ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

Using Instruments Controller System in Project Room

ชื่อนักศึกษา

1. นายธรรมนูญ	กรเพ็ชร	รหัสประจำตัว	41031206
2. นางสาวศิรินันท์	พิลาแสน	รหัสประจำตัว	41031228
3. นายเสรี	ขุนไชย	รหัสประจำตัว	41031235
4. นายเอกพันธุ์	ชินกลาง	รหัสประจำตัว	41031237

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา

อาจารย์ที่ปรึกษาร่วม อาจารย์อมรชัย ชัยชนะ

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์อมรชัย	ชัยชนะ	.....
2. อาจารย์พีระวุฒิ	สุวรรณจันทร์	.....
3. ผศ.วิสุทธิ์	อธิพรธรรม	.....
4. อาจารย์โกศล	ตราชู	.....
5. อาจารย์ปิยะ	ศุภวราสุวัฒน์	.....

วัน/เดือน/ปีที่สอบ วันเสาร์ที่ 13 พฤษภาคม พ.ศ. 2543 เวลา 14.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 8 เดือน ๒๕ พ.ศ. ๒๕๔๓

ปริญญานิพนธ์

ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

USING INSTRUMENT CONTROLER SYSTEM IN PROJECT ROOM

นายธรรมนูญ กรเพ็ชร

นางสาวศิรินันท์ พิลาแสน

นายเสรี ขุนไชย

นายเอกพันธุ์ ชินกลาง

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

Using Instrument Controler System In Project Room

วัตถุประสงค์

1. เพื่อศึกษาระบบการใช้เครื่องมือในห้องปฏิบัติการโครงงาน
2. เพื่อวิเคราะห์ออกแบบวงจรควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน
3. เพื่อสร้างระบบความปลอดภัยในการใช้เครื่องมือในห้องปฏิบัติการโครงงาน
4. เมื่อเกิดความเสียหายกับเครื่องมือสามารถหาผู้รับผิดชอบได้
5. สามารถใช้งานเป็นประโยชน์แก่ส่วนรวมในภาควิชาครุศาสตร์วิศวกรรมได้

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถเข้าใจหลักการทำงานของระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน และเข้าใจการทำงานของไมโครคอนโทรลเลอร์
2. สามารถสร้างความเป็นระบบระเบียบในการใช้งานเครื่องมือในห้องปฏิบัติการโครงงาน
3. สามารถออกแบบวงจรที่ใช้ควบคุมการใช้งานเครื่องมือในห้องปฏิบัติการโครงงาน
4. สามารถสร้างระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน
5. สามารถทำการเพิ่มเครื่องมือในการใช้งานในห้องปฏิบัติการโครงงานเพื่อรองรับการพัฒนาเครื่องมือและอุปกรณ์ในอนาคตได้

ชื่อหัวข้อ	ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน
นักศึกษา	นายธรรมนูญ กรเพ็ชร
	นางสาวศิรินันท์ พิลาแสน
	นายเสรี ขุนไชย
	นายเอกพันธุ์ ชินกลาง
อาจารย์ที่ปรึกษา	ผศ. ดร.ธีระพล เทพหัสดิน ณ อยุธยา
อาจารย์ที่ปรึกษาร่วม	อาจารย์อมรชัย ชัยชนะ
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
ปีการศึกษา	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เสนอระบบควบคุมการใช้เครื่องในห้องปฏิบัติการโครงงาน โครงสร้างของระบบประกอบด้วยหน่วยประมวลผล ได้แก่ ไมโครคอนโทรลเลอร์ 2 ชุด คือ ทำหน้าที่ควบคุมรีเลย์จ่ายไฟ 16 ชุด และทำหน้าที่รับส่งข้อมูลโดยรับข้อมูลจากบาร์โค้ด คีย์เมทริกซ์สวิตช์ และส่งข้อมูลไปบันทึกไว้ยังชุดหน่วยความจำ ซึ่งจะทำการรับส่งข้อมูลแบบอนุกรมกับไมโครคอมพิวเตอร์โดยผ่านพอร์ต RS-458 อีกทีหนึ่ง เพื่อแสดงผลรายละเอียดการใช้งาน

Thesis Title	Using Instruments Controler System In Project Room	
Students	Mr. Thammanoon	Gornphet
	Miss Sirinan	Pilasan
	Mr. Saree	Khunchai
	Mr. Aekkapun	Chinkhang
Adivisor	Assist . Prof. Dr.Threrapon	Thephasadin na ayuthya
Co- Adivisor	Mr. Amornchai	Chaichana
Education Level	Bachelor of Science in Industrial Education	
Program in	Telecommunication Engineering	
Acadamic Year	1999	

ABSTRACT

This thesis persents terproject of Using instrument controler system in project room. The system consists 2 unit of Microcontroller. Fist units of Microtroller can be controllers 16 power supply units. Secound Microcontroller units can be communication data with read data from barcode or key matrix switchs and write data in ram . As microcontroller units can be connected to microcomputer unit via RS-485 . Microcomputer can be to make a speech user instruments in project room.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี ด้วยการช่วยเหลือและการให้คำแนะนำปรึกษา ตลอดจนแนวความคิดต่าง ๆ พร้อมทั้งข้อเสนอแนะแนวทางแก้ไขปัญหาในการดำเนินงาน รวมทั้งด้านเวลาและสถานที่ เครื่องมือ อุปกรณ์ ต่าง ๆ จากท่านอาจารย์ที่ปรึกษาปริญญานิพนธ์ และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่านรวมทั้งสมาชิกในกลุ่มที่รวมมือกันทำงานจนสำเร็จ

คณะผู้จัดทำขอกราบขอบพระคุณท่านบุพการีซึ่งให้โอกาส และให้การสนับสนุนในการศึกษามาโดยตลอด รวมทั้งคณาจารย์และเจ้าหน้าที่ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้กรุณาให้งบประมาณสนับสนุนในการจัดทำครั้งนี้ จึงทำให้ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ดี

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาคภาษาไทย	I
บทคัดย่อภาคภาษ:อังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	IX
สารบัญรูป	X
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญของโครงงาน	1
1.2 ขีดความสามารถของโครงงาน	1
1.3 เนื้อหาโดยสังเขป	1
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 กล่าวนำ	3
2.2 การสื่อสารข้อมูลแบบขนาน	3
2.3 การสื่อสารข้อมูลแบบอนุกรม	4
2.3.1 การสื่อสารข้อมูลแบบอนุกรมที่แบ่งตามทิศทางของข้อมูล	5
2.3.2 การสื่อสารข้อมูลแบบอนุกรมตามมาตราฐาน RS - 232 C	6
2.3.3 การสื่อสารทางเดียว	7
2.3.4 ฮาร์ดแวร์แฮนด์เช็คกิ้ง	8
2.3.5 จาก DTE ไปสู่ DCE	8
2.3.6 DCE ไปยัง DTE	9
2.3.7 การสื่อสารสองทาง	10
2.3.8 สัญญาณทางไฟฟ้า	11
2.3.9 RS - 449,442A และ 423 - A	12
2.3.10 RS - 449,442A และ RS - 232C	13
2.3.11 การเชื่อมต่อ RS - 232C และแอปเปิลแมคอินทอช	13
2.3.12 ความจำเป็นของนัลโมเด็ม	13

สารบัญ (ต่อ)

เรื่อง	หน้า
2.3.13 ปัญหาของแฮนด์เช็คกิ้ง	14
2.3.14 การเบรกเอาต์บ็อกซ์	14
2.3.15 การสื่อสารข้อมูลแบบอนุกรมด้วย RS - 485	16
2.3.16 มาตรฐาน RS - 422 / RS - 485	18
2.3.17 การใช้งาน PCMCIA RS - 422 / RS - 485	19
2.3.18 2 x RS - 422 และ RS - 485	19
2.3.19 การรับ - ส่งข้อมูล	20
2.4 บาร์โค้ด	20
2.5 ชนิดของตัวบาร์โค้ด	21
2.5.1 ไดโอดเปล่งแสง (LED)	21
2.5.2 แสงอินฟราเรด (IR)	22
2.5.3 แสงเลเซอร์ (LASER)	22
2.5.4 ไฟเบอร์ออปติก	22
2.6 การทำงานของบาร์โค้ด	23
2.7 ผลของช่องรับแสง	25
2.8 รูปแบบของรหัสบาร์โค้ด	26
2.8.1 รหัส 3 ใน 9 หรือรหัส 39	26
2.8.2 รหัสแทรก 2 ใน 5	29
2.8.3 รหัสแบบ Codebar	32
2.8.4 รหัสสากล UPC (Universal Product Code)	33
2.8.5 รหัสสากล UPC (Universal Product Code)	33
2.8.6 รหัสตัวเลขของยุโรป EAN (European Article Numbering)	35
2.9 การอินเตอร์เฟสกับคีย์บอร์ด	37
2.9.1 แบบเชื่อมต่อสวิตท์โดยตรงกับพอร์ต	37
2.9.2 การเชื่อมต่อสวิตท์แบบเมทริกซ์	38
2.9.3 การเชื่อมต่อสวิตท์ผ่านรีจิสเตอร์	39

สารบัญ (ต่อ)

เรื่อง	หน้า
2.9.4 การเชื่อมต่อสวิตท์แบบมัลติเพล็กซ์	40
2.9.5 แบบเชื่อมต่อสวิตช์เข้ากับระบบบัสข้อมูล	40
2.10 การติดต่อกับ 8255	40
2.10.1 ลักษณะพื้นฐานของ 8255	40
2.10.2 รายละเอียดการจัดขาของ 8255	41
2.10.3 การเชื่อมต่อไมโครคอนโทรลเลอร์ 8051 กับ 8255	42
2.11 คุณสมบัติของไมโครคอนโทรลเลอร์ 8051 กับ 8255	46
2.12 โครงสร้างภายนอกของ MCS -51	47
2.13 โครงสร้างภายในของ MCS -51	50
2.14 ANT - 31PJ Version	50
2.15 รายละเอียดต่างๆ บนคีย์บอร์ด	53
2.15.1 8255 Input / Output Port	53
2.15.2 จอแสดงผลผลึกเหลว (LCD) Port	54
2.15.3 วงจร Reset และ Watch	55
2.15.4 Serial Port	55
2.15.5 Working Area PCB	56
2.15.6 แนวทางการพัฒนาโปรแกรม	56
บทที่ 3 การออกแบบ การสร้าง และการทำงาน	58
3.1 กล่าวนำ	58
3.2 การออกแบบวงจรควบคุม	59
3.2.1 ส่วนของวงจรควบคุม	59
3.2.2 ส่วนของวงจรรับข้อมูล	60
3.2.3 ส่วนของการแสดงผล	64
3.2.4 สัญญาณเสียง	65
3.2.5 ส่วนของวงจรควบคุมการจ่ายไฟ	65
3.2.6 ส่วนการสื่อสารข้อมูล	66

สารบัญ (ต่อ)

เรื่อง	หน้า
3.2.7 ส่วนของวงจรแสดงผลเวลา	70
บทที่ 4 การทดลองและผลการทดลอง	71
4.1 การทดลองส่วนควบคุมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล	71
4.1.1 ลำดับขั้นการทดลอง	71
4.1.2 ผลการทดลอง	71
4.2 การทดลองส่วนการแสดงผลผลึกเหลว (LCD)	72
4.2.1 ลำดับขั้นตอนการทดลอง	72
4.2.2 ผลการทดลอง	72
4.3 การทดลองส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด	73
4.3.1 ลำดับขั้นตอนการทดลอง	73
4.3.2 ผลการทดลอง	73
4.4 การทดลองการรับข้อมูลจากคีย์เมทริกซ์	74
4.4.1 ลำดับขั้นตอนการทดลอง	74
4.4.2 ผลการทดลอง	74
4.5 การทดลองการควบคุมวงจรการจ่ายไฟให้กับโต๊ะอุปกรณ์	75
4.5.1 ลำดับขั้นตอนการทดลอง	75
4.5.2 ผลการทดลอง	75
4.6 การทดลองส่วนของการสื่อสารข้อมูล	76
4.6.1 ลำดับขั้นตอนการทดลอง	76
4.6.2 ผลการทดลอง	76
4.7 การทดลองส่วนของนาฬิกาดิจิตอล	79
4.7.1 ลำดับขั้นตอนการทดลอง	79
4.7.2 ผลการทดลอง	79
4.8 ส่วนของวงจรรวมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล	80
4.8.1 ลำดับขั้นตอนการทดลอง	80
4.8.2 ผลการทดลอง	80

สารบัญ (ต่อ)

เรื่อง	หน้า
บทที่ 5 บทสรุป ปัญหา แนวทางการแก้ไข และการพัฒนา	83
5.1 สรุป	83
5.2 ปัญหาของโครงงาน	83
5.3 แนวทางการแก้ปัญหา	84
5.4 แนวทางการพัฒนาโครงงาน	84
ภาคผนวก ก เครื่องต้นแบบ	85
ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์	88
ภาคผนวก ค ผังการทำงานและโปรแกรมการทำงาน	98
ภาคผนวก ง รายละเอียดของอุปกรณ์	154
ภาคผนวก จ คู่มือการใช้งานระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน	181
บรรณานุกรม	187
ประวัติผู้แต่ง	188

สารบัญตาราง

ตาราง	หน้า
ตารางที่ 2.1 การเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA	17
ตารางที่ 2.2 ตารางเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA	18
ตารางที่ 2.3 ชนิดของตัวอ่านบาร์โค้ด	22
ตารางที่ 2.4 ผลของช่องแสงที่มีแรงดันต่อเอาต์พุต	24
ตารางที่ 2.5 โครงสร้างตัวอักขระในรหัส	28
ตารางที่ 2.6 ชุดอักขระของบาร์โค้ดรหัสแทรก 2 ใน 5	29
ตารางที่ 2.7 ระบบจำนวนของตัวอักขระ	34
ตารางที่ 2.8 ตัวเลขของรหัสสากล	24
ตารางที่ 2.9 แอดเดรสของพอร์ตและรีจิสเตอร์ของ 8255	32
ตารางที่ 2.10 การทำงานของ 8255 เมื่อสัญญาณ $\overline{RD}$ และ $\overline{WR}$ เป็นค่าต่าง ๆ	44
ตารางที่ 2.11 แอดเดรสของรีจิสเตอร์ภายใน 8255	45
ตารางที่ 2.12 คุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS-51	46
ตารางที่ 2.13 หน้าที่พิเศษของแต่ละขาพอร์ต P3	49
ตารางที่ 2.14 คุณสมบัติของบอร์ด ANT-31PJ Version 2.0	52
ตารางที่ 2.15 หน่วยความจำของบอร์ด ANT-31PJ	53
ตารางที่ 2.16 ตำแหน่งแอดเดรสของ 8255	54
ตารางที่ 2.17 Memory Map	54
ตารางที่ 4.1 การแสดงผลของ LED เมื่อมีการกดคีย์เมทริกซ์สวิตช์	84

สารบัญรูป

รูป	หน้า
รูปที่ 2.1 โครงสร้างการสื่อสารข้อมูลแบบขนาน	3
รูปที่ 2.2 โครงสร้างการสื่อสารข้อมูลแบบอนุกรม	4
รูปที่ 2.3 รูปแบบของการติดต่อสื่อสารข้อมูลแบบอนุกรม	5
รูปที่ 2.4 การเชื่อมต่อทางเดียวอย่างง่าย	8
รูปที่ 2.5 การสื่อสารทางเดียวพร้อมด้วยแฮนด์เช็คกิ้งจาก DET ไปยัง DCE	9
รูปที่ 2.6 การสื่อสารทางเดียวพร้อมด้วยแฮนด์เช็คกิ้งจาก DET ไปยัง DTE	10
รูปที่ 2.7 การสื่อสารสองทางพร้อมด้วยวงจรแฮนด์เช็คกิ้งหลัก	11
รูปที่ 2.8 การเชื่อมต่อ RS-232 แบบมาตรฐานเก้าเส้น	11
รูปที่ 2.9 หัวต่อแบบ DB-9 ในไอบีเอ็มพีซีเอที	14
รูปที่ 2.10 การเชื่อมต่อ DET (ไอบีเอ็มพีซี) ไปยัง DCE (โมเด็ม) แบบมาตรฐาน	15
รูปที่ 2.11 การเชื่อมต่อแบบนัลโมเด็ม	15
รูปที่ 2.12 โครงสร้างของการสื่อสารข้อมูลอนุกรมแบบ RS-485	16
รูปที่ 2.13 ระดับแรงดันทางเอาต์พุตของตัวอ่านบาร์โค้ด Vws คือแรงดันเมื่อผ่านแถบขาว และ Vbs คือแรงดันเมื่อผ่านแถบดำ	23
รูปที่ 2.14 รูปแบบของการเข้ารหัส LA	25
รูปที่ 2.15 อักขระในบาร์โค้ด 3 ใน 9	26
รูปที่ 2.16 รูปแบบของบาร์โค้ดรหัสแทรก 2 ใน 5	27
รูปที่ 2.17 ผังการจัดวางบาร์โค้ดรหัสแทรก 2 ใน 5	30
รูปที่ 2.18 บาร์โค้ดสมบูรณ์ของจำนวน 0123 ในรูปแบบรหัสแทรก 2 ใน 5	31
รูปที่ 2.19 การเข้ารหัส Codebar A37859B	32
รูปที่ 2.20 รูปแบบการเข้ารหัส UPC	33
รูปที่ 2.21 ผังการจัดวางข้อมูลสตริงของบาร์โค้ดรหัส UPC	35
รูปที่ 2.22 ฟิลด์ซ้ายและฟิลด์ขวาของข้อมูลตัวอักขระ	36
รูปที่ 2.23 บาร์โค้ดรหัส EA	37
รูปที่ 2.24 การต่อเชื่อมสวิตช์แบบต่าง ๆ	38
รูปที่ 2.25 ผังการทำงานภายในไอซีเบอร์	43

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 2.26 ขาสัญญาณต่าง ๆ ของไอซีเบอร์ 8255	44
รูปที่ 2.27 การเชื่อมต่อระหว่าง 8255 กับไมโครคอนโทรลเลอร์ 8051	45
รูปที่ 2.28 การจัดตำแหน่งขาต่าง ๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51	48
รูปที่ 2.29 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51	50
รูปที่ 2.30 Memory Map และ I/O Map	57
รูปที่ 3.1 โครงสร้างการทำงานของชุดควบคุมการจ่ายไฟอัตโนมัติ	58
รูปที่ 3.2 วงจรควบคุม	60
รูปที่ 3.3 ชุดเครื่องอ่านบาร์โค้ด	61
รูปที่ 3.4 วงจรส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด	62
รูปที่ 3.5 คีย์เมทริกซ์สวิตช์	63
รูปที่ 3.6 ส่วนรับข้อมูลเมทริกซ์สวิตช์	64
รูปที่ 3.7 ลักษณะแสดงผลของจอแสดงผลผลึกเหลว (LCD)	65
รูปที่ 3.8 วงจรสร้างสัญญาณเสียง เมื่อมีการรูดบัตร	65
รูปที่ 3.9 ตัวอย่างวงจรควบคุมการจ่ายไฟโดยใช้รีเลย์ 1 ชุด	66
รูปที่ 3.10 วงจรควบคุมการจ่ายไฟ	67
รูปที่ 3.11 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ต่ออยู่กับไมโครคอนโทรลเลอร์	68
รูปที่ 3.12 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ต่ออยู่กับเครื่องอ่านบาร์โค้ด	68
รูปที่ 3.13 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ต่ออยู่กับไมโครคอมพิวเตอร์	69
รูปที่ 3.14 ชิป RTC เบอร์ DS1202 ต่อเข้าพอร์ต 1 ของ 8031	70
รูปที่ 4.1 บอร์ดวงจรควบคุมที่สามารถทำงานได้	72
รูปที่ 4.2 การทดลองนำข้อมูลแสดงผลที่จอแสดงผลผลึกเหลว (LCD)	73
รูปที่ 4.3 การแสดงผลของ LED เมื่อกดคีย์	74
รูปที่ 4.4 วงจรควบคุมการจ่ายไฟ	76
รูปที่ 4.5 การทดลองเครื่องสื่อสารผ่าน RS-485	77
รูปที่ 4.6 วงจรสื่อสารข้อมูล	78
รูปที่ 4.7 วงจรนาฬิกา DS1202	79
รูปที่ 4.8 วงจรรวมการรับข้อมูล และแสดงผล	80

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ 4.9 วงจรรวมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล	80
รูปที่ 4.10 การต่อวงจรทั้งหมดเข้าด้วยกัน	82
รูปที่ 4.11 ผลที่ได้จากการรูดบัตรบาร์โค้ด	82
รูปที่ ก.1 แผงวงจรรวมชุดควบคุมการจ่ายไฟสำหรับห้องปฏิบัติการโครงงาน	86
รูปที่ ก.2 การต่อแผงวงจรรวมเข้ากับไมโครคอมพิวเตอร์	86
รูปที่ ก.3 การต่อส่วนสื่อสารข้อมูลเข้ากับไมโครคอมพิวเตอร์	87
รูปที่ ก.4 ลักษณะหน้าจอมอนิเตอร์ที่ใช้ควบคุมระบบ	87
รูปที่ ข.1 วงจรรวมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล	89
รูปที่ ข.2 ด้านวางอุปกรณ์ของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	90
รูปที่ ข.3 ด้านบนของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	90
รูปที่ ข.4 ด้านล่างอุปกรณ์ของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	91
รูปที่ ข.5 ด้านวางอุปกรณ์ของวงจรควบคุมรีเลย์	91
รูปที่ ข.6 ด้านบนของวงจรควบคุมรีเลย์	92
รูปที่ ข.7 ด้านล่างวงจรควบคุมรีเลย์	92
รูปที่ ข.8 ด้านวางอุปกรณ์ของวงจรรีเลย์	93
รูปที่ ข.9 ด้านบนของวงจรรีเลย์	93
รูปที่ ข.10 ด้านล่างของวงจรรีเลย์	94
รูปที่ ข.11 ด้านวางอุปกรณ์ของวงจรเรกกูเลย์เตอร์	94
รูปที่ ข.12 ด้านบนของวงจรเรกกูเลย์เตอร์	95
รูปที่ ข.13 ด้านล่างของวงจรเรกกูเลย์เตอร์	95
รูปที่ ข.14 ด้านวางอุปกรณ์ของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	96
รูปที่ ข.15 ด้านบนของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	96
รูปที่ ข.16 ด้านล่างของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED	97
รูปที่ ค.1 ผังการทำงานของโปรแกรม	99
รูปที่ ค.2 โปรแกรมควบคุมระบบการใช้เครื่องมือในห้องปฏิบัติการโครงงาน	100
รูปที่ จ.1 การจัดโต๊ะเดินสายไฟพร้อมชั้นวางอุปกรณ์	182
รูปที่ จ.2 การติดตั้งชุดอ่านบาร์โค้ดและตัวเก็บข้อมูล	183

สารบัญรูป (ต่อ)

รูป	หน้า
รูปที่ จ.3 การติดตั้งชุดควบคุมรีเลย์ 16 ช่อง	183
รูปที่ จ.4 แสดงหน้าจอคอมพิวเตอร์เมื่อเข้าสู่ระบบ	185

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

เนื่องจากปัจจุบันห้องปฏิบัติการโครงงานของภาควิชาครุศาสตร์วิศวกรรมมีไว้ใช้งานส่วนรวม และไม่มีความเป็นระบบระเบียบในการจัดเก็บและการใช้งานเครื่องมือเท่าที่ควร ซึ่งปัญหาการใช้งานห้องปฏิบัติการโครงงานที่พบเห็นมีอยู่มากมาย ไม่ว่าจะเป็นการใช้งานไม่ถูกวิธี หาผู้รับผิดชอบไม่ได้เมื่อเครื่องมือหรืออุปกรณ์เกิดความเสียหาย สำหรับห้องปฏิบัติการโครงงานนี้ จะช่วยลดปัญหาดังกล่าว และหาผู้รับผิดชอบได้จากสาเหตุการสูญหายของเครื่องมือ และการชำรุดของเครื่องมือได้

1.2 ขีดความสามารถของโครงงาน

โครงงานเรื่อง ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน สามารถจำแนกขอบเขตได้เป็นข้อ ๆ ดังนี้

1. สามารถเลือกป้อนรหัสเป็นบัตรบาร์โค้ดหรือกดปุ่ม
2. สามารถควบคุมการจ่ายไฟได้
3. สามารถแสดงรหัสผู้ต้องการใช้เครื่องมือและหมายเลขเครื่องที่ใช้งาน ออกที่อุปกรณ์แสดงผลผลึกเหลว (LCD) ได้
4. สามารถบันทึกรหัสผู้ใช้เครื่องมือพร้อมทั้งเวลาที่ใช้เครื่องมือได้

1.3 เนื้อหาโดยสังเขป

ปริญญานิพนธ์ฉบับนี้มีเนื้อหาทั้งหมด 5 บทดังต่อไปนี้

บทที่ 1 บทนำ ซึ่งเป็นเนื้อหาเกี่ยวกับ ความเป็นมา และความสำคัญของปัญหาที่ต้องทำให้เกิดโครงงานนี้ขึ้นรวมทั้งยังกล่าวถึงวัตถุประสงค์ ขอบเขตและประโยชน์ของปริญญานิพนธ์ในครั้งนี้

บทที่ 2 ทฤษฎีและหลักการ จะกล่าวถึงเนื้อหาที่นำมาอ้างอิงและเป็นแนวทางในการออกแบบสร้างชุดควบคุมการจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงาน

บทที่ 3 การออกแบบและการสร้างจะเป็นเนื้อหาโดยละเอียดตั้งแต่ขั้นตอนในการออกแบบวงจรในส่วนต่าง ๆ การนำส่วนต่าง ๆ มาอินเตอร์เฟสกัน เพื่อให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

บทที่ 4 การทดลองและผลการทดลอง ในบทนี้เป็นการนำเสนอการทดลองและผลการทดลอง โดยแบ่งการทดลองออกเป็นส่วน ๆ ตามการออกแบบและการสร้าง พร้อมบันทึกผลการทดลองในแต่ละส่วน

บทที่ 5 บทสรุป ปัญหาแนวทางแก้ไข และพัฒนา ซึ่งเป็นการสรุปผลเกี่ยวกับความสามารถประสิทธิภาพการทำงานของชุดจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงาน และกล่าวถึงปัญหาที่เกิดขึ้น นับตั้งแต่การเริ่มสร้างโครงงานจนกระทั่งโครงงานเสร็จสมบูรณ์ตลอดจนแนวทางแก้ไขปัญหาที่เกิดขึ้น พร้อมทั้งเสนอแนวทางการพัฒนาชุดควบคุมการจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงานให้สามารถประยุกต์ใช้งานได้อย่างกว้างขวาง และ ปรับปรุงให้มีประสิทธิภาพยิ่งขึ้น

ภาคผนวก ก รูปต้นแบบ

ภาคผนวก ข วงจรและแผ่นวงจรพิมพ์

ภาคผนวก ค ผังการทำงานและโปรแกรมการทำงาน

ภาคผนวก ง รายละเอียดของอุปกรณ์

ภาคผนวก จ คู่มือการใช้งานระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

บทที่ 2
ทฤษฎีและหลักการ

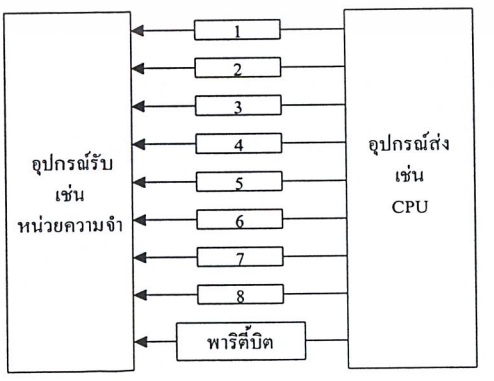
2.1 กล่าวนำ

โดยทั่ว ๆ ไป หลักใหญ่ของการส่งข้อมูลในคอมพิวเตอร์ หรือระหว่างคอมพิวเตอร์ด้วยกันจะมีลักษณะการส่งหรือการสื่อสารข้อมูลอยู่ 2 ลักษณะ คือ การสื่อสารข้อมูลแบบขนาน และการ สื่อสารข้อมูลแบบอนุกรม ซึ่งแต่ละลักษณะจะมีรายละเอียดดังนี้ คือ

2.2 การสื่อสารข้อมูลแบบขนาน

ลักษณะของการสื่อสารข้อมูลแบบขนานนั้น จะเป็นการสื่อสารข้อมูลที่ข้อมูลจะรับ - ส่งโดยผ่านสายหรือช่องสัญญาณพร้อมกันหลาย ๆ เส้น ดังแสดงในรูปที่ 2.1 โดยที่จำนวนของสัญญาณจะมีจำนวนไม่แน่นอนต้องขึ้นอยู่กับโครงสร้างการประมวลผลข้อมูลระบบนั้น ๆ ข้อดีของการสื่อสารข้อมูลแบบนี้ คือ สามารถสื่อสารข้อมูลกันได้ในระยะเวลาสั้น ๆ แต่ก็มีข้อเสีย คือจะสิ้นเปลืองสายสัญญาณเป็นจำนวนมากและถ้ายิ่งใช้ในการสื่อสารข้อมูลในระยะทางไกล ๆ นอกจากจะสิ้นเปลืองค่าใช้จ่ายจำนวนมากแล้ว ยังทำให้สัญญาณถูกลดทอนไปด้วย

โดยทั่วไปแล้วการสื่อสารข้อมูลแบบขนาน นิยมนำไปใช้กับการสื่อสารข้อมูลในระยะเวลาสั้น ๆ ที่ต้องการสื่อสารข้อมูลด้วยอัตราเร็ว เช่นการเชื่อมต่อของสัญญาณ ระหว่างหน่วยประมวลผลกลางกับอุปกรณ์รอบข้าง หรือ การสื่อสารข้อมูลของเครื่องมือวัด และ อุปกรณ์ต่าง ๆ กับเครื่องไมโครคอมพิวเตอร์



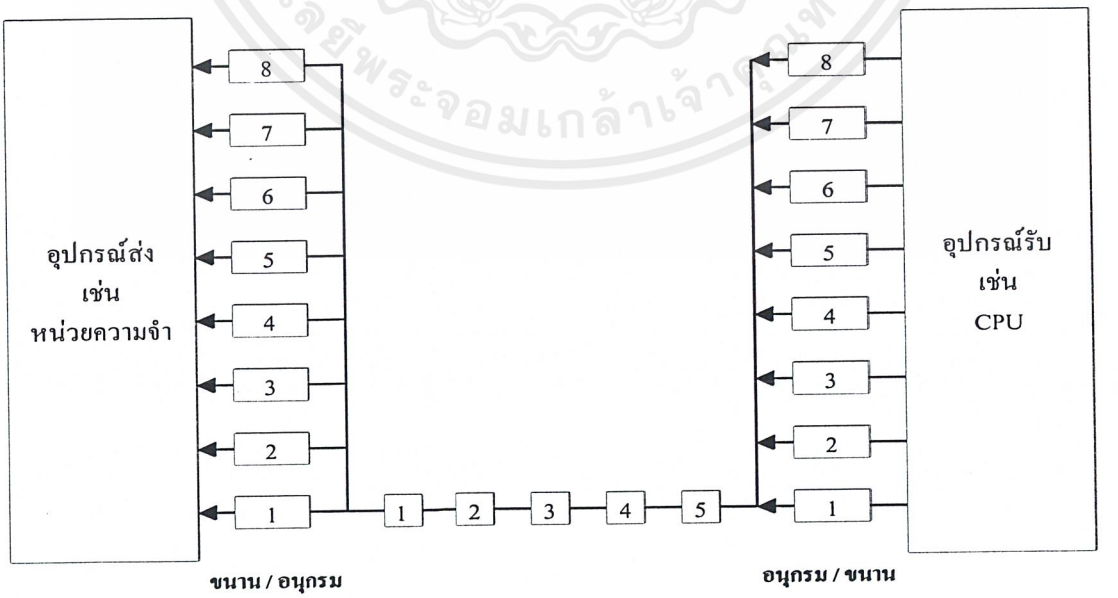
รูปที่ 2.1 โครงสร้างของการสื่อสารข้อมูลแบบขนาน

2.3 การสื่อสารข้อมูลแบบอนุกรม

ลักษณะของการสื่อสารข้อมูลแบบอนุกรมนั้น จะเป็นการสื่อสารที่จะทำการรับ - ส่ง ข้อมูลโดยใช้สายจำนวนน้อย ซึ่งปกติจะใช้เพียง 1 คู่ เท่านั้น คือ สายสัญญาณที่จะใช้เป็นสายข้อมูลและสายกราวด์

ลักษณะของการรับ-ส่งข้อมูลนั้น ข้อมูลจะถูกส่งออกไปหรือรับเข้ามา ในลักษณะที่เป็นบิตต่อบิต ซึ่งถ้าหากเปรียบเทียบกันกับการสื่อสารแบบขนานที่จำนวนข้อมูล และอัตราเร็วในการสื่อสารข้อมูลเท่ากันแล้วจะพบว่าการสื่อสารข้อมูลแบบอนุกรมจะต้องใช้เวลาในการรับ-ส่ง ข้อมูลมากกว่าอย่างแน่นอนแต่เมื่อพิจารณาข้อดีของการสื่อสารข้อมูลแบบอนุกรมแล้ว จะพบว่าข้อดีของการสื่อสาร ข้อมูลแบบอนุกรมคือ การใช้สายสัญญาณน้อยกว่าและ สามารถส่งสัญญาณในระยะทางที่ไกลกว่า แม้ว่าอัตราการลดทอนหรือผิดเพี้ยนของสัญญาณที่มีผล จากความยาวของสายสัญญาณจะมีค่าเท่ากับการสื่อสารข้อมูลแบบขนาน การสื่อสารข้อมูลแบบอนุกรม จะมีวิธีในการที่จะลดผลจากการลดทอนของสัญญาณนี้โดยอาศัยหลักการรับ-ส่งสัญญาณ แบบดิฟเฟอเรนเชียล

ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงเหมาะสำหรับใช้กับการสื่อสารข้อมูลในระยะ ไกล หรือการสื่อสารที่ต้องใช้สายหรือช่องสัญญาณในการรับ - ส่งข้อมูลจำนวนน้อย เช่น การสื่อสารข้อมูลโครงข่ายแบบท้องถิ่น (Local Area Network : LAN)



รูปที่ 2.2 โครงสร้างของการสื่อสารข้อมูลแบบอนุกรม

2.3.1 การสื่อสารข้อมูลแบบอนุกรมที่แบ่งตามทิศทางของข้อมูล

นอกจากที่ได้กล่าวมาแล้วนั้นจะพบว่า การสื่อสารข้อมูลแบบอนุกรมยังสามารที่จะแบ่งตามลักษณะของทิศทางในการสื่อสารข้อมูล ตามโครงสร้าง และความต้องการของระบบได้ดังต่อไปนี้

1) การสื่อสารข้อมูลในทิศทางเดียวตลอดเวลาหรือแบบซิมเพล็กซ์ (Simplex)

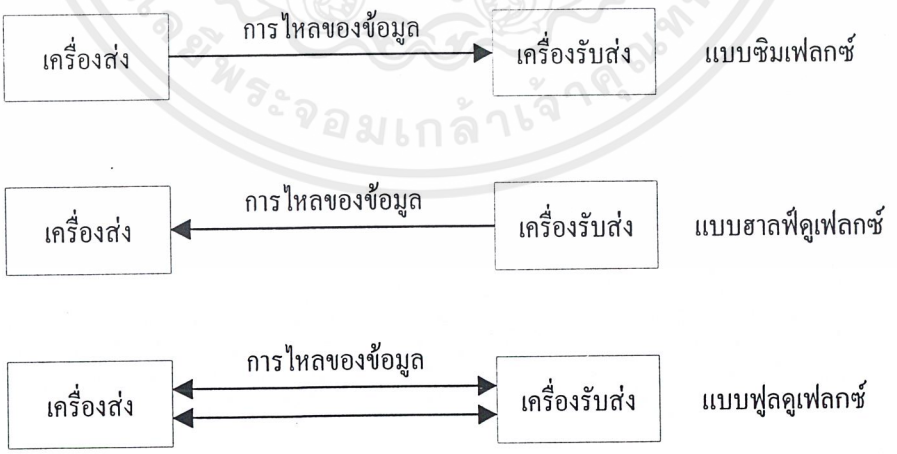
เป็นการสื่อสารข้อมูลที่ข้อมูลสามารถส่งข้อมูลได้ในทางเดียวเท่านั้น เมื่อทำการสื่อสารในทิศทางใดก็จะใช้ทิศทางนั้นตลอดเวลา ไม่มีการเปลี่ยนแปลงทิศทาง เช่น การส่งสัญญาณจากภาพจากสถานีโทรทัศน์ไปยังเครื่องรับโทรทัศน์ หรือการส่งข้อมูลจากศูนย์บริการไปยังวิทยุติดตามตัว

2) การสื่อสารข้อมูลแบบ 2 ทิศทางตลอดเวลาหรือแบบฮาล์ฟดูเพล็ก (Half duplex)

เป็นการสื่อสารข้อมูลที่สามารถส่งได้ 2 ทิศทาง โดยจะทำการส่งในลักษณะของการผลัดกันรับและส่ง โดยในขณะเวลาหนึ่งนั้นสัญญาณจะไปได้ในทิศทางเดียวเท่านั้น ดังนั้นอุปกรณ์แต่ละตัวที่จะเชื่อมต่อหรือสื่อสารข้อมูลในลักษณะนี้จะต้องเป็นไปได้ทั้งตัวรับและตัวส่งซึ่งมีชื่อ เรียกว่า **ทรานซีฟเวอร์ (Tranceiver)** และจะต้องมีวงจรที่จะเลือกว่า ณ เวลานั้นจะทำงานเป็นตัวรับหรือตัวส่ง

3) การสื่อสารข้อมูลแบบ 2 ทิศทางตลอดเวลาหรือแบบฟูลดูเพล็ก (Full duplex)

เป็นการสื่อสารข้อมูลที่คล้ายกับแบบฮาล์ฟดูเพล็ก แต่เป็นการสื่อสารข้อมูลใน 2 ทิศทางแบบตลอดเวลา



รูปที่ 2.3 รูปแบบของการติดต่อสื่อสารข้อมูลแบบอนุกรม

2.3.2 การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

การสื่อสารข้อมูลแบบอนุกรมที่มีการใช้งานอยู่ในปัจจุบันนี้นั้น ได้มีการกำหนดมาตรฐานในการรับส่งข้อมูลไว้หลายแบบด้วยกันแต่ที่ได้รับความนิยมนำมาใช้งานอย่างกว้างขวาง ก็คือ การสื่อสารข้อมูลแบบอนุกรมตามมาตรฐาน RS-232C

ทั้งนี้เนื่องจากความต้องการในการสื่อสารข้อมูล ผ่านทางเครือข่ายโทรศัพท์ มีมากขึ้นเรื่อย ๆ ดังนั้นจึงได้กำหนดมาตรฐานที่เรียกว่า RS-232C ขึ้นเพื่อใช้เป็นมาตรฐานแก่อุปกรณ์ที่ถูกผลิตขึ้นจากบริษัทต่าง ๆ ในสหรัฐอเมริกา Bell System operating telephone companies เป็นบริษัทหลักบริษัทแรกที่เป็นผู้ผลิตและติดตั้งระบบสื่อสารข้อมูล และเป็นผู้ผลิตอุปกรณ์ต่าง ๆ ที่ใช้ในการอินเตอร์เฟสอุปกรณ์ดิจิตอลกับเครื่องข่ายโทรศัพท์รายใหญ่อุปกรณ์นี้ก็คือ Bell modem ซึ่งถูกพัฒนาโดย Bell Laboratories และถูกใช้เป็นมาตรฐานในงานอุตสาหกรรมจนถึงปัจจุบันนี้ (บริษัทต่าง ๆ มักจะลอกข้อกำหนดต่าง ๆ ของ Bell ไปใช้งานเพื่อให้สินค้าของบริษัทนั้น ๆ สามารถใช้กับอุปกรณ์ของ Bell ได้) ดังนั้นความต้องการเกี่ยวกับข้อมูล ข้อกำหนด ในการอินเตอร์เฟสกับโมเด็มจึงมีเพิ่มขึ้นเรื่อย ๆ เพื่อตอบสนองต่อความต้องการนี้ EIA, Bell System และผู้ผลิตโมเด็มรายอื่น ๆ จึงได้ร่วมมือกันตั้งมาตรฐาน RS-232C ขึ้น

มาตรฐาน RS-232C ได้ถูกตีพิมพ์โดย EIA ในปี ค.ศ 1969 โดยตัวอักษร RS แทน "Recommended Standard" 232 แทนหมายเลขของมาตรฐาน ส่วนอักษร C แสดงให้เห็นว่ามาตรฐานได้รับการแก้ไขกี่ครั้ง

การที่มาตรฐานนี้เป็นที่นิยมใช้ ก็เนื่องจาเป็นระบบการสื่อสารข้อมูลที่ใช้ใน เครื่องไมโคร คอมพิวเตอร์ IBM PC ซึ่งเป็นคอมพิวเตอร์ที่มีใช้อย่างแพร่หลายตั้งแต่อดีตมาจนถึงปัจจุบัน

มาตรฐาน RS-232C จะมีโครงสร้างการสื่อสารเป็นแบบจุดต่อจุดเท่านั้น โดยมีลักษณะสมบัติทางไฟฟ้า และทางกายภาพ

เพื่อที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกันทำงานร่วมกันได้ มาตรฐานหลายชนิดจึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันกว้างขวางที่สุดคือ RS-232-C ถูกประกาศในปี 1969 โดย Electronic Industries Association มาตรฐาน RS-232C ที่ร่างขึ้นในตอนเริ่มแรกสำหรับกำหนดการเชื่อมต่อระหว่างเทอร์มินัล (terminal) และโมเด็ม ระบุคุณสมบัติทางไฟฟ้าของวงจรระหว่างอุปกรณสองตัว และกำหนดชื่อและหมายเลขแก่สายที่จำเป็นสำหรับการเชื่อมต่อวงจร ชื่อวงจรตามมาตรฐาน RS-323-C (AA,AB เป็นต้น) จำได้ยากในทางปฏิบัติจึงใช่ชื่อย่อแทน

ตัวอย่างเช่น สายเส้นที่ 2 มีชื่อ BA แต่ใช่กันทั่วไปว่า TXD (Transmitted Data) ตามมาตรฐาน RS-232C สายเส้นที่ 2 นำข้อมูลจากเทอร์มินัลไปสู่โมเด็ม เพื่อให้การทำงานถูกต้องเทอร์มินัลต้องส่ง

เอาต์พุต ออกที่สายเส้นที่ 2 และโมเด็มต้องรัยสาบข้อมูลบนสายเส้นที่ 2 เพราะฉะนั้นสายเส้นที่ 2 เป็นสายส่งข้อมูลสำหรับอุปกรณ์บางอย่างและเป็นสายรับข้อมูลสำหรับอุปกรณ์อย่างอื่น การเชื่อมต่อโดยตรงจากสายเส้นที่ 2 บนอุปกรณ์หนึ่งเข้ากับสายเส้นที่ 2 บนอุปกรณ์อีกตัวหนึ่ง สามารถทำได้ต่อเทื่ออุปกรณ์หนึ่งส่งข้อมูลบนสายเส้นที่ 2 และอีกตัวหนึ่งรับข้อมูลบนสายเส้นที่ 2 เพื่อป้องกันไม่ให้อุปกรณ์ส่งข้อมูลบนสายเส้นเดียวกัน อุปกรณ์จึงถูกแบ่งออกเป็นสองชนิด อุปกรณ์อย่างเทอร์มินัล ซึ่งใช้สายเส้นที่ 2 สำหรับเอาท์พุต เรียกว่า DTE(Data Terminal Equipment) อุปกรณ์อย่างเช่นโมเด็มซึ่งใช้สายเส้นที่ 2 สำหรับอินพุต เรียกว่า DCE (Data Communication Equipment)

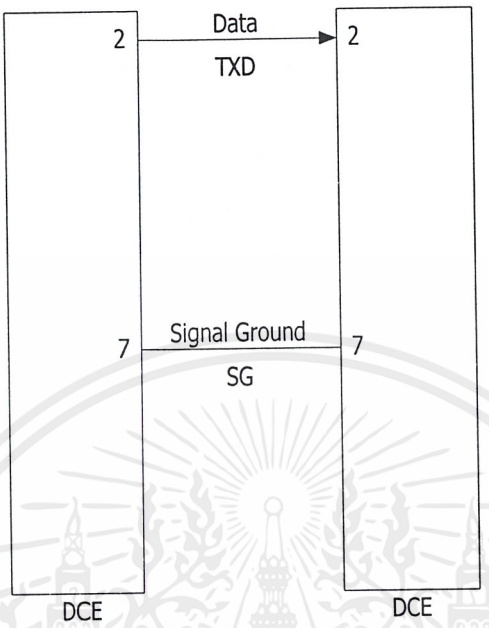
2.3.2 อุปกรณ์ DTE และ DCE

ตามมาตรฐาน RS-232C อุปกรณ์ DTE ควรใช้หัวต่อตัวผู้ และอุปกรณ์ DCE ควรใช้หัวต่อตัวเมีย อย่างไรก็ตามผู้ผลิตไม่ได้ปฏิบัติตามกฎนี้เสมอ ดังนั้นจึงไม่อาจแยกแยะอุปกรณ์ DTE และ DCE โดยการมองผ่านๆได้เสมอไป

เมื่อทราบว่าอุปกรณ์หนึ่งเป็น DTE และอีกตัวหนึ่งเป็น DCE ในทางทฤษฎีแล้วสามารถเชื่อมต่อได้อย่างง่ายดาย โดยการเชื่อมต่อที่มีหมายเลขตรงกัน เช่น เส้นที่ 2 กับ 2, 3 กับ 3 เป็นต้น เรียกว่าการเชื่อมต่อแบบตรงไปตรงมา แต่มีผู้ผลิตบางรายที่ทำไม่ได้มาตรฐานและทำให้เกิดปัญหาหลายอย่าง ปัญหาเหล่านี้จะได้รับการกล่าวถึงในภายหลัง เช่นเดียวกับวิธีจัดหาสถานการณ์ที่อุปกรณ์ทั้งสองเป็น DTE หรือ DCE เหมือนกัน ในตอนนี้ให้ถือว่าอุปกรณ์หนึ่งเป็น DTE และอีกตัวหนึ่งเป็น DCE และแต่ละฝ่ายส่งสัญญาณที่อีกฝ่ายต้องการบนสายที่ตรงกัน

2.3.3 การสื่อสารทางเดียว

วงจรหลักที่ถูกใช่สำหรับการสื่อสารมีอยู่สามวงจร คือสายเส้นที่ 2 สำหรับข้อมูลจาก DTE ไปยังDCEสายเส้นที่3 สำหรับข้อมูลจาก DCE ไปยัง DTE และสายเส้นที่ 7 สำหรับซิกแนลกราวนด์ (Signal ground) ซึ่งเป็นจุดอ้างอิงร่วมสำหรับขั้วและแรงดันไฟฟ้าของสายอื่น ในกรณีที่ง่ายที่สุดซึ่งมีเพียงอุปกรณ์หนึ่งส่งและอีกตัวรับใช่สายเพียงสองเส้นก็เพียงพอคือสายเส้นที่ 2 หรือ 3 และเส้นที่ 7 ดังในรูป 2.4



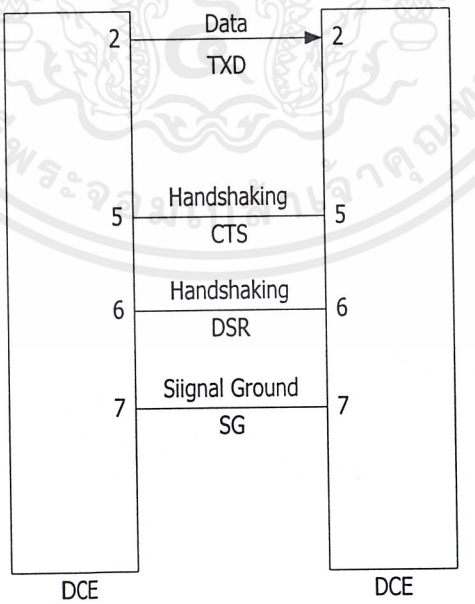
รูปที่2.4 การเชื่อมต่อทางเดียวอย่างง่าย

2.3.4 ฮาร์ดแวร์แฮนด์เช็คกิ้ง

ในหลายๆ กรณี อุปกรณ์ฝ่ายส่งจำเป็นต้องรู้ว่าอุปกรณฝ่ายรับพร้อมที่จะรับข้อมูลหรือไม่ ตัวอย่างเช่น การส่งข้อมูลไปที่เครื่องพิมพ์ ความเร็วของการสื่อสารอาจเร็วกว่าความเร็วของเครื่องพิมพ์ เครื่องพิมพ์ต้องระงับการส่งข้อมูลของคอมพิวเตอร์จนกว่ามันพร้อมที่จะรับข้อมูล ในทำนองเดียวกันกับการส่งข้อมูลจากคอมพิวเตอร์เครื่องหนึ่งไปยังอีกเครื่องหนึ่ง และคอมพิวเตอร์ตัวที่สองไม่สามารถประมวลข้อมูลได้เร็ว เท่ากับอัตราที่ข้อมูลเข้ามาทั้งสองกรณี ต้องมีข่าวสารส่งจากอุปกรณ์ฝ่ายรับไปยังอุปกรณ์ฝ่ายส่ง เพื่อแจ้งว่ามันพร้อมหรือไม่ข่าวสารนี้เรียกว่า โฟลร์คอนโทรล (flow control) หรือแฮนด์เช็คกิ้ง (hand chaking) แฮนด์เช็คกิ้งมีสองแบบคือ ฮาร์ดแวร์และซอฟต์แวร์ ทั่งคู่เกี่ยวข้องกับสัญญาณที่ส่งกลับมาจากอุปกรณ์ฝ่ายรับไปยังอุปกรณ์ฝ่ายส่ง ในฮาร์ดแวร์แฮนด์เช็คกิ้งอุปกรณ์ฝ่ายรับส่งแรงดันไฟฟ้าบวกไปตามวงจรแฮนด์เช็คกิ้ง เมื่อมันพร้อมที่จะรับข้อมูลเมื่อคอมพิวเตอร์ฝ่ายส่งได้รับแรงดันไฟฟ้าลบ มันจะรู้ว่าต้องหยุดส่งข้อมูล ในซอฟแวร์แฮนด์เช็คกิ้งสัญญาณแฮนด์เช็คกิ้งเป็นอักขระพิเศษ ที่ถูกส่งไปตามวงจรข้อมูล แทนวงจรแฮนด์เช็คกิ้งการใช้ฮาร์ดแวร์แฮนด์เช็คกิ้งอย่างน้อยต้องมีการเชื่อมต่อเพิ่มเติมหนึ่งเส้น เพื่อนำพาสัญญาณที่ให้จำนวนของสายทั้งหมดเป็นสามเส้นคือ สายข้อมูล ซิกแนลกราวนด์ และแฮนด์เช็คกิ้ง

2.3.5 จาก DTE ไปสู่ DCE

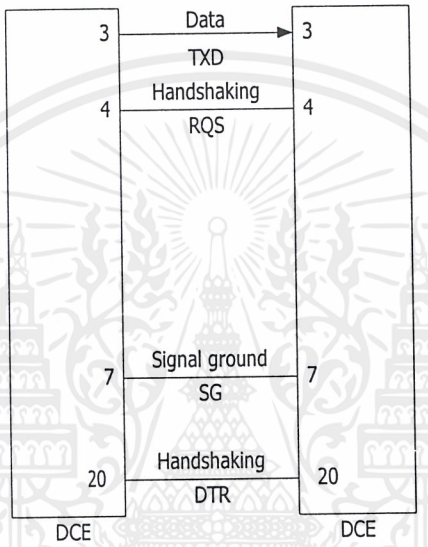
เมื่ออุปกรณ์ DTE ส่งขอมูลไปยังอุปกรณ์ DCE ข้อมูลถูกส่งไปตามสายเส้นที่ 2 และใช้สายเส้นที่ 7 ซิกแนลกราวนด์ ตามปกติอุปกรณ์ DCE ควบคุมการส่งแฮนด์เช็คกิ้งจากอุปกรณ์ DTE บนสายเส้นที่ 6 ชื่อว่า DSR (Data Set Ready) ถ้าเครื่องพิมพ์เป็น DCE และคอมพิวเตอร์เป็น DTE สายเส้นที่ 6 บนคอมพิวเตอร์ถูกเชื่อมถูกเชื่อมต่อเข้ากับสายเส้นที่ 6 บนเครื่องพิมพ์ และเครื่องพิมพ์จะรักษาแรงดันไฟฟ้าบวกบนสายเส้นที่ 6 ตราบเท่าที่มันสามารถรับข้อมูล เมื่อมันต้องการบอกคอมพิวเตอร์ให้หยุดการส่งข้อมูล มันจะลดแรงดันไฟฟ้าบนสายไฟฟ้าเส้นที่ 6 ให้เป็นสถานะลบบ่อยครั้งที่วงจรแฮนด์เช็คกิ้งชุดที่สอง คือสายเส้นที่ 5 ถูกใช้โดยอุปกรณ์ DCE เพื่อควบคุมการส่งอุปกรณ์ DTE วงจรนี้มีชื่อว่า CTS (Clear To Send) เมื่อสายแฮนด์เช็คกิ้งทั้งสองเส้นถูกใช้ อุปกรณ์ DTE ต้องได้รับการออกแบบให้ส่งข้อมูลก็ต่อเมื่อสายทั้งสองเป็นไฮ (high) หรือแรงดันไฟฟ้าบวก บางครั้งสายนั้นอาจมีความหมายต่างกัน เช่นเส้นหนึ่งอาจบอกอุปกรณ์ฝ่ายส่งให้หยุดการพิมพ์จนกระทั่งข้อมูลถูกพิมพ์ไปได้จำนวนหนึ่ง และเส้นที่เหลืออาจจะแจ้งว่ากระดาษของเครื่องพิมพ์หมด อย่างไรก็ตามความหมายเหล่านี้ไม่ได้มีมาตรฐาน เนื่องจากคอมพิวเตอร์หลายชนิดถูกโปรแกรมไม่ให้ส่งข้อมูล ถ้าสายแฮนด์เช็กกิ้งทั้งสองไม่เป็นไฮ แม้แต่เครื่องพิมพ์ที่ไม่ได้กำหนดความหมายพิเศษกับสายชุดที่สองอย่างน้อยก็ควรรักษาแรงดันไฟฟ้าบวกไว้ อย่างไรก็ตามบ้างครั้งสัญญาณชุดที่สองต้องถูกสร้างหลอกขึ้นมาโดยการต่อมันเข้ากับชุดแรก



รูปที่ 2.5 การสื่อสารทางเดียวพร้อมด้วยแฮนด์เช้กกิ้งจาก DTE ไปยัง DCE

2.3.6 DCE ไปยัง DTE

เมื่ออุปกรณ์ DCE จะสื่อสารกับอุปกรณ์ DTE สายเส้นที่ 3 ต้องถูกใช้สำหรับส่งผ่านข้อมูล และถ้าต้องการแฮนด์เช็คกิ้ง จะใช้สายเส้นที่ 20 เพื่อส่ง แฮนด์เช็คกิ้ง จากอุปกรณ์ DTE ไปยัง DCE สายเส้นที่ 20 มีชื่อว่า DRT (Data terminal Ready) สายแฮนด์เช็คกิ้ง ชุดที่สองคือสายเส้นที่ 4 เรียกว่า Request To Send (RQS หรือ RT1S4ไม่ได้ถูกนำมาเสมอไป)

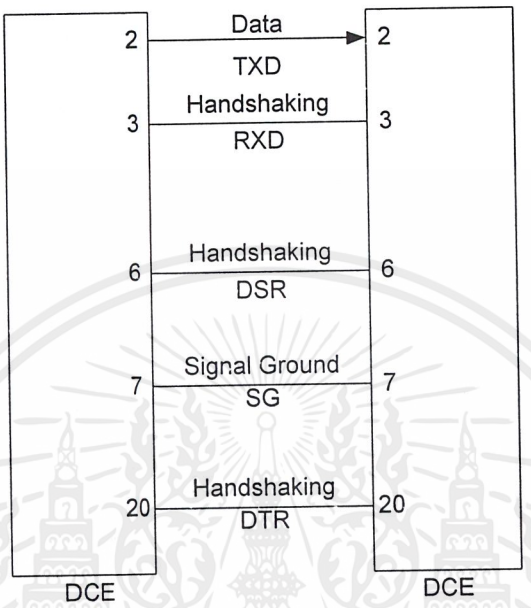


รูปที่ 2.6 การสื่อสารทางเดียวพร้อมด้วยแฮนด์เช็คกิ้งจากDTE ไปยัง DCE

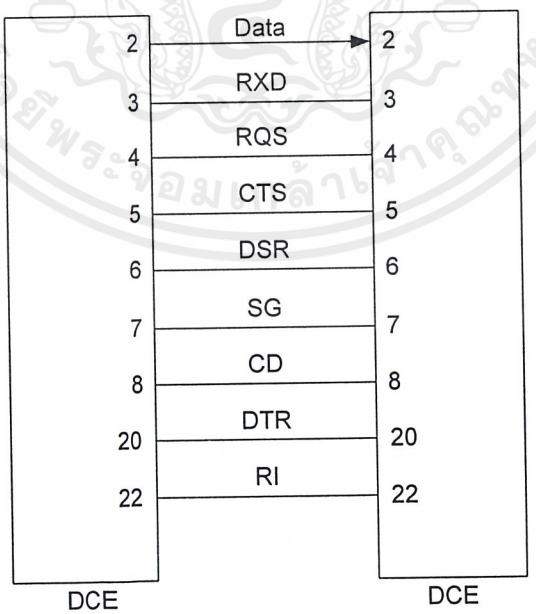
2.3.7 การสื่อสารสองทาง

ในหลายกรณีที่ข้อมูลถูกส่งผ่านในสองทิศทาง โดยเฉพาะเมื่อคอมพิวเตอร์สองตัวสื่อสารกัน รวมทั้งกรณีที่ใช้ซอฟต์แวร์แฮนด์เช็คกิ้งด้วยเช่นกันจำนวนสายที่น้อยที่สุดที่จำเป็นในการสื่อสารสองทางคือ สามเส้น ได้แก่ สายข้อมูลในแต่ละทิศทาง และซิกแนลกราวนด์ การเพิ่มเติมสายแฮนด์เช็คกิ้ง ในแต่ละทิศทาง ทำให้จำนวนสายรวมเป็นห้าเส้น ดังแสดงในรูป 2.7 เมื่อสายแฮนด์เช็คกิ้ง ชุดที่สองถูกนำมาใช้เพิ่มเติมลงในแต่ละทิศทางสายทั้งหมดที่ใช้คือเจ็ดเส้น บางครั้งอาจมีการเพิ่มสายอีกสองเส้น เพื่อทำให้โมเด็มสามารถให้ข้อมูลมากขึ้นแก่คอมพิวเตอร์หรือเทอร์มินัล ได้แก่ CD (Carrier Detect) ถูกเชื่อมต่อเข้ากับขา 8 เพื่อแจ้งการคงอยู่ของสัญญาณพาหะ และ RI (Ring Indicator) ถูกเชื่อมต่อกับขา 22 เพื่อแสดงว่าโมเด็มกำลังถูกเรียกโดยอุปกรณ์ระยะไกลซึ่งก็คือการตรวจสอบสัญญาณกริ่งของโทรศัพท์นั้นเอง จำนวนวงจรทั้งหมดจะกลายเป็นเก้าตามรูปที่ 2.8 แม้ว่าจะมีวงจรอื่นอีกหลายวงจรถูกกำหนดโดย RS-232C แต่ทั้งเก้าวงจรนี้ถูกใช้กันมากที่สุด

และเป็นเพียงชุดเดียวที่โดยปกติถูกเชื่อมต่อกับไมโครคอมพิวเตอร์ซึ่งเป็นสาเหตุที่ทำให้ไมโครคอมพิวเตอร์ ใช้หัวต่อ 9 ขา แทน 25 ขา สำหรับนำพาสัญญาณที่จำเป็นทั้งหมดของวงจร RS-232C



รูปที่ 2.7 การสื่อสารสองทางพร้อมด้วยวงจรแฮนด์เช็คกิ้งหลัก



รูปที่ 2.8 การเชื่อมต่อ RS-232 แบบมาตรฐานเก้าเส้น

2.3.7 นัลโมเด็ม

ดังที่ได้กล่าวมาก่อนหน้านี้แล้วในตอนแรก RS-232C นำไปใช้สำหรับกำหนดการเชื่อมต่อระหว่างเทอร์มินัลซึ่งเป็น DTE กับโมเด็มซึ่งเป็น DCE ต่อมาได้มีการนำไปประยุกต์ใช้กับการเชื่อมต่อระหว่างอุปกรณ์อื่นอีกหลายชนิดที่ไม่ได้ถูกกำหนดให้เป็น DTE หรือ DCE อย่างเป็นกิจลักษณะ เช่น ไมโครคอมพิวเตอร์หรือเครื่องพิมพ์เนื่องจากไม่มีมาตรฐานที่ชี้ชัดว่าอุปกรณ์ควรเป็น DTE หรือ DCE บ่อยครั้งจำเป็นต้องเชื่อมต่ออุปกรณ์ DTE สองตัว หรืออุปกรณ์ DCE สองตัวเข้าด้วยกัน ในลักษณะเช่นนี้ต้องการเชื่อมต่อสาย 2 บนอุปกรณ์ตัวแรกกับสาย 3 บนอุปกรณ์ตัวที่สอง และสาย 3 บนอุปกรณ์ตัวแรกกับสาย 2 บนตัวที่สอง สายแฮนด์เช็คกิ้งก็ต้องถูกไขว้ในทำนองเดียวกันการไขว้สายอาจทำได้โดยการต่ออุปกรณ์ด้วยสายสัญญาณซึ่งไขว้ไว้แล้วหรือโดยการซื้อหัวต่อพิเศษที่เชื่อมต่ออุปกรณ์ทั้งสองและทำการไขว้สายที่จำเป็นไว้ภายใน ไม่ว่ากรณีใด สายที่อยู่ตรงกลางหรือหัวต่อจะถูกเรียกว่า นัลโมเด็ม (Null modem) มันทำให้อุปกรณ์ DTE สองตัวคุยกันได้โดยไม่ต้องมีอุปกรณ์ DCE เป็นทางผ่าน หรือในทางกลับกันคือระหว่าง DCE กับ DCE การต่อสายสำหรับนัลโมเด็มแสดงไว้ในแผนภาพที่ ตีพิมพ์ไว้ในภาคผนวกของหนังสือเล่มนี้

2.3.8 สัญญาณทางไฟฟ้า

มาตรฐาน RS-232C กำหนดคุณลักษณะของสัญญาณทางไฟฟ้าที่ถูกใช้ในการเชื่อมต่ออนุกรมโดยตรง มีเพียงสองลักษณะคือ SPACE แสดงถึงไบนารี 0 หรือแรงดันไฟบวกและ Mark แสดงถึงไบนารี 1 หรือแรงดันไฟฟ้าลบบนข่ายสายข้อมูล (เช่นสาย 2 และ 3) แรงดันไฟฟ้าบวกแสดงถึงค่าลอจิก(logic) 0 และแรงดันไฟฟ้าลบแสดงถึงค่าลอจิก 1 บนสายแฮนด์เช็คกิ้ง (เช่น DTR และ DSR) แรงดันไฟฟ้าบวกแสดงถึงข้อมูลได้ ส่วนแรงดันไฟฟ้าลบหมายถึงการหยุดส่งข้อมูลแรงดันไฟฟ้าบวก (สถานะ Space) อยู่ระหว่าง +5 ถึง +15 โวลต์ สำหรับเอาท์พุต และระหว่าง +3 และ+15 โวลต์ สำหรับอินพุต ความแตกต่างที่มีไว้เผื่อกรณีที่แรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณ ในทำนองเดียวกัน แรงดันไฟฟ้าลบ(สถานะ Mark) ถูกกำหนดระหว่าง -5 ถึง -15 โวลต์สำหรับเอาต์พุต และ -3 ถึง -15โวลต์ สำหรับอินพุตสังเกตว่า ถ้าให้สัญญาณสายยาวเกินไประดับแรงดันสัญญาณไฟฟ้าจะตกลงเกินขอบเขตที่ยอมได้ นอกจากนี้ ความจุไฟฟ้าที่เกิดขึ้นจะมีผลกับคุณภาพของสัญญาณ โโยทำให้เกิดการเปลี่ยนสถานะจากแรงดันไฟฟ้าบวกไปลบไม่ชัดเจน RS-232C ไม่ได้มุ่งหวังให้นำไปใช้กับระยะทางไกล และโดยทั่วไป 50 ฟุตเป็นระยะทางไกลที่สุดในการใช้สายสัญญาณปกติที่อัตราการส่งข้อมูลปกติ ถ้าอุปกรณ์อยู่ห่างกันมาก อาจจำเป็นต้องใช้โมเด็มหรืออุปกรณ์อื่น

2.3.9 RS-449,442-A และ 423-A

มาตรฐานที่ใหม่กว่า RS-232C มาตรฐานหนึ่งคือ RS-449 ซึ่งใช้จัดการกับปัญหาเดียวกันแต่ยอมรับความเร็วในการส่งข้อมูลสูงกว่าและลดการเกิดครอสทอล์ค (crosstalk) RS-449 กำหนดให้ใช้หัวต่อแบบ 37 ขา และกรณีที่ไม่เพียงพออาจเพิ่มหัวต่อ 9 ขา อีกตัวหนึ่ง RS-449 ครอบคลุมข้อกำหนดทางกลและคำอธิบายวงจร แต่ไม่รวมคุณลักษณะทางไฟฟ้า มันถูกมุ่งหวังให้ใช้ในการเชื่อมกับ RS-422-A และ RS-423-A ซึ่งอธิบายคุณลักษณะทางไฟฟ้าของวงจรแบบสมดุล (Balanced circuit) และแบบไม่สมดุล (Un balanced circuit) ตามลำดับ วงจรแบบสมดุลถูกใช้ในการส่งข้อมูลความเร็วสูงกว่า หรือซึ่งที่มีปัญหาในการส่งข้อมูลในสายสองสาย สภาวะ Mark และ Space ถูกส่งโดยการเปลี่ยนขั้วของสัญญาณในสายทั้งสอง ซึ่งอ้างอิงในกันและกันแทนการใช้สายเส้นเดียวที่เปลี่ยนขั้วโดยอ้างอิงจากซิกแนลกราวนด์ที่ใช้ร่วมกัน

Electronic Industries Association (EIA) กำลังพยายามโน้มน้าวให้ใช้คำว่า EIA แทนRS ในมาตรฐานต่างๆแต่ในหนังสือเล่มนี้จะใช่ชื่อ RS ซึ่งเป็นที่รู้จักกันมากกว่า

2.3.10 การเชื่อมต่อ RS-449 และ RS-232C

มันเป็นไปได้ที่จะเชื่อมต่อระหว่างอุปกรณ์ RS-449 และ RS-232C EIA เอกสารนี้ประยุกต์กับวงจรRS-423 เท่านั้น

2.3.11 การเชื่อมต่อ RS-232-C และแอปเปิลแมคอินทอช

ในทางทฤษฎีเป็นไปไม่ได้ที่จะเชื่อมต่อแอปเปิลแมคอินทอช (Apple Macintosh) ด้วยวงจร RS-422 ซึ่ง แมคอินทอชใช้อยู่ไม่มีซิกแนลกราวนด์ร่วมสำหรับให้สายข้อมูลที่ถูกส่งและรับใช้เป็นจุดอ้างอิง ถึงกระนั้นก็ตามแมคอินทอชได้รับการออกแบบ ให้ได้รับการเชื่อมต่อเป็นไปได้ เครื่องพิมพ์ ImageWriter ซึ่งใช้กับแมคอินทอชเป็นอุปกรณ์ DTE RS-232C และสายสัญญาณที่ใช้ในการเชื่อมต่อแมคอินทอชกับImageWrite มีหัวต่อ 25 ขา ซึ่งสามารถใช้สำหรับเชื่อมต่ออุปกรณ์อนุกรมอื่นได้ (มีข้อสังเกตุว่าหัวต่อ 9 ขา ของแมคอินทอชไม่มีความสัมพันธ์ กับข้อกำหนดการเชื่อมต่อขาใน RS-449

ต่อไปนี้เป็นลูกเล่นบางอย่างที่ช่วยในการแก้ปัญหาที่อาจพบขณะเชื่อมต่ออุปกรณ์อนุกรม

2.3.12 ความจำเป็นของนัลโมเด็ม

ดังที่กล่าวมาแล้วว่า จำเป็นต้องใช้นัลโมเด็มเมื่อจะต้องเชื่อมต่ออุปกรณ์สองตัวที่เป็น DTE หรือ DCE ทั้งคู่ แต่ในบางกรณีที่อาจไม่ทราบว่าอุปกรณ์ที่กำลังจะเชื่อมต่อเป็น DTE หรือ DCE (อุปกรณ์รุ่นใหม่ เช่น เมาส์ หรือปากกาแสงมักก่อให้เกิดปัญหานี้) เนื่องจากมีความเป็นไปได้เสมอที่อุปกรณทั้งสองอาจจะเหมือนกัน จึงควรลองใช้นัลโมเด็มถ้าไม่สามารถส่งข้อมูลด้วยการต่อแบบตรงไปตรงมา

2.3.13 ปัญหาของแฮนด์เช็คกิ้ง

ถ้าเครื่องพิมพ์ไม่ตอบสนองต่ออุปกรณ์ฝ่ายส่ง มันอาจเป็นเครื่องพิมพ์ที่ต้องการสัญญาณบนสายแฮนด์เช็คกิ้งสองชุดเป็นไฮ ในขณะที่เครื่องคอมพิวเตอร์ให้สัญญาณเพียงเส้นเดียว กรณีนี้เกิดขึ้นบ่อยกับไอบีเอ็มพีซี ซึ่งมีความสามารถในการใช้สัญญาณทั้งสองเส้น แต่ต้องมีการโปรแกรมเป็นพิเศษบ่อยครั้งที่สัญญาณชุดที่สองถูกสร้างหลอก โดยการเชื่อมต่อสายแฮนด์เช็คกิ้งชุดที่สองเข้ากับชุดหลักทางปลายด้านเครื่องพิมพ์

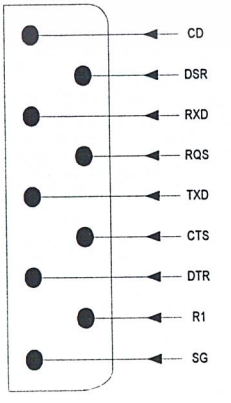
ถ้าไมโครคอมพิวเตอร์ไม่ได้จัดเตรียมสัญญาณแฮนด์เช็คกิ้งเลย และเครื่องพิมพ์ยืนยันที่รับหนึ่งหรือสองสัญญาณ ก็สามารถสร้างหลอกขึ้นได้เช่นกัน โดยป้อนกลับสัญญาณแฮนด์เช็คกิ้งที่เครื่องพิมพ์ส่งออกมากลับเข้าเครื่องพิมพ์เอง เสมือนเป็นสัญญาณที่ถูกส่งเข้ามา

ถ้าคอมพิวเตอร์ไม่ส่งเมื่อควรจะเป็นอาจเกิดมันรอสัญญาณแฮนด์เช็คกิ้งที่ยังไม่ได้รับ ถ้าเครื่องพิมพ์ทำให้สายแฮนด์เช็คกิ้งเป็นไฮเพียงเส้นเดียวลองเชื่อมต่อกัยสายแฮนด์เช็คกิ้งอื่นทางด้านคอมพิวเตอร์เข้ากับมัน

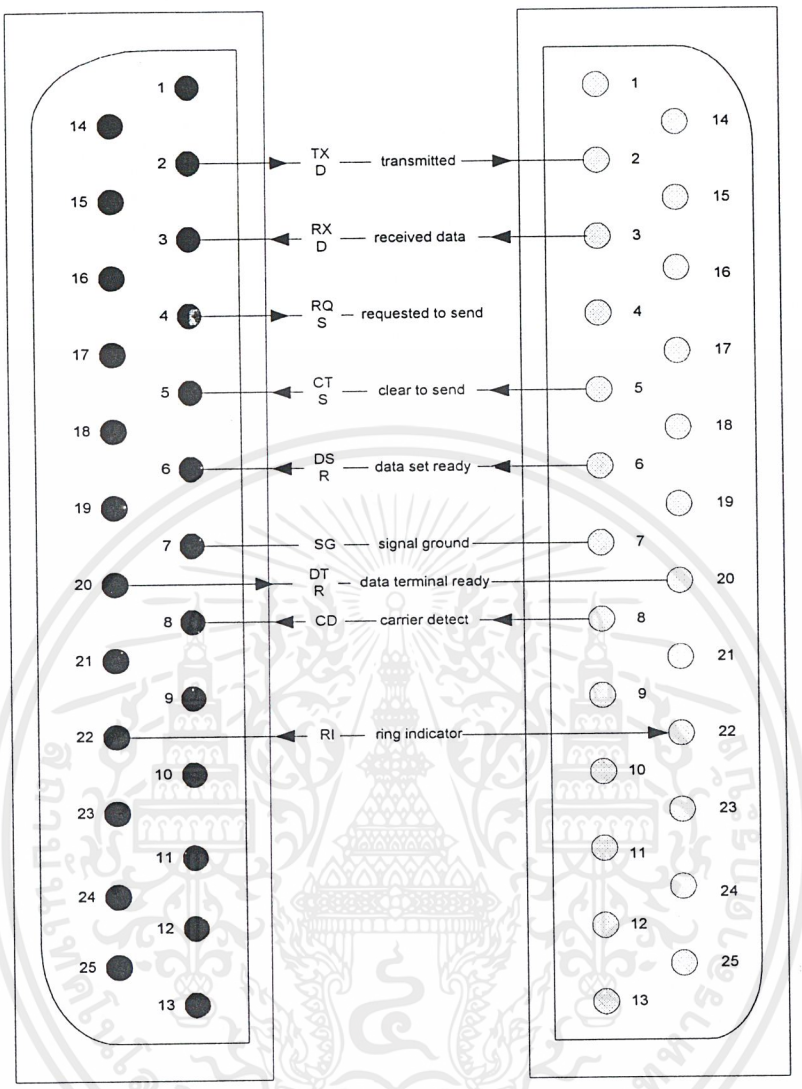
2.3.14 การเบรกเอาต์บ็อกซ์

ถ้าทำการเชื่อมต่อหลายรูปแบบ ขอแนะนำให้ใช้เบรกเอาต์บ็อกซ์ (breakout box) อุปกรณ์เล็กๆชิ้นนี้มีหัวต่อ D-type สองตัวซึ่งสามารถแทรกเข้าไประหว่างอุปกรณ์อนุกรมสองตัว แต่ละสายมีไฟหนึ่งดวง ซึ่งจะติดเมื่อมีสัญญาณบนวงจรนั้น เบรกเอาต์บ็อกซ์ทำให้มองเห็นเมื่อข้อมูลกำลังถูกส่ง สายไหนที่มันกำลังเดินทาง และสายแฮนด์เช็คกิ้งเส้นใดกำลังนำพาแรงดันไฟฟ้าบวก การสร้างสายเข้าและออกและการเชื่อมสายทำได้โดยการใส่สายต่อลงในซ็อกเกต

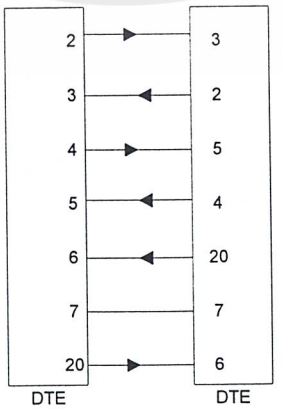
ด้วยการใช่เบรกเกอร์เอาต์บ็อกซ์ คุณสามารถทดลองการเปลี่ยนการเชื่อมต่อโดยไม่ต้องมีการบัดกรีสายเข้าออก หลังจากได้การเชื่อมต่อที่ถูกต้องแล้วจึงค่อยนำไปสร้างเป็นสัญญาณที่เหมาะสม



รูปที่ 2.9 หัวต่อแบบ DB-9 ในไอบีเอ็มพีซีเอที



รูปที่ 2.10 การเชื่อมต่อ DTE (ไอบีเอ็มพีซี) ไปยัง DCE (โมเด็ม) แบบมาตรฐาน



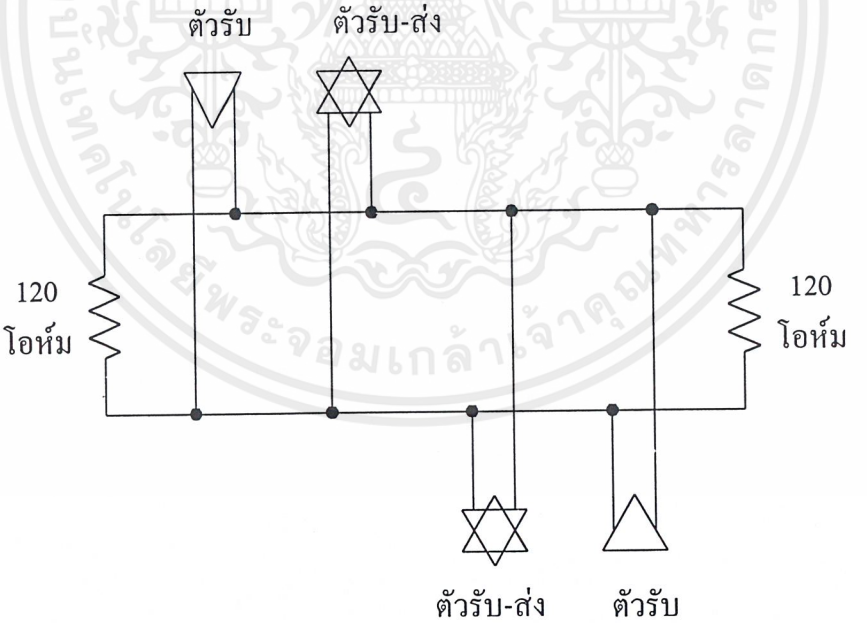
รูปที่ 2.11 การเชื่อมต่อแบบนัลโมเด็ล

2.3.15 การสื่อสารข้อมูลแบบอนุกรมด้วย RS - 485

การสื่อสารข้อมูลตามมาตรฐานที่ได้กล่าวมาแล้วในข้างต้น คือ RS-232C นั้นเป็นมาตรฐานการสื่อสารข้อมูลในแบบที่ใช้การสื่อสารข้อมูลระหว่างอุปกรณ์ หรือแบบจุดต่อจุด (Point to Point)

ด้วยเหตุนี้ จึงได้มีการพัฒนามาตรฐาน ของการสื่อสารข้อมูลแบบใหม่ขึ้น เพื่อรองรับความต้อง การดังกล่าวนี้ซึ่งก็คือมาตรฐาน RS - 485

สำหรับมาตรฐาน RS - 485 นั้นจะเป็นมาตรฐานที่อาศัย หลักการทำงานของการส่งสัญญาณแบบดิฟเฟอเรนเชียลเช่นเดียวกับมาตรฐาน RS - 422A แต่สามารถสื่อสารข้อมูลได้ 2 ทิศทาง ในสายนำสัญญาณเพียงชุดเดียว ซึ่งเป็นการสื่อสารข้อมูลแบบฮาล์ฟดูเพล็กซ์ จากผลการทำงานดิฟเฟอเรนเชียลนี้ทำให้ระยะทาง และ ความสูงขึ้นเช่นเดียวกับมาตรฐาน RS - 422A แต่มาตรฐาน RS - 485 นั้นสามารถที่จะสื่อสารข้อมูลตามมาตรฐาน RS - 485 นั้นเป็นการสื่อสารแบบหลายจุด (Multipoint Communication) โดยโครงสร้างการสื่อสารข้อมูลแบบ RS - 485 ดังแสดงในรูปที่ 2.12



รูปที่ 2.12 โครงสร้างของการสื่อสารข้อมูลอนุกรมแบบ RS-485

ตารางที่ 2.1 การเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA

พารามิเตอร์	RS - 232A	RS -423A	RS - 422A	RS - 485
โหมดการทำงาน	Single-ended	Single-ended	Differrential	Differrential
จำนวนของตัวรับและตัวส่งที่ยอมรับ	ตัวส่ง 1 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	1 ตัวส่ง 10 ตัวรับ	32 ตัวส่ง 32 ตัวรับ
ความยาวของคู่สายสัญญาณรับส่งข้อมูล	50 ฟุต	4000 ฟุต	4000 ฟุต	4000 ฟุต
อัตราการส่งข้อมูล	20 k	100 k	10 M	10 M
สูงสุด (bit / sec)				
แรงดันไฟฟ้าโหมดร่วมสูงสุด	± 2.5 V	± 6 V	+ 6 V - 2.5 V	+ 12 V - 7 V
Driver output	ต่ำสุด ± 5 V สูงสุด ± 15 V	ต่ำสุด ± 3.6 V สูงสุด ± 6 V	ต่ำสุด ±2 V	ต่ำสุด ±1.5 V
Driver Load (w)	3k ถึง 7k	ต่ำสุด 450	ต่ำสุด 100	ต่ำสุด 60
Driver slew rate	30 V/μS สูงสุด		NA	NA
กระแสสูงสุดเมื่อเอาต์พุตลัดวงจร	500 mA ลัดวงจรกับ Vcc หรือ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND	150 mA ลัดวงจรกับ GND 250 mA ลัดวงจรกับ 8V หรือ 12 V
ค่าความต้านทานเอาต์พุตของตัวส่ง (w)	NA- power ON 300 - power off	NA - power ON 60 k - power off	NA - power ON 60 k - power off	120 k power on, off
ค่าความต้านทานอินพุตของตัวรับ (w)	3 k ถึง 7 k	4 k	4 k	12 k
ความไวตัวรับ	±3 V	±200 mV	±200 mV	±200 mV

ตารางที่ 2.2 ตารางเปรียบเทียบมาตรฐานการสื่อสารข้อมูลของ EIA

รูปแบบการสื่อสารข้อมูล	แบบขนาน	แบบอนุกรม
1. ระยะทาง	ปกติจะน้อยกว่า 100 ฟุต	ส่งได้ตั้งแต่ระยะทางสั้น ๆ ไปจนถึงระยะทางเป็นไมล์
2. ความเร็ว	อัตราความเร็วสูงมาก ในระยะที่ไม่ไกลนัก กำหนดได้เป็นบิต/วินาที	อัตราความเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วง 0 ถึง 2 ล้านบิต/วินาที
3. ระดับของสัญญาณ	ในการอินเตอร์เฟสจะใช้ระดับของสัญญาณที่ใช้กับอุปกรณ์ (TTL) คือสัญญาณลอจิก 1 และ 0 จะแทนด้วยระดับแรงดัน 5V และ 0V ตามลำดับ	ใช้มาตรฐานของ EIA RS-232C คือมีระดับสัญญาณไฟฟ้าขนาด 12 V หรืออาจจะใช้มาตรฐาน 20 mA current loop หรืออาจจะใช้ระดับสัญญาณ (TTL) ก็ได้ (ใช้กันน้อยมาก)
4. ความผิดพลาดของสัญญาณ	ถ้าส่งในระยะทางที่ไกล ความผิดพลาดของข้อมูลจะเกิดขึ้นง่าย	การผิดพลาดของสัญญาณจะมีน้อยลง
5. ค่าใช้จ่าย	ถ้าส่งในระยะทางที่ไกล ๆ จะสิ้นเปลืองค่าใช้จ่ายมาก เพราะต้องใช้สายส่งสัญญาณหลายเส้น	สิ้นเปลืองน้อยกว่าหลายเท่า ถึงแม้ว่าจะใช้อุปกรณ์เปลี่ยนสัญญาณข้อมูลจากแบบขนานไปเป็นแบบอนุกรมแล้วส่งผ่านสายส่งใช้อุปกรณ์ ในการแปลงสัญญาณกลับมาเป็นขนานอีกก็ยังลงทุนน้อยกว่า

2.3.16 มาตรฐาน RS-422/RS-485

RS-485 ออกแบบมาเพื่อช่วยการสื่อสารอินเตอร์เฟส ต่อจาก RS-232C (เป็นมาตรฐานที่ใช้กันกว้างขวาง) RS-422 และ RS-485 ใช้ในโรงงานรักษาสภาพแวดล้อมที่มีพื้นที่การรับส่งข้อมูลที่ระยะทางไกลได้ ใช้งานโดยต่อเข้ากับ ชุดอุปกรณ์ใช้งานได้สองแบบคือ desktop และ portable PC และสามารถใช้เชื่อมกันกับ RS-232-C และ ISDN-2 อีกด้วย

1) PCMCIA RS-422/RS-485 ports

2) 2 xRS-422/RS-485 ports

2.3.17 การใช้งาน PCMCIA RS-422/RS-485

ลักษณะเด่นของ PCMCIA RS-422 และ RS-485

พอร์ต	2 พอร์ต
UARTs	16c550 (16 byte FIFO)
I/O+ IRQ	I/O อัตโนมัติ ที่ขา IRQ
	สามารถแสดงการปรับสถานะ I/O และขา IRQ
	I/O เลือก 3E8h/2E8h (COM3/4), 330h ถึง 360h.
	IRQ เลือกขา 3 , 4 , 5 , 7 , 10 , 11 , 12 และ 15
Baudrate	50 bps - 115 kbps
Signals	Supports Tx , Rx , RTS and CTS singnals
Wiring	Two and four wirte operation
Connectors	2 x DB9 male
Form factor	PCMCIA type II
Power usage	+5V. 80mA typical

2.3.18 2 x RS-422 และ RS-485

ลักษณะเด่นของ 2 x RS-422 และ RS-485

พอร์ต	2 พอร์ต
UARTs	16550 (16 bit FIFO)
I/O	0x0000-0xFFFF
IRQ	2, 3, 4, 5, 7, 10, 11, 12, 15
Baudrate	50 bps - 921.6 kbps
Singnals	Support Tx, Rx, RTS and CTS singnals
Control mode	Automatic data direction or RTS controlled
Termination	120 ohm resistors, can be disabled
Connectors	2 x DB9 male
Form factor	157mm x 83mm
Power usage	+5V. 240mA max

2.3.19 การรับ - ส่งข้อมูล

การรับส่งข้อมูลของชุดควบคุมการจ่ายไฟสำหรับห้องโครงงานส่วนของการรับ - ส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ และไมโครคอมพิวเตอร์ ในโครงงานนี้จะใช้ 2 ส่วนด้วยกันคือ ส่วนรับ - ส่งข้อมูลระหว่างชุดควบคุมและไมโครคอมพิวเตอร์ อีกส่วนก็คือ ส่วนรับ- ส่งข้อมูลระหว่างชุดควบคุมการจ่ายไฟ และไมโครคอมพิวเตอร์ประกอบด้วยเฟรมต่าง ๆ ดังนี้

ไบต์สตาร์ท คือ ไบต์เริ่มต้นเพื่อบอกกับไมโครคอนโทรลเลอร์ ว่ามีข้อมูลเข้ามา โดยใช้ สัญลักษณ์ " * "

ไบต์เลขบอร์ด คือ ไบต์สำหรับเลือกว่าจะให้คอนโทรลเลอร์ ตัวใดทำงานโดยใช้ เลข 1, 2

ไบต์คำสั่ง คือ ไบต์ที่สั่งให้คอนโทรลเลอร์ส่งสัญญาณ ไปให้รีเลย์ทำงานจะใช้ สัญลักษณ์ S แทน C

S แทนการสั่งให้รีเลย์สวิตช์ปิด (ทำงาน)

C แทนการสั่งให้รีเลย์สวิตช์เปิด (ไม่ทำงาน)

ไบต์เลขเครื่อง คือ ไบต์ที่แสดงหมายเลขเครื่อง จะให้หมายเลขเครื่องใดทำงานโดยใช้ สัญลักษณ์ 0 -15

ไบต์สตอบ คือ ไบต์ที่แสดงว่าได้สิ้นสุดการส่งข้อมูล ใช้สัญลักษณ์เป็น " ; "

2.4 บาร์โค้ด

ในระบบที่มีการปฏิบัติการงานอัตโนมัติที่มีจำนวนมาก ๆ เครื่องจักรจะถูกนำมา ใช้เพื่อความสะดวกโดยการเปลี่ยนแปลงรูปแบบข้อมูลจากแบบเดิมที่มนุษย์เข้าใจเป็นรูปแบบ ของรหัสแทน ซึ่งรหัสนี้อาจใช้แทนตัวแปลเดียวหรือหลาย ๆ ตัวก็ได้ สำหรับงานที่แตกต่างกันไป

บาร์โค้ด คือรหัสที่ใช้แทนสิ่งเหล่านั้นในรูปของเครื่องจักรที่อ่านรหัสแถบสีดำและช่อง ว่างสีขาวในอัตราส่วนที่กำหนด ซึ่งจะแทนอักขระแต่ละตัว โดยมีเซนเซอร์เป็น ตัวอ่านเครื่องหมายจากแถบนั้นออกมาเพื่อประมวลผล ต่อไปในขั้นตอนของสัญญาณทางไฟฟ้า กรรมวิธีในการทำงานนั้นอาจเปรียบเทียบกับการทำงานในร่างกายมนุษย์คือมีสายตาเป็นตัวตรวจจับ และสมองเป็นตัวประมวลผลและหรือสั่งงาน

บาร์โค้ดจัดเป็นรูปแบบการใช้งานที่ง่ายที่สุด รวมทั้งราคาและความน่าเชื่อถือได้ นับว่าเหมาะสมที่สุดที่จะ ใช้งานกับระบบข้อมูลของคอมพิวเตอร์ ตัวอย่างการนำบาร์โค้ดไปใช้งานและ

ก็ช่วยงานได้มาก คือ ระบบไปรษณีย์อัตโนมัติ โดยการนำไปใช้คัดเลือกชนิดของจดหมาย และปลายทางที่จะส่งไป

ชนิดของตัวตรวจจับบาร์โค้ดทั่ว ๆ ไปแบ่งได้ 2 ประเภท คือชนิดมือถือและแบบที่ตั้งอยู่กับที่สำหรับแบบมือถือนั้น ผู้ปฏิบัติงานฝึกหัดงานเพียงเล็กน้อยก็สามารถที่จะทำงานได้ และสามารถที่จะทำงานได้รวดเร็ว และถูกต้องกว่าการใช้งานใช้คนป้อนข้อมูลมากอีกทั้งการเปลี่ยนแปลงรูปแบบบาร์โค้ด ในผลิตภัณฑ์นั้นก็ง่ายมาก มีข้อมูลที่น่าสนใจ เปรียบเทียบให้เห็นความผิดพลาดซึ่งเกิดจากการใช้บาร์โค้ดจะมีแต่ 1 ใน 10,000 ในขณะที่หากใช้คนป้อนข้อมูลความผิดพลาดจะสูงถึง 1 ใน 300 อัตราการผิดพลาด ที่ได้จากการใช้บาร์โค้ดสามารถลดลงโดยใช้วิธีการตรวจเช็คตัวเลขและ เทคนิคการป้องกันข้อมูล ในรูปแบบอื่นร่วมด้วย

ข้อแตกต่างของรูปแบบต่าง ๆ ของบาร์โค้ดทุกวันนี้มีอยู่มากมาย ขึ้นอยู่กับความเหมาะสมในงานแต่ละชนิดไป โดยจะขึ้นอยู่กับตัวถอดรหัสของบาร์โค้ด ซึ่งจะต้องตรงกับชนิดของบาร์โค้ดนอกเหนือจากจะต้องมีระบบของแหล่งกำเนิดแสง และตัวอ่านตามปกติแล้วการแสดงผลจะแสดงเอาต์พุตออกทางตัวเลขแสดงผล LED หรือต่อไปยังอินพุตของระบบคอมพิวเตอร์โดยการต่อผ่านทางพอร์ต RS232 มาตรฐาน

การใช้แสงเลเซอร์ในการอ่านบาร์โค้ดเพิ่งจะมีมาได้เมื่อต้นทศวรรษที่ 70 โดยใช้งานร่วมกับระบบไมโครโปรเซสเซอร์ซึ่งการประมวลจากตัวอ่านนี้ หากว่ามีความสนใจในการอ่านรูปแบบรหัสของบาร์โค้ดที่แตกต่างกันจะมีความยืดหยุ่นใน การใช้งานมากกว่า 1 รูปแบบในสายงานการผลิต ยกตัวอย่างในอุตสาหกรรมเวชภัณฑ์จะใช้รหัส UPC เป็นหลักในขณะที่ใช้รหัส 39 สำหรับการใช้งานรูปแบบใหม่

2.5 ชนิดของตัวบาร์โค้ด

ตัวบาร์โค้ดพื้นฐานมี 4 ชนิดดังแสดงในตารางที่ 2.3 ซึ่งจะบอกตั้งแต่ราคาต่อหน่วยและลักษณะการทำงาน สำหรับรายละเอียดของตัวอ่านแต่ละชนิด มีดังต่อไปนี้

2.5.1 ไดโอดเปล่งแสง (LED)

ตัวอ่านไดโอดแบบเปล่งแสงจะมีราคาต่อหน่วยต่ำแต่การทำงานนั้นอาจถูกรบกวนด้วยแสงสว่างจากสภาพแวดล้อมได้ การใช้งานตัวอ่าน ต้องสัมผัสกับวัตถุที่จะอ่านและตัวฉลากบาร์โค้ดหากเกิดความสกปรกจะทำให้ประสิทธิภาพ การอ่านข้อมูลลดลงโดยทั่วไป ๆ ไปแล้วความลึกในการฉายแสง (depth of field) จะมีค่าอยู่ในระดับ 0.075 นิ้ว

ตารางที่ 2.3 ชนิดของตัวอ่านบาร์โค้ด

ชนิดของตัวอ่านบาร์โค้ด	แหล่งกำเนิดแสง	ราคาต่อหน่วย
LED	ไดโอดเปล่งแสงสีแดง	ต่ำ
IR	แสงย่านอินฟาเรด	ปานกลาง
แสงแบนด์แคบ	แสงเลเซอร์	สูงสุด
ไฟเบอร์ออปติก	ใช้แสงจากสภาพแวดล้อมภายนอก	สูง

2.5.2 แสงอินฟราเรด (IR)

ตัวอ่านชนิดนี้จะคล้ายกับไดโอดเปล่งแสง แต่มีข้อดีกว่าคือ มีผลรบกวนจากแสงสว่างจากสภาพแวดล้อมรอบ ๆ น้อยมา และไม่มีปัญหาในการอ่านเนื่องมาจากความสกปรกของผิวบาร์โค้ด

2.5.3 แสงเลเซอร์ (Laser)

ปัจจุบันนี้ตัวอ่านที่ใช้แสงเลเซอร์จะมีราคาแพงที่สุดในบรรดาตัวอ่านบาร์โค้ด มีความลึกในการฉายแสง (depth of field) อยู่ในระดับ 3 นิ้ว ซึ่งก็หมายความว่าสามารถที่จะใช้งานอ่านบาร์โค้ด ที่ระยะไกลออกไปได้ ตัวอ่านไม่จำเป็นต้องติดกับผิวป้ายบาร์โค้ด โดยทั่วไปจะใช้แสงที่มีความยาวคลื่นประมาณ 750 นาโนเมตร ซึ่งเป็นแสงที่อยู่ในช่วงที่สายตาไม่สามารถมองเห็นได้ แต่การทำงานจะมี LED สีแดงช่วยเล็งหาเป้าหมายขณะทำงาน

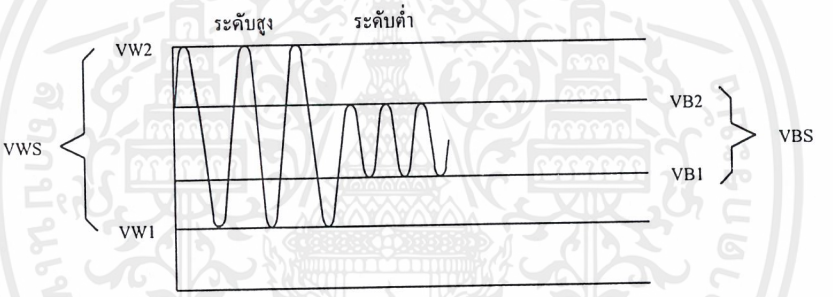
2.5.4 ไฟเบอร์ออปติก

ใช้แสงสว่างจากสิ่งแวดล้อม มีราคาต่อหน่วย อยู่ในระดับค่อนข้างสูง แต่ก็น้อยกว่าแบบเลเซอร์ ไฟเบอร์ออปติค คือเส้นใยแสงขนาดเล็กที่ทำหน้าที่เป็นตัวนำแสง โครงสร้างของเส้นใยแก้วนำแสงเดินทางผ่านเรียกว่า " Core " กับส่วนนอกเรียกว่า " Cladding " ซึ่งห่อหุ้มส่วนที่เป็น Core ไว้ทำหน้าที่ให้แสงอยู่เฉพาะภายใน Core และทำหน้าให้เส้นใยแก้วแข็งแรงมากขึ้น ทั้ง Core และ Cladding ทำจากสารไดอิเล็กตริกใส เช่น พลาสติกหรือแก้ว โดยการทำให้ค่าดัชนีการหักเหของ Core มากกว่า Cladding เล็กน้อยประมาณ 0.2 - 3 % และนอกจาก Core และ Cladding แล้ว เส้นใยแสงจะถูกหุ้มด้วย Buffer อีก 1-2 ชั้น (Buffer Coating เป็นไนลอนหรือพลาสติก) เพื่อเพิ่มความแข็งแรง และป้องกันรอยขีดข่วน

2.6 การทำงานของบาร์โค้ด

ตัวอ่านบาร์โค้ดหรือแสกนเนอร์เป็นอุปกรณ์ที่ใช้ในการตรวจจับบาร์โค้ด ซึ่งจะให้เอาต์พุตเป็นสถานะต่ำเมื่อพบแถบเส้นสีดำและมีสถานะสูงเมื่อพบสีขาวรูปแบบของสัญญาณดัง รูปที่ 2.14 ซึ่งทำหน้าที่ เหมือนตาในระบบบาร์โค้ด โดยการเปลี่ยนแถบเส้นขาวดำที่เห็นให้เป็นสัญญาณทางไฟฟ้าส่วนประกอบหลัก ๆ ที่ต้องนำมาพิจารณาในการตัดสินใจเลือกชนิดของตัวอ่านบาร์โค้ดคือ

1) รูปแบบของสัญญาณทางเอาต์พุตที่ต้องการว่าเป็นแอนะล็อกหรือดิจิตอล
2) ชนิดของตัวอ่านหรือแสงที่ใช้อ่าน
3) ตัวอ่านเป็นชนิดที่ต้องสัมผัสหรือไม่สัมผัสกับผิวของวัตถุ
4) ตัวอ่านอยู่กับที่หรือสามารถเคลื่อนย้ายได้
5) สิ่งแวดล้อมในบริเวณที่ใช้งานว่ามีสภาพแสงรบกวนต่อการทำงานหรือไม่



รูปที่ 2.13 ระดับแรงดันทางเอาต์พุตของตัวอ่านบาร์โค้ด Vws คือแรงดันเมื่อผ่านแถบขาว และ Vbs คือแรงดันเมื่อผ่านแถบดำ

การเลือกตัวอ่านที่เหมาะสม นั้นขึ้นอยู่กับความเข้ากันได้ทั้งหมดของการประยุกต์ใช้งาน เช่น รูปแบบของป้ายหรือฉลาก ตัวถอดรหัส และระบบทั้งหมด

การตัดสินใจว่าเลือกเอาต์พุตว่าให้เป็นอนาลอก หรือดิจิตอลจะต้องพิจารณาด้านความต้องการทางอินพุตของส่วนถอดรหัสข้อมูล และชนิดของแสงที่ต้องใช้อ่านจะขึ้นอยู่กับสภาพแวดล้อมภายนอกประเภทของการนำไปใช้งาน และ ชนิดของป้ายบาร์โค้ดที่มีตัวอ่านเป็นอินฟาเรดสามารถอ่านป้ายที่มีความสกปรก ซึ่งความสกปรกนี้จะพบได้บ่อยในบริเวณการบรรจุหีบห่อสินค้า

ชนิดของหมึกที่ใช้พิมพ์ป้าย บาร์โค้ด ต้องสัมผัสกับแสงที่ใช้ในตัวอ่านด้วยส่วนการเลือกตัวอ่านเป็นชนิดอยู่นิ่งหรือเคลื่อนที่รวมทั้งสัมผัส กับ ผิวหรือไม่ขึ้นอยู่กับประเภทของการใช้งาน

หลังจากได้ข้อมูลจากแสกนเนอร์ หรือตัวอ่านบาร์โค้ดแล้ว สัญญาณจะถูกส่งต่อมายังส่วนประมวลผลข้อมูล เพื่อแปลความหมายโดยการเปรียบเทียบสัญญาณทางไฟฟ้าที่ได้จากการอ่านสัญญาณบาร์และสเปซ สิ่งที่ใช้พิจารณาการเลือกตัวถอดรหัสแปลผลสำหรับบาร์โค้ด

1) ความเข้ากันได้กับชนิดของตัวอ่าน
2) สัญญาณทางเอาต์พุตที่ต้องการ
3) ต้องการชนิดที่เคลื่อนย้ายได้หรือไม่
4) แสดงผลเพียงอย่างเดียวหรือจะให้พิมพ์ข้อมูลออกมาด้วย
5) ต้องการคีย์ในการป้อนข้อมูลหรือไม่
6) ความสามารถในการถอดรหัสบาร์โค้ดได้หลายรูปแบบหรือไม่
7) ปัจจัยการใช้งานทางด้านสภาพแวดล้อม

เครื่องมือที่ใช้อ่านบาร์โค้ด หรือว่าตัวประมวลผลข้อมูล จะต้องมีระบบควบคุมระดับการอ่านข้อมูลอัตโนมัติ (Automatic Gain Control) เพื่อชดเชยกรณีที่แถบบาร์โค้ดมีการพิมพ์แถบไม่ชัดเจน หรือมีความเข้มเบาบางเกินไป

ตารางที่ 2.4 ผลของช่องแสงที่มีแรงดันต่อเอาต์พุต

ขนาดของช่อง (นิ้ว)	ค่าแรงดันต่ำสุด (มิลลิโวลต์)	ค่าแรงดันปกติ (มิลลิโวลต์)
แรงดันจากแถบกว้าง		
0.004	100	150
0.006	200	300
0.008	400	600
0.010	620	930
0.012	900	1350
แรงดันจากแถบแคบ		
0.004	50	90
0.006	100	180
0.008	200	360
0.010	310	558
0.012	450	810

2.7 ผลของช่องรับแสง

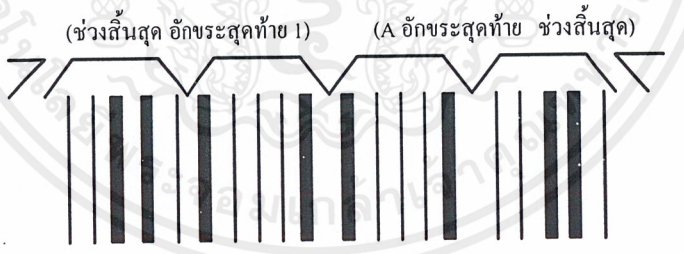
ตัวอ่านบาร์โค้ดจะมีช่องแสงเป็นทาง ให้แสงสะท้อนจากผิวบาร์โค้ด ผ่านส่วนตรวจจับ และช่องแสงนี้ ต้องมีขนาดเล็กกว่าความกว้างของแถบบาร์โค้ด แรงดันที่เอาต์พุตจะขึ้นอยู่กับขนาดของช่องแสงตัวแสดงในตารางที่ 2.4

ค่าพารามิเตอร์ที่สำคัญอีกค่าหนึ่งของตัวอ่านบาร์โค้ด คือ ค่าของดัชนีความละเอียด (Resolution Index หรือ RI) อัตราส่วนของสัญญาณแถบแคบ VD (Narrowbar Signal) หารด้วยค่าของสัญญาณแถบกว้าง VS (Widebar Signal) โดยที่ค่าของสัญญาณแถบแคบ VD คือ ระดับของสัญญาณไฟฟ้าที่อ่านผ่านแถบที่แคบที่สุดของบาร์โค้ด และ สัญญาณแถบกว้าง VS คือ ค่าความแตกต่างระหว่างขนาดของสัญญาณที่ได้จากบริเวณแถบดำ และ แถบขาว ของป้ายบาร์โค้ด สัมพันธ์กับสัญญาณที่สร้าง โดยความกว้างของบาร์และสเปซหาได้โดยใช้สมการดังต่อไปนี้

$$RI = VD / VS \times 100\ \%$$

เมื่อ $VS = V_{w2} - V_{w1}$

$VD = V_{w1} - V_{B1}$



รูปที่ 2.14 รูปแบบของการเข้ารหัส LA

ซึ่งการวัดอ้างอิงตามมาตรฐาน NBS (Nationnal Bureal of Standard) สัญญาณที่วัดได้จากตัวอ่านบาร์โค้ดจะขึ้นกับความเร็วในการสแกนผ่านไปบนป้ายเวลาขาขึ้น และเวลาขาลงของสัญญาณที่อ่านจะอยู่ระหว่าง 10 เปอร์เซ็นต์ และ 90 เปอร์เซ็นต์ของสัญญาณภายในเวลา 40 ไมโครวินาที

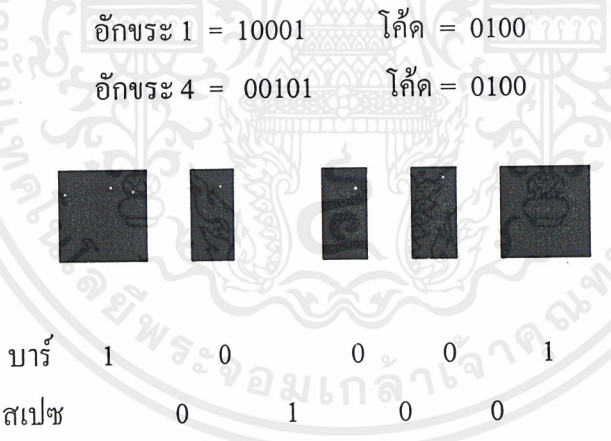
2.8 รูปแบบของรหัสบาร์โค้ด

ความแตกต่าง ของรหัสตัวอักขระบาร์โค้ดที่ได้รับการพัฒนาขึ้นมา ใช้งานในทุกวันนี้มีลักษณะรูปแบบมากมาย ซึ่งจะเปลี่ยนแปลงขึ้นอยู่กับรูปแบบการตรวจเช็คความผิดพลาดความหนาแน่นในการพิมพ์อักขระต่อนิ้ว ชนิดของตัวอักขระที่ใช้งาน ไม่ว่าจะเป็นตัวอักขระหรือว่าตัวเลข ซึ่งสามารถนำมาเข้ารหัส และ ประยุกต์ใช้งานจริงได้รูปแบบ ของรหัสบาร์โค้ดที่ใช้กันในทุกวันนี้มี 5 แบบ หลัก ๆ ดังนี้

2.8.1 รหัส 3 ใน 9 หรือรหัส 39

รหัส 39 ประกอบด้วยส่วนประกอบแถบกว้าง 3 ส่วนซึ่งเป็นแถบทึบหรือบาร์และแถบว่างหรือสเปซ จากทั้งหมด 9 ส่วน ดังแสดงในรูปที่ 2.7 ซึ่งในบาร์โค้ดจะประกอบด้วย

1. ช่วงแถบว่างที่อยู่แต่ละด้านของบาร์โค้ด
2. ส่วนแสดงการเริ่มต้น
3. ข้อมูลของตัวอักขระ



รูปที่ 2.15 อักขระในบาร์โค้ด 3 ใน 9

1) รูปแบบโครงสร้างของรหัส 39

ในรหัสแบบ 39 ความกว้างของแถบบาร์ และ สเปซ จะอยู่ในรูปแบบของตัวเลขฐานสองโดยแถบที่แคบจะแทนด้วย 0 และแถบกว้างจะแทนด้วยเลข 1 ดังนั้น รหัส 3 ใน 9 ข้อมูล 1 ฟิลด์ จะประกอบด้วย แถบกว้าง 3 แถบ จึงมีเลขฐาน 2 ค่า 1 อยู่ 3 ตัวและที่เหลือจะเป็นค่า 0 อยู่ 6 ตัว

รหัสของแถบบาร์โค้ดจะประกอบด้วยรหัสเริ่มต้นทางด้านซ้ายสุด และ รหัสหยุดที่ทางขวาสุด ขอบเขตระหว่างแถบที่แสดงการเริ่มต้น และ แถบหยุด จะเป็นส่วนบรรจุข้อมูลซึ่งสามารถบรรจุสูงสุดได้ถึง 32 ตัวอักขระ แต่ก็จะขึ้นอยู่กับความสามารถของอุปกรณ์ที่ใช้ร่วมด้วย

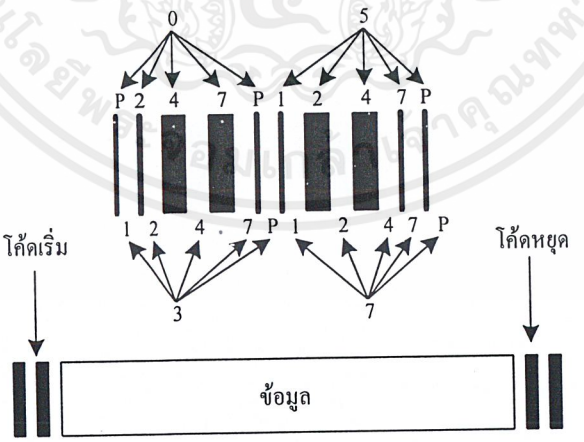
2) ผังจัดการข้อมูลสตริงในรหัส 39

รหัสโค้ด 39 นี้มีอยู่ 2 ระดับ (กว้างและแคบ) นั่นคือแถบเส้นและช่องว่างจะเป็นได้ 2 ระดับ ไม่กว้างก็แคบต่อช่วงกว้างเท่ากับ 1: 2.5

อักขระแต่ละตัวจะถูกแทนด้วยส่วนประกอบ 9 ส่วน โดยจะเป็นแถบเส้น 5 ส่วน และส่วนช่องว่างอีก 4 ส่วน สำหรับ 3 ใน 9 ส่วนนั้น เป็นส่วนกว้างซึ่งก็คือแถบเส้น หรือ ช่องกว้าง (Wide) แทนด้วยไบนารี่ "1" และส่วนที่แคบ (Narrow) ก็แทนด้วยไบนารี่ "0" และจะมีช่องว่างแคบ ๆ กั้นระหว่างอักขระแต่ละตัว

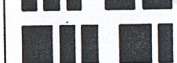
3) การจัดการข้อมูลสตริงในรหัสแทรก 2 ใน 5

ตัวอย่างของข้อมูลบาร์โค้ด สำหรับเลข 1 และเลข 4 เปรียบเทียบให้ดูดังแสดงในรูปที่ 2.15 ตัวอักษรทั้งหมดของรหัส 39 แสดงไว้ในตารางที่ 2.5 แถบช่องว่างหรือว่าบาร์ และสเปซแต่ละอันสามารถเลือกได้ ซึ่งจะแคบหรือกว้างขึ้นอยู่กับการแปลงรหัส ตัวอักขระแต่ละตัวประกอบด้วยแถบกว้าง 3 แถบ และแถบแคบ 6 แถบตัวเลข 1ใช้แทนส่วนกว้าง และเลข 0 แทนส่วนแคบตัวอักษรอื่น ๆ แบ่งแยกโดยช่องว่างรูปที่ 2.16 รูปแบบของบาร์โค้ดรหัสแทรก 2 ใน 5



รูปที่ 2.16 รูปแบบของบาร์โค้ดรหัสแทรก 2 ใน 5

ตารางที่ 2.5 โครงสร้างตัวอักขระในรหัส

ตัวอักขระ	รูปแบบ	บาร์	สเปซ	ตัวอักขระ	รูปแบบ	บาร์	สเปซ
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	,		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	Space		01010	1000
H		10010	0010	@		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		00001	0001	%		00000	0111

2.8.2 รหัสแทรก 2 ใน 5

เฉพาะข้อมูลตัวเลขเท่านั้นที่สามารถเข้ารหัสแบบแทรก 2 ใน 5 ความหนาแน่นของข้อมูลสูงสุด คือ 18 ตัวอักขระต่อนิ้วรหัสอาจผิดพลาดได้ หากไม่เข้ารหัสเป็นตัวเลขคู่ของตัวอักษรเข้ารหัสในสัญลักษณ์ตัวอักษรตัวแรกของคู่แทนโดยบาร์ และตัวอักษรตัวที่ 2 แทน โดยสเปซรหัสแทรก 2 ใน 5 ในอุตสาหกรรมผลิยา , ในร้านอาหารและอุตสาหกรรมผลิตสิ่งพิมพ์ ฯลฯ

1) คุณสมบัติของรหัสแทรก 2 ใน 5 หรือ บาร์โค้ด USS - I 2/5

ชนิดของตัวอักขระ : ตัวเลข

ความยาวของข้อมูล : เปลี่ยนแปลงได้ แต่ต้องเป็นจำนวนคู่

การถอดรหัส : ได้ทั้งสองทิศทาง (Bi - Directional)

ความหนาแน่นของข้อมูล : สูงสุด 18 ตัวอักขระต่อนิ้ว

ตัวอักขระพิเศษ : มีความแตกต่างกันรูปของการเริ่มและการหยุด

ตารางที่ 2.6 ชุดอักขระของบาร์โค้ดรหัสแทรก 2 ใน 5

ตัวเลข (ฐานสิบ)	9 รหัส 2 ใน 5 (ดัดแปลงจาก BCD)	ค่าไบนารี
0	00110	6
1	10001	17
2	01001	9
3	11001	25
4	00101	5
5	10100	20
6	01100	12
7	00011	3
8	10010	18
9	01010	10
อักขระเริ่มต้น	00	0
อักขระหยุด	10	2

2) คุณสมบัติของรหัส Codebar

ตัวอักขระ : ตัวเลข 0 ถึง 9 รวมทั้งตัวอักขระพิเศษอีก 6 ตัว คือ $, - , : , / ,. และ + รวมทั้งตัวอักขระหยุด - เริ่มอีก 4 ตัว A , B , C และ D

ความยาวของชุดข้อมูล : เปลี่ยนแปลงได้

การถอดรหัส : ได้ทั้งสองทิศทาง

ความหนาแน่นข้อมูล : สูงสุด 12.8 ตัวอักขระต่อนิ้ว

3) รูปแบบโครงสร้างของรหัส 2 ใน 5

รูปแบบสำหรับตัวอักขระจะประกอบด้วยบาร์ และสเปซสลับกันไป แทนค่าตัวเลขฐานสิบในรูปของตัวเลขฐานสอง 5 บิต (4 บิตแสดงค่าตัวเลข และ 1 บิตพาริตี้)รูปแบบของบาร์โค้ด จะประกอบด้วยส่วน เริ่มต้น ส่วนของข้อมูล และส่วนการแสดงการสิ้นสุดของรหัส

อักขระ โค้ด

5 10100

บาร์ 0 1 0 0

สเปซ 1 0 1 0 0

(ก) แสดงส่วนประกอบของรหัสแบบแทรก 2 ใน 5

อักขระ โค้ด

1 10001

4 00101

บาร์ 1 0 0 0 1

สเปซ 0 0 1 0 1

(ข) แสดงการอ่านรหัสจากแถบบาร์โค้ด

รูปที่ 2.17 ผังการจัดวางบาร์โค้ดรหัสแทรก 2 ใน 5

การเข้ารหัสตัวเลข ต้องประกอบด้วยจำนวนคู่เสมอ ตัวเลขที่เข้ารหัสแล้ว ดังรูปที่ 2.18 สัญลักษณ์ ของบาร์โค้ดบ่อยครั้ง จะประกอบด้วย ตัวอักษรที่คนเราสามารถอ่านได้อย่างง่าย ๆ ซึ่งทำให้ง่ายต่อการตรวจเช็คเมื่อไม่สามารถอ่านบาร์โค้ดได้โดยตัวเลข จะเขียนไว้ด้านบนหรือด้านล่างของบาร์โค้ดเสมอ ความยาวของแถบบาร์โค้ดสามารถคำนวณด้วยสมการ

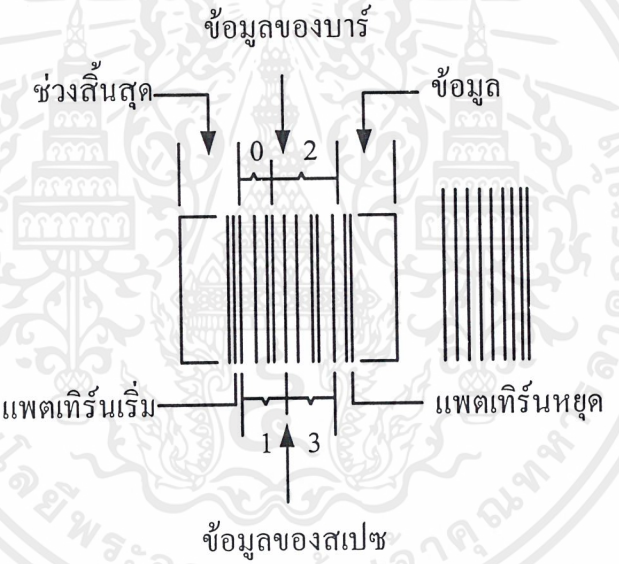
$$L = [P(4N + 6)]X + 2Q$$

ซึ่ง P คือ จำนวนของคู่อักขระ

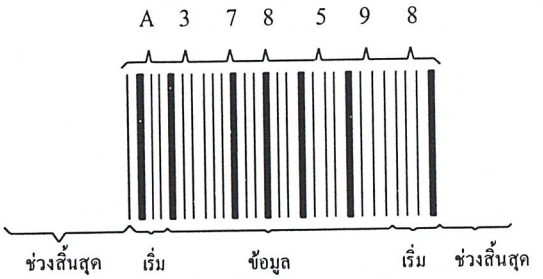
N คือ อัตราส่วนของแถบกว้าง

X คือ ขนาดความกว้างของแถบแคบ

และ Q คือ ความกว้างของแถบแสดงการสิ้นสุดรหัส



รูปที่ 2.18 บาร์โค้ดสมบูรณ์ของจำนวน 0123 ในรูปแบบรหัสแทรก 2 ใน 5



รูปที่ 2.19 การเข้ารหัส Codebar A37859B

ส่วนประกอบของข้อมูล ดังแสดงในรูปที่ 2.17 ในรูป (ก) แสดงส่วนประกอบของรหัสแบบแทรก 2 ใน 5 โดยอ่านค่าจากแถบได้ 5 และจากช่องว่างก็ได้ 5 โดยดูจากแถบหรือช่องว่างกว้างแทนด้วย 1 และในทางตรงกันข้ามแถบหรือช่องว่างแคบแทนด้วย 0 ส่วนในรูปที่ 2.17 (ข) แสดงการอ่านรหัสจากแถบได้ 1และจากช่องว่างได้ 4

4) ส่วนประกอบของบาร์โค้ด

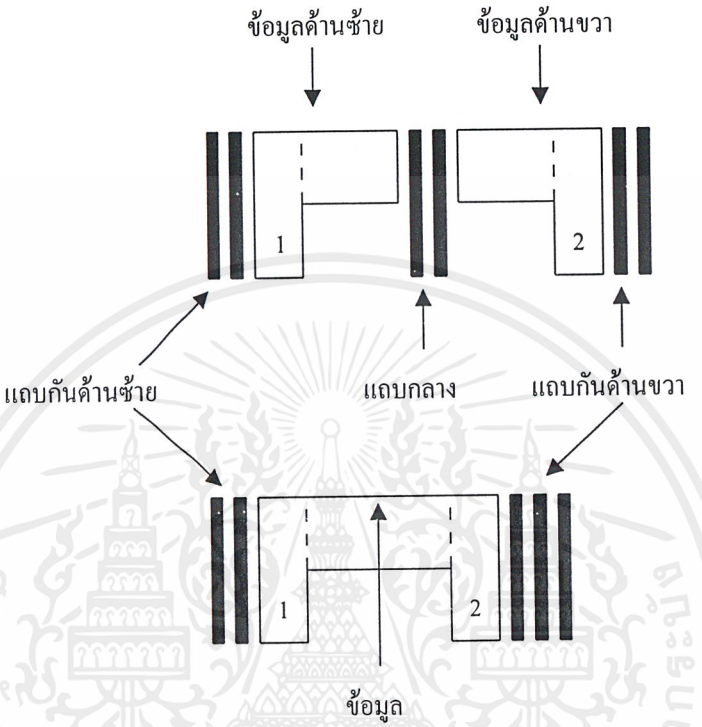
ส่วนแสดงการเริ่มต้น และ หยุดของบาร์โค้ดแสดงในรูปที่ 2.18 ส่วนเริ่มต้นจะอยู่ทางซ้ายของข้อมูลทั้งหมด ประกอบด้วย 4 ส่วนแคบ ๆ โดยสลับกันระหว่างบาร์และสเปซ ส่วนแถบแสดงการหยุดจะอยู่ทางด้านขวาของข้อมูลทั้งหมด ประกอบด้วยแถบใหญ่แล้วตามด้วยแถบเล็กโดยมีช่องว่างแคบ ๆ สลับกันไว้และ ในส่วนประกอบของบาร์โค้ดจะมีส่วนที่แสดงการสิ้นสุดของรหัสที่อยู่ปิดหัวท้ายของรหัส

2.8.3 รหัสแบบ Codebar

Codebar สามารถใช้กับข้อมูลตัวเลขและตัวอักขระพิเศษอีก 6 ตัวคือ $, - , : , / ,. , และ + และตัวอักษร 4 ตัวที่แสดงการเริ่มต้นและหยุด คือ A, B, C และ D สัญลักษณ์ของ Codebar ใช้สำหรับ เปลี่ยนแปลงความยาวของข้อมูล ซึ่งคุณสมบัติต่าง ๆ ได้ให้ไว้ในตารางที่ 2.7 ส่วนประกอบของข้อมูลแต่ละตัวจะแสดงให้ในรูปที่ 2.19

Codebar แต่ละตัวประกอบด้วยขอบเขตแสดงการสิ้นสุด,ส่วนแสดงการเริ่มต้นหรือหยุดและส่วนของข้อมูล ซึ่งข้อมูลสามารถเปลี่ยนแปลงความยาวได้ถึง 32 ตัวอักษร ตัวอย่างดังแสดงในรูปที่ 2.19 ตัวอักษร แต่ละตัวแทนโดยส่วนประกอบ 7 ส่วน มี 4 บาร์และ 3 สเปซ ระหว่างแถบตัวอักขระแสดงว่าเริ่มต้น หรือ หยุดมี 4 ตัว สามารถใช้เป็นตัวเริ่มต้นหรือหยุดได้ ส่วนประกอบการเพิ่มข้อมูลสัญลักษณ์ภายในตัวอักขระ แบ่งโดย ช่องว่างระหว่างตัวอักขระประกอบด้วยส่วนของช่องว่างแคบ ๆ 1 ช่อง ส่วประกอบเบื้องต้นสำหรับตัวอักษร Codebar ทั้งหมด

แสดงไว้ในตารางที่ 2.7 ซึ่งส่วนประกอบกว้างแทนไบนารี 1 และส่วนประกอบแคบ แทนไบนารี 0 แต่ละตัวอักขระสามารถแทนโดยไบนารีขนาด 1 บิต เท่านั้น



รูปที่ 2.20 รูปแบบการเข้ารหัส UPC

2.8.4 รหัสสากล UPC (Universal Product Code)

บาร์โค้ด UPC ชุดอักขระประกอบด้วยตัวเลขและอีก 3 ส่วนพิเศษ คือส่วรเริ่มต้น, ส่วนหยุดและตัวอักขระตัวอักษรแต่ละตัวสร้างขึ้นโดย 4 ส่วนคือ 2 บาร์และ 2 สเปซ มี 2 ตัวอย่างข้อมูลพื้นฐานดังแสดงในรูปที่ 2.20

2.8.5 รหัสสากล UPC (Universal Product Code)

รหัสสากล UPC จะใช้เพื่อเข้ารหัสรายการต่าง ๆ ในบัตรประจำตัวในหน่วยงานต่าง ๆ และใช้ในระบบการบรรจุหีบห่อสินค้าอุปโภคบริโภคในสหรัฐฯ โดยที่ประชุมของ UPC ซึ่งไม่เป็นสากลเท่าไรนัก ระบบตัวอักษรเฉพาะและชนิดพิเศษใช้ของข้อมูล การจัดการของระบบตัวเลข ดังแสดงในตารางที่ 2.7 ตัวเลขของรหัสสากลที่แสดงให้เห็นในตารางที่ 2.8 ข้อมูลตัวอย่างสำหรับตัวเลข 5 มี 2 แบบ ได้แสดงไว้ในรูปที่ 2.20 ซึ่งแถบทึบจะแทนค่า 1 และช่องว่างจะแทนค่า

0 ในสัญลักษณ์ของบาร์โค้ดรหัสประเภทนี้มีความผิดพลาดต่ำ จึงได้มีการพัฒนาไปประยุกต์ใช้กับเครดิตการ์ด

ตารางที่ 2.7 ระบบจำนวนของตัวอักขระ

ตัวอักขระ	การใช้งาน
0	รหัส UPC ปกติ
2	รายการสุ่มน้ำหนักจำพวกพืชผลและเนื้อสัตว์
3	รายการเกี่ยวกับสุขภาพเช่น รหัสสากลของยา
4	การปฏิบัติการในร้านที่ไม่ใช่รายการอาหารพร้อมรหัสตัวเลขเพื่อตรวจสอบและป้องกัน รวมทั้งการใช้งานในรูปแบบที่มีจำกัด
5	สำหรับใช้กับคูปอง
อื่น ๆ	สำรองไว้ใช้งาน

ตารางที่ 2.8 ตัวเลขของรหัสสากล

ตัวอักขระ	อักขระฟิลด์ซ้าย พาริตี้คี่	อักขระฟิลด์ขวา พาริตี้คู่
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	1011100
5	0110001	1001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100

2.8.6 รหัสตัวเลขของยุโรป EAN (European Article Numbering)

ลักษณะรหัสตัวเลขของยุโรปที่ใช้ในทวีปยุโรป ซึ่งมักจะใช้คู่กับรหัส UPC เป็นรหัสที่ใช้สำหรับข้อมูลตัวเลขตัวอย่างดังแสดงในรูปที่ 2.21 และ 2.22 ซึ่งเป็นรหัส EAN-13

1) การพิมพ์ฉลากบาร์โค้ด

ในอุตสาหกรรมผลิตบาร์โค้ด โดยทั่ว ๆ ไปป้ายบาร์โค้ด ที่ใช้จะใช้การพิมพ์โดยคอมพิวเตอร์โดยเครื่องพิมพ์ดอตเมตริกซ์ จะคำนึงถึงชนิดของรหัสที่ใช้เป็นสำคัญบาร์โค้ดจะพิมพ์โดยมาตรฐานที่จะทำให้ผิดพลาดน้อยที่สุดผู้ใช้บาร์โค้ดโดยทั่ว ๆ ไป มีระบบการตรวจสอบ 4 ตัวแปรที่สำคัญคือรูปแบบการวางป้าย รูปแบบการเข้ารหัส คุณภาพความเข้ม (Contrast) ของการพิมพ์ และ ความกว้างของแถบและช่องว่างของบาร์โค้ดการเปลี่ยนแปลงในความกว้างของแถบและช่องว่างบนบาร์โค้ดที่กำหนด จะเกิดจากกรรมวิธีในการพิมพ์บาร์โค้ด การสร้างป้ายบาร์โค้ดจะต้องขึ้นอยู่กับความต้องการของผู้ใช้ป้ายบาร์โค้ด สามารถเลือกได้มีทั่งที่พิมพ์ไว้แล้ว และระบบที่สามารถนำไปสร้างบาร์โค้ดใหม่ได้เองตามต้องการ

ป้ายบาร์โค้ดที่มีใช้งานอยู่ทุกวันนี้แบ่งได้เป็น 2 รูปแบบตามลักษณะการพิมพ์ คือมีบาร์โค้ดที่พิมพ์สำเร็จไว้แล้วและป้ายบาร์โค้ดที่ต้องพิมพ์ใช้เอง ซึ่งแต่ละแบบก็มีข้อดีข้อเสียอยู่ในตัวเอง

ข้อดี ของการใช้ป้ายสำเร็จคือ

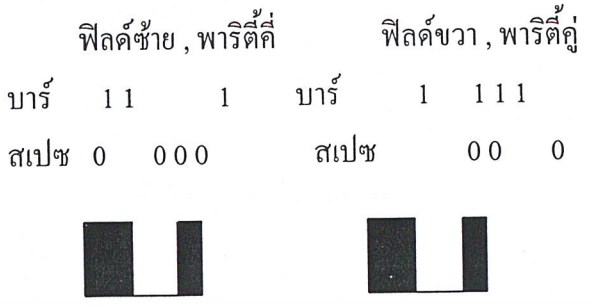
1) สามารถที่จะพิมพ์บาร์โค้ดที่มีความสามารถหนาแน่นข้อมูลสูง ๆ ได้
2) มีความเชื่อถือได้ของการพิมพ์สูงเพราะส่วนใหญ่จะพิมพ์โดยระบบออฟเซต
3) สามารถที่จะพิมพ์บนวัสดุอื่นนอกเหนือจากการพิมพ์โดยกระดาษได้
4) ไม่ต้องยุ่งยากหาเครื่องพิมพ์
5) สามารถที่จะสร้างบาร์โค้ดติดเพื่อการบรรจุในผลิตภัณฑ์ที่แตกต่างกันได้

ข้อเสีย

1) ราคาต่อหน่วยจะสูงกว่า
2) จะต้องเตรียมข้อมูลของป้ายไว้ล่วงหน้าก่อน

เริ่ม	NSC	ซ้าย	กลาง	ขวา	ตรวจสอบ	หยุด
101	0	0	01010	5	ตรวจสอบ	101

รูปที่ 2.21 ผังการจัดวางข้อมูลสตริงของบาร์โค้ดรหัส UPC



รูปที่ 2.22 ฟิลด์ซ้ายและฟิลด์ขวาของข้อมูลตัวอักขระ

การพิมพ์บาร์โค้ดโดยเครื่องพิมพ์สามารถที่จะพิมพ์นอกสถานที่ได้ จากเครื่องพิมพ์หลายรูปแบบถึงแม้ว่าจะมีคุณภาพของบาร์โค้ดด้อยกว่าใช้ป้ายสำเร็จ แต่มีความยืดหยุ่นในการใช้งานสูง เครื่องพิมพ์ใช้ได้หลายแบบคือ

1) เทอร์มอลพรินเตอร์

1.1) ราคาของป้ายขึ้นอยู่กับชนิดของกระดาษที่ใช้

1.2) คุณภาพการพิมพ์อยู่ในระดับปานกลาง

1.3) ความคงทนของป้ายปานกลาง

1.4) ราคาของระบบมีตั้งแต่ราคาถูกถึงปานกลาง

2) ดอตเมตริกพรินเตอร์

2.1) ใช้กระดาษชนิดราคาถูก

2.2) ความหนาแน่นของการพิมพ์ปานกลาง

2.3) พิมพ์บาร์โค้ดได้เต็มรูปแบบ

2.4) ราคาของระบบอยู่ในช่วงกว้าง

2.3) คุณภาพการพิมพ์ไม่แน่นอน

3) การพิมพ์ตัวอักขระเต็ม

3.1) ราคากระดาษต่ำ

3.2) อุปกรณ์มีราคาสูง

3.3) สามารถพิมพ์ป้ายที่มีความหนาแน่นสูงได้

3.4) ป้ายอาจหลุดได้ง่าย

4) เลเซอร์พรินเตอร์

4.1) อุปกรณ์มีราคาสูง

4.2) พิมพ์บนพื้นผิวได้หลายชนิด

4.3) คุณภาพการพิมพ์สูง

รูปที่ 2.23 บาร์โค้ดรหัส EA

จะเห็นว่ารูปแบบของบาร์โค้ดปัจจุบันมีหลายชนิด ขึ้นอยู่กับประเภทของงานซึ่งนิยมใช้กันมากโดยเฉพาะในระบบงานบริการต่าง ๆ ที่ต้องการความรวดเร็วและความถูกต้องในการทำงานสูงซึ่งหากใช้บาร์โค้ดแล้ว จะช่วยให้มีความยืดหยุ่นในการทำงาน นับเป็นการนำเทคโนโลยีมาใช้อำนวยความสะดวกในอีกรูปแบบหนึ่ง

2.9 การอินเตอร์เฟสกับคีย์บอร์ด

โครงงานที่ถูกสร้างขึ้นส่วนมากแล้วจำเป็นต้องติดต่อกับอุปกรณ์อินพุต เช่น ตัวเซ็นเซอร์ต่าง ๆ สัญญาณไฟฟ้า , คีย์บอร์ด เป็นต้น โดยเฉพาะคีย์บอร์ดที่ใช้ติดต่อกับผู้ใช้งานในการควบคุมเครื่อง การอินเตอร์เฟสกับคีย์บอร์ด 6 รูปแบบที่ใช้งานกันอยู่ทั่วไปสำหรับตรวจจับสถานะการปิดเปิดของสวิตช์ในที่นี้คงยกตัวอย่างเฉพาะ การเชื่อมต่อสวิตช์ในแต่ละรูปแบบเป็นหลักเพื่อให้สามารถมองเห็นแนวทางการเขียนโปรแกรมได้ซึ่งแต่ละรูปแบบการเชื่อมต่อสวิตช์ก็จะมี ข้อดีข้อเสียแตกต่างกันไป

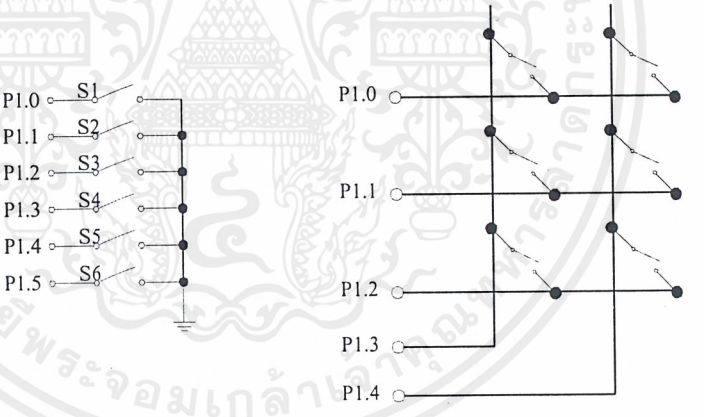
2.9.1 แบบเชื่อมต่อสวิตช์โดยตรงกับพอร์ต

รูปที่ 2.24 (ก) การเชื่อมต่อสวิตช์โดยตรงกับพอร์ตซึ่งเป็นแบบที่ง่ายที่สุด การกดสวิตช์แต่ละตัวจะทำให้ขาพอร์ตถูกต่อลงกราวด์โดยตรง ซึ่งไม่จำเป็นต้องมีวงจรจำกัดกระแสก็ได้เพราะว่าภายในตัวไมโครคอนโทรลเลอร์มีตัวต้านทานพูลอัพต่ออยู่แล้ว การเขียนโปรแกรมทำได้ง่ายมาก โดยการใช้คำสั่งตรวจสอบสถานะของแต่ละบิตในพอร์ต เช่นคำสั่ง JB หรือ JNB ข้อเสียของวง

จรแบบนี้คือสิ้นเปลืองจำนวนขาพอร์ตจำนวนมากถ้าต้องการใช้สวิตช์มากเท่าใด ก็ต้องขยายพอร์ตให้มากตามไปด้วย ซึ่งเป็นเรื่องยุ่งยาก เช่น ถ้าต้องการออกแบบเป็นคีย์บอร์ดสำหรับป้อนตัวอักษรขนาด 60 คีย์ผู้ออกแบบต้องจัดสร้างขาพอร์ตให้ได้ถึง 60 ขาพอร์ต ซึ่งแทบจะเป็นไปได้ยากและสิ้นเปลืองอุปกรณ์มาก

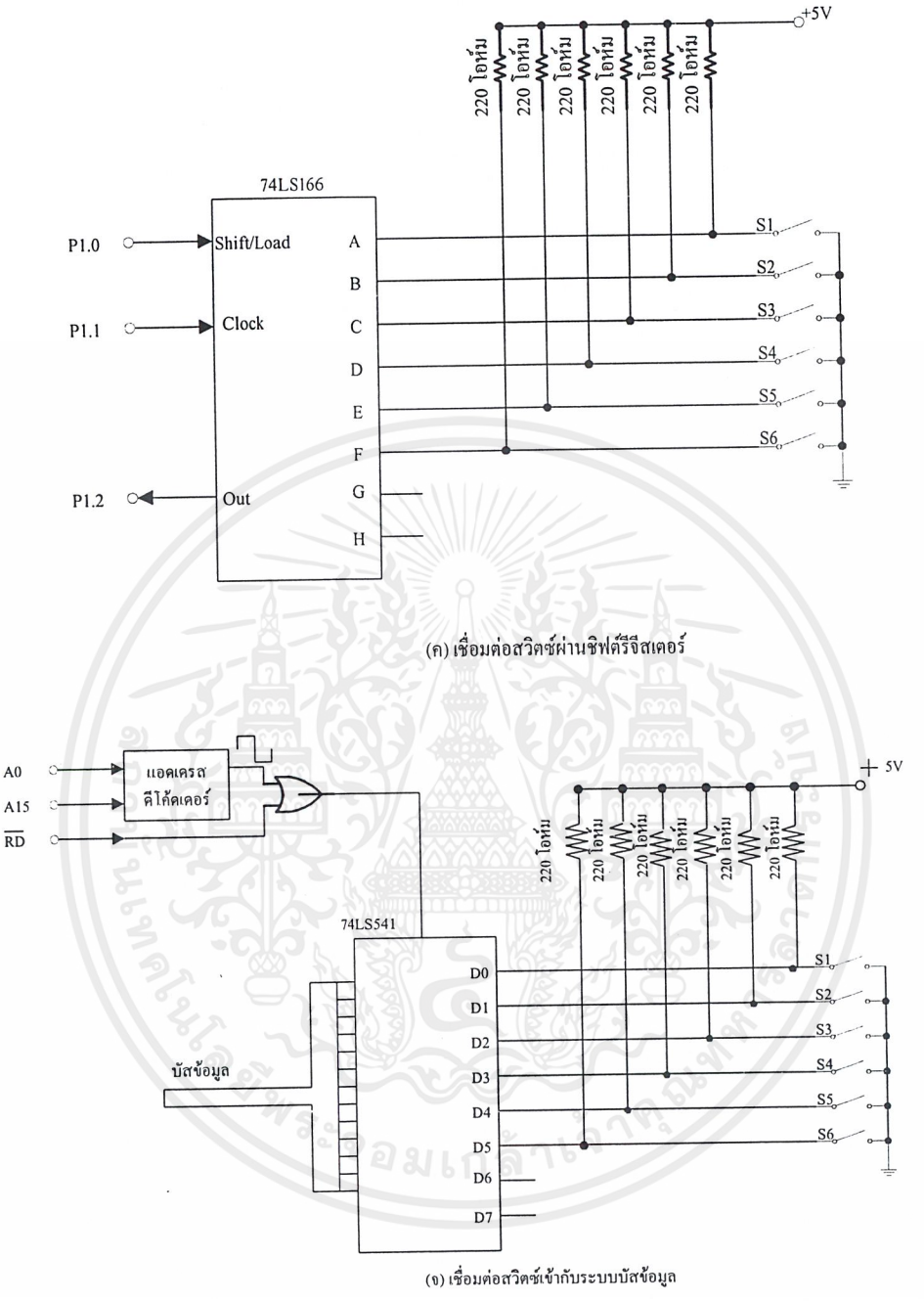
2.9.2 การเชื่อมต่อสวิตซ์แบบเมทริกซ์

รูปที่ 2.25 (ข) การเชื่อมต่อสวิตช์แบบเมทริกซ์แต่ละตัวจะถูกเชื่อมต่อกันแบบแถว และคอลัมน์ในรูปแบบของเมทริกซ์ การตรวจสอบการกดคีย์ใดบนคีย์บอร์ดทำได้โดยการป้อนค่าตรวจสอบค่าหนึ่งไปยังด้านคอลัมน์และ ตรวจสอบการเปลี่ยนแปลง ที่เกิดขึ้นทางด้านแถว หรือกล่าวได้ว่าตำแหน่งของการกดคีย์ได้จากการเขียนโปรแกรมทางด้านสแกนคีย์ด้านแถวข้อดีของ การเชื่อมต่อสวิตช์แบบเมทริกซ์ คือสามารถใช้สวิตช์ได้มากขึ้น ในขณะที่สิ้นเปลืองขาพอร์ตจำนวนน้อย เช่นใช้ขาพอร์ตเพียง 16 ขาพอร์ตสามารถต่อคีย์บอร์ดได้ถึง 64 คีย์ ข้อเสียของวงจรนี้คือไม่สามารถรับการกดคีย์พร้อมกันได้ และต้องเขียนโปรแกรมในการตรวจสอบคีย์ที่ยุ่งยากซับซ้อน



(ก) เชื่อมต่อสวิตช์แบบมัลติเพล็กซ์ (ข) เชื่อมต่อสวิตช์แบบเมทริกซ์

รูปที่ 2.24 การเชื่อมต่อสวิตช์แบบต่าง ๆ



รูปที่ 2.24 (ต่อ) การเชื่อมต่อสวิตช์แบบต่าง ๆ

2.9.3 แบบเชื่อมต่อสวิตช์ผ่านชิฟต์รีจิสเตอร์

รูปที่ 2.24 (ค) การเชื่อมต่อสวิตช์ผ่านชิฟต์รีจิสเตอร์จะเห็นได้ว่าวงจรนี้ต้องการขาพอร์ตน้อยมาเพียง 3 ขาพอร์ต โดยใช้ขาพอร์ตหนึ่งกำหนดโหมดที่ขา SHIFT/LOAD ให้อยู่ในสภาวะโหลดเพื่ออ่านสถานสวิตช์ทุกตัวเข้าสู่ชิฟรีจิสเตอร์ขั้นต่อไป คือ เปลี่ยนโหมดการทำงานให้อยู่ใน

สภาวะชิฟต์ข้อมูลและให้กำเนิดพัลส์จำนวน 8 พัลส์ เพื่อทำการชิฟต์ข้อมูลสถานะของสวิตช์ผ่านขาพอร์ต P_{12} ไปใช้งาน นั่นคือใช้ขาพอร์ตเพียง 3 ขาพอร์ตเท่านั้นก็ทำงานได้แล้วรวมทั้งการเขียนโปรแกรมควบคุมก็ทำได้ง่าย ถ้าต้องการเพิ่มจำนวนคีย์มากขึ้น ทำได้โดยเพิ่มชิฟต์รีจิสเตอร์มากขึ้นให้เท่าเทียมกัน อย่างไรก็ตาม ข้อเสียที่เกิดขึ้นคือใช้เวลาในการอ่านสถานะของคีย์ทั้งหมดเป็นเวลานานจนกระทั่งชิฟต์ข้อมูลได้ครบตามจำนวนคีย์

2.9.4 แบบเชื่อมต่อสวิตช์มัลติเพล็กซ์

วงจรการเชื่อมต่อแบบนี้แสดงในรูปที่ 2.24 (จ) ต้องการขาพอร์ตจำนวน 4 ขาพอร์ต พอร์ต P1.0 , P1.1 และ P1.2 ทำหน้าที่ควบคุมไอซีมัลติเพล็กซ์เซอร์เพื่อเลือกสวิตช์ S_1- S_6 ที่ต้องการจะติดต่อด้วยสถานะของคีย์ที่ถูกเลือกจะถูกส่งกับไปยังไมโครคอนโทรลเลอร์ผ่านขาพอร์ต P1.3 วงจรแบบนี้สามารถเพิ่มจำนวนคีย์ได้ง่ายและได้เป็นจำนวนมาก อีกทั้งการเขียนโปรแกรมก็ทำได้ง่ายเช่นเดียวกัน

2.9.5 แบบเชื่อมต่อสวิตช์เข้ากับระบบบัสข้อมูล

วงจรการเชื่อมต่อแบบนี้แสดงในรูปที่ 2.24 (จ) ทำการเชื่อมต่อสวิตช์เข้ากับระบบบัสข้อมูลเพื่อทำการอ่านสถานะของสวิตช์เข้าสู่ระบบบัสข้อมูลของไมโครคอนโทรลเลอร์ การเชื่อมต่อแบบนี้มักถูกนำมาใช้เมื่อไม่มีขาพอร์ตเหลือไว้ใช้งานเลย การติดต่อกับคีย์บอร์ด ทำได้โดยการอ้างตำแหน่งแอดเดรสไปยังวงจรดีโค้ดเดอร์ และ ทำการแอกตีฟขาสัญญาณ RD เพื่อส่งสัญญาณไปอีนาเบิลไปยังไอซีบัฟเฟอร์ส่งผ่านสถานะของไอซีต่างๆเข้าสู่ระบบบัสข้อมูลถึงแม้ว่าถ้าพิจารณาดูที่การเขียนโปรแกรมจะเห็นได้ว่า ไม่ยากเลยแต่ในทางปฏิบัติการต่อวงจรแบบนี้จะทำให้เกิดความยุ่งยากในทางฮาร์ดแวร์มาก เพราะต้องการเชื่อมต่อวงจรเข้ากับทั้งระบบบัสแอดเดรส และบัสข้อมูล

2.10 การติดต่อกับ 8255

การเพิ่มอุปกรณ์อินพุต/เอาต์พุตของ 8051 นั้น สามารถใช้ไอซีเบอร์ 8255 ซึ่งเป็นทั้งอุปกรณ์อินพุตเอาต์พุตภายในตัวที่สามารถโปรแกรมด้วยซอฟต์แวร์ได้ ทำให้สามารถใช้งานได้ดียิ่งขึ้น

2.10.1 ลักษณะพื้นฐานของ 8255

8255 จะใช้สำหรับ การส่งหรือรับข้อมูลแบบขนาน ระหว่างไมโครคอนโทรลเลอร์ กับอุปกรณ์ภายนอก ซึ่งสามารถเปลี่ยนแปลงการทำงานของพอร์ตอินพุตให้เป็น พอร์ตเอาต์พุตได้ โดยการส่งข้อมูลควบคุมจากไมโครคอนโทรลเลอร์ไปให้ 8255 ก่อนที่จะเริ่มต้นทำงาน

จากรูปที่ 2.25 เป็นผังการทำงานภายในของ 8255 ซึ่งสามารถแบ่งการทำงานได้ดังนี้ คือบล็อกจำนวน 4 บล็อกที่อยู่ทางด้านขวามือของรูป เป็นส่วนที่จะเชื่อมต่อกับอุปกรณ์ภายนอก โดยจะใช้สาย PA0-PA7 , PB0-PB7 และ PC0-PC7 เป็นทางผ่านของข้อมูลระหว่างอุปกรณ์ภายนอกกับ 8255 สายสัญญาณเหล่านี้จะถูกแบ่งออกเป็นอินพุต/เอาต์พุตพอร์ต 3 กลุ่ม คือพอร์ต A(PA) , พอร์ต B (PB) และ พอร์ต C (PC)

สำหรับบล็อกกลุ่มถัดมา ซึ่งอยู่บริเวณกลาง ๆ คือ Group A Control และ Group B Control ซึ่งจะทำหน้าที่กำหนดการทำงานของอินพุต/เอาต์พุตทั้ง 3 พอร์ตจากรูปที่ 2.25 เราจะสังเกตเห็นว่า พอร์ต C จะประกอบด้วยพอร์ตขนาด 4 บิต จำนวน 2 พอร์ต กลุ่มหนึ่งจะถูกควบคุมโดย Group A Control และอีกกลุ่มจะถูกควบคุมโดย Group B Control เพื่อประโยชน์ในการทำงาน

ส่วนบล็อกกลุ่มสุดท้าย คือ Data Bus Buffer และ Read/write Control Logic จะเป็นส่วนที่ติดต่อระหว่างไมโครคอนโทรลเลอร์กับ 8255โดย Data Bus Buffer จะเป็นบัฟเฟอร์ให้กับบัสข้อมูลของไมโครคอนโทรลเลอร์ส่วน Read / Write Control Logic จะทำหน้าที่ควบคุมให้ข้อมูลเข้าหรือออกจากรีจิสเตอร์ภายในตัวของ 8255 ได้อย่างถูกต้อง

2.10.2 รายละเอียดการจัดขาของ 8255

รูปที่ 2.26 ขาสัญญาณต่าง ๆ ของไอซีเบอร์ 8255จากรูปที่ 2.26 รายละเอียดขาของ 8255 มีดังนี้ คือ

ขาสัญญาณ	**รายละเอียด**
DO-D7	ข้อมูลอินพุต/เอาต์พุตแบบสองทิศทาง (Bi-directional Bus) ซึ่งจะเป็นทางผ่านของข้อมูลต่างๆ ของ 8255 กับไมโครคอนโทรลเลอร์
CS/ (Chip Select Input)	เป็นขาเลือกอุปกรณ์ให้ทำงานเมื่อขานี้มีลอจิกเป็น 0 จะสามารถอ่านหรือเขียนข้อมูลกับ 8255 ได้
RD/ (Read Input)	สัญญาณบอกสถานะ การอ่านข้อมูลจาก 8255 โดยจะทำงานเมื่อลอจิกเป็น 0
WR/ (Write Input)	สัญญาณบอกสถานะ การเขียนข้อมูลจาก บัสข้อมูลจากไมโครคอนโทรลเลอร์ ไปยัง 8255 โดยจะทำงานเมื่อลอจิกเป็น 0
A0-A1 (Address Input)	สัญญาณระบุตำแหน่งรีจิสเตอร์ภายในของ 8255
RESET	สัญญาณรีเซ็ตการทำงานภายในของ 8255 เพื่อให้เริ่มต้น ทำงานใหม่ โดยจะทำงานเมื่อมีสถานะเป็น 1

PA0-PA7, PB0-PB7 กลุ่มของสัญญาณอินพุต/เอาต์พุตขนาด 8 บิต ใช้ต่อเข้ากับอุปกรณ์ภาย นอก

PC0-PC7 กลุ่มของสัญญาณอินพุต/เอาต์พุทขนาด 8 บิต ใช้ต่อกับอุปกรณ์ภายนอกและ PC0-PC7 นี้ยังแบ่งออกเป็น 2 กลุ่ม โดยแต่ละกลุ่มจะมีขนาด 4 บิต คือ กลุ่มแรกจะใช้ควบคุม PB0-PB7 และกลุ่มที่ 2 ใช้ควบคุม PA0- PA7

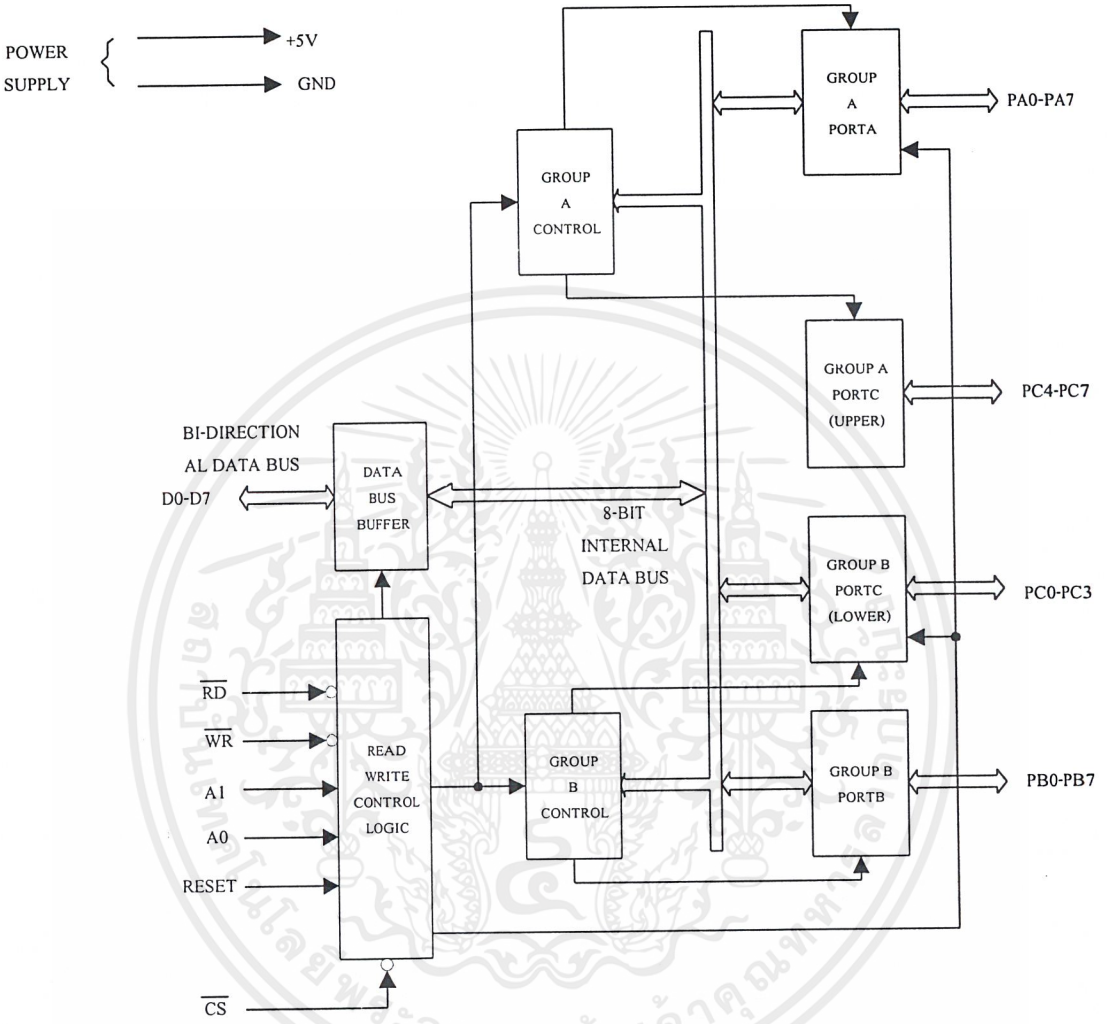
2.10.3 การเชื่อมต่อไมโครคอนโทรลเลอร์ 8051 กับ 8255

8255 มีสัญญาณแอดเดรสจำนวน 2 เส้น คือ A0 และ A1 ทำให้สามารถเลือกแอดเดรสได้ $2^2 = 4$ แอดเดรส ซึ่งแต่ละแอดเดรสก็คือ รีจิสเตอร์หรือพอร์ตภายใน 8255 ดังแสดงในตารางที่ 2.9

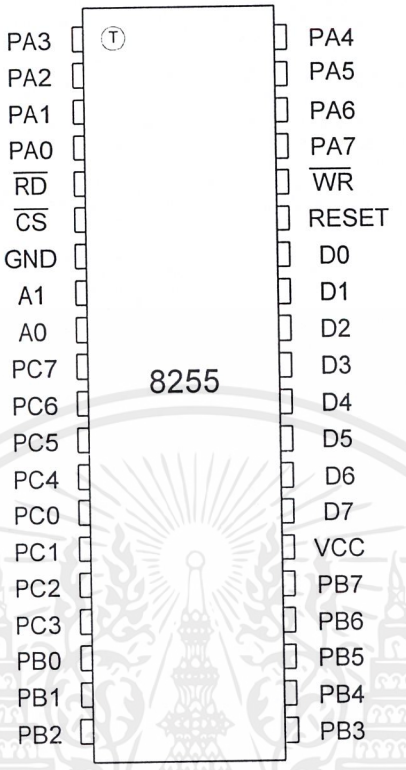
ตารางที่ 2.9 แอดเดรสของพอร์ตและรีจิสเตอร์ของ 8255

A1	A0	รีจิสเตอร์หรือพอร์ต
0	0	พอร์ต A
0	1	พอร์ต B
1	0	พอร์ต C
1	1	รีจิสเตอร์ควบคุม

เมื่อทำการรวมสัญญาณ $\overline{RD}$ และ $\overline{WR}$ กับขา A0 และ A1 แล้วนั่นคือ จะเป็นการอ่านหรือเขียนข้อมูลทางขา D0-D7 กับรีจิสเตอร์หรือพอร์ตของ 8255 ดังแสดงในตารางที่ 2.10



รูปที่ 2.25 ผังการทำงานภายในไอซีเบอร์ 8255



รูปที่ 2.26 ขาสัญญาณต่าง ๆ ของไอซีเบอร์ 8255

ตารางที่ 2.10 การทำงานของ 8255 เมื่อสัญญาณ $\overline{RD}$ และ $\overline{WR}$ เป็นค่าต่าง ๆ

$\overline{RD}$	$\overline{WR}$	A1	A0	ความหมาย
0	1	0	0	เขียนข้อมูลไปยังพอร์ต A
1	0	0	0	อ่านข้อมูลจากพอร์ต A
0	1	0	1	เขียนข้อมูลไปยังพอร์ต B
1	0	0	1	อ่านข้อมูลจากพอร์ต B
0	1	1	0	เขียนข้อมูลไปยังพอร์ต C
1	0	1	0	อ่านข้อมูลจากพอร์ต C
0	1	1	1	เขียนข้อมูลไปยังรีจิสเตอร์ควบคุม
1	0	1	1	สภาวะไม่ถูกต้อง (Illegal Read Register)

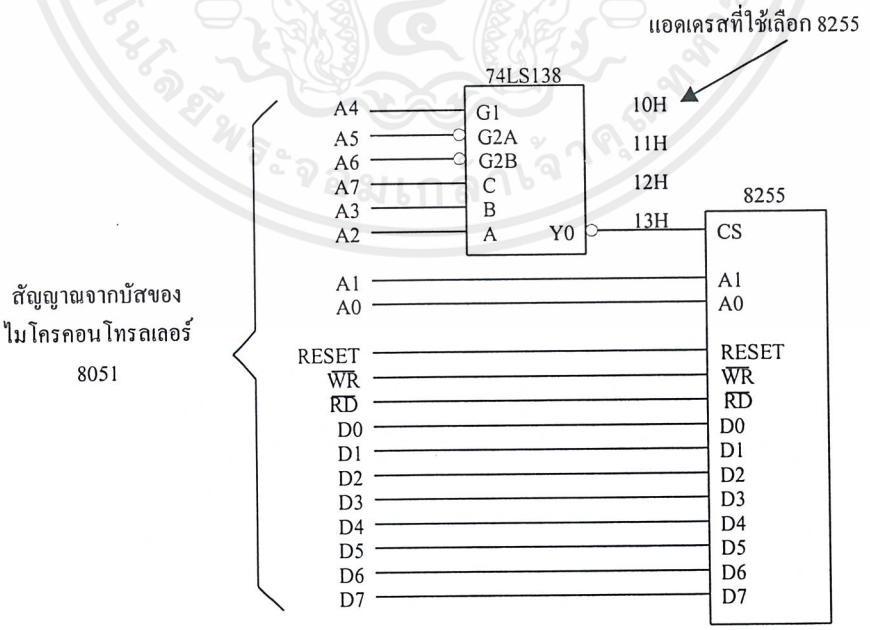
โดยทั่วไปมักจะกำหนดให้แอดเดรสทั้ง 4 ของ 8255 อยู่ที่ในช่วงใดช่วงหนึ่งของระบบ เช่น 10H, 11H, 12H และ 13H และ ใช้ขาแอดเดรสที่เกิน จาก A0 และ A1 มาทำการถอดรหัส

แอดเดรส เพื่อสร้างสัญญาณเลือกอุปกรณ์ $\overline{CS}$ ในช่วงแอดเดรสที่ต้องการ ดังแสดงในรูปที่ 2.27 จะเห็นว่า สัญญาณ $\overline{CS}$ จะเป็นลอจิก 0 เมื่อบัสแอดเดรส A2-A7 มีค่าเป็น 0000100XX (XX คือ A1 และ A0) ดังนั้นวงจรนี้แอดเดรสของรีจิสเตอร์ภายใน 8255 จะมีค่าตามตารางที่ 2.12

ส่วนขาควบคุมอื่น ๆ คือ $\overline{RD}$ และ $\overline{WR}$ ของ 8255 จะเชื่อมต่อกับขาสัญญาณ $\overline{RD}$ และ $\overline{WR}$ ของไมโครคอนโทรลเลอร์โดยตรง จะทำให้แอดเดรสพอร์ตของ 8255 อยู่ในพื้นที่ของหน่วยความจำสำหรับข้อมูลภายนอกของ 8051 ส่วนขา Reset จะทำงานที่ลอจิก 1 ดังนั้นจะต่อกับขา Reset ของ 8051 ได้โดยตรง ส่วนขา D0-D7 สามารถเชื่อมต่อโดยตรงกับบัสของ 8051 เช่นกัน ดังแสดงในรูปที่ 2.27 เช่นกัน

ตารางที่ 2.11 แอดเดรสของรีจิสเตอร์ภายใน 8255 จากรูปที่ 2.26

แอดเดรส	ความหมาย
10H	พอร์ต A
11H	พอร์ต B
12H	พอร์ต C
13H	รีจิสเตอร์ควบคุม



รูปที่ 2.27 การเชื่อมต่อระหว่าง 8255 กับไมโครคอนโทรลเลอร์ 8051

2.11 โครงสร้างภายในของ MCS – 51

ก่อนที่จะนำไมโครคอนโทรลเลอร์เบอร์ใดไปใช้งานก็แล้วแต่ ผู้ออกแบบต้องมีความรู้เกี่ยวกับโครงสร้างภายในของไมโครคอนโทรลเลอร์เบอร์นั้น ๆ ด้วย ไมโครคอนโทรลเลอร์ในตระกูล MCS – 51 นั้นมีอยู่หลายเบอร์ให้เลือกซึ่งแต่ละเบอร์จะมีความสามารถพิเศษมากน้อยแตกต่างกันไปดังแสดงในตารางที่ 2.12 ซึ่งเป็นเบอร์พื้นฐานในตระกูลนี้

คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล MCS – 51 มีดังนี้

1. เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
2. มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
3. มีขาสัญญาณอินพุทเอาท์พุทจำนวน 32 บิต
4. สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (Extemal Data Memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K
5. สามารถเชื่อมต่อโปรแกรมหน่วยความจำภายนอก (Extemal Program Memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K
6. มีหน่วยความจำโปรแกรมภายในตัว (On-Chip Pro-gram Memory) ขนาด 4 K โดยเฉพาะเบอร์ 8052 จะมีส่วนความจำในหน่วยนี้ถึง 8 K สำหรับเบอร์ 8031 และ M8032 จะมีหน่วยความจำในส่วนนี้

ตารางที่ 2.12 แสดงคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS – 51

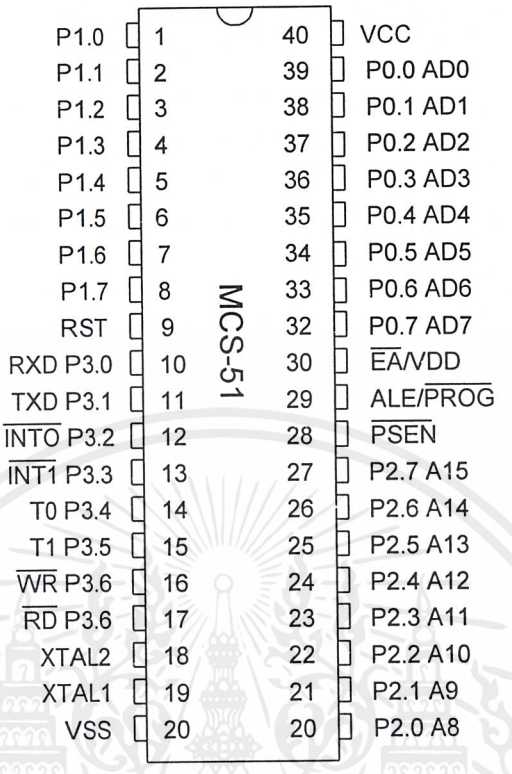
ชื่อเบอร์	หน่วยความจำภายใน		จำนวนไทเมอร์/เคาน์เตอร์	จำนวนอินเตอร์รัปต์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16 Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16 Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16 Bit	5
8032AH	ไม่มี	256 x 8 RAM	3 x 16 Bit	6
8031AH	ไม่มี	128 x 8 RAM	2 x 16 Bit	5
8031	ไม่มี	128 x 8 RAM	2 x 16 Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16 Bit	5
8751H-12	4K x 8 EPROM	128 x 8 RAM	2 x 16 Bit	5

7. มีหน่วยความจำข้อมูลภายในตัว (On-Chip Data Memory) ขนาด 128 ไบต์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์
8. หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น
9. มีไทเมอร์/เคาน์เตอร์ (Time/Counters) ขนาด 16 บิต จำนวน 12 ตัว โดยเฉพาะเบอร์ 8032 หรือ 8052 จะมีไทเมอร์/เคาน์เตอร์ จำนวน 3 ตัว
10. การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิด โดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้ 2 ระดับ
11. มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบ Full Duplex
12. มีคำสั่งในการคำนวณทางคณิตศาสตร์และตรรกศาสตร์
13. คำสั่งโดยส่วนใหญ่ใช้เวลาการทำงานเพียง 1 ไมโครวินาที เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์
14. ต้องการแหล่งจ่ายไฟ 5 โวลต์ เพียงชุดเดียว

2.12 โครงสร้างภายนอกของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS – 51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.28 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้

1. ขา V_{CC} เป็นขาป้อนแรงดันไฟเลี้ยง +5 V.
2. ขา V_{SS} เป็นขากราวด์
3. ขาพอร์ต 0 (Port 0) มี 8 ขา ได้แก่ขา $P_{0.0}$- $P_{0.7}$ เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้ งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำงานเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนด ให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็น พอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอก โดยทำหน้า ที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ (A0-A7) ซึ่งจะใช้งานเป็นแบบมัลติเพลกซ์ กับการรับข้อมูลขนาด 8 บิต
4. ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ P1.0-P1.7 เป็นขาพอร์ตอินพุตเอาต์พุต 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้พอร์ตต้อง ทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต P1.0 และ P1.1 จะถูกนำมาใช้งานเป็น ขาT2



รูปที่ 2.28 แสดงการจัดตำแหน่งขาต่าง ๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51

และ T2EX ตามลำดับด้วย

5. ขาพอร์ต 2 (Port 2) มี 8 ขา ได้แก่ขา P2.0-P2.7 เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้ งานทั่วไปโดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละ อินพุต พอร์ต เพื่อกำหนดให้ เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุต แล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ ในการกำหนด ตำแหน่งแอดเดรสไบต์สูง (A8-A15)
6. ขาพอร์ต 3 (Port 3) มี 8 ขา ได้แก่ขา P3.0-P3.7 เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งาน ทั่วไปโดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิต ของ พอร์ต เพื่อกำหนดให้เป็น พอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้ว มันยังถูกใช้งานในหน้าที่พิเศษต่าง ๆ ดังแสดงในตารางที่ 2.13
7. ขารีเซต (RST) ใช้สำหรับรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคง สถานะเป็น 1 อย่างน้อย 2 แมชีนไซเกิล ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่
8. ขา ALE / $\overline{\text{PROG}}$ เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตช์ ค่าตำแหน่งแอดเดรสไบต์ ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้

ยังทำ หน้าที่ เป็นอินพุตรับพัลส์ ในการโปรแกรม (Program Pulse Input) ในส่วนของ หน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS- 51 ที่มีหน่วยความจำโปรแกรมภายในเป็น EPROM

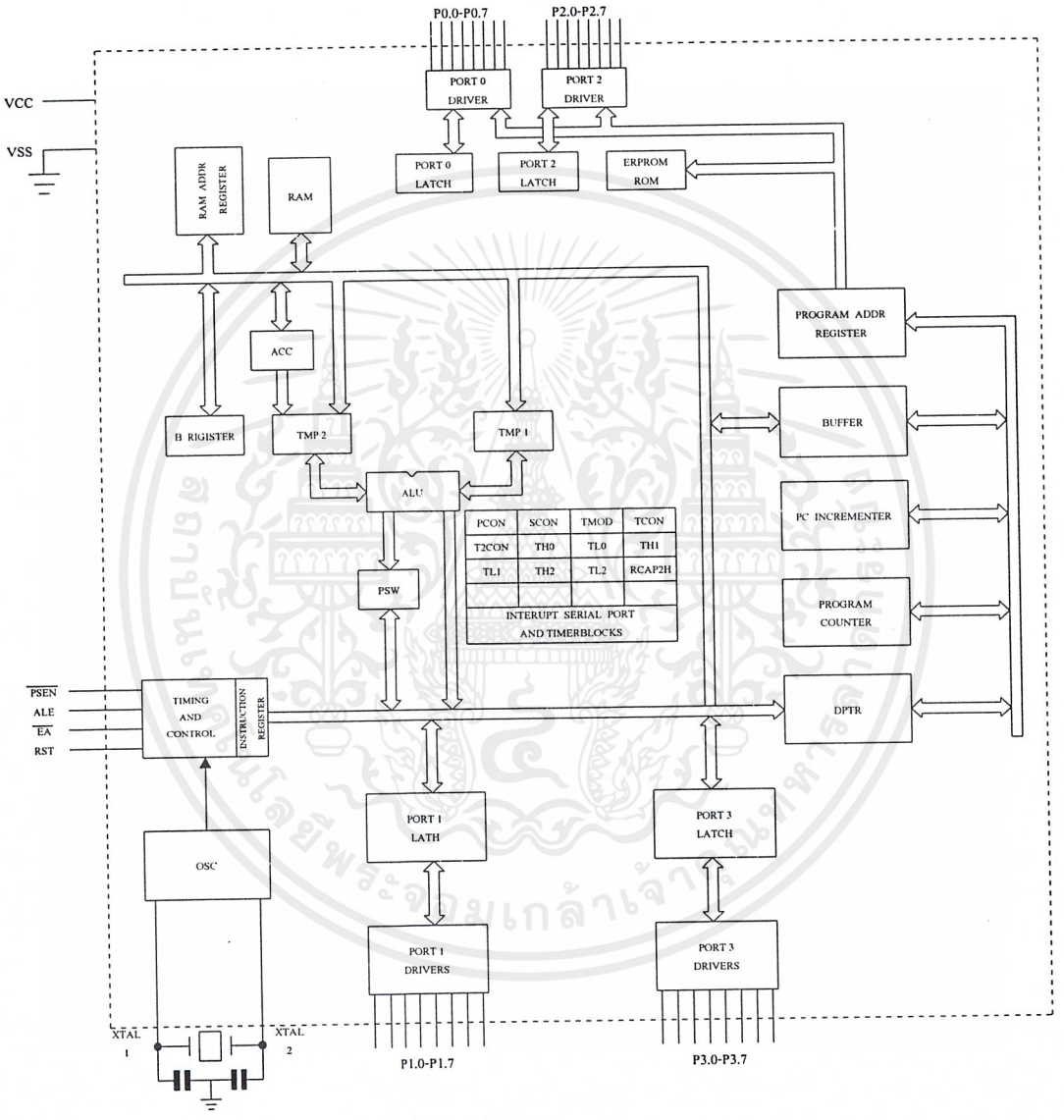
ตารางที่ 2.13 แสดงหน้าที่พิเศษของแต่ละขาพอร์ต P3

ขาพอร์ต	หน้าที่พิเศษ
P3.0	RXD (Serial Input Port)
P3.1	TXD (Serial Output Port)
P3.2	$\overline{\text{INT0}}$ (External Interrupt 0)
P3.3	$\overline{\text{INT1}}$ (External Interrupt 1)
P3.4	TO (Timer 0 External Input)
P3.5	T1 (Timer 1 External Input)
P3.6	$\overline{\text{WR}}$ (External Data Memory Write Strobe)
P3.7	$\overline{\text{RD}}$ (External Data Memory Read Strobe)

9. ขา $\overline{\text{PSEN}}$ (Program Store Enable) ทำหน้าที่ เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอกขานี้จะส่งสัญญาณ สโตรบจำนวน 2 ครั้ง ในแต่ละแมชีนไซเกิล แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด
10. ขา $\overline{\text{EA}}$ / VPP (External Access enable/VPP) เป็นขาสำหรับเลือกใช้งานหน่วยความจำโปรแกรมจากภายในหรือภายนอก โดยถ้ามีสถานะเป็น 0 จะหมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วย ความจำภายนอกที่ตำแหน่งแอดเดรส 0-0FFFH (0-1FFFH ถ้าเป็นเบอร์ 8052) อย่างไรก็ตามถ้าบิตป้องกัน (security bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ไมโครคอนโทรลเลอร์จะ ยังไม่รับคำสั่งจาก หน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (Vpp) ขนาด 21 V. เพื่อใช้ในระหว่างการโปรแกรม EPROM
11. ขา XTAL1 และขา XTAL2 เป็นขาอินพุตและเอาต์พุตของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปริไฟเออร์ (Inverting Oscillator Amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.13 โครงสร้างภายในของ MCS-51

โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังรูปที่ 2.29 โดยส่วนที่มีเครื่องหมายดอกจัน (*) จะมีเฉพาะในเบอร์ 8032 และ 8052 เท่านั้น



รูปที่ 2.29 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

2.14 ANT - 31PJ Version 2.0

การเรียนรู้ที่ดีย่อมเกิดจากการได้สัมผัสงานด้วยตัวเอง เพราะฉะนั้นในปัจจุบันนักศึกษาที่เรียนรู้ทางอิเล็กทรอนิส์หรือทางอุตสาหกรรมต่างๆ มักจะต้องทำ PROJECT ควบคู่ไปกับการเรียน

เสมอ หนึ่งในบรรดาสิ่งต่างๆ ที่มักจะถูกใช้ในการพัฒนางาน PROJECT ก็คือ ไมโครคอนโทลเลอร์ ในตระกูล MCS - 51 ในจำนวนบอร์ดไมโครคอนโทลเลอร์ ที่เหมาะกับการเรียนรู้ ANY - 31PJ เป็นบอร์ดที่เน้นสำหรับงานพัฒนาโดยเฉพาะ ซึ่งบอร์ด ANT - 31PJ VERSION 1.0 การพัฒนาการเรียนรู้ยังอยู่อยู่ในรูปแบบที่จำกัด คือ การพัฒนาการทางด้านโปรแกรมยังทำได้ไม่มากเท่าที่ควร ทางบริษัทจึงได้ทำการเพิ่มส่วนของ Option ซึ่งเข้ามาช่วยในการพัฒนาความเข้าใจตลอดจนความสะดวกแก่ผู้ใช้ที่จะทำการศึกษา

ANT - 31PJ VERSION 2.0จะมีความแตกต่างกับ VERSION 1.0 ในหลายส่วนดังนี้ ใช้ชิปเบอร์ต่างๆ ในตระกูล MCS - 51ได้หลากเบอร์กว่า คือ เบอร์ AT89C51, AT89T52 (ของบริษัท ATMEL) , 87C51FA หรือ แม้แต่ DA5000T เป็นไมโครคอลเลอร์ ของบริษัท DALLAS ซึ่งสามารถเขียนโปรแกรมลงภายในตัวได้ถึง 32 KBYTE โดยภายในตัวจะมีชุดของ NV SRAM อยู่ ทั้งยังมีส่วนประกอบของโปรแกรมมอนิเตอร์ ไว้ใช้ติดต่อกับคอมพิวเตอร์ในการโหลดโปรแกรม ที่ทำการพัฒนาได้อย่างไม่จำกัดจำนวนครั้ง บนบอร์ดยังสามารถใช้หน่วยความจำภายนอก ได้ 2 ส่วน โดยทำให้สามารถใช้งานหน่วยความจำได้สูงสุดถึง 62 KBYTE (ส่วนของ Program Memory ที่ U3 จำนวน 32 kbyte และในส่วนของ Data Memory+ Program Memory ที่ U3 ซึ่งเป็น RAM จำนวน 30 kbyte โดยการ Decode จาก PAL16L8 ส่วน Address จำนวน 2 kbyte จะถูกนำไปใช้ในการ Dcode อุปกรณ์โดยรอบ) พร้อมทั้งส่วน 8255 Uaerport ที่มี I/O ในการใช้งานทั้งหมด 24 บิต ส่วน LCD Port จะต่อเข้ากับ LCD Module ได้โดยตรงแต่จะเปลี่ยนการอ้าง Address ไปเป็น FA00H - FBFFH และการเพิ่มการเชื่อมขา RD,WR ในการควบคุม Enable LCD อีกด้วย นอกจากนี้ยังมีส่วนที่เป็น Working Area ซึ่งได้ทำการเพิ่มขนาดให้มาขึ้นเป็น 4" x 2" พร้อมด้วยจุดต่อสัญญาณ ต่างๆ ที่สำคัญเพิ่มเติมขึ้นมาจากเดิมอีก ทำให้สะดวกมากขึ้น ในการต่ออุปกรณ์พัฒนาส่วนอื่นๆ เพิ่มเติมเข้าไปภายในยังได้ทำการเปลี่ยนอุปกรณ์ Decode จาก 74HCT138 มาเป็น IC PAL16L8 (SL) เข้ามาทำการ Decode อุปกรณ์แทน FCH - FDFFH (เป็น User) ตำแหน่ง FE00H - FFFFH (เป็น User2) ในการต่อพัฒนาเพิ่มเติมอีกด้วย ส่วนระบบ Reset และ Watch Dog จะยังคงข้อดีของ MAX1232 ในการตรวจสอบระดับแรงดันไฟ (โดยเมื่อระดับแรงดันต่ำกว่า 4.5V. หรือ 10% จากระดับ VCC MAX1232 จะทำการ Reret ระบบ)ซึ่งช่วยให้งานมีความแน่นอนยิ่งขึ้น ทางด้าน Port อนุกรมที่ใช้ทำการติดต่อกับ PC ได้ทำการเปลี่ยนมาใช้ DS275 ซึ่งเป็นตัวแปลงระดับแรงดันจาก PC ลงสู่บอร์ดแทนการใช้ MAX232 มีข้อเสียตรงที่ใช้อุปกรณ์จำพวกพาสซีฟ มากเกินไปทำให้เกิดความยุ่งยากแก่การใช้งานอุปกรณ์ในส่วนของการเลือกใช้ เช่น REM31 หรือ BASIC32 ซึ่งเป็นโปรแกรมที่ช่วยให้การพัฒนาบอร์ดได้โดยผ่านเครื่อง PC ก็ยังคงข้อดีไว้เช่นเดิม

ตารางที่ 2.14 คุณสมบัติของ บอร์ด ANT - 31PJ VERSION 2.0

CUP	80C31 (On Board), 87C51FA,89C52,DS5000T
Clock	11.0592 MHz
Memory	U2 0/32K Epromsocket (2764,27128,27256) Address 0000H - 7FFFH (PMEM 27256) U3 8/30K RAM PMEM + DMEM (6264,62256) Address 8000H - 80FFH (DMEM) ,8100H - F7FFH (PMEM)
Port	24 Bit 8255User port (26 PIN Connector) LCD Port (Dot Matrix Only) (20 PIN Connector) Serial RS232 (DS275 = Option) (3 PIN Connector) Serial RS485 (75176 = Option) (2 PIN Connector)
Reset	MAX1232 (Watchdog)
On Board	1. 2-PIN 5Vdc. Connector 2. Test Point For Clip (VCC,GND) 4" X 2" Working Area PCB
Jumper	Watchdog (EN,DIS) EA (INT,EXT) DS5OOT (R,P) ROM U2 (2764,27128,27256) RAM U3 (6264,62256)
Power	5Vdc Current 80 mA (With 27C256 EPROM) 5VDC 220 mA (Full Option)
Size	4"x5"

2.15 รายละเอียดต่างๆ บนบอร์ด

1) หน่วยความจำ

การจัดหน่วยความจำของบอร์ด ANT - 31PJ แบ่งได้เป็น 2 SOCKET โดยสามารถใส่หน่วยความจำได้สูงสุดถึง 64 KBYTE และสามารถเลือกแบบเบอร์ต่างๆ ได้ดังนี้

ตารางที่ 2.15 หน่วยความจำของบอร์ด ANT - 31PJ

Socket	ขนาดหน่วยความจำ	เบอร์หน่วยความจำที่ใช้	แนวทางการใช้งาน
U2	8-32K	2764,27128,27256	สำหรับโปรแกรมคอมพิวเตอร์
Socket	ขนาดหน่วยความจำ	เบอร์หน่วยความจำที่ใช้	แนวทางการใช้งาน
U3	8-32K	6264,62256	สำหรับเก็บข้อมูลหรือใช้ทดสอบโปรแกรมโดยสามารถเลือกตำแหน่ง ADDRESS ให้เป็น 0000H หรือ 8000H

2) การเลือก Address ของ U3 นี้ สามารถเลือกได้เป็น 2 ช่วงดังนี้

2.1) Address 0000H - 1FFFH โดยจะมองเห็นหน่วยความจำเป็น Datamemory เท่านั้นซึ่งสามารถใช้งานได้ทั่วไป หรือจะใช้กับ Basic 32 ก็ได้ (Basic 32 คือ EPROM ที่มี Basic อยู่ในตัว ทำให้สามารถพัฒนาโปรแกรมด้วยภาษา Basic ได้)

2.2) Address 8000H - 9FFFH โดยจะมองเห็นหน่วยความจำเป็นทั้ง Datamemory และ Progremmemort ซึ่งใช้งานได้ทั่วไป ขณะที่สามารถใช้กับ REM 31 ได้ด้วย โดยจะทำให้การพัฒนาโปรแกรมทำได้อย่างสะดวก (REM 31 คือ EPROM ที่มีโปรแกรมมอนิเตอร์ที่ช่วยในการพัฒนาโปรแกรมบอร์ด ผ่านทางเครื่อง PC ได้)

การเลือกเบอร์ของหน่วยความจำหรือการเลือกตำแหน่งของ Address สามารถทำได้ด้วยตัว Jumper ที่อยู่ทางด้านล่างของ Socket โดยกำหนดตัว Jumper ให้ตรงกับตำแหน่งที่ต้องการ

2.15.1 8255 Input/Output Port

บอร์ด ANT - 31PJ V2.0 มีหน่วย I/O อยู่ 1 ตัว โดยผู้ใช้สามารถใช้งานได้อย่างอิสระ ซึ่งใช้เบอร์ 8255 ยอดนิยม ขั้วต่อเป็นแบบ 26 PIN Header มาตราฐาน โดยสามารถใช้บอร์ดขยายต่างๆ

ในตระกูล EX - Series ได้ทันที หรือจะออกต่อไปประยุกต์ใช้งานต่างๆ ได้ตามต้องการ ตำแหน่ง Address ของ 8255 จะจัดอยู่ในส่วน Datamemory โดยใช้ Address ดังต่อไปนี้

ตารางที่ 2.16 ตำแหน่ง Address ของ 8255

Port	Address
Port A	F800H
Port B	F801H
Port C	F802H
Control	F803H

การใช้งาน 8255 นี้ จำเป็นจะต้องส่ง Control Code ไปยัง Control Port ก่อนเสมอ ซึ่งจะเป็นการกำหนดคุณสมบัติต่างๆ ของ Port A,B,C ทั้งนี้ให้ดูรายละเอียดจาก DATA SHEET ของ 8255 อีกที อย่างไรก็ตามการใช้งานในโหมด 0 ซึ่งเป็น โหมดที่ใช้กันเป็นส่วนใหญ่

2.15.2 LCD Port

บอร์ด ANT - 31PJ V2.0 จะมี LCD Port ให้พร้อม โดยสามารถต่อเข้ากับ LCD Module แบบ Dot Matrix ได้ ทันที ซึ่งใช้ขาสัญญาณทั้งหมด 14 ขา แต่ในบอร์ด ANT - 31PJ V2.0 ไม่ได้ออกแบบมาเผื่อ สำหรับให้ผู้ใช้ต้องการใช้กับ LCD แบบ Graphic ผู้ใช้ต้องการทำการต่อใช้งานเองภายหลัง การใช้งาน LCD Port นี้จะมีการจัดวงจรในแบบ Memory Map ซึ่งจะช่วยให้การเขียนโปรแกรมทำได้ง่าย โดยจะมองเห็นตำแหน่งต่างๆ เป็นในรูปแบบ Address ดังจะสรุปได้ดังนี้

ตารางที่ 2.17 Memory Map

Address	ลักษณะของพอร์ตที่ติดต่อ
FA00H	สำหรับเขียนคำสั่ง (RS=0 R/W=0)
FA01H	สำหรับอ่าน Busy (RS=0 R/W=1)
FA02H	สำหรับเขียนข้อมูล (RS=1 R/W=0)
FA03H	สำหรับอ่านข้อมูล (RS=1 R/W=1)

การใช้ LCD แบบ Dot Matrix นี้สามารถจะเลือกรุ่นใดๆ ก็ได้ ที่มีจำนวนตัวอักษร และจำนวนบรรทัดตามที่ต้องการ เพราะใช้สัญญาณแบบเดียวกันหมด จะแตกต่างในด้านโปรแกรมเท่า

นั้น การนับหมายเลขขั้วต่อของ LCD Port จะไม่เหมือนการนับโดยทั่วๆ ไป จึงควรดูให้แน่ใจก่อนต่อใช้งาน อีกประการหนึ่งหมายเลขขั้วต่อที่ด้านตัว LCD มนก็จะมีหลายแบบ คืออาจจะเป็นแถวคู่หรือแถวเดี่ยวก็ได้ ทั้งนี้หมายเลข 1-14 ของขาสัญญาณจะตรงกันหมด กล่าวคือต่อตามหมายเลขให้ตรงกันเป็นใช้ได้ รายละเอียดของการใช้งานตัว LCD นี้ ในอ่านเพิ่มเติมได้จากคู่มือของ LCD ที่ได้นำมาใช้อีกที

2.15.3 วงจร Reset และ Watch

วงจร Reset และบอร์ด ANT - 31PJ V2.0 จะใช้ชิปเบอร์ MAX 1232 โดยจะใช้สำหรับการ Reset ระบบ รวมทั้งมีวงจร Watchdogให้พร้อมด้วย ซึ่งการ Reset แบบนี้จะให้ผลดีกว่าวงจรแบบ RC ทั่วๆ ไป โดยจะทำการ Reset ทั้งช่วง Power Up และ Down จะแน่ใจได้ว่าระบบไมโครจะไม่เกิดสภาพรวนจากการ Reset อย่างแน่นอน การ Reset จาก MAX 1232 จะตั้งระดับไว้ที่ 10 % ของ VCC คือ ระบบ Reset จะทำงานเมื่อ VCC ต่ำกว่า 4.5 V

ส่วนวงจร Watchdog ก็ยิ่งช่วยให้ระบบเกิดเสถียรภาพมากขึ้น โดยสามารถเลือก Enable หรือ Disable ได้ด้วย Jumper ที่อยู่ด้านขวาของชิป PAL1618 (SLP04) ในกรณีที่ตั้งไว้ที่ Disable ระบบจะต่อ ขา/ST ของ MAX 1232 เข้ากับขา ALE ของตัวไมโคร ซึ่งหมายถึงว่าจะมีสัญญาณมาตลอดเวลา แต่ถ้าตั้งไว้ที่ Enable ระบบจะต่อเข้ากับ Port 1 ของ CPU ในตำแหน่งขา P1.7 (ขา 8 ของ CPU) เพราะฉะนั้นโปรแกรมที่เขียนขึ้นจะต้องมีการส่งสัญญาณมากระตุ้นที่ตำแหน่งนี้ เสมอภายใน 1.2 sec (การส่งสัญญาณมากระตุ้นทำได้ด้วยการใช้คำสั่งCPL P1.7 ก็เพียงพอแล้ว) ในกรณีที่สัญญาณขาดหายไป ซึ่งอาจจะเกิดจากการ Hang ของระบบ MAX 1232 ก็จะทำการ Reset ระบบทันทีโดยจะทำให้ระบบทั้งหมดกลับมาทำงานได้ตามปกติต่อไป

2.15.4 Serial Port

Serial Port ของบอร์ด ANT - 31PJ V2.0 ใช้ต่อตรงจากชิป 8031 โดยผ่าน DS275 ซึ่งจะแตกต่างจาก ANT - 31PJ V2.0 ซึ่งใช้ MAX 232 เพื่อทำหน้าที่เปลี่ยนระดับแรงไฟให้เข้ามาตราฐาน RS 232 โดย DS275 จะมีอุปกรณ์การต่อใช้งานร่วมที่น้อยกว่า MAX 232 และขั้วต่อที่ใช้เป็นแบบ 3 PIN โดยเป็นมาตราฐาน สามารถใช้กับความเร็วได้สูงสุดถึง 19200 BPS เช่นเดียวกับ MAX 232

ใน บอร์ด ANT - 31PJ V2.0 ได้เพิ่ม Serial Port RS485 เพื่อนำไปใช้งานพัฒนาที่มากขึ้นไปจากเดิมโดย RS485 ที่ได้เลือกนำมาเป็น Potion ที่ต้องต่อเพิ่มเติมเข้าไป สามารถใส่ได้ 2 เบอร์ คือ MAX 485 และ 75176 มาตราฐานการส่งและรับสามารถตั้งค่าความเร็วได้เช่นเดียวกับ RS232 เพียงแต่จะใช้สายสัญญาณน้อยกว่า คือ TX กับ RX ส่วนขา GND จะไม่นำมาใช้และทำการส่งได้ไกลขึ้น (ตามData Sheet ส่งได้ไกล 1.2 Km.) การควบคุมเป็นแบบ Half - Duplex ใช้ขา P3.4 ในการควบคุม

2.15.5 Working Area PCB

เป็นพื้นที่ PCB สำหรับให้ผู้ใช้พัฒนาวงจรต่างๆ เพิ่มเติมได้ตามต้องการ โดยมีขนาดมากถึง 4"x2" และที่สำคัญคือ มีขาสัญญาณต่างๆ ที่สำคัญของ MCS - 51 ให้พร้อม โดยอยู่รอบๆ บริเวณนี้ ซึ่งช่วยให้ผู้เดินสายเข้ากับวงจรได้ง่ายและสะดวกเป็นอย่างยิ่ง

2.15.6 แนวทางการพัฒนาโปรแกรม

การพัฒนาโปรแกรมบนบอร์ด ANT - 31PJ V2.0 สามารถทำได้ทั้งภาษา Basic และภาษา Assembly โดยจะมีแนวทางดังต่อไปนี้

1) การพัฒนาด้วยภาษา Assembly ผ่านเครื่องมือประเภท EPROM Emulator

วิธีนี้คือ การเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ PC ก่อน จากนั้นก็แปลเป็น HEX File และ นำ HEX File นี้ไปอัดลงตัว EPROM แล้วจึงนำมาเสียบทดลองกับบอร์ดแต่แทนที่จะใช้ตัว EPROM จริงๆ ก็ใช้เครื่องมือแบบ EPROM Emulator แทน ซึ่งจะช่วยให้สะดวกและรวดเร็วกว่า ไม่สิ้นเปลืองตัว EPROM และไม่ต้องทำการเสียบเข้าออกบ่อยๆ ด้วย (บอร์ด คือ EE - 232) การพัฒนาแบบนี้นับว่าเป็นขั้นพื้นฐานที่สุดโปรแกรมของผู้ใช้ก็จะเริ่มต้นที่ Address 0000H เลย ทำให้ผู้ใช้มีความชัดเจนในการทำโปรมแกรมได้เป็นอย่างดี รวมทั้งสามารถใช้ Hardware ทุกส่วนได้อย่างอิสระแต่จุดอ่อนก็มีบ้าง โดยเฉพาะกับผู้ที่เริ่มต้นเรียนใหม่ คือไม่มีตัวช่วยในการ Debug ตัวโปรแกรม ไม่สามารถตั้ง Break หรือทำการ Single Step ได้

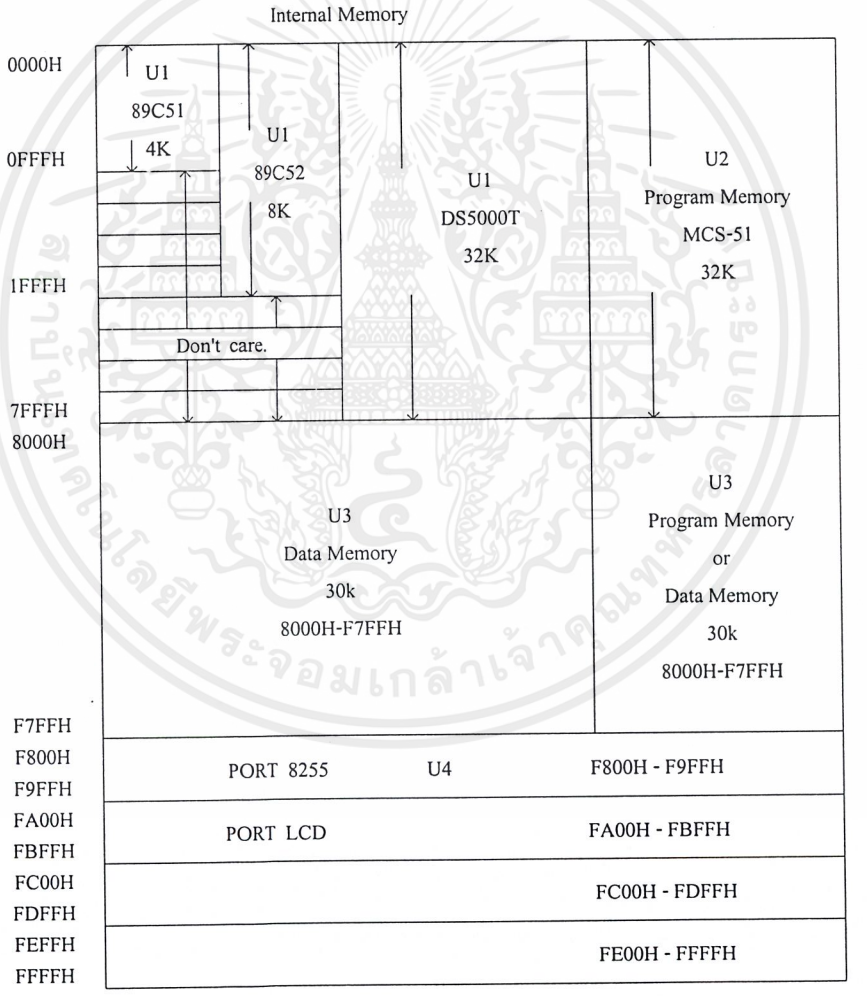
2) พัฒนาด้วยภาษา ASSEMBLY ผ่านโปรแกรม REM31

วิธีนี้จะใช้โปรแกรม REM31 เป็นตัวช่วยในการพัฒนา โดยโปรแกรม REM31 จะอยู่ในรูปของ EPROM ที่นำมาเสียบลงที่ตำแหน่ง U2 ซึ่ง REM31 นี้จะทำหน้าที่ติดต่อกับเครื่องคอมพิวเตอร์ PC ช่วยให้ผู้ใช้สามารถเข้าถึงบอร์ดได้ โดยอาศัยจอภาพและคีย์บอร์ดของเครื่อง PC สามารถใช้งานได้ทำนองเดียวกับคำสั่ง Debug.Com บน DOS คือ สามารถดูค่าในหน่วยความจำบนบอร์ด ANT - 31PJ ได้ดูค่า Register ได้สั่งให้โปรแกรมทำงานทีละคำสั่ง (Single Step) รวมทั้งค่า BREAKได้ และอื่นๆ อีกมากมาย วิธีนี้นับว่าเหมาะสมและคุ้มค่าอย่างมาก รวมทั้งมีราคาที่ไม่แพงด้วย จุดอ่อนคือ โปรแกรมของผู้ใช้จะเริ่มต้นที่ Address 8100H ซึ่งเมื่อถึงขั้นที่จะนำไปใช้งานต้องทำการเปลี่ยน Origin เป็นตำแหน่ง 0000H ก่อน และในการพัฒนานั้น ผู้ใช้จะต้องทำโปรแกรมได้ในขนาดที่ไม่เกิน 8 kbyte

3) การพัฒนาด้วยภาษา BASIC - 32

วิธีนี้ใช้โปรแกรม BASIC - 32 ที่อยู่ในรูปของ EPROM มาเสียบที่ U2 (ซึ่งวิธีนี้ต้องเปลี่ยน CPU เป็น 80C32 และเปลี่ยน RAM ที่ตำแหน่ง U3 เป็น 32 kbyte เสียก่อน) โดยใช้โปรแกรม Basic - 32 ก็ตัวแปลภาษา Basic ในแบบ Interpreterนั่นเองซึ่งเมื่อต่อเข้ากับคอมพิวเตอร์

PC แล้วผู้ใช้จะเขียนโปรแกรมเป็นภาษา Basic ได้ทันที ภาษา Basic ในที่นี้จะเหมือนกับ Basic - 32 ของ Intel ทุกประการ แต่จะเพิ่มเติมคำสั่งบางอย่างเข้าไป เพื่อให้ใช้งานได้สะดวกยิ่งขึ้น การพัฒนาโปรแกรมด้วยภาษา Basic จะทำได้ง่าย และสะดวกกว่าภาษา Assembly จึงต้องดูลักษณะงานที่จะนำไปใช้ด้วย ว่าจะย่อมรับได้หรือไม่ ขณะเดียวกัน บอร์ด ANT - 31PJ มีช่องเสียบหน่วยความจำเพียง 2 Socket เท่านั้น จึงไม่เพียงพอที่จะใช้ภาษา Basic ในแบบ Auto Start ได้ (คือ การกำหนดให้โปรแกรมทำงานทันทีเมื่อเปิดเครื่อง โดยจะต่อกับเครื่อง PC หรือไม่ก็ตาม) การใช้งานจะต้องทำการโหลดโปรแกรมทุกครั้ง ทั้งนี้เนื่องจากภาษานี้จะต้องเสียบตัว EPROM (Basic - 32) ไว้ที่ U2 เสมอ และต้องใช้เป็น RAM เป็น Working Area ที่ U3 เสมอด้วยเช่นกัน

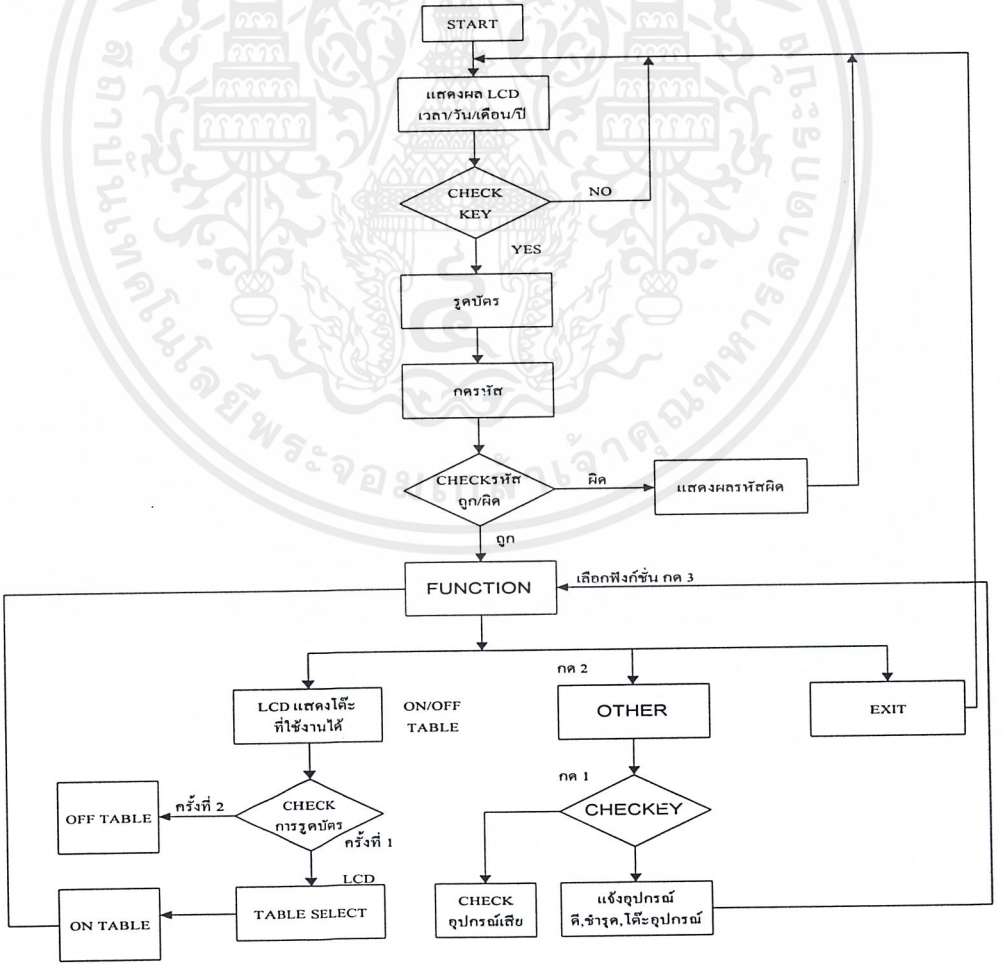


รูปที่ 2.30 Memory Map และ I/O Map

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

3.1 กล่าวนำ

ในการออกแบบชุดควบคุมการจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงาน จะประกอบด้วย 3 ส่วนคือ ส่วนประมวลผล และส่วนของวงจรซึ่งส่วนประมวลผลได้แก่ ไมโครคอนโทรลเลอร์ ส่วนของวงจรได้แก่ ส่วนวงจรรับข้อมูลจากบัตรบาร์โค้ด วงจรรับข้อมูลจากคีย์เมทริกซ์สวิตช์ , วงจรควบคุมการจ่ายไฟให้กับห้องปฏิบัติการโครงงาน และส่วนสุดท้ายคือส่วนของวงจรสื่อสารข้อมูล โดยสามารถแสดง โครงสร้างการทำงานของชุด ควบคุมการจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงาน ในรูปที่ 3.1 ซึ่งมีการทำงานดังนี้



รูปที่ 3.1 โครงสร้างการทำงานของชุดควบคุมการจ่ายไฟอัตโนมัติสำหรับห้องปฏิบัติการโครงงาน

ส่วนของวงจรรับข้อมูลจากเครื่องอ่านบาร์โค้ด และส่วนของวงจรรับข้อมูลจากคีย์บอร์ดเมทริกซ์ จะต่ออยู่กับไมโครคอนโทรลเลอร์ โดยมีไมโครคอนโทรลเลอร์เป็นศูนย์กลาง โดยส่วนรับข้อมูลอ่านบัตรนักศึกษาเข้ามาและส่งข้อมูลไปให้ไอซี 89C51 เมื่อผู้ที่ต้องการใช้เครื่องมือ ก็จะทำการรูดบัตรนักศึกษาที่มีรหัสบาร์โค้ดผ่านตัวบาร์โค้ดและส่วนรับข้อมูลจากคีย์เมทริกซ์สวิตช์จะรับรหัสข้อมูลประจำตัวเป็นเลขจำนวน 8 ตัว จากนั้น ไอซี 89C51 จะรับรหัสข้อมูลจากเครื่องอ่านบาร์โค้ดและจากปุ่มกดรหัส เพื่อทำการประมวลผลรหัสข้อมูลที่รับเข้ามา แล้วส่งสัญญาณไปแสดงผลที่ LCD แสดงรหัสประจำตัวนักศึกษา และข้อมูลที่ได้จากการกดเมทริกซ์สวิตช์เพื่อนำมาประมวลผลในการใช้งานฟังก์ชั่นต่าง ๆ และทำการเก็บข้อมูลที่ได้จากบาร์โค้ดคีย์เมทริกซ์สวิตช์และนาฬิกา นำไปเก็บไว้ในหน่วยความจำข้อมูลภายใน (Ram) เพื่อรอส่งข้อมูลไปที่ไมโครคอมพิวเตอร์ผ่านพอร์ตสื่อสารแบบอนุกรม RS-485 เพื่อใช้ในการตรวจสอบจำนวนผู้ใช้วัน เวลา ที่ใช้และอุปกรณ์ที่ชำรุดเสียหายภายในห้องปฏิบัติการโครงงาน

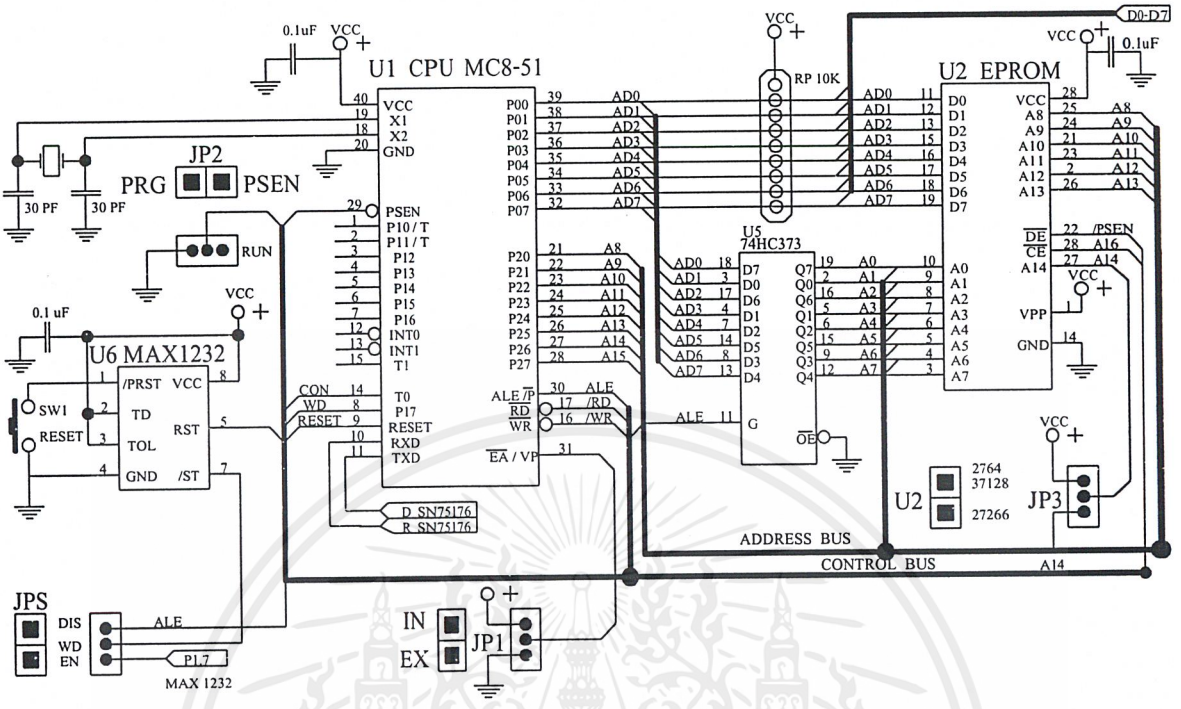
ซึ่งในบทนี้จะกล่าวถึงการออกแบบ , การสร้าง และการทำงานของระบบของส่วนวงจร

3.2 การออกแบบวงจร

การออกแบบชุดควบคุมการจ่ายไฟอัตโนมัติ สำหรับห้องปฏิบัติการโครงงาน จะแยกเป็นสามส่วนคือส่วนวงจรควบคุม , ส่วนของวงจรรับข้อมูล , ส่วนของการแสดงผล , ส่วนวงจรควบคุมการจ่ายไฟ,และส่วนของวงจรสื่อสารข้อมูลซึ่งมีการทำงานดังนี้

3.2.1 ส่วนของวงจรควบคุม

ส่วนของวงจรควบคุมจะทำหน้าที่ควบคุมการรับข้อมูลการแสดงผลรวมทั้งการประมวลผลเพื่อทำการรับข้อมูล และส่งข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ซึ่งในวงจรนี้ได้ใช้ไอซีไมโครคอนโทรลเลอร์เบอร์ 80C31 และ 89C51 เป็นตัวประมวลผลข้อมูลที่รับเข้ามาทั้งทางเครื่องอ่านเบอร์โค้ด และคีย์เมทริกซ์สวิตช์เมื่อทำการประมวลผลข้อมูลแล้วก็จะนำข้อมูลแสดงผลออกทางอุปกรณ์แสดงผล LCD หลังจากนั้นก็จะทำการประมวลผลข้อมูลที่รับเข้ามากับข้อมูลที่มีอยู่เพื่อจะทำการใช้งานตามความต้องการ



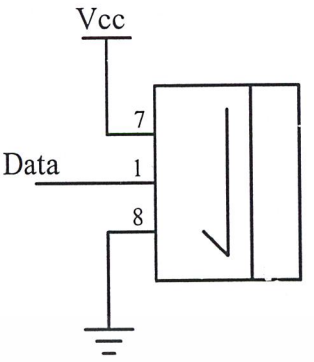
รูปที่ 3.2 วงจรควบคุม

3.2.2 ส่วนของวงจรรับข้อมูล

ในส่วนของวงจรรับข้อมูลทำหน้าที่ในการรับข้อมูลเพื่อทำการส่งข้อมูลให้แก่ไมโครคอนโทรลเลอร์ เพื่อนำไปประมวลผล ซึ่งสามารถแบ่งได้เป็นสองส่วนคือ

1. ส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด

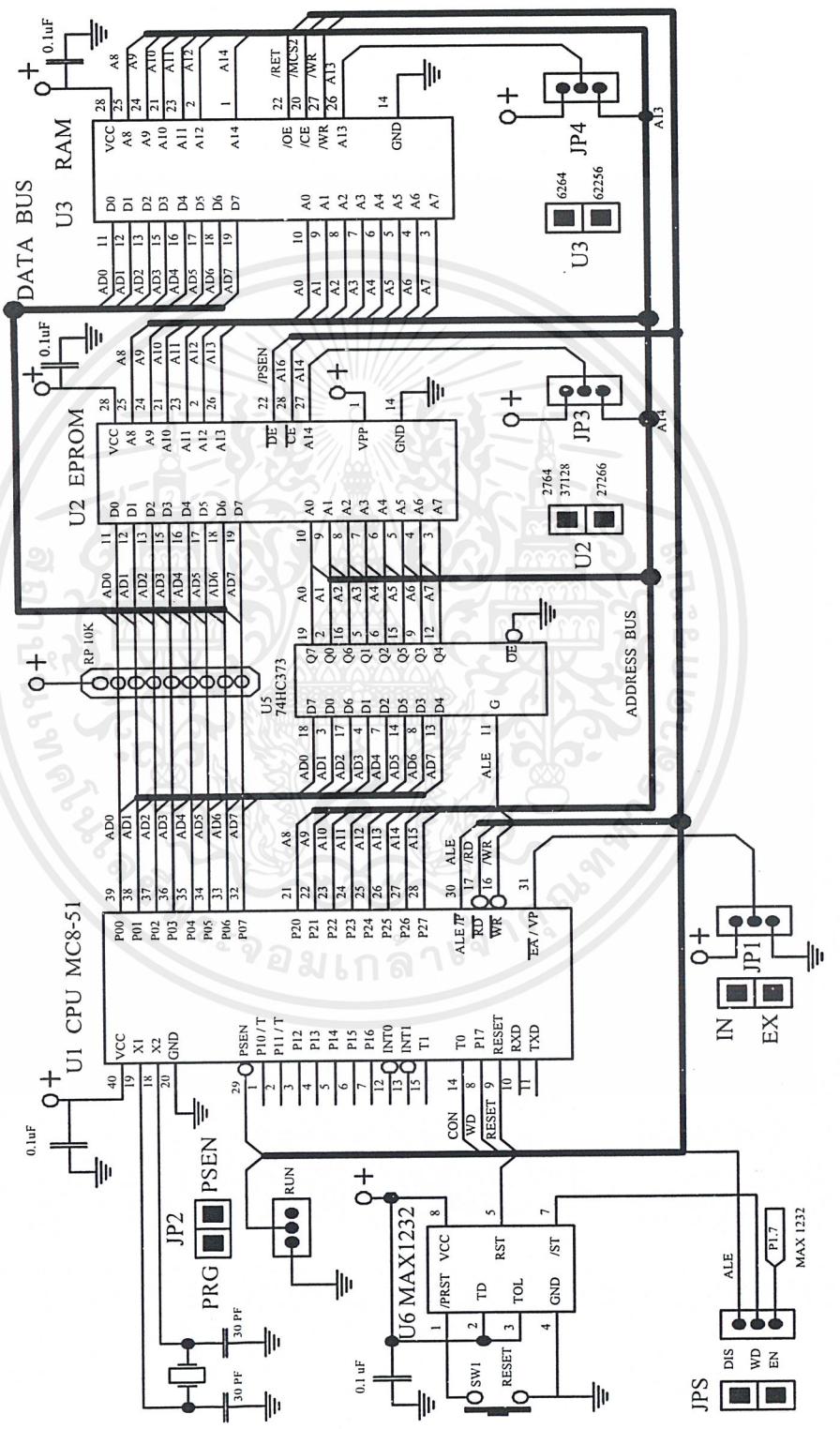
ในส่วนของวงจรรับข้อมูลจากเครื่องอ่านบาร์โค้ดทำหน้าที่ในการรับข้อมูลจารการรูดบัตรนักศึกษา จากนั้นทำการส่งข้อมูลให้แก่ พอร์ตอินพุตของไอซี 80C31 เพื่อนำไป ประมวลผลแสดงในรูปที่ 3.3



รูปที่ 3.3 ชุดเครื่องอ่านบาร์โค้ด

ในการออกแบบ ส่วนของการอ่านข้อมูลรหัสบาร์โค้ดนั้นได้นำบาร์โค้ดไปต่อร่วมกับวงจรควบคุมซึ่งในส่วนของวงจรควบคุมประกอบด้วยไอซีไมโครคอนโทรลเลอร์ 80C31 รอม และแรมซึ่งใช้ในการเก็บข้อมูล

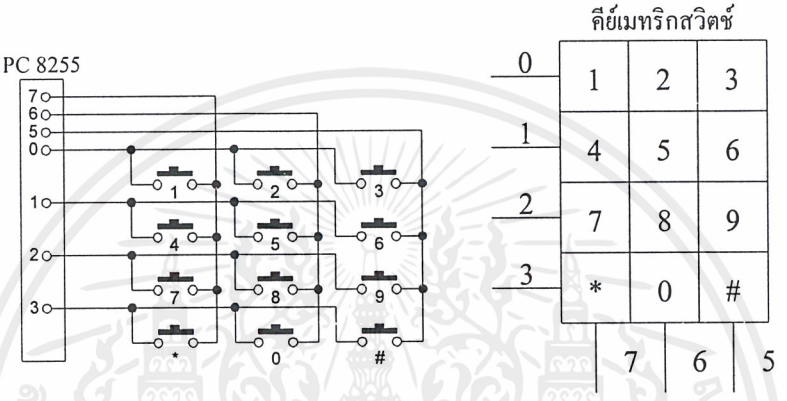
เมื่อมีผู้ต้องการใช้เครื่องมือในห้องปฏิบัติการโครงงานผู้ใช้สามารถกระทำได้โดยกดคีย์เมทริกซ์สวิตช์ใด ๆ และนำบัตรนักศึกษาที่มีรหัสบาร์โค้ด ซึ่งเป็นรหัสประจำตัวของผู้ถือบัตรจำนวน 8 ตัวเลข ในการเข้าใช้เครื่องได้นั้นรหัสของบัตรผู้นั้นต้องตรงกับรหัส Pass Word ที่ทำการกด เมื่อทำการรูดบัตรนักศึกษาผ่านเครื่องอ่านบาร์โค้ดรหัสข้อมูลจากบัตรจะถูกอ่านข้อมูลออกมาและข้อมูลจะถูกส่งออกที่ขา Data ของบาร์โค้ด รหัสข้อมูลที่ออกมาจากเอาต์พุตของเครื่องอ่านบาร์โค้ด จะส่งข้อมูลไปยังพอร์ตอินพุต P1.0 ของไอซีไมโครคอนเทลเลอร์ 80C31 โดยที่ไมโครคอนโทรลเลอร์จะทำหน้าที่แปลงข้อมูลที่อ่านได้จากบาร์โค้ดเป็นรหัส ASCII และทำการประมวลผลต่อไปผล ดังรูป 3.4



รูปที่ 3.4 วงจรส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด

2. ส่วนของวงจรรับข้อมูลคีย์เมทริกซ์สวิตช์

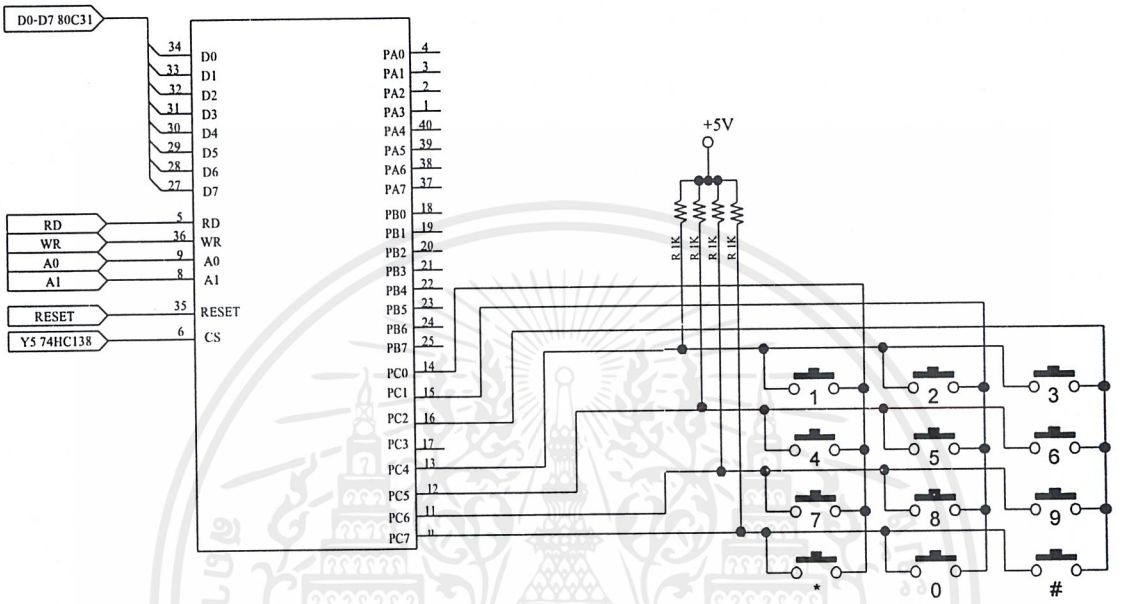
ในส่วนของวงจรรับข้อมูลคีย์เมทริกซ์สวิตช์จะทำหน้าที่ในการรับสัญญาณข้อมูลจากการกดรหัสของผู้ที่ต้องการใช้เครื่องมือบนคีย์เมทริกซ์สวิตช์



รูปที่ 3.5 คีย์เมทริกซ์สวิตช์

คีย์เมทริกซ์สวิตซ์มี 12 ปุ่มกด ประกอบด้วยตัวเลข 0-9 และเครื่องหมาย 2 เครื่องหมาย คือ* และ# โดยแบ่งการเดินสายออกเป็นทางแนวนอน และแนวตั้งของแต่ละปุ่มกด ดังรูปที่ 3.5 คีย์เมทริกซ์สวิตช์จะให้ผู้ใช้กดกหัส 4 ตัว โดยการกดแต่ละครั้งจะได้เอาต์พุตทางแนวตั้งและแนวนอนอย่างละค่า โดยที่เอาต์พุตจะให้รหัสที่ไม่ซ้ำกัน 12 ค่า และจะมีตัวต้านทานต่อแบบ Pull up ที่ 4 บิตบนของพอร์ต C

การทำงานของเมทริกซ์สวิตช์จะเริ่มจากการรับข้อมูลทาง 3 บิตล่างของพอร์ต C และ 4 บิตบนจะรอรับข้อมูลเมื่อผู้ใช้กดสวิตช์ตัวใดตัวหนึ่งบนปุ่มกดรหัสของข้อมูลจะถูกส่งไปให้อินพุตของไอซี 8255 ทางด้าน 4 บิตบนของ Port C ดังรูปที่ 3.6



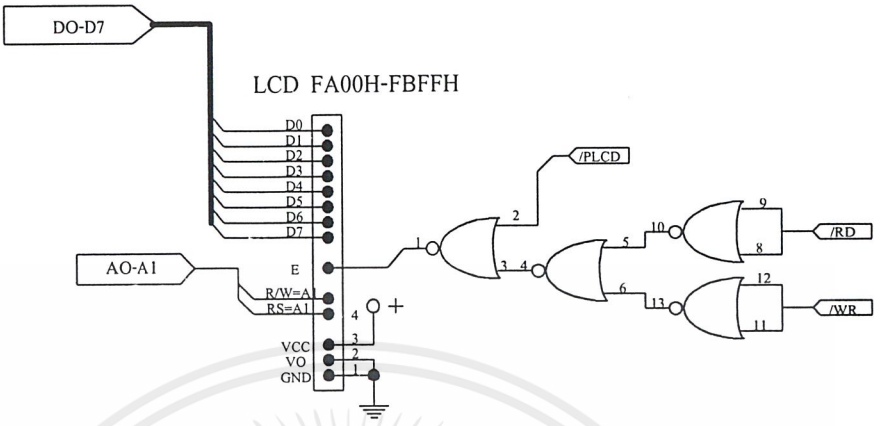
รูปที่ 3.6 ส่วนรับข้อมูลเมทริกซ์สวิตช์

3.2.3 ส่วนของการแสดงผล

ในส่วนของการแสดงผลจะใช้จอแสดงผลผลึกเหลว (LCD) ทำหน้าที่ในการแสดง วัน เวลา และเมื่อเข้าสู่ฟังก์ชั่นใช้งาน เมื่อใช้งานและทำการรูดบัตรข้อมูลที่ได้จากการประมวลผลของไมโครคอนโทรลเลอร์ที่ได้รับข้อมูลมาจากเครื่องอ่านบาร์โค้ดแสดงผลที่จอแสดงผลผลึกเหลว(LCD) บรรทัดที่ 3 จะแสดงรหัสประจำตัวของผู้ที่ต้องการใช้เครื่องมือ บรรทัดที่ 4 จะรอการกด Pass Word ของผู้ใช้จากเมทริกสวิตช์จากการกด

เมทริกซ์สวิตช์

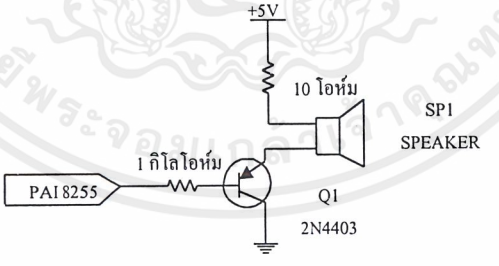
การทำงานของวงจรจอแสดงผลผลึกเหลว (LCD) เริ่มจากการส่งข้อมูลกำหนดการอ่านหรือเขียน แล้วจึงกำหนดการอ่านหรือเขียนนั้นเป็นข้อมูลหรือคำสั่ง (RS) ส่งข้อมูลออกไปทางขา Data (D0-D7) เป็นการทำงานแบบ 8 บิต การแสดงผลบรรทัดที่ 1 ,2 ,3 และ 4 ต้องกำหนดบัสเดรสซึ่งเป็นตำแหน่งในการแสดงผลบรรทัดที่ 1 จะอยู่ที่แอดเดรส 80h บรรทัดที่ 2 จะอยู่ที่แอดเดรส C0h บรรทัดที่ 3 จะอยู่ที่แอดเดรส 94h และบรรทัดที่ 4 จะอยู่ที่ 0dh



รูปที่ 3.7 ลักษณะแสดงผลของจอแสดงผลผลึกเหลว (LCD)

3.2.4 สัญญาณเสียง

เมื่อทำการรูดบัตรที่มีรหัสแถบบาร์โค้ดผ่านเครื่องอ่านบาร์โค้ด จะมีสัญญาณเสียงออกทางบัซเซอร์ ทั้งรหัสข้อมูลไม่ผ่าน และรหัสข้อมูลผ่าน โดยวงจรจะรับสัญญาณควบคุมมาจาก PA1 ของไอซี 8255 มาไบอัสทรานซิสเตอร์ให้ทำงานขับบัซเซอร์ให้กำเนิดสัญญาณเสียงเมื่อมีการรูดบัตร

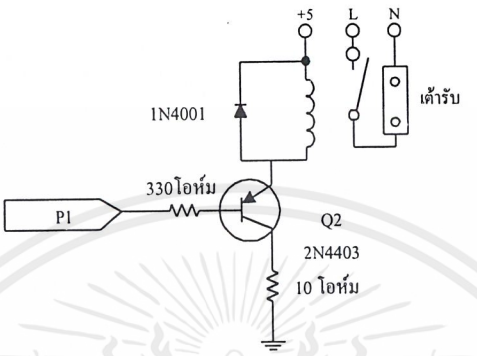


รูปที่ 3.8 วงจรสร้างสัญญาณเสียง เมื่อมีการรูดบัตร

3.2.5 ส่วนของวงจรควบคุมการจ่ายไฟ

ในส่วนของการควบคุมการจ่ายไฟ ทำหน้าที่ในการจ่ายไฟให้กับเครื่องมืออุปกรณ์ การทำงานต่อเมื่อมีการรับข้อมูลและส่งข้อมูลไปประมวลผลที่ไมโครคอนโทรลเลอร์หลังจากนั้นข้อมูลจะถูกส่งผ่าน RS-485 จะไปทำการตรวจสอบข้อมูลว่าตรงกับฐานข้อมูลที่บันทึกไว้หรือไม่ ถ้าตรงก็จะส่งข้อมูลไปที่ไมโครคอนโทรลเลอร์เพื่อทำการตรวจสอบว่ามีชุดจ่ายไฟว่างหรือไม่ว่างไม

โครคอนโทรลเลอร์ก็จะส่งสัญญาณออกทางพอร์ต P1 ของไมโครคอนโทรลเลอร์เพื่อไบอัสให้ทรานซิสเตอร์ 2N2222 ทำงานเมื่อทรานซิสเตอร์ทำงานก็จะส่งสัญญาณไปควบคุมรีเลย์ทำงานจ่ายไฟให้กับเครื่องคอมพิวเตอร์ส่วนไดโอด 1N4001 จะทำหน้าที่ป้องกันแรงดันย้อนกลับ



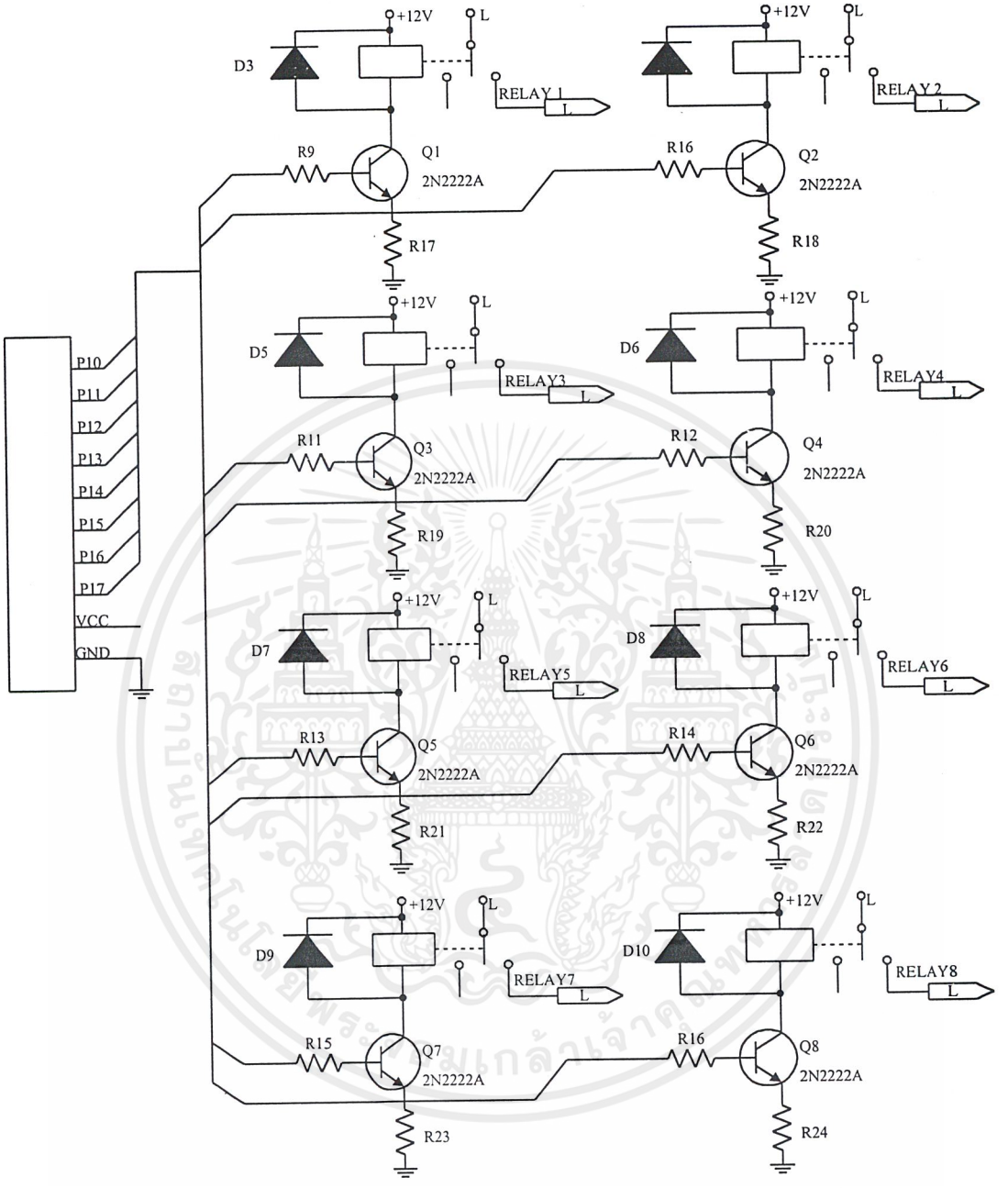
รูปที่ 3.9 ตัวอย่างวงจรควบคุมการจ่ายไฟโดยใช้รีเลย์ 1 ชุด

จากวงจรควบคุมการจ่ายไฟดังรูปที่ 3.9 เป็นการควบคุมการจ่ายไฟโดยใช้อุปกรณ์รีเลย์ 1 ตัว ซึ่งสามารถควบคุมการจ่ายไฟให้กับโต๊ะอุปกรณ์ 1 ตัว แต่ในความเป็นจริงต้องควบคุมโต๊ะอุปกรณ์ถึง 16 ตัว โดยจะนำวงจรดังรูปที่ 3.9 มาต่อเพิ่มเข้าไป 8 ชุด ดังรูปวงจรที่ 3.10 เพื่อควบคุมอุปกรณ์การจ่ายไฟ โดยทั้ง 8 ชุด จะควบคุมพอร์ต P1 ของ MCS-51 และทั้ง 8 ชุด ควบคุมโดยพอร์ต P3

3.2.6 ส่วนสื่อสารข้อมูล

ในการสื่อสารข้อมูล จากไมโครคอนโทรลเลอร์ไปยังไมโครคอมพิวเตอร์ได้กำหนดรูปแบบของเครื่อข่ายแบบบัส การทำงานจะใช้สายตัวกลางแยกออกไปเป็นกิ่งก้าน ซึ่งจะส่งข้อมูลไปยังตัวกลางโดยมี ไมโครคอมพิวเตอร์ซึ่งทำหน้าที่เป็นตัวประมวลผลสัญญาณ ซึ่งจะรับส่งรหัสสัญญาณจากไมโครคอนโทรลเลอร์ทุกตัวโดยผ่านพอร์ต RS-485 เมื่อไมโครคอนโทรลเลอร์ได้รับข้อมูลจากเครื่องอ่านบาร์โค้ด หรือคีย์เมทริกซ์สวิตช์ แล้วนำไปประมวลผลข้อมูล และทำการส่งข้อมูลไปยังไมโครคอมพิวเตอร์ ในการรับส่งข้อมูลจะใช้ไอซี SN75176

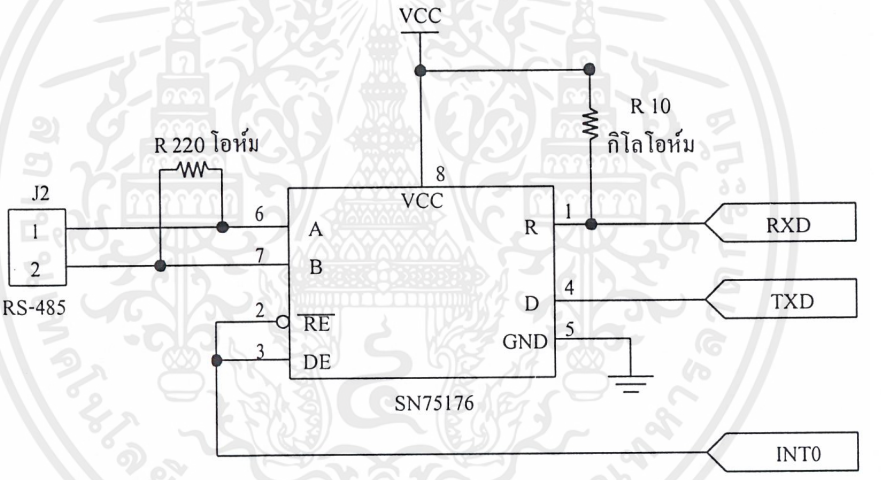
ในการส่งข้อมูลจากไมโครคอนโทรลเลอร์ 80C31 ข้อมูลจะถูกส่งออกทางขาที่ 11 ซึ่งเป็นขา TXD และข้อมูลจะเข้าที่ขา 4 ของไอซี SN75176 ซึ่งเป็นอุปกรณ์ที่มีตัวรับส่งอยู่ในชิฟเดียวกัน ขณะเดียวกันข้อมูลจะถูกส่งเข้าที่ขา INT 0 ของไอซี 80C51 เพื่อควบคุมการทำงานโดยต่อเข้าที่ขา 2



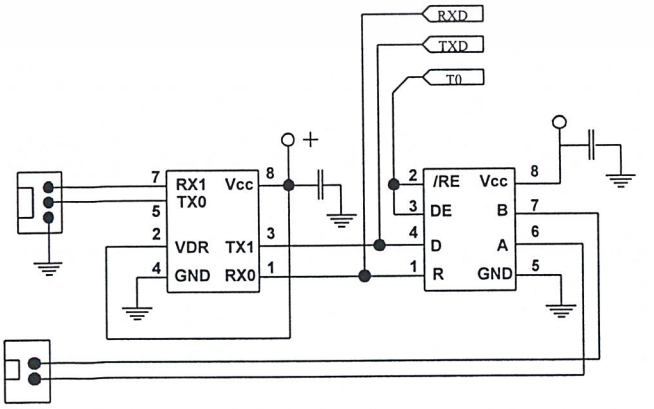
รูปที่ 3.10 วงจรควบคุมการจ่ายไฟ

และขา 3 ของไอซี SN75176 เพื่อให้มีการรับส่งข้อมูล ข้อมูลจะถูกส่งออกทางขาที่ 6 และขาที่ 7 เพื่อต่อไปยังไมโครคอมพิวเตอร์จะมีตัวแปลงจาก RS-485 เป็น RS-232 เพื่อส่งข้อมูลเข้า DB9 ของไมโครคอมพิวเตอร์ การรับข้อมูลจากไมโครคอนโทรลเลอร์ ข้อมูลจถูกส่งเข้าที่ขา 6 ขาที่ 7 ของไอซี SN75176 ข้อมูลจะถูกส่งออกที่ขา 1 และขาที่ 10 ของไมโครคอนโทรลเลอร์ซึ่งเป็นขา RXD

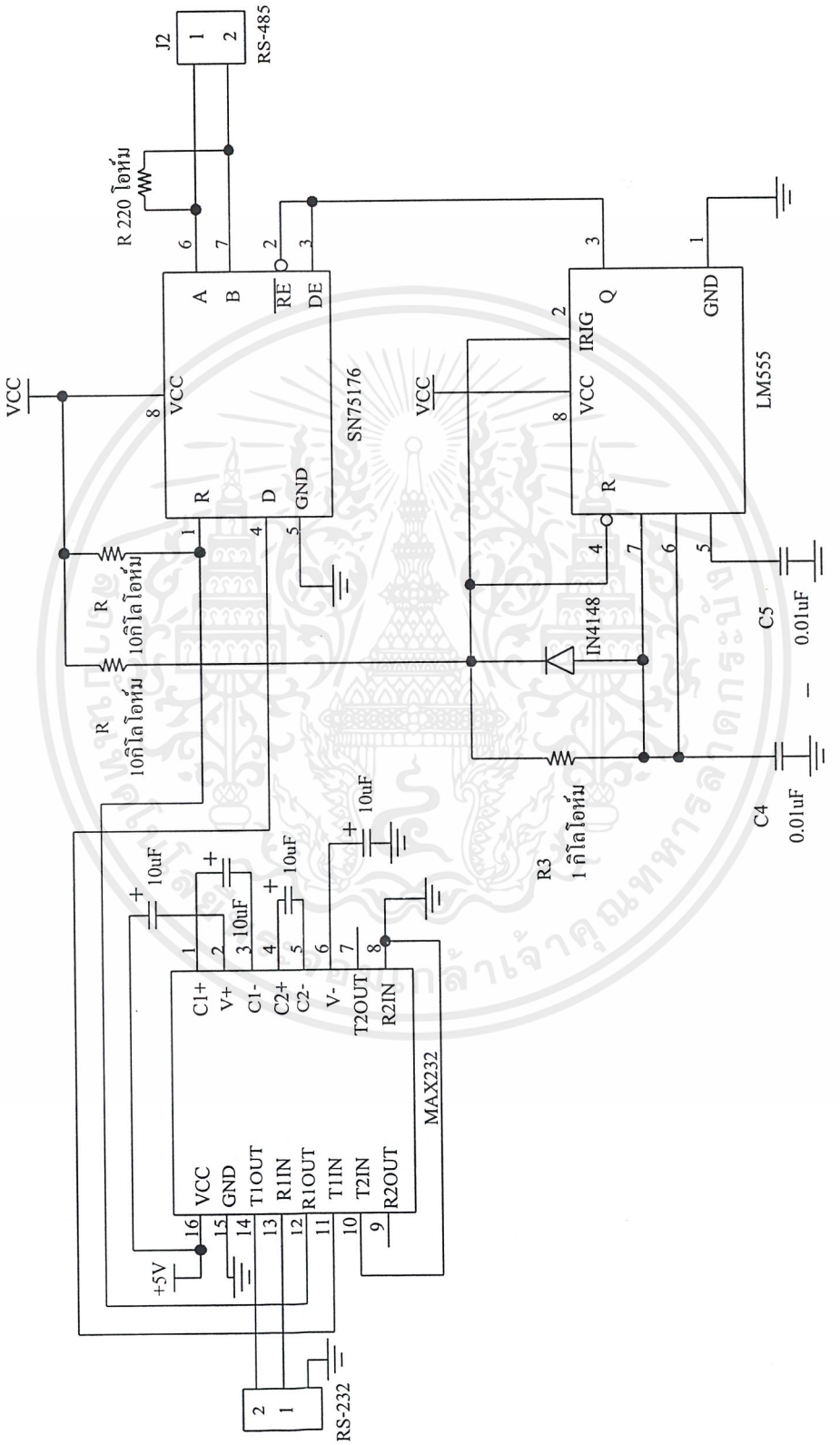
ในการสื่อสารข้อมูลระหว่างไมโครคอนโทรลเลอร์ และไมโครคอมพิวเตอร์นั้น ส่วนของการรับส่งข้อมูลของไมโครคอมพิวเตอร์นั้น ส่วนของการรับส่งข้อมูลของไมโครคอมพิวเตอร์นั้น จะมีส่วนของวงจรแปลงข้อมูลระหว่าง RS-485 และ RS-232 โดยมีการทำงานคือเมื่อมีข้อมูลส่งเข้ามาที่ขา 6 และที่ขา 7 ของไอซี SN75176 ขาที่ 2 และขาที่ 3 ของไอซี SN75176 และส่งข้อมูลออกทางขา 1 ข้อมูลจะถูกส่งออกที่ขา 12 ของไอซี MAX232 ซึ่งเป็นไอซีแปลงข้อมูลจาก RS-485 เป็น RS-232 และส่งออกทางขาที่ 13 และขาที่ 14 ส่งเข้าทางพอร์ต DB9 ของคอมพิวเตอร์เพื่อทำการเปรียบเทียบและการประมวลผลต่อไป ส่วนของการส่งข้อมูลไมโครคอมพิวเตอร์ข้อมูลจะถูกส่งออกทางพอร์ต DB9 เข้าทางขาที่ 13 และขาที่ 14 แปลงข้อมูลที่ได้จาก RS-232 และส่งสัญญาณผ่าน RS-485 ส่งเข้าที่ขาที่ 4 ของไอซี SN75176 ขณะที่ทำการส่งข้อมูลขาที่ 2 และขาที่ 3 ของไอซี SN75176 และส่งข้อมูลออกทางขาที่ 6 และขาที่ 7



รูปที่ 3.11 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ต่ออยู่กับไมโครคอนโทรลเลอร์



รูปที่ 3.12 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ติดต่ออยู่กับเครื่องอ่านบาร์โค้ด

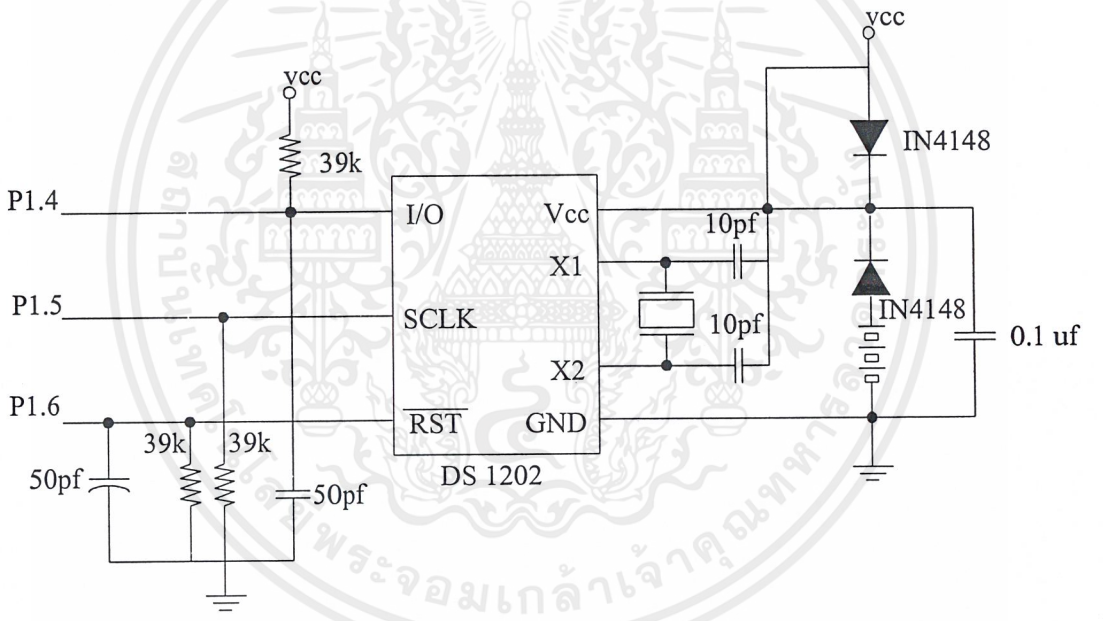


รูปที่ 3.13 วงจรสื่อสารข้อมูลผ่าน RS-485 ที่ต่ออยู่กับไมโครคอมพิวเตอร์

3.2.3 ส่วนของวงจรแสดงผลเวลา

การติดต่อกับชิปเบอร์ DS1202 จะต้องให้ RS+ และ SCLK เป็น 0 ทั้งคู่ก่อน หลังจากนั้น RST จะต้องเป็น 1 เพื่อให้ติดต่อกับรีจิสเตอร์ DS1202 ได้ และระหว่างติดต่อกับ DS1202 นั้น RST จะต้องเป็น 1 โดยการอินพุตข้อมูลเข้า DS 1202 จะเกิดขึ้นที่ขาขึ้นของ SCLK การเอาต์พุตข้อมูลจาก DS1202 จะเกิดขึ้นที่ขาลงของ SCLK และหลังจากเลิกติดต่อกับ DS1202 จะต้องให้ RST เป็น 0 อีก

จากวงจรจะใช้ไอซี DS1202 โดยที่จะต่อขาสัญญาณเข้าที่ขา P1.4 , P1.5 , P1.6 ที่ตัวไอซี 80C31 ของเครื่องอ่านบาร์โค้ด ส่วนสวิตช์ของ วงจรนาฬิกาจะต่อเข้าที่ขา PB4 - PB7 เพื่อใช้ในการตั้งเวลาเมื่อเวลาเกิดการผิดพลาด



รูปที่ 3.14 ชิป RTC เบอร์ DS1202 ต่อเข้าพอร์ต 1 ของ 8031

บทที่ 4

การทดลองและผลการทดลอง

เพื่อให้ง่ายแก่การทดลองและการตรวจสอบการทำงานของระบบ จึงได้แบ่งการทดลองออกเป็น 6 ส่วน คือส่วนที่หนึ่งการทดลองส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด ส่วนที่สองส่วนรับข้อมูลจากคีย์เมทริกซ์สวิตช์ ส่วนที่สามเป็นการทดลองส่วนของการแสดงผลส่วนที่สี่เป็นส่วนของการควบคุมการจ่ายไฟให้กับเครื่องคอมพิวเตอร์ ส่วนที่ห้าส่วนของการสื่อสารข้อมูลและส่วนที่หกส่วนของการแสดงผลของระบบ

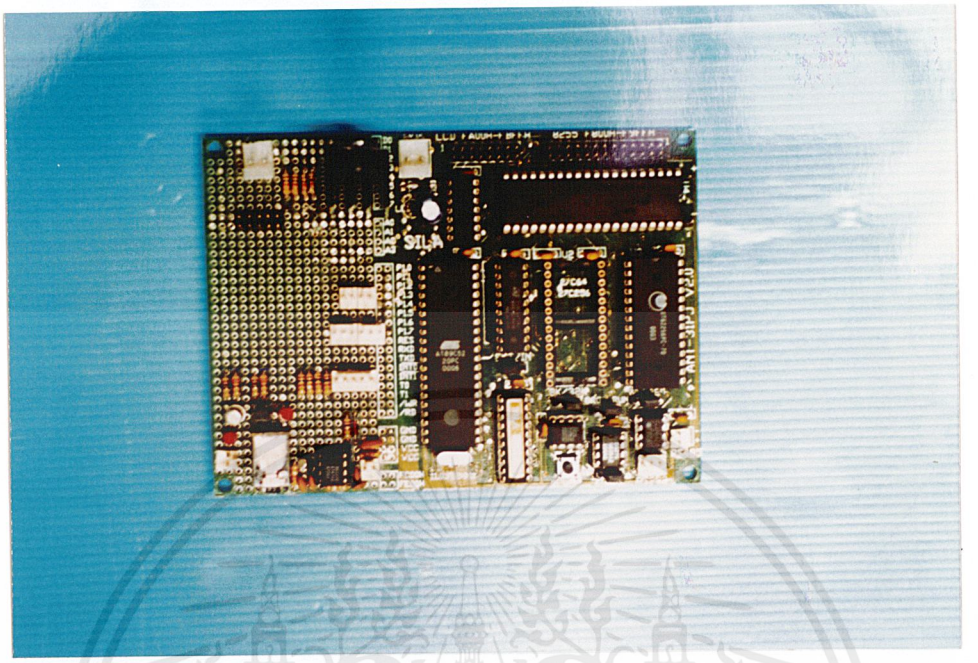
4.1 การทดลองส่วนควบคุมอุปกรณ์รับข้อมูล และอุปกรณ์แสดงผล

4.1.1 ลำดับขั้นการทดลอง

1. ประกอบวงจรควบคุมรับข้อมูล และอุปกรณ์แสดงผลตามรูปที่ 3.2
2. ตรวจสอบความเรียบร้อยของวงจร ป้อนแรงดันให้กับไอซีแต่ละตัว
3. ใช้โปรแกรมควบคุมรีเลย์ เพื่อทดสอบการทำงานของวงจร โดยต่อ LED ออกที่พอร์ต PA ของไอซี 8255 ของเครื่องอ่านบาร์โค้ด และต่อ LED ออกที่พอร์ต P1 ของเครื่องควบคุมรีเลย์เพื่อทดสอบการทำงานของเครื่อง
4. ใช้โปรแกรมทดสอบหน่วยความจำแรม(RAM)โดยการเขียนข้อมูลไปเก็บไว้ที่ตำแหน่งหน่วยความจำแรม (RAM) แล้วดึงข้อมูลจากตำแหน่งแรมมาแสดงผลออกที่อุปกรณ์แสดงผล LED

4.1.2 ผลการทดลอง

จากการทดลอง ส่วนควบคุมรับข้อมูลและอุปกรณ์แสดงผลได้ทำการทดลองเขียนโปรแกรมควบคุมรีเลย์ทดสอบวงจรควบคุมรีเลย์ออกทางพอร์ต P1 ของไอซี 89C51 โดยเขียนโปรแกรมสั่งให้พอร์ต P1 ส่งสัญญาณไปควบคุม LED จากการสั่งงานของเครื่องอ่านบาร์โค้ดโดยจะส่งสัญญาณควบคุมผ่านทางพอร์ตอนุกรม RS-485 จากการทดลอง วงจรสามารถส่งข้อมูลออกที่อุปกรณ์แสดงผลตามโปรแกรมจากที่เขียนข้อมูลไว้ที่เครื่องอ่านบาร์โค้ด ก็สามารถแสดงผลได้



รูปที่ 4.1 บอร์ดวงจรควบคุมที่สามารถทำงานได้

4.2 การทดลองส่วนการแสดงผลจอแสดงผลผลึกเหลว (LCD)

4.2.1 ลำดับขั้นการทดลอง

1. ประกอบวงจรตามรูปที่ 3.7 แล้วนำไปต่อรวมกับวงจรควบคุมในการทดลองที่ 4.1
2. ตรวจสอบความเรียบร้อยของวงจรและจ่ายไฟให้กับวงจร
3. เขียนโปรแกรมส่งข้อมูลออกแสดงผลที่อุปกรณ์แสดงผลจอแสดงผลผลึกเหลว (LCD)
4. รันโปรแกรม
5. กดสวิตช์รีเซต สังเกตการแสดงผลที่อุปกรณ์แสดงผลจอแสดงผลผลึกเหลว (LCD)

4.2.2 ผลการทดลอง

จากการทดลอง เมื่อทำการประกอบวงจรตรวจเช็คความเรียบร้อยเสร็จ แล้วรันโปรแกรมและทำการกดสวิตช์รีเซ็ตข้อมูลที่ถูกเขียนไว้ในโปรแกรม ก็จะแสดงผลออกที่อุปกรณ์แสดงผลจอแสดงผลผลึกเหลว (LCD) ซึ่งจอแสดงผลผลึกเหลว (LCD) นี้จะใช้ในการทดลองร่วมกับเครื่องอ่านบาร์โค้ด และคีย์เมทริกซ์สวิตซ์อีกครั้ง

4.3 การทดลองส่วนรับข้อมูลจากเครื่องอ่านบาร์โค้ด

4.3.1 ลำดับขั้นการทดลอง

1. นำวงจรควบคุมที่ได้ทำการทดลองในหัวข้อ 4.1 มาทำการทดลองอ่านข้อมูลบาร์โค้ดจากเครื่องอ่านบาร์โค้ดนำขา Data ของเครื่องอ่านบาร์โค้ดต่อที่พอร์ต P1.7 ของไอซีไมโครคอนโทรลเลอร์ 80C31 และจ่ายไฟให้กับบาร์โค้ด ดังรูปวงจรที่ 3.4
2. ตรวจสอบความเรียบร้อยของวงจร จ่ายไฟให้กับวงจร
3. เข้าสู่โปรแกรมที่ใช้ในการอ่านบาร์โค้ดในการทดลองนี้ใช้โปรแกรม X-Talk
4. รันโปรแกรม X-Talk
5. กดสวิตช์รีเซ็ต
6. นำบัตรนักศึกษาที่มีรหัสบาร์โค้ดมาทำการรูดบัตรผ่านเครื่องอ่านอ่านบาร์โค้ดในทิศทางขึ้นหรือลง

4.3.2 ผลการทดลอง

จากการทดลองอ่านข้อมูลจากบัตรนักศึกษา ข้อมูลจะแสดงผลออกที่โปรแกรม X-Talk เมื่อทำการรูดบัตรนักศึกษา โปรแกรมก็จะอ่านข้อมูลและแสดงผลออกที่จอแสดงผลผลึกเหลว (LCD)

รูปที่ 4.2 การทดลองนำข้อมูลแสดงผลที่จอแสดงผลผลึกเหลว (LCD)

4.4 การทดลองส่วนรับข้อมูลจากคีย์เมทริกซ์สวิตช์

4.4.1 ลำดับขั้นการทดลอง

1. ประกอบวงจรตามรูปที่ 3.6 แล้วนำไปต่อรวมกับวงจรควบคุมในการทดลองที่ 4.1
2. ตรวจสอบความเรียบร้อยของวงจร จ่ายไฟให้กับวงจร
3. เขียนโปรแกรมทดสอบอ่านข้อมูลจากการกดเมทริกซ์สวิตช์ โดยกำหนดค่าคีย์แต่ละคีย์ให้มีค่าตามรหัส BCD 8421 และให้แสดงผลออก LED ทางพอร์ต PA

4.4.2 ผลการทดลอง

จากการทดลองเขียนโปรแกรมรับค่าจากเมทริกซ์สวิตช์ เมื่อทำการทดสอบคีย์ LED ก็จะแสดงผลดังนี้

รูปที่ 4.3 การแสดงผลของ LED เมื่อกดคีย์

ตารางที่ 4.1 การแสดงผลของ LED เมื่อมีการกดคีย์เมทริกสวิตช์

ค่าคีย์เมทริกซ์สวิทช์	การแสดงผลของจอแสดงผลผลึกเหลว (LCD)							
	L8	L7	L6	L5	L4	L3	L2	L1
0	○	○	●	○	○	○	○	○
1	○	○	○	○	○	○	○	●
2	○	○	○	○	○	○	●	○
3	○	○	○	○	○	○	●	●
4	○	○	○	○	○	●	○	○
5	○	○	○	○	○	●	○	●
6	○	○	○	○	○	●	●	○
7	○	○	○	○	○	●	●	●
8	○	○	○	○	●	○	○	○
9	○	○	○	○	●	○	○	●
*	○	●	○	○	○	○	○	○
#	●	○	○	○	○	○	○	○

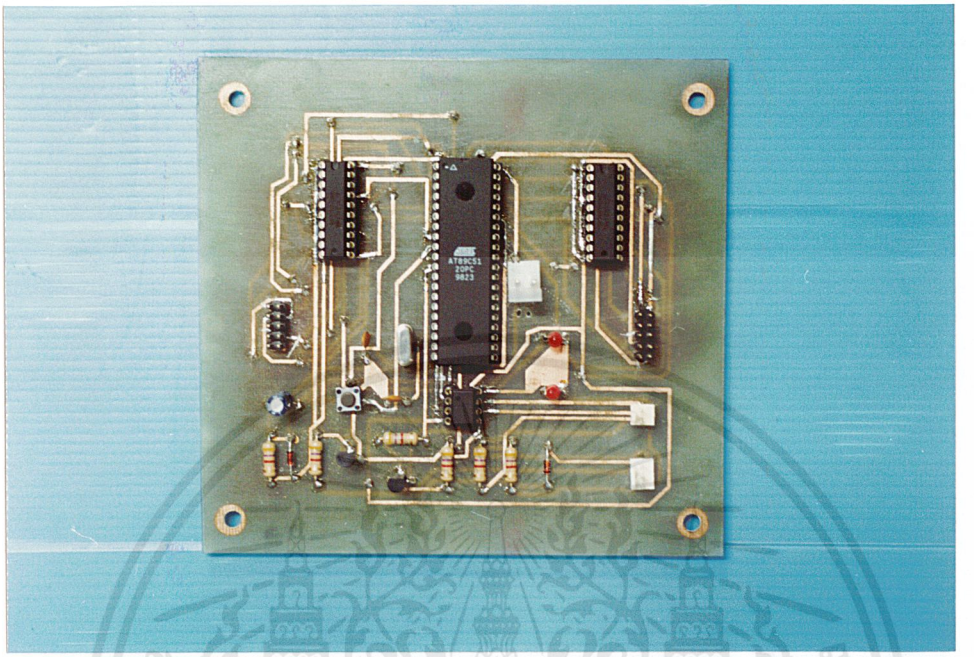
4.5 การทดลองวงจรควบคุมการจ่ายไฟให้กับโต๊ะอุปกรณ์

4.5.1 ลำดับขั้นการทดลอง

1. ประกอบวงจรตามรูปที่ 3.10 โดยนำขาเบสของทรานซิสเตอร์ต่อเข้าที่พอร์ต P1 ของไอซีไมโครคอนโทรเลอร์ 89C51
2. ตรวจสอบความเรียบร้อยของวงจร และจ่ายไฟให้กับวงจร
3. เขียนโปรแกรมส่งสัญญาณออกทางพอร์ต P1
4. รันโปรแกรม

4.5.2 ผลการทดลอง

จากการทดลองส่งสัญญาณออกทาง P1 ทำให้ทรานซิสเตอร์ทำงานจ่ายกระแสออกทางขาเบส ทำให้รีเลย์ที่ต่อทางขาเบสได้รับแรงดันและทำงานสับสวิตว์ป้อนไฟให้กับโต๊ะอุปกรณ์ทำงาน ส่วนไดโอดที่ต่อทางขาคอลเลคเตอร์เป็นตัวป้องกันแรงดันไหลย้อนกลับ



รูปที่ 4.4 วงจรควบคุมการจ่ายไฟ

4.6 การทดลองส่วนการสื่อสารข้อมูล

4.6.1 ลำดับขั้นการทดลอง

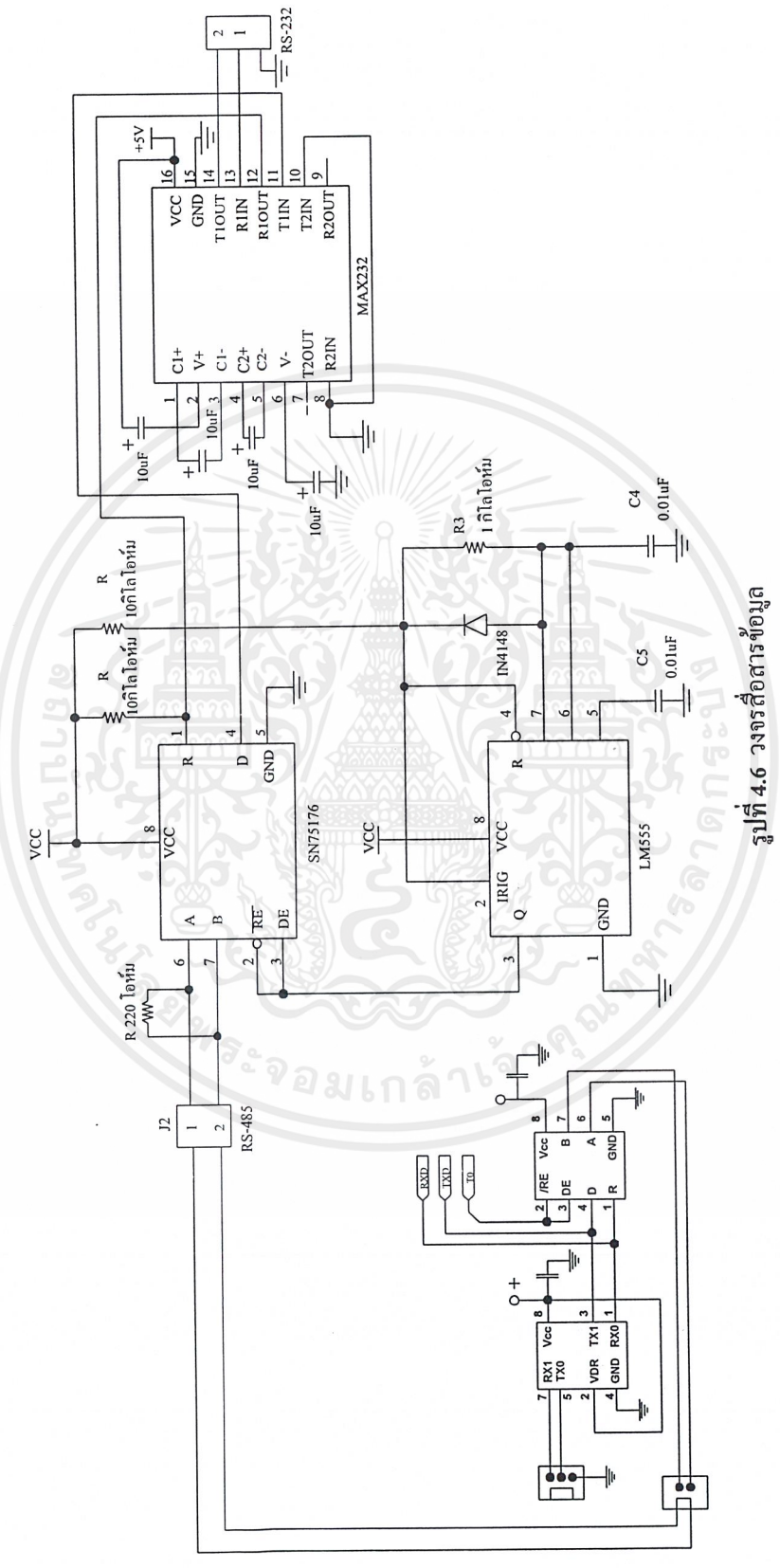
1. ประกอบวงจรสื่อสารข้อมูลตามรูปที่ 4.6
2. ตรวจสอบความเรียบร้อยของวงจร จ่ายไฟให้กับวงจร
3. เข้าสู้โปรแกรมทดสอบในการส่งข้อมูล ในการทดลองวงจรนี้ได้ใช้โปรแกรม Visual Basic ในการรับ-ส่งข้อมูลระหว่างเครื่องอ่านบาร์โค้ดกับเครื่องคอมพิวเตอร์ และในการส่งข้อมูลจากเครื่องอ่านบาร์โค้ดไปยังเครื่องควบคุมรีเลย์ โดยใช้โปรแกรม X-Talk
4. ทดลองรับ-ส่งข้อมูล

4.6.2 ผลการทดลอง

จากการทดลอง ระหว่างเครื่องอ่านบาร์โค้ดกับเครื่องคอมพิวเตอร์ในการสื่อสารข้อมูลกันผ่านวงจรสื่อสารข้อมูลโดยใช้โปรแกรมสำเร็จรูปVisual Basicในการทดลองนี้เครื่องคอมพิวเตอร์สามารถที่จะส่งสัญญาณไปควบคุมให้เครื่องอ่านบาร์โค้ด ทำการส่งข้อมูลที่มีอยู่ในหน่วยความจำ

ข้อมูล (RAM) ที่มีอยู่ไปเก็บไว้ในคอมพิวเตอร์ และเครื่องควบคุมรีเลย์สามารถรับข้อมูลจากเครื่องอ่านบาร์โค้ดไปสั่งงานให้รีเลย์แต่ละตัวทำงานได้

รูปที่ 4.5 การทดลองเครื่องสื่อสารผ่าน RS-485



รูปที่ 4.6 วงจรสื่อสารข้อมูล

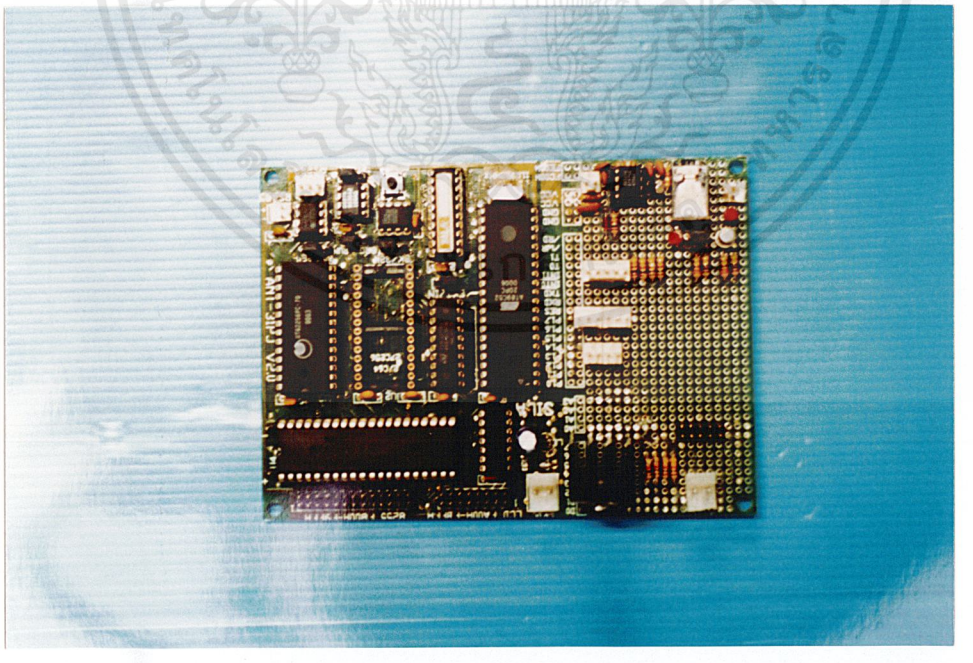
4.7 การทดลองส่วนของนาฬิกาดิจิตอล

4.7.1 ลำดับขั้นการทดลอง

1. ประกอบวงจรตามรูปที่ 3.14
2. ตรวจสอบความเรียบร้อยของวงจร จ่ายไฟให้กับวงจร
3. เข้าสู่โปรแกรมทดสอบในการส่งข้อมูล ในการทดลองวงจรนี้ได้ใช้โปรแกรม X-Talk ในการรับ-ส่งข้อมูลระหว่างเครื่องอ่านบาร์โค้ดกับวงจรนาฬิกา จากวงจรจะใช้ไอซี DS1202 โดยที่จะต่อขาสัญญาณเข้าที่ขา P1.4 , P1.5 , P1.6 ที่ตัวไอซี 80C31 ของเครื่องอ่านบาร์โค้ด ส่วนสวิตช์ของ วงจรนาฬิกาจะต่อเข้าที่ขา PB4 - PB7 เพื่อใช้ในการตั้งเวลาเมื่อเวลาเกิดการผิดพลาด
4. ทดลองโปรแกรมนาฬิกา DS1202

4.7.2 ผลการทดลอง

จากการทดลองเมื่อทำการรันโปรแกรมนาฬิกา DS1202 ที่สั่งให้ไปแสดงผลจะที่จอแสดงผลผลึกเหลว (LCD) โดยจะบอก เวลา วัน / วันที่ / เดือน / ปี เช่น 13: 00:00 S/13/05/00 เป็นต้น



รูปที่ 4.7 วงจรนาฬิกา DS1202

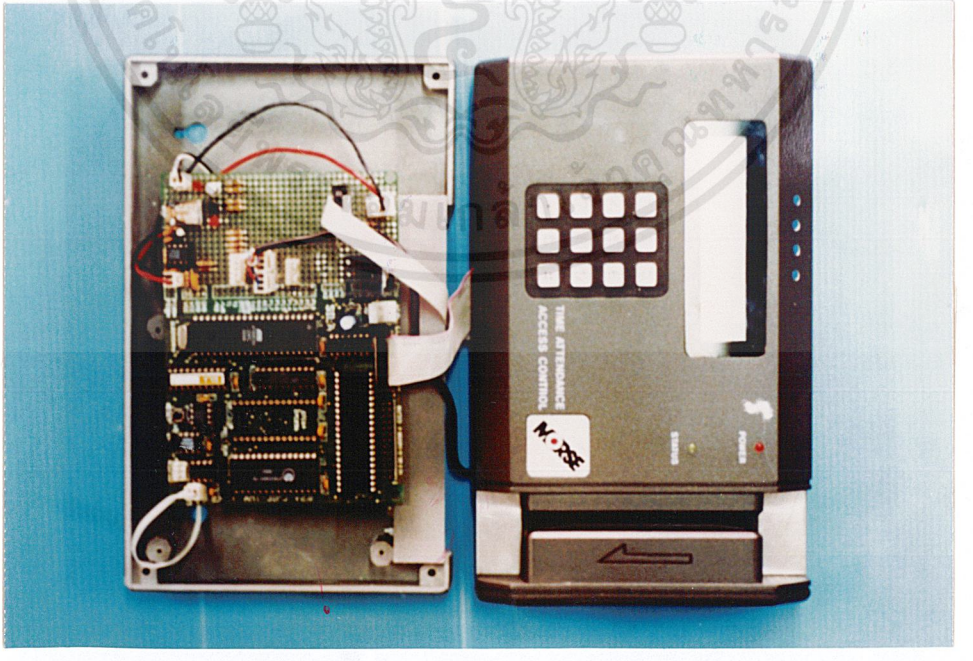
4.8 ส่วนของวงจรรวมอุปกรณ์รับข้อมูล และอุปกรณ์แสดงผล

4.8.1 ลำดับขั้นการทดลอง

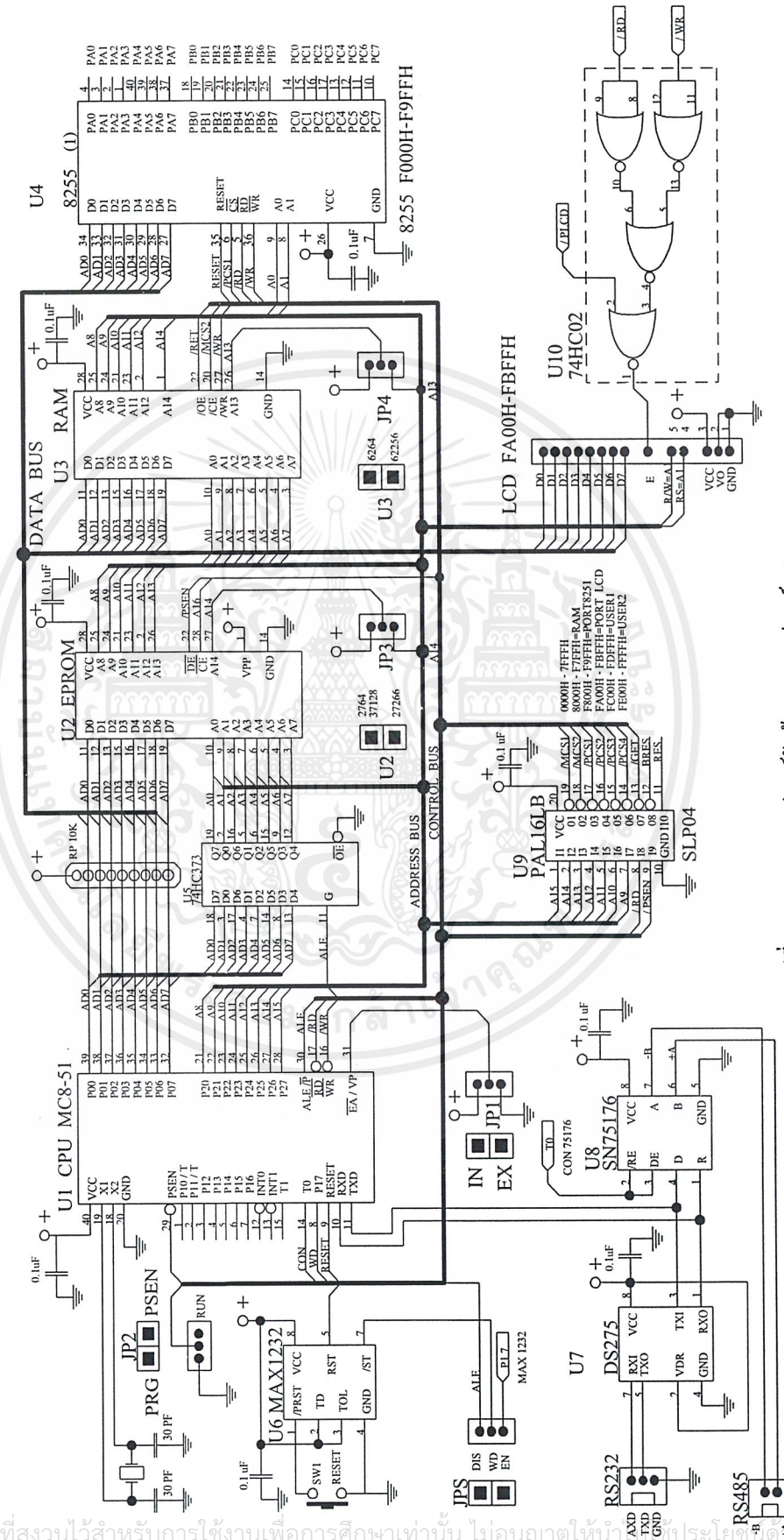
1. ประกอบวงจรตามรูปที่ 4.9
2. ตรวจสอบความเรียบร้อยของวงจร จ่ายไฟให้กับวงจร
3. รวมโปรแกรมที่ใช้เขียนควบคุมการทำงานของอุปกรณ์แต่ละส่วนเข้าด้วยกัน
4. รันโปแกรม
5. กดสวิตช์รีเซต
6. ทดลองรับข้อมูลจากอุปกรณ์รับข้อมูล และนำออกไปแสดงผลที่อุปกรณ์ แสดงผลจอแสดงผลผลึกเหลว (LCD)

4.8.2 ผลการทดลอง

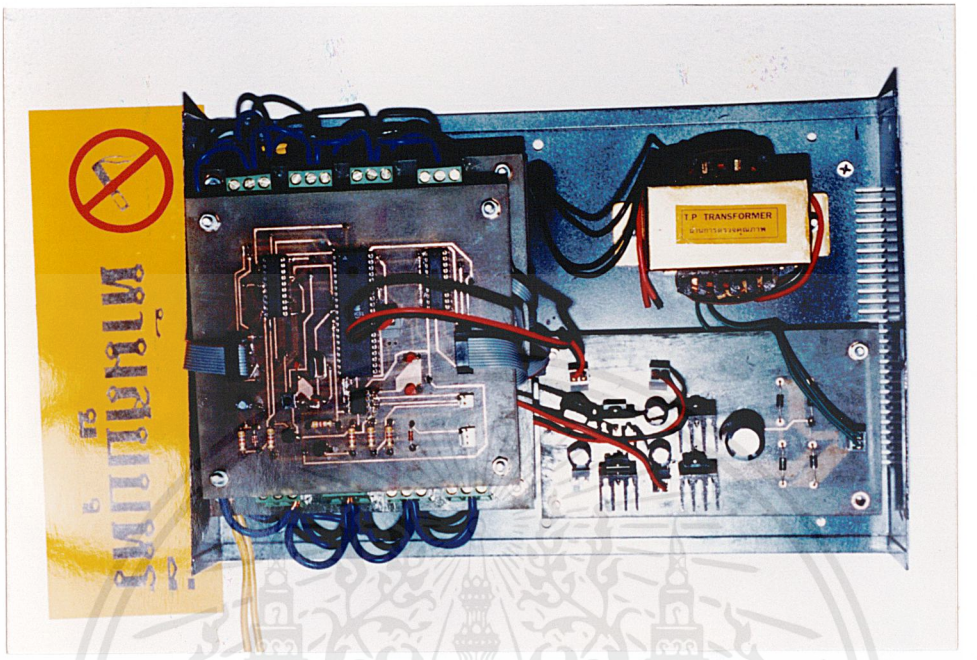
จากการทดลองรวมวงจรรับข้อมูล และอุปกรณ์แสดงผลเข้าด้วยกัน ซึ่งอุปกรณ์รับข้อมูลได้แก่เครื่องอ่านบาร์โค้ด และคีย์เมทริกสวิตช์ส่วนอุปกรณ์แสดงผลได้แก่ จอแสดงผลผลึกเหลว (LCD) เมื่อรวมวงจรเข้าด้วยกันแล้วในส่วนของโปรแกรมควบคุมต้องมีการปรับปรุงเพื่อให้วงจรแต่ละวงจรสามารถทำงานร่วมกันได้



รูปที่ 4.8 วงจรรวมการรับข้อมูล และแสดงผล



รูปที่ 4.9 วงจรรวมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล



รูปที่ 4.10 การต่อวงจรทั้งหมดเข้าด้วยกัน

รูปที่ 4.11 ผลที่ได้จากการรูดบัตรบาร์โค้ด

บทที่ 5

บทสรุป ปัญหา แนวทางแก้ไข และการพัฒนา

5.1 สรุป

ปริญญานิพนธ์ฉบับนี้เสนอผลงานของ โครงงานระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงานซึ่งใช้งานเพื่อตรวจสอบการใช้งานอุปกรณ์ปฏิบัติงานในห้องโครงงานว่าผู้ใดได้ใช้อุปกรณ์ตัวใดที่โต๊ะใดบ้าง การใช้งานอุปกรณ์ต่าง ๆ ภายในห้องโครงงานผู้เข้าไปใช้งานต้องมีข้อมูลหรือรหัสเก็บไว้ในฐานข้อมูลโดยผู้ใช้เครื่องสามารถที่จะเลือกป้อนรหัสบาร์โค้ด หรือการกดปุ่มเมทริกซ์สวิตช์ เมื่อผู้ใช้ต้องการใช้อุปกรณ์ต่าง ๆ จะต้องป้อนรหัสที่หน้าประตู อุปกรณ์ก็จะแสดงรหัสของผู้นั้น พร้อมวัน เวลา ในการใช้งาน และจะแสดงหมายเลขโต๊ะอุปกรณ์ที่ผู้ต้องการใช้ให้ทราบว่าผู้ใช้ต้องไปใช้อุปกรณ์ที่โต๊ะใดได้บ้าง แต่ถ้าโต๊ะอุปกรณ์ไม่ว่าง ก็จะแสดงว่าไม่ว่าง ซึ่งระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงานนี้ สามารถที่จะนำไปใช้ในหน่วยงานต่าง ๆ ได้

5.2 ปัญหาของโครงงาน

จากการทดลอง ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน ซึ่งได้ทดลองต่ออุปกรณ์ทีละส่วน แล้วนำทุกส่วนมาประกอบรวมกันเป็นวงจรสมบูรณ์ในการทดลองแต่ละส่วนนั้นสามารถสรุปปัญหาแนวทางการแก้ไขปัญหาในการทดลองเป็นข้อ ๆ ดังนี้

1. ในการทดลองส่วนการสื่อสารข้อมูลลายวงจรบนแผ่นวงจรมีปัญหา คือบนจุดบัดกรีนั้นเกิดขาดขึ้น การทดลองนั้นสามารถสื่อสารข้อมูลได้ แต่ข้อมูลที่ได้มีการผิดพลาด

2. ในการทดลองในส่วนของการสื่อสารข้อมูลไอซี SN75176 ซึ่งเป็นไอซีที่ใช้ในการสื่อสารข้อมูลของพอร์ตอนุกรม RS-485 เกิดการเสียขึ้นระหว่างการทดลองทำให้เกิดการล่าช้าในการทำงาน

3. ในการทดลองในส่วนของอุปกรณ์แสดงผลผลึกเหลว (LCD) คู่มือการใช้งานที่ทางบริษัทให้มานั้นไม่ตรงกับอุปกรณ์เมื่อทำการทดลองทำให้อุปกรณ์นั้นเกิดการเสียหาย

4. ในการทดลองส่วนเครื่องอ่านบาร์โค้ดเครื่องอ่านบาร์โค้ดมีราคาแพงมาก และหาซื้อยาก เสียเวลาในการจัดซื้อ

5. ในการทดลองส่วนของการอ่านข้อมูลจากบัตรแถบบาร์โค้ด ในการรูดบัตรแถบบาร์โค้ดถ้ารูดเร็วเกินไป หรือช้าเกินไปจะไม่สามารถอ่านข้อมูลได้

5.3 แนวทางการแก้ปัญหา

1. แก้ไขโดยตรวจสอบวงจรโดยการใช้มัลติมิเตอร์ตรวจวัดทีละจุด และสามารถหาจุดที่ลายวงจรขาดได้
2. แก้ไขโดยการจัดมาใหม่และซื้อสำรองไว้ซึ่งอาจมีการเสียอีก
3. แก้ไขโดยแจ้งให้ทางบริษัททราบ และทางบริษัทได้ทำการเปลี่ยนจอแสดงผลผลเหลว (LCD) ให้ใหม่โดยไม่คิดเงิน
4. แก้ไขโดยสอบถามราคาตามบริษัทที่จำหน่าย เพื่อที่จะได้จัดซื้อในราคาที่เหมาะสม
5. แก้ไขโดยรูดบัตรบาร์โค้ดในความเร็วที่สม่ำเสมอไม่เร็วหรือช้าเกินไป

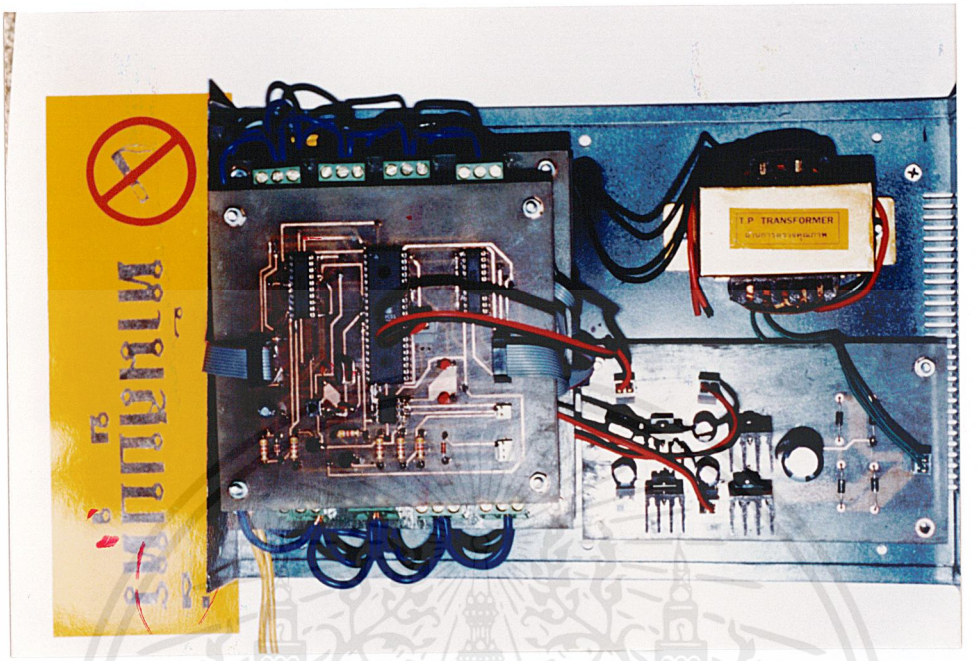
5.4 แนวทางการพัฒนาโครงงาน

ระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน สามารถทำงานได้ตามขีดความสามารถในวัตถุประสงค์ที่กำหนดไว้แต่โครงงานนี้ยังสามารถเพิ่มขีดความสามารถในการทำงานได้อีกคือ

1. ในส่วนของโปรแกรมควบคุมระบบสามารถที่จะเพิ่มรหัสผู้ที่ใช้อุปกรณ์ได้
2. ในส่วนของการแสดงผลหน้าจอแสดงผลผลึกเหลว (LCD) และจอมอนิเตอร์ของฐานข้อมูลสามารถที่จะเขียนโปรแกรมให้หน้าจอมีเมนูเพื่อที่สามารถเลือกใช้งานได้ง่ายขึ้น

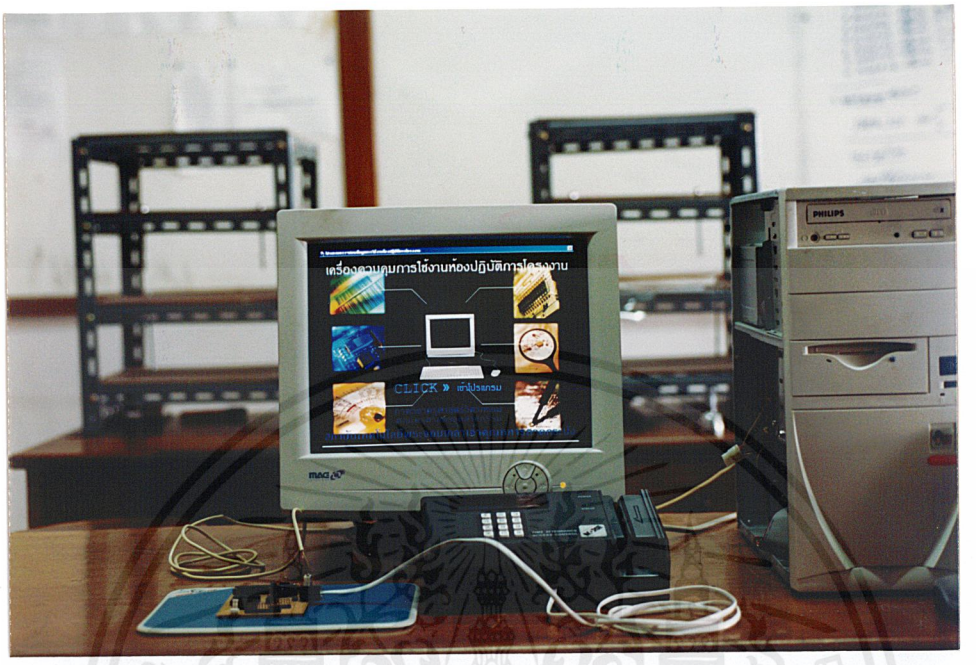
ภาคผนวก ก

รูปต้นแบบ

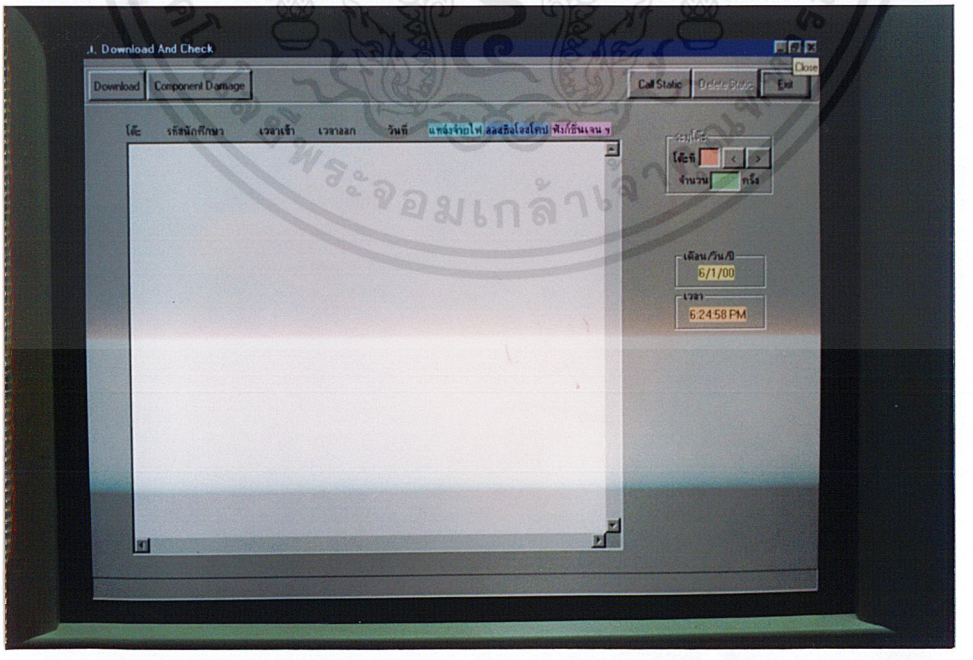


รูปที่ ก. 1 แผงวงจรรวมชุดควบคุมการจ่ายไฟสำหรับห้องปฏิบัติการ โครงงาน

รูปที่ ก.2 การต่อแผงวงจรรวมเข้ากับไมโครคอมพิวเตอร์



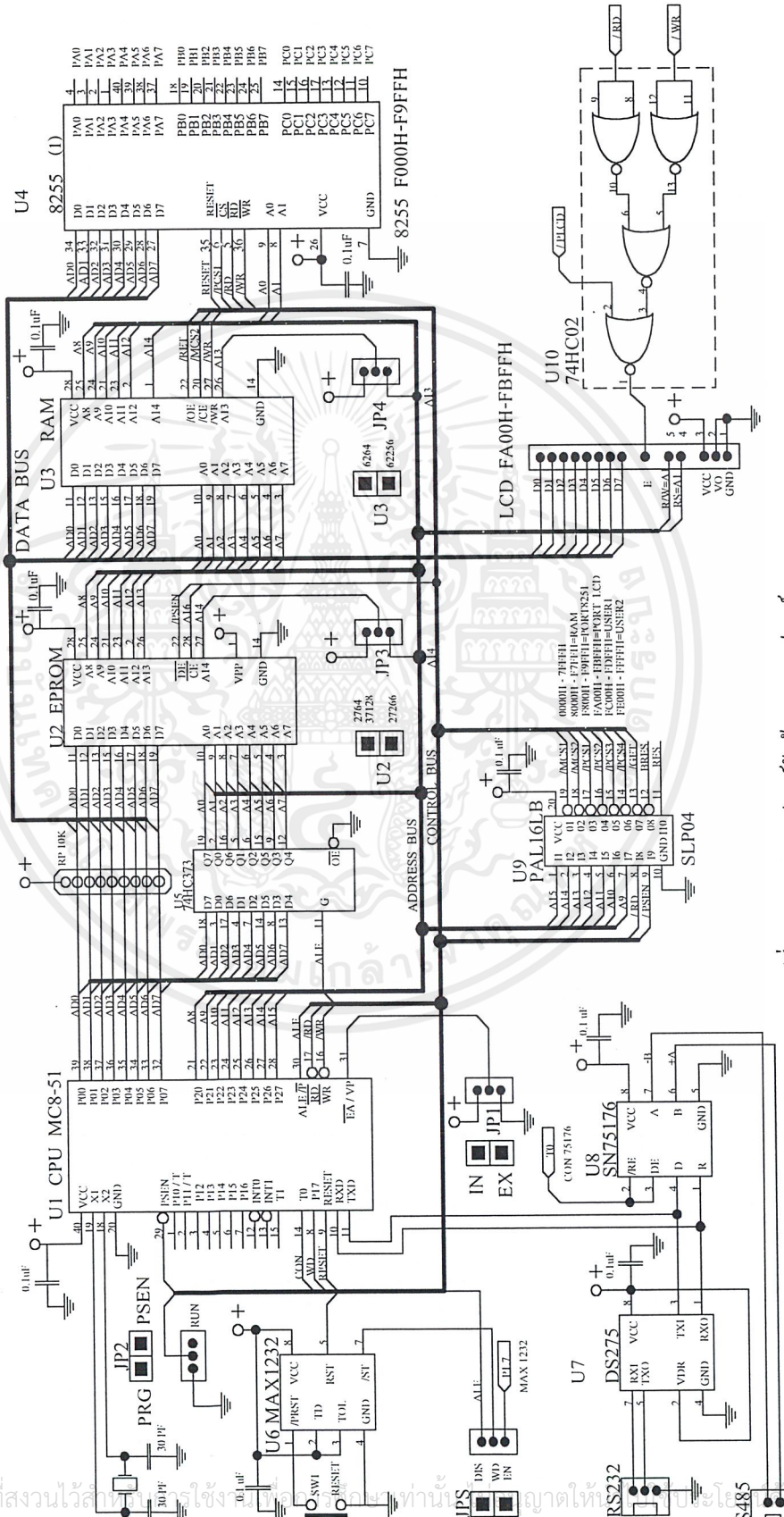
รูปที่ ก.3 การต่อส่วนสื่อสารข้อมูลเข้ากับไมโครคอมพิวเตอร์



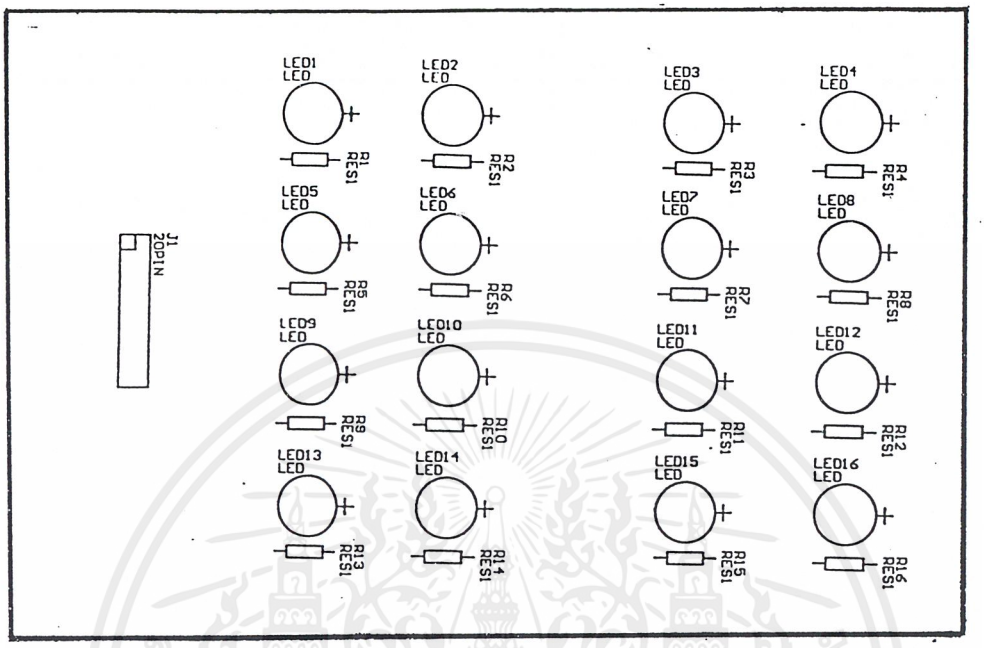
รูปที่ ก.4 ลักษณะหน้าจอของจอมอนิเตอร์ที่ใช้ควบคุมระบบ

ภาคผนวก ข

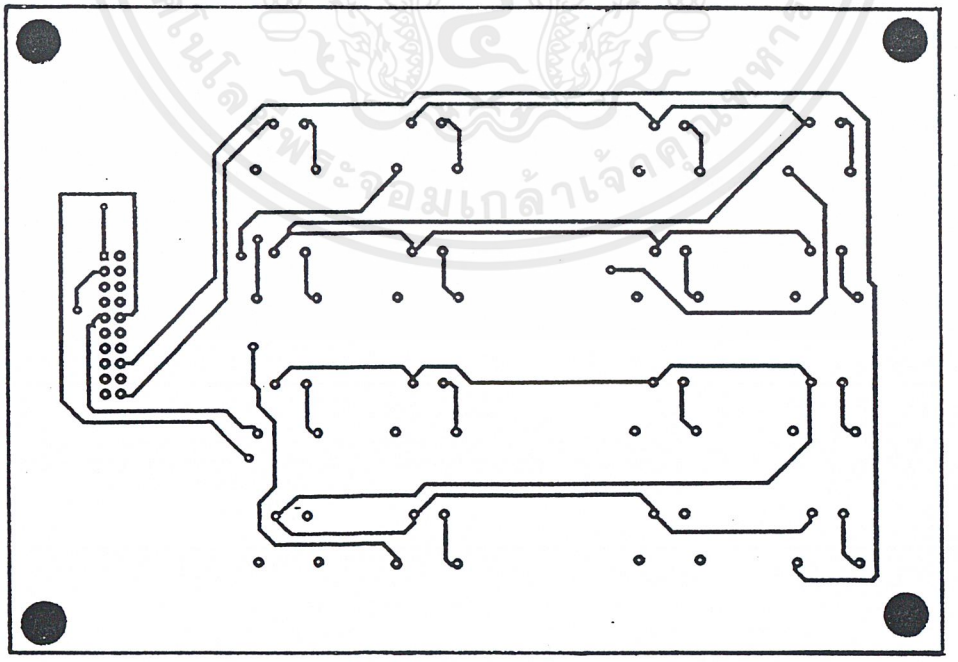
วงจรและแผ่นวงจรพิมพ์



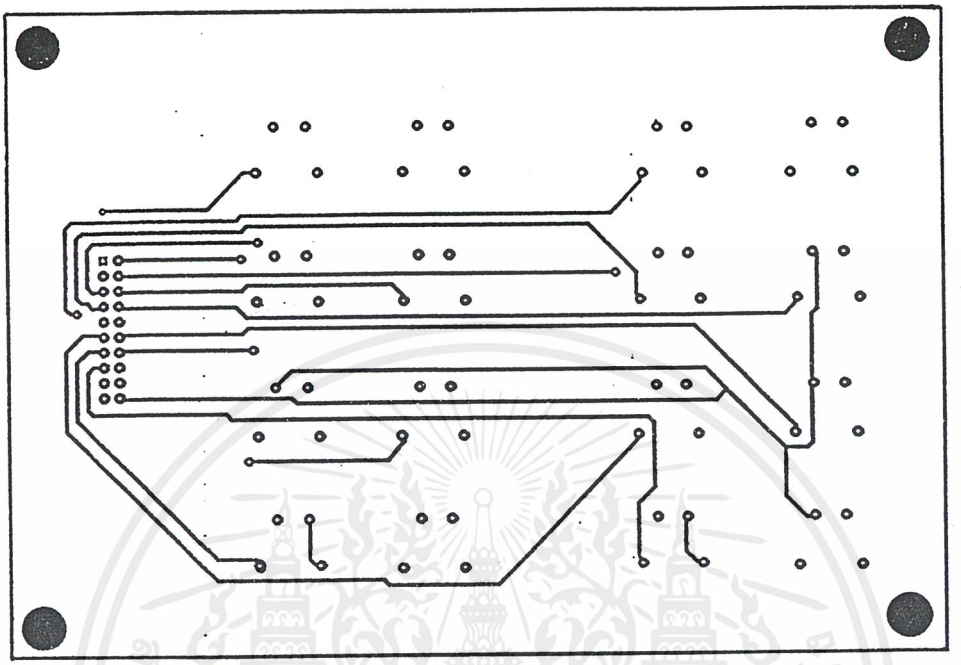
รูปที่ ข.1 วงจรรวมอุปกรณ์รับข้อมูลและอุปกรณ์แสดงผล



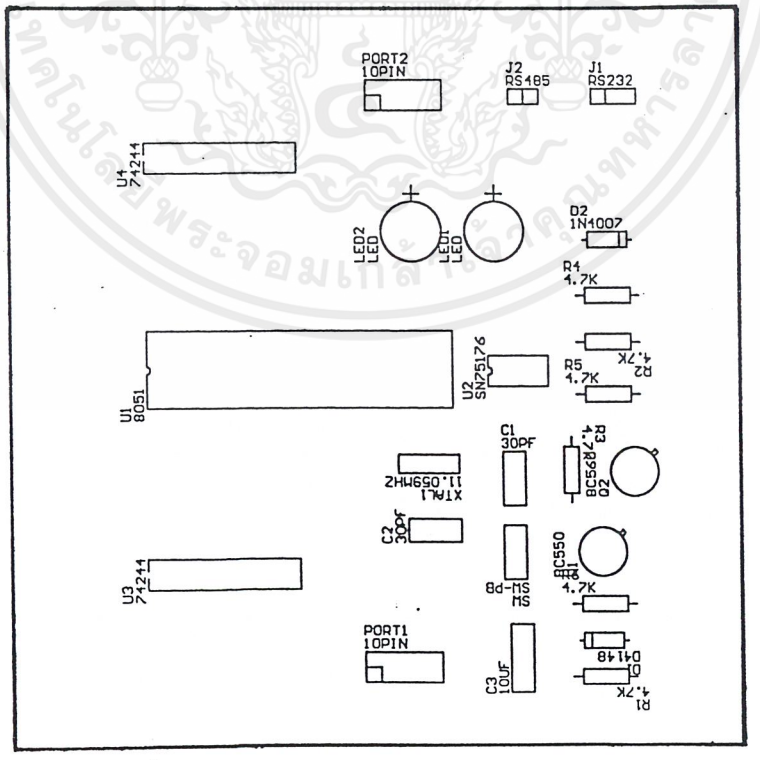
รูปที่ ข.2 ด้านวางอุปกรณ์ของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED



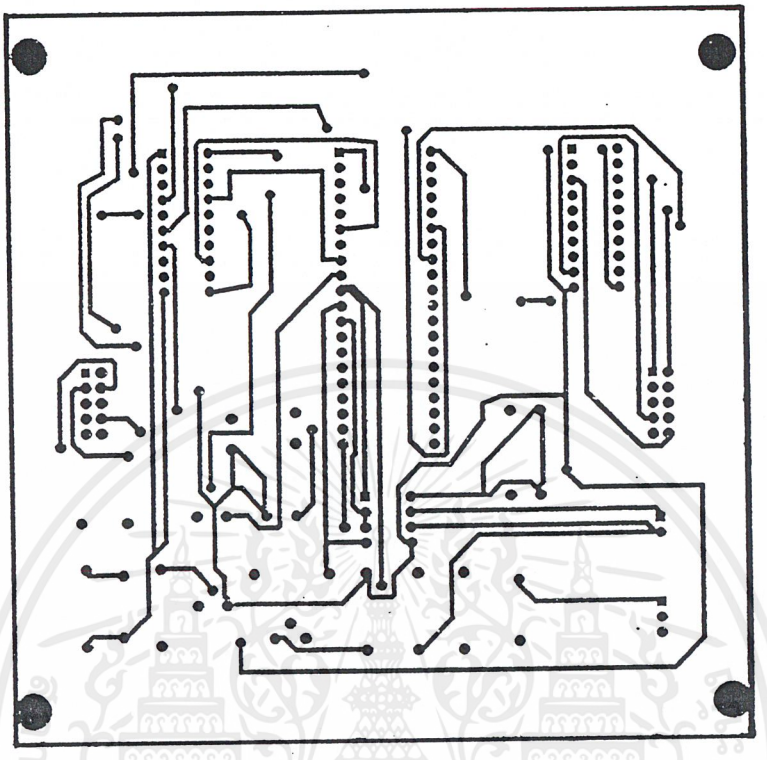
รูปที่ ข.3 ด้านบนของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED



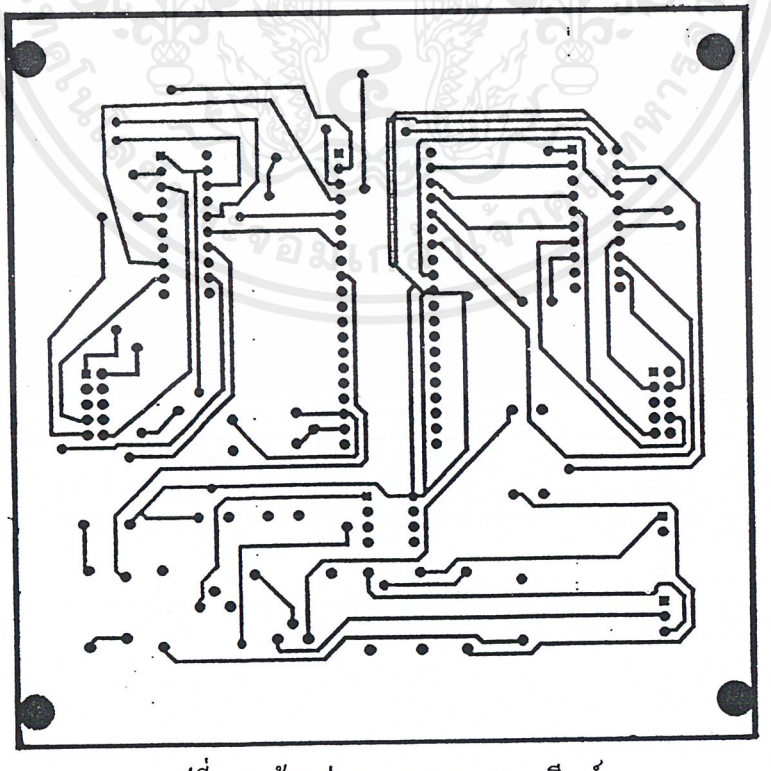
รูปที่ ข.4 ด้านล่างของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED



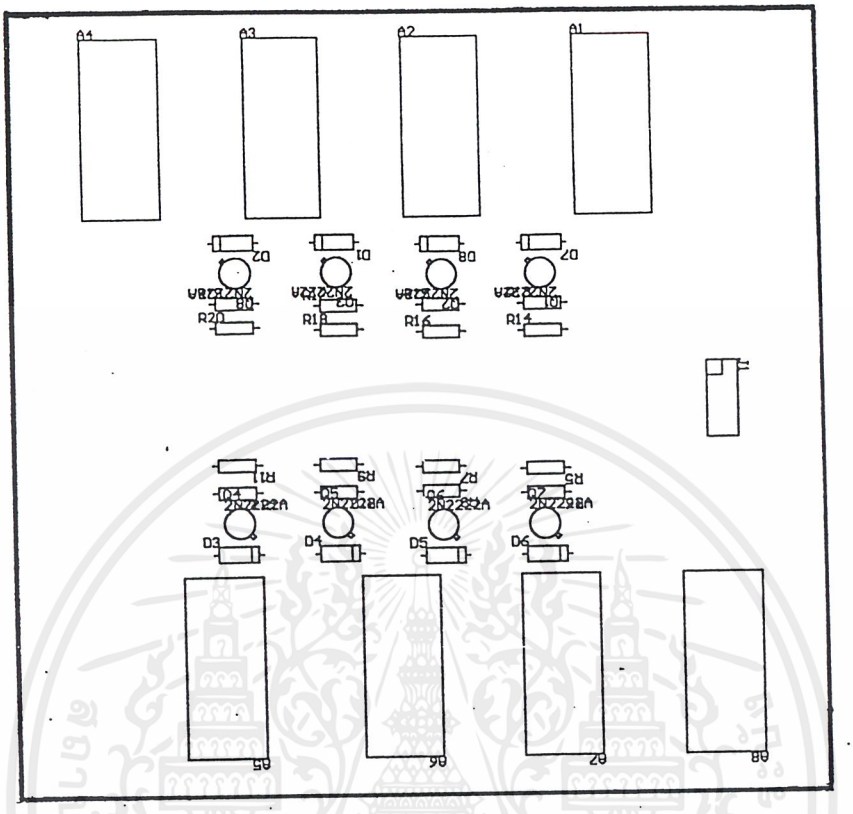
รูปที่ ข.5 ด้านวางอุปกรณ์ของวงจรควบคุมรีเลย์



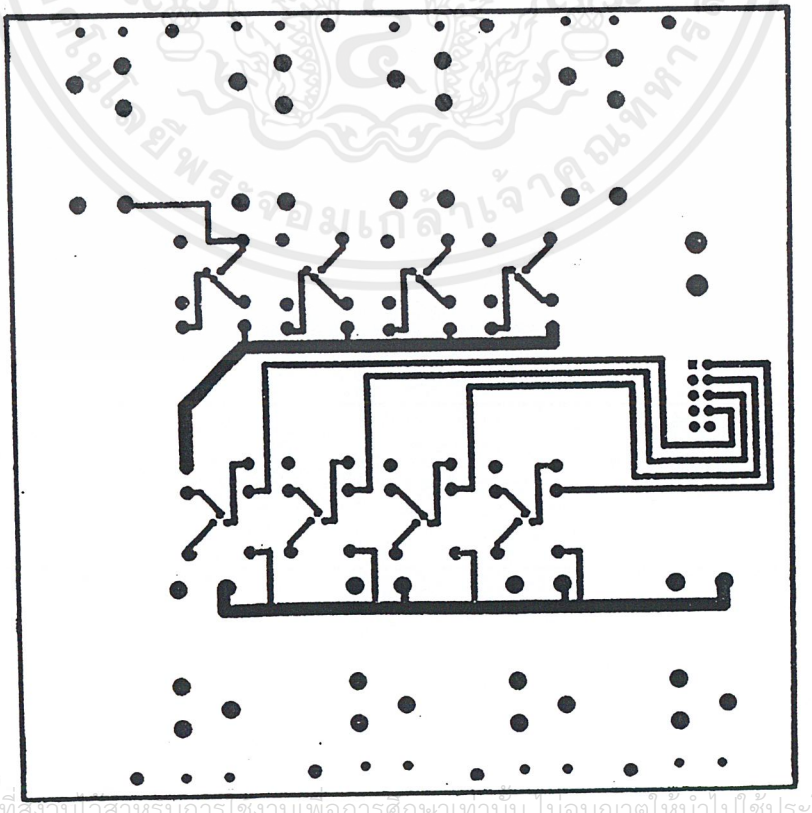
รูปที่ ข.6 ด้านบนของวงจรควบคุมรีเลย์



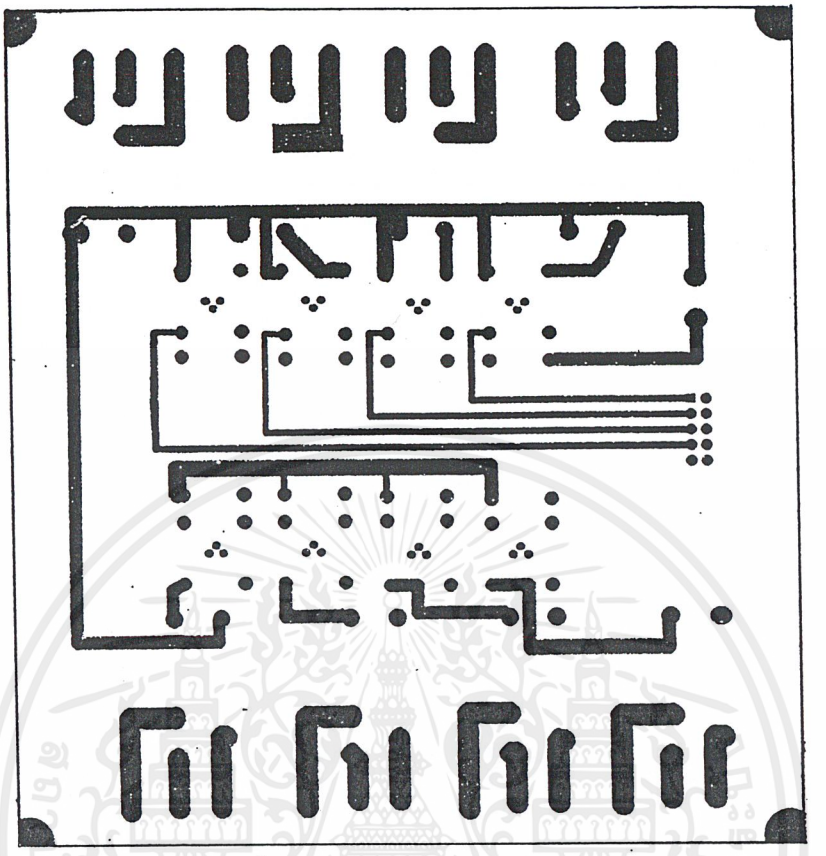
รูปที่ ข.7 ด้านล่างของวงจรควบคุมรีเลย์



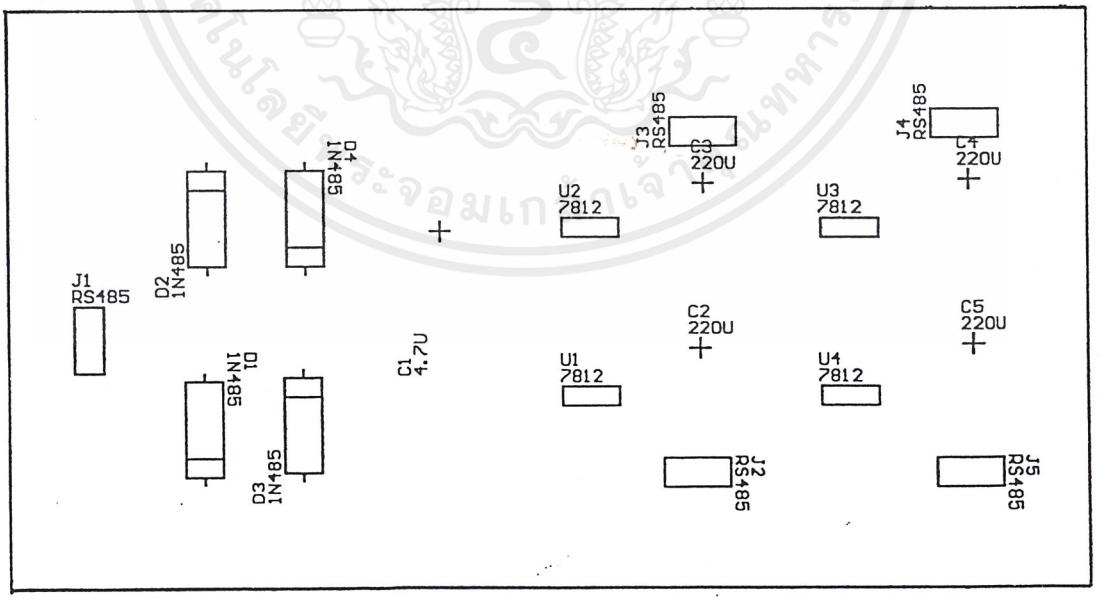
รูปที่ ข.8 ด้านวางอุปกรณ์ของวงจรรีเลย์



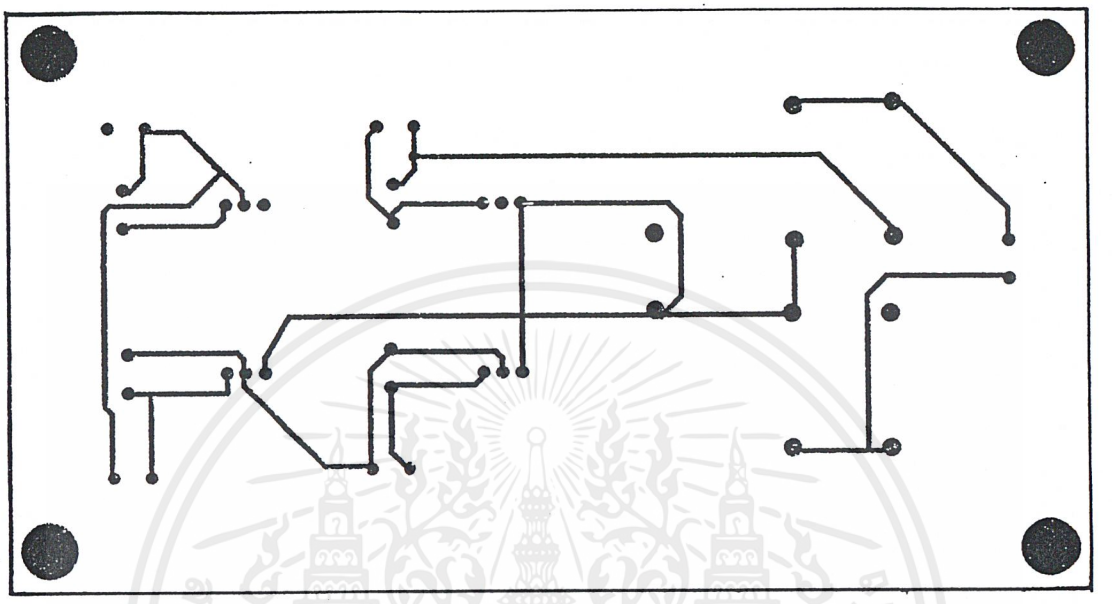
รูปที่ ข.9 ด้านบนของวงจรรีเลย์



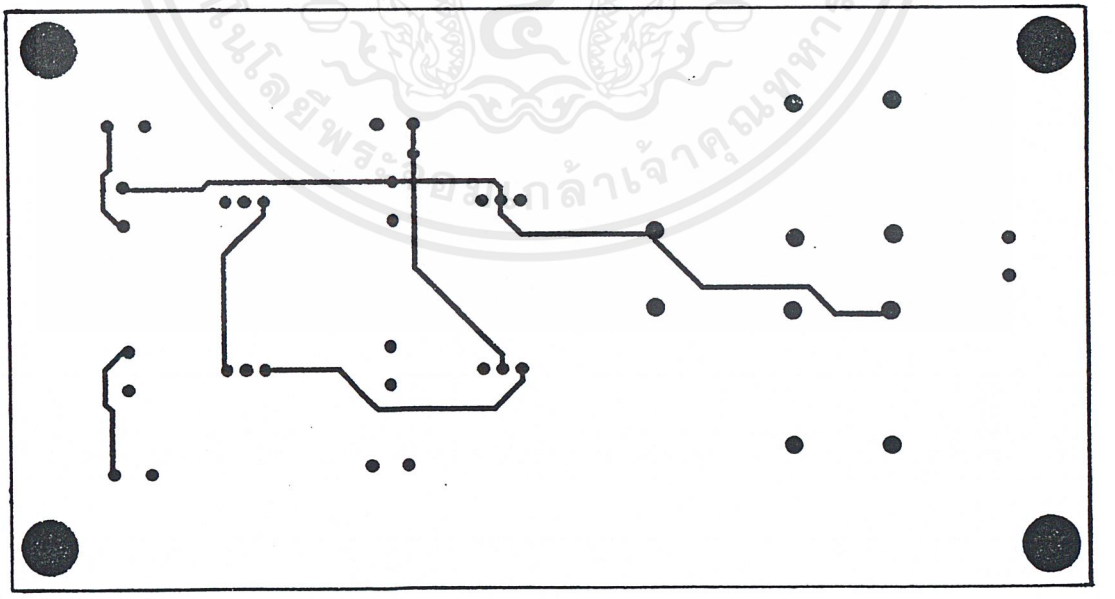
รูปที่ ข.10 ด้านล่างของวงจรรีเลย์



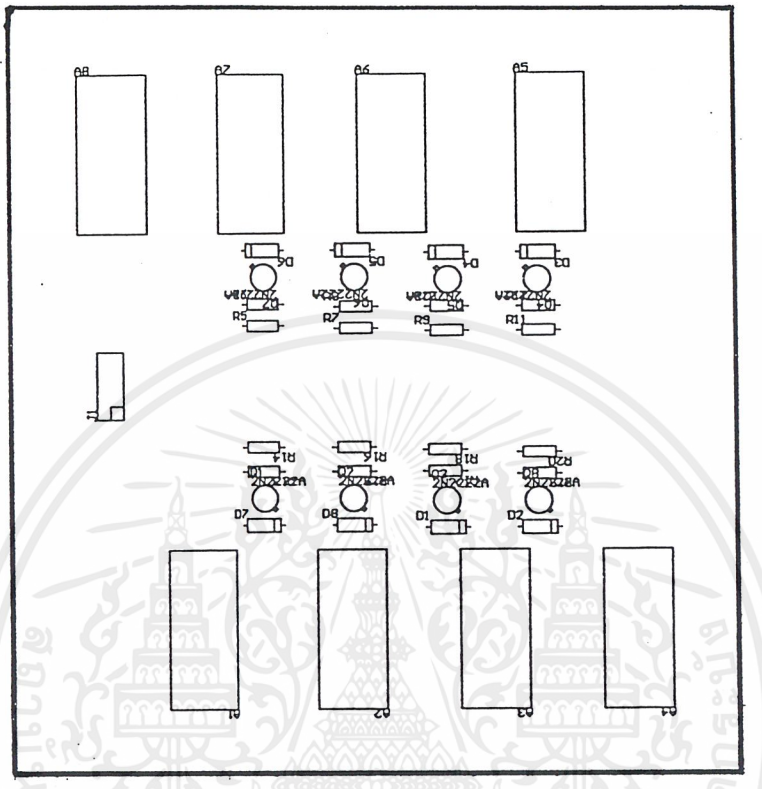
รูปที่ ข.11ด้านวางอุปกรณ์ของวงจรเรกกูเลเตอร์



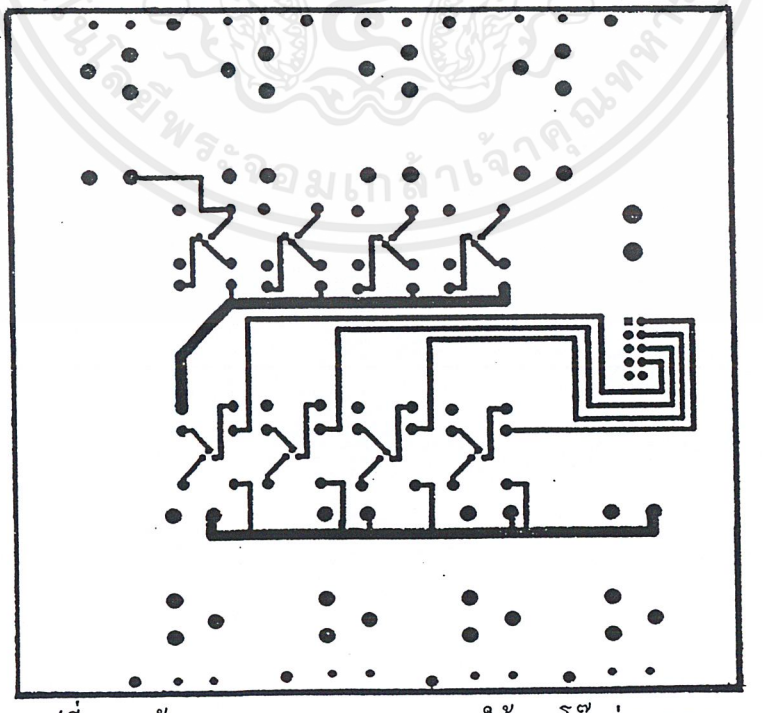
รูปที่ ข.12 ด้านบนของวงจรเรกกูเรเตอร์



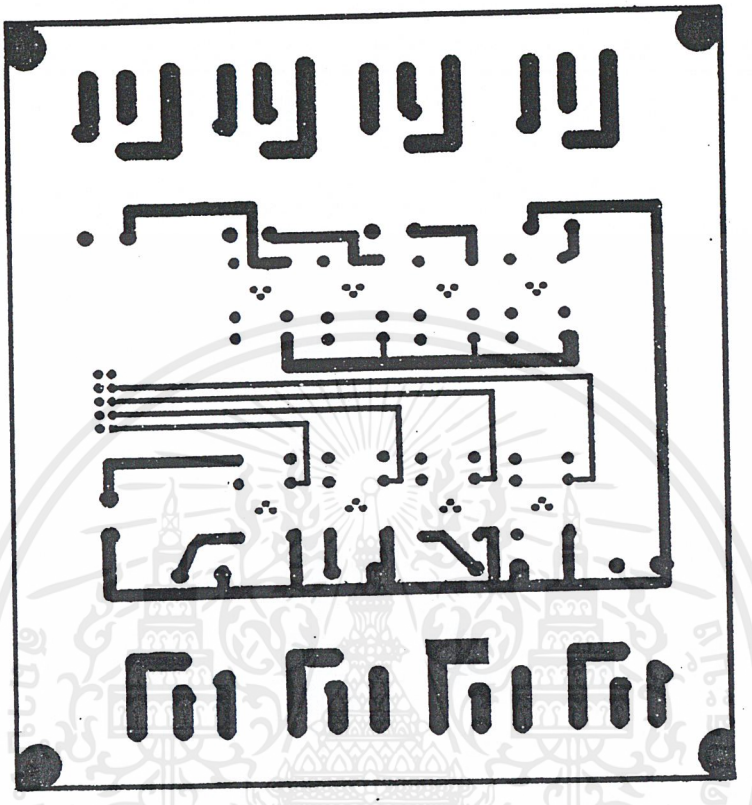
รูปที่ ข.13 ด้านล่างของวงจรเรกกูเรเตอร์



รูปที่ ข.14 ด้านวางอุปกรณ์ของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED



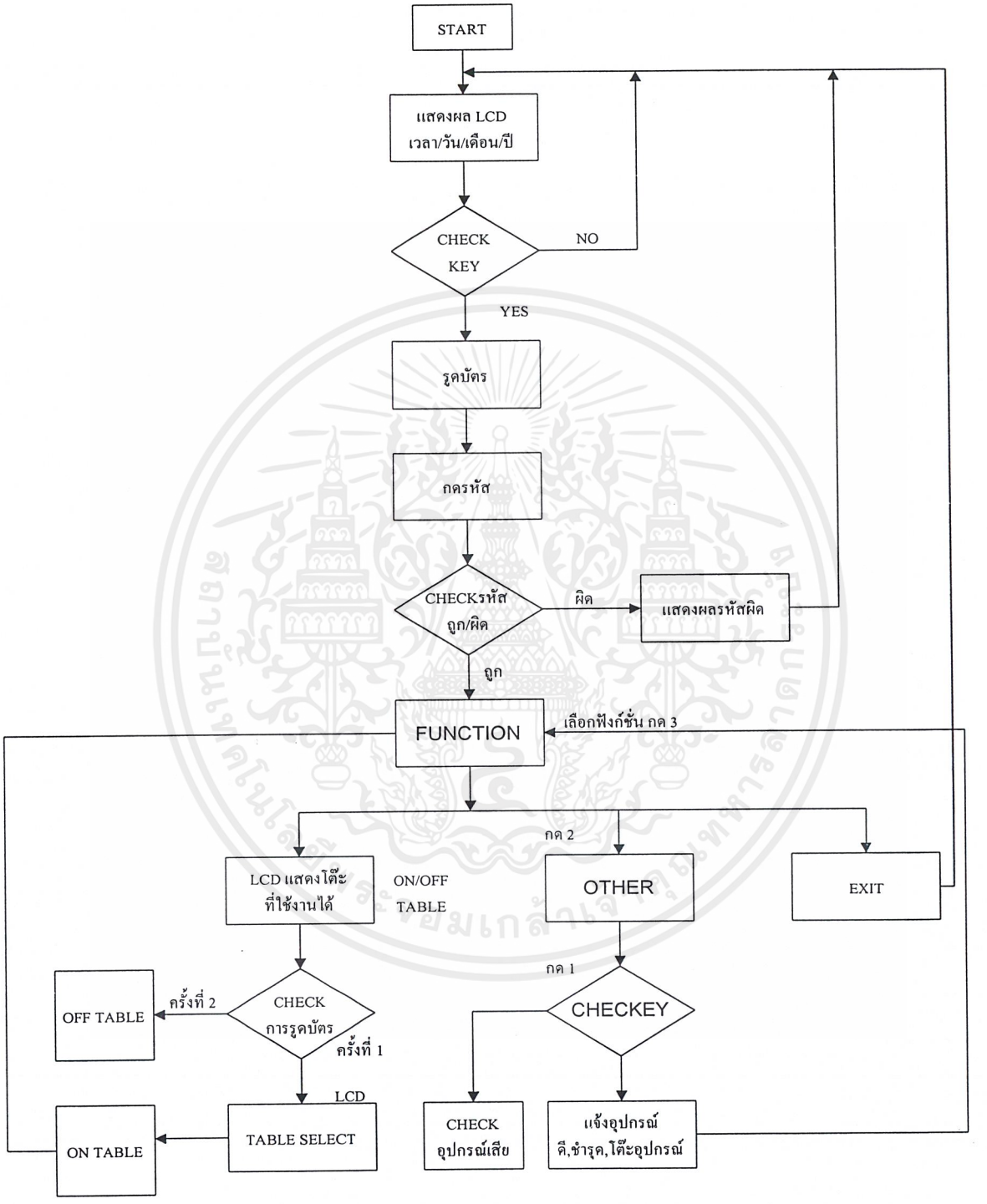
รูปที่ ข.15 ด้านบนของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED



รูปที่ ข.16 ด้านล่างของวงจรแสดงผลการใช้งานโต๊ะผ่าน LED

ภาคผนวก ค

ผังการทำงานและโปรแกรมการทำงาน



รูปที่ ค.1 ผังการทำงานของโปรแกรม

```
;*****************************************;
;                 CONTROL LCD
;*****************************************;
  COMMAND         EQU     0FA00H            ; READ-WRITE  REGISTER
  READBUSY        EQU     0FA01H            ; READ BF AND ADDRESS
  WRITEDATA       EQU     0FA02H            ; WRITE  CHARACTER
  READDATA        EQU     0FA03H            ; READ  DATA FROM DD RAM
;*****************************************;
;                 CONTROL 8255
;*****************************************;


  PORTA       EQU    0F800H
  PORTB       EQU    0F801H
  PORTC       EQU    0F802H
  PORTP       EQU    0F803H

 ;*****************************************;
 ;                ADRESS CLOKE
 ;*****************************************;
 IO           EQU      P1.4
 SCLK         EQU      P1.5
 RST          EQU      P1.6
 SEC1         EQU      70H
 SEC2         EQU      71H
 MIN1         EQU      72H
 MIN2         EQU      73H
 HOUR1        EQU      74H
 HOUR2        EQU      75H
 DATE1        EQU      76H
 DATE2        EQU      77H
 MONTH1       EQU      78H
 MONTH2       EQU      79H
 DAY1         EQU      7AH
 DAY2         EQU      7BH
 YEAR1        EQU      7CH
 YEAR2        EQU      7DH
TEMP_KEY      EQU      7EH
TEMP_SEC      EQU      7FH
TEMP_MIN      EQU      51H
TEMP_HR       EQU      52H
TEMP_DATE     EQU      53H
TEMP_MONTH    EQU      54H
TEMP_DAY      EQU      55H
TEMP_YEAR     EQU      56H
RAM_SEC       EQU      57H
RAM_MIN       EQU      58H
RAM_HR        EQU      59H
RSEC1         EQU      5AH
RSEC2         EQU      5BH
RMIN1         EQU      5CH
RMIN2         EQU      5DH
RHR1          EQU      5EH
RHR2          EQU      5FH

 ;*************************************
 ;                ADDRESS BACODE
 ;*************************************
BARBIT       EQU     P1.7
BAROUT       EQU     10H
BARREV       EQU     30H
BARBUF       EQU     0F200H
TABLE1       EQU     0F000H
TABLE2       EQU     0F018H
TABLE3       EQU     0F030H
TABLE4       EQU     0F048H
TABLE5       EQU     0F060H
TABLE6       EQU     0F078H
TABLE7       EQU     0F090H
TABLE8       EQU     0F0A8H
TABLE9       EQU     0F0C0H
```

```
TABLE10 EQU     0F0D8H
TABLE11 EQU     0F0F0H
TABLE12 EQU     0F108H
TABLE13 EQU     0F120H
TABLE14 EQU     0F138H
TABLE15 EQU     0F150H
TABLE16 EQU     80F0H
RAM1    EQU     8000H
RAM2    EQU     8800H
RAM3    EQU     9000H
RAM4    EQU     9800H
RAM5    EQU     0A000H
RAM6    EQU     0A800H
RAM7    EQU     0AC00H
RAM8    EQU     0B200H
RAM9    EQU     0B800H
RAM10   EQU     0BE00H
RAM11   EQU     0C400H
RAM12   EQU     0CA00H
RAM13   EQU     0D000H
RAM14   EQU     0D800H
RAM15   EQU     0E000H
RAM16   EQU     0E800H
SET_RAM          EQU    0F500H

ORG     0000H

MOV    SP,#20H

;*************************************
;              INIT 8255
;*************************************

;REG=A,DPTR
LCALL  DELBC
MOV    DPTR,#PORTP
MOV    A,#8AH
MOVX   @DPTR,A

;*************************************
;            INIT SERIAL PORT
;*************************************
INIT_SER:MOV    SCON,#52H
MOV    TMOD,#20H
MOV    TH1,#0FDH
SETB   TR1

;*************************************
;              SET TABLE
;*************************************

SET:     MOV    DPTR,#TABLE1
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE2
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE3
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE4
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE5
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE6
MOV    A,#00H
MOVX   @DPTR,A
MOV    DPTR,#TABLE7
MOV    A,#00H
```

```
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE8
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE9
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE10
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE11
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE12
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE13
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE14
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE15
        MOV     A,#00H
        MOVX    @DPTR,A
        MOV     DPTR,#TABLE16
        MOV     A,#00H
        MOVX    @DPTR,A

        ;*************************************
        ;               MAIN PROGRAM
        ;*************************************
MAIN1:  LCALL   CLOK
        LCALL   L_HELLO
        LCALL   LCD1
        MOV     A,#84H
        LCALL   LCDWI
        SETB    BARBIT
        CALL    BARIN
        CALL    BARCV
        JC      MAINE
        MOV     R0,#BAROUT      ;SEND
MAIN2:  MOV     A,@R0
        CJNE    A,#0DH,MAIN4
        LCALL   DELAY2
        LCALL   UBEEP_OK
        CLR     RI
        SJMP    MAIN3
MAIN4:  LCALL   LCDWD
        LCALL   DELBC
        INC     R0
        INC     DPTR
        LJMP    MAIN2
MAINE:  MOV     R2,#9FH         ;READ ERROR
        MOV     R3,#0
        LCALL   SOUND
        LCALL   DELAYK
        LJMP    MAIN1

        ;*************************************
        ;               CHECK PASS WORD
        ;*************************************

MAIN3:  MOV     A,#0CAH
        LCALL   LCDWI
        MOV     R0,#BAROUT
        INC     R0
        INC     R0
        INC     R0
        INC     R0
        CLR     A
```

```
MOV       A,@R0
          MOV     44H,A
          INC     R0
          CLR     A
          MOV     A,@R0
          MOV     45H,A
          INC     R0
          CLR     A
          MOV     A,@R0
          MOV     46H,A
          INC     R0
          CLR     A
          MOV     A,@R0
          MOV     47H,A
PASS:     CLR     A
          LCALL   SCAN1
          MOV     40H,A
          LCALL   LCDWD
          CLR     A
          LCALL   SCAN1
          MOV     41H,A
          LCALL   LCDWD
          CLR     A
          LCALL   SCAN1
          MOV     42H,A
          LCALL   LCDWD
          CLR     A
          LCALL   SCAN1
          MOV     43H,A
          LCALL   LCDWD
          LCALL   DELAY2

COMPA:    MOV     A,40H
          CJNE    A,44H,NEX
          MOV     A,41H
          CJNE    A,45H,NEX
          MOV     A,42H
          CJNE    A,46H,NEX
          MOV     A,43H
          CJNE    A,47H,NEX
          LCALL   UBEEP_OK
          LCALL   L_OK
          SJMP    MAIN
NEX:      LCALL   NO_PASS
          LJMP    MAINE

          ;***************************************
          ;              SELEC FUNCTION
          ;***************************************

MAIN:     LCALL   LCD2
          LCALL   SCAN1
          CJNE    A,#'1',OTHER
          LCALL   OFF1
          LJMP    MAIN

OTHER:    CJNE    A,#'2',EXIT
          LCALL   LCD3
          LCALL   CHECK
          LJMP    MAIN
EXIT:     CJNE    A,#'3',NEX4
          LJMP    MAIN1
NEX4:     LJMP    MAIN

          ;***************************************
          ;
          ;***************************************

CHECK:    LCALL   SCAN1
          CJNE    A,#'1',DAM
          LCALL   CHECKD
```

```
          RET
DAM:      CJNE   A,#'2',LOAD
          LCALL  SELEC
          LCALL  CK_OTHER
          RET
LOAD:     CJNE   A,#'3',CHECK
          LCALL  LCD5
CK_LOAD:  LCALL  SCAN1
          CJNE   A,#'1',NO_LOAD
          LCALL  LOAD_DATA
          RET
NO_LOAD:  CJNE   A,#'2',CK_LOAD
          RET
          ;**************************************
          ;              CHECK TABLE DAMAGE
          ;**************************************

CHECKD:   MOV    DPTR,#TABLE1
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK2
          LJMP   SHOW_D1
CHECK2:   MOV    DPTR,#TABLE2
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK3
          LJMP   SHOW_D1
CHECK3:   MOV    DPTR,#TABLE3
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK4
          LJMP   SHOW_D1
CHECK4:   MOV    DPTR,#TABLE4
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK5
          LJMP   SHOW_D1
CHECK5:   MOV    DPTR,#TABLE5
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK6
          LJMP   SHOW_D1
CHECK6:   MOV    DPTR,#TABLE6
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK7
          LJMP   SHOW_D1
CHECK7:   MOV    DPTR,#TABLE7
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK8
          LJMP   SHOW_D1
CHECK8:   MOV    DPTR,#TABLE8
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK9
          LJMP   SHOW_D1
CHECK9:   MOV    DPTR,#TABLE9
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK10
          LJMP   SHOW_D1
CHECK10:  MOV    DPTR,#TABLE10
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK11
          LJMP   SHOW_D1
CHECK11:  MOV    DPTR,#TABLE11
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',CHECK12
```

```
          LJMP    SHOW_D1
CHECK12:  MOV     DPTR,#TABLE12
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',CHECK13
          LJMP    SHOW_D1
CHECK13:  MOV     DPTR,#TABLE13
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',CHECK14
          LJMP    SHOW_D1
CHECK14:  MOV     DPTR,#TABLE14
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',CHECK15
          LJMP    SHOW_D1
CHECK15:  MOV     DPTR,#TABLE15
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',CHECK16
          LJMP    SHOW_D1
CHECK16:  MOV     DPTR,#TABLE16
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',J_NODAM
          LJMP    SHOW_D1
J_NODAM:  LCALL   NO_DAM
          RET

          ;*************************************
          ;              SHOW TABLE,DAMAEG
          ;*************************************

SHOW_D1:  LCALL   L_TDAM
          LCALL   DELAY2
          LCALL   INIT
          MOV     DPTR,#TABLE1
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',SHOW_D2
          MOV     A,#81H
          LCALL   LCDWI
          LCALL   SHOW_1
          LCALL   DELAY2
SHOW_D2:  MOV     DPTR,#TABLE2
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',SHOW_D3
          MOV     A,#0C1H
          LCALL   LCDWI
          LCALL   SHOW_2
          LCALL   DELAY2
SHOW_D3:  MOV     DPTR,#TABLE3
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',SHOW_D4
          MOV     A,#83H
          LCALL   LCDWI
          LCALL   SHOW_3
SHOW_D4:  MOV     DPTR,#TABLE4
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',SHOW_D5
          MOV     A,#0C3H
          LCALL   LCDWI
          LCALL   SHOW_4
SHOW_D5:  MOV     DPTR,#TABLE5
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',SHOW_D6
          MOV     A,#85H
```

```
          LCALL  LCDWI
           LCALL  SHOW_5
SHOW_D6:   MOV    DPTR,#TABLE6
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D7
           MOV    A,#0C5H
           LCALL  LCDWI
           LCALL  SHOW_6
SHOW_D7:   MOV    DPTR,#TABLE7
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D8
           MOV    A,#87H
           LCALL  LCDWI
           LCALL  SHOW_7
SHOW_D8:   MOV    DPTR,#TABLE8
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D9
           MOV    A,#0C7H
           LCALL  LCDWI
           LCALL  SHOW_8
SHOW_D9:   MOV    DPTR,#TABLE9
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D10
           MOV    A,#8AH
           LCALL  LCDWI
           LCALL  SHOW_9
           LCALL  DELAY2
SHOW_D10:  MOV    DPTR,#TABLE10
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D11
           MOV    A,#0C9H
           LCALL  LCDWI
           LCALL  SHOW_10
           LCALL  DELAY2
SHOW_D11:  MOV    DPTR,#TABLE11
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D12
           MOV    A,#8CH
           LCALL  LCDWI
           LCALL  SHOW_11
SHOW_D12:  MOV    DPTR,#TABLE12
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D13
           MOV    A,#0CCH
           LCALL  LCDWI
           LCALL  SHOW_12
SHOW_D13:  MOV    DPTR,#TABLE13
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D14
           MOV    A,#8FH
           LCALL  LCDWI
           LCALL  SHOW_13
SHOW_D14:  MOV    DPTR,#TABLE14
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D15
           MOV    A,#0CFH
           LCALL  LCDWI
           LCALL  SHOW_14
SHOW_D15:  MOV    DPTR,#TABLE15
           INC    DPTR
           MOVX   A,@DPTR
           CJNE   A,#'D',SHOW_D16
```

```
          MOV     A,#92H
          LCALL   LCDWI
          LCALL   SHOW_15
SHOW_D16: MOV     DPTR,#TABLE16
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP1
          MOV     A,#0D2H
          LCALL   LCDWI
          LCALL   SHOW_16

         ;*************************************
         ;               DAMAEG REPORT
         ;*************************************

C_SUP1:   LCALL   DELAY2
          LCALL   SCAN1
          CJNE    A,#'1',C_SUP2
          MOV     DPTR,#TABLE1
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP2
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE1
          LCALL   SUB_SUP
SCOP1:    MOV     DPTR,#TABLE1
          LCALL   SUB_SCOP
FUN_G1:   MOV     DPTR,#TABLE1
          LCALL   SUB_FUN
          RET

C_SUP2:   CJNE    A,#'2',C_SUP3
          MOV     DPTR,#TABLE2
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP3
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE2
          LCALL   SUB_SUP
SCOP2:    MOV     DPTR,#TABLE2
          LCALL   SUB_SCOP
FUN_G2:   MOV     DPTR,#TABLE2
          LCALL   SUB_FUN
          RET

C_SUP3:   CJNE    A,#'3',C_SUP4
          MOV     DPTR,#TABLE3
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP4
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE3
          LCALL   SUB_SUP
SCOP3:    MOV     DPTR,#TABLE3
          LCALL   SUB_SCOP
FUN_G3:   MOV     DPTR,#TABLE3
          LCALL   SUB_FUN
          RET

C_SUP4:   CJNE    A,#'4',C_SUP5
          MOV     DPTR,#TABLE4
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP5
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE4
          LCALL   SUB_SUP
SCOP4:    MOV     DPTR,#TABLE4
          LCALL   SUB_SCOP
FUN_G4:   MOV     DPTR,#TABLE4
          LCALL   SUB_FUN
```

```
          RET
C_SUP5:   CJNE   A,#'5',C_SUP6
          MOV    DPTR,#TABLE5
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP6
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE5
          LCALL  SUB_SUP
SCOP5:    MOV    DPTR,#TABLE5
          LCALL  SUB_SCOP
FUN_G5:   MOV    DPTR,#TABLE5
          LCALL  SUB_FUN
          RET

C_SUP6:   CJNE   A,#'6',C_SUP7
          MOV    DPTR,#TABLE6
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP7
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE6
          LCALL  SUB_SUP
SCOP6:    MOV    DPTR,#TABLE6
          LCALL  SUB_SCOP
FUN_G6:   MOV    DPTR,#TABLE6
          LCALL  SUB_FUN
          RET
C_SUP7:   CJNE   A,#'7',C_SUP8
          MOV    DPTR,#TABLE7
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP8
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE7
          LCALL  SUB_SUP
SCOP7:    MOV    DPTR,#TABLE7
          LCALL  SUB_SCOP
FUN_G7:   MOV    DPTR,#TABLE7
          LCALL  SUB_FUN
          RET

C_SUP8:   CJNE   A,#'8',C_SUP9
          MOV    DPTR,#TABLE8
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP9
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE8
          LCALL  SUB_SUP
SCOP8:    MOV    DPTR,#TABLE8
          LCALL  SUB_SCOP
FUN_G8:   MOV    DPTR,#TABLE8
          LCALL  SUB_FUN
          RET

C_SUP9:   CJNE   A,#'9',C_SUP10
          MOV    DPTR,#TABLE9
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP10
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE9
          LCALL  SUB_SUP
SCOP9:    MOV    DPTR,#TABLE9
          LCALL  SUB_SCOP
FUN_G9:   MOV    DPTR,#TABLE9
          LCALL  SUB_FUN
          RET
R_SUP1:   LJMP   C_SUP1
C_SUP10:  CJNE   A,#'*',R_SUP1
```

```
          LCALL  SCAN1
          CJNE   A,#'1',R_SUP1
          LCALL  SCAN1
          CJNE   A,#'0',C_SUP11
          MOV    DPTR,#TABLE10
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP11
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE10
          LCALL  SUB_SUP
SCOP10:   MOV    DPTR,#TABLE10
          LCALL  SUB_SCOP
FUN_G10:  MOV    DPTR,#TABLE10
          LCALL  SUB_FUN
          RET
C_SUP11:  CJNE   A,#'1',C_SUP12
          MOV    DPTR,#TABLE11
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP12
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE11
          LCALL  SUB_SUP
SCOP11:   MOV    DPTR,#TABLE11
          LCALL  SUB_SCOP
FUN_G11:  MOV    DPTR,#TABLE11
          LCALL  SUB_FUN
          RET

C_SUP12:  CJNE   A,#'2',C_SUP13
          MOV    DPTR,#TABLE12
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP13
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE12
          LCALL  SUB_SUP
SCOP12:   MOV    DPTR,#TABLE12
          LCALL  SUB_SCOP
FUN_G12:  MOV    DPTR,#TABLE13
          LCALL  SUB_FUN
          RET
C_SUP13:  CJNE   A,#'3',C_SUP14
          MOV    DPTR,#TABLE13
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP14
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE13
          LCALL  SUB_SUP
SCOP13:   MOV    DPTR,#TABLE13
          LCALL  SUB_SCOP
FUN_G13:  MOV    DPTR,#TABLE13
          LCALL  SUB_FUN
          RET

C_SUP14:  CJNE   A,#'4',C_SUP15
          MOV    DPTR,#TABLE14
          INC    DPTR
          MOVX   A,@DPTR
          CJNE   A,#'D',C_SUP15
          LCALL  L_TTDAM
          MOV    DPTR,#TABLE14
          LCALL  SUB_SUP
SCOP14:   MOV    DPTR,#TABLE14
          LCALL  SUB_SCOP
FUN_G14:  MOV    DPTR,#TABLE14
          LCALL  SUB_FUN
          RET
C_SUP15:  CJNE   A,#'5',C_SUP16
```

```
          MOV     DPTR,#TABLE15
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',C_SUP16
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE15
          LCALL   SUB_SUP
SCOP15:   MOV     DPTR,#TABLE15
          LCALL   SUB_SCOP
FUN_G15:  MOV     DPTR,#TABLE15
          LCALL   SUB_FUN
          RET

C_SUP16:  CJNE    A,#'6',JUM_SUP1
          MOV     DPTR,#TABLE16
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',JUM_SUP1
          LCALL   L_TTDAM
          MOV     DPTR,#TABLE16
          LCALL   SUB_SUP
SCOP16:   MOV     DPTR,#TABLE16
          LCALL   SUB_SCOP
FUN_G16:  MOV     DPTR,#TABLE16
          LCALL   SUB_FUN
          RET
JUM_SUP1:LJMP     C_SUP1

SUB_SUP:  INC     DPTR
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',JUM_RET
          LCALL   L_SUP
          LCALL   DELAYC
          RET
SUB_SCOP:INC      DPTR
          INC     DPTR
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',JUM_RET
          LCALL   L_SCOP
          LCALL   DELAYC
          RET
SUB_FUN:  INC     DPTR
          INC     DPTR
          INC     DPTR
          INC     DPTR
          MOVX    A,@DPTR
          CJNE    A,#'D',JUM_RET
          LCALL   L_FUN
          LCALL   DELAYC
JUM_RET:  RET

CK_OTHER:LCALL    SCAN1
          CJNE    A,#'1',CK_2
          LCALL   SUPPLY
          MOV     DPTR,#TABLE1
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE1
          LCALL   CKK
          MOV     DPTR,#TABLE1
          LCALL   SUB_OK
          RET
CK_2:     CJNE    A,#'2',CK_3
          LCALL   SUPPLY
          MOV     DPTR,#TABLE2
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE2
          LCALL   CKK
          MOV     DPTR,#TABLE2
          LCALL   SUB_OK
```

```
        RET
CK_3:   CJNE   A,#'3',CK_4
        LCALL  SUPPLY
        MOV    DPTR,#TABLE3
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE3
        LCALL  CKK
        MOV    DPTR,#TABLE3
        LCALL  SUB_OK
        RET
CK_4:   CJNE   A,#'4',CK_5
        LCALL  SUPPLY
        MOV    DPTR,#TABLE4
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE4
        LCALL  CKK
        MOV    DPTR,#TABLE4
        LCALL  SUB_OK
        RET
CK_5:   CJNE   A,#'5',CK_6
        LCALL  SUPPLY
        MOV    DPTR,#TABLE5
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE5
        LCALL  CKK
        MOV    DPTR,#TABLE5
        LCALL  SUB_OK
        RET
CK_6:   CJNE   A,#'6',CK_7
        LCALL  SUPPLY
        MOV    DPTR,#TABLE6
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE6
        LCALL  CKK
        MOV    DPTR,#TABLE6
        LCALL  SUB_OK
        RET
CK_7:   CJNE   A,#'7',CK_8
        LCALL  SUPPLY
        MOV    DPTR,#TABLE7
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE7
        LCALL  CKK
        MOV    DPTR,#TABLE7
        LCALL  SUB_OK
        RET
CK_8:   CJNE   A,#'8',CK_9
        LCALL  SUPPLY
        MOV    DPTR,#TABLE8
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE8
        LCALL  CKK
        MOV    DPTR,#TABLE8
        LCALL  SUB_OK
        RET
CK_9:   CJNE   A,#'9',CK_10
        LCALL  SUPPLY
        MOV    DPTR,#TABLE9
        LCALL  CK_TOOLS
        MOV    DPTR,#TABLE9
        LCALL  CKK
        MOV    DPTR,#TABLE9
        LCALL  SUB_OK
        RET
R_CK1:  LJMP   CK_OTHER
CK_10:  CJNE   A,#'*',R_CK1
        LCALL  SCAN1
        CJNE   A,#'1',R_CK1
        LCALL  SCAN1
        CJNE   A,#'0',CK_11
        LCALL  SUPPLY
```

```
          MOV     DPTR,#TABLE10
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE10
          LCALL   CKK
          MOV     DPTR,#TABLE10
          LCALL   SUB_OK
          RET
CK_11:    CJNE    A,#'1',CK_12
          LCALL   SUPPLY
          MOV     DPTR,#TABLE11
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE11
          LCALL   CKK
          MOV     DPTR,#TABLE11
          LCALL   SUB_OK
          RET
CK_12:    CJNE    A,#'2',CK_13
          LCALL   SUPPLY
          MOV     DPTR,#TABLE12
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE12
          LCALL   CKK
          MOV     DPTR,#TABLE12
          LCALL   SUB_OK
          RET
CK_13:    CJNE    A,#'3',CK_14
          LCALL   SUPPLY
          MOV     DPTR,#TABLE13
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE13
          LCALL   CKK
          MOV     DPTR,#TABLE13
          LCALL   SUB_OK
          RET
CK_14:    CJNE    A,#'4',CK_15
          LCALL   SUPPLY
          MOV     DPTR,#TABLE14
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE14
          LCALL   CKK
          MOV     DPTR,#TABLE14
          LCALL   SUB_OK
          RET
CK_15:    CJNE    A,#'5',CK_16
          LCALL   SUPPLY
          MOV     DPTR,#TABLE15
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE15
          LCALL   CKK
          MOV     DPTR,#TABLE15
          LCALL   SUB_OK
          RET
CK_16:    CJNE    A,#'6',J_OTHER
          LCALL   SUPPLY
          MOV     DPTR,#TABLE16
          LCALL   CK_TOOLS
          MOV     DPTR,#TABLE16
          LCALL   CKK
          MOV     DPTR,#TABLE16
          LCALL   SUB_OK
          RET
J_OTHER:  LJMP    CK_OTHER

SUB_OK:   INC     DPTR
          MOVX    @DPTR,A
          LCALL   L_OK
          LCALL   DELAY2
          RET
```

```
SUPPLY:  LCALL   L_TOOLS
         LCALL   SCAN1
         CJNE    A,#'1',SCOP
         LCALL   S_SUP
G_SUP:   LCALL   SCAN1
         CJNE    A,#'1',B_SUP
         MOV     49H,#'0'
         MOV     A,#'1'
         RET
B_SUP:   CJNE    A,#'2',SUPPLY
         MOV     49H,#'D'
         MOV     A,#'1'
         RET

SCOP:    CJNE    A,#'2',FUN_GEN
         LCALL   S_SCOP
G_SCOP:  LCALL   SCAN1
         CJNE    A,#'1',B_SCOP
         MOV     4AH,#'0'
         MOV     A,#'2'
         RET
B_SCOP:  CJNE    A,#'2',G_SCOP
         MOV     4AH,#'D'
         MOV     A,#'2'
         RET

FUN_GEN: CJNE    A,#'3',NEX2
         LCALL   S_FUN
G_FUN:   LCALL   SCAN1
         CJNE    A,#'1',B_FUN
         MOV     4BH,#'0'
         MOV     A,#'3'
         RET
B_FUN:   CJNE    A,#'2',G_FUN
         MOV     4BH,#'D'
         MOV     A,#'3'
         RET
NEX2:    LJMP    SUPPLY

CK_TOOLS:INC     DPTR
         INC     DPTR
         CJNE    A,#'1',CK_TOOL2
         CLR     A
         MOV     A,49H
         MOVX    @DPTR,A
         RET
CK_TOOL2:INC     DPTR
         CJNE    A,#'2',CK_TOOL3
         CLR     A
         MOV     A,4AH
         MOVX    @DPTR,A
         RET
CK_TOOL3:INC     DPTR
         CJNE    A,#'3',JUM_TOOL
         CLR     A
         MOV     A,4BH
         MOVX    @DPTR,A
         RET
JUM_TOOL:LCALL   DELAY2
         RET

CKK:     INC     DPTR
         INC     DPTR
         MOVX    A,@DPTR
         CJNE    A,#'D',CKK2
         SJMP    CKKTD
CKK2:    INC     DPTR
         MOVX    A,@DPTR
         CJNE    A,#'D',CKK3
         SJMP    CKKTD
CKK3:    INC     DPTR
```

```
        MOVX    A,@DPTR
        CJNE    A,#'D',CKK0
        SJMP    CKKTD
CKK0:   MOV     A,#'0'
        RET
CKKTD:  MOV     A,#'D'
        RET

        ;*************************************
        ;               TURN OFF TABLE
        ;*************************************

OFF1:   CLR     A
        MOV     DPTR,#TABLE1
        MOVX    A,@DPTR
        CJNE    A,#01H,OFF2
        LCALL   SUB_OFF
        CJNE    A,#0,OFF2
        LCALL   L_OFF
        LCALL   SHOW_1
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE1
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#01H
        MOV     41H,#'0'
        MOV     42H,#'1'
        MOV     DPTR,#RAM1
        LCALL   SAVE
        RET
OFF2:   CLR     A
        MOV     DPTR,#TABLE2
        MOVX    A,@DPTR
        CJNE    A,#02H,OFF3
        LCALL   SUB_OFF
        CJNE    A,#0,OFF3
        LCALL   L_OFF
        LCALL   SHOW_2
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE2
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#02H
        MOV     41H,#'0'
        MOV     42H,#'2'
        MOV     DPTR,#RAM2
        LCALL   SAVE
        RET
OFF3:   CLR     A
        MOV     DPTR,#TABLE3
        MOVX    A,@DPTR
        CJNE    A,#03H,OFF4
        LCALL   SUB_OFF
        CJNE    A,#0,OFF4
        LCALL   L_OFF
        LCALL   SHOW_3
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE3
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#03H
        MOV     41H,#'0'
        MOV     42H,#'3'
        MOV     DPTR,#RAM3
        LCALL   SAVE
        RET
OFF4:   CLR     A
        MOV     DPTR,#TABLE4
```

```
        MOVX    A,@DPTR
        CJNE    A,#04H,OFF5
        LCALL   SUB_OFF
        CJNE    A,#0,OFF5
        LCALL   L_OFF
        LCALL   SHOW_4
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE4
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#04H
        MOV     41H,#'0'
        MOV     42H,#'4'
        MOV     DPTR,#RAM4
        LCALL   SAVE
        RET
OFF5:   CLR     A
        MOV     DPTR,#TABLE5
        MOVX    A,@DPTR
        CJNE    A,#05H,OFF6
        LCALL   SUB_OFF
        CJNE    A,#0,OFF6
        LCALL   L_OFF
        LCALL   SHOW_5
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE5
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#05H
        MOV     41H,#'0'
        MOV     42H,#'5'
        MOV     DPTR,#RAM5
        LCALL   SAVE
        RET
OFF6:   CLR     A
        MOV     DPTR,#TABLE6
        MOVX    A,@DPTR
        CJNE    A,#06H,OFF7
        LCALL   SUB_OFF
        CJNE    A,#0,OFF7
        LCALL   L_OFF
        LCALL   SHOW_6
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE6
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#06H
        MOV     41H,#'0'
        MOV     42H,#'6'
        MOV     DPTR,#RAM6
        LCALL   SAVE
        RET
OFF7:   CLR     A
        MOV     DPTR,#TABLE7
        MOVX    A,@DPTR
        CJNE    A,#07H,OFF8
        LCALL   SUB_OFF
        CJNE    A,#0,OFF8
        LCALL   L_OFF
        LCALL   SHOW_7
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE7
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#07H
        MOV     41H,#'0'
        MOV     42H,#'7'
```

```
        MOV     DPTR,#RAM7
        LCALL   SAVE
        RET
OFF8:   CLR     A
        MOV     DPTR,#TABLE8
        MOVX    A,@DPTR
        CJNE    A,#08H,OFF9
        LCALL   SUB_OFF
        CJNE    A,#0,OFF9
        LCALL   L_OFF
        LCALL   SHOW_8
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE8
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#08H
        MOV     41H,#'0'
        MOV     42H,#'8'
        MOV     DPTR,#RAM8
        LCALL   SAVE
        RET
OFF9:   CLR     A
        MOV     DPTR,#TABLE9
        MOVX    A,@DPTR
        CJNE    A,#09H,OFF10
        LCALL   SUB_OFF
        CJNE    A,#0,OFF10
        LCALL   L_OFF
        LCALL   SHOW_9
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE9
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#09H
        MOV     41H,#'0'
        MOV     42H,#'9'
        MOV     DPTR,#RAM9
        LCALL   SAVE
        RET
OFF10:  CLR     A
        MOV     DPTR,#TABLE10
        MOVX    A,@DPTR
        CJNE    A,#10H,OFF11
        LCALL   SUB_OFF
        CJNE    A,#0,OFF11
        LCALL   L_OFF
        LCALL   SHOW_10
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE10
        MOV     A,#00H
        MOVX    @DPTR,A
        LCALL   SET_12
        MOV     40H,#10H
        MOV     41H,#'1'
        MOV     42H,#'0'
        MOV     DPTR,#RAM10
        LCALL   SAVE
        RET
OFF11:  CLR     A
        MOV     DPTR,#TABLE11
        MOVX    A,@DPTR
        CJNE    A,#11H,OFF12
        LCALL   SUB_OFF
        CJNE    A,#0,OFF12
        LCALL   L_OFF
        LCALL   SHOW_11
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE11
        MOV     A,#00H
```

```
        MOVX   @DPTR,A
        LCALL  SET_12
        MOV    40H,#11H
        MOV    41H,#'1'
        MOV    42H,#'1'
        MOV    DPTR,#RAM11
        LCALL  SAVE
        RET
OFF12:  CLR    A
        MOV    DPTR,#TABLE12
        MOVX   A,@DPTR
        CJNE   A,#12H,OFF13
        LCALL  SUB_OFF
        CJNE   A,#0,OFF13
        LCALL  L_OFF
        LCALL  SHOW_12
        LCALL  DATA_CLOK
        MOV    DPTR,#TABLE12
        MOV    A,#00H
        MOVX   @DPTR,A
        LCALL  SET_12
        MOV    40H,#12H
        MOV    41H,#'1'
        MOV    42H,#'2'
        MOV    DPTR,#RAM12
        LCALL  SAVE
        RET
OFF13:  CLR    A
        MOV    DPTR,#TABLE13
        MOVX   A,@DPTR
        CJNE   A,#13H,OFF14
        LCALL  SUB_OFF
        CJNE   A,#0,OFF14
        LCALL  L_OFF
        LCALL  SHOW_13
        LCALL  DATA_CLOK
        MOV    DPTR,#TABLE13
        MOV    A,#00H
        MOVX   @DPTR,A
        LCALL  SET_12
        MOV    40H,#13H
        MOV    41H,#'1'
        MOV    42H,#'3'
        MOV    DPTR,#RAM13
        LCALL  SAVE
        RET
OFF14:  CLR    A
        MOV    DPTR,#TABLE14
        MOVX   A,@DPTR
        CJNE   A,#14H,OFF15
        LCALL  SUB_OFF
        CJNE   A,#0,OFF15
        LCALL  L_OFF
        LCALL  SHOW_14
        LCALL  DATA_CLOK
        MOV    DPTR,#TABLE14
        MOV    A,#00H
        MOVX   @DPTR,A
        LCALL  SET_12
        MOV    40H,#14H
        MOV    41H,#'1'
        MOV    42H,#'4'
        MOV    DPTR,#RAM14
        LCALL  SAVE
        RET
OFF15:  CLR    A
        MOV    DPTR,#TABLE15
        MOVX   A,@DPTR
        CJNE   A,#15H,OFF16
        LCALL  SUB_OFF
        CJNE   A,#0,OFF16
```

```
          LCALL   L_OFF
          LCALL   SHOW_15
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE15
          MOV     A,#00H
          MOVX    @DPTR,A
          LCALL   SET_12
          MOV     40H,#15H
          MOV     41H,#'1'
          MOV     42H,#'5'
          MOV     DPTR,#RAM15
          LCALL   SAVE
          RET
OFF16:    CLR     A
          MOV     DPTR,#TABLE16
          MOVX    A,@DPTR
          CJNE    A,#16H,SHOW
          LCALL   SUB_OFF
          CJNE    A,#0,SHOW
          LCALL   L_OFF
          LCALL   SHOW_16
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE16
          MOV     A,#00H
          MOVX    @DPTR,A
          LCALL   SET_12
          MOV     40H,#16H
          MOV     41H,#'1'
          MOV     42H,#'6'
          MOV     DPTR,#RAM16
          LCALL   SAVE
          RET

SUB_OFF:  MOV     R1,#5
LOOP_OFF: INC     DPTR
          DJNZ    R1,LOOP_OFF
          MOV     R0,#BAROUT
CK_OFF:   MOV     A,@R0
          CJNE    A,#0DH,CK_OFFA
          MOV     A,#0
          RET
CK_OFFA:  MOV     40H,A
          CLR     A
          MOVX    A,@DPTR
          CJNE    A,40H,JUM_OFF
          INC     DPTR
          INC     R0
          SJMP    CK_OFF
JUM_OFF:  MOV     A,#1
          RET

          ;**************************************
          ;              SHOW TABLE
          ;**************************************

SHOW:     LJMP    CEK
TA1:      LCALL   SELEC
          LCALL   INIT_LCD
          MOV     DPTR,#TABLE1
          MOVX    A,@DPTR
          CJNE    A,#00H,TA2
          MOV     A,#81H
          LCALL   LCDWI
          LCALL   SHOW_1
TA2:      MOV     DPTR,#TABLE2
          MOVX    A,@DPTR
          CJNE    A,#00H,TA3
          MOV     A,#0C1H
          LCALL   LCDWI
          LCALL   SHOW_2
TA3:      MOV     DPTR,#TABLE3
```

```
        MOVX    A,@DPTR
        CJNE    A,#00H,TA4
        MOV     A,#83H
        LCALL   LCDWI
        LCALL   SHOW_3
TA4:    MOV     DPTR,#TABLE4
        MOVX    A,@DPTR
        CJNE    A,#00H,TA5
        MOV     A,#0C3H
        LCALL   LCDWI
        LCALL   SHOW_4
TA5:    MOV     DPTR,#TABLE5
        MOVX    A,@DPTR
        CJNE    A,#00H,TA6
        MOV     A,#85H
        LCALL   LCDWI
        LCALL   SHOW_5
TA6:    MOV     DPTR,#TABLE6
        MOVX    A,@DPTR
        CJNE    A,#00H,TA7
        MOV     A,#0C5H
        LCALL   LCDWI
        LCALL   SHOW_6
TA7:    MOV     DPTR,#TABLE7
        MOVX    A,@DPTR
        CJNE    A,#00H,TA8
        MOV     A,#87H
        LCALL   LCDWI
        LCALL   SHOW_7
TA8:    MOV     DPTR,#TABLE8
        MOVX    A,@DPTR
        CJNE    A,#00H,TA9
        MOV     A,#0C7H
        LCALL   LCDWI
        LCALL   SHOW_8
TA9:    MOV     DPTR,#TABLE9
        MOVX    A,@DPTR
        CJNE    A,#00H,TA10
        MOV     A,#8AH
        LCALL   LCDWI
        LCALL   SHOW_9
TA10:   MOV     DPTR,#TABLE10
        MOVX    A,@DPTR
        CJNE    A,#00H,TA11
        MOV     A,#0C9H
        LCALL   LCDWI
        LCALL   SHOW_10
TA11:   MOV     DPTR,#TABLE11
        MOVX    A,@DPTR
        CJNE    A,#00H,TA12
        MOV     A,#8CH
        LCALL   LCDWI
        LCALL   SHOW_11
TA12:   MOV     DPTR,#TABLE12
        MOVX    A,@DPTR
        CJNE    A,#00H,TA13
        MOV     A,#0CCH
        LCALL   LCDWI
        LCALL   SHOW_12
TA13:   MOV     DPTR,#TABLE13
        MOVX    A,@DPTR
        CJNE    A,#00H,TA14
        MOV     A,#8FH
        LCALL   LCDWI
        LCALL   SHOW_13
TA14:   MOV     DPTR,#TABLE14
        MOVX    A,@DPTR
        CJNE    A,#00H,TA15
        MOV     A,#0CFH
        LCALL   LCDWI
        LCALL   SHOW_14
```

```
TA15:     MOV     DPTR,#TABLE15
          MOVX    A,@DPTR
          CJNE    A,#00H,TA16
          MOV     A,#92H
          LCALL   LCDWI
          LCALL   SHOW_15
TA16:     MOV     DPTR,#TABLE16
          MOVX    A,@DPTR
          CJNE    A,#00H,JUM_ON1
          MOV     A,#0D2H
          LCALL   LCDWI
          LCALL   SHOW_16

JUM_ON1:  LJMP    ON1

SHOW_1:   MOV     A,#'1'
          LCALL   LCDWD
          RET
SHOW_2:   MOV     A,#'2'
          LCALL   LCDWD
          RET
SHOW_3:   MOV     A,#'3'
          LCALL   LCDWD
          RET
SHOW_4:   MOV     A,#'4'
          LCALL   LCDWD
          RET
SHOW_5:   MOV     A,#'5'
          LCALL   LCDWD
          RET
SHOW_6:   MOV     A,#'6'
          LCALL   LCDWD
          RET
SHOW_7:   MOV     A,#'7'
          LCALL   LCDWD
          RET
SHOW_8:   MOV     A,#'8'
          LCALL   LCDWD
          RET
SHOW_9:   MOV     A,#'9'
          LCALL   LCDWD
          RET
SHOW_10:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'0'
          LCALL   LCDWD
          RET
SHOW_11:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'1'
          LCALL   LCDWD
          RET
SHOW_12:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'2'
          LCALL   LCDWD
          RET
SHOW_13:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'3'
          LCALL   LCDWD
          RET
SHOW_14:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'4'
          LCALL   LCDWD
          RET
SHOW_15:  MOV     A,#'1'
          LCALL   LCDWD
          MOV     A,#'5'
          LCALL   LCDWD
```

```
        RET
SHOW_16: MOV    A,#'1'
        LCALL   LCDWD
        MOV     A,#'6'
        LCALL   LCDWD
        RET

CEK:    CLR     A
        MOV     DPTR,#TABLE1
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK2
        LJMP    TA1
CEK2:   CLR     A
        MOV     DPTR,#TABLE2
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK3
        LJMP    TA1
CEK3:   CLR     A
        MOV     DPTR,#TABLE3
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK4
        LJMP    TA1
CEK4:   CLR     A
        MOV     DPTR,#TABLE4
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK5
        LJMP    TA1
CEK5:   CLR     A
        MOV     DPTR,#TABLE5
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK6
        LJMP    TA1
CEK6:   CLR     A
        MOV     DPTR,#TABLE6
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK7
        LJMP    TA1
CEK7:   CLR     A
        MOV     DPTR,#TABLE7
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK8
        LJMP    TA1
CEK8:   CLR     A
        MOV     DPTR,#TABLE8
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK9
        LJMP    TA1
CEK9:   CLR     A
        MOV     DPTR,#TABLE9
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK10
        LJMP    TA1
CEK10:  CLR     A
        MOV     DPTR,#TABLE10
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK11
        LJMP    TA1
CEK11:  CLR     A
        MOV     DPTR,#TABLE11
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK12
        LJMP    TA1
CEK12:  CLR     A
        MOV     DPTR,#TABLE12
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK13
        LJMP    TA1
CEK13:  CLR     A
        MOV     DPTR,#TABLE13
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK14
```

```
        LJMP    TA1
CEK14:  CLR     A
        MOV     DPTR,#TABLE14
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK15
        LJMP    TA1
CEK15:  CLR     A
        MOV     DPTR,#TABLE15
        MOVX    A,@DPTR
        CJNE    A,#00H,CEK16
        LJMP    TA1
CEK16:  CLR     A
        MOV     DPTR,#TABLE16
        MOVX    A,@DPTR
        CJNE    A,#00H,JUM_FULL
        LJMP    TA1
JUM_FULL:LCALL  FULL
        RET

        ;*************************************
        ;              TURN ON TABLE
        ;*************************************
ON1:    LCALL   SCAN1
        CJNE    A,#'1',ON2
        CLR     A
        MOV     DPTR,#TABLE1
        MOVX    A,@DPTR
        CJNE    A,#00H,ON1
        LCALL   L_ON
        LCALL   SHOW_1
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE1
        MOV     A,#01H
        MOVX    @DPTR,A
        MOV     40H,#01H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON2:    CJNE    A,#'2',ON3
        CLR     A
        MOV     DPTR,#TABLE2
        MOVX    A,@DPTR
        CJNE    A,#00H,ON3
        LCALL   L_ON
        LCALL   SHOW_2
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE2
        MOV     A,#02H
        MOVX    @DPTR,A
        MOV     40H,#02H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON3:    CJNE    A,#'3',ON4
        CLR     A
        MOV     DPTR,#TABLE3
        MOVX    A,@DPTR
        CJNE    A,#00H,ON4
        LCALL   L_ON
        LCALL   SHOW_3
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE3
        MOV     A,#03H
        MOVX    @DPTR,A
        MOV     40H,#03H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON4:    CJNE    A,#'4',ON5
        CLR     A
        MOV     DPTR,#TABLE4
```

```
        MOVX    A,@DPTR
        CJNE    A,#00H,ON5
        LCALL   L_ON
        LCALL   SHOW_4
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE4
        MOV     A,#04H
        MOVX    @DPTR,A
        MOV     40H,#04H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON5:    CJNE    A,#'5',ON6
        CLR     A
        MOV     DPTR,#TABLE5
        MOVX    A,@DPTR
        CJNE    A,#00H,ON6
        LCALL   L_ON
        LCALL   SHOW_5
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE5
        MOV     A,#05H
        MOVX    @DPTR,A
        MOV     40H,#05H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON6:    CJNE    A,#'6',ON7
        CLR     A
        MOV     DPTR,#TABLE6
        MOVX    A,@DPTR
        CJNE    A,#00H,ON7
        LCALL   L_ON
        LCALL   SHOW_6
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE6
        MOV     A,#06H
        MOVX    @DPTR,A
        MOV     40H,#06H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON7:    CJNE    A,#'7',ON8
        CLR     A
        MOV     DPTR,#TABLE7
        MOVX    A,@DPTR
        CJNE    A,#00H,ON8
        LCALL   L_ON
        LCALL   SHOW_7
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE7
        MOV     A,#07H
        MOVX    @DPTR,A
        MOV     40H,#07H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON8:    CJNE    A,#'8',ON9
        CLR     A
        MOV     DPTR,#TABLE8
        MOVX    A,@DPTR
        CJNE    A,#00H,ON9
        LCALL   L_ON
        LCALL   SHOW_8
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE8
        MOV     A,#08H
        MOVX    @DPTR,A
        MOV     40H,#08H
        LCALL   SUB_ON
        LCALL   DELAYC
```

```
        RET
ON9:    CJNE    A,#'9',ON10
        CLR     A
        MOV     DPTR,#TABLE9
        MOVX    A,@DPTR
        CJNE    A,#00H,ON10
        LCALL   L_ON
        LCALL   SHOW_9
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE9
        MOV     A,#09H
        MOVX    @DPTR,A
        MOV     40H,#09H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
J_ON1:  LJMP    ON1
ON10:   CJNE    A,#'*',J_ON1
        LCALL   SCAN1
        CJNE    A,#'1',J_ON1
        LCALL   SCAN1
        CJNE    A,#'0',ON11
        CLR     A
        MOV     DPTR,#TABLE10
        MOVX    A,@DPTR
        CJNE    A,#00H,ON11
        LCALL   L_ON
        LCALL   SHOW_10
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE10
        MOV     A,#10H
        MOVX    @DPTR,A
        MOV     40H,#10H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON11:   CJNE    A,#'1',ON12
        CLR     A
        MOV     DPTR,#TABLE11
        MOVX    A,@DPTR
        CJNE    A,#00H,ON12
        LCALL   L_ON
        LCALL   SHOW_11
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE11
        MOV     A,#11H
        MOVX    @DPTR,A
        MOV     40H,#11H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON12:   CJNE    A,#'2',ON13
        CLR     A
        MOV     DPTR,#TABLE12
        MOVX    A,@DPTR
        CJNE    A,#00H,ON13
        LCALL   L_ON
        LCALL   SHOW_12
        LCALL   DATA_CLOK
        MOV     DPTR,#TABLE12
        MOV     A,#12H
        MOVX    @DPTR,A
        MOV     40H,#12H
        LCALL   SUB_ON
        LCALL   DELAYC
        RET
ON13:   CJNE    A,#'3',ON14
        CLR     A
        MOV     DPTR,#TABLE13
        MOVX    A,@DPTR
        CJNE    A,#00H,ON14
```

```
          LCALL   L_ON
          LCALL   SHOW_13
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE13
          MOV     A,#13H
          MOVX    @DPTR,A
          MOV     40H,#13H
          LCALL   SUB_ON
          LCALL   DELAYC
          RET
ON14:     CJNE    A,#'4',ON15
          CLR     A
          MOV     DPTR,#TABLE14
          MOVX    A,@DPTR
          CJNE    A,#00H,ON15
          LCALL   L_ON
          LCALL   SHOW_14
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE14
          MOV     A,#14H
          MOVX    @DPTR,A
          MOV     40H,#14H
          LCALL   SUB_ON
          LCALL   DELAYC
          RET
ON15:     CJNE    A,#'5',ON16
          CLR     A
          MOV     DPTR,#TABLE15
          MOVX    A,@DPTR
          CJNE    A,#00H,ON16
          LCALL   L_ON
          LCALL   SHOW_15
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE15
          MOV     A,#15H
          MOVX    @DPTR,A
          MOV     40H,#15H
          LCALL   SUB_ON
          LCALL   DELAYC
          RET
ON16:     CJNE    A,#'6',NEX_ON1
          CLR     A
          MOV     DPTR,#TABLE16
          MOVX    A,@DPTR
          CJNE    A,#00H,NEX_ON1
          LCALL   L_ON
          LCALL   SHOW_16
          LCALL   DATA_CLOK
          MOV     DPTR,#TABLE16
          MOV     A,#16H
          MOVX    @DPTR,A
          MOV     40H,#16H
          LCALL   SUB_ON
          LCALL   DELAYC
          RET
NEX_ON1:  LJMP    ON1

SUB_ON:   LCALL   GO_RELAY
          MOV     A,40H
          LCALL   SBYTE
          MOV     R1,#5
LOOP_ON:  INC     DPTR
          DJNZ    R1,LOOP_ON

          ;**************************************
          ;              SAVE TABLE TURN ON
          ;**************************************
SAVE_ON:  MOV     R0,#BAROUT
N1:       CLR     A
          MOV     A,@R0
          CJNE    A,#0DH,N2
```

```
          LCALL   CLOK_ON
          RET
N2:       MOVX    @DPTR,A
          INC     DPTR
          INC     R0
          SJMP    N1
CLOK_ON:  MOV     A,HOUR2
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,HOUR1
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,#':'
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,MIN2
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,MIN1
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,#':'
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,SEC2
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,SEC1
          MOVX    @DPTR,A
          RET

         ;****************************************
         ;                SAVE    DATA
         ;****************************************
SAVE:     LCALL   GO_RELAY
          MOV     A,40H
          LCALL   SBYTE
          MOVX    A,@DPTR
          CJNE    A,#'*',SAVE_1
          MOV     R3,#40
SCAN_RAM: INC     DPTR
          DJNZ    R3,SCAN_RAM
          MOVX    A,@DPTR
          CJNE    A,#'*',SAVE_1
          SJMP    SCAN_RAM

SAVE_1:   MOV     A,#'*'
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,41H
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,42H
          MOVX    @DPTR,A
          INC     DPTR
SAVE_OFF: MOV     R0,#BAROUT
F1:       CLR     A
          MOV     A,@R0
          CJNE    A,#0DH,F2
          SJMP    CLOK_IN
F2:       MOVX    @DPTR,A
          INC     DPTR
          INC     R0
          SJMP    F1
CLOK_IN:  MOV     A,48H
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,49H
          MOVX    @DPTR,A
          INC     DPTR
          MOV     A,#':'
```

```
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,4BH
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,4CH
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#':'
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,4EH
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,4FH
          MOVX   @DPTR,A
          INC    DPTR
CLOK_OUT:MOV     A,HOUR2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,HOUR1
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#':'
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,MIN2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,MIN1
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#':'
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,SEC2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,SEC1
          MOVX   @DPTR,A
          INC    DPTR
DAY_OUT:  MOV    A,DAY1
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#'/'
          MOVX   @DPTR,A
          INC    DPTR
YEAR_OUT:MOV     A,DATE2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,DATE1
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#'/'
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,MONTH2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,MONTH1
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,#'/'
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,YEAR2
          MOVX   @DPTR,A
          INC    DPTR
          MOV    A,YEAR1
          MOVX   @DPTR,A
SAVE_TAB:INC     DPTR
```

```
        MOV     A,45H
        CJNE    A,#'D',GOOD1
RAM_SUP: MOVX   @DPTR,A
        INC     DPTR
        MOV     A,46H
        CJNE    A,#'D',GOOD2
RAM_SCOP:MOVX   @DPTR,A
        INC     DPTR
        MOV     A,47H
        CJNE    A,#'D',GOOD3
RAM_FUN: MOVX   @DPTR,A
        LCALL   DELAYC
        RET
GOOD1:  LCALL   GOOD
        SJMP    RAM_SUP
GOOD2:  LCALL   GOOD
        SJMP    RAM_SCOP
GOOD3:  LCALL   GOOD
        SJMP    RAM_FUN
GOOD:   MOV     A,#'G'
        RET
        ;*************************************
        ;              GO SERAIL PORT
        ;*************************************
SBYTE:  JNB     TI,$
        CLR     TI
        MOV     SBUF,A
        RET

GO_RELAY:MOV    P3.4,#1
        MOV     A,#'R'
        LCALL   SBYTE
        RET
GO_COM1: MOV    P3.4,#1
        MOV     A,#'C'
        LCALL   SBYTE
        RET


SET_TOFF:MOV    A,#8BH
        LCALL   LCDWI
        RET
SHOW_OFF:MOVX   A,@DPTR
        LCALL   LCDWD
        MOV     A,#0C4H
        LCALL   LCDWI
        MOV     R0,#BAROUT
OFF:    MOV     A,@R0
        CJNE    A,#0DH,OFF_A
        LCALL   DELAYC
        RET
OFF_A:  LCALL   LCDWD
        INC     R0
        SJMP    OFF

DATA_CLOK:LCALL   READ                  ; READ DS1202
        LCALL   DISP_T
        LCALL   DISP_D
        LCALL   DISP_Y
        RET
        ;*************************************
        ;            LOAD DATA TO COMPUTER
        ;*************************************
LOAD_DATA:      LCALL   SHOW_LOAD
                MOV     A,#'1'
                LCALL   LCDWD
                MOV     DPTR,#RAM1
                LCALL   SUB_LOAD
                LCALL   SHOW_LOAD
                MOV     A,#'2'
                LCALL   LCDWD
```

```
MOV     DPTR,#RAM2
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'3'
LCALL   LCDWD
MOV     DPTR,#RAM3
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'4'
LCALL   LCDWD
MOV     DPTR,#RAM4
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'5'
LCALL   LCDWD
MOV     DPTR,#RAM5
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'6'
LCALL   LCDWD
MOV     DPTR,#RAM6
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'7'
LCALL   LCDWD
MOV     DPTR,#RAM7
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'8'
LCALL   LCDWD
MOV     DPTR,#RAM8
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'9'
LCALL   LCDWD
MOV     DPTR,#RAM9
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'1'
LCALL   LCDWD
MOV     A,#'0'
LCALL   LCDWD
MOV     DPTR,#RAM10
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'1'
LCALL   LCDWD
MOV     A,#'1'
LCALL   LCDWD
MOV     DPTR,#RAM11
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'1'
LCALL   LCDWD
MOV     A,#'2'
LCALL   LCDWD
MOV     DPTR,#RAM12
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'1'
LCALL   LCDWD
MOV     A,#'3'
LCALL   LCDWD
MOV     DPTR,#RAM13
LCALL   SUB_LOAD
LCALL   SHOW_LOAD
MOV     A,#'1'
LCALL   LCDWD
MOV     A,#'4'
LCALL   LCDWD
MOV     DPTR,#RAM14
```

```
                LCALL   SUB_LOAD
                LCALL   SHOW_LOAD
                MOV     A,#'1'
                LCALL   LCDWD
                MOV     A,#'5'
                LCALL   LCDWD
                MOV     DPTR,#RAM15
                LCALL   SUB_LOAD
                LCALL   SHOW_LOAD
                MOV     A,#'1'
                LCALL   LCDWD
                MOV     A,#'6'
                LCALL   LCDWD
                MOV     DPTR,#RAM16
                LCALL   SUB_LOAD
                LCALL   RESET_RAM
                LCALL   UBEEP_OK
                LCALL   L_OK
                RET
SHOW_LOAD:      LCALL   L_LOAD
                MOV     A,#0CCH
                LCALL   LCDWI
                RET

SUB_LOAD:       MOVX    A,@DPTR
                CJNE    A,#'*',NEX_GO
                MOV     R3,#40
LOAD_GO:        MOVX    A,@DPTR
                LCALL   SBYTE
                INC     DPTR
                DJNZ    R3,LOAD_GO
                SJMP    SUB_LOAD
NEX_GO:         RET
                ;*********************************
                ;              RESET RAM
                ;*********************************

RESET_RAM:      MOV     DPTR,#RAM1
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM2
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM3
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM4
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM5
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM6
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM7
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM8
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM9
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM10
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM11
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
```

```
                MOV     DPTR,#RAM12
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM13
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM14
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM15
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                MOV     DPTR,#RAM16
                MOV     A,#0FFFFH
                MOVX    @DPTR,A
                RET

                ;*********************************
                ;          MEMORY TO RAM
                ;*********************************
SET_12:  INC    DPTR
         MOVX   A,@DPTR
         MOV    44H,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    45H,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    46H,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    47H,A
         MOV    R4,#8
J_SET12: INC    DPTR
         DJNZ   R4,J_SET12
         INC    DPTR
SET_8:   MOVX   A,@DPTR
         MOV    48H,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    49H,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    4AH,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    4BH,A
SET_4:   INC    DPTR
         MOVX   A,@DPTR
         MOV    4CH,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    4DH,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    4EH,A
         INC    DPTR
         MOVX   A,@DPTR
         MOV    4FH,A
         RET


         ;*************************************
         ;              SHOW LCD
         ;*************************************

L_HELLO:        LCALL  INIT
                MOV     DPTR,#L_HEL
                LCALL   LCDLD
                MOV     R2,#06H
                LCALL   DELAYL
```

```
                RET

L_HEL:          DB      "******  HELLO  *****"
                DB      "### Bacode pleas ###"
LCD1:           LCALL  INIT_LCD
                MOV     DPTR,#LCDT1
                LCALL   LCDLD
                LCALL   DELBC
                RET

LCDT1:          DB      "ID :                 "
                DB      "PASSWORD :           "
LCD2:     LCALL  INIT
          MOV    DPTR,#PAGE2
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
PAGE2:    DB     "      Funtion        "
          DB     "1.Use 2.Other 3.Exit"
LCD3:     LCALL  INIT
          MOV    DPTR,#PAGE3
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
PAGE3:    DB     "1.Check tool damaged"
          DB     "2.Inform damaged     "
LCD4:     LCALL  INIT
          MOV    DPTR,#PAGE4
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
PAGE4:    DB     "      Table          "
          DB     "Id :                 "
LCD5:     LCALL  INIT
          MOV    DPTR,#PAGE5
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
PAGE5:    DB     " **** Load sure ****"
          DB     " 1.Yes         2.No "
L_LOAD:   LCALL  INIT
          MOV    DPTR,#PAGE6
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
PAGE6:    DB     "**** Load Data **** "
          DB     "      Table          "
L_TOOLS:  LCALL  INIT
          MOV    DPTR,#LTOOLS
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
LTOOLS:   DB     "   1.Power supply    "
          DB     "2.Scop      3.Fun_Gen"
S_SUP:    LCALL  INIT
          MOV    DPTR,#SSUP
          LCALL  LCDLD
          MOV    R2,#10H
          LCALL  DELBC
          RET
SSUP:     DB     "1.Good supply        "
          DB     "2.No use supply      "
S_SCOP:    LCALL  INIT
          MOV    DPTR,#SSCOP
          LCALL  LCDLD
```

```
          MOV     R2,#10H
          LCALL   DELBC
          RET
SSCOP:    DB      "1.Good ocilloscop   "
          DB      "2.No use ocilloscop "
S_FUN:    LCALL   INIT
          MOV     DPTR,#SFUN
          LCALL   LCDLD
          MOV     R2,#10H
          LCALL   DELBC
          RET
SFUN:     DB      "1.Good funtion gen  "
          DB      "2.No use funtion gen"
L_OK:     LCALL   INIT
          MOV     DPTR,#LOK
          LCALL   LCDLD
          MOV     R2,#06H
          LCALL   DELAYL
          RET
LOK:      DB      "*******  OK  *******"
          DB      "                    "
SELEC:    LCALL   INIT
          MOV     DPTR,#SEL1
          LCALL   LCDLD
          MOV     R2,#07H
          LCALL   DELAYL
          RET
SEL1:     DB      "*** Selec table ***"
          DB      "                    "
L_TDAM:   LCALL   INIT_LCD
          MOV     DPTR,#LTDAM
          LCALL   LCDLD
          MOV     R2,#10H
          LCALL   DELAYL
          RET
LTDAM:    DB      "## Table dam aegd ##"
          DB      "                    "
L_TTDAM:  LCALL   INIT
          MOV     DPTR,#LTTDAM
          LCALL   LCDLD
          MOV     R2,#3H
          LCALL   DELBC
          RET
LTTDAM:   DB     "## Tools dam aegd ##"
          DB     "                    "
NO_PASS:  LCALL   INIT
          MOV     DPTR,#NOPASS
          LCALL   LCDLD
          LCALL   DELAYK
          RET
NOPASS:   DB      "##  No Password  ## "
          DB      "                    "
L_ON:     LCALL   INIT
          MOV     DPTR,#LON
          LCALL   LCDLD
          LCALL   DELBC
          MOV     A,#08EH
          LCALL   LCDWI
          RET
LON:      DB      "***  On table    ***"
          DB      "                    "
L_OFF:    LCALL   INIT
          MOV     DPTR,#LOFF
          LCALL   LCDLD
          LCALL   DELBC
          MOV     A,#08EH
          LCALL   LCDWI
          RET
LOFF:     DB      "*** Off table    ***"
          DB      "                    "
FULL:     LCALL   INIT
```

```
        MOV     DPTR,#FULL1
        LCALL   LCDLD
        MOV     R2,#10H
        LCALL   DELAYC
        RET
FULL1:  DB      "######  FULL  ######"
        DB      "                    "
NO_DAM: LCALL   INIT
        MOV     DPTR,#NODAM
        LCALL   LCDLD
        MOV     R2,#08H
        LCALL   DELAYL
        RET
NODAM:  DB      "    No Dam aegd     "
        DB      "                    "
L_ERROR: LCALL  INIT
        MOV     DPTR,#LERROR
        LCALL   LCDLD
        MOV     R2,#07H
        LCALL   DELAYL
        RET
LERROR: DB      "        ERROR       "
        DB      "                    "

L_SUP:  MOV      A,#0C0H
        LCALL   LCDWI
        MOV      DPTR,#TAB7
        LCALL    LCDLD1
        RET
L_SCOP: MOV      A,#0C0H
        LCALL    LCDWI
        MOV      DPTR,#TAB8
        LCALL    LCDLD1
        RET
L_FUN:  MOV      A,#0C0H
        LCALL    LCDWI
        MOV      DPTR,#TAB9
        LCALL    LCDLD20
        RET

TAB7:   DB       '1.Power supply      '
TAB8:   DB       '2.OscilloscopScop   '
TAB9:   DB       '3.Funtion generater '
DELBC:  MOV     R6,#45H
DELBC1: MOV     R7,#7FH
        DJNZ    R7,$
        DJNZ    R6,DELBC1
        RET

        ;*************************************
        ;                SCANKEY
        ;*************************************

   SCAN1:       MOV       DPTR,#PORTC
                MOV       A,#0FEH
                MOVX      @DPTR,A
        KEY1:   MOVX      A,@DPTR
                CJNE      A,#0EEH,KEY4
                MOV       A,#'1'
                LCALL     UBEEP
                LCALL     DELAYK
                RET
        KEY4:   CJNE      A,#0DEH,KEY7
                MOV       A,#'4'
                LCALL     UBEEP
                LCALL     DELAYK
                RET
        KEY7:   CJNE      A,#0BEH,KEY10
                MOV       A,#'7'
                LCALL     UBEEP
                LCALL     DELAYK
```

```
                RET
          KEY10: CJNE     A,#07EH,SCAN2
                 MOV      A,#'*'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
         SCAN2:  MOV      A,#0FDH
                 MOVX     @DPTR,A
          KEY2:  MOVX     A,@DPTR
                 CJNE     A,#0EDH,KEY5
                 MOV      A,#'2'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
          KEY5:  CJNE     A,#0DDH,KEY8
                 MOV      A,#'5'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
          KEY8:  CJNE     A,#0BDH,KEY0
                 MOV      A,#'8'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
          KEY0:  CJNE     A,#07DH,SCAN3
                 MOV      A,#'0'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
         SCAN3:  MOV      A,#0FBH
                 MOVX     @DPTR,A
          KEY3:  MOVX     A,@DPTR
                 CJNE     A,#0EBH,KEY6
                 MOV      A,#'3'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
          KEY6:  CJNE     A,#0DBH,KEY9
                 MOV      A,#'6'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
          KEY9:  CJNE     A,#0BBH,KEY12
                 MOV      A,#'9'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
         KEY12:  CJNE     A,#07BH,NEX_SCAN1
                 MOV      A,#'#'
                 LCALL    UBEEP
                 LCALL    DELAYK
                 RET
     NEX_SCAN1:  LJMP     SCAN1

        ;**************************************
        ;                SOUND SUB
        ;**************************************

UBEEP:   MOV     4FH,A
         MOV     R2,#4FH
         MOV     R3,#0FH
         CALL    SOUND
         MOV     A,4FH
         RET
UBEEP_OK:MOV     R2,#60H
         MOV     R3,#0
         CALL    SOUND
         RET


SOUND:   MOV     R5,#0
```

```
        MOV     R4,#80H
SOUND1: CALL    SOUNDS
        CJNE    R5,#1,SOUND1
        RET
SOUNDS: MOV     DPTR,#PORTA         :OUT1
        MOVX    A,@DPTR
        SETB    ACC.0
        MOVX    @DPTR,A
        CALL    SOUNDX
        MOV     DPTR,#PORTA         :OUT0
        MOVX    A,@DPTR
        CLR     ACC.0
        MOVX    @DPTR,A
        CALL    SOUNDX
        RET
SOUNDX: MOV     A,R2
SOUNDX1:CALL    SOUNDY
        DEC     A
        JNZ     SOUNDX1
        RET
SOUNDY: DJNZ    R4,SOUNDY1
        MOV     R4,#80H
        DJNZ    R3,SOUNDY1
        MOV     R5,#1
SOUNDY1:RET

        ;*************************************
        ;               BARCV SUB
        ;*************************************

BARCV:  MOV     DPTR,#BARBUF
        MOVX    A,@DPTR
        MOV     B,#2
        MUL     AB
        MOV     B,A
        MOV     DPTR,#BARBUF+1
        MOVX    A,@DPTR
        CJNE    A,B,$+3
        JC      BARRV
        MOV     DPTR,#BARBUF
        MOV     R0,#BAROUT
BARCV1: MOV     R2,#9
        MOV     R3,#0
BARCV2: MOVX    A,@DPTR
        CJNE    A,#0FFH,BARCV3
        SJMP    BARCVE
BARCV3: CJNE    A,B,$+3
        CPL     C
        MOV     A,R3
        RLC     A
        MOV     R3,A
        INC     DPTR
        DJNZ    R2,BARCV2
        INC     DPTR
        CALL    BARTB
        JC      BARCVE
        CJNE    A,#'*',BARCV4
        CJNE    R0,#BAROUT,BARCV35
        SJMP    BARCV1
BARCV35:MOV     @R0,#0DH
        CLR     C
        RET
BARCV4: MOV     @R0,A
        INC     R0
        SJMP    BARCV1
BARCVE: SETB    C
        RET
BARRV:  MOV     DPTR,#BARBUF
        MOV     R0,#BARREV
BARRV1: MOV     R2,#9
        MOV     R3,#0
```

```
BARRV2:  MOVX   A,@DPTR
         CJNE   A,#0FFH,BARRV3
         SJMP   BARRVE
BARRV3:  CJNE   A,B,$+3
         CPL    C
         MOV    A,R3
         RRC    A
         MOV    R3,A
         INC    DPTR
         DJNZ   R2,BARRV2
         INC    DPTR
         MOV    A,R3
         RLC    A
         MOV    R3,A
         CALL   BARTB
         JC     BARRVE
         CJNE   A,#'*',BARRV4
         CJNE   R0,#BARREV,BARRV35
         SJMP   BARRV1
BARRV35: MOV    R1,#BAROUT
         DEC    R0
BARRV37: CJNE   R0,#BARREV-1,BARRV38
         MOV    @R1,#0DH
         CLR    C
         RET
BARRV38: MOV    A,@R0
         MOV    @R1,A
         DEC    R0
         INC    R1
         SJMP   BARRV37
BARRV4:  MOV    @R0,A
         INC    R0
         SJMP   BARRV1
BARRVE:  SETB   C
         RET

         ;**************************************
         ;          BARTB SUB
         ;**************************************

BARTB:   PUSH   DPH
         PUSH   DPL
         MOV    DPTR,#BARTBX
         JC     BARTB1
         MOV    DPTR,#BARTBY
BARTB1:  CLR    A
         MOVC   A,@A+DPTR
         CJNE   A,#0FFH,BARTB2
         SJMP   BARTBE
BARTB2:  CLR    C
         SUBB   A,R3
         JZ     BARTB5
         INC    DPTR
         INC    DPTR
         SJMP   BARTB1
BARTBE:  SETB   C
         POP    DPL
         POP    DPH
         RET
BARTB5:  INC    DPTR
         CLR    A
         MOVC   A,@A+DPTR
         CLR    C
         POP    DPL
         POP    DPH
         RET             ;EXIT OK
;-------------------------------------------------------------------------

BARTBX:  DB     00100001B,'1'     ;BIT 9 = 1 TABLE

         DB     01100000B,'3'
```

```
        DB      00110000B,'5'
        DB      00100100B,'8'
        DB      00001001B,'A'
        DB      01001000B,'C'
        DB      00011000B,'E'
        DB      00001100B,'H'
        DB      00000011B,'K'
        DB      01000010B,'M'
        DB      00010010B,'O'
        DB      00000110B,'R'

        DB      10000001B,'U'
        DB      11000000B,'W'
        DB      10010000B,'Y'
        DB      10000100B,'.'
        DB      0FFH

BARTBY: DB      00110100B,'0'    ;BIT 9 = 0 TABLE
        DB      01100001B,'2'
        DB      00110001B,'4'
        DB      01110000B,'6'
        DB      00100101B,'7'
        DB      01100100B,'9'
        DB      01001001B,'B'
        DB      00011001B,'D'
        DB      01011000B,'F'
        DB      00001101B,'G'
        DB      01001100B,'I'
        DB      00011100B,'J'
        DB      01000011B,'L'
        DB      00010011B,'N'
        DB      01010010B,'P'
        DB      00000111B,'Q'
        DB      01000110B,'S'
        DB      00010110B,'T'
        DB      11000001B,'V'
```

```
        DB      10010001B,'X'
        DB      11010000B,'Z'
        DB      10000101B,'-'
        DB      11000100B,' '
        DB      10010100B,'*'
        DB      10101000B,'$'
        DB      10100010B,'/'
        DB      10001010B,'+'
        DB      00101010B,'%'
        DB      0FFH
        ;*************************************
        ;               BARIN SUB
        ;*************************************
BARIN:  MOV     DPTR,#BARBUF
        JB      BARBIT,$
        JNB     BARBIT,$
BARIN1: CLR     A
BARIN2: CALL    BARDL
        INC     A
        JZ      BARIN5
        JB      BARBIT,BARIN2
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
BARIN3: CALL    BARDL
        INC     A
        JZ      BARIN5
        JNB     BARBIT,BARIN3
        MOVX    @DPTR,A
        INC     DPTR
        SJMP    BARIN1
BARIN5: MOV     A,#0FFH
        MOVX    @DPTR,A
        RET
BARDL:  MOV     R2,#8
        DJNZ    R2,$
        RET

;---------- LCD INT ----------------
;REG=A,R2
INIT_LCD:       MOV     R2,#2
                LCALL   DLLCD
                MOV     A,#3FH
                LCALL   LCDWI
                MOV     A,#0FH
                LCALL   LCDWI
                MOV     A,#01H
                LCALL   LCDWI
                MOV     R2,#2
                LCALL   DLLCD
                RET

;********** LCD DELAY SUB **********
;DELAY SUBROUTINE

;IN=R2
;REG=R2,R3,R4
DLLCD:          MOV     R3,#45H
DLLCD1:         MOV     R4,#4FH
                DJNZ    R4,$
```

```
                DJNZ    R3,DLLCD1
                DJNZ    R2,DLLCD
                RET

;************** LCDDIS **************
LCDLD20:
                MOV     R3,#20
                SJMP    AGAIN

LCDLD:          MOV     A,#80H
                LCALL   LCDWI
                LCALL   LCDLD1
                MOV     A,#0C0H
                LCALL   LCDWI
                LCALL   LCDLD1
                LCALL   DELBC
                RET
LCDLD1:         MOV     R3,#20
AGAIN:          CLR     A
                MOVC    A,@A+DPTR
                PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#WRITEDATA
                MOVX    @DPTR,A
                LCALL   LCDWI
                POP     DPH
                POP     DPL
                INC     DPTR
                DJNZ    R3,AGAIN
                LCALL   DELBC
                RET
;****************** LCDWI ******************
LCDWI:          PUSH    DPH
                PUSH    DPL
                MOV     DPTR,#COMMAND
                MOVX    @DPTR,A
                MOV     DPTR,#READBUSY
LCDWI1:         MOVX    A,@DPTR
                JB      ACC.7,LCDWI1
                POP     DPL
                POP     DPH
                RET

;****************** LCDWD ******************
LCDWD:          PUSH    DPH
                PUSH    DPL
                MOV     DPTR,#WRITEDATA
                MOVX    @DPTR,A
                MOV     DPTR,#READBUSY
LCDWD1:         MOVX    A,@DPTR
                JB      ACC.7,LCDWD1
                POP     DPL
                POP     DPH
                RET


;********** DISP DELAY SUB **********
DEL_DISP:       MOV     R2,#7FH
DEL_DISP1:      MOV     R3,#7FH
                DJNZ    R3,$
                DJNZ    R2,DEL_DISP1
                DJNZ    R5,DEL_DISP
                RET
              DELAYC:   MOV     R5,#1FH
              DELAY_0:  MOV     R3,#0
              DELAY1:   MOV     R4,#0
                        DJNZ    R4,$
                        DJNZ    R3,DELAY1
                        DJNZ    R5,DELAY_0
                        RET
```

```
DELAYL:   MOV     R3,#0
DELAYL1:  MOV     R4,#0
          DJNZ    R4,$
          DJNZ    R3,DELAYL1
          DJNZ    R2,DELAYL
          RET
DELAYK:   MOV     R3,#0FH
LOOP3:    MOV     R4,#0FFH
LOOP2:    MOV     R5,#30H
LOOP1:    DJNZ    R5,LOOP1
          DJNZ    R4,LOOP2
          DJNZ    R3,LOOP3
          RET

;****************************************
;              CLOKE DIGITAL
;****************************************

CLOK:             LCALL   INIT              ;Setup LCD
                  MOV     A,#1
                  MOV     TEMP_KEY,A
                  LCALL   PROJ
                  MOV     R2,#0FFH
PRO_ROOM:         LCALL   SE2
                  MOV     A,4FH
                  CJNE    A,#0F0H,GO_USE
                  DJNZ    R2,PRO_ROOM
                  LCALL   AA
                  MOV     R2,#0BFH
USE_ROOM:         LCALL   SE2
                  MOV     A,4FH
                  CJNE    A,#0F0H,GO_USE
                  DJNZ    R2,USE_ROOM
                  LCALL   BB
                  MOV     R2,#0BFH
PUSS_KEY:         LCALL   SE2
                  MOV     A,4FH
                  CJNE    A,#0F0H,GO_USE
                  DJNZ    R2,PUSS_KEY
                  SJMP    CLOK
GO_USE:           LCALL   UBEEP
                  RET
SE2:              LCALL   READ              ; READ DS1202
                  LCALL   DISP_T
                  LCALL   DISP_D
                  LCALL   DISP_Y
                  MOV     A,TEMP_KEY
                  CJNE    A,#1,EDIT_T       ; Stanby key
NEXTs:            JNB     RI,SE1
                  CLR     RI
                  MOV     A,SBUF
                  CJNE    A,#'L',SE1
                  LCALL   LOAD_DATA
                  SJMP    SE1
EDIT_T:           CJNE    A,#2,EDIT_D
                  LCALL   EDIT_T1
                  SJMP    SE1
EDIT_D:           CJNE    A,#3,EDIT_Y
                  LCALL   EDIT_D1
                  SJMP    SE1
EDIT_Y:           CJNE    A,#4,SE1
                  LCALL   EDIT_Y1
                  SJMP    SE1
SE1:              MOV     DPTR,#PORTC
                  MOV     A,#0F0H
                  MOVX    @DPTR,A
                  MOVX    A,@DPTR
                  MOV     4FH,A
                  LCALL   SCANKEY
                  RET
```

```
;**************** SCAN KEY ***************;
      SCANKEY:  MOV     DPTR,#PORTB          ;Scankey
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                MOVX    A,@DPTR
                ANL     A,#0F0H

                CJNE    A,#70H,NOTKEY
                MOV     A,TEMP_KEY
                ADD     A,#01H
                MOV     TEMP_KEY,A
                CJNE    A,#07H,NOTKEY
                MOV     A,#01H
                MOV     TEMP_KEY,A
                SJMP    NOTKEY
        NOTKEY: RET

;**************** EDIT TIME ***************;
      EDIT_T1:  MOV     DPTR,#COMMAND        ;Setting Time
                MOV     A,#80H
                MOVX    @DPTR,A
                MOV     DPTR,#TAB4
                LCALL   JJ
           LL:  LCALL   READ
                LCALL   DISP_T
                LCALL   WR_DIS
                LCALL   DELAY
                MOV     DPTR,#PORTB
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR

                LCALL   DELAY4
                MOVX    A,@DPTR
                ANL     A,#0F0H

                CJNE    A,#70H,PP
                MOV     A,TEMP_HR
                CJNE    A,#23H,NOT60AA
                MOV     A,#0
                SJMP    NEXTAA
      NOT60AA:  ADD     A,#1
                DA      A
       NEXTAA:  MOV     TEMP_HR,A
                LJMP    KK

           PP:  CJNE    A,#0D0H,EE
                MOV     A,TEMP_MIN
                CJNE    A,#59H,NOT60A
                MOV     A,#0
                SJMP    NEXTA
       NOT60A:  ADD     A,#1
                DA      A
        NEXTA:  MOV     TEMP_MIN,A
                LJMP    KK

           EE:  CJNE    A,#0E0H,EXITKEY
                MOV     A,TEMP_SEC
                CJNE    A,#59H,NOT60
                MOV     A,#0
                SJMP    NEXT
       NOT60:   ADD     A,#1
                DA      A
         NEXT:  MOV     TEMP_SEC,A
           KK:  LCALL   DELAY
                LCALL   DELAY
                LCALL   WRITE
```

```
EXITKEY:    MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            MOVX    A,@DPTR
            ANL     A,#0F0H
            CJNE    A,#0B0H,LL
            LCALL   WRITE
            MOV     TEMP_KEY,#03
            RET

;***************** EDIT DATE,DAY ************;
EDIT_D1:    MOV     DPTR,#COMMAND       ;Setting Day,Date
            MOV     A,#80H
            MOVX    @DPTR,A
            MOV     DPTR,#TAB5
            LCALL   JJ
LL1:        LCALL   READ
            LCALL   DISP_D
            LCALL   WR_DIS
            LCALL   DELAY
            MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            ANL     A,#0F0H

            CJNE    A,#0D0H,GH
            MOV     A,TEMP_DAY
            CJNE    A,#07H,NOT60U
            MOV     A,#1
            SJMP    NEXTU
NOT60U:     ADD     A,#1
            DA      A
NEXTU:      MOV     TEMP_DAY,A
            LJMP    KK9

GH:         CJNE    A,#0E0H,EXITKEY1
            MOV     A,TEMP_DATE
            CJNE    A,#31H,NOT601
            MOV     A,#1
            SJMP    NEXT1
NOT601:     ADD     A,#1
            DA      A
NEXT1:      MOV     TEMP_DATE,A

KK9:        LCALL   DELAY
            LCALL   DELAY
            LCALL   WRITE

EXITKEY1:   MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            LCALL   DELAY4
            MOVX    A,@DPTR
            MOVX    A,@DPTR
            ANL     A,#0F0H
            CJNE    A,#0B0H,LL1
            LCALL   WRITE
            MOV     TEMP_KEY,#04
            RET

;***************** EDIT MONTH,YEAR***********;
EDIT_Y1:    MOV     DPTR,#COMMAND       ;Setting Month,Year
            MOV     A,#80H
```

```
                MOVX    @DPTR,A
                MOV     DPTR,#TAB6
                LCALL   JJ
          LL2:  LCALL   READ
                LCALL   DISP_Y
                LCALL   WR_DIS
                LCALL   DELAY
                MOV     DPTR,#PORTB
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                ANL     A,#0F0H
                CJNE    A,#0E0H,KEY33
                MOV     A,TEMP_YEAR
                CJNE    A,#99H,NOT603
                MOV     A,#0
                SJMP    NEXT3
       NOT603:  ADD     A,#1
                DA      A
        NEXT3:  MOV     TEMP_YEAR,A
                LJMP    HH

        KEY33:  CJNE    A,#0D0H,EXITKEY2
                MOV     A,TEMP_MONTH
                CJNE    A,#12H,NOT604
                MOV     A,#1
                SJMP    NEXT4
       NOT604:  ADD     A,#1
                DA      A
        NEXT4:  MOV     TEMP_MONTH,A
           HH:  LCALL   DELAY
                LCALL   DELAY
                LCALL   WRITE

     EXITKEY2:  MOV    DPTR,#PORTB
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                LCALL   DELAY4
                MOVX    A,@DPTR
                MOVX    A,@DPTR
                ANL     A,#0F0H
                CJNE    A,#0B0H,LL2
                LCALL   WRITE
                MOV     TEMP_KEY,#05
                RET

;***************** WRITE TO DISPLAY **********;
        WR_DIS: MOV     A,TEMP_SEC
                ANL     A,#0FH
                MOV     SEC1,A
                MOV     A,TEMP_SEC
                ANL     A,#0F0H
                SWAP    A
                MOV     SEC2,A

                MOV     A,TEMP_MIN
                ANL     A,#0FH
                MOV     MIN1,A
                MOV     A,TEMP_MIN
                ANL     A,#0F0H
                SWAP    A
                MOV     MIN2,A

                MOV     A,TEMP_HR
                ANL     A,#0FH
                MOV     HOUR1,A
                MOV     A,TEMP_HR
                MOV     A,#0F0H
```

```
                SWAP    A
                MOV     HOUR2,A

                MOV     A,TEMP_DATE
                ANL     A,#0FH
                MOV     DATE1,A
                MOV     A,TEMP_DATE
                MOV     A,#0F0H
                SWAP    A
                MOV     DATE2,A

                MOV     A,TEMP_MONTH
                ANL     A,#0FH
                MOV     MONTH1,A
                MOV     A,TEMP_MONTH
                MOV     A,#0F0H
                SWAP    A
                MOV     MONTH2,A

                MOV     A,TEMP_DAY
                ANL     A,#0FH
                MOV     DAY1,A
                MOV     A,TEMP_DAY
                MOV     A,#0F0H
                SWAP    A
                MOV     DAY2,A

                MOV     A,TEMP_YEAR
                ANL     A,#0FH
                MOV     YEAR1,A
                MOV     A,TEMP_YEAR
                MOV     A,#0F0H
                SWAP    A
                MOV     YEAR2,A

                MOV     A,RAM_SEC
                ANL     A,#0FH
                MOV     33H,A
                MOV     A,RAM_SEC
                ANL     A,#0F0H
                SWAP    A
                MOV     34H,A

                MOV     A,RAM_MIN
                ANL     A,#0FH
                MOV     35H,A
                MOV     A,RAM_MIN
                ANL     A,#0F0H
                SWAP    A
                MOV     36H,A

                MOV     A,RAM_HR
                ANL     A,#0FH
                MOV     37H,A
                MOV     A,RAM_HR
                MOV     A,#0F0H
                SWAP    A
                MOV     38H,A
                RET

;**************** WRITE  TIMER *************;
        WRITE:  MOV     R6,#8EH
                MOV     R7,#0
                LCALL   BYTEWR

                MOV     R6,#80H
                MOV     R7,TEMP_SEC
                LCALL   BYTEWR

                MOV     R6,#82H
                MOV     R7,TEMP_MIN
```

```
                LCALL   BYTEWR

                MOV     R6,#84H
                MOV     R7,TEMP_HR
                LCALL   BYTEWR

                MOV     R6,#86H
                MOV     R7,TEMP_DATE
                LCALL   BYTEWR

                MOV     R6,#88H
                MOV     R7,TEMP_MONTH
                LCALL   BYTEWR

                MOV     R6,#8AH
                MOV     R7,TEMP_DAY
                LCALL   BYTEWR

                MOV     R6,#8CH
                MOV     R7,TEMP_YEAR
                LCALL   BYTEWR

                MOV     R6,#0C0H
                MOV     R7,RAM_SEC
                LCALL   BYTEWR

                MOV     R6,#0C2H
                MOV     R7,RAM_MIN
                LCALL   BYTEWR

                MOV     R6,#0C4H
                MOV     R7,RAM_HR
                LCALL   BYTEWR
                RET

;**************** SHOW  LCD ****************;
          FF:   MOV     A,50H
                LCALL   WRITEZ
                RET

        PROJ:   MOV     DPTR,#COMMAND
                MOV     A,#80H
                MOVX    @DPTR,A
                MOV     DPTR,#TAB1
                SJMP    JJ
          AA:   MOV     DPTR,#COMMAND
                MOV     A,#80H
                MOVX    @DPTR,A
                MOV     DPTR,#TAB2
                SJMP    JJ
          BB:   MOV     DPTR,#COMMAND
                MOV     A,#80H
                MOVX    @DPTR,A
                MOV     DPTR,#TAB3
          JJ:   MOV     R0,#21
     MAIN1F:    CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITEZ
                INC     DPTR
                DJNZ    R0,MAIN1F
                RET

       TAB1:    DB      ' *** PROJECT ROOM ***'
       TAB2:    DB      ' *** Puss any key ***'
       TAB3:    DB      ' * Use project room *'
       TAB4:    DB      ' HR.MN.SC <===Setting'
       TAB5:    DB      ' Setting=>DY.DT      '
       TAB6:    DB      ' Setting======> MO.YE'


     WRITEZ:    PUSH    ACC
```

```
                PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#WRITEDATA
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     DPH
                POP     DPL
                POP     ACC
                RET

WAITBF:         PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#READBUSY
  RDY1:         MOVX    A,@DPTR
                JB      ACC.7,RDY1
                POP     DPH
                POP     DPL
                RET

  INIT:         PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#COMMAND
                MOV     A,#38H
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#0CH
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#6
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#1
                MOVX    @DPTR,A
                MOV     A,#80H
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     DPH
                POP     DPL
                RET

;*** READ TIME,DATE,MONTH,DAY,YEAR,ALARM ***;
   READ:        MOV     R6,#81H;READ SEC
                LCALL   BYTERD
                MOV     TEMP_SEC,R7
                MOV     A,R7
                ANL     A,#0FH
                MOV     70H,A

                MOV     A,R7
                ANL     A,#0F0H
                SWAP    A
                MOV     71H,A
        ;*************************************;
                MOV     R6,#83H;READ MIN
                LCALL   BYTERD
                MOV     TEMP_MIN,R7
                MOV     A,R7
                ANL     A,#0FH
                MOV     72H,A

                MOV     A,R7
                ANL     A,#0F0H
                SWAP    A
                MOV     73H,A
        ;*************************************;
                MOV     R6,#85H;READ HR
                LCALL   BYTERD
                MOV     TEMP_HR,R7
                MOV     A,R7
                ANL     A,#0FH
                MOV     74H,A
```

```
            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     75H,A
;*********************************;
            MOV     R6,#87H;READ DATE
            LCALL   BYTERD
            MOV     TEMP_DATE,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     DATE1,A

            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     DATE2,A
;*********************************;
            MOV     R6,#89H;READ MONTH
            LCALL   BYTERD
            MOV     TEMP_MONTH,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     MONTH1,A

            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     MONTH2,A
;*********************************;
            MOV     R6,#8BH;READ DAY
            LCALL   BYTERD
            MOV     TEMP_DAY,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     DAY1,A

            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     DAY2,A
;*********************************;
            MOV     R6,#8DH;READ YEAR
            LCALL   BYTERD
            MOV     TEMP_YEAR,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     YEAR1,A

            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     YEAR2,A
;*********************************;
            MOV     R6,#0C1H ;RAM_SEC
            LCALL   BYTERD
            MOV     RAM_SEC,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     33H,A

            MOV     A,R7
            ANL     A,#0F0H
            SWAP    A
            MOV     34H,A
;*********************************;
            MOV     R6,#0C3H ;RAM_MIN
            LCALL   BYTERD
            MOV     RAM_MIN,R7
            MOV     A,R7
            ANL     A,#0FH
            MOV     35H,A
```

```
                MOV     A,R7
                ANL     A,#0F0H
                SWAP    A
                MOV     36H,A
    ;************************************;
                MOV     R6,#0C5H;RAM_HR
                LCALL   BYTERD
                MOV     RAM_HR,R7
                MOV     A,R7
                ANL     A,#0FH
                MOV     37H,A

                MOV     A,R7
                ANL     A,#0F0H
                SWAP    A
                MOV     38H,A
                RET

;***************  WRITE CLOCK  ************;
      BYTEWR:   CLR     P1.6
                LCALL   DELAY4
                CLR     P1.5
                LCALL   DELAY4
                SETB    P1.6
                LCALL   DELAY4
                MOV     B,#08H
                CLR     C

     BYTEWR1:   MOV     A,R6
                RRC     A
                MOV     R6,A
                MOV     P1.4,C
                LCALL   SCLKW
                DJNZ    B,BYTEWR1
                MOV     B,#8
                CLR     C

     BYTEWR2:   MOV     A,R7
                RRC     A
                MOV     R7,A
                MOV     P1.4,C
                LCALL   SCLKW
                DJNZ    B,BYTEWR2
                CLR     RST
                LCALL   DELAY4
                RET

;**************  READ CLOCK  *************;
      BYTERD:   CLR     P1.6
                LCALL   DELAY4
                CLR     P1.5
                LCALL   DELAY4
                SETB    P1.6
                LCALL   DELAY4
                MOV     B,#8
                CLR     C

     BYTERD1:   MOV     A,R6
                RRC     A
                MOV     R6,A
                MOV     P1.4,C
                LCALL   SCLKW
                DJNZ    B,BYTERD1
                MOV     B,#8
                MOV     R7,#0
     BYTERD2:   LCALL   SCLKR
                MOV     A,R7
                MOV     C,P1.4
                RRC     A
                MOV     R7,A
                DJNZ    B,BYTERD2
```

```
                CLR     RST
                LCALL   DELAY4
                RET

;***************  Enabel Write  ***********;
        SCLKW:  CLR     P1.5
                LCALL   DELAY4
                SETB    P1.5
                LCALL   DELAY4
                RET

;***************  Enabel Read  *************;
        SCLKR:  SETB    P1.5
                LCALL   DELAY4
                CLR     P1.5
                LCALL   DELAY4
                RET

;***************  MODE DELAY  **************;
        DELAY:  MOV     R1,#090H
          L11:  MOV     R0,#0
          L12:  DJNZ    R0,L12
                DJNZ    R1,L11
                RET

       DELAY3:  MOV     R1,#050H
          L31:  MOV     R0,#0
          L32:  DJNZ    R0,L32
                DJNZ    R1,L31
                RET

       DELAY2:  MOV     R1,#01H
          L21:  MOV     R0,#07FH
          L22:  DJNZ    R0,L22
                DJNZ    R1,L21
                RET

       DELAY4:  MOV     R0,#5
                DJNZ    R0,$
                RET

       DELAYX:  MOV     R5,#01H
       DELAYY:  MOV     R3,#0
       DELAYZ:  MOV     R4,#0
                DJNZ    R4,$
                DJNZ    R3,DELAYZ
                DJNZ    R5,DELAYY
                RET

;***************  DISPLAY  TIME  ***********;
        DISP_T: MOV     DPTR,#COMMAND
                MOV     A,#0C0H
                MOVX    @DPTR,A
                LCALL   WAITBF

                MOV     A,75H
                LCALL   CHECK0
                MOV     HOUR2,50H
                LCALL   FF

                MOV     A,74H
                LCALL   CHECK0
                MOV     HOUR1,50H
                LCALL   FF

                LCALL   WWW
                LCALL   FF

                MOV     A,73H
                LCALL   CHECK0
                MOV     MIN2,50H
```

```
                LCALL   FF

                MOV     A,72H
                LCALL   CHECK0
                MOV     MIN1,50H
                LCALL   FF

                LCALL   WWW
                LCALL   FF

                MOV     A,71H
                LCALL   CHECK0
                MOV     SEC2,50H
                LCALL   FF

                MOV     A,70H
                LCALL   CHECK0
                MOV     SEC1,50H
                LCALL   FF
                RET

;**************  DISPLAY DAY,DATE  *******;
      DISP_D:   MOV     DPTR,#COMMAND
                MOV     A,#0C8H
                MOVX    @DPTR,A
                LCALL   WAITBF

                LCALL   XXX
                LCALL   FF

                LCALL   XXX
                LCALL   FF

                MOV     A,DAY1
                LCALL   CHECK1
                MOV     DAY1,50H
                LCALL   FF

                LCALL   ZZZ
                LCALL   FF

                MOV     A,DATE2
                LCALL   CHECK0
                MOV     DATE2,50H
                LCALL   FF

                MOV     A,DATE1
                LCALL   CHECK0
                MOV     DATE1,50H
                LCALL   FF
                RET

;**************  DISPLAY MONTH,YEAR  ********;
      DISP_Y:   MOV     DPTR,#COMMAND
                MOV     A,#0CEH
                MOVX    @DPTR,A
                LCALL   WAITBF

                LCALL   ZZZ
                LCALL   FF

                MOV     A,MONTH2
                LCALL   CHECK0
                MOV     MONTH2,50H
                LCALL   FF

                MOV     A,MONTH1
                LCALL   CHECK0
                MOV     MONTH1,50H
                LCALL   FF
```

```
                LCALL   ZZZ
                LCALL   FF

                MOV     A,YEAR2
                LCALL   CHECK0
                MOV     YEAR2,50H
                LCALL   FF

                MOV     A,YEAR1
                LCALL   CHECK0
                MOV     YEAR1,50H
                LCALL   FF
                RET

;**************  Check Data  **************;
      CHECK0:   CJNE    A,#00H,NUM10
                MOV     50H,#'0'
                SJMP    EXITC0
       NUM10:   CJNE    A,#01H,NUM20
                MOV     50H,#'1'
                SJMP    EXITC0
       NUM20:   CJNE    A,#02H,NUM30
                MOV     50H,#'2'
                SJMP    EXITC0
       NUM30:   CJNE    A,#03H,NUM40
                MOV     50H,#'3'
                SJMP    EXITC0
       NUM40:   CJNE    A,#04H,NUM50
                MOV     50H,#'4'
                SJMP    EXITC0
       NUM50:   CJNE    A,#05H,NUM60
                MOV     50H,#'5'
                SJMP    EXITC0
       NUM60:   CJNE    A,#06H,NUM70
                MOV     50H,#'6'
                SJMP    EXITC0
       NUM70:   CJNE    A,#07H,NUM80
                MOV     50H,#'7'
                SJMP    EXITC0
       NUM80:   CJNE    A,#08H,NUM90
                MOV     50H,#'8'
                SJMP    EXITC0
       NUM90:   MOV     50H,#'9'
      EXITC0:   RET

;**************  Check Data Day **************;
      CHECK1:   CJNE    A,#01H,NUM11
                MOV     50H,#'M'
                SJMP    EXITC
       NUM11:   CJNE    A,#02H,NUM21
                MOV     50H,#'T'
                SJMP    EXITC
       NUM21:   CJNE    A,#03H,NUM31
                MOV     50H,#'W'
                SJMP    EXITC
       NUM31:   CJNE    A,#04H,NUM41
                MOV     50H,#'t'
                SJMP    EXITC
       NUM41:   CJNE    A,#05H,NUM51
                MOV     50H,#'F'
                SJMP    EXITC
       NUM51:   CJNE    A,#06H,NUM61
                MOV     50H,#'Z'
                SJMP    EXITC
       NUM61:   MOV     50H,#'S'
       EXITC:   RET

;****************************************;
         WWW:   MOV     50H,#':'
                RET
         XXX:   MOV     50H,#' '
```

```
        RET
ZZZ:    MOV     50H,#'/'
        RET

        END
```

รูปที่ ค.2 โปรแกรมควบคุมระบบการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

ภาคผนวก ง

รายละเอียดของอุปกรณ์

รายการอุปกรณ์วงจรรีเลย์

อุปกรณ์	ค่า/เบอร์	จำนวน
รีเลย์	12 โวล์ท	16 ตัว
TR	2N2222	16 ตัว
D	1N4001	16 ตัว
R	330 โอห์ม	16 ตัว
R	10 โอห์ม	16 ตัว
Socket IC 10 ขา		2 ตัว
Jack Vcc		2 ตัว
Terminal	220 โวล์ท	16 ตัว

รายการอุปกรณ์ควบคุมรีเลย์

อุปกรณ์	ค่า/เบอร์	จำนวน
IC	MCS-8051	1 ตัว
IC	74244	2 ตัว
IC	SN75176	1 ตัว
X-Tal	11.059 MHz	1 ตัว
D	1N4007	1 ตัว
D	4148	1 ตัว
C ชนิดโพลีเอทเทอร์รีน	30 PF	2 ตัว
C ชนิดอิเล็คโตรไลท์	10uF	1 ตัว
SW- PB		1 ตัว
TR	BC550	1 ตัว
TR	BC560	2 ตัว
LED สีแดง		2 ตัว
R	4.7 K.	6 ตัว
Soket IC 10 ขา		2 ตัว

อุปกรณ์	ค่า/เบอร์	จำนวน
RS-485		1 ตัว
RS-232		1 ตัว

รายการอุปกรณ์วงจรแสดงผลการใช้งานโต๊ะผ่าน LED

อุปกรณ์	ค่า/เบอร์	จำนวน
LED สีเขียว		16 ตัว
Socket 20 ขา		1 ตัว
R	10 โอห์ม	16 ตัว

รายการอุปกรณ์วงจรเรกกูเลเตอร์

อุปกรณ์	ค่า/เบอร์	จำนวน
D	1N485	4 ตัว
C ชนิดอิเล็คโตรไลท์	4.7 μF	1 ตัว
C ชนิดอิเล็คโตรไลท์	220 μF	4 ตัว
Jack Vcc.		5 ตัว

รายการอุปกรณ์วงจร ANT 31PJ

อุปกรณ์	ค่า/เบอร์	จำนวน
IC	MCS 8951	1
IC	MAX 1232	1
IC	DS 275	1
IC	SN 75176	1
IC	PAL 16L8	1

อุปกรณ์	ค่า/เบอร์	จำนวน
IC	27256	1
IC	62256	1
IC	6255	1
R PACK	10K	1
RS	232	1
RS	485	1
SW	PB	1
CRYSTAL	11.0592 MHz	1
C ชนิดโพลีเอทเทอร์รีน	30 PF	2
Cชนิดโพลีเอทเทอร์รีน	0.1μF	8
JUMPER		5
Socket	26 ขา	1
Socket	16 ขา	1
Socket	8	1
Jack Vcc.		2

DALLAS SEMICONDUCTOR

www.dalsemi.com

Application Note 95

Interfacing the DS1307/1308 with an 8051-Compatible Microcontroller

INTRODUCTION

The DS1307 Serial Real Time Clock, which incorporates a 2–wire serial interface, can be controlled using an 8051–compatible DS5000 Secure Microcontroller. The DS1307 is connected directly to two of the I/O ports on a DS5000 microcontroller and the 2–wire handshaking is handled by low–level drivers, which are discussed in this application note.

DS1307 DESCRIPTION

The DS1307 Serial Real Time Clock is a low–power, full BCD clock/calendar plus 56 bytes of nonvolatile SRAM. Address and data are transferred serially via the 2–wire bi–directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with AM/PM indicator. The DS1307 has a built–in power sense circuit which detects power failures and automatically switches to the battery supply.

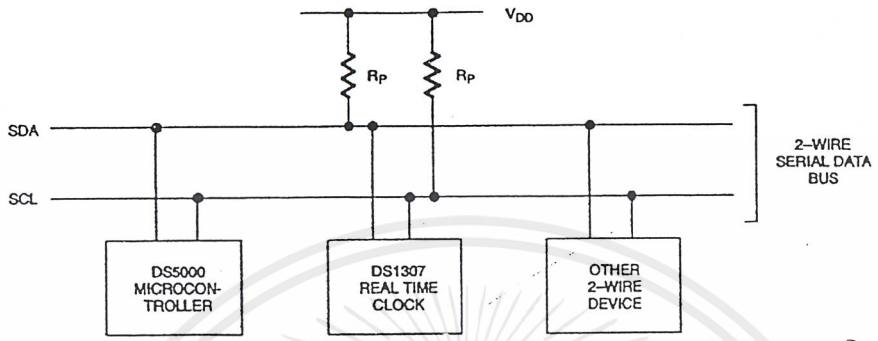
DS1307 OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. The START and STOP conditions are generated using the low level drives, SEND_START and SEND_STOP found in the attached DS5000 code. Also the subroutines SEND_BYTE and READ_BYTE provide the 2–wire handshaking required for writing and reading 8–bit words to and from the DS1307.

HARDWARE CONFIGURATION

The system is configured as shown in Figure 1. The DS1307 has the 2–wire bus connected to two I/O port pins of the DS5000: SCL – P1.0, SDA – P1.1. The V_{DD} voltage is 5V, $R_P = 5K\Omega$ and the DS5000 is using a 12-MHz crystal. The other peripheral device could be any other device that recognizes the 2–wire protocol, such as the DS1621 Digital Thermometer and Thermostat. The interface with the D5000 was accomplished using the DS5000T Kit hardware and software. This development kit allows the PC to be used as a dumb terminal using the DS5000's serial ports to communicate with the keyboard and monitor.

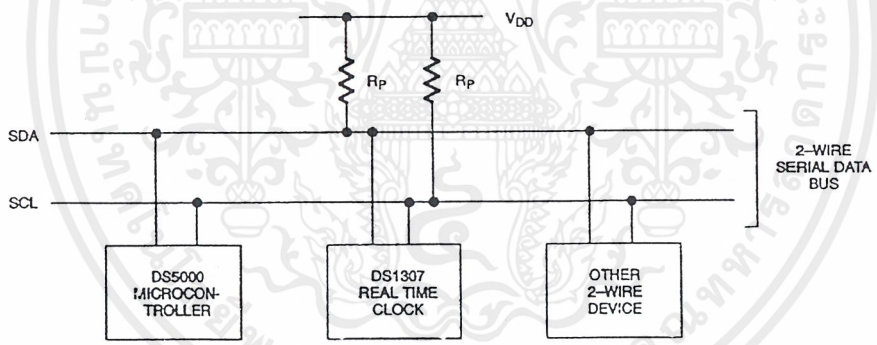
TYPICAL 2–WIRE BUS CONFIGURATION Figure 1



The following bus protocol has been defined (see Figure 2).

APPLICATION NOTE 95

TYPICAL 2–WIRE BUS CONFIGURATION Figure 1



The following bus protocol has been defined (see Figure 2).

- During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line while the clock line is high will be interpreted as control signals.

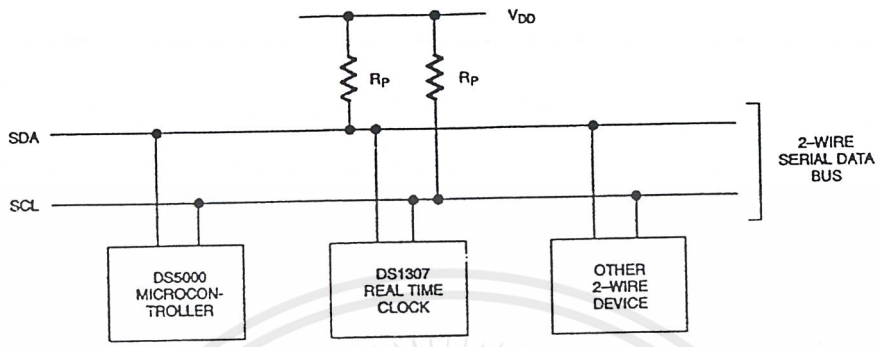
Accordingly, the following bus conditions have been defined:

Start data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

Stop data transfer: A change in the state of the data line from low to high, while the clock line is high, defines the STOP condition.

Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock signal. The data on the line must be changed during the low period of the clock signal. There is one clock pulse per bit of data.

TYPICAL 2–WIRE BUS CONFIGURATION Figure 1



The following bus protocol has been defined (see Figure 2).

- During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Start data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

Stop data transfer: A change in the state of the data line from low to high, while the clock line is high, defines the STOP condition.

Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock signal. The data on the line must be changed during the low period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between the START and the STOP conditions is not limited, and is determined by the master device. The information is transferred byte–wise and each receiver acknowledges with a ninth bit.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

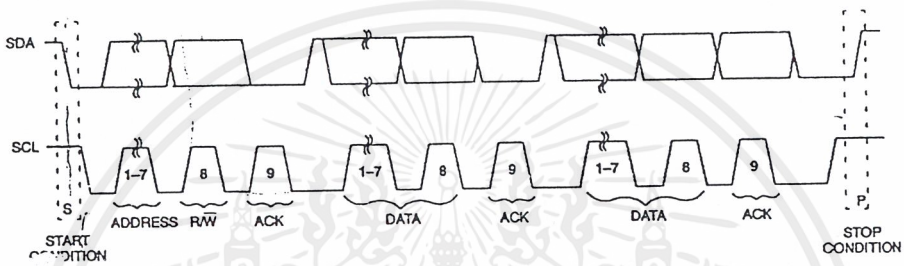
A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable low during the high period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line high to enable the master to generate the STOP condition.

Figure 2 details how data transfer is accomplished on the 2–wire bus. Depending on the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a not acknowledge is returned.

DATA TRANSFER ON 2–WIRE SERIAL BUS Figure 2



The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

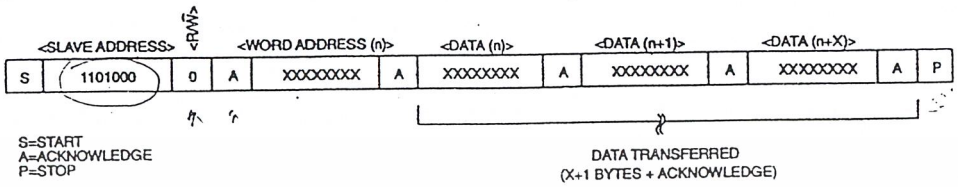
The DS1307 may operate in the following two modes:

1. **Slave receiver mode (DS1307 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit (see Figure 3). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7–bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$) which for a write is a 0. After receiving and decoding the address byte, the DS1307 outputs an acknowledge on the SDA line. After the DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307. This will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

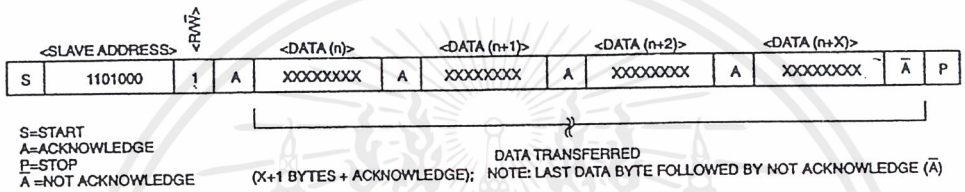
2. **Slave transmitter mode (DS1307 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 4). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7–bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$),which for a read is a 1. After receiving and decoding the address byte, the DS1307 inputs an acknowledge on the SDA line. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation

of a read mode, the first address that is read is the last one stored in the register pointer. The DS1307 must be sent a Not–Acknowledge bit by the master to terminate a read.

DATA WRITE – SLAVE RECEIVER MODE Figure 3



DATA READ – SLAVE TRANSMITTER MODE Figure 4



SOFTWARE OPERATION

DS5000 INTERFACE

The software presented in Appendix 1 is written to interface the DS5000 with the DS1307 over the 2–wire interface. The DS5000 was programmed using Dallas Semiconductor's DS5000T Evaluation Kit, which allows a PC to be used as a dumb terminal. The KIT5K software environment supplied with the DS5000T Evaluation Kit provides a high–level interface for loading application software to the DS5000 or for setting its configuration parameters via the Program command. The KIT5K software includes a dumb terminal emulator to allow users to run application software in the DS5000, which communicates with the user via a PC COM port.

DS1307 SOURCE CODE

The first section of the code found in the Appendix is used to configure the DS5000 for serial communication with the PC. Also at the beginning of the code is the MASTER_CONTROLLER subroutine which is used to control the demonstration software.

The subroutines that immediately follow the MASTER_CONTROLLER subroutine are the low level drivers for controlling the 2–wire interface. They are not specific to the DS1307 but can be used with any 2–wire compatible slave–only device. These subroutines are:

SEND_START
This subroutine is used to generate the Start condition on the 2–wire bus.

SEND_STOP
This subroutine is used to generate the Stop condition on the 2–wire bus.

SEND_BYTE
This subroutine sends an 8–bit word, MSB first, over the 2–wire bus with a 9th clock pulse for the Acknowledge pulse.

READ_BYTE
This subroutine reads an 8–bit word over the 2–wire bus. It checks for the LASTREAD flag to be cleared indicating when the last read from the slave device is to occur. If it is not the last read, the DS5000 sends an Acknowledge pulse on the 9th clock and if it is the last read from the slave device, the DS5000 sends a Not–Acknowledge.

SCL_HIGH
This subroutine transitions the SCL line low–to–high and ensures the SCL line is high before continuing.

DELAY and DELAY_4
These two subroutines have been included to ensure that the 2–wire bus timing is maintained.

The rest of the code included in the appendix is specifically designed to demonstrate the functions of the DS1307. The functions that are demonstrated are:

Setting Time
The time is read in from the keyboard and stored in the DS5000 scratchpad memory. It is then transferred, over the 2–wire interface, to the DS1307.

Set RAM
A single hex byte is read in from the keyboard and written to the entire user RAM of the DS1307.

Read Date/Time
The date and time are read, over the 2–wire bus, and stored in the DS5000 scratchpad memory. It is then written to the screen. This continues until a key is pressed on the keyboard.

Read RAM
The entire user RAM of the DS1307 is read into the DS5000 scratchpad memory and then written to the PC monitor.

OSC On/ OSC Off
The DS1307 clock oscillator can be turned on or off.

SQW/OUT On/ SQW/OUT Off
The SQW/OUT can be turned on or off. It will toggle at 1 Hz.

AC ELECTRICAL CHARACTERISTICS Table 1

PARAMETER	SYMBOL	ACTUAL	UNITS
SCL Clock Frequency	f_{SCL}	59	kHz
Bus Free Time Between a STOP and START condition	t_{BUF}	5.7	µs
Hold Time (repeated) START Condition	$t_{HD:STA}$	6.2	µs
LOW Period of SCL Clock	t_{LOW}	10.5	µs
HIGH Period of SCL Clock	t_{HIGH}	6.5	µs
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	5.3	µs
Data Hold Time	$t_{HD:DAT}$	5.5	µs
Data Set-up Time	$t_{SU:DAT}$	3.1	µs
Set-up Time for STOP Condition	$t_{SU:STO}$	5.4	µs

CONCLUSION

It has been shown that it is very straight forward to interface the DS1307 or any other 2–wire slave device to an 8051–compatible microcontroller. The only concern must be that the 2–wire timing specification is not violated by the low level drivers on the microcontroller. The delay subroutines have been inserted into the code for this purpose. The values in Table 1 are the actual timing parameters observed in the hardware setup used to develop this application note.

Features

- Compatible with MCS-51™ Products
- 4 Kbytes of In-System Reprogrammable Flash Memory
  Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

8-Bit Microcontroller with 4 Kbytes Flash

Description

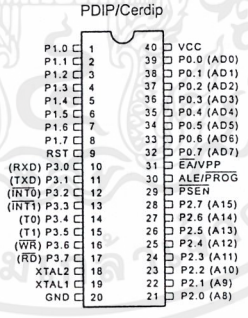
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4 Kbytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

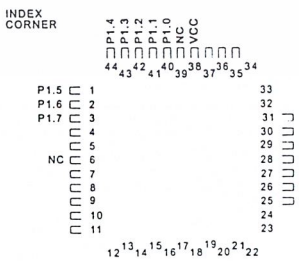
The AT89C51 provides the following standard features: 4 Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is

(continued)

Pin Configurations

PDIP/Cerdip





AT89C51

Features

- Compatible with MCS-51™ Products
- 4 Kbytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

8-Bit Microcontroller with 4 Kbytes Flash

Description

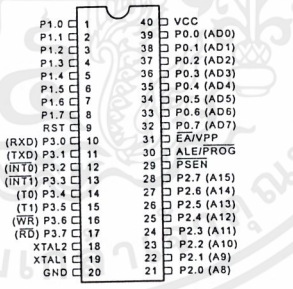
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4 Kbytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C51 provides the following standard features: 4 Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is

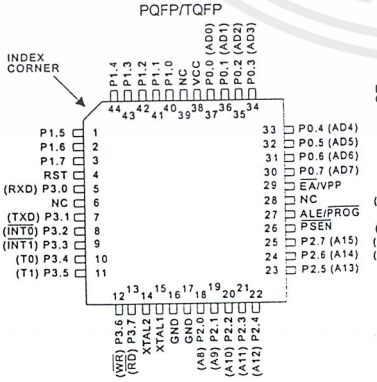
(continued)

Pin Configurations

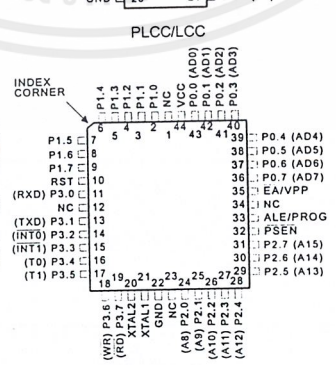
PDIP/Cerdip



PQFP/TQFP

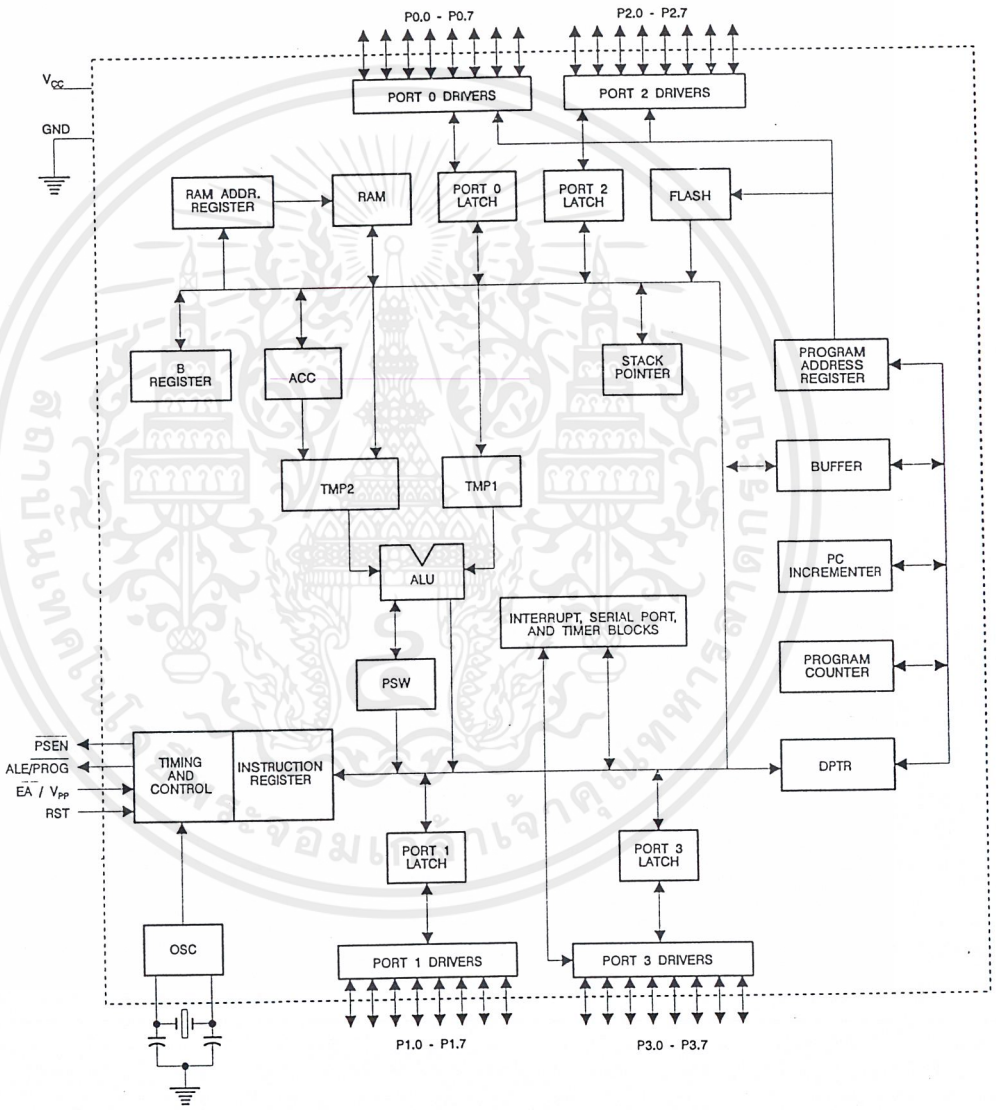


PLCC/LCC



0265E

Block Diagram



Description (Continued)

designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Description

V_{CC}

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 may also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 1 also receives the low-order address bytes during Flash programming and program verification.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (extenal interrupt 0)
P3.3	$\overline{INT1}$ (extenal interrupt 1)
P3.4	T0 (timer 0 extenal input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (extenal data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and programming verification.

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcrontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89C51 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/V_{PP}

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming, for parts that require 12-volt V_{PP}.

(continued)

Pin Description (Continued)

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

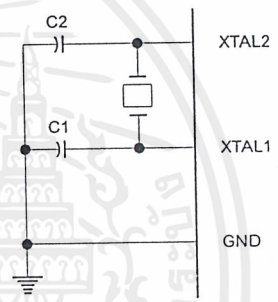
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

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

Power Down Mode

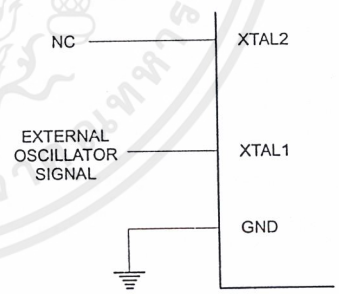
In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Figure 1. Oscillator Connections



Notes: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Status of External Pins During Idle and Power Down

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

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

Lock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory. $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP} = 12 V	V_{PP} = 5 V
Top-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H)=1EH (031H)=51H (032H)=FFH	(030H)=1EH (031H)=51H (032H)=05H

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}$/V_{PP} to 12 V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling: The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 51H indicates 89C51
(032H) = FFH indicates 12 V programming
(032H) = 05H indicates 5 V programming

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L	⎍	H/12V[1]	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L	⎍	H/12V	H	H	H	H
	Bit - 2	H	L	⎍ [2]	H/12V	H	H	L	L
	Bit - 3	H	L	⎍	H/12V	H	L	H	L
Chip Erase		H	L	⎍	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Notes: 1. The signature byte at location 032H designates whether V_{PP} = 12 V or V_{PP} = 5 V should be used to enable programming.
2. Chip Erase requires a 10 ms $\overline{PROG}$ pulse.

Figure 3. Programming the Flash

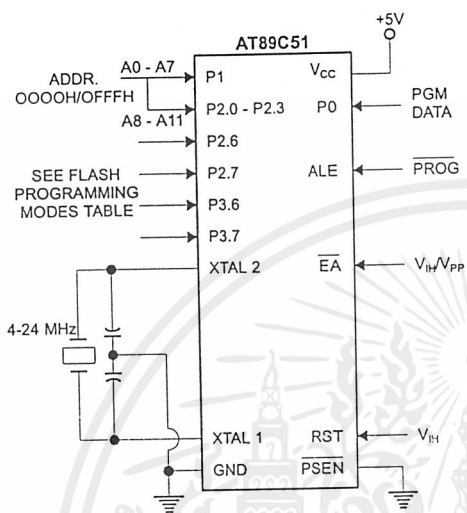
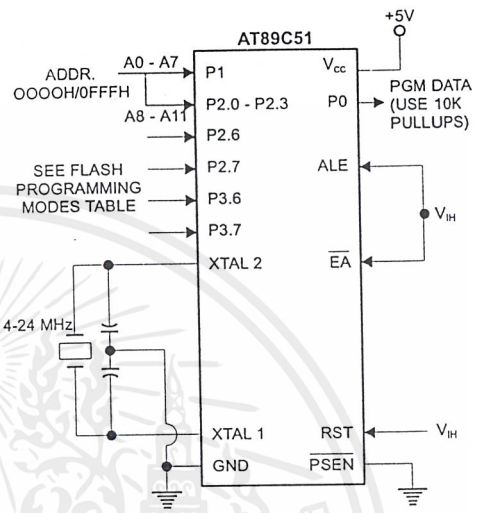


Figure 4. Verifying the Flash



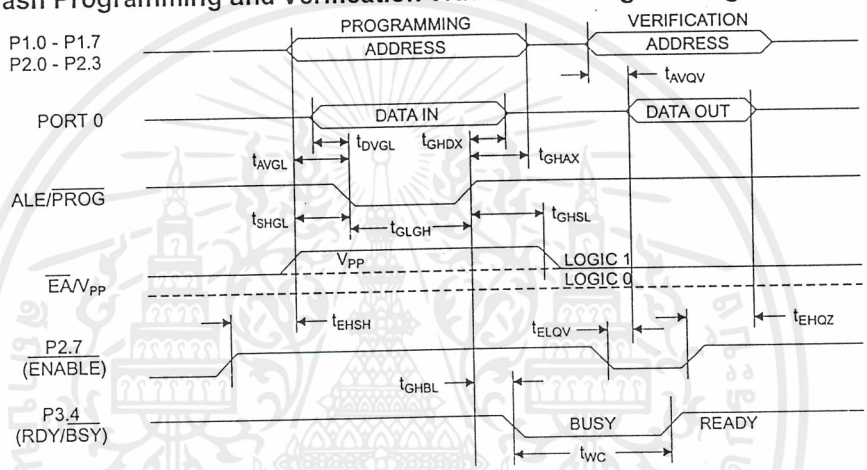
Flash Programming and Verification Characteristics

T_A = 21°C to 27°C, V_{CC} = 5.0 ± 10%

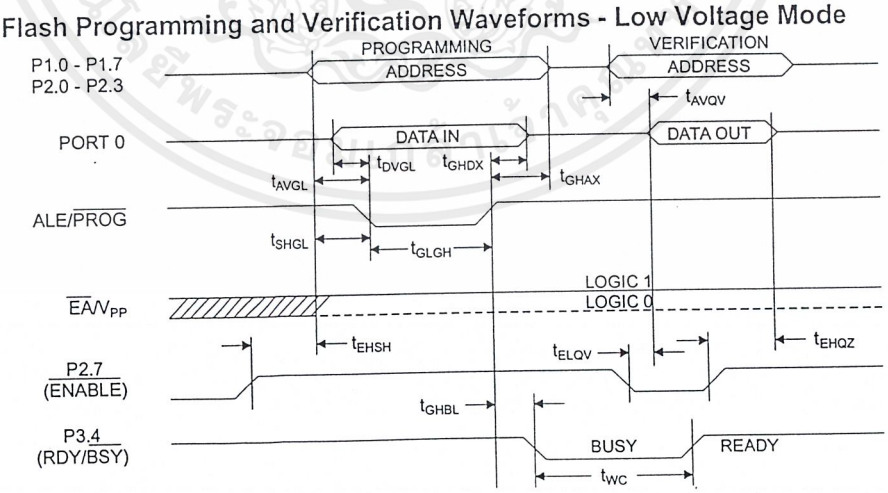
Symbol	Parameter	Min	Max	Units
V_{PP}[1]	Programming Enable Voltage	11.5	12.5	V
I_{PP}[1]	Programming Enable Current		1.0	mA
1/t_{CLCL}	Oscillator Frequency	4	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	48t_{CLCL}		
t_{GHAX}	Address Hold After $\overline{PROG}$	48t_{CLCL}		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	48t_{CLCL}		
t_{GHDX}	Data Hold After $\overline{PROG}$	48t_{CLCL}		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48t_{CLCL}		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}[1]	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		48t_{CLCL}	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		48t_{CLCL}	
t_{EHQV}	Data Float After $\overline{ENABLE}$	0	48t_{CLCL}	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms - High Voltage Mode



Flash Programming and Verification Waveforms - Low Voltage Mode



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0 V to +7.0 V
Maximum Operating Voltage	6.6 V
DC Output Current	15.0 mA

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

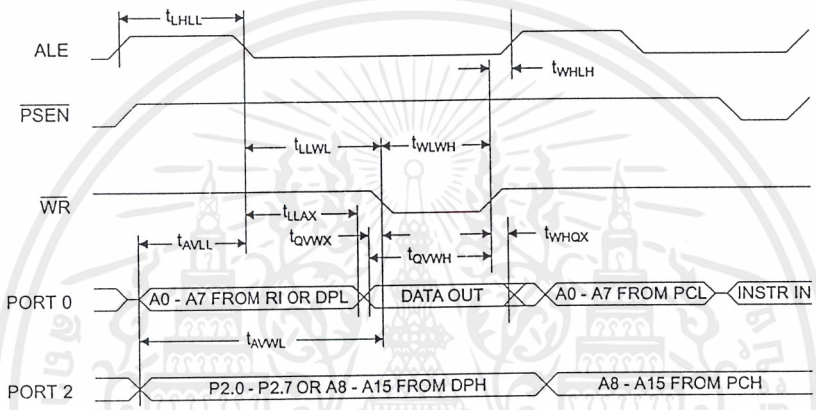
D.C. Characteristics

T_A = -40°C to 85°C, V_{CC} = 5.0 V ± 20% (unless otherwise noted)

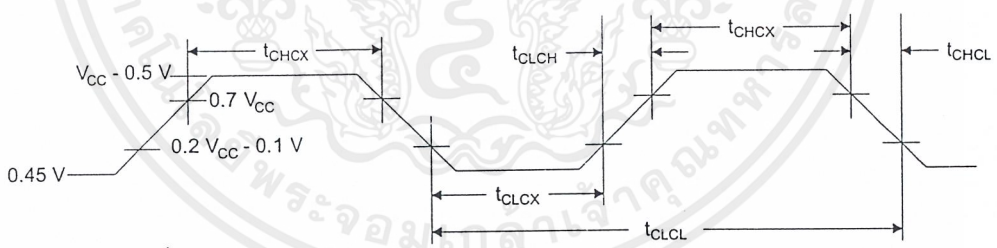
Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45 V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2 V		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	kΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power Down Mode[2]	V_{CC} = 6 V		100	µA
		V_{CC} = 3 V		40	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10 mA
Maximum I_{OL} per 8-bit port:
Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total IOL for all output pins: 71 mA
If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum VCC for Power Down is 2 V.

External Data Memory Cycle



External Clock Drive Waveforms



External Clock Drive

Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

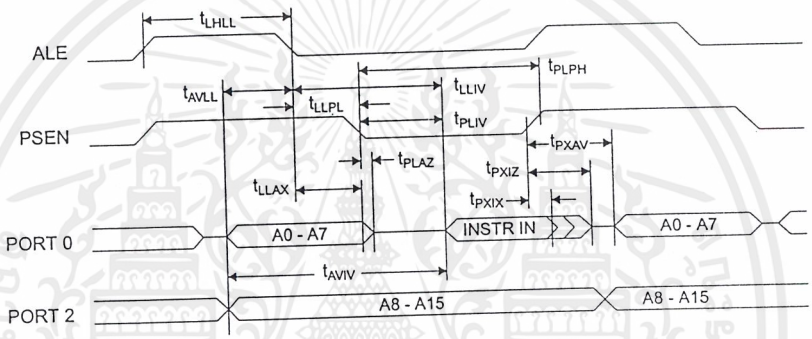
A.C. Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all other outputs = 80 pF)

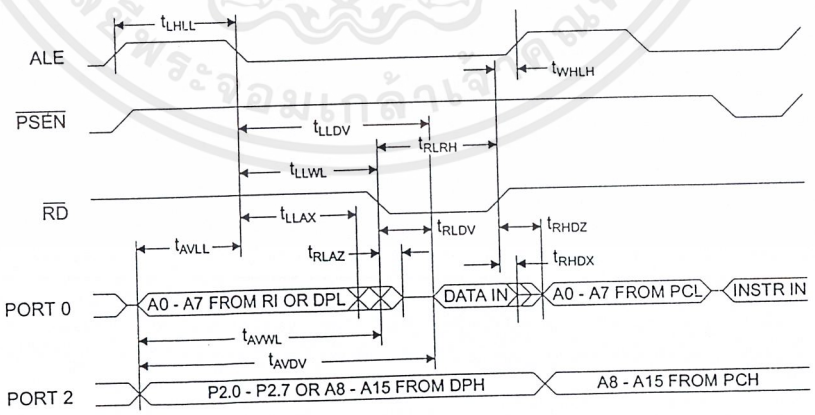
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	24	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
t_{AVLL}	Address Valid to ALE Low	28		tCLCL-13		ns
t_{LLAX}	Address Hold After ALE Low	48		tCLCL-20		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		4tCLCL-65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-13		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-20		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-45	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-10	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}$-55	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
t_{LLDV}	ALE Low to Valid Data In		517		8tCLCL-150	ns
t_{AVDV}	Address to Valid Data In		585		9tCLCL-165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-20		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-120		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		t_{CLCL}-20		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-20	t_{CLCL}+25	ns

External Program Memory Read Cycle



External Data Memory Read Cycle

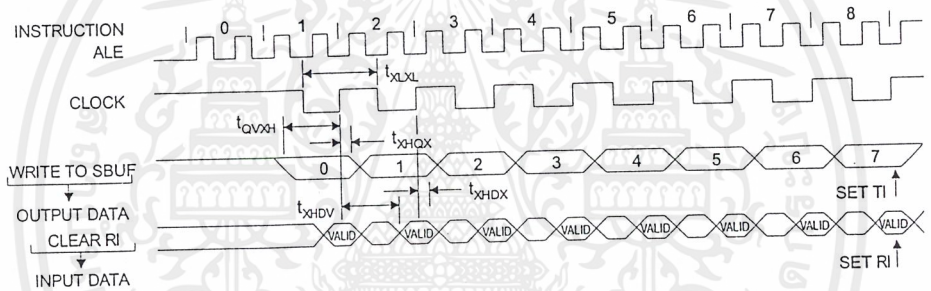


Serial Port Timing: Shift Register Mode Test Conditions

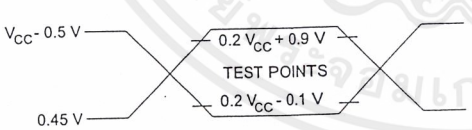
(V_{CC} = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		12t_{CLCL}		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		10t_{CLCL}-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		2t_{CLCL}-33		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		10t_{CLCL}-133	ns

Shift Register Mode Timing Waveforms

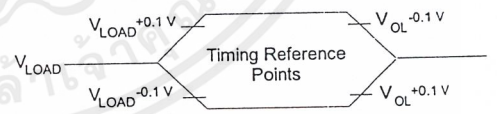


AC Testing Input/Output Waveforms[1]



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5 V for a logic 1 and 0.45 V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms[1]



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5 V ± 20%	AT89C51-12AC AT89C51-12JC AT89C51-12PC AT89C51-12QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-12AI AT89C51-12JI AT89C51-12PI AT89C51-12QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
		AT89C51-12AA AT89C51-12JA AT89C51-12PA AT89C51-12QA	44A 44J 40P6 44Q	Automotive (-40°C to 125°C)
	5 V ± 10%	AT89C51-12DM AT89C51-12LM	40D6 44L	Military (-55°C to 125°C)
		AT89C51-12DM/883 AT89C51-12LM/883	40D6 44L	Military/883C Class B, Fully Compliant (-55°C to 125°C)
16	5 V ± 20%	AT89C51-16AC AT89C51-16JC AT89C51-16PC AT89C51-16QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-16AI AT89C51-16JI AT89C51-16PI AT89C51-16QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
		AT89C51-16AA AT89C51-16JA AT89C51-16PA AT89C51-16QA	44A 44J 40P6 44Q	Automotive (-40°C to 125°C)
20	5 V ± 20%	AT89C51-20AC AT89C51-20JC AT89C51-20PC AT89C51-20QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
		AT89C51-20AI AT89C51-20JI AT89C51-20PI AT89C51-20QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
24	5 V ± 20%	AT89C51-24AC AT89C51-24JC AT89C51-24PC AT89C51-24QC	44A 44J 44P6 44Q	Commercial (0°C to 70°C)
		AT89C51-24AI AT89C51-24JI AT89C51-24PI AT89C51-24QI	44A 44J 44P6 44Q	Industrial (-40°C to 85°C)

Ordering Information

Package Type	
44A	44 Lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
40D6	40 Lead, 0.600" Wide, Non-Windowed, Ceramic Dual Inline Package (Cerdip)
44J	44 Lead, Plastic J-Leaded Chip Carrier (PLCC)
44L	44 Pad, Non-Windowed, Ceramic Leadless Chip Carrier (LCC)
40P6	40 Lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44Q	44 Lead, Plastic Gull Wing Quad Flatpack (PQFP)

ภาคผนวก จ

คู่มือการใช้งานระบบควบคุม
การใช้เครื่องมือในห้องปฏิบัติการโครงงาน

คู่มือการใช้งานระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

อุปกรณ์ที่ต้องการ

1) ชุดอ่านบาร์โค้ดและเก็บรวบรวมข้อมูล
2) ชุดควบคุมการทำงานของรีเลย์ 16 ช่อง
3) เครื่องคอมพิวเตอร์
4) สายพ่วงต่อ RS -485

สิ่งที่ควรรู้เกี่ยวกับระบบควบคุมการใช้เครื่องมือในห้องปฏิบัติการโครงงาน

การติดตั้ง

1) จัดโต๊ะให้เหมาะสมตามต้องการ แล้วเดินสายไฟไปตามโต๊ะต่างๆ 16 ชุด
2) จัดทำชั้นวางอุปกรณ์ ซึ่งตัวหนึ่งมี 3 ชั้น เพื่อวางออสซิลโลสโคป เพาว์เวอร์ซัพพลาย ฟังก์ชั่นเจนเนอร์เรเตอร์
3) ติดตั้งชั้นอุปกรณ์ตามโต๊ะต่างๆ 12 ตัว

รูปที่ จ.1 การจัดโต๊ะเดินสายไฟพร้อมชั้นวางอุปกรณ์ภายในห้องปฏิบัติการโครงงาน

1) ติดตั้งชุดอ่านบาร์โค้ดและตัวเก็บข้อมูลไว้ทางด้านหน้าของห้องโครงงาน

รูปที่ จ.2 การติดตั้งชุดอ่านบาร์โค้ดและตัวเก็บข้อมูล

2) ติดตั้งชุดควบคุมการทำงานของรีเลย์ 16 ช่อง

รูปที่ จ.3 การติดตั้งชุดควบคุมรีเลย์ 16 ช่อง

ขั้นตอนการใช้งาน

1) เมื่อเข้ามาสู่ห้องปฏิบัติการโครงงาน ถ้าต้องการใช้อุปกรณ์ที่ติดตั้งไว้ตามโต๊ะ 12 ชุด ซึ่งประกอบด้วย เพาว์เวอร์ซัพพลาย ออสซิลโลสโคป และฟังก์ชั่นเจนเนอร์เรเตอร์ ให้ท่านสัมผัสคีย์ใดๆ ที่เครื่องอ่านบาร์โค้ด เพื่อเข้าสู่โปรแกรมการใช้งาน

2) ที่หน้าจอแสดงผลผลึกเหลว (LCD) จะปรากฏข้อความ Hello

3) ให้ท่านรูดบัตรนักศึกษา และกด Pass Word (ซึ่งเป็นรหัส 4 ตัวท้ายของรหัสนักศึกษาเอง) ถ้ารหัสถูกต้องเครื่องก็จะเข้าสู่ฟังก์ชั่นใช้งาน

4) เมื่อเข้าสู่ฟังก์ชั่น

ถ้าต้องการใช้อุปกรณ์ที่ติดตั้งตามโต๊ะต่างๆ ให้กดเลข 1

ถ้าต้องการเข้าสู่ฟังก์ชั่นอื่นๆ ให้กดเลข 2

ถ้าต้องการออกจากระบบการใช้งาน ให้กดเลข 3

เมื่อกดเลข 1 หน้าจอแสดงผลผลึกเหลว (LCD) จะแสดงข้อความว่า Select Table

จากนั้น สักครู่ ที่จอจอแสดงผลผลึกเหลว (LCD) จะปรากฏตัวเลข 1 - 6 เพื่อจะแสดงว่าโต๊ะใดยังว่างอยู่ ถ้าโต๊ะว่างเครื่องก็จะบอกว่าเครื่องที่เท่าไรว่าง

5) ท่านสามารถกดเลือกโต๊ะที่ว่างเพื่อใช้งานได้เลย

6) ถ้าท่านต้องการเลิกใช้งาน ให้ท่านกลับไปเริ่มปฏิบัติตามข้อ 1 - 4 ซึ่งหน้าจอขณะนี้จะอยู่ที่ฟังก์ชั่นหลัก ซึ่งท่านสามารถกด 1 อีกครั้งเครื่องจะทำการตรวจสอบว่าท่านได้เข้าสู่ระบบใช้งานเป็นครั้งที่ 2 เครื่องจะทำการตัดไฟ (OFF) โต๊ะที่ท่านใช้งานอยู่โดยอัตโนมัติ

7) หากท่านต้องการแจ้งอุปกรณ์ชำรุด ก่อนจะออกจากระบบ ให้กดเลข 2 ในฟังก์ชั่นหลัก จอแสดงผลผลึกเหลว (LCD) จะแสดงข้อความ

1. Check tool damage
2. Inform damage

ถ้ากดเลข 1 เครื่องจะทำการตรวจสอบว่าในแต่ละโต๊ะมีเครื่องชำรุดหรือไม่ ถ้าไม่พบอุปกรณ์ชำรุด จะปรากฏข้อความ NO damage ถ้ามีอุปกรณ์ชำรุดเครื่องจะแสดงว่าที่โต๊ะใดมีอุปกรณ์ชำรุด จากนั้น ถ้าท่านยังต้องการทราบอีกว่า อุปกรณ์ที่ชำรุดคือตัวใด ให้ท่านกดหมายเลขของโต๊ะที่ปรากฏบนจอได้เลย

8) ถ้าหากท่านต้องการจะแจ้งอุปกรณ์ที่ท่านใช้งานอยู่ว่าชำรุด ให้ท่านกดเลข 2 ที่ฟังก์ชั่นหลัก หน้าจอแสดงผลผลึกเหลว (LCD) จะปรากฏข้อความว่า Select Table เพื่อถามว่าอุปกรณ์ที่โต๊ะใดชำรุด จากนั้นท่านก็สามารถแจ้งหมายเลขโต๊ะได้เลย สักครู่หน้าจอแสดงผลก็จะแสดงราย

ชื่ออุปกรณ์บนโต๊ะ 3 ตัว เพื่อที่จะให้ท่านระบุว่าตัวใดเสีย จากนั้นท่านก็สามารถแจ้งได้เลยว่าอุปกรณ์ตัวใดเสีย

9) ถ้าหากว่าท่านต้องการแจ้งว่าท่านได้ซ่อมอุปกรณ์ที่ชำรุดเรียบร้อยแล้ว ท่านก็ทำตามข้อ 8 หน้าจอจะปรากฏข้อความว่า

1. Good
2. No. Use

ให้ท่านกดเลข 1 เพื่อแจ้งว่าซ่อมเสร็จแล้วสามารถใช้งานได้ตามปกติ

10) ถ้าหากท่านไม่ต้องการใช้งานฟังก์ชั่นอื่นๆ ก็กดเลข 3 ที่ฟังก์ชั่นหลัก เพื่อออกจากระบบ แล้วรอการใช้งานครั้งต่อไป

การใช้งานเครื่องคอมพิวเตอร์ เพื่อติดต่อกับเครื่องอ่านบาร์โค้ด

1) CLICK ที่โปรแกรม Pro. Project remode

2) เครื่องคอมพิวเตอร์ก็จะเข้าสู่ระบบ โดยแสดงหน้าจอดังนี้

รูปที่ จ.4 แสดงหน้าจอคอมพิวเตอร์เมื่อเข้าสู่ระบบ

3) เมื่อต้องการอ่านข้อมูลจากเครื่องอ่านบาร์โค้ด ซึ่งเก็บรายละเอียดที่ใช้งานทั้งหมดให้ท่านClick ที่ Download ข้อมูลจะมาเก็บที่ File Program Project room ซึ่งข้อมูลที่เข้ามาจะถูกเก็บ

ไว้ในชื่อ SC เดือน / วัน / ปี โดยอัตโนมัติ และถ้าดับเบิ้ลคลิก ที่ชื่อ SC อีกครั้ง หน้าจอจะแสดงรายการว่ามีการใช้อุปกรณ์ในห้องกี่ครั้ง

4) เมื่อท่านต้องการตรวจสอบดูว่าในห้องโครงงานมีอุปกรณ์ใดที่ได้แจ้งชำรุดไว้ให้ CLICK ที่ปุ่มค้นหาอุปกรณ์ชำรุด ถ้าหากมีอุปกรณ์ชำรุด หน้าจอจะแสดงโต๊ะที่มีอุปกรณ์ชำรุด ว่าดีหรือเสีย พร้อมบอกรหัสนักศึกษา วันที่ เวลา ที่นักศึกษาผู้นั้นใช้งาน

5) ถ้าหากท่านต้องการตรวจสอบดูว่าในแต่ละโต๊ะ มีนักศึกษามาใช้กี่คนและมีใครใช้บ้าง ใช้กดหมายเลขโต๊ะ ที่ท่านต้องการทราบ หน้าจอมอนิเตอร์ก็จะแสดงข้อมูลภายในโต๊ะทั้งหมด

6) ถ้าต้องการดูสถิติการใช้งานก็ให้ Click ที่ Call Static และทำการเลือก File ที่ต้องการทราบแล้วOpen ที่ File นั้น ท่านก็สามารถตรวจสอบสถิติที่ใช้ทั้งหมดที่ File นั้นได้

7) ถ้าต้องการทราบสถิติเก่าๆ ที่เก็บไว้ก็ Click ที่ Dellec Static

8) ถ้าต้องการเลิกใช้งานก็ Click Exit ได้เลย

บรรณานุกรม

ยืน ภู่วรวรรณ และไพศาล สงวนหมู่. **การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เน็ตเวิร์ค.** กรุงเทพฯ : หจก. เอช-เอน การพิมพ์.2536

สมยศ จุณณะปิยะ. **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51**. กรุงเทพฯ : คณะวิศวกรรมศาตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2539

สุนทร วิทูรสุรพจน์. **การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051**. กรุงเทพฯ : บริษัท เอ็ช. เอ็น. กรุ๊ป จำกัด. 2537

สุวิพล สิทธิชีวภาค. **พื้นฐานแห่งการสื่อสารข้อมูล**. กรุงเทพฯ : ด่านสุทธาการพิมพ์. ม.ป.ป.

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายธรรมนูญ กรเพ็ชร
วันเดือนปีเกิด	10 ธันวาคม พ.ศ. 2520
สถานที่เกิด	จังหวัดอุตรดิตถ์
ภูมิลำเนาเดิม	466/2 ต. ท่าอิฐ ถ. ย่านเสือศิลาอาจ อ. เมือง จ. อุตรดิตถ์
ที่อยู่ปัจจุบัน	466/2 ต. ท่าอิฐ ถ. ย่านเสือศิลาอาจ อ. เมือง จ. อุตรดิตถ์
โทรศัพท์	(055) 414565

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนอนุบาลอุตรดิตถ์
มัธยมตอนต้น	โรงเรียนอุตรดิตถ์
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคอุตรดิตถ์
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีวิทยาเขตนนทบุรี
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
คติพจน์	ชีวิต คือการต่อสู้

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวศิรินันท์ พิลาแสน
วันเดือนปีเกิด	27 เมษายน พ.ศ. 2520
สถานที่เกิด	จังหวัดสกลนคร
ภูมิลำเนาเดิม	310 หมู่ 15 ถ. สกล-กาฬสินธ์ ต. ห้วยยาง อ. เมือง จ. สกลนคร
ที่อยู่ปัจจุบัน	310 หมู่ 15 ถ. สกล-กาฬสินธ์ ต. ห้วยยาง อ. เมือง จ. สกลนคร
โทรศัพท์	(042)714725

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนเชิงชุมราษฎร์นุกูล
มัธยมตอนต้น	โรงเรียนสกลราชวิยานุกูล
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคสกลนคร
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคสกลนคร
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
คติพจน์	ความพยามอยู่ที่ไหน ความสำเร็จอยู่ที่นั้น

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายเสรี ขุนไชย
วันเดือนปีเกิด	5 สิงหาคม พ.ศ. 2521
สถานที่เกิด	จังหวัดนครพนม
ภูมิลำเนาเดิม	48 หมู่ 7 ต. ท่าค้อ อ.เมือง จ. นครพนม
ที่อยู่ปัจจุบัน	48 หมู่ 7 ต. ท่าค้อ อ.เมือง จ. นครพนม
โทรศัพท์	(042)514304

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบ้านนาหลวง
มัธยมตอนต้น	โรงเรียนนครพนมวิทยาคม
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคนครพนม
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคนครพนม
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
คติพจน์	วันนี้ต้องดีกว่าเมื่อวาน

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายเอกพันธ์ ชินกลาง
วันเดือนปีเกิด	4 พฤศจิกายน พ.ศ. 2521
สถานที่เกิด	จังหวัดนครราชสีมา
ภูมิลำเนาเดิม	1 หมู่ 8 ต. หนองงูเหลือม
	อ. เฉลิมพระเกียรติ จ. นครราชสีมา
ที่อยู่ปัจจุบัน	1 หมู่ 8 ต. หนองงูเหลือม
	อ. เฉลิมพระเกียรติ จ. นครราชสีมา
โทรศัพท์	(044)207099
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนมารีวิทยา
มัธยมตอนต้น	โรงเรียนบุญวัฒนา
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคนครราชสีมา
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคนครราชสีมา
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
คติพจน์	ทำวันนี้ให้ดีที่สุด