เครื่องบันทึกเวลาพนักงานโดยใช้รหัสแถบ

BARCODE TIME RECORDING MACHINE

โดย

นาย กวิน	ยะมาลี	36014014
นาย ชาลี	วรกุลพิพัฒน์	36014116

อาจารย์ที่ปรึกษา

ผศ. พลผดุง ผดุงกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชา อิเล็คทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปริญญานิพนธ์ ปีการศึกษา 2539

ภาควิชาอิเล็คทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องบันทึกเวลาพนักงานโดยใช้รหัสแถบ

ผู้จัดทำ

1. นาย กวิน ยะมาลี
2. นาย ชาลี วรกุลพิพัฒน์

อาจารย์ที่ปรึกษา

(ผศ.พลผดุง ผดุงกุล)

เครื่องบันทึกเวลาพนักงานโดยใช้รหัสแถบ

BARCODE TIME RECORDING MACHINE

ผู้จัดทำ

1. นาย กวิน ยะมาลี รหัสประจำตัว 36014014
2. นาย ชาลี วรกุลพิพัฒน์ รหัสประจำตัว 36014116

โครงงานได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

(ผศ.พลผดุง ผดุงกุล)

อาจารย์ที่ปรึกษา

เครื่องบันทึกเวลาพนักงานโดยใช้รหัสแถบ

นาย กวิน ยะมาลี
นาย ชาลี วรกุลพิพัฒน์
ผศ. พลผดุง ผดุงกุล อาจารย์ที่ปรึกษา

บทคัดย่อ

เครื่องบันทึกเวลาพนักงานจากรหัสแถบ ใช้รหัสแถบระบบ 3 ใน 9 มีวิธีการทำงานโดยการลากหัวอ่านไปบนแถบรหัส เพื่อนำมาแปลงเป็นหมายเลขประจำตัวของพนักงาน แล้วแสดงผลออกทางจอ LCD และบันทึกเข้าหน่วยความจำ เพื่อนำไปใช้งานต่อไป ระบบทั้งหมดนี้ ถูกควบคุมการทำงานโดยใช้ ไมโครคอนโทรลเลอร์ตระกูล 8051

BARCODE TIME RECORDING MACHINE

Kawin Yamali

Chalee Vorakulpipat

Asst. Prof. Pholphadung Phadungkul Advisor

Abstract

This *Time Recording Machine* uses "3 of 9" code. This machine functions by having head reader running through bar code to transfer code to "Identification No."and having this identification displayed on LCD and registered in memory for further use.This machine is controlled by Single Chip Microcontroller system 8032 family Architectural.

สารบัญ

	หน้า
บทที่ 1 บทนำ	1
1.1 ข้อดีของรหัสแถบ	1
บทที่ 2 รหัสแถบ	2
2.1 ความเป็นมาของรหัสแถบ	2
2.2 โครงสร้างของ 8032	4
2.2.1 การจัดหน่วยความจำของ 8032	4
2.2.2 การใช้งานพอร์ท	4
2.2.3 การติอต่อกับหน่วยความจำภายนอก	6
2.2.4 ไทม์เมอร์/เคาน์เตอร์	6
2.2.5 อินเทอร์รัพท์	8
2.2.6 การรีเซ็ท	11
บทที่ 3 การออกแบบ	12
3.1 การออกแบบฮาร์ดแวร์	12
3.2 การออกแบบซอฟท์แวร์	16
3.3 การแสดงผลออกทางจอแสดงผล	17
3.4 ส่วนนาฬิกาเวลาจริง (real time clock)	20
3.5 การส่งข้อมูลผ่านพอร์ทสื่อสารอนุกรมในไมโครโปรเซสเซอร์ 8032	21
3.6 การรับข้อมูลจากพอร์ทอนุกรมเพื่อบันทึกเป็นเท็กซ์ไฟล์ (text file)	21
บทที่ 4 ผลการทดลอง	30
4.1 อุปกรณ์ที่ใช้ในการทดลองและบันทึกผล	30
4.2 วิธีการทดลอง	30
4.3 ผลการทดลอง	31
บทที่ 5 สรุปและวิจารณ์ผลการทดลอง	33
เอกสารอ้างอิง	34
กิตติกรรมประกาศ	35
ภาคผนวก ก โปรแกรม	36
ภาคผนวก ข ข้อมูลวงจรรวม	58
ภาคผนวก ค ข้อมูลจำเพาะของสายสัญญาณหัวอ่าน	78

สารบัญรูป

หน้า

รูปที่ 2.1 รหัสแถบชนิด 3 ใน 9 4
รูปที่ 2.2 โครงสร้างของพอร์ท 0 และ 2 5
รูปที่ 2.3 โครงสร้างของพอร์ท 1 5
รูปที่ 2.4 โครงสร้างของพอร์ท 3 5
รูปที่ 2.5 ไทม์เมอร์โหมด 0 6
รูปที่ 2.6 ไทม์เมอร์โหมด 2 7
รูปที่ 2.7 ไทม์เมอร์โหมด 7
รูปที่ 2.8 แหล่งกำเนิดสัญญาณอินเทอร์รัพท์ 8
รูปที่ 2.9 อินเทอร์รัพท์เอนาเบิลรีจิสเตอร์ 9
รูปที่ 2.10 ไดอะแกรมเวลาของการตอบสนองการอินเทอร์รัพท์ 10
รูปที่ 3.1 บล็อคไดอะแกรมของวงจร 12
รูปที่ 3.2 สัญญาณอินเทอร์รัพท์ 13
รูปที่ 3.3 วงจรเอกซ์คลูซีฟออร์สัญญาณอินเทอร์รัพท์ 13
รูปที่ 3.4 วงจรไมโครโปรเซสเซอร์ 15
รูปที่ 3.5 จอแสดงผล 18
รูปที่ 3.6 แผนภูมิรูปภาพแสดงขั้นตอนการทำงานของเครื่องบันทึกเวลา พนักงาน 22
รูปที่ 3.7 แผนภูมิรูปภาพแสดงการกำหนดค่าเริ่มต้น 23
รูปที่ 3.8 แผนภูมิรูปภาพแสดงการเก็บค่าเวลาของแถบและการตรวจสอบ โอเวอร์โฟล 24
รูปที่ 3.9 แผนภูมิรูปภาพแสดงการแปลงค่าเวลาเป็นรหัส 3 ใน 9 25
รูปที่ 3.10 แผนภูมิรูปภาพแสดงการแสดงผลและการเก็บข้อมูล 26
รูปที่ 3.11 แผนภาพแสดงการชดเชยเวลา 27
รูปที่ 3.12 แผนภาพแสดงการแปลงข้อมูลเป็นเลขฐานสอง 28
รูปที่ 3.13 แผนภาพแสดงการส่งข้อมูลให้ PC 29
รูปที่ 4.1 หัวอ่านรหัสแถบ 30
รูปที่ 4.2 สัญญาณจากหัวอ่านและสัญญาณเอาท์พุทของเอกซ์คลูซีฟออร์ซึ่ง วัดโดยสตอเรจสโคป 31
รูปที่ 4.3 เท็กซ์ไฟล์ที่ได้จากการส่งข้อมูลไปยังคอมพิวเตอร์ 32

สารบัญตาราง

	หน้า
ตารางที่ 2.1 อักขระของรหัสแถบชนิด 3 ใน 9	3
ตารางที่ 2.2 ลำดับความสำคัญของการอินเทอร์รัพท์	9
ตารางที่ 2.3 ตำแหน่งของเวกเตอร์อินเทอร์รัพท์	10
ตารางที่ 2.4 ค่าภายในรีจิสเตอร์ SFR หลังการรีเซ็ท	11
ตารางที่ 4.1 ผลการแสดงค่าออกทางหน้าจอเมื่อลากหัวอ่านด้วยความเร็วต่างกัน	32

บทที่ 1
บทนำ

จำนวนประชากรของประเทศ ซึ่งมีจำนวนเพิ่มขึ้นในแต่ละวัน ทำให้ระบบการค้า การจัดเก็บสินค้าในคลังสินค้า การขอยืมหนังสือ และ การเข้าใช้บริการต่างๆ ในปัจจุบัน จำเป็นจะต้องพัฒนาระบบการทำงาน เพื่อให้สอดคล้องกับปริมาณการใช้บริการที่เพิ่มขึ้นเป็นเงาตามตัว

ระบบหนึ่งซึ่งเป็นที่ยอมรับกันในปัจจุบันว่าสามารถให้ความสะดวกสบายในการทำงาน คือ ระบบบรหัสแถบ (Barcode) ระบบนี้จะเป็นการป้อนข้อมูลเข้าไปในคอมพิวเตอร์ เพื่อแสดงรายละเอียดของการให้บริการนั้น หรือ รายละเอียดของสินค้าบนจอมอนิเตอร์ หรือ ส่วนแสดงผลอื่น ๆ

1.1 ข้อดีของรหัสแถบ

- ราคาถูก
- ข้อผิดพลาดน้อย
- ทนทานเพราะไม่มีส่วนประกอบทางอิเลคทรอนิคส์
- ป้อนข้อมูลได้รวดเร็วกว่าเมื่อเทียบกับการป้อนข้อมูลทางแป้นพิมพ์ (keyboard)

เครื่องคอมพิวเตอร์เป็นเพียงส่วนประมวลผลข้อมูลที่ได้จากหัวอ่านของรหัสแถบเท่านั้น โดยลักษณะของรหัสแถบที่ดี ควรมีความผิดพลาดน้อย ของทั้งรหัสแสดงการเริ่มต้น การจบรหัสแถบ และข้อความภายในควรชัดเจน ไม่มีการชำรุดของแถบ

บทที่ 2
รหัสแถบ

2.1 ความเป็นมาของรหัสแถบ

ในปี ค.ศ. 1949 สหรัฐอเมริกาได้ออกสิทธิบัตรรับรองรหัสแถบที่เรียกว่า "Circular Bar Code" ต่อมาในปี ค.ศ. 1960 ก็มีการรับรองรหัสแถบแบบที่เรียกว่า "Rail Identification Symbol" หลังจากนั้นเป็นต้นมา เทคนิคของรหัสแถบรูปแบบต่าง ๆ ก็มีมากขึ้น และเริ่มใช้งานจริงจังเมื่อปี ค.ศ. 1970 เมื่อคณะกรรมการบริหารด้านห้างสรรพสินค้าของสหรัฐอเมริกา ได้นำรหัสที่เรียกว่า "Universal Product Code" หรือ "UPC" ซึ่งเป็นรหัสที่ใช้กันแพร่หลายในสหรัฐอเมริกา และ ยุโรปตั้งแต่ ค.ศ. 1973 และ ค.ศ. 1977 ตามลำดับ

การใช้งานในด้านอื่น ๆ เริ่มตั้งแต่ปี ค.ศ. 1980 เป็นต้นมา เมื่อหน่วยป้องกันประเทศของสหรัฐอเมริกา ใช้เพื่อเช็คอุปกรณ์เครื่องมือและยุทโธปกรณ์คงคลัง ขณะที่โรงงานอุตสาหกรรมนำไปใช้เพื่องานต่าง ๆ มากขึ้น แต่คนทั่วไปเริ่มคุ้นเคยกับรหัสแถบเป็นอย่างดีจากรหัสสินค้าและการชำระเงินที่ใช้คอมพิวเตอร์รวมราคาออกมาจากการอ่านรหัสแถบบนสินค้าเหล่านั้น จากความสะดวกเหล่านี้ทำให้สามารถลดพนักงาน ณ จุดนี้ลงไปได้ ในปี ค.ศ.1981 มีห้างสรรพสินค้ามากกว่า 4,000 แห่ง ในสหรัฐอเมริกาและแคนาดา ใช้รหัสแถบในธุรกิจนี้ นอกจากนี้ยังใช้กับกิจการอื่น เช่น ห้องสมุด บริการสุขภาพ งานเอกสาร งานการทหาร การวิจัย การผลิตสินค้า เป็นต้น

สำหรับ รหัสแถบที่เราใช้กันอยู่ในปัจจุบัน สามารถสร้างขึ้นมาจาก ความกว้าง และ ความแคบของแถบดำและแถบขาว โดยมีสิ่งที่ควรระวังในการสร้างรหัสแถบซึ่งเกี่ยวกับวัสดุที่ใช้ คือ หมึกซึ่งอาจจะดูดกลืนแสงที่ส่งออกมาจากหัวอ่าน โดยเฉพาะอย่างยิ่งหัวอ่านที่เป็นหลอด LED สีแดง จะใช้รหัสแถบที่มีขนาดความกว้างของแถบที่มีขนาดเล็กมากไม่ได้

หลักการของรหัสแถบ

รหัสแถบเป็นการแทนข้อมูลที่เป็นรหัสเลขฐานสอง (binary code) ในรูปแบบของแถบดำและแถบขาว ที่มีความกว้างและความแคบของแถบแต่ละแถบมีขนาดไม่เท่ากัน กล่าวคือ แถบที่มีความกว้างจะแทนค่าเป็น "1" และ แถบที่ความแคบจะแทนค่าเป็น "0" แถบ ขาว - ดำ จะมีชื่อเรียก คือ

แถบสีดำที่กว้าง เรียกว่า Wide Bar
แถบสีดำที่แคบ เรียกว่า Narrow Bar
แถบสีขาวที่กว้าง เรียกว่า Wide Space
แถบสีขาวที่แคบ เรียกว่า Narrow Space

การอ่านรหัสแถบ

ในการอ่านรหัสแถบ จะใช้หลักการเปลี่ยนรหัสแถบที่ใช้ไปเป็นรหัสแอสกี้ โดยอาศัยความแตกต่างกันระหว่างแถบและช่องว่าง โดยที่ช่องว่างจะมีการสะท้อนของแสงได้มากกว่าบริเวณที่เป็นแถบสีดำ (เนื่องจากสีดำดูดกลืนแสงได้มากกว่าสีขาว) ตัวอ่าน จะประกอบด้วย ตัวกำเนิดแสงที่ส่งผ่าน

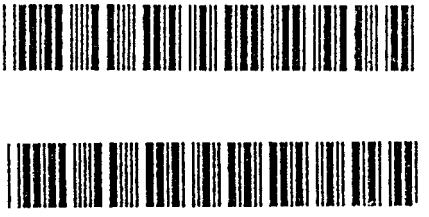
เลนซ์ โดยถูกจัดให้มีจุดรวมแสงที่เล็กมาก กับตัวรับแสงที่มีความไวสูง ทั้ง 2 อย่างนี้ จะบรรจุในตัวอ่านเดียวกัน ซึ่งมีหลายรูปแบบ แต่แบบที่เป็นพื้นฐานที่สุด คือ ตัวอ่านลักษณะปากกา (Wand type) ตัวอ่านจะถูกลากผ่านรหัสแถบ ในขณะที่ตัวกำเนิดแสงจะทำให้แสงกระทบกับรหัสแถบและสะท้อนกลับไปยังตัวรับแสง ทำให้เกิดสภาวะลอจิก "0" หรือ สภาวะลอจิก "1" ตลอดความกว้างของทุกแถบแล้วจะเทียบกับแพตเทิร์นที่ได้กำหนดไว้แล้ว ในตัวอ่านรหัสแถบจะใช้ตัวกำเนิดแสงสีแดงหรือขาว โดยส่วนใหญ่จะนิยมใช้แสงสีแดงมากกว่า เนื่องจากแสงสีขาวต้องการพลังงานและความเข้มของแสงสูงกว่าแสงสีแดง

รหัสแถบชนิด 3 ใน 9

เป็นรหัสแถบที่ใช้แทนอักขระได้ 44 อักขระ โดยเป็นตัวอักษรภาษาอังกฤษ 26 ตัวอักษร เป็นตัวเลข 0 - 9 อีก 10 ตัว และ อักขระพิเศษ 8 ตัว เป็นรหัสที่พัฒนามาจากรหัสแถบชนิด 2 ใน 5 โดยนำแถบดำ 5 แถบ และ แถบขาวอีก 4 แถบ มาใช้รวมกันเป็น 9 แถบต่อ 1 อักขระ

Characters	Bars	Spaces	Characters	Bars	Spaces
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	01001	0001	%	00000	0111

ตารางที่ 2.1 อักขระของรหัสแถบชนิด 3 ใน 9



รูปที่ 2.1 รหัสแถบชนิด 3 ใน 9

โดยแถบดำ มีแถบกว้าง 2 แถบ และแถบแคบ 3 แถบ แถบขาว หรือ ช่องว่างประกอบด้วยแถบกว้าง 1 แถบ และแถบแคบ 3 แถบ ทั้งแถบดำ และ แถบขาวรวมกันเป็น 9 แถบ ใน 9 แถบนี้ จะมีแถบกว้างซึ่งมีสถานะทางลอจิก เป็น "1" อยู่ 3 แถบ มีรหัสเริ่มต้นและรหัสสิ้นสุด เป็นอักขระ "*" เหมือนกัน ซึ่งมีรหัสเลขฐานสองเป็นรหัสแถบคือ แถบดำ 00110 และ แถบขาว 1000 ข้อดีของรหัสชนิดนี้ คือ สามารถใช้งานได้กว้างขวางมากกว่าชนิดอื่น

2.2 โครงสร้างของไมโครโปรเซสเซอร์ 8032

ลักษณะของไมโครโปรเซสเซอร์ 8032 ที่นำมาใช้งาน

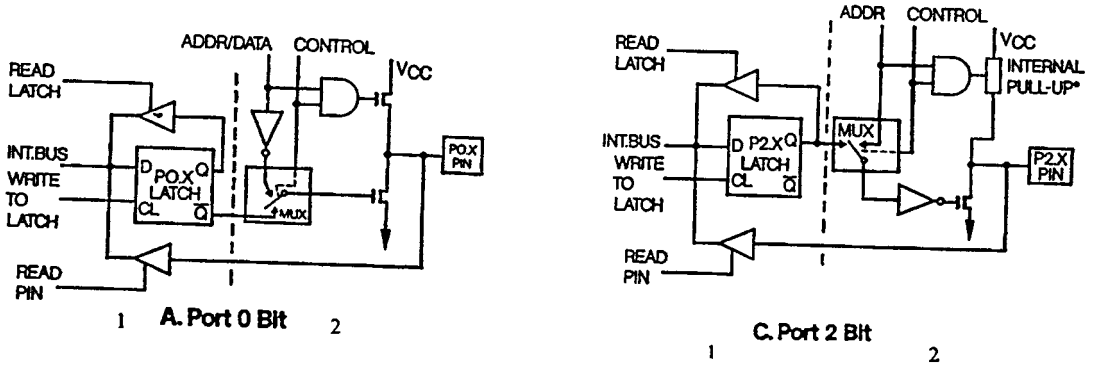
- หน่วยประมวลผลกลาง 8 บิท
- พอร์ท I/O 4 พอร์ท
- อ้างตำแหน่งหน่วยความจำได้ 64 กิโลไบท์
- มีไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิท 3 ตัว
- 8 อินเทอร์รัพพ์ซอร์ส , 6 อินเทอร์รัพพ์เวกเตอร์
- ทำงานแบบ ฟูลดูเพล็ก ในขณะส่งข้อมูลแบบอนุกรม และ แบบขนาน

2.2.1 การจัดหน่วยความจำของ 8032

ไมโครโปรเซสเซอร์8032 แบ่งหน่วยความจำออกเป็น 2 ส่วน สามารถอ้างตำแหน่งหน่วยความจำได้ 64 กิโลไบท์ หน่วยความจำทั้ง 2 ส่วน คือ หน่วยความจำโปรแกรม และ หน่วยความจำข้อมูล และยังมีแรมภายในอีก 256 ไบท์

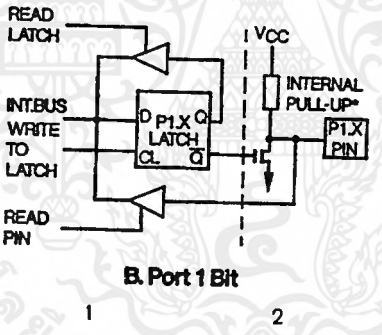
2.2.2 การใช้งานพอร์ท

- พอร์ท 0 และพอร์ท 2 ใช้ในการติดต่อกับหน่วยความจำโปรแกรมภายนอกหรือหน่วยความจำข้อมูล โดยพอร์ท 0 ทำหน้าที่ส่ง แอดเดรส 8 บิทล่าง และ รับ-ส่งข้อมูล 8 บิทไปยังหน่วยความจำภายนอก ส่วนพอร์ท 2 ส่งแอดเดรส 8 บิทบนเท่านั้น ในลักษณะนี้จะต้องมีการแลทช์แอดเดรส 8 บิทล่างเพื่อใช้ชี้ตำแหน่งของหน่วยความจำข้อมูลที่ต้องการ ใน 1 ค่าของรอบเวลาทำงาน จะมีการแลทช์แอดเดรส ด้วยสัญญาณ ALE ถึง 2 ครั้ง ซึ่งโครงสร้างภายในของพอร์ท 0 และพอร์ท 2 แสดงในรูปที่ 2.2



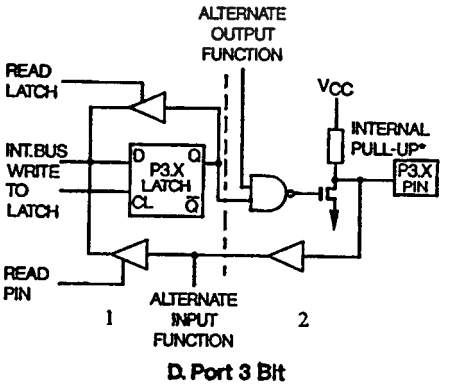
รูปที่ 2.2 โครงสร้างของพอร์ท 0 และ 2

- **พอร์ท 1** ภายในมีดีฟลิปฟลอปต่ออยู่กับขาเบสของทรานซิสเตอร์ ซึ่งขาอีมิตเตอร์ ต่ออยู่กับกราวนด์ และขาคอลเลคเตอร์ ต่ออยู่กับขา I/O ของพอร์ท ซึ่งเดิมเมื่อ เอาท์พุทของดีฟลิปฟลอปเป็นสถานะสูงอยู่ ทำให้ทรานซิสเตอร์ทำงาน จึงทำให้สัญญาณที่คอลเลคเตอร์มีสภาวะต่ำ ซึ่งขาคอลเลคเตอร์ต่ออยู่กับขาI/O ของพอร์ททำให้มีสภาวะต่ำอยู่ตลอดด้วย จะทำให้การอ่านข้อมูลจากขาเกิดการผิดพลาดได้โดยจะมีสภาวะต่ำอยู่ตลอดเวลาไม่ว่าอินพุทที่เข้ามาจะมีสภาวะเป็นเช่นไร



รูปที่ 2.3 โครงสร้างของพอร์ท 1

- **พอร์ท 3** ใช้สำหรับรับส่งข้อมูลได้และยังใช้เป็นขาอินพุทของ SFR (Special Function Register) และ



รูปที่ 2.4 โครงสร้างของพอร์ท 3

เป็นขา I/O ของพอร์ทอนุกรมด้วย โครงสร้างภายในของพอร์ท 1 และ พอร์ท 3 แสดงดังรูปที่ 2.3 และ2.4 ตามลำดับ

ข้อแตกต่างของพอร์ท0ก็คือไม่มีตัวต้านทานพูลอัพภายใน ส่วน FET ตัวบนจะใช้เพียงให้เอาท์พุท เป็น 1 ในขณะที่ติดต่อกับหน่วยความจำภายนอก ในกรณีอื่น FET ตัวนี้จะถูกทำให้คัทออฟ ฉะนั้นพอร์ท 0 ที่ใช้เป็ นพอร์ทเอาท์พุทจะเป็นแบบโอเพนเดรนการเขียนค่า 0FFh ไปที่พอร์ท 0จะทำให้ FET ทั้ง 2 ตัวหยุดทำงาน เป็นผลให้เกิดสภาวะอิมพิแดนซ์ ขึ้นได้

2.2.3 การติอต่อกับหน่วยความจำภายนอก

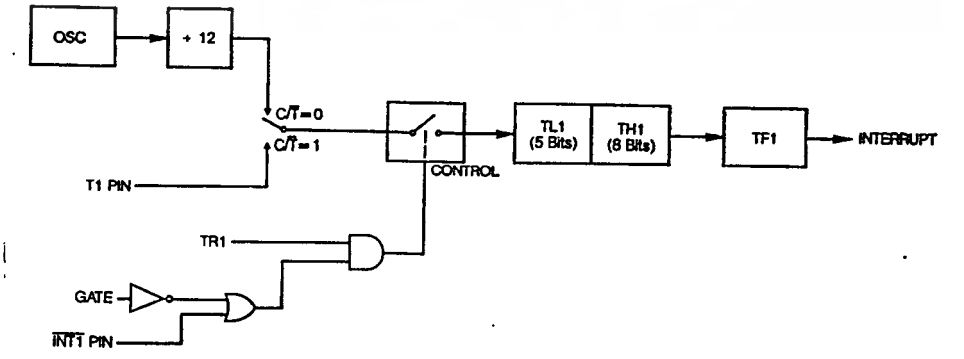
การติดต่อกับหน่วยความจำภายนอกทำได้ 2 แบบ คือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล โดยการติดต่อกับหน่วยความจำโปรแกรมจะใช้สัญญาณ PSEN (Program Store Enable) โดยสัญญาณ PSEN จะแอคทีฟ 2 ครั้งในทุก ๆ ค่าของรอบเวลาทำงาน (ยกเว้นคำสั่ง movx) เป็นสัญญาณอ่านโปรแกรมภายนอก ส่วนการติดต่อกับหน่วยความจำข้อมูลจะใช้สัญญาณ RD และ WR การติดต่อกับข้อมูลภายนอกจะใช้ได้ทั้ง 16 บิท ด้วยคำสั่ง MOVX A, @DPTR หรือ 8 บิท ด้วยคำสั่ง MOVX A,@R1 เมื่อต้องใช้แอดเดรส 16 บิทโดยไบท์สูงจะส่งออกทางพอร์ท 2

2.2.4 ไทม์เมอร์/เคาน์เตอร์

ไมโครโปรเซสเซอร์ 8032 มี ไทม์เมอร์/เคาน์เตอร์ อยู่ 3 ตัว โดยสัญญาณอินพุทที่ป้อนให้นั้นทำงานที่ขอบขาลง คือต้องเป็นสัญญาณสูง 1 ค่าของรอบเวลาทำงานและเป็นต่ำ 1 ค่าของรอบเวลาทำงาน ฉะนั้นความถี่สูงสุดที่ เคาน์เตอร์จะนับได้นั้นประมาณ 1/24 ของความถี่ออสซิเลเตอร์ การทำงานของ ไทม์เมอร์/เคาน์เตอร์ แบ่งเป็น 2 โหมด คือ

- โหมด 0

การทำงานในโหมดนี้ รีจิสเตอร์ TLx ถูกใช้เพียง 5 บิท และ THx ใช้ 8 บิท ทั้งหมดรวมเป็น 13 บิท ในโหมดนโดยการนับจากค่าที่ทุกบิทเป็นสูงไปจนทุกบิทเป็นต่ำจะเกิดโอเวอร์โฟล และ จะให้สัญญาณอินเทอร์รัพท์โดยการเซ็ท TF0 หรือ TF1 โดยการเลือกโหมดนั้นจะกำหนดได้จากรีจิสเตอร์ TMOD ซึ่ง ไทม์เมอร์/เคาน์เตอร์ จะทำงานได้ก็ต่อเมื่อ TR x = 1 และ GATE = 0 หรือ INTx = 1 ถ้าเซ็ท



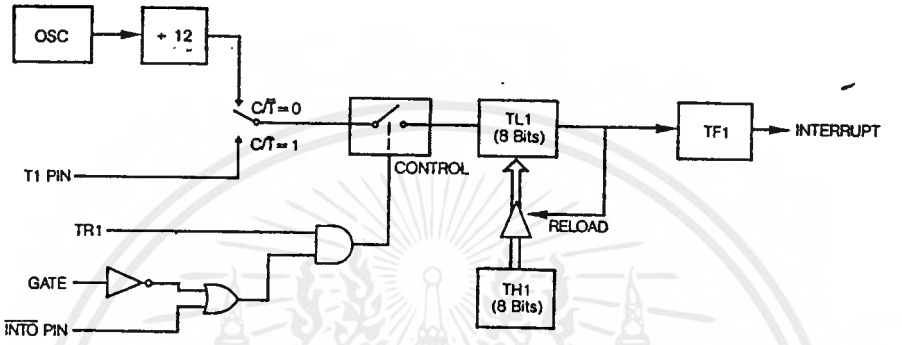
รูปที่ 2.5 ไทม์เมอร์โหมด 0

GATE = 1 ผลคือ ไทม์เมอร์/เคาน์เตอร์ จะถูกควบคุมด้วยสัญญาณ INTx จากภายนอก ประโยชน์ในการทำงานแบบนี้คือ ใช้วัดความกว้างของพัลส์จากอินพุทภายนอก TRx เป็นบิทควบคุมซึ่งอยู่ใน TCON

- โหมด 1

การทำงานเหมือนโหมด 0 ยกเว้น รีจิสเตอร์ที่ใช้จะเป็น 16 บิท

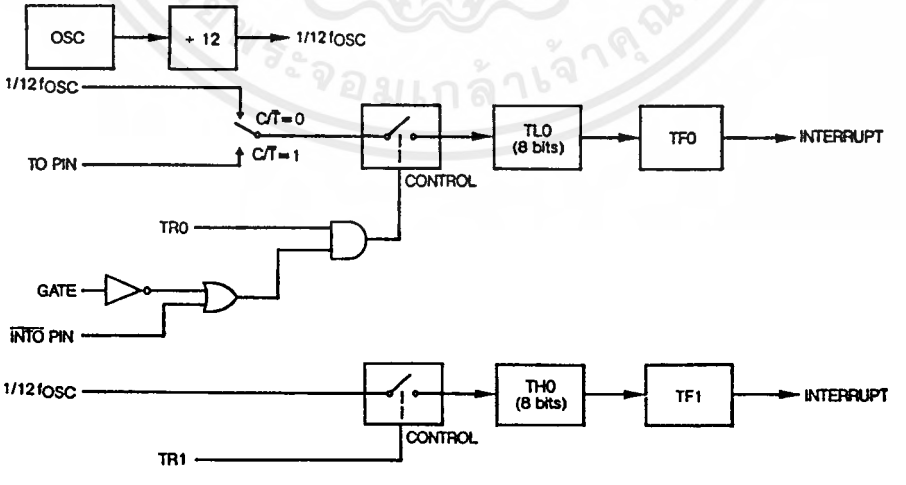
- โหมด 2



รูปที่ 2.6 ไทม์เมอร์โหมด 2

ในโหมด 2 รีจิสเตอร์จะเป็นแบบ 8 บิท โดยที่ TLx จะสามารถโหลดข้อมูลจาก THx ได้ใหม่ (auto reload) เมื่อเกิดโอเวอร์โฟลจาก TLx โดยที่ค่าใน THx จะไม่ถูกเปลี่ยนแปลง การทำงานอื่นๆ จะเหมือนโหมด 0

- โหมด 3

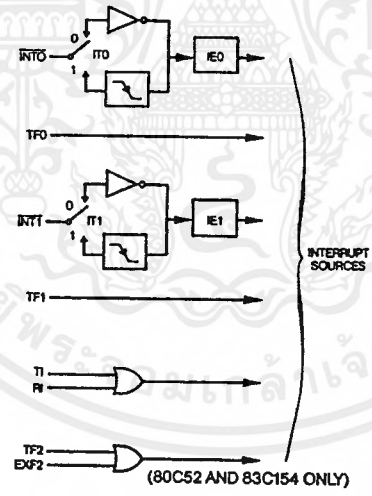


รูปที่ 2.7 ไทม์เมอร์โหมด 3

ในโหมดนี้จะแยก TL0 และ TH0 ออกใช้อย่างอิสระ จะใช้บิทควบคุมคือ GATE, TR0, INT0 และ TF0 ส่วน TH0 ถูกใช้เป็นไทม์เมอร์นับค่าของรอบเวลาทำงาน และรับช่วงการใช้ TR1 และ TF1 ของไทม์เมอร์ 1 ในโหมด 3 แล้วไทม์เมอร์ 1 จะสามารถสลับใช้ระหว่างโหมด 3 และ โหมดอื่นๆ ได้ หรือ ใช้เป็นบอร์ดเรทเจนเนอเรเตอร์

2.2.5 อินเทอร์รัพท์

ไมโครโปรเซสเซอร์8032 จะมี 8 แหล่งกำเนิดสัญญาณอินเทอร์รัพท์ (8 Interrupt Source) และมี 6 เวกเตอร์อินเทอร์รัพพ์ ขาเอกซ์เทอนอลอินเทอร์รัพพ์มีเพียง 2 ขา คือ ขา INT0 และ ขา INT1 ซึ่งสามารถโปรแกรมให้ทำงานที่ เลเวลแอกทีฟ หรือ ทรานซิชันแอกทีฟ (ทริกขอบ) ขึ้นอยู่กับบิท IT0 และ IT1 ซึ่งอยู่ใน TCON แฟลกที่กำเนิดสัญญาณอินเทอร์รัพท์ที่แท้จริงคือ IE0 และ IE1 ใน TCON เมื่อมีสัญญาณอินเทอร์รัพท์จากภายนอก แฟลกที่กำเนิดสัญญาณอินเทอร์รัพท์จะถูกเซ็ทหรือเคลียร์โดยฮาร์ดแวร์และ CPU จะกระโดดไปทำงานที่ เซอวิสรูทีน โดยที่ต้องโปรแกรมให้รับการอินเทอร์รัพท์แบบทริกขอบของสัญญาณ ถ้าชนิดของอินเทอร์รัพท์ถูกเซ็ท ให้ทำงานโดยระดับของสัญญาณแฟลกของอินเทอร์รัพท์ต้องเคลียร์โดยซอฟท์แวร์ภายใน เซอวิสรูทีน ของอินเทอร์รัพท์นั้น



รูปที่ 2.8 แหล่งกำเนิดสัญญาณอินเทอร์รัพท์

-อินเทอร์รัพท์ของ ไทม์เมอร์ 0 ไทม์เมอร์ 1 เกิดขึ้นโดยแฟลก TF0 และ TF1 เมื่อเกิดอินเทอร์รัพท์ขึ้น แฟลกจะถูกเคลียร์โดยฮาร์ดแวร์เมื่อ CPU กระโดดไปทำงานที่ เซอวิสรูทีน

-อินเทอร์รัพท์ของพอร์ทอนุกรม เกิดขึ้นจากด้านรับ หรือด้านส่งข้อมูลโดยที่โปรแกรมต้องตรวจเช็คว่าเป็นการอินเทอร์รัพท์จากด้านรับ (RI) หรือ ด้านส่ง (TI) และจะต้องทำการเคลียร์แฟลกอินเทอร์รัพท์ด้วยซอฟท์แวร์

แหล่งกำเนิดสัญญาณอินเทอร์รัพท์แต่ละตัวสามารถจะทำการดิสเอเบิลหรือเอนาเบิลโดยการเซ็ต หรือ เคลียร์บิทที่อยู่ในรีจิสเตอร์ IE คือ บิทที่ 7 (EA) ฉะนั้นเมื่อต้องการใช้อินเทอร์รัพท์ต้องไม่ลืมที่จะเซ็ตบิท EA ด้วย หลังจากก็ทำการเอนาเบิลสัญญาณอินเทอร์รัพท์ที่ต้องการ

(MSB) (LSB)

$\overline{EA}$	X	ET2	ES	ET1	EX1	ET0	EX0

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

รูปที่ 2.9 อินเทอร์รัพพ์เอนาเบิลรีจิสเตอร์

ลำดับความสำคัญของการอินเทอร์รัพท์

แหล่งสัญญาณอินเทอร์รัพท์แต่ละสัญญาณสามารถโปรแกรมได้ว่าให้เป็นลำดับความสำคัญสูง หรือ ลำดับความสำคัญต่ำ โดยที่ลำดับความสำคัญต่ำจะถูกอินเทอร์รัพท์ด้วยสัญญาณอินเทอร์รัพท์ที่มีความสำคัญสูงกว่า และ ระดับความสำคัญสูงจะไม่ถูกอินเทอร์รัพท์โดยสัญญาณอินเทอร์รัพท์ที่มีความสำคัญต่ำกว่า ถ้ามีการอินเทอร์รัพท์ด้วยลำดับความสำคัญเท่ากันมากกว่า 1 สัญญาณหน่วยประมวลผลกลาง จะทำการตรวจ (polling) และ ตัดสินใจว่าจะให้บริการกับสัญญาณอินเทอร์รัพท์ตัวใด ดังนั้นในแต่ละลำดับความสำคัญ ยังมีการจัดลำดับความสำคัญไว้อีก ดังรายละเอียดต่อไปนี้

SOURCE	Priority within level
IE0	Highest
TF0	
IE1	
TF1	
R1 , T1	Lowest

ตารางที่ 2.2 ลำดับความสำคัญของการอินเทอร์รัพท์

ในตารางลำดับความสำคัญข้างบนนี้ ในกรณีของสัญญาณอินเทอร์รัพท์ ที่มีระดับความสำคัญเท่ากันมากกว่า 1 สัญญาณ

การทำงานของสัญญาณอินเทอร์รัพท์

แฟลกของสัญญาณอินเทอร์รัพท์จะถูกสุ่ม (sampling) ใน S5P2 ของทุก ค่าของรอบเวลาทำงาน และจะทำการตรวจ (polling) การอินเทอร์รัพท์จาก 5 แหล่งสัญญาณในค่าของรอบเวลาทำงานที่ผ่านมาแล้ว จะมีการเรียกไปยังส่วนของโปรแกรมการอินเทอร์รัพท์ หากว่าไม่ถูกขัดขวางด้วยสภาวะใดสภาวะหนึ่งต่อไปนี้

1. กำลังทำคำสั่งอยู่ในโปรแกรมบริการอินเทอร์รัพท์มีความสำคัญเท่ากันหรือสูงกว่า
2. ไม่ใช่รอบสุดท้ายของคำสั่งที่กำลังปฏิบัติอยู่
3. คำสั่งที่ปฏิบัติอยู่นั้นคือ RETI

ในข้อ 2. เพื่อเป็นการประกันว่า คำสั่งถึงรอบสุดท้ายแล้ว จะไม่เกิดการอินเทอร์รัพท์จนกว่าจะปฏิบัติคำสั่งนั้นจบเสียก่อน

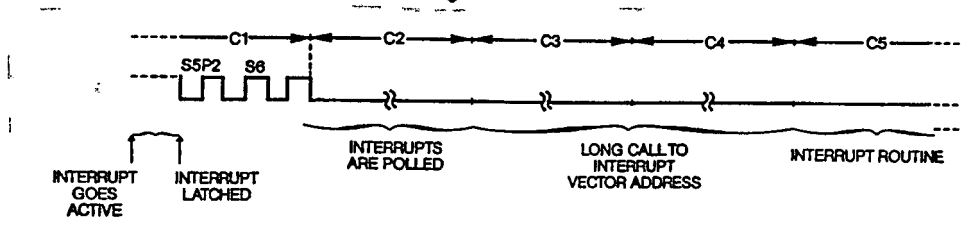
ในข้อ 3. กรณีที่กำลังทำคำสั่ง RETI หน่วยประมวลผลจะส่งแอดเดรสคืนให้โปรแกรมเคาน์เตอร์หลังจากคำสั่ง RETI และ ปฏิบัติอีกหนึ่งคำสั่งในโปรแกรมหลัก ต่อจากนั้นจึงทำการตอบสนองการอินเทอร์รัพท์

SOURCE	Vector Address
IE0	0003h
TF0	000Bh
IE1	0013h
TF1	001Bh
R1 , T1	0023h

ตารางที่ 2.3 ตำแหน่งของ เวกเตอร์อินเทอร์รัพท์

เวลาในการตอบสนองการอินเทอร์รัพท์

ถ้าการอินเทอร์รัพท์จากภายนอกเกิดขึ้นในสภาวะปกติหน่วยประมวลผลกลางจะใช้เวลาตั้งแต่การเซ็ทอินเทอร์รัพท์แฟลกตรวจสอบ (polling) จนถึงกระโดดไปทำงานในรูทีนย่อย อย่างน้อยที่สุด 3 รอบรวมกับเวลาของการทำคำสั่งในเซอร์วิสรูทีน



รูปที่ 2.10 ไดอะแกรมเวลาของการตอบสนองการอินเทอร์รัพท์

2.2.6 การรีเซ็ท

สัญญาณรีเซ็ทเป็นสัญญาณอินพุททางขา 9 การรีเซ็ทจะสมบูรณ์ต้องรักษาระดับสภาวะสูงของสัญญาณ อย่างน้อย 2 ค่าของรอบเวลาทำงาน (24 คาบออสซิลเลเตอร์) การรีเซ็ทภายในตัวหน่วยประมวลกลาง จะเริ่มในระหว่างรอบ ที่ 2 นับตั้งแต่ขา RST เป็นสูง การรีเซ็ทจะมีผลต่อรีจิสเตอร์ ดังตารางที่ 2.4

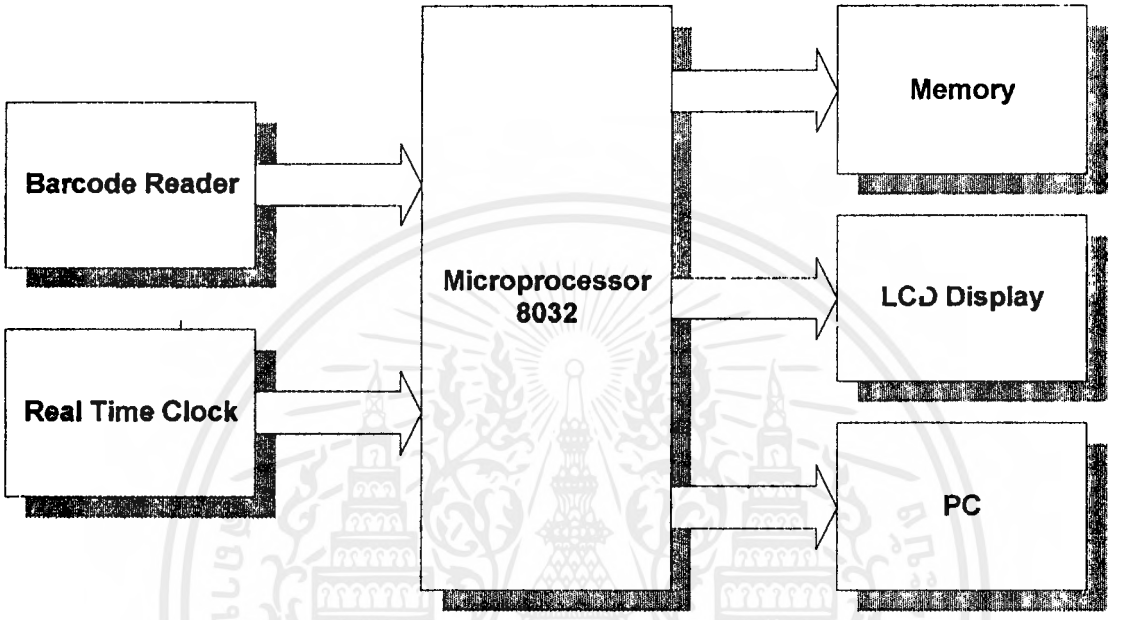
Register	Content
PC	00h
ACC	00h
B	00h
PSW	00h
SP	07h
DPTR	00h
P0 - P3	FFh
IP	XX000000
IE	0X000000
TMOD	00h
TCON	00h
TH0	00h
TL0	00h
TH1	00h
TL1	00h
SCON	00h
SBUF	00h
PCON	00h

ตารางที่ 2.4 ค่าภายในรีจิสเตอร์ SFR หลังการรีเซ็ท

บทที่ 3
การออกแบบ

3.1 การออกแบบฮาร์ดแวร์

ฮาร์ดแวร์สามารถแบ่งออกเป็นบล็อกไดอะแกรมดังรูปที่ 3.1



รูปที่ 3.1 บล็อคไดอะแกรมของวงจร

ในส่วนของอินพุทจะต้องสามารถนำสัญญาณที่เป็นข้อมูลมาเก็บไว้ภายในหน่วยความจำ โดยส่วนหน่วยประมวลผลกลางจะเป็นตัวนำสัญญาณอินพุทไปจัดเก็บไว้ในหน่วยความจำและสามารถนำข้อมูลจากหน่วยความจำมาประมวลผลหรือดีโค้ดและแสดงผลที่จอแสดงผล

สัญญาณอินพุทจากสัญญาณภายนอก

สัญญาณอินพุท เป็นข้อมูลที่มีลักษณะเป็นดิจิตอล 2 สถานะ (high,low)

1. สัญลักษณ์ของสัญญาณ เป็นสัญญาณ 2 สถานะที่แถบดำเป็น "0" หรือ ต่ำ แถบขาวเป็น "1" หรือ สูง ซึ่งมีลักษณะเป็นพัลส์แตกต่างกันออกไปตามลักษณะของแถบบาร์โค้ด โดยแต่ละแถบของบาร์โค้ดจะมีความกว้างไม่เท่ากัน ทำให้สัญญาณพัลส์ที่ออกมามีความกว้างไม่เท่ากันด้วย (นอกจากนี้การได้ข้อมูลที่แน่นอน จะขึ้นอยู่กับอัตราเร็วของการลากหัวอ่านบาร์โค้ดด้วย)

2. จำนวนข้อมูล จำนวนข้อมูลหรือจำนวนพัลส์จะขึ้นอยู่กับจำนวนแถบของบาร์โค้ด

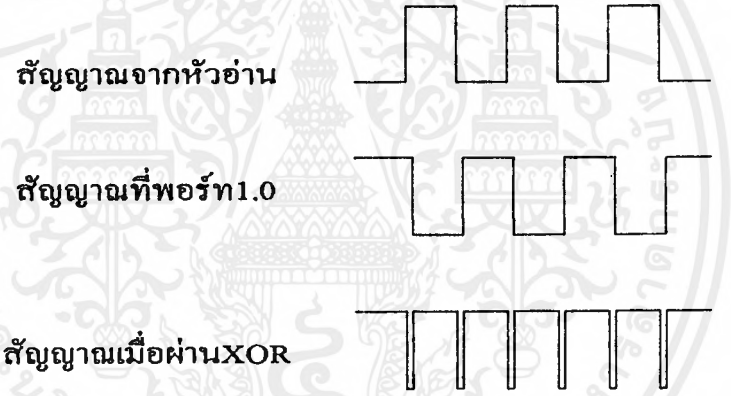
จุดประสงค์ของการออกแบบส่วนอินพุท

1. ต้องการเก็บจำนวนแถบของข้อมูล
2. ต้องการเก็บขนาดความกว้างของแต่ละข้อมูล ทั้งแถบขาวและดำ โดยใช้ฐานเวลาของไมโครโปรเซสเซอร์ 8032 เป็นตัวเปรียบเทียบ

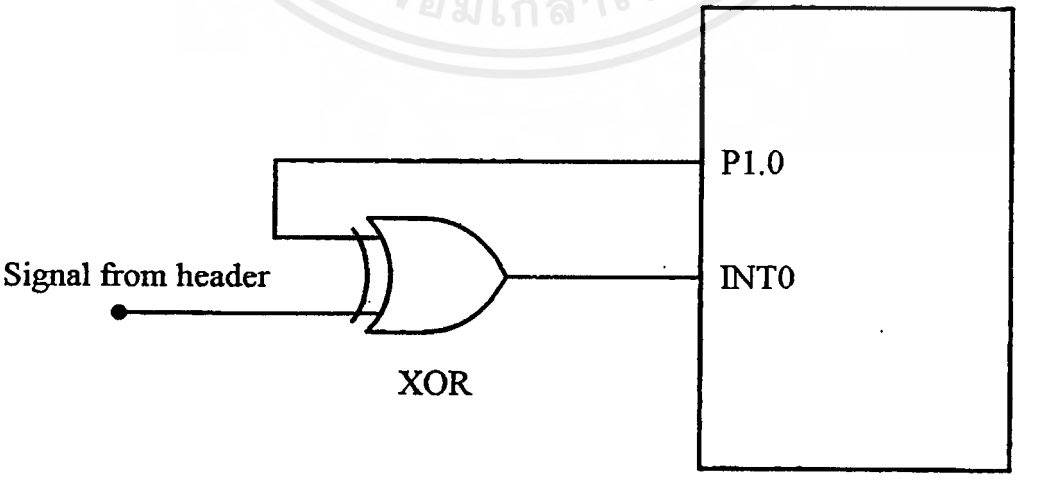
การออกแบบ

1. การเก็บจำนวนของแถบ เนื่องจากอินพุทไม่มีจำนวนข้อมูลและอัตราการส่งข้อมูลที่คงที่แน่นอน จึงต้องทำการวัดคาบเวลาของสัญญาณโดยใช้รีจิสเตอร์ไทม์เมอร์ (Timer Register) ภายในไมโครโปรเซสเซอร์ 8032 ซึ่งมีอยู่ 2 ตัวคือ ไทม์เมอร์/เคาน์เตอร์ 0 และ ไทม์เมอร์/เคาน์เตอร์ 1 (ในที่นี้ใช้เฉพาะไทม์เมอร์รีจิสเตอร์ 0) โดยควบคุมให้ไทม์เมอร์รีจิสเตอร์ทำงาน เมื่อเกิดการ อินเทอร์รัพท์ที่ขา INT0 เป็นครั้งแรก และ ไทม์เมอร์หยุดการทำงาน เมื่อเกิดการอินเทอร์รัพท์ที่ขา INT0 ครั้งใหม่ จากนั้นจึงเก็บข้อมูลที่อยู่ในรีจิสเตอร์ไทม์เมอร์ไปไว้ที่หน่วยความจำภายนอก เมื่อเสร็จกระบวนการเก็บข้อมูลจึงให้ไทม์เมอร์รีจิสเตอร์ทำงาน ต่อไปจะมีการทำงานเช่นนี้ทุกครั้งที่มีการเปลี่ยนแปลงสถานะของสัญญาณอินพุท (ช่วงทริกที่ขอบขา ทั้งขาขึ้นและขาลง เพื่อที่จะเก็บข้อมูลได้ทั้งแถบดำและแถบขาว)

การทำให้เกิดสัญญาณอินเทอร์รัพท์ทุกครั้งที่สัญญาณอินพุทเปลี่ยนสถานะ สามารถอธิบายได้จากรูปข้างล่าง



รูปที่ 3.2 สัญญาณอินเทอร์รัพท์



รูปที่ 3.3 วงจรเอกซ์คลูซีฟออร์สัญญาณอินเทอร์รัพท์

ในตอนเริ่มแรกขณะที่หัวอ่านปากกาลอย สัญญาณจากหัวอ่านมีค่าต่ำหรือ "0" ในขณะที่หัวอ่านจดลงบนรหัสแถบ (กระดาษซึ่งเป็นสีขาว) สัญญาณจากหัวอ่านจะเปลี่ยนเป็นค่าสูง หรือ "1" เทคนิคการสร้าง

สัญญาณที่จะให้เกิดสัญญาณอินเทอร์รัพท์ทุกครั้งที่มีการเปลี่ยนสัญญาณจากหัวอ่านปากกา โดยการใช้เกทเอกซ์คลูซีฟออร์ ให้อินพุทหนึ่งมาจากหัวอ่าน อีกอินพุทหนึ่งมาจากการให้ค่าของไมโครโปรเซสเซอร์ซึ่งใช้พอร์ท1.0 เป็นตัวกำหนด โดยกำหนดค่าเริ่มต้นให้มีค่าสูง ขณะที่สัญญาณจากหัวอ่านปากกาเป็นต่ำ เมื่อผ่านเกทเอกคลูซีฟออร์แล้วได้ค่าเป็นสูง เมื่อเกิดการเปลี่ยนแปลงของสัญญาณจากหัวอ่านครั้งแรก หลังจากผ่านเกทเอกคลูซีฟออร์แล้วสัญญาณที่ได้มีค่าต่ำ จากนั้นหน่วงเวลาแล้วกลับค่าของพอร์ท1.0 ทำให้สัญญาณที่ได้จากเอกคลูซีฟออร์ กลับเป็นมีค่าสูงอีกครั้ง ดังนั้นลักษณะของสัญญาณจะเป็นดังรูปที่ 3.2

2. การวัดขนาดความกว้างของแถบ โดยต้องการรู้ว่าข้อมูลแถบใดมีค่าสูง แถบใดมีค่าต่ำ โดยปล่อยให้ไทม์เมอร์ทำงานจนกระทั่งเกิดการอินเทอร์รัพท์เข้ามาที่ขา INT0 แล้วเก็บค่าของรอบเวลาการทำงาน (machine cycle) ที่อยู่ในไทม์รีจิสเตอร์ไว้ในหน่วยความจำเพื่อนำมาใช้ต่อไป แต่ถ้าไทม์เมอร์รีจิสเตอร์เกิดการโอเวอร์โฟลว์ (overflow) ซึ่งหมายถึงเป็นรหัสแถบสุดท้ายแล้ว ให้หยุดกระบวนการเก็บข้อมูลแล้วเข้าทำงานในกระบวนการประมวลผลข้อมูลต่อไป

จากลักษณะการรับข้อมูลดังกล่าวก็ยังไม่สามารถเก็บข้อมูลที่ตรงกับความเป็นจริงได้ เนื่องจากความเร็วในการลากหัวอ่านปากกาของผู้ลาก ในขณะที่ลากไปในแต่ละช่วง(ช่วงหัว ช่วงกลาง และช่วงปลายของโมดูล) จะไม่เท่ากันทั้งหมด ดังนั้นเมื่อจัดเก็บข้อมูลเข้าไปในหน่วยความจำแล้ว จะมีการชดเชยขนาดข้อมูลโดยซอร์ฟแวร์ต่อไป

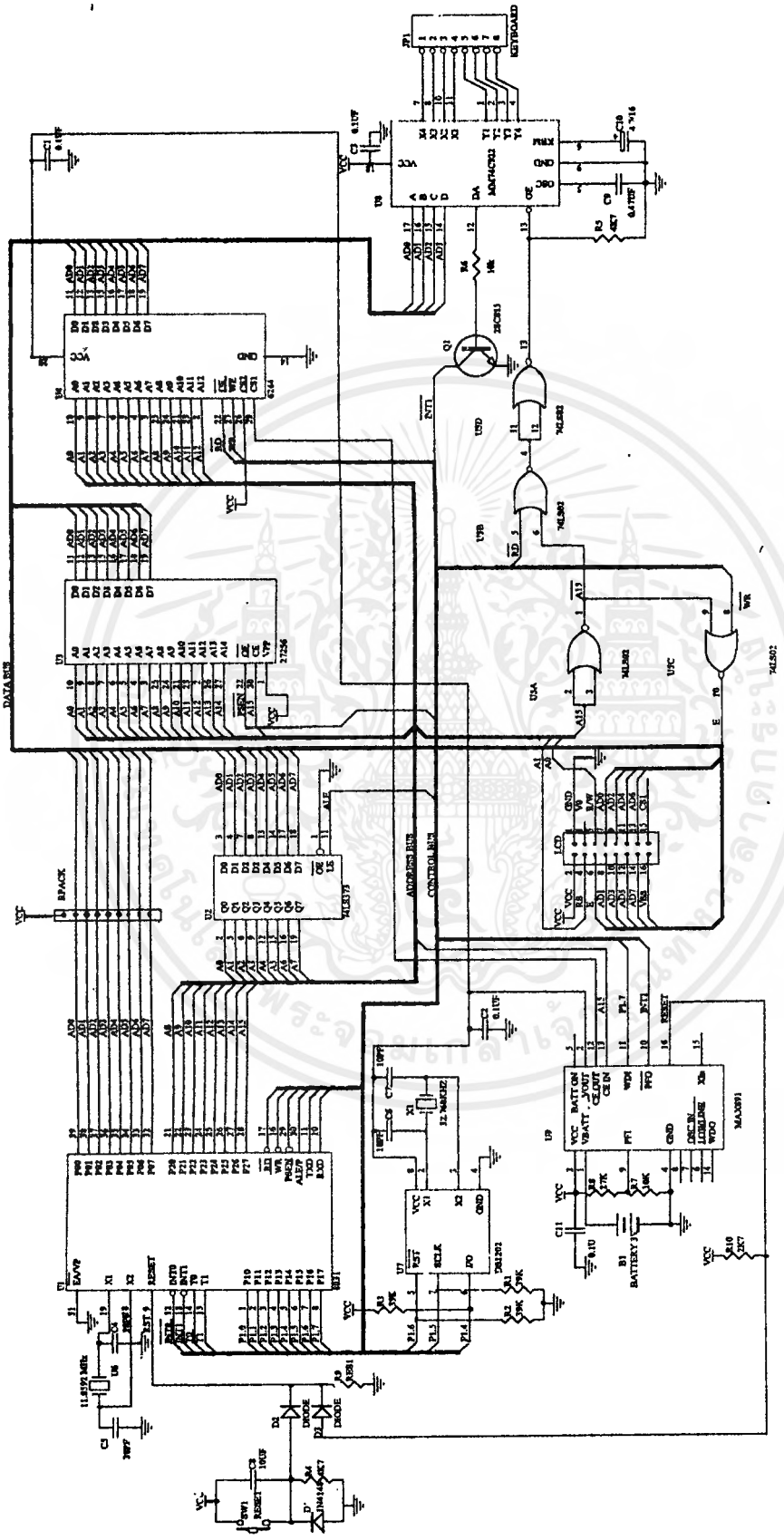
ส่วนประมวลผลข้อมูล ส่วนนี้ประกอบด้วย

1. หน่วยประมวลผลกลาง ใช้ไมโครโปรเซสเซอร์ 8032 ใช้ในการรับสัญญาณอินเทอร์รัพท์ประมวลผล จัดเก็บข้อมูลและส่งข้อมูลที่ได้จากการประมวลเป็นเอาท์พุทออกไป ซึ่งภายในหน่วยประมวลผลมี ไทม์เมอร์/เคาน์เตอร์ 2 ตัว และ พอร์ทขนานอีก 4 พอร์ท โดยใช้ พอร์ท 0 ส่ง 8 บิทล่างและส่ง-รับข้อมูล 8 บิท และ พอร์ท 2 ใช้ส่ง แอดเดรส 8 บิทบนของหน่วยความจำภายนอกที่ต้องการติดต่อ ในเครื่องนี้ต่อหน่วยความจำข้อมูลภายนอก เพิ่มเพื่อใช้เก็บข้อมูล

หลักการต่ออุปกรณ์เหล่านี้เข้ากับไมโครโปรเซสเซอร์ โดยดูการทำงาน ได้จากการอ่านและเขียนของหน่วยความจำโปรแกรม และ หน่วยความจำข้อมูล โดยในวงจรจะมีการใช้ ไอซี 74LS373 เป็นตัวแลทช์แอดเดรส 8 บิทล่างและเปลี่ยนสถานะในทุกขาของพอร์ท 0 ให้เป็น อิมพีแดนซ์สูง (high impedance) และพร้อมที่จะรับข้อมูลที่จะส่งออกมาจากอีพรอม

2. หน่วยความจำและส่วนดีโคดหน่วยความจำ

หน่วยความจำแยกเป็น หน่วยความจำโปรแกรมภายนอก และ หน่วยความจำข้อมูลภายใน การติดต่อระหว่างหน่วยประมวลผล กับ หน่วยความจำ ด้วยการดีโคดหน่วยความจำ ใช้ไอซี 74LS138 เป็นตัวดีโคดแอดเดรส โดยกำหนดให้แอดเดรส 0000H ถึง 1FFFH จะติดต่อกับหน่วยความจำโปรแกรม และแอดเดรส 2000H ถึง 2FFFH จะติดต่อกับหน่วยความจำข้อมูล เพื่อเก็บข้อมูลจากอินพุท



รูปที่ 3.4 วงจรไมโครโปรเซสเซอร์

3.2 การออกแบบซอร์ฟแวร์

การทำงานของโปรแกรมทั้งหมดสามารถแยกเป็นส่วนหลักๆได้ 2 ส่วนคือ

ส่วนที่ 1

เป็นส่วนที่รับสัญญาณเข้ามาจากหัวอ่าน แล้วนำมาเก็บไว้ใน หน่วยความจำข้อมูล (หน่วยความจำภายนอก) เพื่อที่จะนำข้อมูลมาทำการทำงานในขั้นตอนต่อไป

หลักการทำงานมีดังนี้คือ สัญญาณที่เข้ามามีลักษณะเป็นพัลส์เข้ามาเพื่อเป็นการอินเทอร์รัพท์ ที่ขา INT0 เมื่อเกิดการอินเทอร์รัพท์ จะทำให้ไทม์เมอร์รีจิสเตอร์ทำงาน และไทม์เมอร์รีจิสเตอร์จะหยุดทำงานเมื่อเกิดการอินเทอร์รัพท์ครั้งใหม่

การวัดขนาดความกว้างของสัญญาณที่อินพุทเข้ามาโดยใช้ไทม์เมอร์รีจิสเตอร์ภายในหน่วยประมวลผล เป็นตัวนับจำนวนรอบการทำงาน ถ้าพัลส์มีขนาดกว้างมาก ค่าของรอบการทำงาน ก็จะมากตามไปด้วยแต่ถ้าพัลส์มีขนาดไม่กว้างมาก จำนวนของรอบการทำงาน ก็จะมีจำนวนน้อยไปด้วยเมื่อนับสัญญาณที่เข้ามาครบ 1 พัลส์ ก็จะนำค่าการนับที่ได้จากไทม์เมอร์รีจิสเตอร์ไปเก็บไว้ในหน่วยความจำข้อมูลภายนอก ในตำแหน่งที่กำหนดไว้และทุกๆ การเปลี่ยนแปลงของสถานะก็จะมีการเก็บข้อมูลในตำแหน่งถัดไปของ หน่วยความจำภายนอกจนกว่าจะเกิดการโอเวอร์โฟลว์ของไทม์เมอร์รีจิสเตอร์จึงจะทำการหยุดเก็บข้อมูลและเข้าทำงานในส่วนที่ 2 ต่อไป

ส่วนที่ 2

ในส่วนนี้สามารถแยกการทำงานได้เป็น 3 ส่วนหลักๆคือ

1. ทำหน้าที่หาค่าเฉลี่ยของจำนวนค่าของรอบเวลาการทำงาน ที่เก็บไว้ในหน่วยความจำข้อมูล โดยนำค่าของรอบเวลาการทำงาน 9 รอบมารวมกัน แล้วหารผลลัพธ์ที่ได้จากการรวมมาเก็บ เพื่อเป็นค่าอ้างอิงในการแปลงหรัสแถบเป็นตัวเลขต่อไป

2. จะทำหน้าที่เปลี่ยนข้อมูลให้กลายเป็นลักษณะของรหัสแถบ โดยจะเริ่มจากการนำจำนวนรอบของเวลาการทำงาน ของแต่ละแถบรหัสมาเปรียบเทียบกับผลลัพธ์ที่ได้จากส่วนที่ 1 ถ้าหากค่าของรอบเวลาการทำงาน มีค่ามากกว่าค่าอ้างอิงก็จะกำหนดให้มีค่า เป็น "1" แต่ถ้าค่าของรอบเวลาการทำงานมีค่าน้อยกว่าค่าอ้างอิง ก็จะกำหนดให้มีค่าเป็น "0" นำค่ารหัสที่ได้ ไปเก็บไว้ในหน่วยความจำทีละอักขระ จากนั้นจึงจะเริ่มทำงานในส่วนถัดไป

3. เป็นส่วนที่ใช้ในการถอดรหัสข้อมูลซึ่งมีวิธีการถอดรหัสโดยการ เทียบค่าจากตาราง (look up table) โดยจะมีการสร้างตารางรหัสของรหัสแถบไว้ หลังจากที่ได้รหัส จากส่วนที่ 2 แล้ว ก็จะนำค่าของแต่ละอักขระไปเปรียบเทียบกับตาราง เมื่อค่าในตารางตรงกับรหัส ก็จะนำค่าในตารางไปแปลงเป็นรหัสของพนักงานแล้วจึงนำไปเก็บไว้ในหน่วยความจำเพื่อนำไปใช้งานต่อไป และ นำค่าที่ได้ออกแสดงผลออกทางจอแสดงผล (LCD)

3.3 การแสดงผลออกทางจอแสดงผล

การแสดงผลให้จอแสดงผล แบบ 16 อักขระ 2 บรรทัด โดยการติดต่อกับจอแสดงผล ใช้การติดต่อแบบเมโมรีแมพ (memory map) โดยผ่านบัสจอแสดงผล (LCD Bus) ขนาด 20 เส้น ทำให้การอ่านและเขียนข้อมูลกับจอแสดงผล เหมือนกับการเขียนข้อมูลลงหน่วยความจำภายนอก นอกจากนั้นยังสามารถตรวจสอบแฟลกความพร้อมของจอแสดงผลได้

ชุดคำสั่งและการแสดงข้อความ

การเขียนหรืออ่านข้อมูลกับจอแสดงผล ก็คือ การกำหนดคุณสมบัติต่างๆ ในการใช้งานของจอแสดงผลเพื่อให้ปรากฏข้อความบนจอแสดงผล โดยมีรูปแบบดังนี้

คำสั่งหนึ่งชุด ประกอบด้วย 3 ส่วน

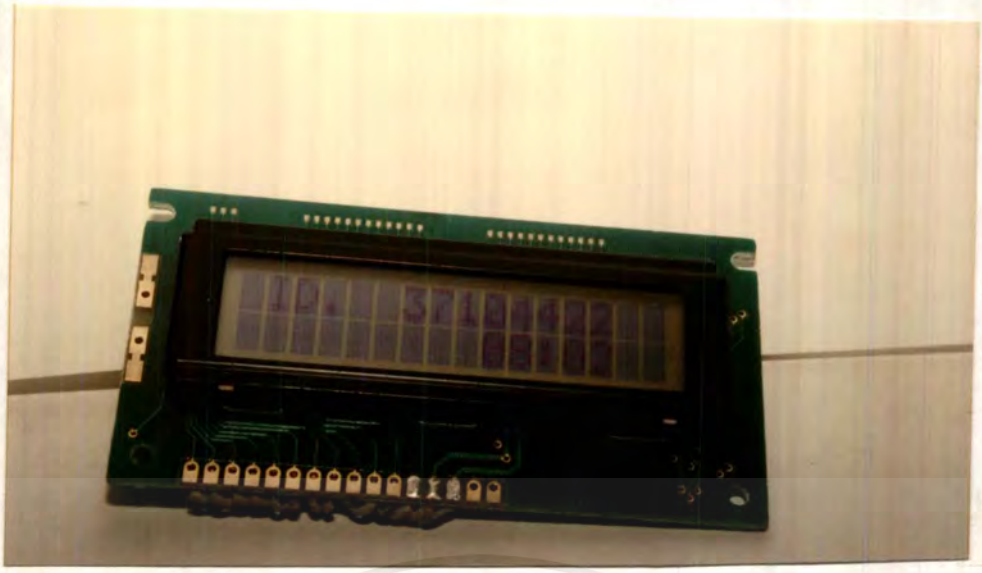
1. RS (Register Selection) ทำหน้าที่ในการกำหนดว่าข้อมูลที่กำลังจะส่งต่อไป เป็นชนิดชุดคำสั่ง หรือ ข้อมูล โดยถ้า RS = 0 จะเป็นชุดคำสั่ง และถ้า RS = 1 จะเป็นข้อมูลที่ต้องการจะแสดงผล

2. R/W (Read / Write) ทำหน้าที่ในการกำหนดว่าจะเป็นการอ่านข้อมูลจากจอแสดงผล หรือเขียนข้อมูลลงไป โดยถ้า R/W = 0 จะเป็นการเขียนลงไปยังจอแสดงผล และถ้า R/W = 1 จะเป็นการอ่านข้อมูลจากจอแสดงผล

3. Data Bit หรือ บิทข้อมูล คือ ส่วนที่เป็นข้อมูลที่เขียนลงไปในจอแสดงผลซึ่งจะเป็นเลขฐานสอง เป็นรหัสแอสกีแทนตัวอักษรต่างๆ

ชุดคำสั่งควบคุม

	RS	R/W	Data Bit (binary)
1. Clear Display	0	0	0 0 0 0 0 0 0 1
2. Currsor at Home	0	0	0 0 0 0 0 0 1 x
3. Entry Mode Set	0	0	0 0 0 0 0 1 I/D S
4.Display On/Off	0	0	0 0 0 0 1 D C B
5. Function Set	0	0	0 0 1 DL N F x x
6. Display Shift	0	0	0 0 0 1 S/C R/L x x
7. Set CGRAM Add.	0	0	0 1 CGARAM Add.
8. Set DDRAM Add.	0	0	1 DDRAM Add.
9. Busy Add. Read	0	1	BF -----ADD-----
10. GRAM,DDRAM WR	1	0	1 0 --Write Data--
11. GRAM,DDRRAM RD	1	1	1 1 --Read Data--



รูปที่ 3.5 จอแสดงผล

รายละเอียดของแต่ละคำสั่ง

1. เคลียร์จอแสดงผล (Clear Display)

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

สำหรับการเคลียร์จอแสดงผลโดยจะทำการเขียนตัวเว้นวรรคลงใน DDRAM ทั้งหมด และกำหนดค่าแอดเดรส DDRAM ให้เป็น 0 พร้อมทั้ง จะกลับไปที่ตำแหน่งซ้ายบนสุดของจอภาพ

2. เลื่อนเคอร์เซอร์ไปที่ตำแหน่งโฮม (Cursor at Home)

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

สำหรับกำหนดค่าแอดเดรส DDRAM ให้เป็น 0 พร้อมทั้ง เคอร์เซอร์จะไปอยู่ที่ตำแหน่งซ้ายสุดบนสุดของจอภาพ โดยที่ข้อมูลใน DDRAM ไม่มีการเปลี่ยนแปลง

3. กำหนดโหมดการเลื่อนเคอร์เซอร์ (Entry Mode Set)

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

I/D = 0 กำหนดทิศทางของเคอร์เซอร์และ DDRAM ให้เป็นแบบลดลง

I/D = 1 กำหนดทิศทางของ เคอร์เซอร์ และ DDRAM ให้เป็นแบบ เพิ่มขึ้น

S = 0 เมื่อเขียนข้อมูลแล้ว ตัวเคอร์เซอรจะถูกเลื่อนไปตามทิศทางของ I/D

S = 1 เมื่อเขียนข้อมูลแล้ว ตัวเคอร์เซอรจะอยู่กับที่ แต่ตัวอักษรจะถูกผลักไปตามทิศทางของ I/D

4. แสดง/ไม่แสดงผล (Display On/Off)

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

D = 0 กำหนดให้ไม่แสดงผล

D = 1 กำหนดให้แสดงผล

C = 0 กำหนดให้ไม่แสดงเคอร์เซอร์

C = 1 กำหนดให้แสดงเคอร์เซอร์

B = 0 กำหนดให้ไม่มีการกระพริบที่ตำแหน่งเคอร์เซอร์

B = 1 กำหนดให้มีการกระพริบที่ตำแหน่งเคอร์เซอร์

5. การเลื่อนตำแหน่ง (Display Shift)

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

S/C = 0 กำหนดให้เลื่อนเคอร์เซอร์ตามทิศทาง R/L ไป 1 ตำแหน่ง

S/C = 0 กำหนดให้เลื่อนข้อความตามทิศทาง R/L ไป 1 ตำแหน่ง (เลื่อนทุกบรรทัด)

R/L = 0 กำหนดให้มีทิศทางไปทางซ้าย

R/L = 1 กำหนดให้มีทิศทางไปทางขวา

6. กำหนดฟังก์ชั่น (Function Set)

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

DL = 0 กำหนดให้การติดต่อกับจอแสดงผลเป็นแบบ 4 บิท

DL = 1 กำหนดให้การติดต่อกับจอแสดงผล เป็นแบบ 8 บิท

N = 1 กำหนดบรรทัดแบบ 1/8 บรรทัด

N = 1 กำหนดบรรทัดแบบ 1/16 บรรทัด

F = 0 กำหนดให้เป็นตัวอักษรแบบ 5 x 7 จุด

F = 1 กำหนดให้เป็นตัวอักษรแบบ 5 x 10 จุด

7. กำหนดแอดเดรส CGARAM (Set CGRAM Adress)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	----------CGRAM Address----------						

สำหรับการกำหนดแอดเดรสของ CGRAM

8. กำหนดแอดเดรส DDARAM (Set DDRAM Address)

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

สำหรับการกำหนดแอดเดรสของ DDRAM

9. ตรวจแฟลกบีซี (Busy Flag) และ อ่านแอดเดรส

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

สำหรับการอ่านค่า BF ซึ่งบอกความพร้อมของจอแสดงผล ในการรับข้อมูล BF = 0 หมายถึงพร้อมรับข้อมูล BF = 1 หมายถึงว่ายังไม่พร้อมรับข้อมูล

การเขียนโปรแกรมเพื่อควบคุมการทำงาน จะใช้การส่งโดย

1. คำสั่งที่ต้องการจะเขียนไปจอแสดงผล จะเขียนไปที่แอดเดรส FA00H
2. คำสั่งที่ต้องการจะอ่าน (BF) จากจอแสดงผล จะเขียนไปที่แอดเดรส FA01H
3. ข้อมูลที่ต้องการจะเขียนไปยังจอแสดงผล จะเขียนไปที่แอดเดรส FA02H
4. ข้อมูลที่ต้องการจะอ่านจากจอแสดงผล จะเขียนไปที่แอดเดรส FA03H

3.4 ส่วนนาฬิกาเวลาจริง (Real Time Clock)

ในส่วนนี้ใช้ไอซี DS1202 ซึ่งการทำงานเป็นอิสระกับไมโครโปรเซสเซอร์และแม้จะปิดเครื่องแล้ว นาฬิกาก็จะยังคงทำงานต่อไป ภายในไอซี DS1202 ประกอบด้วยนาฬิกาเวลาจริงและหน่วยความจำสถิต (Static RAM) ขนาด 24 x 8 ไบท์

ในการติดต่อกับไอซี DS1202 เป็นแบบอนุกรม โดยติดต่อผ่านพอร์ท1.4,1.5,1.6 โดยทำหน้าที่เป็นตัวส่ง-รับข้อมูล ขารีเซ็ท และเป็นนาฬิกาอนุกรม ตามลำดับ ในการติดต่อกับสัญญาณที่ขารีเซ็ทต้องเป็นลอจิก "1" และนาฬิกาอนุกรม 1 ลูกต่อการส่งข้อมูล 1 บิท

การติดต่อสามารถเขียนหรืออ่านข้อมูลของนาฬิกาและหน่วยความจำโดยจะต้องประกอบด้วยไบท์คำสั่ง (command byte) ขนาด 8 บิทและตามด้วยข้อมูลขนาด 8 บิท

ไบท์คำสั่งและแอดเดรส

1	RAM	A_4	A_3	A_2	A_1	A_0	RD/W

RAM/CK เลือกติดต่อกับหน่วยความจำหรือนาฬิกา เมื่อมีค่า "1" จะติดต่อหน่วยความจำ เมื่อมีค่า "0" จะติดต่อกับนาฬิกา

A_4-A_0 แอดเดรสของหน่วยความจำหรือนาฬิกา

RD/W เลือกที่จะอ่านหรือเขียนข้อมูล เมื่อมีค่า "1" จะเป็นการอ่าน เมื่อมีค่า "0" จะเป็นการเขียนแอดเดรสของรีจิสเตอร์

แอดเดรสของรีจิสเตอร์

1. นาฬิกา

SEC	1	0	0	0	0	0	0	RD/W
MIN	1	0	0	0	0	0	1	RD/W
HR	1	0	0	0	0	1	0	RD/W
DATE	1	0	0	0	0	1	1	RD/W

MONTH	1	0	0	0	1	0	0	RD/W
DAY	1	0	0	0	1	0	1	RD/W
YEAR	1	0	0	0	1	1	1	RD/W
CONTROL	1	0	0	0	1	1	1	RD/W

2. หน่วยความจำ

RAM0	1	1	0	0	0	0	0	RD/W
				:				
RAM23	1	1	1	0	1	1	1	RD/W

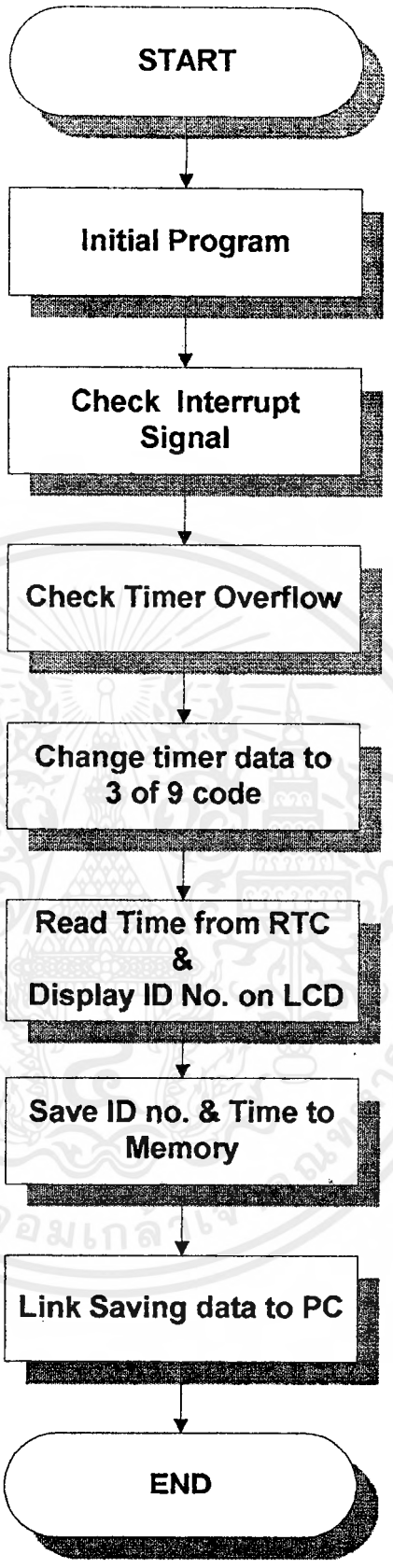
3.5 การส่งข้อมูลผ่านพอร์ตสื่อสารอนุกรมในไมโครโปรเซสเซอร์ 8032

โปรแกรมสำหรับส่งข้อมูลนี้จะทำการตรวจสอบบิต TI เพื่อรอให้ส่งข้อมูลทีละไบท์ โปรแกรมนี้จะใช้โครงสร้างการอินเทอร์รัพท์ของพอร์ตสื่อสารอนุกรม เมื่อส่งข้อมูลเสร็จเป็นตัวตรวจสอบคำสั่งที่ใช้ตรวจสอบว่าข้อมูลส่งเสร็จแล้วหรือไม่ คือคำสั่ง

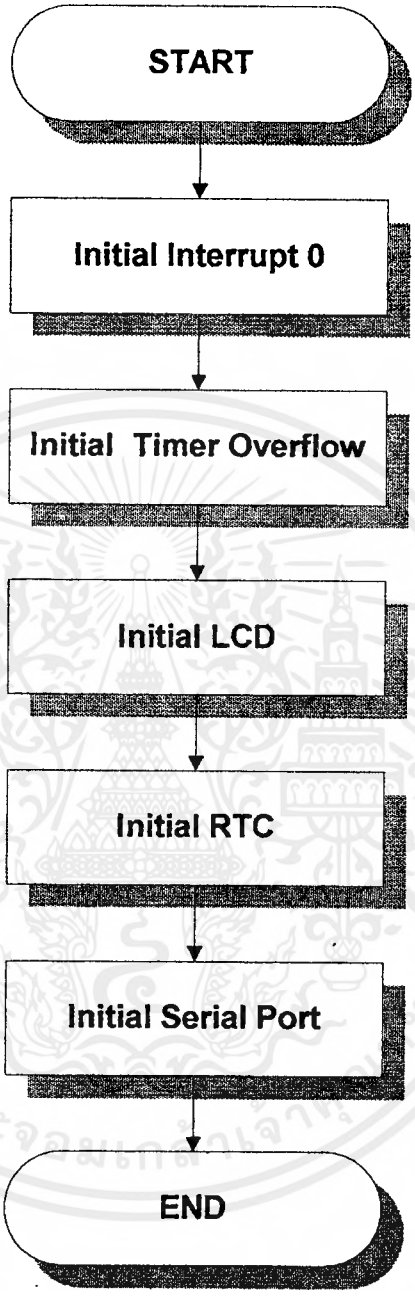
```
JNB     TI,$        ; รอจนกระทั่งข้อมูลถูกส่งเสร็จ
```

3.6 การรับข้อมูลจากพอร์ตอนุกรมเพื่อบันทึกเป็นเท็กซ์ไฟล์ (text file)

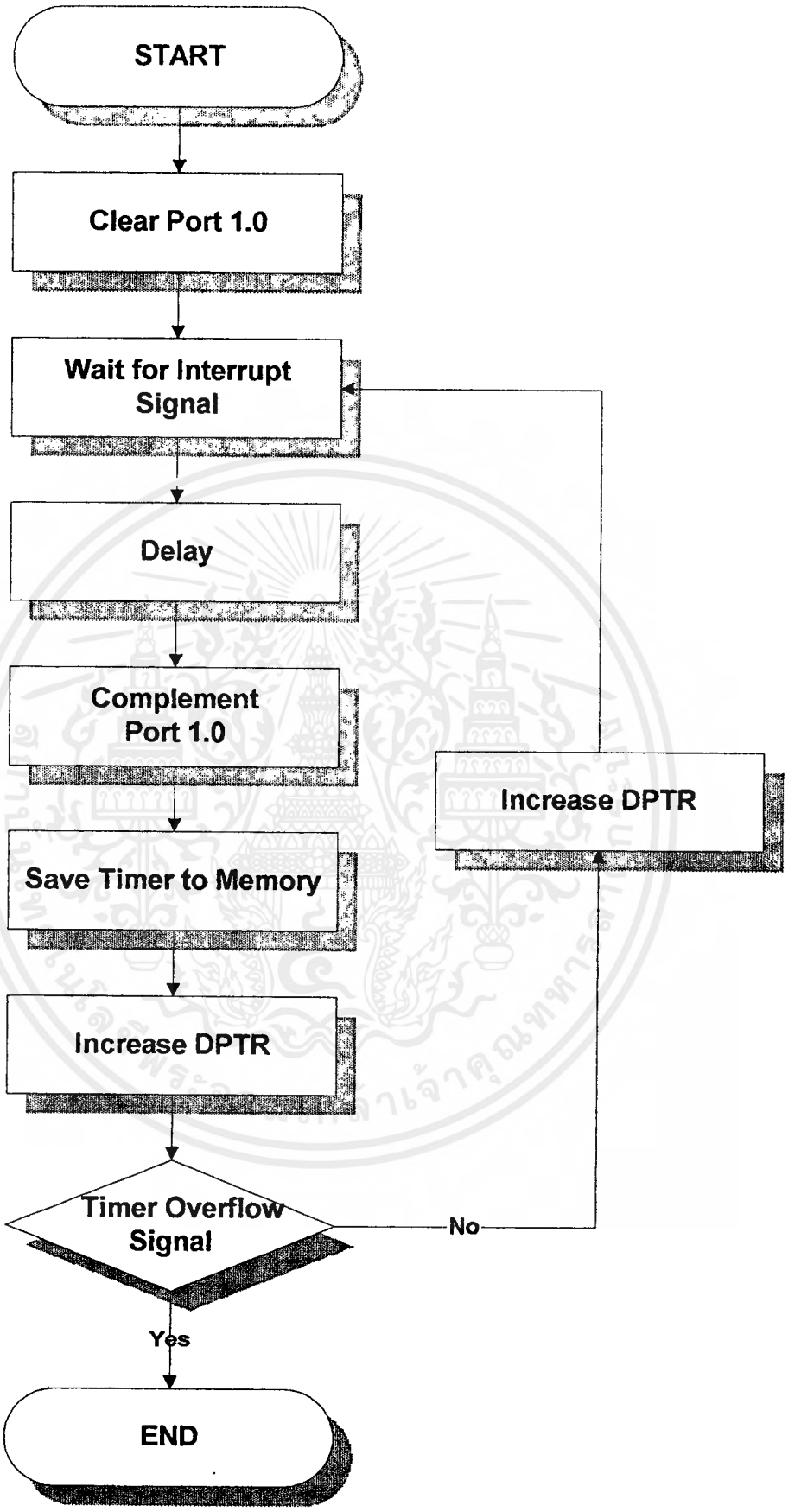
โปรแกรมส่วนนี้เป็นการรอรับข้อมูลจากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์เพื่อทำการเก็บเป็นไฟล์ ซึ่งเขียนโดยใช้ภาษาวิชวลเบสิก (Visual Basic) โดยขั้นแรกทำการเลือกพอร์ตแล้วทำการเซ็ตบอดเรท (Baud Rate) จากนั้นรับข้อมูลจากโปรแกรมในหัวข้อที่แล้วเป็นอินพุท สุดท้ายทำการเขียนข้อมูลเหล่านั้นเป็นเท็กซ์ไฟล์



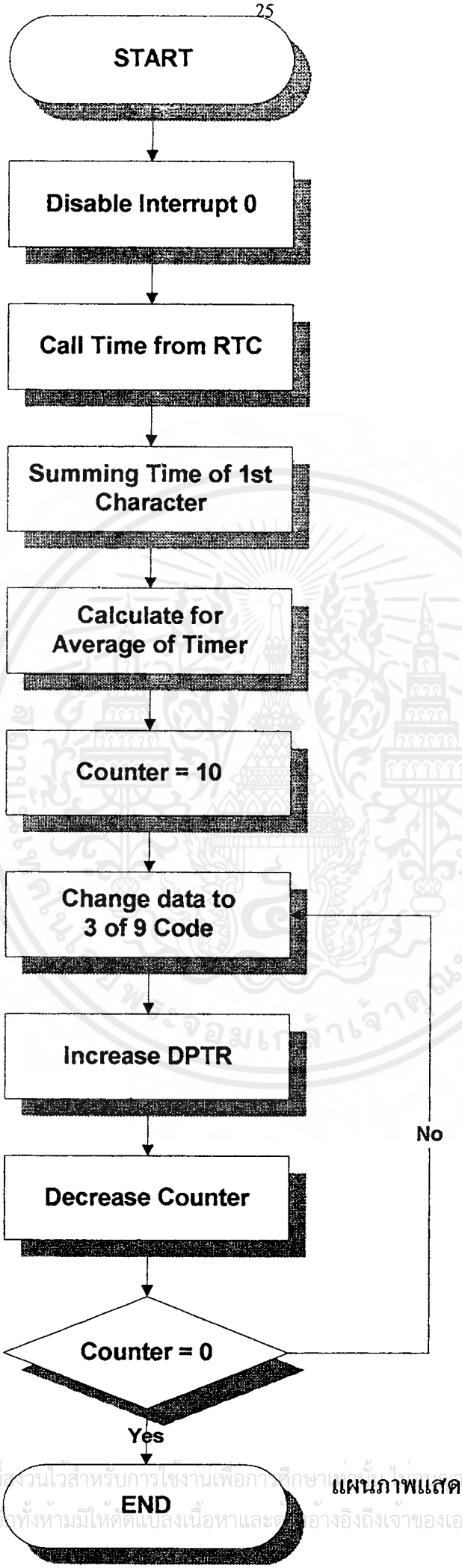
รูปที่ 3.6 แผนภูมิรูปภาพแสดงขั้นตอนการทำงานของเครื่องบันทึกเวลาพนักงาน



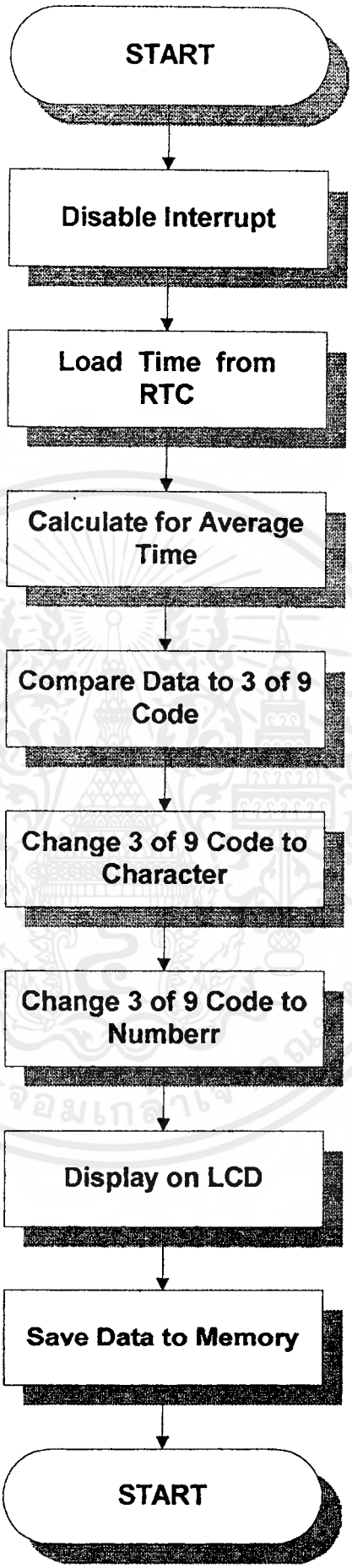
รูปที่ 3.7 แผนภูมิรูปภาพแสดงการกำหนดค่าเริ่มต้น



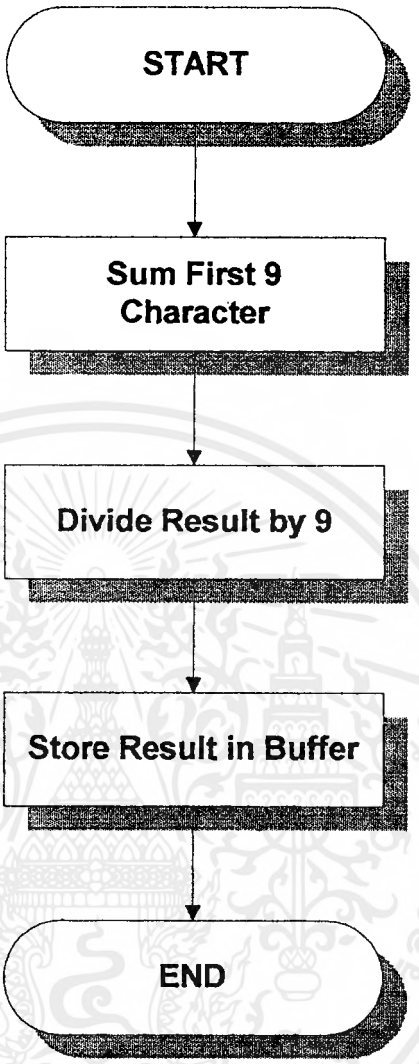
รูปที่ 3.8 แผนภาพแสดงการเก็บค่าเวลาของแถบและการตรวจสอบโอเวอร์โฟลว์



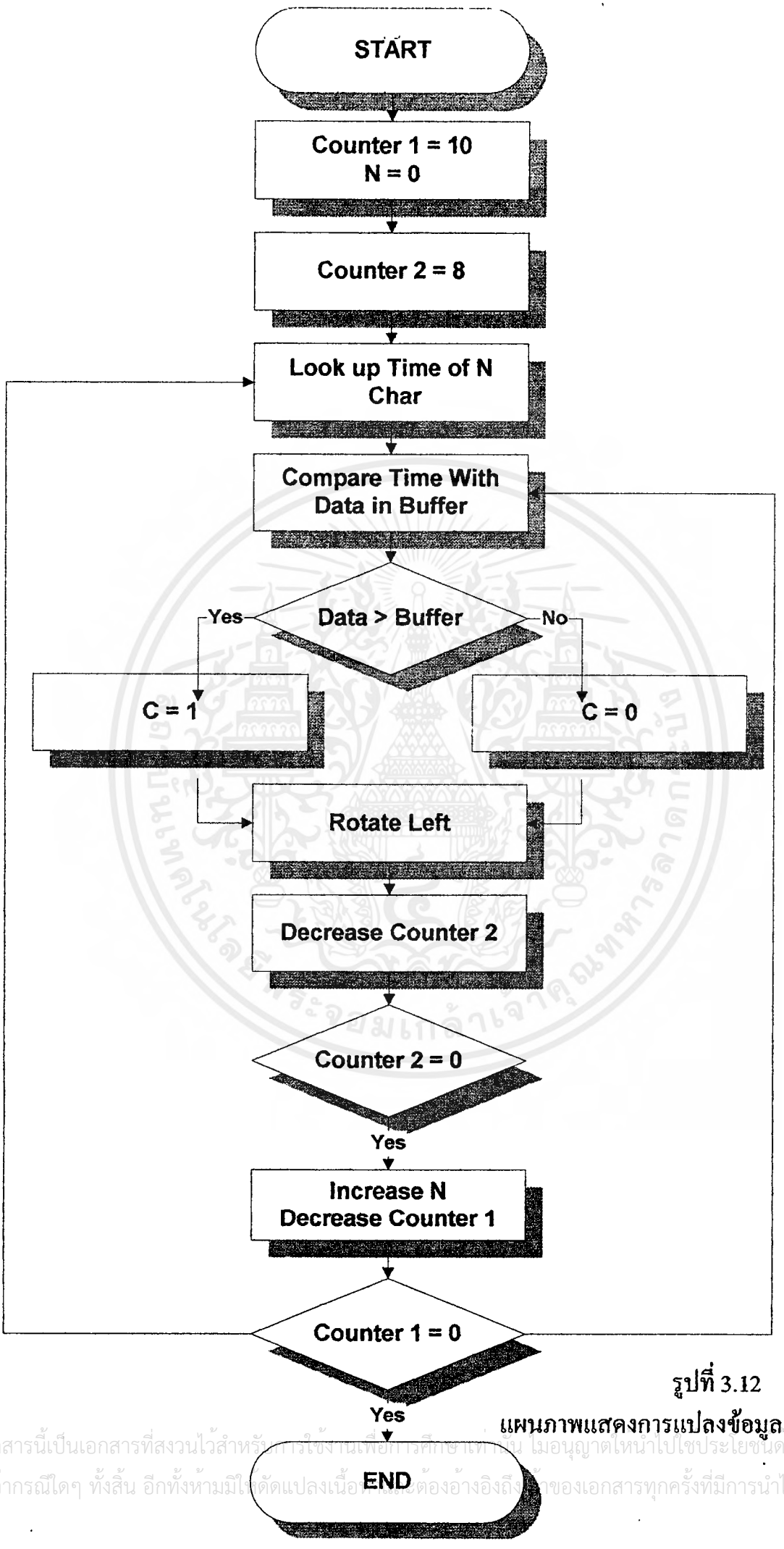
รูปที่ 3.9
แผนภาพแสดงการแปลงค่าเวลาเป็นรหัส 3ใน9



รูปที่ 3.10 แผนภาพแสดงการแสดงผล และ การเก็บข้อมูล

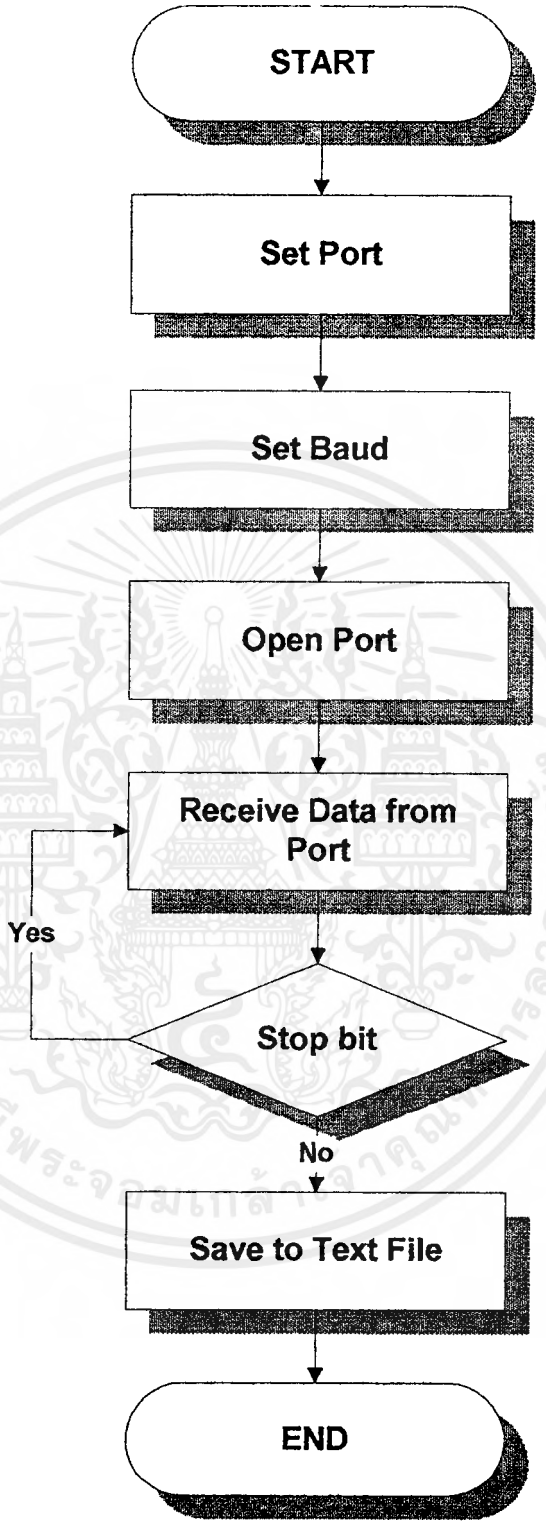


รูปที่ 3.11 แผนภาพแสดงการชดเชยเวลา



รูปที่ 3.12

แผนภาพแสดงการแปลงข้อมูลเป็นเลขฐานสอ



รูปที่ 3.13 แผนภาพแสดงการส่งข้อมูลให้ PC

บทที่ 4

ผลการทดลอง

4.1 อุปกรณ์ที่ใช้ในการทดลองและบันทึกผล

1. สตอเรจสโคป
2. เครื่องคอมพิวเตอร์ส่วนบุคคล
3. หัวอ่านรหัสแถบ
4. แหล่งจ่ายแรงดันไฟ
5. บัตรนักศึกษาซึ่งใช้รหัสแถบชนิด 3 ใน 9
6. บอร์ดวงจรที่ใช้ในการทดลอง

4.2 วิธีการทดลอง

การทดลองจะทำการแบ่งเป็น 2 ส่วน คือ

ส่วนที่ 1 จะทำการวัดสัญญาณที่ออกจากหัวอ่านขณะที่ทำการลากผ่านรหัสแถบเทียบกับสัญญาณที่เป็นเอาท์พุทของ เอกซ์คลูซีฟออร์ หรือสัญญาณที่เข้าไปที่ไมโครคอนโทรลเลอร์ 8032 ซึ่งทำการทดลองดังนี้

1. นำโพรบจับที่ 2 จุดคือ ขาสัญญาณออกจากหัวอ่าน และ ขาที่เป็นสัญญาณออกของ เอกซ์คลูซีฟออร์
2. ทำการลากหัวอ่านผ่านรหัสแถบ
3. เปรียบเทียบสัญญาณทั้งสองพร้อมกัน

รูปที่ 4.1 หัวอ่านรหัสแถบ

ส่วนที่ 2 ทำการดูค่าที่ได้หลังจากการแปลงรหัสแถบจาก LCD ซึ่งทำการทดลองดังนี้

1. ทำการลากหัวอ่านผ่านรหัสแถบโดยลากด้วยความเร็ว 3 ระดับคือ ความเร็วสูงสุดคือทำการลากให้เสร็จภายในเวลา 0.5 วินาที ความเร็วช้าคือใช้เวลาประมาณ 1 วินาที และความเร็วช้ามาก คือใช้เวลาประมาณ 1.5 วินาที ซึ่งในการลากต้องพยายามลากด้วยความเร็วคงที่และต้องไม่ลากตกขอบของรหัสแถบ เปรียบเทียบผลที่ได้โดยดูจาก LCD ดูเวลาจาก LCD ด้วย

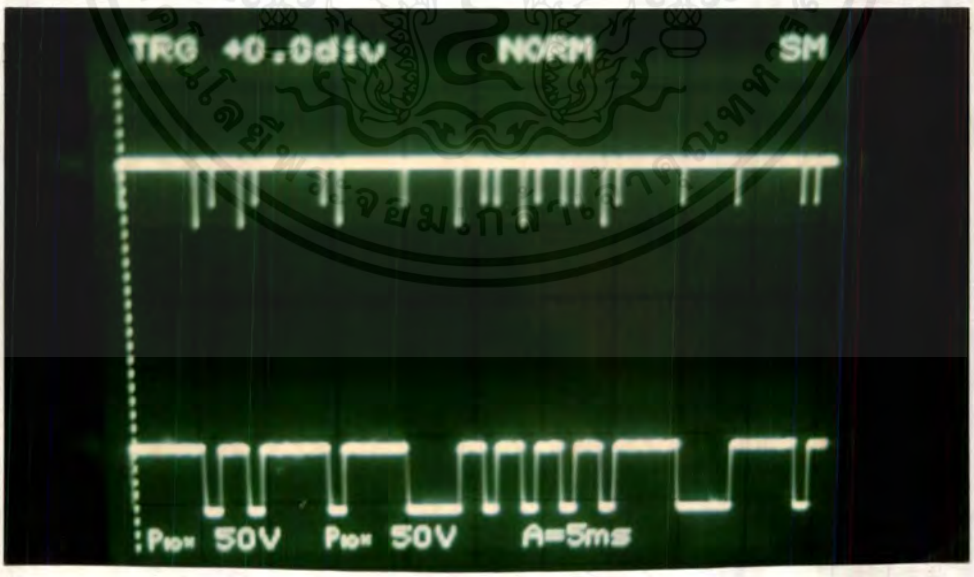
2. ทดลองลากผ่านรหัสแถบอื่นๆอีก 5 รหัสแถบโดยใช้ความเร็ว 3 ระดับเหมือนเดิมบันทึกผลจาก LCD

ส่วนที่ 3 ทำการส่งข้อมูลไปยังคอมพิวเตอร์เพื่อเก็บเป็นเท็กซ์ไฟล์ ดูผลที่ได้

4.3 ผลการทดลอง

ส่วนที่ 1

ผลการทดลองจะได้สัญญาณ 2 สัญญาณด้วยกันคือ สัญญาณที่ได้จากหัวอ่านรหัสแถบแวนและสัญญาณที่เป็นเอาท์พุทของ เอกซ์คลูซีฟออร์ หรือสัญญาณที่เข้าไปที่ไมโครคอนโทรลเลอร์ 8032 ลำดับแรกเมื่อทำการต่อหัวอ่านเรียบร้อยแล้ว ป้อนไฟเลี้ยง +5 โวลท์ เข้าไปแล้วนำหัวอ่านลากผ่านรหัสแถบซึ่งก็คือบัตรนักศึกษา จากนั้นดูสัญญาณจาก สตรอเรจสโคป คือสัญญาณจากหัวอ่านโดยตรงและสัญญาณที่เป็นเอาท์พุทของ เอกซ์คลูซีฟออร์ ซึ่งจะได้ดังรูปที่ 4.1 ซึ่งรูปบนคือสัญญาณจากหัวอ่าน ส่วนรูปล่างคือสัญญาเอาท์พุทของ เอกซ์คลูซีฟออร์



รูปที่ 4.2 สัญญาณจากหัวอ่านและสัญญาณเอาท์พุทของ เอกซ์คลูซีฟออร์ ซึ่งวัดโดยสตรอเรจสโคป

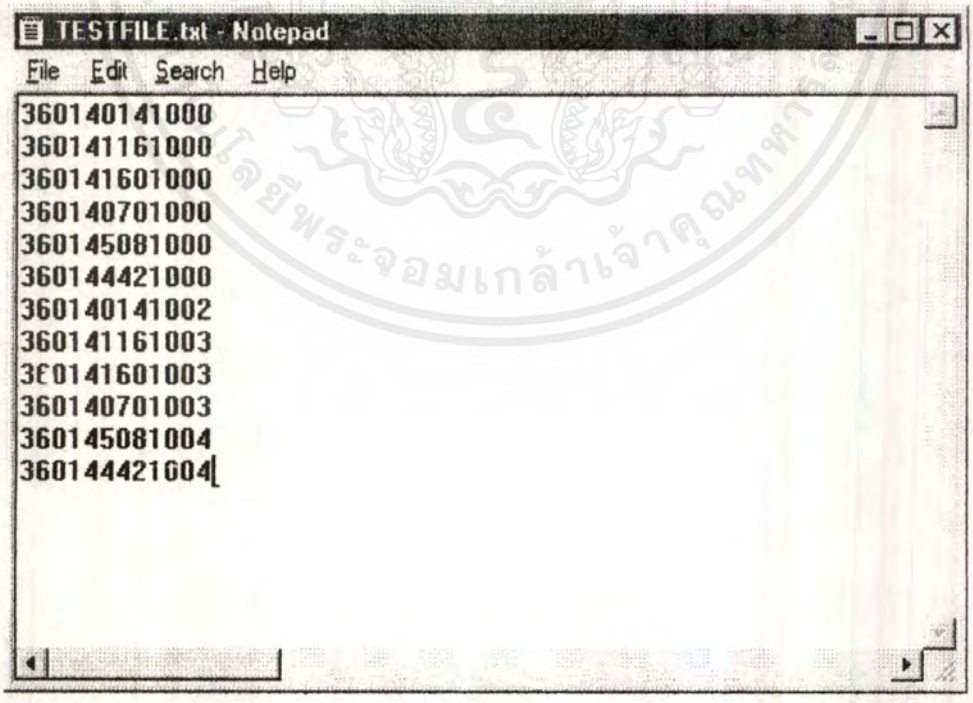
ส่วนที่ 2

หลังจากที่ได้ทำการลากผ่านรหัสแถบด้วยความเร็ว 3 ระดับด้วยความเร็วคงที่แล้วจะได้ผลดังตาราง ซึ่งจะเห็นว่าถ้าทำการลากความเร็วสูงสุดจะเกิดการผิดพลาด ก่อนจะไม่สามารถอ่านได้ ต้องลดความเร็วให้น้อยลง นอกนั้นค่าที่ได้จะถูกต้อง

รหัสที่ถูกต้อง	ความเร็วช้ามาก และค่าเวลา	ความเร็วช้า และค่าเวลา	ความเร็วสูงสุด และค่าเวลา
36014014	36014014 10:00:00	36014014 10:02:55	Error
36014116	36014116 10:00:05	36014116 10:03:12	Error
36014160	36014160 10:00:11	36014160 10:03:24	Error
36014070	36014070 10:00:23	36014070 10:03:44	Error
36014508	36014508 10:00:35	36014508 10:04:15	Error
36014442	36014442 10:00:45	36014442 10:04:59	Error

ตารางที่ 4.1 ผลการแสดงค่าออกทางหน้าจอเมื่อลากหัวอ่านด้วยค่าความเร็วต่างกัน

ส่วนที่ 3 เมื่อทำการส่งข้อมูลไปยังคอมพิวเตอร์แล้วจะเก็บเป็นเท็กซ์ไฟล์ดังรูป



รูปที่ 4.3 เท็กซ์ไฟล์ที่ได้จากการส่งข้อมูลไปยังคอมพิวเตอร์

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

สรุปและวิจารณ์ผลการทดลองในส่วนที่ 1

จะเห็นได้ว่าเมื่อเราทำการลากหัวอ่านผ่านรหัสแถบแล้วความกว้างของสัญญาณในแต่ละช่วงจะต่างกันไปขึ้นอยู่กับความกว้างของแถบต่าง ๆ ของรหัสแถบและความเร็วในการลากหัวอ่านด้วย ส่วนสัญญาณเอาท์พุทของ เอกซ์คลูซีฟออร์ ซึ่งเป็นสัญญาณที่เข้าไปในไมโครคอนโทรลเลอร์นั้น ค่าจะตกเป็น 0 ก็ต่อเมื่อสัญญาณจากหัวอ่านมีการอินเทอร์รัพท์เสมอ ซึ่งก็เป็นไปตามทฤษฎี

สรุปและวิจารณ์ผลการทดลองในส่วนที่ 2

จะเห็นได้ว่าถ้าทำการลากด้วยความเร็วที่ช้าหรือช้ามากจะทำให้ได้ค่าออกมาที่ถูกต้อง แต่ถ้าเกิดทำการลากที่ความเร็วที่มากเกินไปจะทำให้ไม่สามารถอ่านได้เพราะในการลากความเร็วมากๆ จะเกิดความเร่งทำให้ความเร็วในการลากไม่สม่ำเสมอ ดังนั้นจึงแสดงให้เห็นว่าควรลากด้วยความเร็วที่ช้าเพราะความเร็วจะคงที่

สรุปและวิจารณ์ผลการทดลองในส่วนที่ 3

จะเห็นว่าค่าที่แสดงในเท็กซ์ไฟล์มีค่าที่ถูกต้องตรงกับที่แสดงในจอแสดงผล ซึ่งเราสามารถนำค่าเหล่านี้ไปประยุกต์ใช้เก็บเป็นฐานข้อมูลในโปรแกรมอื่น ๆ เช่น Excel , Access , Lotus ฯลฯ

เอกสารอ้างอิง

1. สุเจตน์ จันทรังษ์, "ไมโครคอนโทรลเลอร์ชิพเดี่ยว 8051" ,โครงการตำราวิชาการ วิทยาลัยมหานคร, วิทยาลัยมหานคร, พ.ศ. 2535
2. ประดิษฐ์ ประนยานันทน์, ปิยพงศ์ เผ่าวณิช , "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , พ.ศ. 2536
3. สุนทร วิทูสุรพจน์, "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051" , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , พ.ศ. 2537

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จสมบูรณ์เป็นรูปเล่มขึ้นมานั้นก็ด้วยความร่วมมือและช่วยเหลือจากอาจารย์ภาคอิเล็คทรอนิกส์ ทางผู้จัดทำขอขอบคุณทุกท่านโดยเฉพาะอย่างยิ่งขอขอบคุณ ทาง ผศ. พลผดุง ผดุงกุล ผู้ให้คำปรึกษาทั้งแนวทางทฤษฎีและปฏิบัติ ตลอดจนให้ความอนุเคราะห์เครื่องมืออุปกรณ์ต่าง ๆ สุดท้ายนี้ขอขอบคุณทางภาคอิเล็คทรอนิกส์ที่ให้การสนับสนุนค่าใช้จ่ายในครั้งนี้

ภาคผนวก ก.

โปรแกรม

โปรแกรมการทำงานของ 8032

```
; ****** VARIABLE ******
LCDWRC      EQU 08000H   ;lcd write
control
LCDWRD EQU 08002H        ;lcd write
data

BUT0    EQU 30H
BUT1    EQU 31H
BUT2    EQU 32H
BUT3    EQU 33H
BUT4    EQU 34H
BUT5    EQU 35H
BUT6    EQU 36H
BUT7    EQU 37H
BUT8    EQU 38H
BUT9    EQU 39H
BUTI    EQU 3AH
BUTE    EQU 3BH
BUTC    EQU 3EH
BUTU    EQU 3FH

IDU     EQU 00H
CL      EQU 01H
UPU     EQU 02H

HOURH       EQU 08H
HOURL       EQU 09H
MINH    EQU 0AH
MINL    EQU 0BH
; ****** RESET ******
        ORG     0000H
        LJMP    MAIN
        ORG     0003H
        LJMP        S_INT0
        ORG     000BH
        LJMP        S_TM0
        ORG     0013H
        LJMP        INT_1
        ORG     001BH
        RETI
        ORG     0023H
        RETI

; ****** MAIN PROGRAM ******
        ORG     0030H
MAIN:
        MOV         R7,#0           ;
power up delay
M:
        NOP
        NOP
        DJNZ        R7,M

        SETB        IT0             ;
trigged int0 by
falling edge
```

```
        SETB    IT1             ;
trigged int0 by
                falling edge
        MOV     TMOD,#01H
                ;set timer0 model
        MOV     A,#00111000B
                ;lcd function set
        LCALL   LCDWI
        MOV     A,#00001100B
                ;display on/off
        LCALL   LCDWI
        MOV     A,#00000110B
                ;entry modc set
        LCALL   LCDWI
        MOV     A,#00010100B
                ;display shift
        LCALL   LCDWI
        MOV     SP,#40H
        CLR     P1.6    ;RST=0
        SETB    P1.5            ;
SCLK=1
        LCALL   DELAY4
        MOV     DPTR,#0100H
                ;initial add. for
                saving data
        MOV     A,#02H
                ;add. begin at 0200
h
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#00H
        MOVX @DPTR,A
        LCALL   INITP
LOOP:
        JB      IDU,ID_SUB
        JB      CL,CL_SUB1
        JB      UPU,UP_SUB1
        SJMP    LOOP
CL_SUB1:
        LJMP    CL_SUB
UP_SUB1:
        LJMP    UP_SUB
; ****** GET ID. SUBROUTINE ******
ID_SUB:
        MOV     A,#01H
        LCALL   LCDWI
        MOV     A,#82H
        LCALL   LCDWI
        MOV     A,#'I'
        LCALL   LCDWD
        MOV  A,#'D'
        LCALL   LCDWD
        MOV     A,#'.'
        LCALL   LCDWD
        MOV     A,#85H
        LCALL   LCDWI
ID_LOOP:
        LCALL   WAIT_KEY    ;1
        MOV     A,R0
        LCALL   LCDWD
        MOV     DPTR,#00D4H
        MOVX    @DPTR,A
```

```
LCALL   WAIT_KEY
;2
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00D5H
MOVX    @DPTR,A

LCALL   WAIT_KEY
;3
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00D6H
MOVX    @DPTR,A

LCALL   WAIT_KEY
;4
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00D7H
MOVX    @DPTR,A

LCALL   WAIT_KEY
;5
MOV     A,R0
LCALL   LCDWD
MOV  DPTR,#00D8H
MOVX    @DPTR,A

LCALL   WAIT_KEY
;6
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00D9H
MOVX    @DPTR,A

LCALL   WAIT_KEY
;7
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00DAH
MOVX    @DPTR,A

LCALL   WAIT_KEY
;8
MOV     A,R0
LCALL   LCDWD
MOV     DPTR,#00DBH
MOVX    @DPTR,A
WAIT_ENTER:
MOV     A,#0C0H
LCALL   LCDWI
LCALL   TDISP
LCALL   ALTER
LCALL   SAVE

LCALL   WAIT_KEY
CLR  EX1
CJNE  R0,#BUTE  ,
wait_enter
CLR  IDU
LCALL   INITP
JMP  LOOP
; ****** SET CLOCK ******
CL_SUB:
```

```
MOV     A,#01
LCALL   LCDWI
MOV     A,#80H
MOV     DPTR,#TIME
LCALL   LCDLDP
MOV     A,#0C2H
LCALL   LCDWI
LCALL   WAIT_KEY
MOV     A,R0
LCALL   LCDWD
ANL   A,#0FH
SWAP    A
MOV     HOURH,A
LCALL   WAIT_KEY
MOV     A,R0
LCALL   LCDWD
ANL   A,#0FH
MOV     HOURL,A
MOV     A,#':'
LCALL   LCDWD
LCALL   WAIT_KEY
MOV     A,R0
LCALL   LCDWD
ANL   A,#0FH
SWAP    A
MOV     MINH,A
LCALL   WAIT_KEY
MOV     A,R0
LCALL   LCDWD
ANL   A,#0FH
MOV     MINL,A
MOV     A,HOURH
MOV     R0,A
MOV     A,HOURL
ORL   A,R0
MOV     DPTR,#0110H
MOVX    @DPTR,A
MOV     A,MINH
MOV     R0,A
MOV     A,MINL
ORL   A,R0
MOV     DPTR,#0112H
MOVX    @DPTR,A
LCALL   SETTIME
WAIT_ENTER1:
        LCALL   WAIT_KEY
        CLR   EX1
        CJNE  R0,#
BUTE,WAIT_ENTER1
        CLR   CL
        LCALL   INITP
        JMP   LOOP
; ****** UP LINK TO PC ******
UP_SUB:
        MOV     A,#01H
        LCALL   LCDWI
        MOV     A,#80H
        MOV     DPTR,#LINK1
        LCALL   LCDLDP
        MOV     A,#0C0H
        MOV     DPTR,#LINK2
        LCALL   LCDLDP
        MOV     R2,#0
        LCALL   DELAY
```

```
        PUSH    SCON
        PUSH    TMOD
        PUSH    TH1
        PUSH    TCON
        PUSH    SBUF
        MOV     SCON,#40H
        ;SCON.6
        MOV     TMOD,#20H
        ;TMOD.5
        MOV     TH1,#0FDH
        SETB    TR1             ;
TCON.6
        MOV     DPTR,#0200H
UP_LOOP:
        MOVX    A,@DPTR
        CJNE    A,#0FFH,GO
        MOV     A,#0FFH
        ;stop bit
        ACALL   TRANS
        SJMP    UP_OUT
GO:
        ACALL   TRANS
        INC     DPTR
        SJMP    UP_LOOP
TRANS:
        CLR     TI              ;SCON.1
        MOV     SBUF,A
        ;SBUF
        JNB     TI,$            ;
SCON.1
        RET
UP_OUT:
        POP     SBUF
        POP     TCON
        POP     TH1
        POP     TMOD
        POP     SCON
        MOV     A,#0C0H
        MOV     DPTR,#LINK3
        LCALL   LCDLDP
        MOV     R2,#0
        LCALL   DELAY
        MOV     DPTR,#0100H
        MOV     A,#02H
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#0H
        MOVX    @DPTR,A
        CLR     UPU
        LCALL   INITP
        JMP     LOOP
; ****** INITIAL ROUTINE ******
INITP:
        CLR     P1.0
        MOV     TH0,#0
        MOV     TL0,#0
        SETB    EA              ;
enable all
        SETB    EX1             ;
enable int1
        SETB    EX0             ;
enable int0
```

```
        SETB   ET0              ;enable
timer
                        overflow
        MOV    R7,#0            ;R7 =
COUNTER OF BAR NO.
        MOV    A,#01H           ;
CLEAR LCD
        LCALL  LCDWI
        MOV    A,#80H
        MOV    DPTR,#READY      ;
DISPLAY 'READY'
        LCALL  LCDLDP
        MOV    R1,#01H          ;SKIP
FIRST LOOP
        CLR    IDU
        CLR    CL
        CLR    UPU
        RET

; ****** INT0 ROUTINE ******
S_INT0:
        CLR    TR0              ;STOP
TIMER0
        CLR    EX0              ;
DISABLE INT0
        CLR    EA               ;DISABLE
ALL INTERRUPT
        MOV    R0,#1AH          ;
DELAY
        DJNZ   R0,$
        CPL    P1.0             ;TO XOR
        DJNZ   R1,SECOND        ;
SKIP FIRST LOOP
        MOV    TL0,#0
        MOV    TH0,#0
        MOV    DPTR,#0000H
        SETB   EA
        SETB   EX0
        RETI
SECOND:
        MOV    A,TL0            ;SAVE
DATA TO MEM BEGIN AT 0002H
        MOVX   @DPTR,A
;0002H = LOW BYTE 0003H = HIGH
BYTE
        INC    DPTR
        MOV    A,TH0
        MOVX   @DPTR,A
        INC    DPTR
        MOV    TL0,#0
        MOV    TH0,#0
        SETB   EA               ;ENABLE
ALL INTERRUPT
        SETB   EX0              ;
ENABLE INT0
        INC    R7               ;R7 =
COUNTER OF BAR
        SETB   TR0
        RETI

; ****** INT1 ROUTINE ******
INT_1:
        PUSH   DPL
```

```
        PUSH   DPH
        PUSH   ACC
        MOV    DPTR,#8000H
;OR (FC00H) ADDRESS KEYBOARD
        MOVX   A,@DPTR
        ANL    A,#0FH
        ADD    A,#30H
        MOV    R0,A
        MOV    R4,#1
        CJNE   R0,#BUTI,INT1_1
        SETB   IDU
        SJMP   OINT1
INT1_1:
        CJNE   R0,#BUTC,INT1_2
        SETB   CL
        SJMP   OINT1
INT1_2:
        CJNE   R0,#BUTU,OINT1
        SETB   UPU
OINT1:
        POP    ACC
        POP    DPH
        POP    DPL
        RETI

; ****** TIME OVERFLOW ROUTINE
******
S_TM0:
        CLR    TR0             ;STOP
TIMER0
        CLR    EX0             ;
DISABLE INT0
        CLR    EA              ;DISABLE
ALL INTERRUPT

        CLR    C
        MOV    A,#20
        SUBB   A,R7            ;R7 =
COUNTER OF BAR NO.
        JNC    SELOOP
        CLR    C
        MOV    A,#100
        SUBB   A,R7
        JZ     CONT
        LJMP   ERR
CONT:
        LCALL  CALCULATE
        LCALL  COMPARE

        MOV    A,#0C7H         ;
DISPLAY TIME
        LCALL  LCDWI
        LCALL  TDISP

        JMP    SEL
SELOOP:
        MOV    A,#01
        LCALL  LCDWI
        LCALL  INITP
        RETI

; ****** WAIT KEY ROUTINE ******
WAIT_KEY:
        SETB   EX1
```

```
        MOV   R4,#00H          ;R4 =
COUNTER
        MOV   R0,#00H          ;R0 =
DATA
WAIT_KEY1:
        CJNE  R4,#00H,WAIT_KEY2
        LJMP  WAIT_KEY1
WAIT_KEY2:
        CLR   EX1
        RET

; ****** CALCULATE 9 DATA SUB
******
; IN = NONE
; OUT = R3 - R4 (MSB - LSB)
; USE = R0,R1,R3,R4,R5,R6,R7,A,C,DPTR
CALCULATE:
        MOV   R0,#9            ;R0 =
COUNTER
        MOV   DPTR,#0002H      ;
START ADDRESS
        MOV   A,#0             ;CLEAR
ACCUMALATOR
        MOV   R7,A             ;CLEAR
REGISTER
        MOV   R6,A
        MOV   R5,A
        CLR   C
ADDL:
        MOVX  A,@DPTR
        ADDC  A,R7             ; R5 R6
R7 (MSB - LSB)
        MOV   R7,A             ;
RESULT OF SUMMING
        JNC   NOCARRYL
        INC   R6
        CLR   C
NOCARRYL:
        INC   DPTR
        INC   DPTR
        DJNZ  R0,ADDL
        MOV   R0,#9            ;
COUNTER
        MOV   DPTR,#0003H      ;
START ADDRESS
        CLR   C
ADDH:
        MOVX  A,@DPTR
        ADDC  A,R6
        MOV   R6,A
        JNC   NOCARRYH
        INC   R5
        CLR   C
NOCARRYH:
        INC   DPTR
        INC   DPTR
        DJNZ  R0,ADDH

        MOV   R1,#10           ;R1 =
DIVISOR
        MOV   R0,#17           ;R0 =
COUNTER
DIV:
        MOV   A,R5
```

```
        CLR   C             ;MSB-----
LSB
        SUBB  A,R1          ;R5 R6
R7
        JC    CARRY         ;R3  R4
RESULT OF DIVIDE
        MOV   R5,A          ;MSB -
LSB
        I CALL  RESULT
        LCALL  SHIFT_L
        JMP   EXIT
CARRY:
        LCALL  RESULT
        LCALL  SHIFT_L
EXIT:
        DJNZ  R0,DIV
        RET

; ****** SHIFT LEFT R7->R6->R5 SUB
******
; USE = R5,R6,R7,A,C
SHIFT_L:
        CLR   C
        MOV   A,R7          ;RLC R7-
>R6->R5
        RLC   A
        MOV   R7,A
        MOV   A,R6
        RLC   A
        MOV   R6,A
        MOV   A,R5
        RLC   A
        MOV   R5,A
        RET

; ****** SHIFT LEFT RESULT OF
DIVIDE SUB ******
; IN = C
; USE = R3,R4,A
RESULT:
        CPL   C
        MOV   A,R4
        RLC   A
        MOV   R4,A
        MOV   A,R3
        RLC   A
        MOV   R3,A
        RET

;****** COMPARE SUBROUTINE
******
;IN   R3,R4
;OUT   MEM 00CBH - MEM 00D4H
;USE   R0,R1,R3,R4,A,C,DPTR
COMPARE:
        MOV   R1,#10        ;R1 =
COUNTER CHAR
        MOV   R5,#0CBH      ;R5 =
DPL POINTER
        MOV   DPTR,#0002H
COMPLOOP:
        MOV   R0,#08H       ;R0 =
COUNTER
COMPH:
```

```
        INC   DPTR
        MOVX  A,@DPTR
        CLR   C
        SUBB  A,R3          ;R3 =
AVERAGE HIGH BYTE
        JZ    COMPL
        DEC   DPL           ;RETURN
DPTR TO INPUT DATA
        JMP   OUT
COMPL:
        DEC   DPL           ;
COMPARE LOW BYTE
        MOVX  A,@DPTR
        CLR   C
        SUBB  A,R4          ;R4 =
AVERAGE LOW BYTE
        JZ    EQUAL
        JMP   OUT
EQUAL:
        CLR   C
OUT:
        MOV   A,R7
        CPL   C
        RLC   A
        MOV   R7,A
        INC   DPTR
        INC   DPTR
        DJNZ  R0,COMPH
        MOV   A,R7
        PUSH  DPL
        PUSH  DPH
        MOV   DPH,#0
        MOV   DPL,R5
        MOVX  @DPTR,A
        POP   DPH
        POP   DPL
        INC   R5
        INC   DPL
        INC   DPL
        INC   DPL
        INC   DPL
        INC   DPL
        MOV   A,#01
        LCALL LCDWI
        DJNZ  R1,COMPLOOP
        RET
        ; ****** CHECK DIRECTION OF
        BARCODE ******
        ; IN  = MEM 00CBH - 00D4H
        ; OUT = JUMP TO ANOTHER SUB
        ; REG = R2
        SEL:
              MOV   DPTR,#00CBH
              MOVX  A,@DPTR
              MOV   R2,A          ;R2 =
        BUFFER
              MOV   A,#4AH        ;4AH =
        '*' in FORWARD DIRECTION
              XRL   A,R2
              JZ    FORWARD

              MOV   DPTR,#00D4H
              MOVX  A,@DPTR
              MOV   R2,A
```

```
        MOV    A,#29H          ;29H =
'*' in BACKWARD DIRECTION
        XRL    A,R2
        JZ     BACKWARD

        SJMP   ERR

; ****** FORWARD ******
; IN  = FROM CHECK DIRECTION
; OUT = LTBCF
; REG = DPTR
FORWARD:
        LCALL  LTBCF
        LCALL  DISP
        LCALL  ALTER
        LCALL  SAVE
        MOV    R2,#0
        LCALL  DELAY
        JMP    SELOOP

; ****** BACKWARD ******
; IN  = FROM CHECK DIRECTION
; OUT = LTBCR
; REG = DPTR
BACKWARD:
        LCALL  LTBCR
        LCALL  DISP
        LCALL  ALTER
        LCALL  SAVE
        MOV    R2,#0
        LCALL  DELAY
        JMP    SELOOP

; ****** ERROR ******
; IN  = FROM CHECK DIRECTION
; OUT = RETI
; REG = DPTR
ERR:
        MOV    A,#01
        LCALL  LCDWI
        MOV    DPTR,#ERRTABLE
        LCALL  LCDLDP
        MOV    A,#0C0H
        LCALL  LCDLDP
        MOV    R2,#0
        LCALL  DELAY
        LCALL  INITP
        RETI

; ****** CHANGE BAR TO
CHARACTER FORWARD ROUTINE
*******
; IN  = MEM 00CCH - MEM 00D3H
; OUT = MEM 00D4H - MEM 00DBH
; REG = R1,R4,R5,R6,R7,A,DPTR
LTBCF:
        MOV    R5,#8           ;R5 =
COUNTER CHAR.
        MOV    DPTR,#00CCH
        MOV    R7,#0D4H        ;R7 =
POINTER CHAR.
        MOV    R4,#0DCH        ;R4 =
POINTER NO.
LTBCFLOOP:
```

```
        MOVX   A,@DPTR
        MOV    R6,A            ;R6 =
BUFFER
        MOV    R1,#0           ;R1 =
POINTER
        PUSH   DPL
        PUSH   DPH
B2CFLOOP:
        MOV    DPTR,#BARF      ;
LOOK UP BAR TABLE
        MOV    A,R1
        MOVC   A,@A+DPTR
        XRL    A,R6
        JZ     CODEF
        INC    R1
        CJNE   R1,#10,B2CFLOOP
        JMP    ERR
CODEF:
        MOV    A,R1
        MOV    DPTR,#CHARACTER
;LOOK UP CHARACTER TABLE
        MOVC   A,@A+DPTR
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        INC    R7
        POP    DPH
        POP    DPL
        INC    DPTR
        DJNZ   R5,LTBCFLOOP
        RET
; ****** CHANGE BAR TO
CHARACTER BACKWARD ROUTINE
*******
; IN  = MEM 00CCH - MEM 00D3H
; OUT = MEM 00D4H - MEM 00DBH
; REG = R1,R4,R5,R6,R7,A,DPTR
LTBCR:
        MOV    R5,#8           ;R5 =
COUNTER CHAR.
        MOV    DPTR,#00CCH
        MOV    R7,#0DBH        ;R7 =
POINTER CHAR.
        MOV    R4,#0E3H        ;R4 =
POINTER NO.
LTBCRLOOP:
        MOVX   A,@DPTR
        MOV    R6,A            ;R6 =
BUFFER
        MOV    R1,#0           ;R1 =
POINTER
        PUSH   DPL
        PUSH   DPH
B2CRLOOP:
        MOV    DPTR,#BARR      ;
LOOK UP BAR TABLE
        MOV    A,R1
        MOVC   A,@A+DPTR
        XRL    A,R6
        JZ     CODER
        INC    R1
        CJNE   R1,#45,B2CRLOOP
        JMP    ERR
```

```
CODER:
        MOV    A,R1
        MOV    DPTR,#CHARACTER
;LOOK UP CHARACTER TABLE
        MOVC   A,@A+DPTR
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        DEC    R7
        POP    DPH
        POP    DPL
        INC    DPTR
        DJNZ   R5,LTBCRLOOP
        RET

; ****** PREPARE TIME TO UP TO PC
******
; IN  = MEM 9000H , 9005H
; OUT = MEM 00E4H - MEM 00E7H
; REG = R1,R2,R7,A,DPTR
ALTER:
        MOV    R7,#0E4H
        MOV    DPTR,#0110H
;9000H
        MOVX   A,@DPTR
        MOV    R1,A
        ANL    A,#0F0H
        SWAP   A
        ADD    A,#30H
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        INC    R7

        MOV    A,R1
        ANL    A,#0FH
        ADD    A,#30H
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        INC    R7

        MOV    DPTR,#0112H
                        ;9005H
        MOVX   A,@DPTR
        MOV    R1,A
        ANL    A,#0F0H
        SWAP   A
        ADD    A,#30H
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        INC    R7

        MOV    A,R1
        ANL    A,#0FH
        ADD    A,#30H
        MOV    DPH,#0
        MOV    DPL,R7
        MOVX   @DPTR,A
        RET

; ****** DISPLAY ID NO. ******
; IN  = MEM 00D4H - MEM 00DBH
```

```
; OUT = LCD
; REG = R2,R5,A,DPTR
DISP:
        MOV   A,#80H
        MOV   DPTR,#ID          ;
DISPLAY 'ID'
        LCALL  LCDLDP

        MOV   A,#85H            ;SET
ADDRESS OF LCD
        LCALL  LCDWI
        MOV   DPTR,#00D4H       ;
SET DATA
        MOV   R5,#8H            ;R5 =
COUNTER
DISPLOOP:
        MOVX  A,@DPTR
        LCALL  LCDWD
        INC   DPTR
        DJNZ  R5,DISPLOOP
        MOV   R2,#0
        LCALL  DELAY
        RET

; ****** SAVE DATA TO MEMORY
******
; IN =
; OUT =
; REG = R1,R2,R7,A,DPTR
SAVE:
        MOV   R7,#0D4H          ;R7 =
MEMORY POINTER OF NUMBER
        MOV   R1,#8             ;R1 =
COUNTER
        MOV   DPTR,#0100H
;0100H POINTER HIGH BYTE SAVE
DATA
        MOVX  A,@DPTR
        MOV   R2,A              ;R2 =
BUFFER OF POINTER
        INC   DPTR
        MOVX  A,@DPTR           ;00
FEH POINTER LOW BYTE SAVE DATA
        MOV   DPL,A
        MOV   A,R2
        MOV   DPH,A
SAVELOOP:
        PUSH  DPL
        PUSH  DPH
        MOV   DPH,#0
        MOV   DPL,R7
        MOVX  A,@DPTR
        INC   R7
        POP   DPH
        POP   DPL
        MOVX  @DPTR,A
        INC   DPTR
        DJNZ  R1,SAVELOOP
TLOOP:                  ;SAVE TIME
        MOV   R1,#4
        MOV   R7,#0E4H
TLOOP1:
        PUSH  DPL
        PUSH  DPH
```

```
        MOV   DPH,#0
        MOV   DPL,R7
        MOVX  A,@DPTR
        INC   R7
        POP   DPH
        POP   DPL
        MOVX  @DPTR,A
        INC   DPTR
        DJNZ  R1,TLOOP1

        MOV   A,#0FFH          ;STOP
BIT FLAG
        MOVX  @DPTR,A
        MOV   R2,DPL
        MOV   A,DPH
        MOV   DPTR,#0100H
        MOVX  @DPTR,A
        INC   DPL
        MOV   A,R2
        MOVX  @DPTR,A
        RET

; ****** DISPLAY REGISTER ON LCD
SUB ******
; IN  = A
; OUT = NONE
; REG = R2,R7
DISPR:
        PUSH  07
        MOV   R7,A
        ANL   A,#0F0H          ;
CHANGE DATA TO 1ST 4 BIT
        LCALL LTABLEH
        LCALL LTB
        LCALL LCDWD

        MOV   A,R7
        ANL   A,#0FH           ;
CHANGE DATA TO 2ND 4 BIT
        LCALL LTABLEL
        LCALL LTB
        LCALL LCDWD
        POP   07
        RET

;****** LOOK UP TABLE HIGH BIT SUB
******
; IN  = A
; OUT = A
; REG = A,R0,R1
LTABLEH:
        PUSH  DPL
        PUSH  DPH
        PUSH  00
        PUSH  01
        MOV   R0,A             ;
R0=BUFFER
        MOV   R1,#0            ;
R1=COUNTER
        MOV   DPTR,#TABLEH     ;
LOOK UP 1ST 4 BIT TABLE
TBLHL:
        MOV   A,R1
        MOVC  A,@A+DPTR
```

```
        XRL   A,R0
        JZ    OUTH
        INC   R1
        JMP   TBLHL
OUTH:
        MOV   A,R1
        POP   01
        POP   00
        POP   DPH
        POP   DPL
        RET

; ****** LOOK UP TABLE LOW BIT
SUB ******
; IN  = A
; OUT = A
; REG = A,R0,R1
LTABLEL:
        PUSH  DPL
        PUSH  DPH
        PUSH  00
        PUSH  01
        MOV   R0,A          ;P0 =
BUFFER
        MOV   R1,#0         ;R1 =
COUNTER
        MOV   DPTR,#TABLEL      ;
LOOK UP 2ND 4 BIT TABLE
TBLLL:
        MOV   A,R1
        MOVC  A,@A+DPTR
        XRL   A,R0
        JZ    OUTL
        INC   R1
        JMP   TBLLL
OUTL:
        MOV   A,R1
        POP   01
        POP   00
        POP   DPH
        POP   DPL
        RET

; ****** LOOK UP TABLE SUB ******
; IN  = A
; OUT = A
; REG = A,DPTR
LTB:
        PUSH  DPL
        PUSH  DPH
        MOV   DPTR,#TABLE       ;
LOOK UP NUMBER TABLE
        MOVC  A,@A+DPTR
        POP   DPH
        POP   DPL
        RET

;************SETTIME
SUB.**********************
SETTIME:
        MOV   R6,#8EH       ;WRITE
PROTECTION
        MOV   R7,#00H
        LCALL BYTEWR
```

```
        MOV   R6,#80H      ;WRITE
SECOND AND CLR CHFLAG
        MOV   R7,#00H      ;SEC=0
        LCALL  BYTEWR

        MOV   R6,#82H      ;WRITE
MINUTE
        MOV   DPTR,#0112H
        MOVX  A,@DPTR
        MOV   R7,A         :MIN=0
        LCALL  BYTEWR

        MOV   R6,#84H      ;WRITE
HOUR
        MOV   DPTR,#0110H
        MOVX  A,@DPTR
        MOV   R7,A         ;HOUR=0
        LCALL  BYTEWR

        MOV   R6,#8EH      ;WRITE
PROTECTION "ACTIVE"
        MOV   R7,#80H
        LCALL  BYTEWR
        RET

;*********** BYTEWR SUB.
**********************
;WRITE SINGLE BYTE TO STC
;IN = R6 COMMAND
;  = R7 DATA
;REG = A,B,R6,R7
BYTEWR:
        CLR   P1.4         ;COMMAND
BYTE "WRITE"
        LCALL  DELAY4
        SETB  P1.6         ;RST=1
        LCALL  DELAY4
        MOV   B,#8         ;SEND
COMMAND
        CLR   C
BYTEWR1:
        MOV   A,R6
        RRC   A
        MOV   R6,A
        MOV   P1.4,C
        LCALL  SCLKRW
        DJNZ  B,BYTEWR1

        MOV   B,#8         ;SEND DATA
BYTE
        CLR   C
BYTEWR2:
        MOV   A,R7
        RRC   A
        MOV   R7,A
        MOV   P1.4,C
        LCALL  SCLKRW
        DJNZ  B,BYTEWR2
        CLR   P1.6         ;RST=0
        LCALL  DELAY4
        RET

;********** BYTERD SUB.**********
```

```
;READ SINGLE BYTE FROM STC
;IN = R6 COMMAND
;OUT = R7 DATA
;REG = A,B,R6,R7
BYTERD: SETB   P1.4
        LCALL  DELAY4
        SETB   P1.6
        LCALL  DELAY4
        MOV    B,#8
BYTERD1:
        MOV    A,R6
        RRC    A
        MOV    R6,A
        MOV    P1.4,C
        LCALL  SCLKCOM
        DJNZ   B,BYTERD1

        MOV    B,#8
        MOV    R7,#0
BYTERD2:LCALL  SCLKRW
        MOV    A,R7
        MOV    C,P1.4
        RRC    A
        MOV    R7,A
        DJNZ   B,BYTERD2
        CLR    P1.6
        LCALL  DELAY4
        RET

;********** SCLKCOM SUB. ********
;SERIAL CLOCK FOR WRITE
COMMAND
;A FALLING EDGE
;FOLLOWED BY A RISING EDGE
SCLKCOM:
        CLR    P1.5
        LCALL  DELAY4
        SETB   P1.5
        LCALL  DELAY4
        RET

;********** SCLKRW SUB.*********
;SERIAL CLOCK FOR READ/WRITE
DATA
;A RISING EDGE
;FOLLOWED BY A FALLING EDGE
SCLKRW:
        SETB   P1.5
        LCALL  DELAY4
        CLR    P1.5
        LCALL  DELAY4
        RET

;************ DELAY SUB. **********
;PULSE DELAY
;REG=R1
DELAY4:
        MOV    R1,#5
        DJNZ   R1,$
        RET

;************* TDISP SUB. *********
;TIME DISPLAY
TDISP:
```

```
        MOV   R6,#85H          ;READ
HOUR
        LCALL  BYTERD
        MOV   R6,#85H
        LCALL  BYTERD
        MOV   A,R7
        MOV   DPTR,#0110H      ;
OUT HOUR
        MOVX  @DPTR,A
        LCALL  DISPR
;*************
        MOV   A,#3AH           ;
DISPLAY ':'
        LCALL  LCDWD
        MOV   R6,#83H          ;READ
MINUTE
        LCALL  BYTERD
        MOV   A,R7
        MOV   DPTR,#0112H      ;
OUT MINUTE
        MOVX  @DPTR,A
        LCALL  DISPR
;************
        MOV   A,#3AH
        LCALL  LCDWD
        MOV   R6,#81H          ;READ
SECOND
        LCALL  BYTERD
        MOV   A,R7
        LCALL  DISPR
;************
        RET

;******* LCDWI SUB ************
;LCD WRITE INSTRUCTION (RS=0)
;IN =A
;REG=A
LCDWI:
        PUSH   DPH
        PUSH   DPL
        MOV   DPTR,#LCDWRC
        MOVX  @DPTR,A
        LCALL  DELAYP
        LCALL  DELAYP
        POP   DPL
        POP   DPH
        RET

;********* LCDWD SUB *********
;LCD WRITE DATA (RS=1)
;IN =A
;REG =A
LCDWD:
        PUSH   DPH
        PUSH   DPL
        MOV   DPTR,#LCDWRD
        MOVX  @DPTR,A
        LCALL  DELAYP
        LCALL  DELAYP
        POP   DPL
```

```
        POP   DPH
        RET

; ****** DELAY SUBROUTINE ******
DELAYP:
        PUSH  PSW
        SETB  RS1
        SETB  RS0
        MOV   R3,#2   ; >= 2
DELAY1P:
        MOV   R4,#00H
        DJNZ  R4,$
        DJNZ  R3,DELAY1P
        CLR   RS1
        CLR   RS0
        POP   PSW
        RET

; ****** LCDLDP SUB ******
; LOAD PMEM TO LCD-MODULE
; IN  = DPTR
; OUT = NONE
; REG = A,R2,DPTR
LCDLDP:
        LCALL LCDWI           ;
LOAD ONE LINE
        MOV   R2,#16          ;16
CHAR.
LCDLDPS1:
        CLR   A
        MOVC  A,@A+DPTR       ;
PMEM=MOVC
        LCALL LCDWD           ;
WRITE DATA
        INC   DPTR
        DJNZ  R2,LCDLDPS1
        RET

; ****** DELAY SUB ******
; IN  = R2
; OUT = NONE
; REG = A,R2,R3
DELAY:
        PUSH  03
DELAY0:
        MOV   R3,#80H
DELAY1:
        MOV   A,#8
DELAY2:
        DEC   A
        JNZ   DELAY2
        DJNZ  R3,DELAY1
        DJNZ  R2,DELAY0
        POP   03
        RET

; ****** TABLE *****
LINK1:      DB   ' DATA LINK '
LINK2:      DB   '  TO PC  '
LINK3:      DB   ' FINISHED '
ID:         DB   'ID.       '
TIME:       DB   'SET CLOCK  '
ERRTABLE:   DB   '  ERROR   '
            DB   ' TRY AGAIN '
```

```
READY:          DB   '  READY
'
TABLEH:         DB   00
H,10H,20H,30H,40H,50H,60H,70H
                DB   80
H,90H,0A0H,0B0H,0C0
H,0D0H,0E0H,0F0H
TABLEL:         DB   00
H,01H,02H,03H,04H,05H,06H,07H
                DB   08
H,09H,0AH,0BH,0CH,0DH,0EH,0FH
TABLE:          DB
'0','1','2','3','4','5','6','7'
                DB   '8','9','A','
B','C','D','E','F'
BARF:           DB   90
H,30H,0B0H,18H,98H,38H,12H,92H
                DB   32
H,1AH,84H,24H,0A4H,0CH,8CH,2CH
                DB   06
H,86H,26H,0EH,81H,21H,0A1H,09H
                DB   89
H,29H,03H,83H,23H,0BH,0C0H,60H
                DB   0
E0H,48H,0C8H,68H,42H,0C2H,62H,4AH
                DB   54H,51H,45H,15H
BARR:           DB   84
H,86H,06H,8CH,0CH,0EH,0A4H,24H
                DB   26
H,2CH,90H,92H,12H,98H,18H,1AH
                DB   0
B0H,30H,32H,38H,0C0H,0C2H,42H,0C8H
                DB   48
H,4AH,0E0H,60H,62H,68H,81H,83H
                DB   03
H,89H,09H,0BH,0A1H,21H,23H,29H
                DB   15H,45H,51H,54H
CHARACTER:      DB
'1','2','3','4','5','6','7','8'
                DB   '9','0','A','
B','C','D','E','F'
                DB
'G','H','I','J','K','L','M','N'
                DB
'O','P','Q','R','S','T','U','V'
                DB   'W','X','Y','Z','-','.','
','*'
                DB   '$','/','+','%','r'
ASCII:          DB
'0','1','2','3','4','5','6','7'
                DB   '8','9','0'
; ******* END OF TABLE ******
                END
```

โปรแกรมการส่งข้อมูลไปยังคอมพิวเตอร์โดยใช้ภาษา Visual Basic

```
Private Sub Form_Load()
   ' Use COM2.
   Comm1.CommPort = 2
   ' 9600 baud, no parity, 8 data, and 1 stop bit.
   Comm1.Settings = "9600,N,8,1"
   ' Tell the control to read entire buffer when Input is used.
   Comm1.InputLen = 1
   Comm1.PortOpen = True
   h = ""
   Open "TESTFILE.txt" For Output As #1    ' Open file for output.
   k = 0
   j = 1
   Do
      Dummy = DoEvents()
      INSTRING$ = Comm1.Input
    If j Mod 12 <> 0 Then
      If (INSTRING$ = "3" Or k = 1) And (INSTRING$ = "0 " Or "1" Or "3" Or "4" Or "5" Or "6" Or "7" Or
"8" Or "9") And (INSTRING$ <> "") And (INSTRING$ <> chr(255)) Then
         h = h + INSTRING$
         k = 1
         j = j + 1
      End If
   ElseIf j Mod 12 = 0 Then
      If (INSTRING$ = "3" Or k = 1) And (INSTRING$ = "0 " Or "1" Or "3" Or "4" Or "5" Or "6" Or "7" Or
"8" Or "9") And (INSTRING$ <> "") And (INSTRING$ <> chr(255)
         h = h + INSTRING$ + Chr$(13) + Chr$(10)
         k = 1
         j = j + 1
      End If
   End If
   Loop Until INSTRING$ = chr(255)
   Print #1, h
   Comm1.PortOpen = False
   Close #1
End Sub
```

ภาคผนวก ข

ข้อมูลวงจรรวม

8031AH/8051AH
8032AH/8052AH
MCS® 51
NMOS SINGLE-CHIP 8-BIT MICROCONTROLLERS

Automotive

- **High Performance HMOS Process**
- **Internal Timers/Event Counters**
- **2-Level Interrupt Priority Structure**
- **32 I/O Lines (Four 8-Bit Ports)**
- **64K Program Memory Space**
- **Security Feature Protects EPROM Parts Against Software Piracy**
- **Boolean Processor**
- **Bit-Addressable RAM**
- **Programmable Full Duplex Serial Channel**
- **111 Instructions (64 Single-Cycle)**
- **64K Data Memory Space**
- **Available in PLCC and DIP Packages**

The MCS® 51 microcontroller products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
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
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5

NOTICE:

This datasheet contains information on products in full production. Specifications within this datasheet are subject to change without notice. Verify with your local Intel sales office that you have the latest datasheet before finalizing a design.

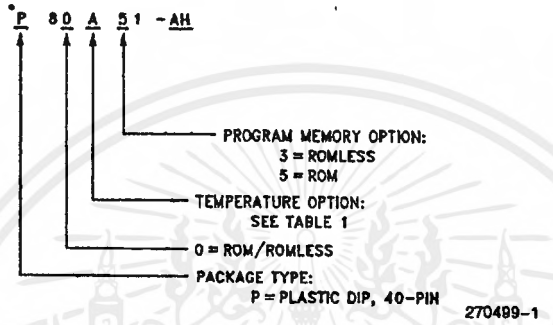
February 1995 Order Number: 270499-006

PRODUCT OPTIONS

Intel's extended and automotive temperature range products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to +70°C ambient. With the extended temperature range option, operational characteristics are guaranteed over the temperature range of −40°C to +85°C ambient. For the automotive temperature range option, operational characteristics are guaranteed over the temperature range of −40°C to +110°C ambient.

The automotive, extended, and commercial temperature versions of the MCS 51 microcontroller product families are available with or without burn-in options.



***Example:**
P, 80A51AH indicates an automotive temperature range version of the 8051AH in a PDIP package with 4 Kbyte ROM program memory.

Figure 1. MCS® 51 Microcontroller Product Family Nomenclature

Table 1. Temperature Options

Temperature Classification	Temperature Designation	Operating Temperature °C Ambient	Burn-In Option
Extended	T	−40 to +85	Standard
	L	−40 to +85	Extended
Automotive	A	−40 to +110	Standard

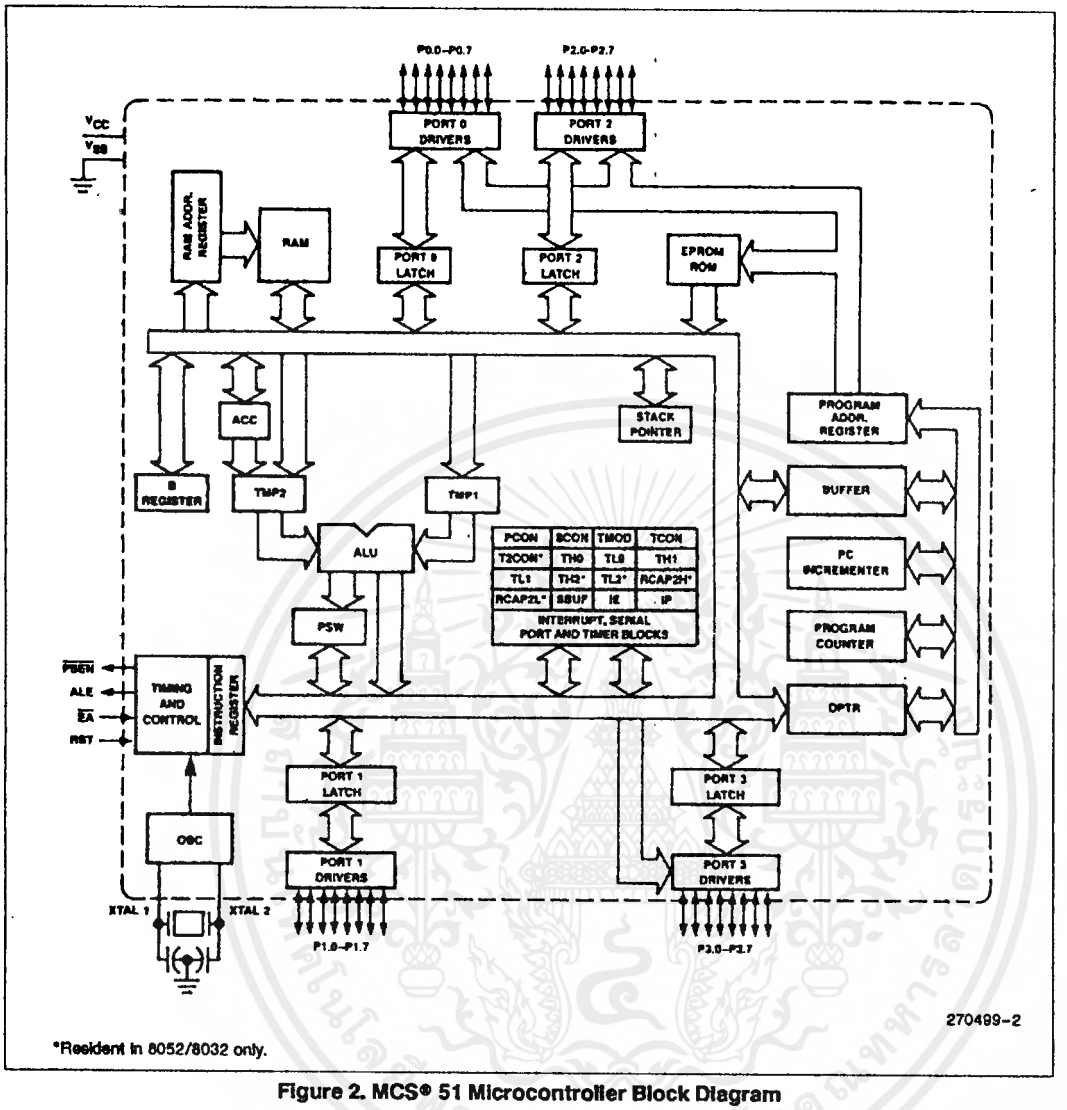


Figure 2. MCS® 51 Microcontroller Block Diagram

PIN DESCRIPTIONS

V_{CC}

Supply voltage.

V_{SS}

Circuit ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs. Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also outputs the code bytes during program verification of the ROM. External pullups are required.

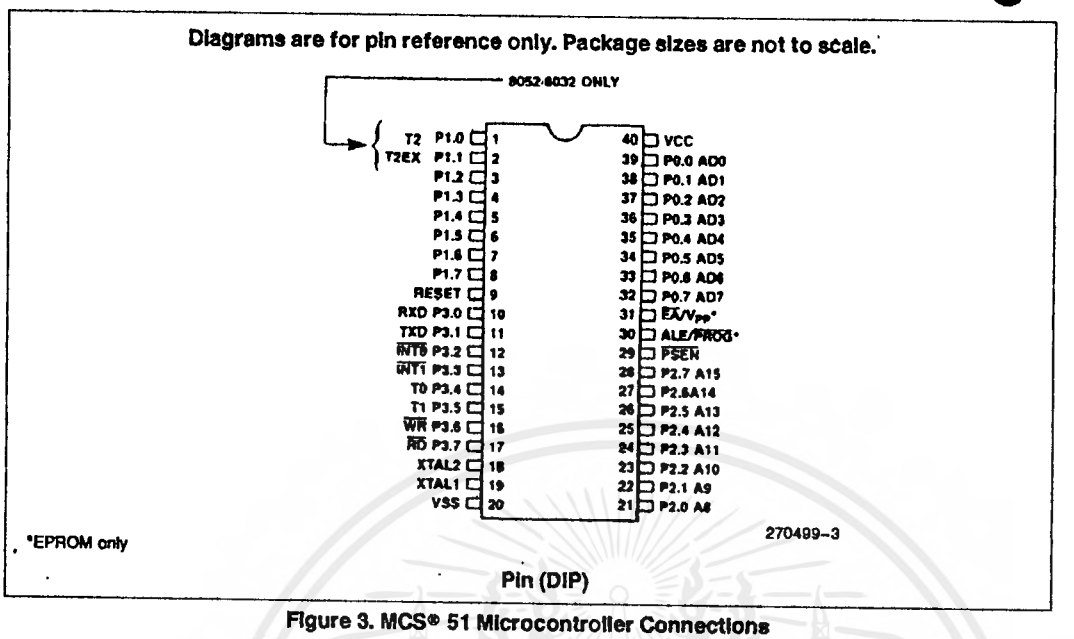


Figure 3. MCS® 51 Microcontroller Connections

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally pulled low will source current (I_{IL} on the datasheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during program verification of the ROM.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally pulled low will source current (I_{IL} on the datasheet) 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 during program verification of the ROM.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current (I_{IL} on the datasheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS 51 microcontroller family, as listed below:

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

RESET

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.

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.

$\overline{PSEN}$

Program Store Enable is the read strobe to external Program Memory.

When the device 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 V_{SS} in order to enable any MCS 51 microcontroller device to fetch code from external Program memory locations 0 to 0FFFH (0 to 1FFFH, in the 8032AH and 8052AH).

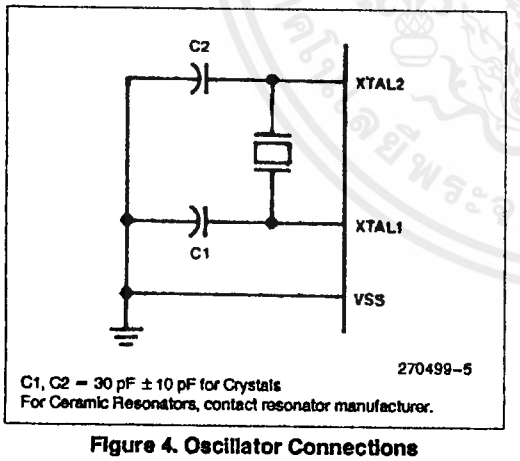


Figure 4. Oscillator Connections

XTAL1

Input to the inverting oscillator amplifier.

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 4. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 5. 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 high and low times specified on the datasheet must be observed.

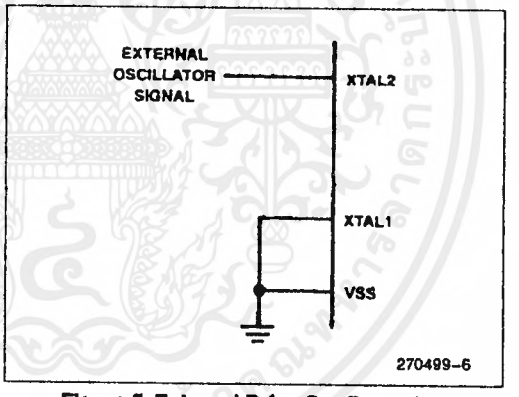


Figure 5. External Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature
Under Bias .....−40°C to +110°C
Storage Temperature .....−65°C to +150°C
Voltage on $\overline{EA}$/VPP Pin to V_{SS} ...−0.5V to +21.5V
Voltage on Any Other Pin to V_{SS} ....−0.5V to +7V
Power Dissipation.....1.5W
Based on package heat transfer limitations not device power consumption.
Maximum Case Temperature
Under Bias .....+125°C

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

Typical Thermal Resistance Junction to Ambient (θ_{JA})

Package	θ_{JA}
Plastic DIP	75°C/W

DC CHARACTERISTICS $T_A = -40°C$ to $+110°C$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)	2.1	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST	2.6	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*		0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)	2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH All Others		−800 −500	μA μA	V_{IN} = 0.45V V_{IN} = 0.45V
I_{IL2}	Logical 0 Input Current (XTAL2)		−4.0	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0)		±10	μA	$0.45 \le V_{IN} \le V_{CC}$
I_{IH1}	Input Current to RST to Activate Reset		500	μA	$V_{IN} < (V_{CC} - 1.5V)$
I_{CC}	Power Supply Current: 8031/8051 8031AH/8051AH 8032AH/8052AH		 175 135 175	 mA mA mA	 All Outputs Disconnected;
C_{IO}	Pin Capacitance		10	pF	Test freq = 1 MHz

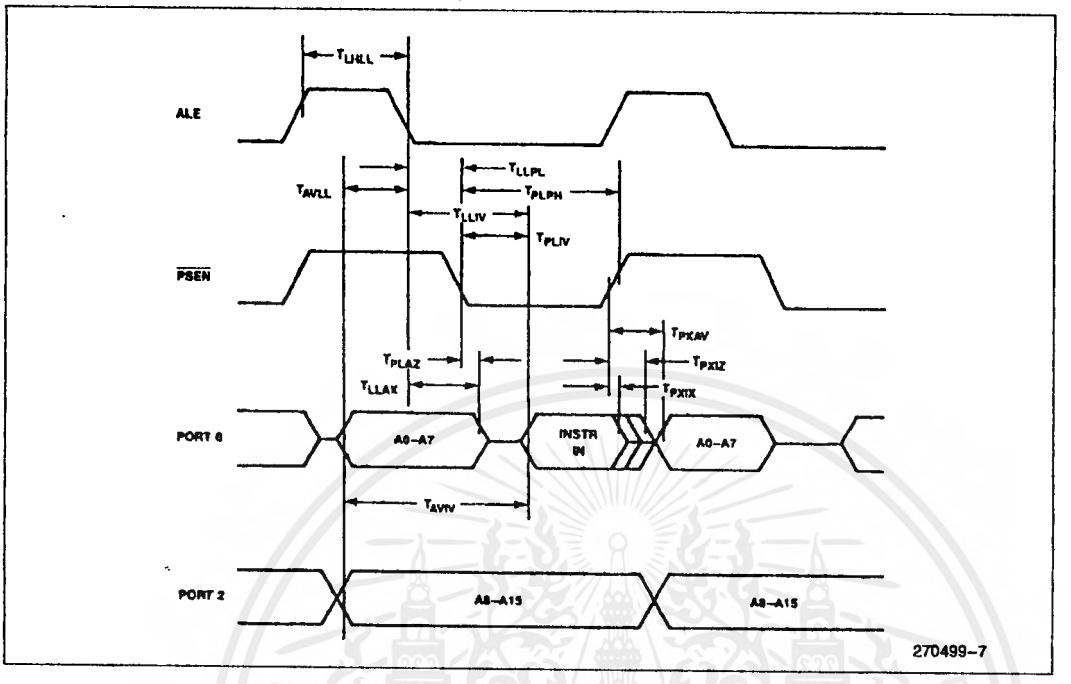
*NOTE:
Capacitive loading on Ports 0 and 2 may cause noise pulses to be superimposed on the VOLs of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

AC CHARACTERISTICS

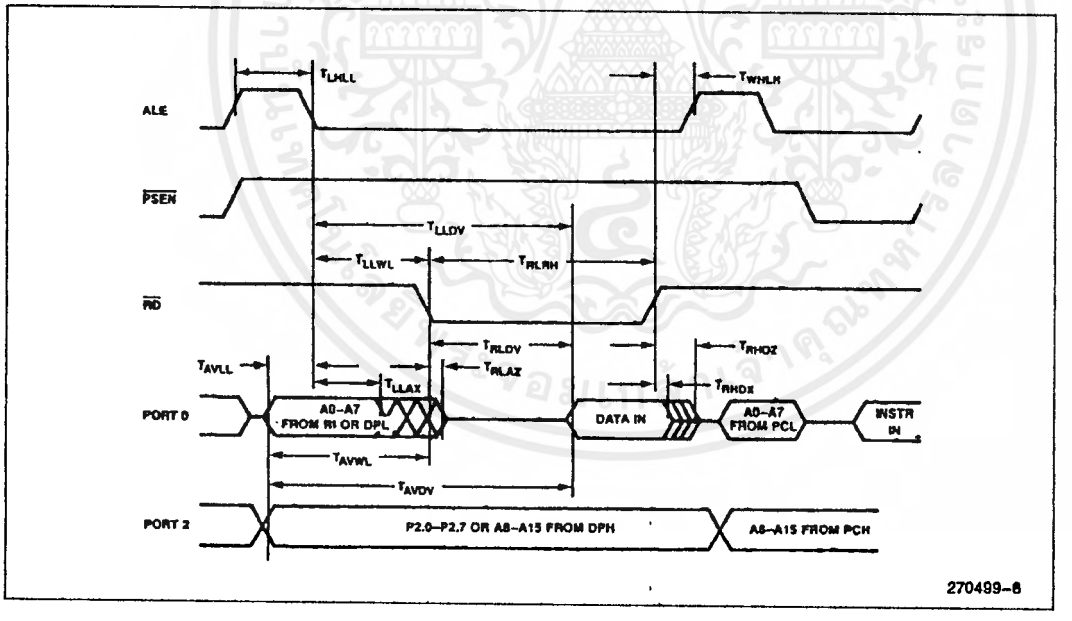
$T_A = -40°C$ to $+110°C$; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$;
Load Capacitance for Port 0, ALE, and PSEN = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/T_{CLCL}$	Oscillator Frequency			3.5	12.0	MHz
T_{LHLL}	ALE Pulse Width	127		$2T_{CLCL}-40$		ns
T_{AVLL}	Address Valid to ALE Low	43		$T_{CLCL}-40$		ns
T_{LLAX}	Address Hold after ALE Low	48		$T_{CLCL}-35$		ns
T_{LLIV}	ALE Low to Valid Instr In		233		$4T_{CLCL}-100$	ns
T_{LLPL}	ALE Low to $\overline{PSEN}$ Low	58		$T_{CLCL}-25$		ns
T_{PLPH}	$\overline{PSEN}$ Pulse Width	215		$3T_{CLCL}-35$		ns
T_{PLIV}	$\overline{PSEN}$ Low to Valid Instr In		125		$3T_{CLCL}-125$	ns
T_{PXIX}	Input Instr Hold after $\overline{PSEN}$	0		0		ns
T_{PXIZ}	Input Instr Float after $\overline{PSEN}$		63		$T_{CLCL}-20$	ns
T_{PXAV}	$\overline{PSEN}$ to Address Valid	75		$T_{CLCL}-8$		ns
T_{AVIV}	Address to Valid Instr In		302		$5T_{CLCL}-115$	ns
T_{PLAZ}	$\overline{PSEN}$ Low to Address Float		20		20	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		$5T_{CLCL}-165$	ns
T_{RHDX}	Data Hold after $\overline{RD}$ High	0		0		ns
T_{RHDZ}	Data Float after $\overline{RD}$ High		97		$2T_{CLCL}-70$	ns
T_{LLDV}	ALE Low to Valid Data In		517		$8T_{CLCL}-150$	ns
T_{AVDV}	Address to Valid Data In		585		$9T_{CLCL}-165$	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		$4T_{CLCL}-130$		ns
T_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		$T_{CLCL}-60$		ns
T_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7T_{CLCL}-150$		ns
T_{WHQX}	Data Hold after $\overline{WR}$ High	33		$T_{CLCL}-50$		ns
T_{RLAZ}	$\overline{RD}$ Low to Address Float		20		20	ns
T_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$T_{CLCL}-40$	$T_{CLCL}+40$	ns

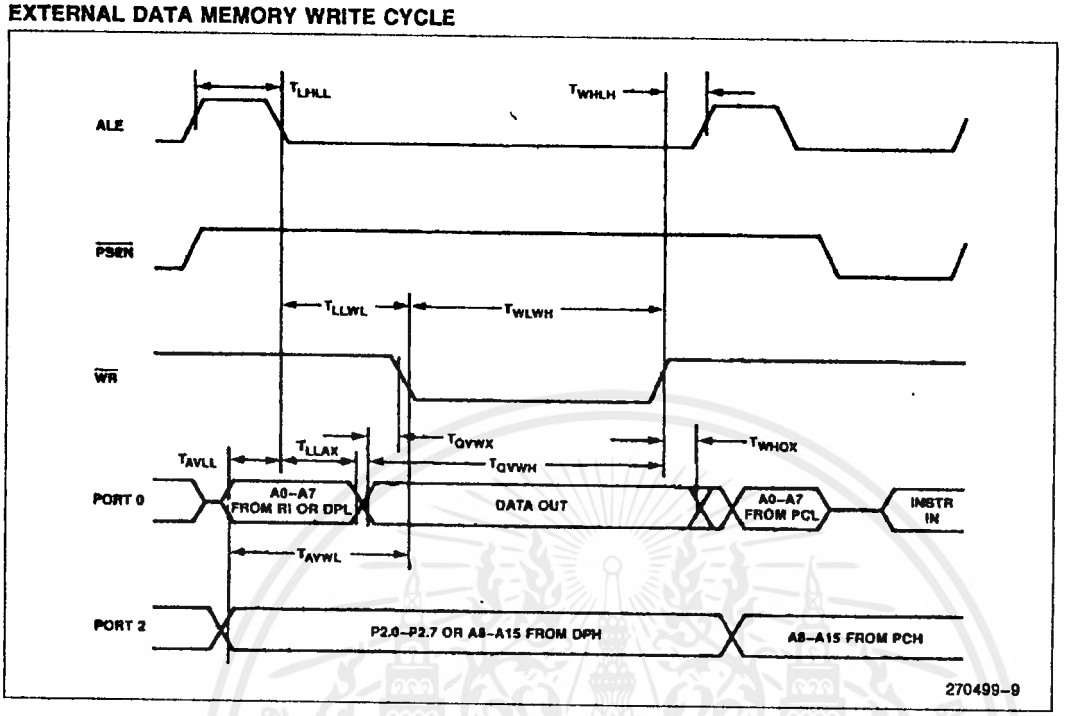
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

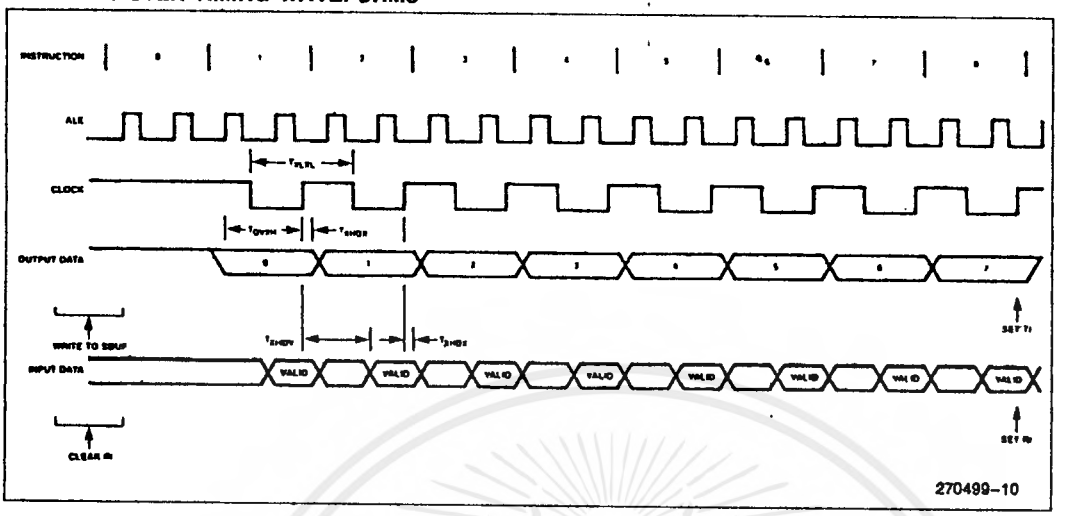


SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = −40°C to +110°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		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		$2T_{CLCL}-117$		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 TIMING WAVEFORMS

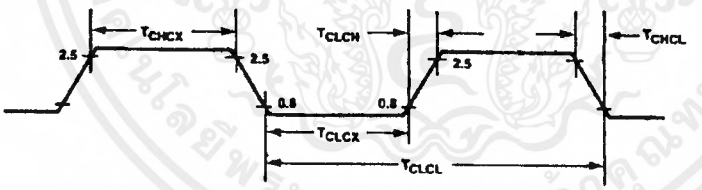


270499–10

EXTERNAL CLOCK DRIVE

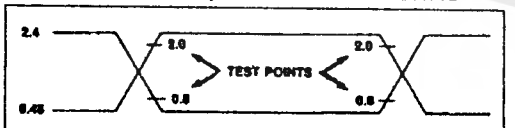
Symbol	Parameter	Min	Max	Units
$1/T_{CLCL}$	Oscillator Frequency	3.5	12	MHz
T_{CHCX}	High Time	20		ns
T_{CLCX}	Low Time	20		ns
T_{CLCH}	Rise Time		20	ns
T_{CHCL}	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORMS



270499–11

AC TESTING INPUT, OUTPUT WAVEFORMS



270499–12

AC Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

DATASHEET REVISION HISTORY

The following are key differences between this datasheet and the -005 version:

1. The "preliminary" status was dropped and replaced with production status (no label).
2. Trademarks were updated.

The following are key differences between the -005 and the -004 version of the datasheet:

1. Preliminary notice was placed on the title page.
2. Figure 2. MCS 51 Block Diagram was modified to include the note found at the bottom of the figure.
3. RST pin in Figure 3 was changed to RESET.
4. RST pin description was changed to RESET pin description.
5. Power dissipation note added below Power dissipation listing in Absolute Maximum Ratings.
6. V_{IH} and V_{IH1} were changed by 0.1V to reflect test conditions.
7. T_{PLPH} was corrected to show test program timing.

The following are key differences between the -004 datasheet and the -003 version of the datasheet:

1. The title was changed to 8031AH/8051AH, 8032AH/8052AH MCS 51 NMOS Single-Chip 8-Bit Microcontrollers.
2. "NC" pin labels changed to "Reserved" in Figure 3.
3. Capacitor value for ceramic resonators deleted in Figure 4.

The following are key differences between the -001 and the -002 version of the datasheet:

1. The title was changed to 8031/8051, 8031AH/8051AH, 8032AH/8052AH, 8751H MCS 51 NMOS Single-Chip 8-Bit Microcontrollers.
2. Removed 8751H-8 from the datasheet.
3. Removed reference to LCC package version.
4. Removed burn-in options from Table 1.
5. Added pin count to Figure 1.
6. Test conditions for I_{IL1} and I_{IH} specifications added to the DC Characteristics.
7. Datasheet revision history added.

DALLAS SEMICONDUCTOR

DS1202, DS1202S
Serial Timekeeping Chip

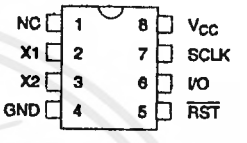
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation
- 24 x 8 RAM for scratchpad data storage
- Serial I/O for minimum pin count
- 2.0-5.5 volt full operation
- Uses less than 300 nA at 2 volts
- Single-byte or multiple-byte (burst mode) data transfer for read or write of clock or RAM data
- 8-pin DIP or optional 16-pin SOIC for surface mount
- Simple 3-wire interface
- TTL-compatible (V_{CC} = 5V)
- Optional industrial temperature range -40°C to +85°C

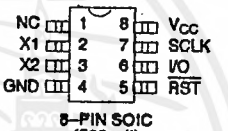
ORDERING INFORMATION

DS1202	8–pin DIP
DS1202S	16–pin SOIC
DS1202S8	8–pin SOIC

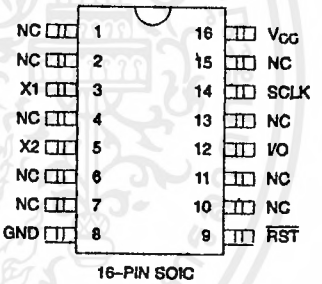
PIN ASSIGNMENT



8 PIN DIP



8–PIN SOIC (208 mil)



16–PIN SOIC

PIN DESCRIPTION

NC	–	No Connection
X1, X2	–	32.768 KHz Crystal Input
GND	–	Ground
$\overline{RST}$	–	Reset
I/O	–	Data Input/Output
SCLK	–	Serial Clock
V_{CC}	–	Power Supply Pin

DESCRIPTION

The DS1202 Serial Timekeeping Chip contains a real time clock/calendar and 24 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time 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 an AM/PM indicator. Interfacing the DS1202 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: (1) $\overline{RST}$ (Reset), (2) I/O (Data line), and (3) SCLK (Serial clock). Data can be transferred to and from the clock/

RAM one byte at a time or in a burst of up to 24 bytes. The DS1202 is designed to operate on very low power and retain data and clock information on less than 1 microwatt.

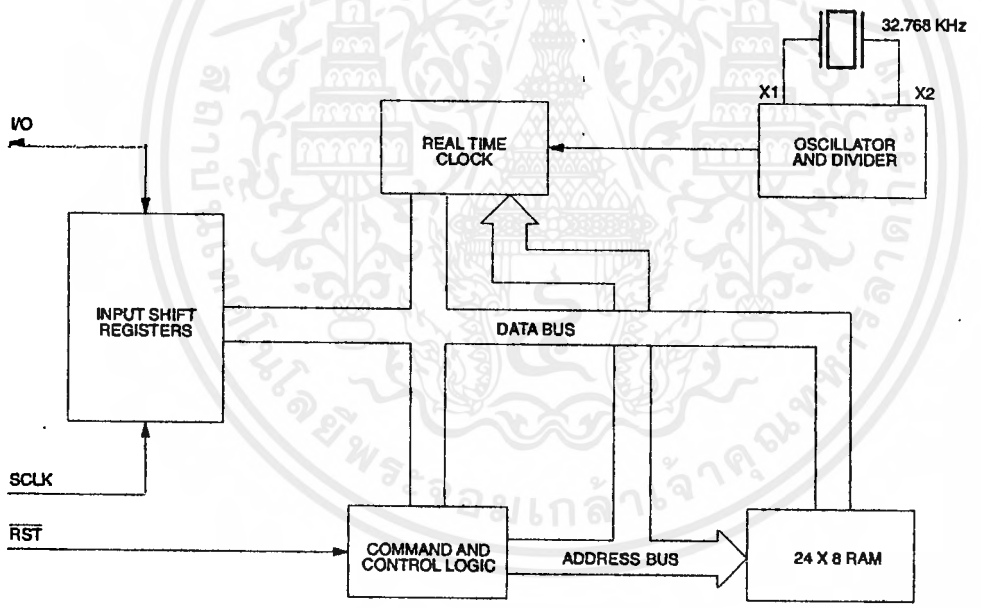
OPERATION

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logic, oscillator, real time clock, and RAM. To initiate any transfer of data, $\overline{RST}$ is taken high and eight bits are loaded into the shift register providing both address and command information. Data is serially input on the rising edge of the SCLK. The first eight bits specify which of 32 bytes will be accessed, whether a read or write cycle will take place, and whether a byte or burst mode transfer is to occur. After the first eight clock cycles have occurred which load the command word into the shift register, additional clocks will output data for a read or input data for a write. The number of clock pulses equals eight plus eight for byte mode or eight plus up to 192 for burst mode.

COMMAND BYTE

The command byte is shown in Figure 2. Each data transfer is initiated by a command byte. The MSB (Bit 7) must be a logic 1. If it is zero, further action will be terminated. Bit 6 specifies clock/calendar data if logic 0 or RAM data if logic 1. Bits one through five specify the designated registers to be input or output, and the LSB (Bit 0) specifies a write operation (input) if logic 0 or read operation (output) if logic 1. The command byte is always input starting with the LSB (bit 0).

DS1202 BLOCK DIAGRAM Figure 1



ADDRESS/COMMAND BYTE Figure 2

7	8	5	4	3	2	1	0
1	RAM / $\overline{CK}$	A4	A3	A2	A1	A0	RD / $\overline{W}$

RESET AND CLOCK CONTROL

All data transfers are initiated by driving the $\overline{RST}$ input high. The $\overline{RST}$ input serves two functions. First, $\overline{RST}$ turns on the control logic which allows access to the shift register for the address/command sequence. Second, the $\overline{RST}$ signal provides a method of terminating either single byte or multiple byte data transfer. A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, data must be valid during the rising edge of the clock and data bits are output on t he falling edge of clock. All data transfer terminates if the $\overline{RST}$ input is low and the I/O pin goes to a high impedance state. Data transfer is illustrated in Figure 3.

DATA INPUT

Following the eight SCLK cycles that input a write command byte, a data byte is input on the rising edge of the next eight SCLK cycles. Additional SCLK cycles are ignored should they inadvertently occur. Data is input starting with bit 0.

DATA OUTPUT

Following the eight SCLK cycles that input a read command byte, a data byte is output on the falling edge of the next eight SCLK cycles. Note that the first data bit to be transmitted occurs on the first falling edge after the last bit of the command byte is written. Additional SCLK cycles retransmit the data bytes should they inadvertently occur so long as $\overline{RST}$ remains high. This operation permits continuous burst mode read capability. Data is output starting with bit 0.

BURST MODE

Burst mode may be specified for either the clock/calendar or the RAM registers by addressing location 31 decimal (address/command bits one through five = logical one). As before, bit six specified clock or RAM and bit 0 specifies read or write. There is no data storage capacity at locations 8 through 31 in the Clock/Calendar Registers or locations 24 through 31 in the RAM registers. When writing to the clock registers in the burst mode, the first eight registers must be written in order for the data to be transferred.

However, when writing to RAM in burst mode it is not necessary to write all 24 bytes for the data to transfer. Each byte that is written to will be transferred to RAM regardless of whether all 24 bytes are written or not.

CLOCK/CALENDAR

The clock/calendar is contained in eight write/read registers as shown in Figure 4. Data contained in the clock/calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the seconds register is defined as the clock halt flag. When this bit is set to logic 1, the clock oscillator is stopped and the DS1202 is placed into a low-power standby mode with a current drain of not more than 100 nanoamps. When this bit is written to logic 0, the clock will start.

AM-PM/12-24 MODE

Bit 7 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

WRITE PROTECT REGISTER

Bit 7 of write protect register is the write protect bit. The first seven bits (bits 0-6) are forced to zero and will always read a zero when read. Before any write operation to the clock or RAM, bit 7 must be zero. When high, the write protect bit prevents a write operation to any other register.

CLOCK/CALENDAR BURST MODE

The clock/calendar command byte specifies burst mode operation. In this mode the eight clock/calendar registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

RAM

The static RAM is 24 x 8 bytes addressed consecutively in the RAM address space.

RAM BURST MODE

The RAM command byte specifies burst mode operation. In this mode, the 24 RAM registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

REGISTER SUMMARY

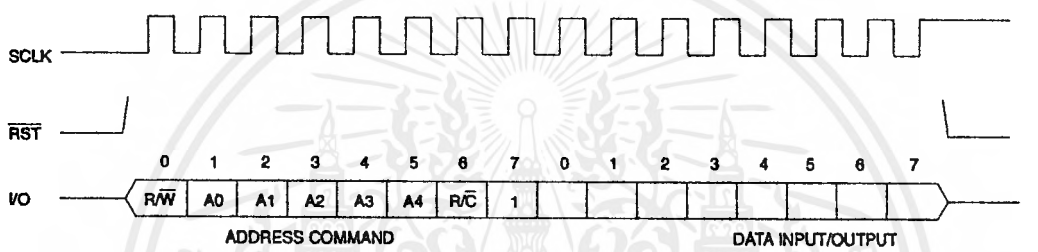
A register data format summary is shown in Figure 4.

CRYSTAL SELECTION

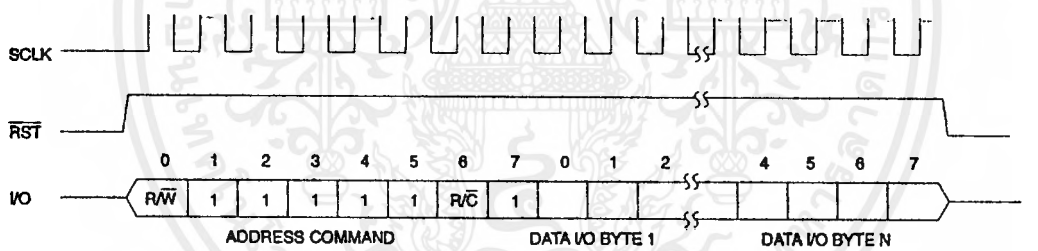
A 32.768 KHz crystal, Daiwa Part No. DT26S, Seiko Part No. DS-VT-200 or equivalent, can be directly connected to the DS1202 via pins 2 and 3 (X1, X2). The crystal selected for use should have a specified load capacitance (CL) of 6 pF. The crystal is connected directly to the X1 and X2 pins. There is no need for external capacitors or resistors. Note: X1 and X2 are very high impedance nodes. It is recommended that they and the crystal be guard–ringed with ground and that high frequency signals be kept away from the crystal area. For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks".

DATA TRANSFER SUMMARY Figure 3

SINGLE BYTE TRANSFER

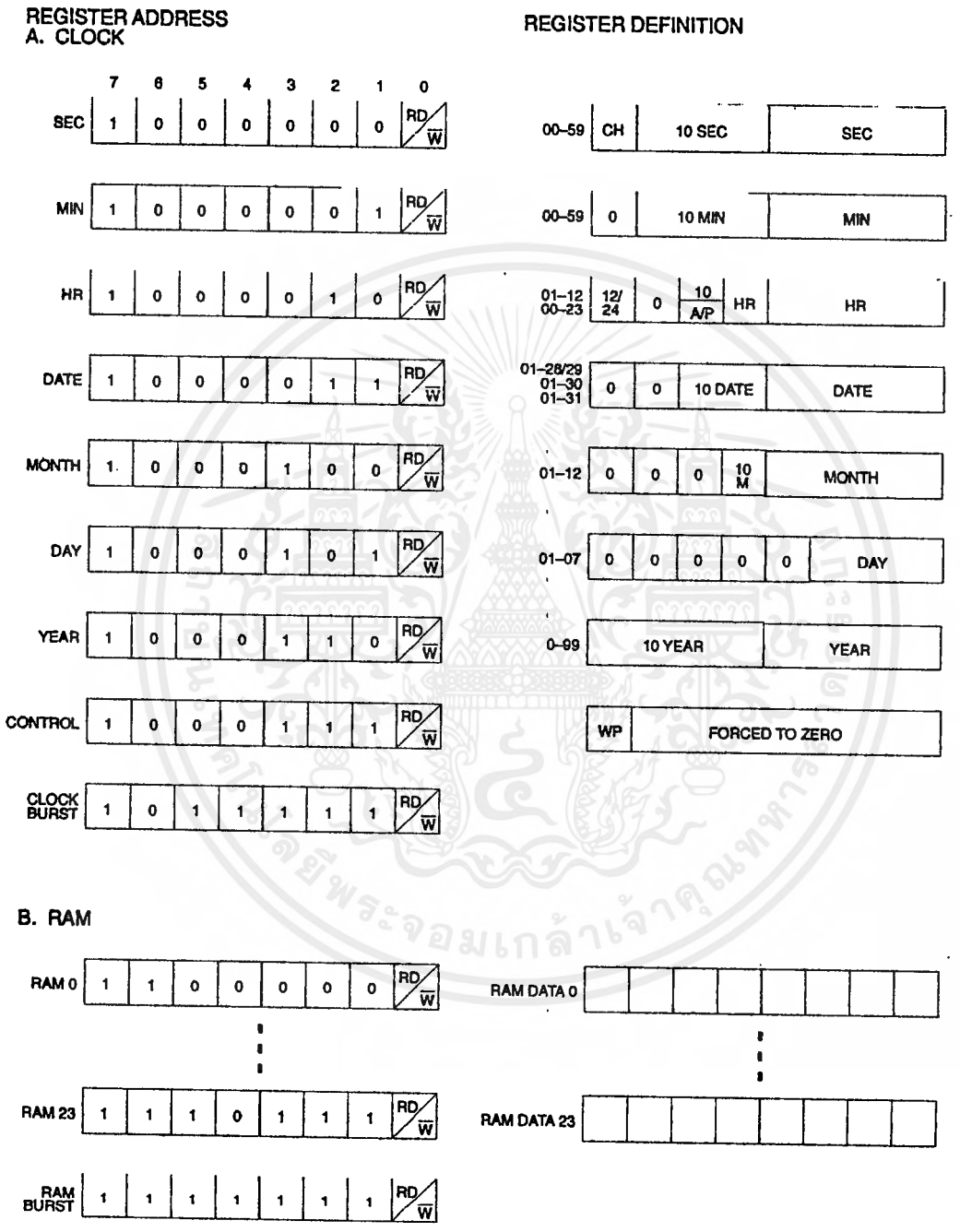


BURST MODE TRANSFER



FUNCTION	BYTE N	SCLK n
CLOCK	8	72
RAM	24	200

REGISTER ADDRESS/DEFINITION Figure 4



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.3V to +7.0V
Operating Temperature	0°C to 70°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

(0°C to 70°C)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}		2.0		5.5	V	1
Logic 1 Input	V_{IH}		2.0		V_{CC}+0.3	V	1
Logic 0 Input	V_{IL}	V_{CC}=2.0V	–0.3		+0.3	V	1
		V_{CC}=5V	–0.3		+0.8		

DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}				+500	µA	6
I/O Leakage	I_{LO}				+500	µA	6
Logic 1 Output	V_{OH}	V_{CC}=2V	1.6			V	2
		V_{CC}=5V	2.4				
Logic 0 Output	V_{OL}	V_{CC}=2V			0.4	V	3
		V_{CC}=5V			0.4		
Active Supply Current	I_{CC}	V_{CC}=2V			.4	mA	5
		V_{CC}=5V			1.2		
Timekeeping Current	I_{CC1}	V_{CC}=2V			0.3	µA	4
		V_{CC}=5V			1		
Leakage Current	I_{CC2}	V_{CC}=2V			100	nA	10
		V_{CC}=5V			100		

*Unless otherwise noted.

CAPACITANCE

(t_A = 25°C)

PARAMETER	SYMBOL	CONDITION	TYP	MAX	UNITS	NOTES
Input Capacitance	C_I		5		pF	
I/O Capacitance	$C_{I/O}$		10		pF	
Crystal Capacitance	C_X		6		pF	

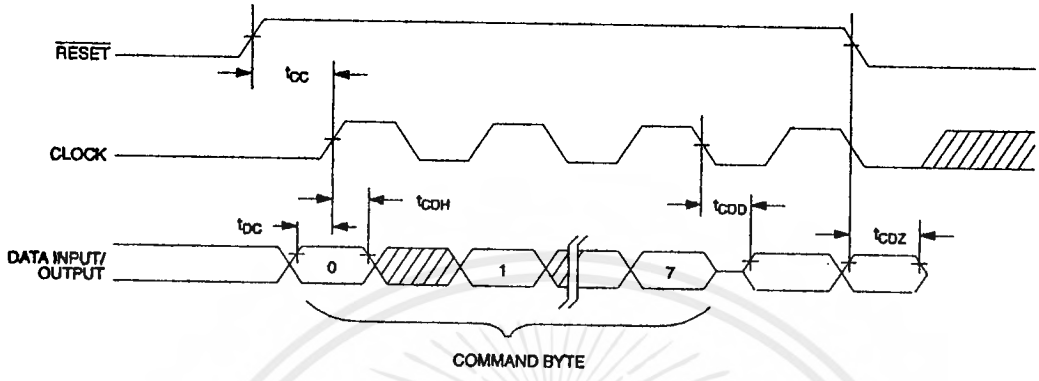
AC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

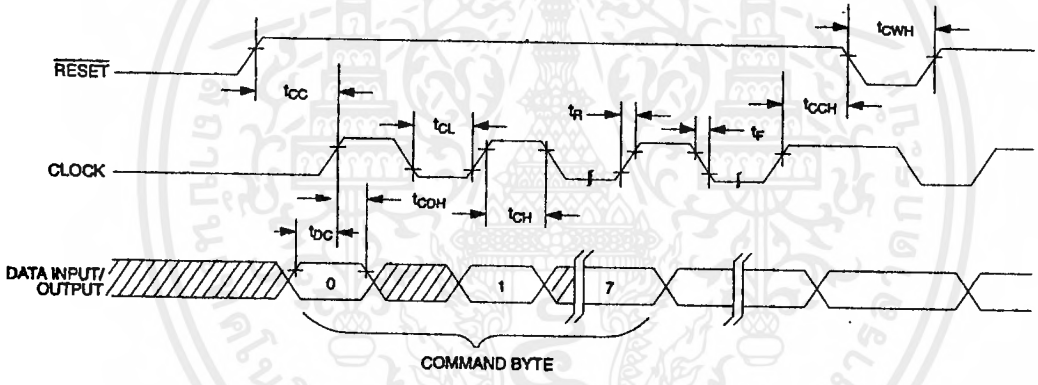
PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Data to CLK Setup	t_{DC}	V_{CC}=2V	200			ns	7
		V_{CC}=5V	50				
CLK to Data Hold	t_{CDH}	V_{CC}=2V	280			ns	7
		V_{CC}=5V	70				
CLK to Data Delay	t_{CDD}	V_{CC}=2V			800	ns	7, 8, 9
		V_{CC}=5V			200		
CLK Low Time	t_{CL}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
CLK High Time	t_{CH}	V_{CC}=2V	1000			ns	7, 12
		V_{CC}=5V	250				
CLK Frequency	f_{CLK}	V_{CC}=2V			0.5	MHz	7, 12
		V_{CC}=5V	DC		2.0		
CLK Rise and Fall	t_R, t_F	V_{CC}=2V			2000	ns	
		V_{CC}=5V			500		
$\overline{RST}$ to CLK Setup	t_{CC}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
CLK to $\overline{RST}$ Hold	t_{CCH}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
$\overline{RST}$ Inactive Time	t_{CWH}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
$\overline{RST}$ to I/O High Z	t_{CDZ}	V_{CC}=2V			280	ns	7
		V_{CC}=5V			70		

*Unless otherwise noted.

TIMING DIAGRAM: READ DATA TRANSFER Figure 5



TIMING DIAGRAM: WRITE DATA TRANSFER Figure 6



NOTES:

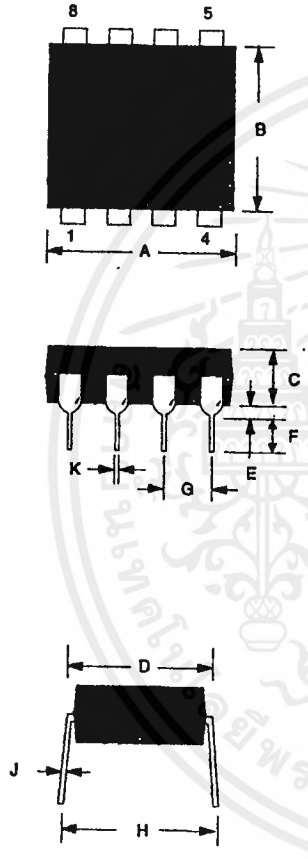
1. All voltages are referenced to ground.
2. Logic one voltages are specified at a source current of 1 mA at V_{CC}=5V and .4 mA at V_{CC}=2V, V_{OH}=V_{CC} for capacitive loads.
3. Logic zero voltages are specified at a sink current of 4 mA at V_{CC}=5V and 1.5 mA at V_{CC}=2V.
4. I_{CC1} is specified with I/O open, $\overline{RST}$ set to a logic 0, and clock halt flag=0 (oscillator enabled).
5. I_{CC} is specified with the I/O pin open, $\overline{RST}$ high, SCLK=2 MHz at V_{CC}=5V; SCLK=500 KHz, V_{CC}=2V and clock halt flag=0 (oscillator enabled).
6. $\overline{RST}$, SCLK, and I/O all have 40KΩ pulldown resistors to ground.
7. Measured at V_{IH}=2.0V or V_{IL}=0.8V and 10 ms maximum rise and fall time.
8. Measured at V_{OH}=2.4V or V_{OL}=0.4V.
9. Load capacitance = 50 pF.

10. I_{CC2} is specified with $\overline{RST}$, I/O, and SCLK open. The clock halt flag must be set to logic one (oscillator disabled).

11. At power–up, $\overline{RST}$ must be at a logic 0 until V_{CC} 2 volts. Also, SCLK must be at a logic 0 when $\overline{RST}$ is driven to a logic one state.

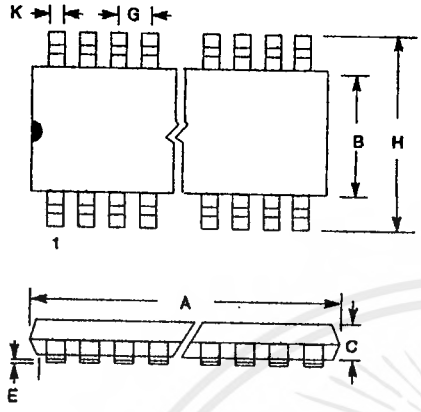
12. If t_{CH} exceeds 100 ms with $\overline{RST}$ in a logic one state, then I_{CC} may briefly exceed I_{CC} specification.

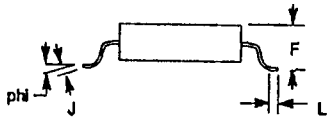
DS1202 SERIAL TIMEKEEPER 8–PIN DIP



PKG	8–PIN	
DIM	MIN	MAX
A IN. MM	.360	.400
B IN. MM	.240	.260
C IN. MM	.120	.140
D IN. MM	.300	.325
E IN. MM	.015	.040
F IN. MM	.110	.140
G IN. MM	.090	.110
H IN. MM	.320	.370
J IN. MM	.008	.012
K IN. MM	.015	.021

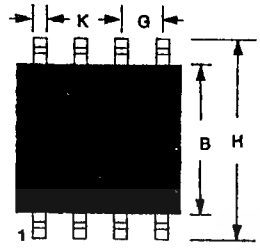
DS1202S SERIAL TIMEKEEPER 16–PIN SOIC

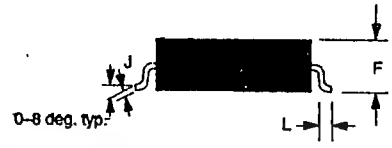


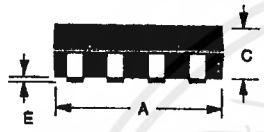


PKG	16-PIN	
DIM	MIN	MAX
A IN. MM	0.500 12.70	0.511 12.99
B IN. MM	0.290 .7.37	0.300 7.65
C IN. MM	0.089 2.26	0.095 2.41
E IN. MM	0.004 0.102	0.012 0.30
F IN. MM	0.094 2.38	0.105 2.68
G IN. MM	0.050 BSC 1.27 BSC	
H IN. MM	0.396 10.11	0.416 10.57
J IN. MM	0.009 0.229	0.013 0.33
K IN. MM	0.013 0.33	0.019 0.48
L IN MM	0.016 0.406	0.040 1.20
phi	0°	8°

DS1202S8 8–PIN SOIC 200 MIL



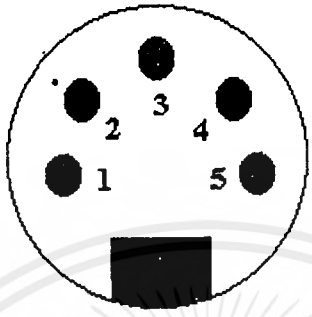




PKG	8-PIN	
DIM	MIN	MAX
A IN. MM	0.203 5.16	0.213 5.41
B IN. MM	0.203 5.16	0.213 5.41
C IN. MM	0.070 1.78	0.074 1.88
E IN. MM	0.004 0.102	0.010 0.390
F IN. MM	0.074 1.88	0.84 2.13
G IN. MM	0.050 BSC 1.27 BSC	
H IN. MM	0.302 7.67	0.318 8.07
J IN. MM	0.006 0.152	0.010 0.254
K IN. MM	0.013 0.33	0.020 0.508
L IN. MM	0.19 4.83	0.030 0.762

ภาคผนวก ค.

ข้อมูลจำเพาะของสายสัญญาณหัวอ่าน



รูปแสดงที่เสียบสายสัญญาณหัวอ่าน

ขา 1 ไม่ใช้

ขา 2 ไม่ใช้

ขา 3 ขากราวนด์

ขา 4 ขาสัญญาณ

ขา 5 ขาไฟเลี้ยง + 5 โวลท์