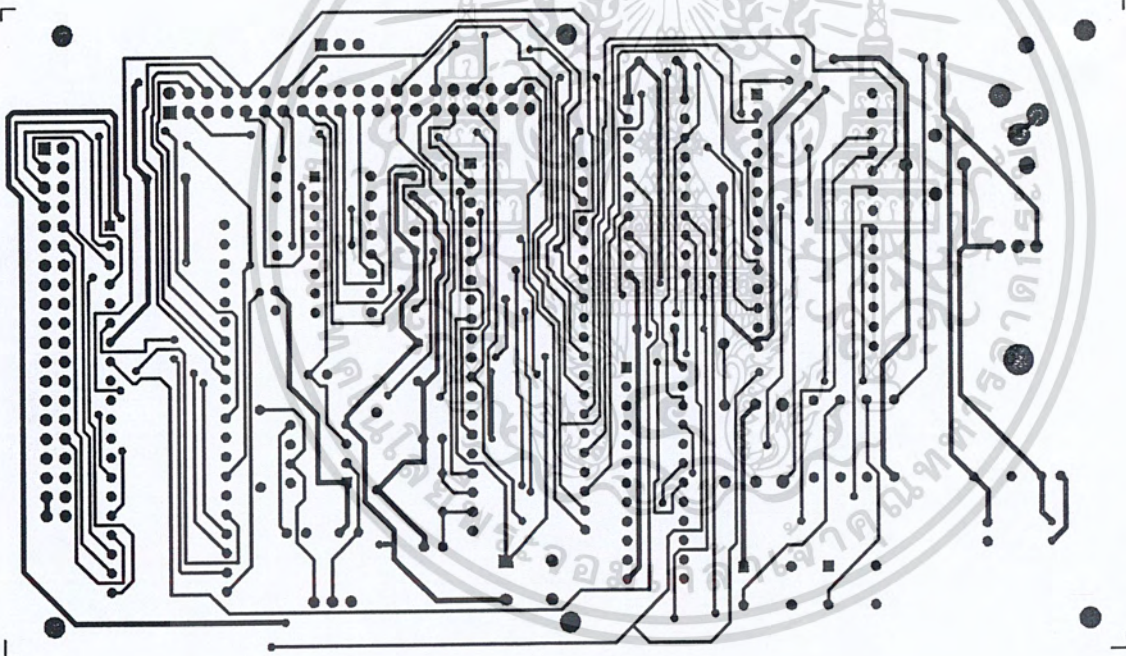
Board Microcontroller 8051

Top layer



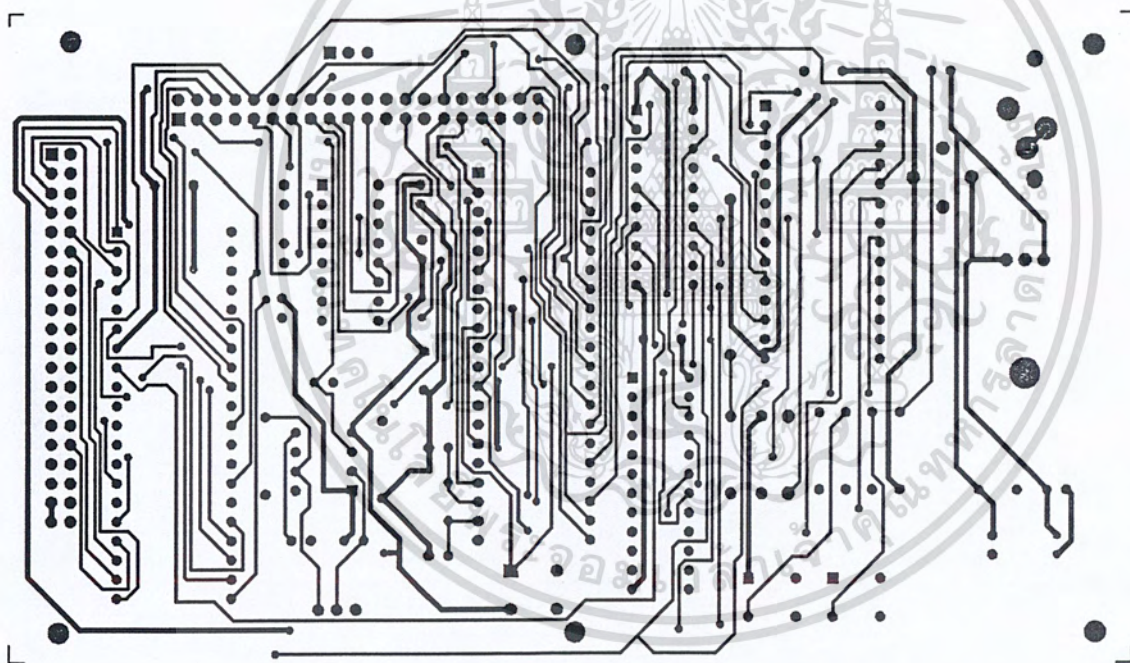
Board Microcontroller 8051

Bottom layer



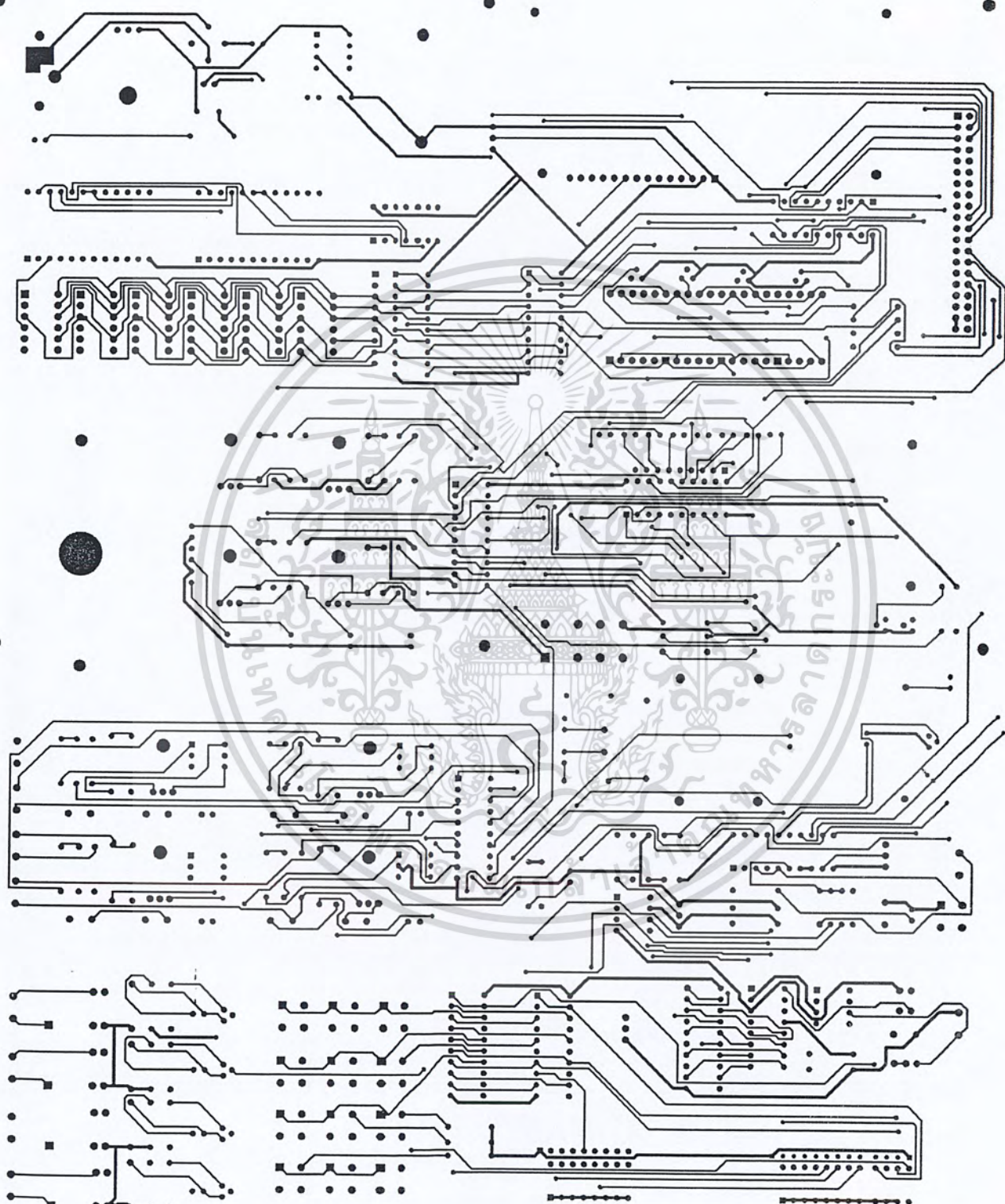
Board Microcontroller 8051

Bottom layer



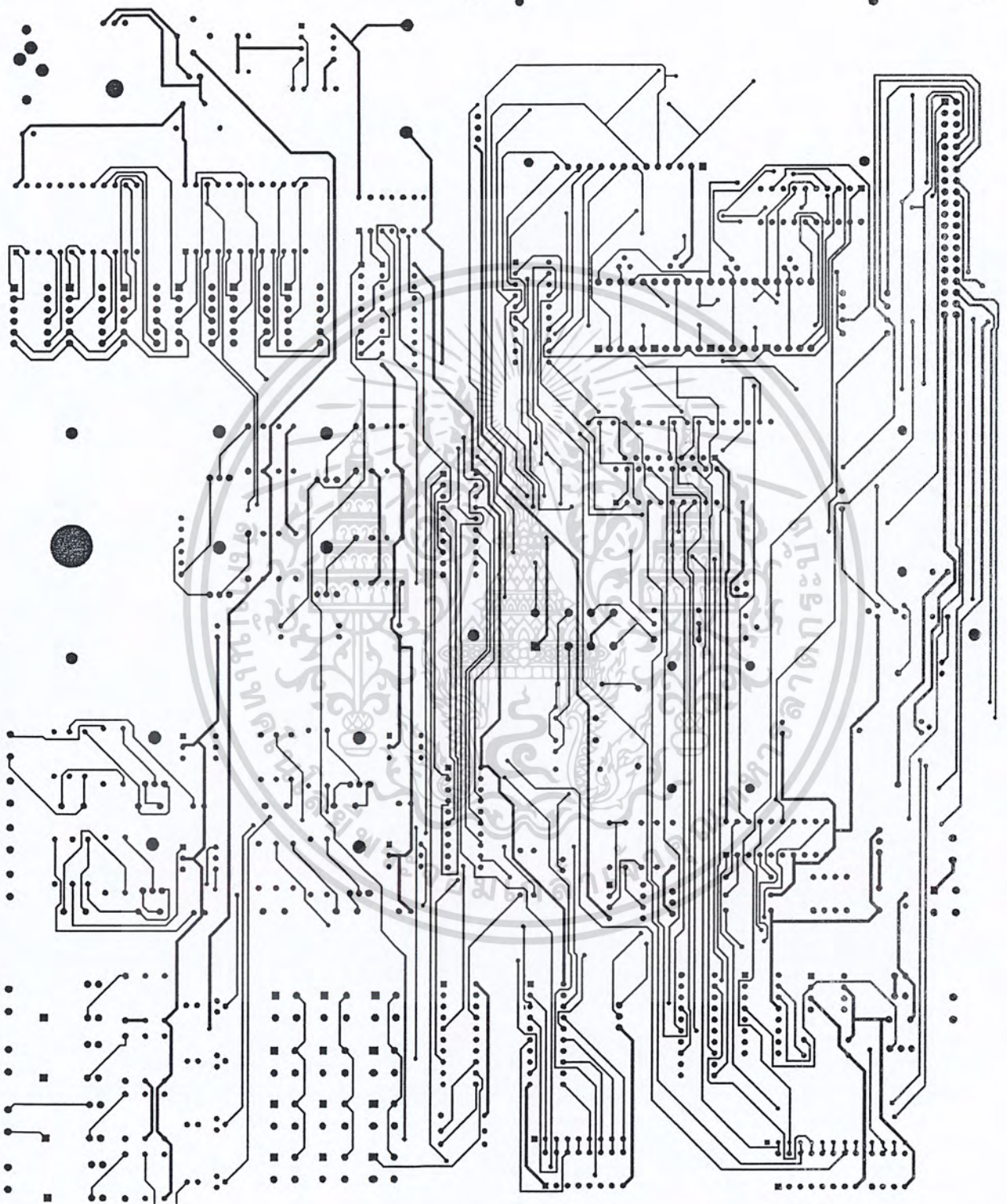
Experiment Input/Output Board

Top layer



Experiment Input/Output Board

Bottom layer



เอกสารอ้างอิง

1. หนังสือ Architectural overview of the MHS C51 family
2. ผศ. สมยศ จุณณะปิยะ , “การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 ”
ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2537
3. ชัยวัฒน์ ลิ่มพรจิตรวิไล และ วรพจน์ กรแก้ววัฒนกุล “เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51”
4. สุนทร วิทูรพจน์ “ การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051”
บริษัท ซีเอ็ดดูเคชั่น จำกัด (มหาชน)
5. “คู่มือการใช้งานและการทดลอง MI-1 Multi interface board (Z80 Version) ”
บ. ANA-DIGITAL R&D LABORATORY
6. “คู่มือการใช้งานและการทดลอง DB-1 Display Board (Z80 Version) ”
บ. ANA-DIGITAL R&D LABORATORY

