



ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ (พัฒนา)

DATA LOGGER FOR AIR POLLUTION (DEVELOPMENT)



โดย
นายเจริญหัตถ์ เลิศบรรณพันธุ์
นายสรารุท คำวิบูลย์
นางสาวอังคณา จารูวาระกุล

วัน เดือน ปี... ๑ ส.ค. ๒๕๕๐
เลขทะเบียน... ๐๓๗ ๑๒๖
เลขเรียกหนังสือ... T ๖๘๒๑๑ ๑.๗๖๖ ๔

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา ๒๕๕๘

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ (พัฒนา)
DATA LOGGER FOR AIR POLLUTION (DEVELOPMENT)



โดย
นายเจริญหัตถ์ เลิศบรรณพันธุ์ 36013101
นายสราวุธ คำวิบูลย์ 36013130
นางสาวอังคณา จารูวาระกุล 36013139
อาจารย์ที่ปรึกษา
ดร. กิติพล ชิตสกุล

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

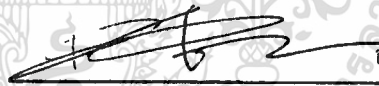
ปริญญานิพนธ์ ปีการศึกษา 2538

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ (พัฒนา)

ผู้จัดทำ

- | | |
|-----------------|---------------|
| 1. นายเจริญหัส | เลิศบรรณพันธ์ |
| 2. นายสรารุท | คำวิบูลย์ |
| 3. นางสาวอังคณา | จารุวาระกุล |



อาจารย์ที่ปรึกษา

(_____)

ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ (พัฒนา)

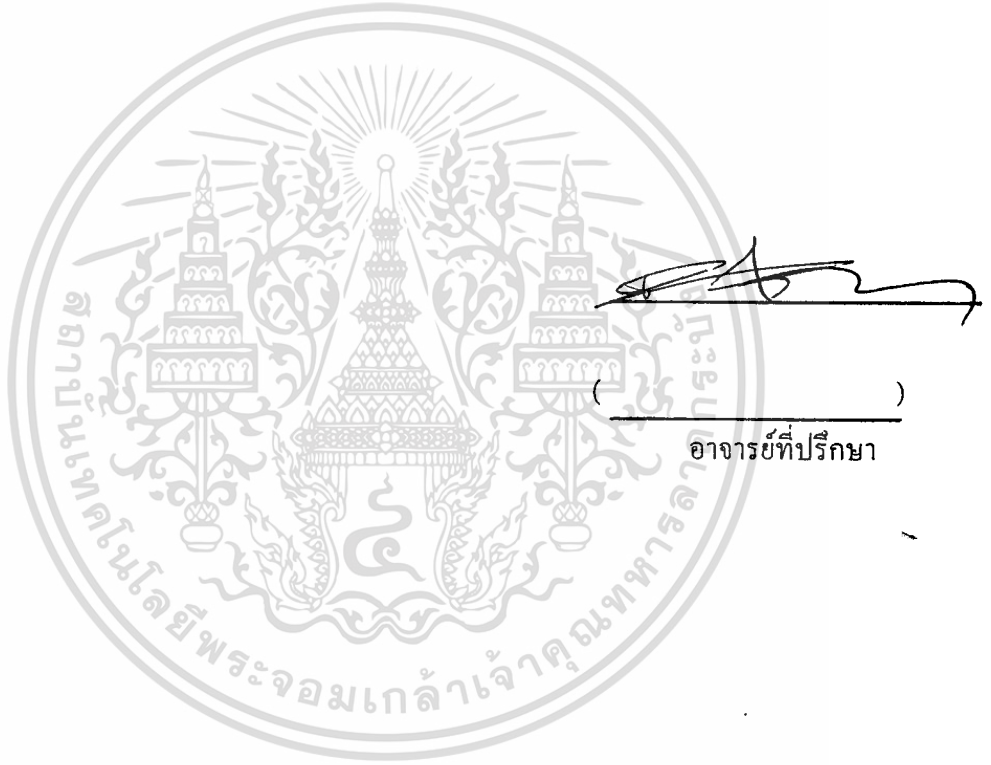
DATA LOGGER FOR AIR POLLUTION (DEVELOPMENT)

นายเจริญหัตถ์ เลิศบรรณพันธุ์

นายสรายุทธ คำวิบูลย์

นางสาวอังคณา จารูวาระกุล

โครงการนี้ได้รับการตรวจสอบแล้ว พร้อมทั้งจะทำการสอบได้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ (พัฒนา)

นายเจริญหัต	เลิศบรรณพันธุ์
นายสราวุธ	คำวิบูลย์
นางสาวอังคณา	จารูวระกุล
ดร. กิตติพล	ชิตสกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2538	

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เป็นโครงการที่พัฒนามาจากระบบเก็บข้อมูลมลพิษในอากาศซึ่งเป็นวิทยานิพนธ์ของปี 2537 โดยโครงการนี้ มีเนื้อหาหลักคือ เราจะเก็บค่าปริมาณก๊าซในอากาศเข้ามาทางหัวเซ็นเซอร์ก๊าซ และเปลี่ยนเป็นข้อมูลดิจิทัล 12 บิต เก็บข้อมูลลงในหน่วยความจำขนาด 32 กิโลไบต์ ซึ่งจะมีไมโครคอนโทรลเลอร์เบอร์ 8031 เป็นตัวควบคุมการทำงานของระบบทั้งหมด และจะมีการแสดงผลบนจอ LCD DISPLAY ในลักษณะของ REAL TIME CLOCK ในวิทยานิพนธ์นี้จะแสดงผลของ ปริมาณ Cigarette's smoke ซึ่งเป็นหนึ่งในก๊าซหลายชนิดที่ตัวเซ็นเซอร์ชนิดนี้วัดได้ และจากการทดลองโครงการนี้ จะสามารถนำข้อมูลแสดงออกไปที่ไมโครคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม (RS232C) โดยจะแสดงผลเป็นค่าของปริมาณก๊าซที่เปลี่ยนไปตามค่าเวลาในลักษณะของกราฟเชิงเส้น ซึ่งจะสามารถไปประยุกต์ในงานเก็บข้อมูลทางสถิติได้ต่อไป

DATA LOGGER FOR AIR POLLUTION (DEVELOPMENT)

Mr. Jaroenhas Lertbanpant

Mr. Sarawut Kumwiboon

Miss. Aungkana Jaruvarakul

Dr. Kitipol Chitsakul

(advisory instructor)

educational year 1995

Abstract

We develop a data logger for air pollution. Based on the micro-controller 8031 and air quality sensor, the machine collect the data and stores in memory. The internal real time clock is also integrated into the machine for indication the logging time on LCD display. The data in 32 Kbytes can be then transfered to a PC via a serial port for display and analysis. In order to test the system, we used the smoke of cigarettes as a pollution. The results of our experiment are shown.

สารบัญ

หน้า

บทคัดย่อ	I
ABSTRACT	II
สารบัญ	III
สารบัญรูป	V
สารบัญตาราง	VII

บทที่ 1 บทนำ	1
บทที่ 2 ระบบตรวจจับก๊าซ	3
2.1 คุณสมบัติหัวตรวจจับก๊าซ NAP-11AS	3
2.2 วงจรสำหรับทดสอบของ NAP-11AS	5
บทที่ 3 การแปลงสัญญาณอะนาล็อกเป็นดิจิตอล	8
3.1 หลักการของ A/D แบบ dual slope	8
3.2 หลักการทำงานของ การเปลี่ยนแปลงอะนาล็อกเป็นดิจิตอล (A/D) ด้วยไอซีเบอร์ 7109	10
บทที่ 4 ไมโครคอนโทรลเลอร์ 8031 และชุด Real time Clock	13
4.1 คุณสมบัติและการใช้งานไมโครคอนโทรลเลอร์ 8031	13
4.2 Real time Clock (RTC) # DS 1202	17
บทที่ 5 บอร์ดสื่อสารอนุกรม	24
5.1 โครงสร้างของพอร์ตสื่อสาร	25
5.2 การใช้งานรีจิสเตอร์ต่าง ๆ บน 8250	27

สารบัญ

	หน้า
บทที่ 6 การออกแบบและผลการทดลอง	36
6.1 แนวทางการออกแบบ	36
6.1.1 การออกแบบชุดไมโครคอนโทรลเลอร์	36
6.1.2 การออกแบบชุดก๊าซเซ็นเซอร์	38
6.1.3 การออกแบบชุดการแปลงสัญญาณอะนาล็อกเป็นดิจิทัล	44
6.2 ผลการทดลอง A/D # 7109	46
6.3 ผลการทดลองชุดก๊าซเซ็นเซอร์ (Gas Sensor)	50
6.4 ผลการทดลองชุด Controller และ RTC	52
6.5 ผลการทดลองเชื่อมต่อไมโครคอมพิวเตอร์	62
6.6 ผลการทดลองระบบโดยรวม	76
บทที่ 7 สรุปผลการทดลองและข้อเสนอแนะ	84
ภาคผนวก	86
ก โปรแกรมการใช้งาน	122
ข LCD - Display	125
ค มาตรฐานการเชื่อมต่อ RS232	127
ง Data Sheets ICs	
กิจกรรมประกาศ	139
เอกสารอ้างอิง	140

สารบัญรูป

รูปที่

หน้า

1.1	แสดง Block Diagram ของโครงการ	1
2.1	โครงสร้าง NAP-11AS	6
2.2	วงจรทดสอบ NAP-11AS	6
2.3	Response Characteristic	6
3.1	แสดงวงจรของ A/D แบบ Dual-Slop	8
3.2	แสดงรูปคลื่นสัปดาห์เข้าที่พหุของ Intergrater	9
3.3	แสดง Timing Diagram ของสัญญาณต่างๆระหว่างการ Conversion	11
3.4	แสดง Conversion timing ของ ICL 7109 A/D Convertor	11
4.1	แสดงการใช้หน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิพ	14
4.2	แสดงการใช้โปรแกรมจากหน่วยความจำสำหรับเก็บในโปรแกรมภายในชิพ	14
4.3.A	แสดงถึงแผนที่ของหน่วยความจำข้อมูลภายในและ SFR	15
4.3.B	แสดงถึงแผนที่การกำหนดตำแหน่งบิต	15
4.4	ตำแหน่งของรีจิสเตอร์ SFR และบิต Address ของ SFR	16
4.5	โครงสร้างภายในของ RTC DS/202	19
4.6	แสดงโครงสร้างของ Command Byte	20
4.7	แสดงข้อมูลการทำงานของรีจิสเตอร์และ Ram ใน DS/202	22
4.8	ชิพ RTC DS/202 ต่อเข้าพอร์ท 1 ของ 8031	23
5.1	โครงสร้างพอร์ทสื่อสารข้อมูลที่ใช้ 8250	25
5.2	รูปแบบข้อมูล 1 เฟรม	27
5.3	ค่าของบิตในรีจิสเตอร์	28
5.4	ค่าของบิตในรีจิสเตอร์แสดงสถานะสายสื่อสาร	30
5.5	ค่าของบิตในรีจิสเตอร์กำหนดอินเตอร์รัพต์	30
5.6	ค่าของบิตในรีจิสเตอร์อินาเบลอินเตอร์รัพต์	32
5.7	ค่าของบิตในรีจิสเตอร์ควบคุมโมเดม	33
5.8	ค่าของบิตในรีจิสเตอร์แสดงสถานะโมเดม	34
6.1	8031 System Decode and Memory	39
6.2	Flowchart ของชุดแสดงผล A/D	40

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3	Flowchart ของชุดแสดงผล RTC	41
6.4	วงจร Gas Sensor Circuit	42
6.5	วงจร A/D Converter 7109	43
6.6	วงจรทดสอบก๊าซเซนเซอร์	51
6.7(ก)	โฟลชาทส่วนที่หนึ่งของโปรแกรมชุด RTC	53
6.7(ข)	โฟลชาทส่วนที่สองของโปรแกรมชุด RTC	54
6.7(ค)	โฟลชาทส่วนที่สามของโปรแกรมชุด RTC	55
6.7(ง)	โฟลชาทส่วนที่สี่ของโปรแกรมชุด RTC	56
6.7(จ)	โฟลชาทส่วนที่ห้าของโปรแกรมชุด RTC	57
6.8(ก)	โฟลชาทแสดงการรับส่งข้อมูลระหว่างชุด 8031 กับไมโครคอมพิวเตอร์	64
6.8(ข)	โฟลชาทแสดงการรับส่งข้อมูลระหว่างชุด 8031 กับไมโครคอมพิวเตอร์	65
6.9(ก)	แสดงกราฟการตรวจวัดข้อมูลจริงแบบใช้พัดลมดูดอากาศชุด 1 และ 2	79
6.9(ข)	แสดงกราฟการตรวจวัดข้อมูลจริงแบบใช้พัดลมดูดอากาศชุด 3 และ 4	80
6.10(ก)	แสดงกราฟการตรวจวัดข้อมูลจริงแบบไม่ใช้พัดลมดูดอากาศชุด 1 และ 2	81
6.10(ข)	แสดงกราฟการตรวจวัดข้อมูลจริงแบบไม่ใช้พัดลมดูดอากาศชุด 3 และ 4	82



สารบัญตาราง

ตารางที่		หน้า
2.1	NAP-11AS Response Characteristic	3
2.2	แสดงค่าของ Gas Sensitive Characteristic	4
2.3	แสดงค่าอุณหภูมิที่มีผลต่อความต้านทาน	5
2.4	แสดงค่าความชื้นที่มีผลต่อความต้านทาน	5
2.5	แสดงส่วนประกอบของคว้นบุหรี	7
5.1	หมายเลขอินพุตเข้าที่พอร์ทของ Com1 และ Com2	26
5.2	การกำหนดแอดเดรสสำหรับหมายเลขพอร์ทต่างๆ	26
5.3	ค่าตัวหารสำหรับการกำหนดอัตราบอด	29
5.4	ฟังก์ชันการอินเตอร์รัพต์รหัสอินเตอร์รัพต์	32
6.1	ผลการทดลองวงจรแปลงอนาลอกเป็นดิจิตอล	47
6.2	แผนผังโดยรวมของชุดโปรแกรมภาษาซี	63

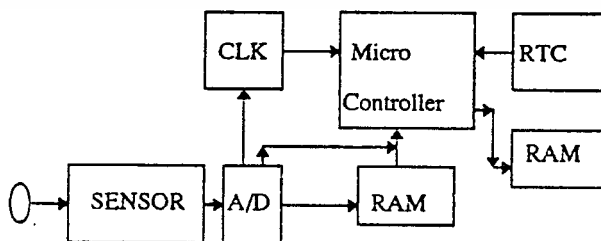


บทที่ 1

บทนำ

สืบเนื่องมาจากปริญญานิพนธ์ เรื่อง "ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ" (Data logger for air pollution) ของปีการศึกษา 2537 ได้กล่าวเกี่ยวกับการนำเอาไมโครคอนโทรลเลอร์ 8031 ไปใช้งานร่วมกับวงจรตรวจจับก๊าซคาร์บอนไดออกไซด์ และวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล(A/D)เบอร์ HI5810 ซึ่งเป็นไอซีขนาด 12 bit ซึ่งจากผลที่ได้นั้นยังไม่เหมาะสมที่จะนำไปใช้ได้จริง ดังนั้นทางผู้จัดทำโครงการนี้จึงมีความคิดที่จะพัฒนาคุณสมบัติต่างๆของโครงการเดิมให้ดีขึ้น เพื่อที่จะนำไปใช้งานทางภาคปฏิบัติการได้สะดวกยิ่งขึ้น โดยมีแนวจุดประสงค์ดังนี้

1. สามารถที่จะนำข้อมูลมาแสดงที่จอ LCD ได้ทันทีในขณะที่ทำการตรวจวัดข้อมูล (Real Time Control)
2. สามารถที่จะเชื่อมต่อข้อมูลจากบรอกคอนโทรลเลอร์กับไมโครคอมพิวเตอร์ เพื่อที่จะสามารถแสดงค่าของข้อมูลในรูปลักษณะที่เป็นกราฟเชิงเส้นได้เพื่อเป็นประโยชน์ในการวิเคราะห์ทางสถิติ
3. สามารถที่จะลดกำลังไฟในการใช้งานได้ดีขึ้น ในขณะที่สามารถเก็บข้อมูลได้มากขึ้น
4. มีรูปร่างของโครงการที่มีขนาดเล็กพอเหมาะกับการเคลื่อนย้ายไปในการเก็บข้อมูลได้สะดวก ซึ่งเราจะสามารถแสดงการทำงานโดยรวมได้ดัง Block diagram ต่อไปนี้



รูปที่ 1.1 แสดง Block diagram ของโครงการ

จากบล็อกไดอะแกรมโครงสร้างฮาร์ดแวร์ที่พัฒนาจากโครงงานเดิม สามารถแยกอธิบายรายละเอียดของภาคต่าง ๆ ได้ดังนี้

การแปลงสัญญาณอะนาล็อกเป็นสัญญาณดิจิทัล

จากโครงงานเดิม ใช้ไอซีเบอร์ HI 5810 ซึ่งทำหน้าที่แปลงสัญญาณอะนาล็อกไปเป็นสัญญาณดิจิทัลแบบขนาน 12 บิต ข้อดีของไอซีเบอร์นี้ คือ ทำงานที่ความเร็วสูงประมาณ 10 ไมโครวินาที และกินกำลังงานและไฟเลี้ยงต่ำ ซึ่งสูญเสียกำลังงานในขณะที่ทำงานเพียง 0.810-0.870 วัตต์ และไฟเลี้ยงใช้งานในช่วง 3-6 โวลต์ ทำให้สามารถใช้จากแบตเตอรี่ได้ แต่มีข้อเสียคือ หาซื้อได้ยากในตลาดและราคาค่อนข้างสูง ดังนั้นเราจึงเปลี่ยนมาใช้ไอซีเบอร์ 7109 แทน ซึ่งสามารถซื้อได้ง่ายในท้องตลาด และราคาก็ค่อนข้างถูกด้วย ทำให้ลดต้นทุนลงไปด้วย

ไมโครคอนโทรลเลอร์และหน่วยความจำสำรอง (Microcontroller and RAM)

จากโครงงานเดิมให้ CPU-8031, 74 HC 373, 6264 X3, RAMPACK(2764), วงจร Reset เราพัฒนาในส่วนของ RAM 6264 X3 ซึ่งสามารถเก็บข้อมูลได้ 24 K ไบต์ มาเป็นเบอร์ 62256 เพียงตัวเดียวสามารถเก็บข้อมูลเพิ่มได้เป็น 32 K ไบต์ ส่วนต่าง ๆ ยังคงเหมือนเดิม

ก๊าซเซ็นเซอร์ (Gas Sensor)

ในโครงงานนี้จะใช้ตัวจับก๊าซเบอร์ NAP-11AS ซึ่งมีคุณสมบัติสามารถตรวจจับก๊าซได้หลายชนิดขึ้นอยู่กับทางเลือกใช้ที่ต้องการซึ่งจะมีรายละเอียดอยู่ในบทที่ 2 ของวิทยานิพนธ์นี้

ชุดการแสดงผลเวลาจริง (Real time clock)

ใช้ชิปไอซีสำเร็จรูป เบอร์ DS1202 “Serial Timekeeper Chip” ใช้เพียง 3 เส้นในการ Interface กับ CPU 8031 คือ RST (reset), I/O (data line), SCLK (serial clock) รายละเอียดต่าง ๆ อยู่ในบทที่ 4

ชุด แอลซีดี (Liquid Control Display ; LCD)

ใช้ Dot Matrix LCD module 20 ตัวอักษร 2 บรรทัด ในการแสดงผล ข้อดีมีน้ำหนักเบา กินกระแสไฟน้อยมาก และทำงานด้วยไฟเลี้ยงในระดับ 5V รายละเอียดต่าง ๆ อยู่ในภาคผนวก

บทที่2

ระบบตรวจจับก๊าซ

2.1 คุณสมบัติของ Gas Sensor เบอร์ NAP-11AS (Indoor Odor Sensor

ตัวจับก๊าซNAP-11AS นี้เป็นเซมิคอนดักเตอร์ Gas Sensor. ก่อนข้างไวในการตรวจจับ ซึ่งจะสามารถตรวจจับก๊าซและกลิ่นที่เกิดขึ้นในชีวิตประจำวันได้ เช่น กลิ่นของควันบุหรี่ กลิ่นของบูดเน่าและกลิ่นต่าง ๆอีกมาก ซึ่งขึ้นอยู่กับ การเลือกใช้งาน ดังแสดงคุณสมบัติในการตอบสนองกลิ่นต่างๆ ดังตารางที่ 2.1

Objects	Olfaction level (Sensuousness)	NAP-11AS Response		Remarks
		Sensitivity	Speed	
1) Cigarette smoke	Strong	Hign	Quick	5 pieces
2) Cosmetics	Strong	Very high	Quick	Cologne 1 puff
3) Insecticides	Strong	Very high	Gentle	Sprayed for 10 sec
4) Fried meat	Med./Strong	High	Quick	300 gms.
5) Fried onion	Strong	High	Quick	3 pcs.
6) Fried green pepper	Medium	Medium	Slow	5 pcs.
7) Coffee	Weak	Low	-	5 cups

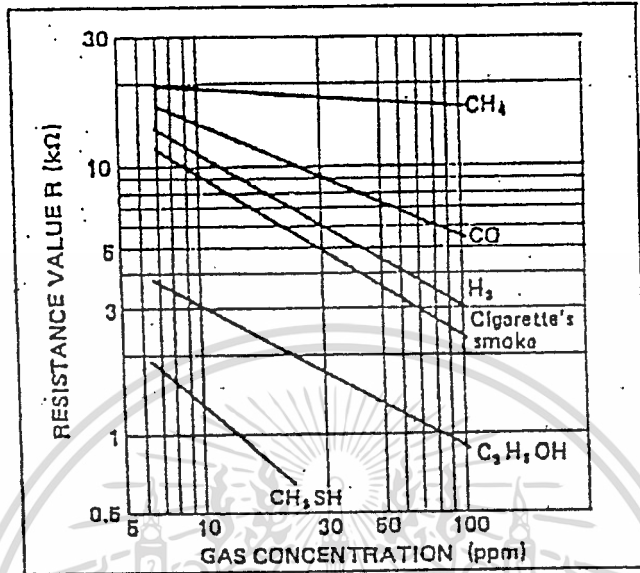
ตารางที่ 2.1 NAP-11AS Response Characteristic

คุณลักษณะของหัวจับก๊าซ

- มีค่าความไวสูง (Sensitive) สูง ในการตรวจจับก๊าซที่มีความเข้มข้นต่ำ(ประมาณ 100 ppm)
- การออกแบบวงจรในการตรวจจับทำได้ง่าย
- Very Stable to ambient humidity

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sensitivity characteristics



ตารางที่ 2.2 แสดงค่าของ Gas Sensitivity Characteristic โครงสร้างพื้นฐานของหัวก๊าซ เซนเซอร์

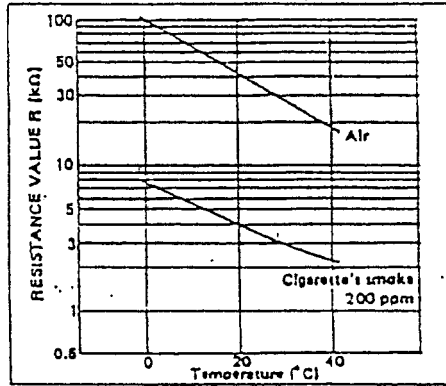
จากรูปที่ 2.2 เป็นการแสดงค่าความต้านทานที่แปรผันไปตามค่าความเข้มข้นของก๊าซ โดยจะคิดความเข้มข้นในหน่วยของหนึ่งในล้านส่วน

คุณสมบัติของหัวเซ็นเซอร์ต่ออุณหภูมิ

หัวตรวจก๊าซ NAP-11AS มีคุณสมบัติเหมือนเป็น เทอร์มิสเตอร์ชนิด NTC ซึ่งจะมีค่าของความต้านทานแปรผกผันกับอุณหภูมิที่เกิดขึ้น ซึ่งสามารถแสดงได้ดังตารางรูปที่ 2.3 ซึ่งในการออกแบบวงจรเซ็นเซอร์จำเป็นต้องคำนึงถึงจุดนี้สำคัญ

Ambient temperature dependency

Relative humidity : 50%



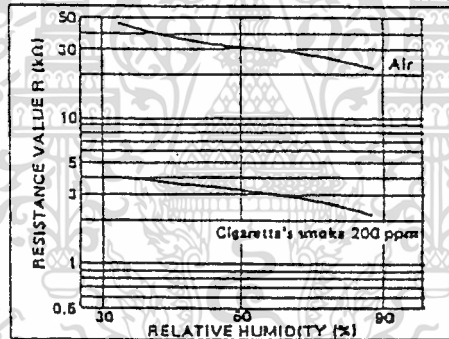
ตารางที่ 2.3 แสดงค่าอุณหภูมิที่มีผลต่อความต้านทาน

คุณสมบัติของหัวก๊าซเซนเซอร์ต่อค่าความชื้น

หัวก๊าซเซนเซอร์ตัวนี้ จะมีข้อดีตรงที่มีค่าของความต้านทานของตัวเซนเซอร์เอง เปลี่ยนไปตามค่าความชื้นน้อยมาก โดยจะแสดงได้จากตาราง 2.4

Ambient humidity dependency

Temperature : 25°C



ตารางที่ 2.4 แสดงค่าความชื้นที่มีผลต่อความต้านทาน

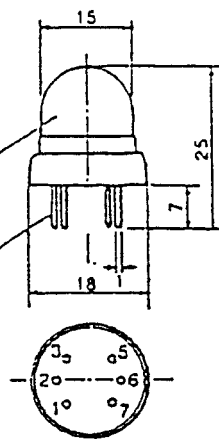
2.2 วงจรสำหรับทดสอบของ NAP-11AS

จากรูปที่ 2.2 เป็นลักษณะการต่อวงจรในการทดสอบการใช้งานของหัวก๊าซเซนเซอร์และในรูปที่ 2.3 เป็นค่าของกราฟการตอบสนองในการเปลี่ยนแปลงแรงดันที่ Output โดยใช้อากาศบริสุทธิ์กับคว้นของนุหรีเป็นตัวเปรียบเทียบไปกับเวลา จะสังเกตได้ว่าที่ช่วงเวลาขณะที่ยังไม่มีการตรวจจับคว้นนุหรีได้จะมีค่าแรงดันประมาณ 2.5 Volts แต่เมื่อมีคว้นนุหรี (เปรียบเทียบที่ความเข้มข้น 200 ppm) เข้ามาจะวัดแรงดัน Output ได้ประมาณ 6 Volts

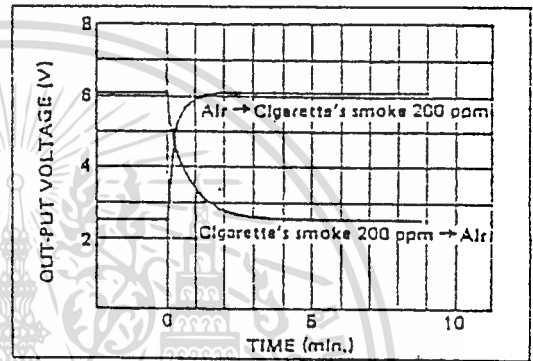
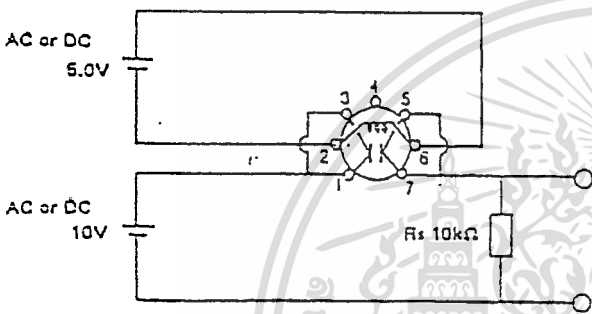
CONFIGURATION OF NAP-11A SERIES

STRAINER (SUS)

NICKEL LEAD PIN



รูปที่ 2.1 โครงสร้าง NAP-11AS



รูปที่ 2.2 วงจรทดสอบ NAP-11AS

รูปที่ 2.3 Response Characteristic

จากรูป 2.2 เป็นวงจรที่ใช้ทดสอบซึ่งเราจำเป็นต้องทราบค่าคุณสมบัติในการนำไปใช้งาน ดังต่อไปนี้

- 1) ค่าแรงดันที่ใช้ในการอุ่นตัวก๊าซ
 - ไฟ ดีซี (D.C) เท่ากับ $5 \pm$ โวลต์
 - ไฟ เอซี (A.C) เท่ากับ 5 ± 0.5 โวลต์
- 2) ค่ากระแสที่ใช้ในการอุ่นตัวก๊าซ
 - ไฟ ดีซี (D.C) เท่ากับ 160 ~ 180 มิลลิแอมป์
 - ไฟ เอซี (A.C) เท่ากับ 160 ~ 180 มิลลิแอมป์
- 3) ค่าแรงดันที่ใช้ในขณะที่ทำการวัด
 - ไฟ ดีซี และ เอซี จะมีค่าได้ไม่เกิน 10 โวลต์ $\pm 10\%$

และจากตารางที่ 2.5 เป็นการแสดงส่วนประกอบของควันบุหรี่ (Cigarette smoke) โดยเปรียบเทียบออกมาในรูปของความไวในการตรวจวัด

Reference data (1) Sensitivity to the ingredients of putrid smells

Putrid smells		Concentration in test	NAP-11AS Sensitivity (Ra/Rg)	Remarks (Smells)
Ingredients	Formula			
Methyl mercaptan	CH ₃ SH	1.0 ppm.	13.5	Putrid Vegetables
Hydrogen sulfide	H ₂ S	1.0	8.5	Putrefactive
Trimethylamine	(CH ₃) ₃ N	1.0	11.3	fermentation
Scatol	CoHoN	1.0	15.5	Fish
Indol	C ₈ H ₇ N	1.0	16.3	Excreta Feces

Reference data (2) Sensitivity to the ingredients of a cigarette smoke

ตารางที่ 2.5 แสดงส่วนประกอบของควันบุหรี่

Putrid smells		Concentration in test	NAP-11AS Sensitivity (Ra/Rg)	Test Condition
Ingredients	Formula			
A cigarette is burnt in 30 m ³			8.0	
Carbon monoxide	CO	1.0 ppm.	1.8	10 ppm.
Acetoaldehyde	CH ₃ CHO	0.5	4.5	10 ppm.
Ammonia	NH ₃	0.3	2.5	10 ppm.
Formaldehyde	HCHO	0.2	3.2	10 ppm.
Acetone	(CH ₃) ₂ CO	0.07	10.5	10 ppm.
Acetic acid	CH ₃ COOH	0.07	6.8	10 ppm.
Nicotine	C ₁₀ H ₁₄ N ₂	0.07	18.0	10 ppm.
Toluene	C ₆ H ₅ CH ₃	0.05	8.3	10 ppm.
Pyriding	C ₅ H ₅ N	0.02	9.3	10 ppm.

* Ra/Rg = (Resistance value in clean air) / (Resistance value in gas)

บทที่ 3

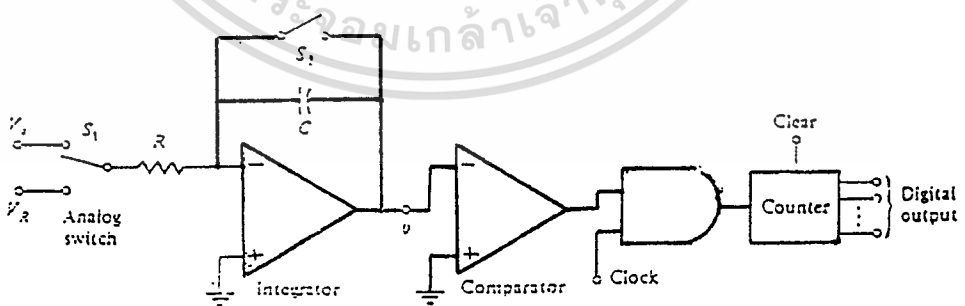
การแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิทัล

ในส่วนของการแปลงสัญญาณอะนาลอกเป็นดิจิทัลได้เลือกใช้ ไอซีเบอร์ ICL7109ซึ่งเป็น A/D Converter ขนาด 12 บิต แบบ Dual slop ซึ่งมีคุณสมบัติที่ดีดังนี้

- มีสูญเสียกำลังงานต่ำเพราะใช้เทคโนโลยีของ LSI-CMOS chip
- กินกระแสต่ำมาก ประมาณ 10 pA max ที่อินพุท
- มีการแปลงสัญญาณแบบ Dual - Slop ทำให้ใช้งานได้สะดวก
- มีการป้องกันสนามไฟฟ้าสถิตที่อินพุท
- สามารถใช้ในการส่งสัญญาณได้ทั้งแบบอนุกรมและแบบขนาน
- มีราคาถูก (หาคู่มือศึกษาได้ง่าย)

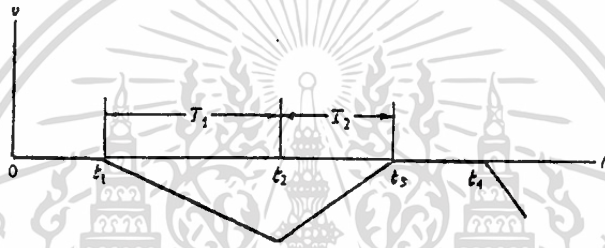
3.1 หลักการของ A/D (Analog-to-digital converse)แบบ dual slop

วงจร A/D แบบ dual- slop เป็นวงจรที่ได้รับความนิยมมากเพราะใช้เทคนิคของ การอินทิเกรตสองครั้งเพื่อให้ความแม่นยำในการ conversion ที่ไม่ขึ้นกับความถี่ของสัญญาณ นาฬิกาวงจร A/D ชนิดนี้ใช้หลักการ conversion โดยอ้อม เช่น เปลี่ยนแรงดันให้กลายเป็นความกว้างของพัลส์ ใช้พัลส์นี้เปิดเกตให้สัญญาณนาฬิกาผ่านแล้วนับจำนวนสัญญาณนาฬิกาที่ผ่านได้ในช่วงที่เปิดเกตนั่น จะได้เป็นสัญญาณดิจิทัลที่ขึ้นอยู่กับแรงดันนั่นเอง



รูปที่ 3.1 แสดงวงจรของ A/D แบบ dual-slop

จากรูปที่ 3.1 ในภาวะตั้งต้น สวิตช์ S2 ปิดวงจร และ วงจรนับ (counter) ถูก clear เมื่อเวลา $t=t_1$ สวิตช์ S1 ต่อแรงดัน V_a เข้ากับอินทิเกรเตอร์ (integrator) และสวิตช์ S2 เปิดวงจร อินทิเกรเตอร์จะทำการอินทิเกรตสัญญาณอะนาล็อก V_a (สมมติค่าเป็นบวก) ในขณะเดียวกันวงจรนับจะทำการนับสัญญาณนาฬิกาไปเรื่อยๆ เมื่อถึงเวลา t_2 วงจรนับจะนับมาถึงค่าเต็มสเกลและจะรีเซ็ต (reset) ตัวเองกลับไปตั้งต้นใหม่พร้อมกับสวิตช์ S_1 ให้วงจรอินทิเกรเตอร์ทำการอินทิเกรตแรงดันอ้างอิง V_{ref} (ขั้วตรงข้ามกับ V_a) เพื่อให้วงจรคอมพาราเตอร์ (comparator) เปิดเกตให้สัญญาณพัลส์ผ่านไปยังวงจรนับ เมื่อแรงดันตกลงสู่ศูนย์ที่เวลา T_2 จะเป็นสัดส่วนกับแรงดันอะนาล็อก V_a และ V_{ref} แต่ไม่ขึ้นกับความถี่ของสัญญาณนาฬิกา



รูปที่ 3.2 แสดงรูปคลื่นตกค้างเอาต์พุตของ integrator ในรูปที่ 3.1

พิจารณารูปคลื่นของแรงดันของอินทิเกรเตอร์ จะแสดงได้ว่าช่วงเวลา T_1 จะคงที่และความชันในช่วง T_2 จะคงที่ 11 และจะต้องกำหนดให้ $|V_{ref}| < |V_a|$ เพื่อว่าช่วงเวลา $T_2 < T_1$ มิฉะนั้นการนับพัลส์ในช่วง T_2 ล้นสเกล

ค่าของแรงดันที่อินทิเกรเตอร์ที่เวลา t_3 เท่ากับ

$$v = -(1/RC) \int V_a dt - (1/RC) \int V_{ref} dt = 0$$

ค่า V_a และ V_{ref} เป็นค่าคงที่

$$V_a(t_2 - t_1) + V_{ref}(t_3 - t_1) = 0$$

$$V_a = |V_{ref}| T_2 / T_1$$

จำนวนพัลส์ในช่วง T_2 คือ n_2 ดังนั้น $T_2 = (n_2)T$ และ $T_1 = (n_1)T = (2^N)T$

$$V_a = |V_{ref}| T_2 / T_1 = n_2 |V_{ref}| / n_1 = n_2 |V_{ref}| / 2^N$$

$$V_a = |Verf| T_2 / T_1 = n_2 |Verf| / n_1 = n_2 |Verf| / 2^N$$

เพราะว่า Verf และ N คงที่ ดังนั้น ค่า V_a จึงเป็นสัดส่วนกับ n_2 สังเกตว่าผลอันนี้ไม่ขึ้นกับค่า time constant RC

ลักษณะของ A/D แบบ dual- slop มีดังต่อไปนี้

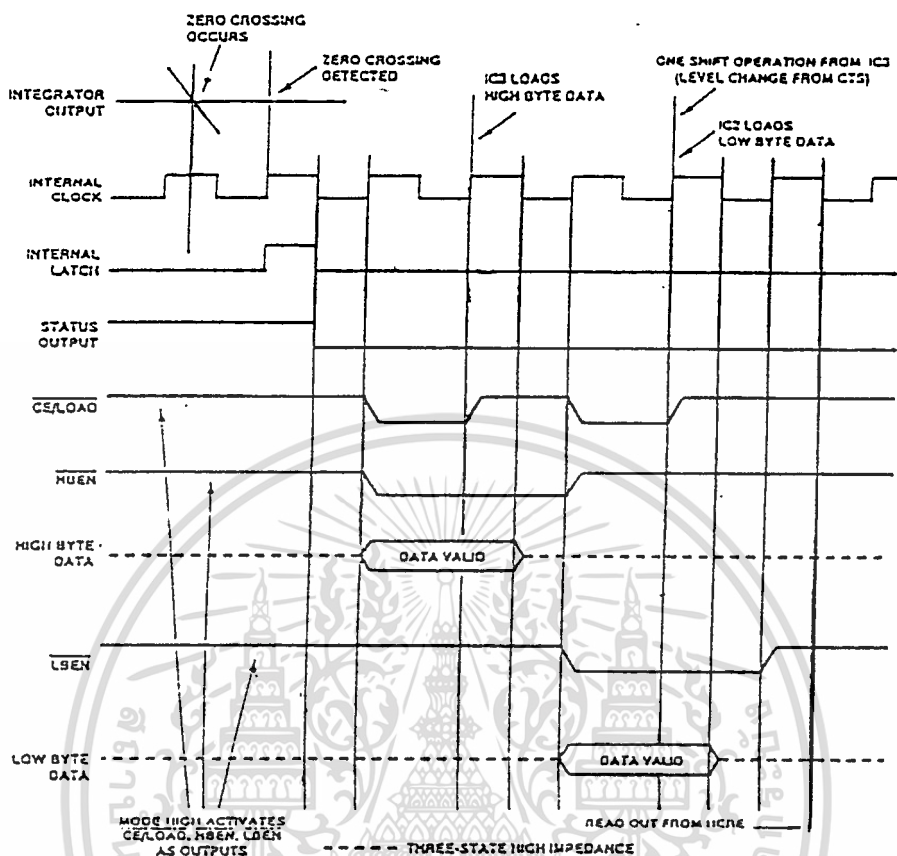
1. ให้ความถูกต้องในการ conversion สูง ไม่ขึ้นกับอัตราเร็วของสัญญาณนาฬิกาและค่าของตัวเก็บประจุ
2. สัญญาณรบกวนต่ำ
3. อัตราเร็วช้า
4. ตอบสนองต่อค่าระดับอินพุตเฉลี่ยระหว่างช่วงเวลาซาร์จ

ดังนั้น Dual-slop A/D converters จึงถูกใช้ในระบบที่ต้องการความถูกต้องแม่นยำ (accuracy) เป็นสำคัญมากกว่าเรื่องความเร็ว

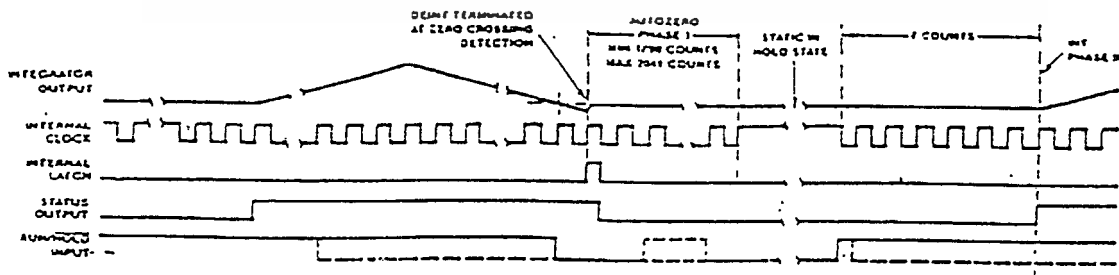
3.2 หลักการทำงานของ A/D เบอร์ 7109

เราจะศึกษาโดยเริ่มดูจาก Timing Diagram ในรูปที่ 3.3 และรูปที่ 3.4 ในรูปที่ 3.3 สัญญาณแรก เป็นสัญญาณที่วัดได้จากขาตัวเก็บประจุ อินทิเกรเตอร์ (Cint) และถัดลงมาเป็นสัญญาณอื่นๆ ของ 7109 ได้แก่ สัญญาณนาฬิกาที่ออกมาทางขาเอาต์พุต STATUS และสัญญาณอินพุต RUN/HOLD การที่ทำให้สัญญาณที่ขา RUN เป็น "1" จะเป็นการเริ่มขบวนการ Conversion

ขณะที่ขาเอาต์พุต STATUS เป็น "0" ข้อมูลใน A/D จะไม่ถูก "update " และในช่วงของการ Conversion ขาเอาต์พุต STATUS จะเป็น "1" ตลอด โดยในขั้นตอนแรกของการ Conversion จะประกอบด้วยซาร์จตัวเก็บประจุของอินทิเกรเตอร์ ด้วยกระแสที่มีขนาดเป็นสัดส่วนกับสัญญาณแรงดันอินพุต $(U_{HI} - U_{LO})/R$ (int) โดยใช้ช่วงเวลา 2048 ลูกของสัญญาณนาฬิกา ช่วงเวลา Integration time เท่ากับ $T = (2048 \text{ clock periods}) \times 58/(\text{ความถี่ของออสซิลเลเตอร์})$ ก่อนที่จะมีการ discharge ออกด้วยกระแสที่ (U_{ref}/R_{int}) การ discharge เราจะวัดเวลาด้วยการนับจำนวนพัลส์ของสัญญาณนาฬิกาภายใน



รูปที่ 3.3 แสดง Timing Diagram ของขาสัญญาณต่างๆ ระหว่างการ Conversion



รูปที่ 3.4 แสดง Conversion Timing ของ ICL 7109 A/D Converter

พัลส์ขาขึ้นของสัญญาณพิกาลูกแรกที่เกิดขึ้นหลังจากแรงดันที่ขา Cint ตกลงผ่านศูนย์ (zero - crossing) จะทำให้เกิดการโอนย้ายข้อมูลจากเกาท์เดอริไปยังรีจิสเตอร์เอาท์พุท เมื่อขาเอาท์พุท STATUS เปลี่ยนเป็นศูนย์แสดงว่าการ Conversion ล้นสุดลงแล้ว ซึ่งเราจะได้ว่า

ข้อมูลในเกาท์เดอริ = 2048 Uin/Uref

โดย Uin คือแรงดันอินพุท

Uref คือแรงดันอ้างอิง

ถ้าหากขาอินพุท RUN ค้างอยู่ที่ระดับลอจิก "1" ตลอดช่วงเฟสของการ Conversion ครึ่งหลัง การ Conversion จะเกิดต่อไปอย่างต่อเนื่องหลังจากผ่านเฟสของ Autozero หากไม่แล้วการเปลี่ยนระดับลอจิกของขา RUN อีกครั้งจะต้องคอยให้สัญญาณพิกาลูกผ่านไป 7 ลูกก่อนเสมอ ดังแสดงในรูปที่ 3.4 ซึ่งแสดงถึงสัญญาณ Handshaking ที่ ICL 7109 เมื่อติดต่อกับอุปกรณ์ภายนอก

จากวงจรขาอินพุท MODE และ SEND จะมีลอจิกเป็น "1" เมื่อขา MODE เป็น "high" ICL7109 จะเข้าสู่โหมด handshake mode ในรูปที่ 3.3 จะพบว่ามีสัญญาณอื่น ๆ ได้แก่ LOAD, HBEN, LBEN และ DATA ในขณะที่ขา MODE ยังอยู่ในสถานะ "high" ขา LOAD, LBEN, HBEN จะเป็นขาเอาท์พุท เมื่อขา HBEN มีลอจิกเป็น "0" แสดงว่าขณะนั้นกำลังส่งข้อมูลไบต์สูงจากเกาท์เดอริรวมทั้งข้อมูลสถานะ OVERRUN และขั้ว POLARITY และจะเริ่มเป็นผลเมื่อสัญญาณขาขึ้นของ LOAD การชิงโครไนซ์ลักษณะดังกล่าวจะเกิดขึ้นเมื่อส่งข้อมูลไบต์ต่ำออก

พิจารณาทางอินพุทของ ICL7109 เมื่อไฟเลี้ยงเป็นแบบบวกลบสมมาตร อินพุทอิมพีแดนซ์ของ IC จะสูงมากถึง 1G ดังนั้นหากต้องการมัลติเพล็กซ์สัญญาณอินพุทเข้า A/D อาจจะใช้วิธี HOLD ค่าอินพุทใน C ที่มีการรั่วไหลของประจุต่ำ ในกรณีที่อินพุทแบบชุดเดียวให้นำขาอินพุททั้งสองขอร์ตทางสัญญาณเข้าหากันผ่าน JP4 และ C1

เมื่อต้องการวัดสัญญาณแบบสมมาตร (ดิฟเฟอเรนเชียล) จะป้อนสัญญาณเข้าอินพุทที่เป็นลบ โดยใช้กราวด์เป็นจุดอ้างอิง และในกรณีที่วัดแบบไม่สมมาตร (สามารถให้อินพุท LO ลงกราวด์โดยผ่าน JP1 ตัวความต้านทาน R1 จะทำหน้าที่ discharge ประจุของ C1 ผ่านความต้านทานของ Source ที่ต้องการวัด

บทที่ 4

MICROCONTROLLER 8031 และชุด RTC

4.1 คุณสมบัติและการใช้

ไมโครคอนโทรลเลอร์เบอร์ 8031 เป็น CPU ในตระกูล MCS-51 ของบริษัทอินเทล ถูกผลิตขึ้นมาให้มีคุณสมบัติเหมาะสมกับงานด้านระบบการควบคุมต่าง ๆ ในลักษณะที่เรียกว่า Single Chip Microcontroller ดังนั้นในโครงงานนี้จึงได้ใช้ CPU เบอร์นี้เป็นตัวควบคุมการทำงานโดยจะเป็นตัวควบคุมการรับและส่งข้อมูลของหน่วยความจำ โดยอาศัยหน่วยความจำภายนอกและพอร์ตอินพุต เอาต์พุต ภายนอก เพื่อให้สามารถทำงานได้ตามที่ต้องการ

คุณสมบัติของ CPU เบอร์ 8031 มีดังนี้

- CPU มีขนาด 8 บิต
- หน่วยความจำภายใน 128 Byte
- พอร์ตสื่อสารข้อมูลอนุกรมแบบ Full duplex 1 ช่อง
- ตัวจับเวลา/ตัวนับ (time/counter) ขนาด 16 บิต 2 ชุด
- พอร์ตอินพุต/เอาต์พุต ขนาด 8 บิต 4 พอร์ต
- โครงสร้างการอินเทอร์รัพต์ 5 แหล่ง
- อ้าง Address โปรแกรมภายนอกได้ 64 Kbyte
- มีวงจรรอสัญญาณในตัวภายใน

ลักษณะการจัดการกับหน่วยความจำภายนอก

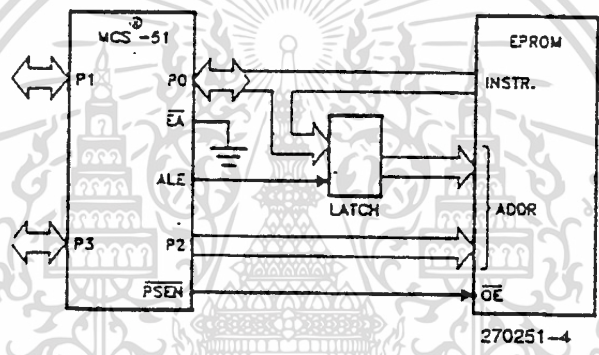
ในระบบของ CPU 8031 จะมีการแบ่งหน่วยความจำออกเป็น 2 ลักษณะ ตามชนิดของแฟ้มข้อมูลที่เก็บดังนี้

- หน่วยความจำของข้อมูล (Data Memory)
- หน่วยความจำโปรแกรม (Program Memory)

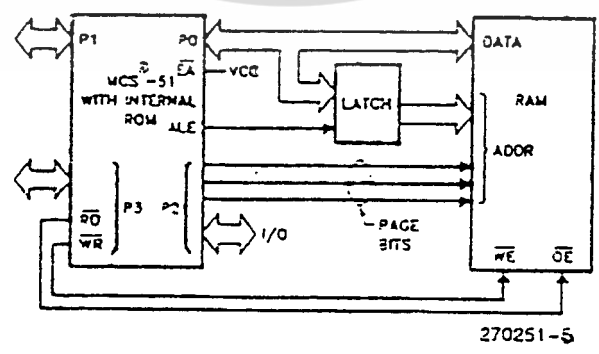
หน่วยความจำข้อมูล หมายถึง ส่วนที่เป็น RAM ซึ่งเราสามารถอ่านหรือเขียนเป็นข้อมูลเปลี่ยนแปลงได้ตลอดเวลาแต่ไม่สามารถประมวลผลโปรแกรมบนหน่วยความจำนี้ได้

หน่วยความจำโปรแกรม หมายถึง หน่วยความจำที่อ่านได้เพียงอย่างเดียว (ROM) ซึ่งเป็นส่วนหนึ่งที่บรรจุโปรแกรมที่ต้องการให้ CPU ทำงาน

โดยหน่วยความจำทั้ง 2 โปรแกรมนี้ ถูกแยกออกจากกันด้วยคำสั่งทาง Software และ ลักษณะการติดต่อกับ Hardware ด้วย คือ จะมีคำสั่งเฉพาะสำหรับการติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่งและจัดสัญญาณสโตรบในการติดต่อกับหน่วยความจำแต่ละชนิดแยกจากกันด้วย ดังรูปที่ 4.1 และรูปที่ 4.2



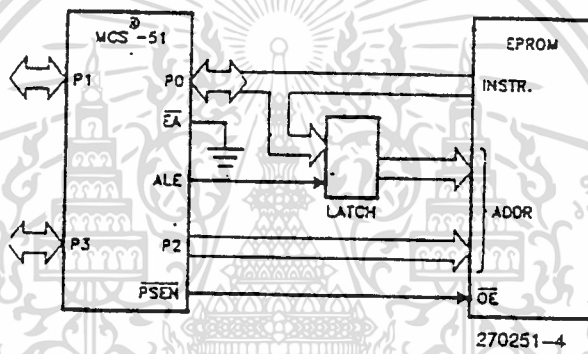
รูปที่ 4.1 แสดงการใช้หน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป



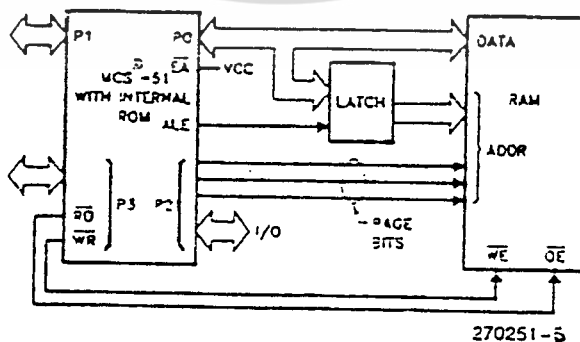
รูปที่ 4.2 แสดงการใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป

หน่วยความจำโปรแกรม หมายถึง หน่วยความจำที่อ่านได้เพียงอย่างเดียว (ROM) ซึ่งเป็นส่วนหนึ่งที่บรรจุโปรแกรมที่ต้องการให้ CPU ทำงาน

โดยหน่วยความจำทั้ง 2 โปรแกรมนี้ ถูกแยกออกจากกันด้วยคำสั่งทาง Software และ ลักษณะการติดต่อกับ Hardware ด้วย คือ จะมีคำสั่งเฉพาะสำหรับการติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่งและจัดสัญญาณสโตรบในการติดต่อกับหน่วยความจำแต่ละชนิดแยกจากกันด้วย ดังรูปที่ 4.1 และรูปที่ 4.2



รูปที่ 4.1 แสดงการใช้หน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป



รูปที่ 4.2 แสดงการใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป

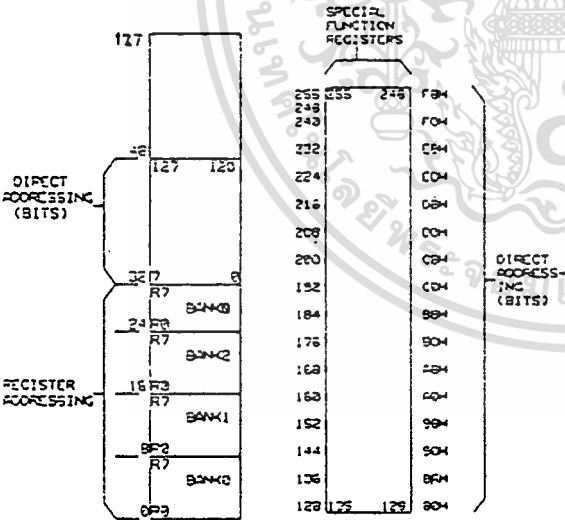
กับหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิปจำนวน 2 กิโลไบต์

หน่วยความจำโปรแกรม

ใน CPU เบอร์ 8031 จะไม่มีหน่วยความจำโปรแกรมภายใน ดังนั้นต้องต่อกับหน่วยความจำภายนอก

หน่วยความจำข้อมูล

ประกอบด้วยความจำข้อมูลภายในและภายนอก หน่วยความจำข้อมูลภายนอก จะเข้าถึงได้ด้วยการใช้คำสั่ง MOVX หน่วยความจำข้อมูลภายในจะแบ่งเป็นลักษณะงาน ดังนี้คือ จำนวน 128 ไบต์ ของบริเวณตำแหน่งล่างในเนื้อที่แรมภายในและอีก 128 ไบต์ เป็นของบริเวณตำแหน่งของแรมภายใน ส่วนบนนี้จะมีเฉพาะในเบอร์ 8032/8052 เท่านั้น และส่วนของ 128 ไบต์ อีกบริเวณหนึ่งใช้เป็นรีจิสเตอร์ฟังก์ชันพิเศษ ขณะที่ใช้ส่วนบนของแรมภายในและบริเวณของ SFR ทั้งสองส่วนนี้จะถูกป้อนส่วนให้ได้ค่าแอดเดรสภายใน แต่จะเข้าถึงในแต่ละบริเวณของทั้งสองบริเวณนี้ได้ด้วยการใช้โหมดการกำหนดเลขที่อยู่ที่แตกต่างกัน



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

รูปที่ 4.3 A แสดงถึงแผนที่ของหน่วยความจำข้อมูลภายในและ SFR

รูปที่ 4.3 B แสดงถึงแผนที่การหน่วยกำหนดตำแหน่งบิต

รูปที่ 4.3 A แสดงถึงแผนที่ของหน่วยความจำข้อมูลภายในและ SFR โดยแบ่งเป็น 4

แบงค์ในแต่ละแบงค์มีรีจิสเตอร์ 8 ตัว มีตำแหน่งตั้งแต่ 0 ถึง 31 ในบริเวณล่างของแรมแบงค์เหล่านี้จะถูกเรียกใช้ให้อินาเบิลได้คราวละหนึ่งแบงค์เท่านั้น ด้วยการกำหนดเริ่มแรกภายในสองบิตของรีจิสเตอร์ OSW ว่าจะเลือกใช้ในแบงค์ภายใน 4 แบงค์ และบริเวณตำแหน่งตั้งแต่ 20H ถึง 2FH จำนวน 16 ตำแหน่ง ๆ ละ 1 ไบต์ สามารถที่จะกำหนดเลขที่อยู่ของแต่ละบิตได้ ดังแสดงในรูปที่ 4.3 B เป็นบิตแรมแอดเดรสและเนื้อที่รีจิสเตอร์ SFR สามารถที่จะกำหนดได้ดังแสดงในรูปที่ 4.4

Direct Byte Address	MSB								LSB	Hardware Register Symbol
240	F7	F6	F5	F4	F3	F2	F1	F0		B
224	E7	E6	E5	E4	E3	E2	E1	E0		ACC
	CY	AC	F0	RS1	RS0	OV		P		
208	D7	D6	D5	D4	D3	D2	D1	D0		PSW
					PS	PT1	PX1	PT0	PX0	
184	-	-	-	BC	BB	BA	B9	BS		IP
176	B7	B6	B5	B4	B3	B2	B1	B0		P3
	EA			ES	ET1	EX1	ET0	EX0		
168	AF	-	-	AC	AB	AA	A9	AS		IE
160	A7	A6	A5	A4	A3	A2	A1	A0		P2
	SM0	SM1	SM2	REN	THS	RB3	TI	RI		
152	9F	9E	9D	9C	9B	9A	99	98		SCON
144	97	96	95	94	93	92	91	90		P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0		
136	SF	SE	SD	SC	SB	SA	S9	SS		TCON
128	S7	S6	S5	S4	S3	S2	S1	S0		P0

รูปที่ 4.4 ตำแหน่งของรีจิสเตอร์ SFR และบิตแอดเดรสของ SFR

การต่อ CPU กับหน่วยความจำภายนอก

เพื่อให้วงจรสามารถทำงานได้อย่างถูกต้องตามที่ต้องการจึงต้องมีการกำหนดขาของ CPU เข้ากับการต่อร่วมกันกับวงจรภายนอกโดยอธิบายได้ดังนี้

- ขา EA/VP ถ้าขาขานี้ลงกราวด์จะเป็นการติดต่อกับโปรแกรมภายนอก ซึ่งจะมีผลต่อพอร์ต P0 และ P2 ดังนี้
- P0 (ขา 32-39) จะมัลติเพล็กซ์ระหว่างข้อมูล D0-D7 และแอดเดรส A0-A7
 - P2 (ขา 21-28) จะเป็นแอดเดรส A8-A15
- ในการใช้งานในลักษณะนี้พอร์ต P0 จะต้องมีการดึงขึ้น (pull up) ด้วย
- ขา ALE/P จะเป็นสัญญาณแอดเดรสแลตซ์ชีนาเบิ้ลออกจากตัว CPU เพื่อบอกอุปกรณ์ภายนอกให้ทำการแลตซ์แอดเดรสต่ำ ที่ถูกส่งออกมาในขณะที่เฟลต์กำลัง
- ขา RD เป็นขาสัญญาณการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก
- ขา WR เป็นขาสัญญาณการเขียนข้อมูลจาก CPU ไปยังหน่วยความจำข้อมูลภายนอก
- ขา PSEN เป็นขาสัญญาณออกจาก CPU ขณะทำการเฟลต์สัญญาณจากหน่วยความจำโปรแกรม

4.2 Real Time Clock (RTC) # DS 1202

วงจรที่สร้างฐานเวลาให้แก่ไมโครคอนโทรลเลอร์รวมแล้วมี 2 วิธี วิธีแรกจะใช้ชิปเพื่อส่งค่าเวลาปัจจุบัน ในขณะใด ๆ ให้แก่ไมโครคอนโทรลเลอร์ ส่วนวิธีที่สองใช้วงจรกำเนิดความถี่ที่มีค่าคงที่ป้อนให้แก่เคาน์เตอร์ของไทม์เมอร์ 0 หรือไทม์เมอร์ 1 เพื่อให้ไทม์เมอร์ทั้งสองอินเตอร์รัปต์ซีพียู ในช่วงเวลาตามที่กำหนดจากวงจรภายนอกได้ แต่ในที่นี้จะใช้วิธีแรก ซึ่งใช้ชิปทำหน้าที่สร้างฐานเวลาจริงให้แก่ระบบ เนื่องจากมีขนาดเล็กและใช้พลังงานน้อยมาก มีความเที่ยงตรงสูง ในปัจจุบันมีอยู่ด้วยกันหลายเบอร์ โดยโครงการนี้เลือกใช้ชิปเบอร์ DS1202 “Serial Timekeeper Chip” มี Real time Clock/Calendar และ Static RAM ขนาด 24 ไบต์ ใช้สายเพียง 3

เส้น ในการเชื่อมต่อกับไมโครคอนโทรลเลอร์ เพื่อรับส่งข้อมูลเกี่ยวกับเวลา ขาที่ใช้รับส่งข้อมูลทั้งสามคือ

- RST (reset)
- I/O (data line)
- SCLK (serial clock)

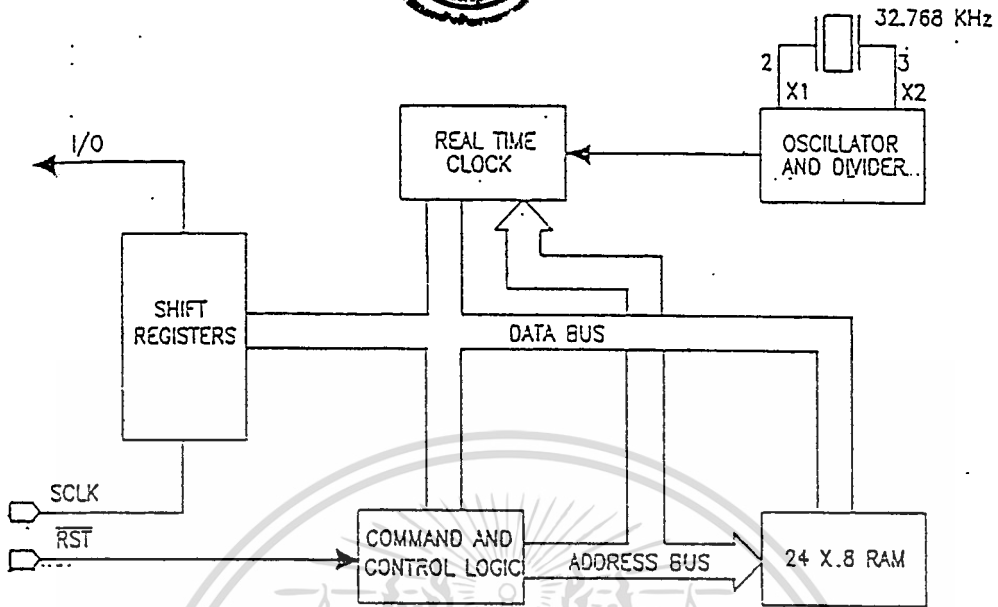
คุณสมบัติที่สำคัญ

- ทำหน้าที่นับวินาที นาฬิกา ชั่วโมง วันที่ของเดือน เดือน ปี รวมทั้งคำนวณปีอธิกสุรทินให้เองโดยอัตโนมัติ
- หน่วยความจำขนาด 24 ไบต์ สำหรับเก็บข้อมูลทั่ว ๆ ไป
- ใช้การติดต่อแบบอนุกรม โดยใช้สายเพียง 3 เส้นเท่านั้น
- ตัวชิปเองให้เลือกทั้งแบบ 8 PIN DIP หรือ 16 PIN SOIC เพื่อใช้สำหรับแผ่นวงจรชนิด surface mount
- ระดับสัญญาณ TTL Compatible ($V_{cc} = 5$ โวลต์)
- ช่วงอุณหภูมิระหว่าง -40 องศา +88 องศาเซลเซียส

โครงสร้างของชิป RTC DS1202

โครงสร้างภายในของ RTC DS1202 ประกอบไปด้วยส่วนที่สำคัญ ๆ ดังแสดงในรูปที่ 4.5 ดังนี้คือ

- shift register
- control logic
- oscillator
- real time clock
- RAM



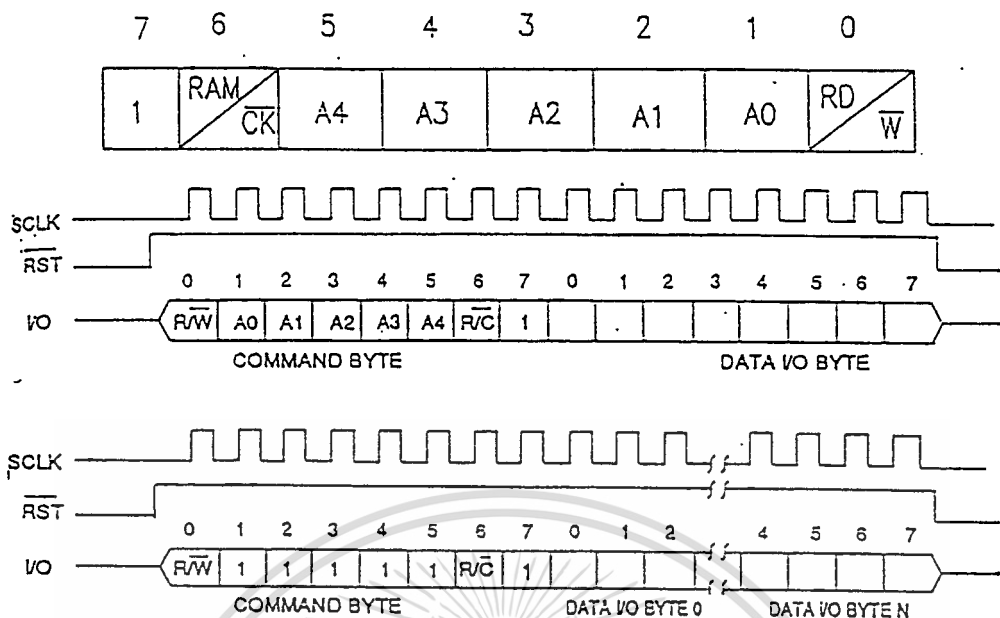
รูปที่ 4.5 โครงสร้างภายในของ RTC DS1202

โครงสร้างของ Command byte

ในการรับหรือส่งข้อมูลใด ๆ ให้แก่ RTC 1202 ไมโครคอนโทรลเลอร์ จะต้องส่งข้อมูลที่ เป็นคำสั่งควบคุมการติดต่อ ซึ่งมีขนาด 8 บิต (1 ไบต์) ที่ชื่อว่า Command byte และให้ขา RST มีสถานะเป็น 1 จากนั้นส่งข้อมูล 8 บิตเข้าไปไว้ใน shift register เพื่อจะเป็นตัวกำหนดการทำงานของ RTC แต่ละบิตมีความหมายแตกต่างกันดังนี้

- MSB (บิต 7) 1 เสมอ ถ้าเป็น 0 การทำงานต่อจากนี้จะถูกหยุดไว้หมด
- บิต 6 ถ้าเป็น 0 จะติดต่อกับรีจิสเตอร์สำหรับเก็บเวลา (clock/calendar) เป็น 1 จะติดต่อกับหน่วยความจำ
- บิต 1 ถึง 5 จะกำหนดตำแหน่งหน่วยความจำ การอ่านหรือเขียนข้อมูล ควบคุมด้วยบิต 0
- บิต 0 ถ้าเป็น 0 หมายถึง การเขียนข้อมูลลงไปในชิป ถ้าเป็น 1 หมายถึง การอ่านข้อมูลจากชิป

ในการส่ง command byte ไปยังชิป RTC จะเริ่มต้นด้วยบิต 0 ก่อนเสมอ



รูปที่ 4.6 แสดงโครงสร้างของ Command byte

Burst Mode หมายถึง การรับหรือส่งข้อมูลครั้งละหลายไบต์ในการติดต่อแต่ละครั้ง โดยสามารถกำหนดได้ว่าข้อมูลที่ต้องการติดต่อเป็นรีจิสเตอร์สำหรับเก็บเวลาหรือหน่วยความจำที่ใช้เก็บข้อมูล กำหนดโดย command byte ในการรับหรือส่งข้อมูลจะต้องเริ่มต้นที่บิต 0 ของหน่วยความจำตำแหน่ง 0 ก่อนเสมอ ไม่ว่าจะเป็นรีจิสเตอร์เก็บเวลา หรือหน่วยความจำเก็บข้อมูลทั่วไป

Write Protect Command Byte เป็นรีจิสเตอร์ตำแหน่งที่ 7 ทำหน้าที่เก็บเวลา ดังแสดงในรูปที่ 4.7 โดยมี Write protect bit ซึ่งเป็นบิตที่ 7 ของรีจิสเตอร์ตัวนี้เป็นตัวกำหนดการทำงาน หาก Write protect bit เป็น 0 หมายถึงสามารถเขียนข้อมูลใด ๆ ลงไปยังรีจิสเตอร์ที่เก็บเวลาหรือหน่วยความจำได้ หากบิตนี้เป็น 1 หมายถึง ชิป RTC อยู่ในสถานะป้องกันการเขียนข้อมูล

Reset and clock control การรับหรือส่งข้อมูลจะต้องเริ่มโดยให้ขา RST เป็น 1 เสมอ โดย RST มีหน้าที่หลัก 2 ประการดังนี้

1. RST ใช้ควบคุมการเขียนหรืออ่านข้อมูลใน shift register
2. RST ใช้เป็นสัญญาณในการหยุดการทำงานใด ๆ ปกติระหว่างการติดต่อ ขา RST จะเป็น 1 ตลอดเวลา หากเป็น 0 จะหมายถึง ยกเลิกการติดต่อ

Data Input ในตอนเริ่มต้นติดต่อระหว่างชิป RTC DS1202 กับไมโครคอนโทรลเลอร์ ไบต์แรกจะต้องเป็น command byte เสมอ หากใน command byte ระบุว่าเป็นการเขียนข้อมูลไปในชิป ข้อมูลจะถูกรับเข้ามาในช่วงขอบขาขึ้นของ SCLK เท่านั้น โดยเริ่มต้นด้วยบิต 0 ก่อนเสมอ และหากเป็นคำสั่งให้รับส่งครั้งละ 1 ไบต์ เมื่อข้อมูลได้รับเข้ามาครบแล้ว สัญญาณ SCLK ที่รับได้เกินจะถูกเฉลยไป หากเป็นคำสั่งให้รับส่งแบบ burst mode ซึ่งรับส่งครั้งละ 24 ไบต์ ก็มีลักษณะเช่นเดียวกันคือ เมื่อรับข้อมูลครบ 24 ไบต์แล้ว สัญญาณ SCLK ที่รับได้เกินจะถูกเฉลยเช่นกัน

Data Output หลังจากรับ command byte แล้ว หากมีระบุว่าเป็นการอ่านข้อมูลจากชิป RTC DS1202 ข้อมูลจะถูกส่งออกจากชิปสู่ภายนอกในช่วงขอบขาลง ของ SCLK หลังจากมีการรับ command byte เรียบร้อยแล้ว นั่นคือ บิตแรกที่จะถูกที่จะถูกส่งออกจาก RTC จะเกิดขึ้นในช่วงขอบขาลงของสัญญาณนาฬิกาที่ 9 นั่นเอง (ต่อจาก command byte)

Clock/Calendar คือ รีจิสเตอร์ 8 ตัว ดังแสดงในรูปที่ 4.7 โดยข้อมูลในรีจิสเตอร์เหล่านี้จะอยู่ในรูปของรหัส BCD เท่านั้น

Clock Halt Flag บิต 7 ของรีจิสเตอร์ที่เก็บค่าวินาที ดังรูปที่ 3.3 จะเป็นตัวบอกให้ชิป RTC DS1202 หยุดการทำงานของวงจรในส่วนออสซิลเลเตอร์ เมื่อบิตนี้มีค่าเป็น 1 ซึ่งเป็นผลให้นาฬิกาภายในชิปหยุดทำงานไปด้วย และจะบังคับให้ชิปอยู่ในสถานะ low power standby mode โดยใช้กระแสไม่เกิน 100 นาโนแอมแปร์ และเมื่อบิตนี้เป็น 0 อีกครั้ง วงจรออสซิลเลเตอร์จะเริ่มทำงานต่อทันที

AM-PM/12-24 mode บิต 7 ของรีจิสเตอร์ที่เก็บค่าชั่วโมง ดังในรูปที่ 3.3 ถูกกำหนดให้เป็น 12/24 how mode select bit นั่นคือ เป็นตัวเลือกว่าจะให้รีจิสเตอร์นี้เก็บค่าชั่วโมงแบบ 12 ชั่วโมง หรือ 24 ชั่วโมง โดย

- บิตที่ 7 เป็น 1 จะเป็นการเลือกให้เก็บค่าแบบ 12 ชั่วโมง โดยมีบิต 5 เป็นตัวบอกว่าเป็นช่วงกลางวันหรือกลางคืน (AM/PM indicator) โดย 1 จะหมายถึง PM และ 0 หมายถึง AM
- บิตที่ 7 เป็น 0 จะเป็นการเลือกให้เก็บค่าแบบ 24 ชั่วโมง และบิต 5 จะเป็นบิตที่แสดงหลักสิบตัวที่ 2 ของชั่วโมง

Clock/Calendar Burst Mode ใน Mode นี้ การรับหรือส่งข้อมูลคราวละ 8 ไบต์ติดต่อกัน (รับหรือส่งข้อมูลในรีจิสเตอร์ที่เก็บค่าเวลาทั้ง 8 ตัว) โดยจะเป็นการรับหรือส่งขึ้นกับ command byte เริ่มต้นด้วยบิต 0- ของรีจิสเตอร์ 0 ก่อนเสมอ

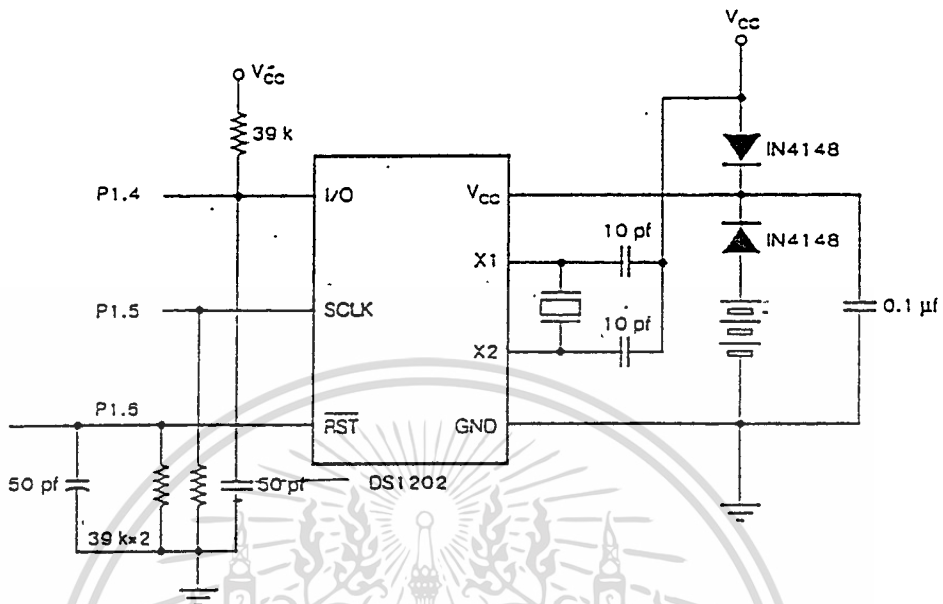
RAM Burst Mode มีข้อมูลจำนวน 24 ไบต์ จะรับส่งกัน โดยเริ่มต้นด้วยบิต 0 ของ หน่วยความจำตำแหน่ง 0 ก่อนเสมอ

REGISTER	FUNCTION	COMMAND ADDRESS (HEX)	WRITE=W READ=R	RANGE DATA (BCD)	REGISTER DEFINITION							
					7	6	5	4	3	2	1	0
0	SECONDS	80	W	00-59	CH	10 SEC			SEC			
		81	R									
1	MINUTES	82	W	00-59	0	10 MIN			MIN			
		83	R									
2	12HRS	84	W	01-12	12	0	AP	HR	HOUR			
	24 HRS	85	R	00-23	24	0	10	HR				
3	DATE	86	W	01-31	0	0	10DATE		DATE			
		87	R									
4	MONTH	88	W	01-12	0	0	0	10M	MONTH			
		89	R									
5	DAY	8A	W	01-07	0	0	0	0	DAY			
		8B	R									
6	YEAR	8C	W	00-99		10 YEAR			YEAR			
		8D	R									
7	WRITE PROTECT	8E	W	00-80	WP	ALWAYS ZERO						
		8F	R									

31	CLOCK BURST	BE	W
		BF	R
0	RAM 0	C0	W
		C1	R
•	•	•	•
		•	•
23	RAM 23	EE	W
31	RAM BURST	FE	W
		FF —	R

รูปที่ 4.7 แสดงข้อมูลการทำงานของรีจิสเตอร์และ RAM ใน DS 1202

การใช้งานของ RTC DS1202 กับ 8031



รูปที่ 4.8 ชิป RTC DS1202 ต่อเข้าพอร์ต 1 ของ 8031

การติดต่อกับชิป RTC DS1202 จะต้องให้ RS+ และ SCLK เป็น 0 ทั้งคู่ก่อน หลังจากนั้น RST จะต้องเป็น 1 เพื่อให้ติดต่อกับรีจิสเตอร์ DS1202 ได้ และระหว่างติดต่อกับ DS1202 RST จะต้องเป็น 1 โดยการอินพุตข้อมูลเข้า DS1202 จะเกิดขึ้นที่ขาขึ้นของ SCLK และการเอาต์พุตข้อมูลจาก DS1202 จะเกิดขึ้นที่ขาลงของ SCLK และหลังจากเลิกติดต่อกับ DS1202 จะต้องให้ RST เป็น 0 อีก

บทที่ 5

พอร์ตสื่อสารอนุกรม

พอร์ตสื่อสารเป็นส่วนสำคัญส่วนหนึ่งที่มีอยู่ในไมโครคอมพิวเตอร์ 16 บิต พอร์ตสื่อสารนี้มีชื่อเรียกอีกอย่างหนึ่งว่า คอม-พอร์ต(com port) ผู้ออกแบบพอร์ตสื่อสารต้องการให้เป็นไปตามมาตรฐานการเชื่อมต่อแบบอนุกรมที่เรียกว่า RS232C พอร์ตนี้เป็นทางออกของข้อมูลที่ผู้ใช้สามารถส่งหรือรับกับระบบอื่นได้ พอร์ตสื่อสารจึงเป็นพอร์ตที่จำเป็นและผู้จัดซื้อไมโครคอมพิวเตอร์มักจะกำหนดด้วยเสมอ พอร์ตสื่อสาร RS232C บนเครื่องไมโครคอมพิวเตอร์ 16 บิตมีโครงสร้างที่สามารถโปรแกรมด้วยการส่งรหัสคำสั่งให้กับชิปหลักได้ อย่างไรก็ตามผู้ออกแบบได้ให้ทางเลือกในการสื่อสารด้วยกระแสวนรอบ(current loop)หรือแบบแรงดันคือ RS232C โดยใช้จัมเปอร์เพื่อเลือกระบบสำหรับตัว RS232C นี่เป็นมาตรฐานแบบอะซิงโครนัสที่โปรแกรมสตาร์ทบิต สตอปบิต และพาริตีบิต อัตราการส่งก็สามารถกำหนดได้ตั้งแต่ 50 บอดถึง 9600 บอด ลักษณะพิเศษของวงจรคือสามารถส่งสัญญาณมาอินเทอร์พรีตซีพียูตามเงื่อนไขได้ และยังมีโครงสร้างแวร์ป้อนกลับเพื่อใช้ในการตรวจสอบระบบว่าทำงานปกติหรือไม่ได้อีกด้วย วงจรพอร์ตสื่อสารของไมโครคอมพิวเตอร์ 16 บิตนี้ใช้ไอซีหมายเลข 8250 เป็นตัวสำคัญของระบบ 8250 เป็นไอซีขนาด 40 ขา มีขีดความสามารถพิเศษดังนี้

- มีบัฟเฟอร์ในตัวเพื่อทำให้ไม่จำเป็นต้องชิงโครนัสการรับส่ง
- ใช้สัญญาณนาฬิกาอิสระต่างหากไม่ขึ้นกับสัญญาณนาฬิกาของระบบ
- มีสัญญาณตอบโต้ควบคุมโมเด็มทั้ง CTS(clear to send) , RTS(request to send) ,DSR(data set ready),

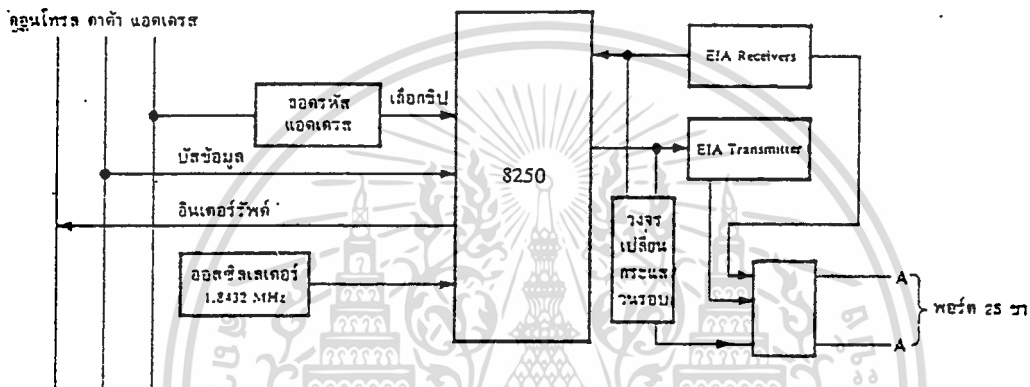
RC(ring indicator) และสัญญาณเค้เทคตัวพาหะ

- ตรวจสอบสตาร์ทบิตที่ผิดพลาด
- ตรวจสอบและสร้างสัญญาณสายขาด(line break)เพื่อใช้ในการตรวจสอบการทำงานของระบบ

ชิป 8250 มีโครงสร้างการทำงานคล้ายกับ 8251 การทำงานของ 8250 ต้องได้รับการโปรแกรมก่อน หลังจากนั้นจะทำงานตามรูปแบบที่ได้โปรแกรมไว้ จนกว่าจะมีโปรแกรมคำสั่งใหม่ อย่างไรก็ตาม 8250 มีขีดความสามารถในการทำงานที่สูงกว่า 8251 หลายอย่าง และด้วยเหตุนี้เอง 8250 จึงเป็นชิปประจำตัวที่ใช้กับเครื่องไมโครคอมพิวเตอร์ 16 บิต

5.1 โครงสร้างของพอร์ตสื่อสาร

พอร์ตสื่อสารของเครื่องไมโครคอมพิวเตอร์ 16 บิต ที่จะกล่าวถึงนี้เป็นระบบมาตรฐานตามแบบเครื่องไอบีเอ็มพีซีเอ็กซ์ที โครงสร้างบล็อกไดอะแกรมแสดงดังรูปที่ 1



รูปที่ 5.1 โครงสร้างพอร์ตสื่อสารข้อมูลที่ใช้ 8250

พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือสล็อตขนาด 31 (62) ขาของระบบนั่นเอง ซีพียูติดต่อกับ 8250 ในลักษณะพอร์ตที่เป็นอินพุตเอาต์พุต การจัดพอร์ตนี้กำหนดหมายเลขพอร์ตอย่างเจาะจง ระบบไมโครคอมพิวเตอร์ 16 บิต มีพอร์ตสื่อสารสองพอร์ตคือ คอม1 (COM1) และคอม2 (COM2) ทั้ง COM1 และ COM2 มีหมายเลขอินพุตเอาต์พุตพอร์ต ดังตารางที่ 5.1

อินพุตเอาต์พุตพอร์ต		เลือกวีรเคอร์	เลือก DLAB
พอร์ต COM1	พอร์ต COM2		
3F8	2F8	บัฟเฟอร์ IX	DLAB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB = 0 (อ่าน)
3F8	2F8	แลตซ์ตัวหาร (LSB)	DLAB = 1
3F9	2F9	แลตซ์ตัวหาร (MSB)	DLAT = 1
3F9	2F9	อีนาเบิลอินเตอร์รัพต์	
3FA	2FA	กำหนดอินเตอร์รัพต์	
3FB	2FB	ควบคุมสายสื่อสาร	
3FC	2FC	ควบคุมโมเด็ม	
3FD	2FD	แสดงสถานะสายสื่อสาร	
3FE	2FE	แสดงสถานะโมเด็ม	

ตารางที่ 5.1 หมายเลขอินพุตเอาต์พุตพอร์ตของ COM1 และ COM2

การเลือกหมายเลขอินพุตเอาต์พุตพอร์ตแยกเป็นสองกลุ่ม กลุ่มหนึ่งคือ COM1 จะกำหนดหมายเลขพอร์ตจาก 3F8 ถึง 3FE อีกกลุ่มหนึ่ง ถ้ากำหนดหมายเลขพอร์ตเป็น 2F8-2FE ในการเลือกหมายเลขรีจิสเตอร์ภายในกำหนดด้วยแอสเครส 3 บิต คือ A0,A1และA2 สำหรับการเลือก COM1 และ COM2 เราใช้แอสเครส A8 เป็นตัวเลือก การเลือกนี้กำหนดเป็นตารางได้ดังตารางที่ 5.2

แอสเครส 3F8-3FE และ 2F8-2FE										DLAB	วีรเคอร์
A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀		
1	1/0	1	1	1	1	1	X	X	X	0	บัฟเฟอร์สำหรับตัวรับข้อมูล (อ่าน) ไสลคิงสำหรับตัวส่งข้อมูล (เขียน)
							0	0	0	0	อีนาเบิลอินเตอร์รัพต์
							0	1	0	X	กำหนดอินเตอร์รัพต์
							0	1	1	X	ควบคุมสายสื่อสาร
							1	0	0	X	ควบคุมโมเด็ม
							1	0	1	X	แสดงสถานะสายสื่อสาร
							1	1	0	X	แสดงสถานะโมเด็ม
							1	1	1	X	ไม่ใช่
							0	0	0	1	แลตซ์ตัวหาร (LSB)
							0	0	1	1	แลตซ์ตัวหาร (MSB)

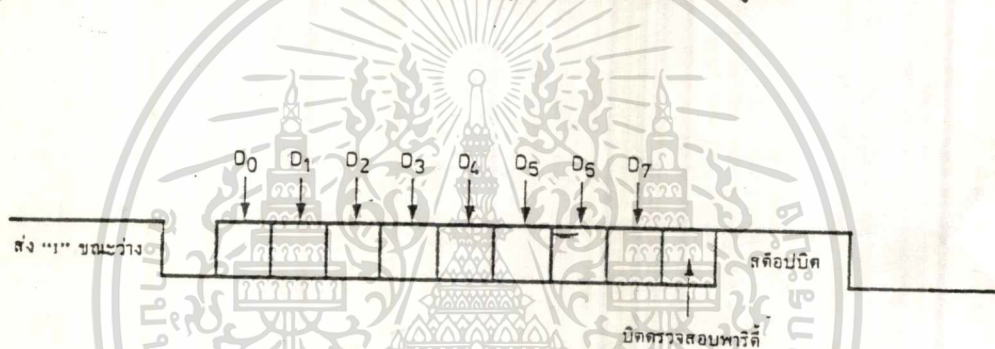
ตารางที่ 5.2 การกำหนดแอสเครสสำหรับหมายเลขพอร์ตต่าง ๆ

การอินเตอร์รัพต์

สำหรับกรณีของระบบสื่อสารอนุกรม ได้กำหนดสัญญาณการอินเตอร์รัพต์ไว้แล้ว คือให้ IRQ4 เป็นสัญญาณอินเตอร์รัพต์ของระบบ COM1 และ IRQ3 เป็นของ COM2 ในการที่จะส่งสัญญาณอินเตอร์รัพต์ต้องให้บิต 3 ของรีจิสเตอร์ควบคุมโมเด็มได้รับการเซตค่าเป็น "1" ก่อน จากนั้น ข้อมูลอินเตอร์รัพต์ที่อยู่ในรีจิสเตอร์อื่นาเปิดอินเตอร์รัพต์จะเป็นตัวส่งการอินเตอร์รัพต์

รูปแบบข้อมูลที่รับหรือส่ง

การสื่อสารข้อมูลของระบบนี้เป็นการสื่อสารแบบอะซิงโครนัส รูปแบบของข้อมูลจะมีสตาร์ทบิต บิตตรวจสอบพาริตีและสต็อปปบิต โครงสร้างของข้อมูลแต่ละเฟรมเป็นดังรูปที่ 5.2



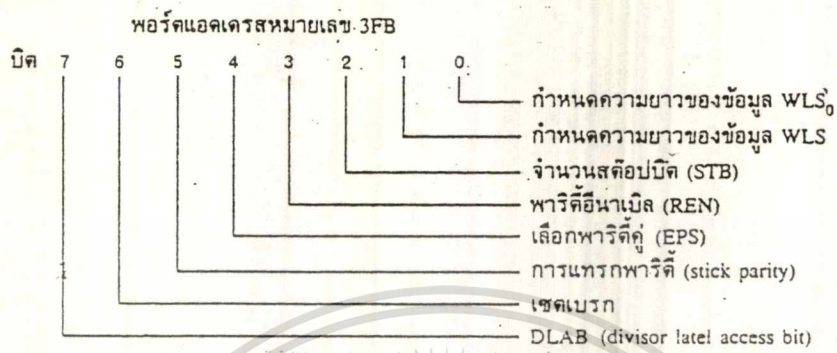
รูปที่ 5.2 รูปแบบข้อมูล 1 เฟรม

ข้อมูลบิตแรกของการส่งเป็นสตาร์ทบิต ระบบจะส่งสตาร์ทบิตก่อนหลังจากนั้นจะตามด้วยข้อมูลและแทรกด้วยบิตตรวจสอบพาริตี ตามด้วยสต็อปปบิต ขนาดของข้อมูลมีค่าได้ตั้งแต่ 5-8 บิต แต่ทว่าไปใช้ 8 บิต สต็อปปบิตมีได้ 1.5 หรือ 2 บิต ค่าเหล่านี้สามารถกำหนดโดยรีจิสเตอร์ควบคุมสายสื่อสาร

5.2 การใช้งานรีจิสเตอร์ต่างๆบน 8250

เพียงจากการใช้อะแดปเตอร์สื่อสารที่จะต้องโปรแกรมค่าไมโครโค้ดเข้าไปใน 8250 ก่อน ดังนั้นผู้ใช้ 8250 จำเป็นต้องเข้าใจว่ารีจิสเตอร์ของ 8250 มีความหมายอย่างไรอธิบายได้ดังนี้

รีจิสเตอร์ควบคุมสายสื่อสาร(line control register) ในการควบคุมรูปแบบของอะซิงโครนัส นั้น ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร รีจิสเตอร์ตัวนี้มี 8 บิต โดยแต่ละบิตมีความหมายดังนี้



รูปที่ 5.8 ค่าของบิตในรีจิสเตอร์ควบคุมสายสื่อสาร

- บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง
- โดยที่บิต 2 เป็นบิตที่ใช้ในการกำหนดจำนวนสตอปบิต ถ้าเป็น "0" หมายถึงใช้สตอปบิต 1 บิต แต่ถ้าบิต 2 เป็น "1" ในกรณีส่งแบบ 5 บิต จะมีความยาวของสตอปบิตเป็น 1.5 บิต แต่ถ้าส่งแบบ 6,7 หรือ 8 บิต ความยาวของสตอปบิตจะเป็น 2
- บิต 3 บิตนี้เป็นบิตแสดงการอานาเบลให้มีการตรวจสอบพาริตี โดยถ้าบิตนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี
- บิต 4 มีค่าเป็น "0" และบิต 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตีคู่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะเป็นพาริตีคู่
- บิต 5 เมื่อบิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" และบิต 4 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี (stick parity) ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" จะมีการกำหนดพาริตีเป็น "1"
- บิต 6 เป็นบิตที่ควบคุมการเบรก เมื่อบิต 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด
- บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิตที่จะมีผลต่อการแลตซ์ตัวหารดังได้กล่าวมาแล้วจากตารางที่ 1 และ 2

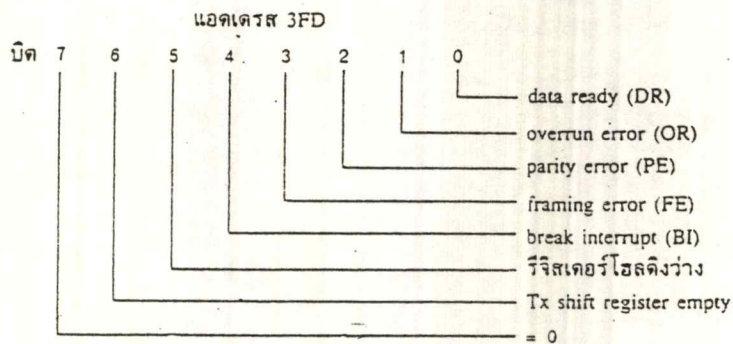
การโปรแกรมอัตราบอด(boud rate generator) อัตราบอดได้รับการกำหนดเทียบกับ สัญญาณนาฬิกา 1.8432 MHz และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1-(2-1) ค่าความถี่เอาต์พุตของ ตัวกำหนดอัตราบอดมีค่าเท่ากับ 16 อัตราบอด ดังนั้นตัวหาร = ความถี่สัญญาณนาฬิกา/(อัตราบอด 16) การกำหนดอัตราบอดด้วยการกำหนดตัวหารนี้ ตัวหารจึงมีค่าที่กำหนดในรีจิสเตอร์ 2 ตัว ตัวหารนี้ จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ DLAB =1 แล้วให้ ลดลงมาในรีจิสเตอร์ 3F8 ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F9 เมื่อ DLAB =1 จะเป็นค่าของ ตัวหาร MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตารางที่ 5.3

อัตราบอด	ตัวหาร		ความถี่เอาต์พุต
	ส่วนสูง	ส่วนต่ำ	
50	2304	900	—
75	1536	600	—
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	—
300	384	180	—
600	192	0C0	—
1200	96	060	—
1800	64	040	—
2000	58	03A	0.69
2400	48	030	—
3600	32	020	—
4800	24	018	—
7200	16	010	—
9600	12	00C	—

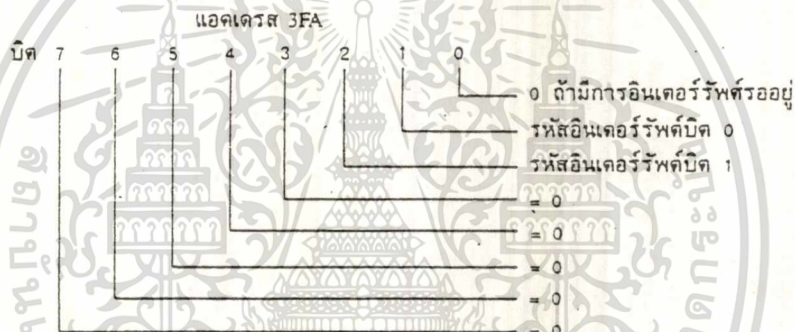
ตารางที่ 5.3 ค่าตัวหารสำหรับการกำหนดอัตราบอด

รีจิสเตอร์แสดงสถานะสายสื่อสาร (line status register) รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อในสายสื่อสาร ค่าของบิตต่างๆ ในรีจิสเตอร์นี้เป็นดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.4 ค่าของบิตในรีจิสเตอร์แสดงสถานะสายสื่อสาร



รูปที่ 5.5 ค่าของบิตในรีจิสเตอร์กำหนดอินเตอร์รัพต์

บิต 0 บิตนี้เป็นบิตที่บอกสถานะการรับข้อมูล ถ้าบิตนี้เป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว หรือจะให้ซีพียูเขียนข้อมูลกลับมายังรีจิสเตอร์นี้ก็ได้อีก

บิต 1 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด overrun error (OR) กล่าวคือขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไป ปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับบนบัฟเฟอร์นี้ บิตนี้จะรีเซตโดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 2 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด parity error (PE) กล่าวคือถ้ามีการตรวจสอบบิตพาริตีแล้วไม่เป็นไปตามที่กำหนดไว้ บิตนี้จะได้รับการรีเซตโดยซีพียู เมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 3 บิตนี้มีค่าเป็น "1" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่นตรวจสอบจำนวนบิต โดยดูที่พาริตีและสตอปบิตไม่เป็นไปตามที่กำหนด

บิต 4 บิตนี้เรียกว่า break interrupt (BI) บิตนี้จะได้รับการเซตค่าให้เป็น "1" ถ้าหากว่ารับข้อมูลอินพุตเป็น "0" เป็นเวลายาวนานว่าเวิร์คของการสื่อสาร

บิต 5 บิตนี้เป็นบิตที่บอกว่า 8250 พร้อมทั้งจะรับข้อมูลจากสายสื่อสาร บิตนี้จะได้รับการเซตค่าเป็น "1" บิตนี้ยังสร้างสัญญาณอินเตอร์รัพต์เพื่อส่งไปบอกซีพียูด้วย บิตนี้จะมีสถานะเซตเมื่อมีการส่งด้วยข้อมูลจากโฮลด์รีจิสเตอร์ไปยังชิปรีจิสเตอร์เพื่อพร้อมที่จะส่ง

บิต 6 เป็นบิตที่จะบอกว่าชิปรีจิสเตอร์ว่างหรือเปล่า บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิต 7 จะเป็น 0 ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพต์ (IIR - interrupt identification register)

ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพต์ในชิป เพื่อให้การทำงานระหว่าง 8250 กับ ซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และเพื่อให้ผู้เขียนซอฟต์แวร์สามารถเขียนซอฟต์แวร์ได้ง่ายและสั้นลงได้มาก 8250 กำหนดความสำคัญของการอินเตอร์รัพต์ไว้ 4 ระดับคือ ระดับแรก-สถานะการรับข้อมูลจากสายสื่อสาร ระดับที่สอง-การพร้อมรับข้อมูล ระดับที่สาม-ขณะรีจิสเตอร์โฮลด์สำหรับส่งข่าว ระดับที่สี่-สัญญาณสถานะโมเด็ม

ในขณะที่มีความต้องการอินเตอร์รัพต์หลายระดับพร้อมกัน 8250 จะให้ระบบที่มีความสำคัญน้อยกว่ารอไว้ก่อนโดยเก็บสถานะการอินเตอร์รัพต์ไว้ในรีจิสเตอร์ กำหนดอินเตอร์รัพต์ความหมายของรีจิสเตอร์นี้มีดังนี้

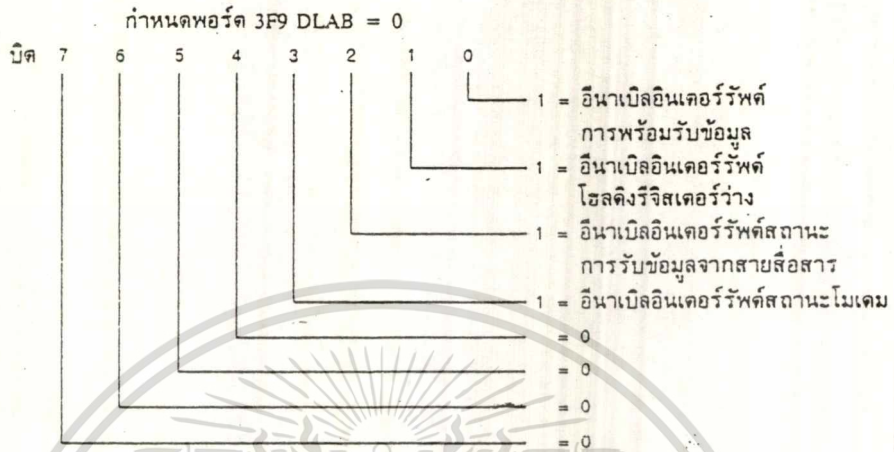
บิต 0 เป็นบิตที่ใช้แสดงว่ามีอินเตอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถให้ซีพียูตรวจสอบโดยวิธีการ polling ได้ถ้าบิตเป็น "1" หมายถึงไม่มีการอินเตอร์รัพต์เกิดขึ้น

บิต 1-2 เป็นบิตที่แสดงความหมาย บอกว่าการอินเตอร์รัพต์ที่เกิดขึ้นนั้นมาจากอินเตอร์รัพต์จากฟังก์ชันใด

บิต 3-7 มีค่าเป็น "0"

รีจิสเตอร์อินาเบิลอินเตอร์รัพต์ (INTRPT -interrupt enable register) ใน COM1 เมื่อให้ DLAB =0 พอร์ต 3F9 จะเป็นรีจิสเตอร์อินาเบิลอินเตอร์รัพต์ ผู้ใช้สามารถกำหนดให้เกิดอินเตอร์รัพต์หรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจิสเตอร์นี้ จากที่กล่าวแล้วว่าการอินเตอร์รัพต์ของ 8250 นี้

มี 4 แบบ ดังนั้นจึงต้องกำหนดการอินาเบิลได้ทั้ง 4 แบบ โดยการใช้ข้อมูลแต่ละบิตของรีจิสเตอร์นี้เพื่อกำหนดการอินาเบิล ข้อมูลที่อยู่ในรีจิสเตอร์นี้มีความหมายดังนี้



รูปที่ 5.6 ค่าของบิตในรีจิสเตอร์อินาเบิลอินเตอร์รัพท์

บิต 2 บิต 1 บิต 0	ระดับ ความสำคัญ	ผลของ อินเตอร์รัพท์	แหล่งเกิด อินเตอร์รัพท์	การวัด ควบคุมอินเตอร์รัพท์
0 0 1 1 1 0	สูงสุด	ไม่เกิด สถานะการรับข้อมูล จากสายส่งสื่อสาร	ไม่เกิด overrun error parity error framing error break interrupt	อ่านข้อมูลจากรีจิสเตอร์ สถานะ สายสื่อสาร
1 0 0	ที่สอง	การพร้อมรับข้อมูล	มีข้อมูลที่ค้างรับ	การอ่านข้อมูลจากบัฟเฟอร์
0 1 0	ที่สาม	โฮลดิ้งรีจิสเตอร์ สำหรับส่งข่าว	โฮลดิ้งรีจิสเตอร์ สำหรับส่งข่าว	อ่านรีจิสเตอร์กำหนด อินเตอร์รัพท์ IIR หรือ เขียนลงไปยังโฮลดิ้ง รีจิสเตอร์สำหรับส่ง
0 0 0	ที่สี่	สถานะโมเด็ม	CIS DSR RI ตรวจสอบสายส่ง โดยตรง	อ่านรีจิสเตอร์แสดงสถานะ ของโมเด็ม

ตารางที่ 5.4 ฟังก์ชันการอินเตอร์รัพท์รหัสอินเตอร์รัพท์

บิต 0 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอินาเบิลอินเตอร์รัพท์การพร้อมรับข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิต 1 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนามิเตอร์รหัสโสดจิงรีจิสเตอร์ว่าง

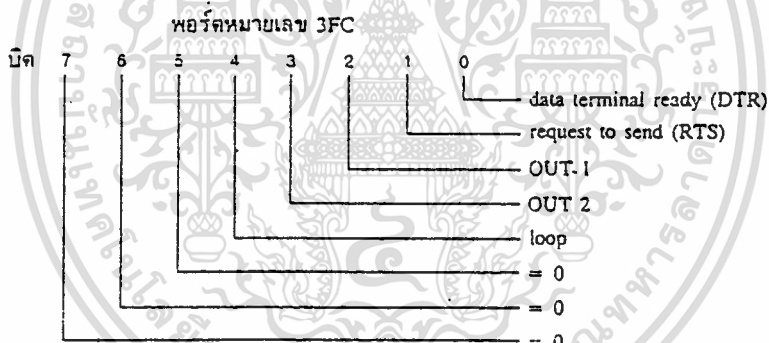
บิต 2 บิตนี้ได้รับการเซตเป็น "1"เมื่อต้องการอีนามิเตอร์รหัสจากสถานะการรับข้อมูลจากสายสื่อสาร

บิต 3 บิตนี้ได้รับการเซตเป็น "1"เมื่อต้องการอีนามิเตอร์รหัสจากสถานะโมเดม

บิต 4-7 ได้รับการกำหนดให้เป็น 0 เสมอ

รีจิสเตอร์ควบคุมโมเดม (modem control register) รีจิสเตอร์ตัวนี้มีไว้ให้ซีพียูส่งผ่านข้อมูลมาเก็บเพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเดม การกำหนดพอร์ตของรีจิสเตอร์ตัวนี้ คือ 3FC ข้อมูลต่างๆที่มีในรีจิสเตอร์ตัวนี้ มีความหมายดังนี้

บิต 0 บิตนี้มีความหมายถึงการควบคุมสัญญาณ DTR เมื่อบิตนี้มีค่าเป็น "1" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "0" และถ้าบิตนี้มีค่าเป็น "0" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "1"



รูปที่ 5.7 ค่าของบิตในรีจิสเตอร์ควบคุมโมเดม

บิต 1 บิตนี้มีความหมายถึงสัญญาณ RTSซึ่งจะมีผลเหมือนกับบิต 0 ในกรณีของ DTR

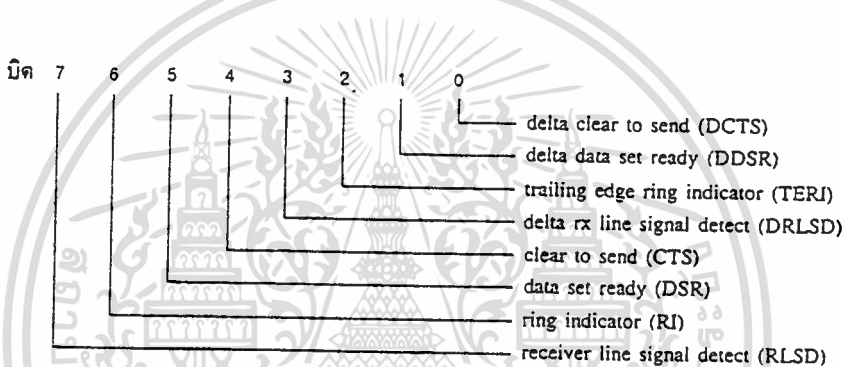
บิต 2 บิตนี้ใช้ควบคุมเอาต์พุต 1 (OUT1) ซึ่งจะมีผลเหมือนกับบิต 0

บิต 3 บิตนี้ใช้ควบคุมเอาต์ 2 (OUT2) ซึ่งจะมีผลเหมือนกับบิต 0

บิต 4 บิตนี้จะใช้สำหรับการกำหนดวงจรรอบสำหรับการตรวจสอบ 8250 เมื่อบิต 4 นี้ได้รับการเซตเป็น "1" สิ่งที่จะเกิดขึ้นเป็นดังนี้ ข้อมูลที่ SOUT จะได้รับการเซตให้เป็นลอจิก "1" ขาข้อมูลอินพุต SIN จะได้รับการแยกตัวออกข้อมูลของเอาต์พุตชิปรีจิสเตอร์จะได้รับการป้อนกลับมายังรีจิสเตอร์ ข้อมูลอินพุต ส่วนสัญญาณ CTS,DSR,RLSD และ RI จะได้รับการแยกออกจากระบบแต่

ข้อมูลอินพุต SIN จะได้รับการแยกตัวออกข้อมูลของเอาต์พุตชิปรีจิสเตอร์จะได้รับการป้อนกลับมายังรีจิสเตอร์ ข้อมูลอินพุต ส่วนสัญญาณ CTS, DSR, RLSD และ RI จะได้รับการแยกออกจากระบบแต่สัญญาณควบคุมโมเดมคือ DTR, RTS, OUT1 และ OUT2 จะต่อเข้ากับสัญญาณทั้งสี่ที่เป็นอินพุต ดังนั้นจึงตรวจสอบระบบการทำงานได้

รีจิสเตอร์แสดงสถานะโมเดม รีจิสเตอร์ตัวนี้จะเป็นตัวรับสถานะจากโมเดมมาเก็บไว้เพื่อให้ซีพียูสามารถอ่านตรวจสอบดูได้ สถานะของข้อมูลจะแยกทีฟเมื่อมีข้อเป็น "1" และจะได้รับการรีเซตเมื่อซีพียูอ่านข้อมูลในรีจิสเตอร์นี้ไป พอร์ตที่ใช้ในการกำหนดเป็นพอร์ตหมายเลข 3FE ข้อมูลภายในรีจิสเตอร์นี้เป็นดังนี้



รูปที่ 5.8 ค่าของบิตในรีจิสเตอร์แสดงสถานะโมเดม

บิต 0 บิตนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ CTS กล่าวคือ เมื่อขา CTS ของ 8250 ได้เปลี่ยนสถานะหลังจากที่ซีพียูได้อ่านสถานะนี้ไปแล้ว บิตนี้จะบอกด้วยการเซต และเมื่อซีพียูอ่านก็จะได้รับการรีเซต "0" และได้รับการเซต "1" เมื่อมีการเปลี่ยนสถานะที่ขา CTS

บิต 1 เหมือนบิต 0 แต่เป็นบิตที่แสดงสถานะการเปลี่ยนแปลงของขา DSR

บิต 2 บิตนี้เป็นบิตแสดงว่าสัญญาณ RI ซึ่งเป็นอินพุตของ 8250 ได้รับการเปลี่ยนจากอน "1" มาเป็นออฟ "0"

บิต 3 บิตนี้เหมือนบิต 0 แต่เป็นบิตแสดงสถานะการเปลี่ยนแปลงของ line signal detector ซึ่งเป็นขาอินพุต RLSD

บิต 4 บิตนี้เก็บสัญญาณคอมพลิเมนต์กับสัญญาณที่ขา CTS

บิต 5 บิตนี้เก็บสัญญาณคอมพลิเมนต์กับสัญญาณที่ขา DSR

อนึ่งถ้าบิต 4 ของ MCR ได้รับการเซตหรือให้ทำลูป (loop) ตรวจสอบ ข้อมูลในบิต 4 จะเหมือนกับ RTS ใน MCR ข้อมูลในบิต 5 จะเหมือนกับ DTR ใน MCR ข้อมูลในบิต 6 จะเหมือนกับ OUT1 ใน MCR ข้อมูลในบิต 7 จะเหมือนกับ OUT2 ใน MCR

รีจิสเตอร์บัฟเฟอร์สำหรับตัวรับข้อมูล (receiver buffer register) เป็นรีจิสเตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนดคือ หมายเลขแอดเดรส 3F8 ขณะที่ DLAB = 0 หากซีพียูอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

รีจิสเตอร์โฮลดิ้งสำหรับตัวส่งข้อมูล (transmitter holding register) เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลรีจิสเตอร์นี้จะรับข้อมูลจากซีพียู โดยที่กำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ DLAB = 0 ข้อมูลของซีพียูที่เอาต์พุตมาที่พอร์ตนี้ก็เพื่อจะส่งค่อออกไปยังสายสารข้อมูล



บทที่ 6

การออกแบบและผลการทดลอง

6.1 แนวทางการออกแบบ

เนื่องจากโครงการเดิมซึ่งเป็นเครื่องต้นแบบของระบบเก็บข้อมูลปริมาณก๊าซ (Data Logger for air pollution) ซึ่งจะมีส่วนใช้งานหลักๆอยู่ 3 ส่วนคือ วงจรก๊าซเซนเซอร์ วงจรแปลงอะนาลอกเป็นดิจิตอล (A/D) และวงจรส่วนคอนโทรลเลอร์ 8031 ซึ่งในส่วนนี้จะมี RAM เก็บข้อได้ 24 K และในการติดต่อกับไมโครคอมพิวเตอร์ผ่าน RS232-C จะแสดงผลของข้อมูลออกมาในรูปเชิงเส้น แต่เนื่องจากยังไม่สมบูรณ์จึงทำการแสดงผลเพียงการ Plot จุดที่เวลาต่างๆ ซึ่งเวลาที่เราจะตรวจวัดข้อมูลจึงทำได้ค่อนข้างลำบาก ทางผู้จัดทำจึงได้พยายามที่จะปรับปรุงบางจุดให้ดีขึ้น โดยมีแนวทางการออกแบบในส่วนต่างๆดังต่อไปนี้

6.1.1 การออกแบบในส่วนของชุดไมโครคอนโทรลเลอร์

ในการออกแบบชุดไมโครคอนโทรลเลอร์จะมีส่วนสำคัญที่จะต้องพิจารณาอยู่ด้วยกัน 2 ส่วนคือ

1) ส่วนของอุปกรณ์ฮาร์ดแวร์ (Hard ware) ซึ่งจะประกอบไปด้วย ไมโครคอนโทรลเลอร์เบอร์ 8051 , ชุดหน่วยความจำสำรองและชุดการแสดงผลโดยใช้ LCD ซึ่งได้แสดงไว้แล้วในผังรูป 6.1 จะเห็นได้ว่าในส่วนของวงจรมันยังมีส่วนประกอบของวงจรอีกพอสมควรเช่นชุด Real Time Clock , ชุดของพอร์ตอนุกรม RS 232 (RS232 Serial Port) จากโครงสร้างของไมโครคอนโทรลเลอร์ 8031 จะมีพอร์ตใช้งานอยู่ 3 พอร์ตคือพอร์ตศูนย์ (Port 0 , Po.0- Po.7 , พอร์ตหนึ่ง (Port 1, P1.0-P 1.7) และพอร์ตสอง (Port 2, P2.0-P2.7) ซึ่งจะมีหน้าที่ใช้งานแตกต่างกันออกไป โดย Port 0 และ Port 2 จะเป็นส่วนของข้อมูลซึ่งส่วนของหน่วยความจำ จะต่ออยู่กับพอร์ตสองพอร์ตนี้คือ เมื่อเราใช้หน่วยความจำที่มีค่า 32 กิโลไบต์ (KB) เราจะต้องใช้ตำแหน่งทั้งหมด 14 เส้นส่วนการต่อเพื่อการขยายพอร์ตเราจะใช้สายสัญญาณข้อมูลทั้งหมด 8 เส้น ต่อเข้าที่อินพุตข้อมูลของไอซีเบอร์ 8255 เพื่อจะทำการต่อเชื่อมกับชุดของส่วนการแปลงสัญญาณอะนาลอกเป็นดิจิตอล (A/D) ซึ่งจะต้องใช้จำนวนเส้นของสัญญาณ 12 เส้น ส่วนสายสัญญาณที่เหลือของพอร์ต 8255 เราจะนำไปต่อกับชุดคีย์บอร์ด (Key

Board) ในการออกแบบฮาร์ดแวร์ของส่วนของไมโครคอนโทรลเลอร์เราจะแยกพิจารณาออกเป็นหัวข้อหลักๆ ได้ดังต่อไปนี้

1.1 การเพิ่มประสิทธิภาพในการเก็บข้อมูล

จากแนวคิดในการที่ต้องการเก็บข้อมูลในลักษณะของ Real Time ซึ่งเราจำเป็นต้องเก็บข้อมูลอยู่ตลอดเวลาพร้อมทั้งแสดงผลที่จอ LCD ด้วย ดังนั้นในการออกแบบหน่วยความจำ จึงต้องมีขนาดพอเพียงที่จะเก็บข้อมูลต่อเนื่องอยู่ได้อย่างน้อย 24 ชั่วโมง ซึ่งจากที่ทดลองคำนวณคร่าวๆ กับปริมาณข้อมูลที่ต้องการเก็บในข้อมูลใน 24 ชั่วโมง โดยมีขนาดของ Sampling rate ที่ 10 วินาที ดังนั้นในการเก็บข้อมูลใน 24 ชั่วโมง (86400 วินาที) จะใช้หน่วยความจำประมาณ 13 KB ซึ่งถ้าเราต้องการให้อัตราการ Sampling เร็วกว่านี้ ก็จะต้องใช้พื้นที่หน่วยความจำเพิ่มขึ้น ซึ่งในโครงการนี้ในขั้นต้นได้เขียนโปรแกรมตั้งค่าการ Sampling ไว้ 3 ค่า คือที่ 10 วินาที 30 วินาที และ 1 นาที และในการเลือกใช้หน่วยความจำจะใช้ขนาด 32 KB ซึ่งเป็นค่าสูงสุดที่จะทำได้เนื่องจากการติดขัดเรื่องของสัญญาณที่จะนำมาทำการดีโค๊ด เพื่อเลือกตำแหน่งของหน่วยความจำ (Data Address)

1.2 การแสดงผลของข้อมูล

ในการที่จะแสดงผลของ LCD เราจะมีการแสดงค่าของปริมาณก๊าซออกมาในรูปของ PPM (Part Per Million) พร้อมกับแสดงค่าเวลาจริงในการตรวจวัดไปพร้อมกัน เพื่อจะสะดวกในการตรวจวัด โดยในโครงการนี้จะพยายามที่จะเขียนโปรแกรมที่เป็นลักษณะวนรูปในการทำงานของ Keyboard เพื่อให้มีปุ่มการใช้งานที่น้อยที่สุด

1.3 การลดกำลังที่ใช้ในวงจร

เนื่องจากโครงการนี้คำนึงถึงการนำไปใช้งานในการเก็บข้อมูลในสถานที่ต่างๆ ได้จริง เราจึงพยายามที่จะออกแบบให้สามารถที่จะออกแบบให้มีวงจรขนาดเล็กและกินกำลังไฟต่ำ โดยใช้ไฟเลี้ยงจากแบตเตอรี่รถยนต์ เพราะในการเก็บข้อมูลในหลายๆ อาจไม่สะดวกในการหาแหล่งจ่ายจากภายนอก (AC 220) เราจึงมีแนวคิดที่จะลดกำลังไฟที่ใช้ในส่วนต่างๆ ดังนี้

1) ไมโครคอนโทรลเลอร์ ในส่วนนี้ค่อนข้างจะเป็นส่วนที่กินกำลังไฟมากที่สุด เพราะต้องมีส่วนประกอบมาก เช่น ไอซีไมโครคอนโทรลเลอร์ เบอร์ 8031 ส่วนของ Latch และ ดีโค๊ดเดอร์ ซึ่งไอซีในส่วนนี้เราจะสามารถเลือกใช้เป็นไอซีชนิด Cmos ซึ่งกินกำลังไฟต่ำ แต่ในส่วนของไอซีเบอร์ 8255 ซึ่งเป็นไอซีที่ช่วยขยายพอร์ตให้กับไอซี 8031 เพื่อต่อกับชุด A/D และชุดของ Display และ

Scan Key ซึ่งไอซีตัวนี้ค่อนข้างจะกินกระแสสูง (ประมาณ 120 mA ที่ไฟ 5 V) เราจึงต้องพยายามใช้พอร์ตที่มีอยู่ให้คุ้มค่าที่สุด เพื่อจะสามารถใช้ไอซี 8255 เพียงตัวเดียว

2) ชุด LCD Display เนื่องจากชุด LCD จะประกอบกันมาเป็น Module จึงลำบากที่จะปรับปรุงกระแสใช้งาน แต่จากคุณสมบัติของ LCD module นี้จะกินกระแสค่อนข้างต่ำมาก ประมาณ 3 mA ที่ 5V จึงสามารถที่จะยอมรับได้

2) ส่วนของซอฟต์แวร์ หรือโปรแกรมควบคุม (Software or Controller Programing)

ในการเขียนโปรแกรมเพื่อควบคุมการทำงานของส่วนไมโครคอนโทรลเลอร์นี้ เราจำเป็นจะต้องใช้ภาษาแอสเซมบลี (Assembly) เฉพาะสำหรับไมโครคอนโทรลเลอร์บอร์ดนี้ ซึ่งในการทำงานเราจะต้องเขียนโปรแกรมควบคุมส่วนหลัก ๆ ดังนี้คือ

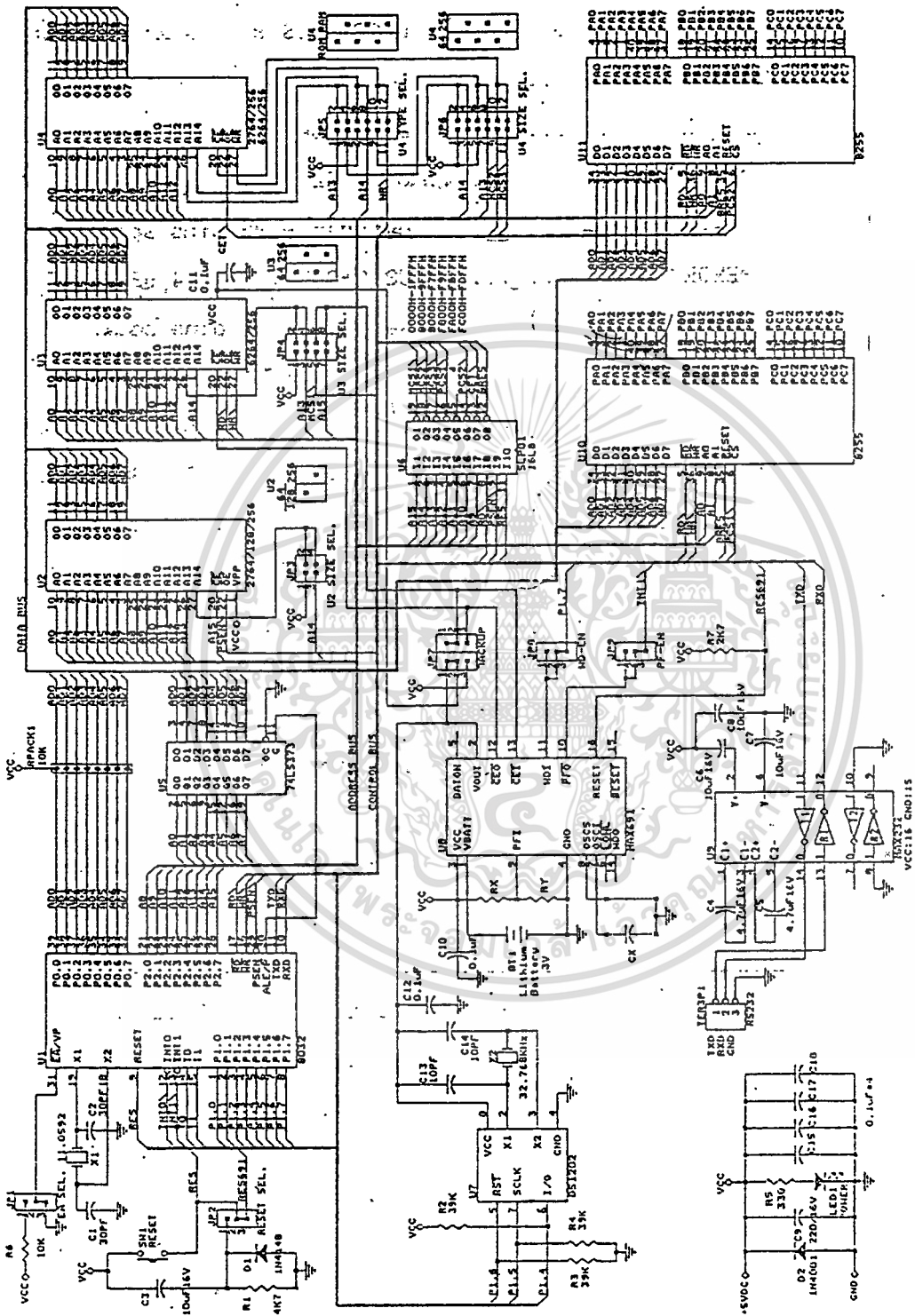
- 2.1 การเก็บข้อมูลและดึงข้อมูลออกจากหน่วยความจำ
- 2.2 การกำหนดตำแหน่งของหน่วยความจำ และตำแหน่งของพอร์ตที่ใช้ติดต่อกับภายนอกเช่น พอร์ต 8255, พอร์ต LCD
- 2.3 การติดต่อกับชุดไมโครคอมพิวเตอร์
- 2.4 การแสดงค่าของเวลาจริง (Real time clock)

ในการเขียนโปรแกรมควบคุมสิ่งที่จำเป็นอันดับแรกคือ การเขียนโฟลวชาต (Flow chart) ซึ่งโฟลวชาตในส่วนของชุดไมโครคอนโทรลเลอร์จะแสดงได้ดังรูปที่ 6.2 และรูป 6.3

6.1.2 การออกแบบชุดก๊าซเซ็นเซอร์

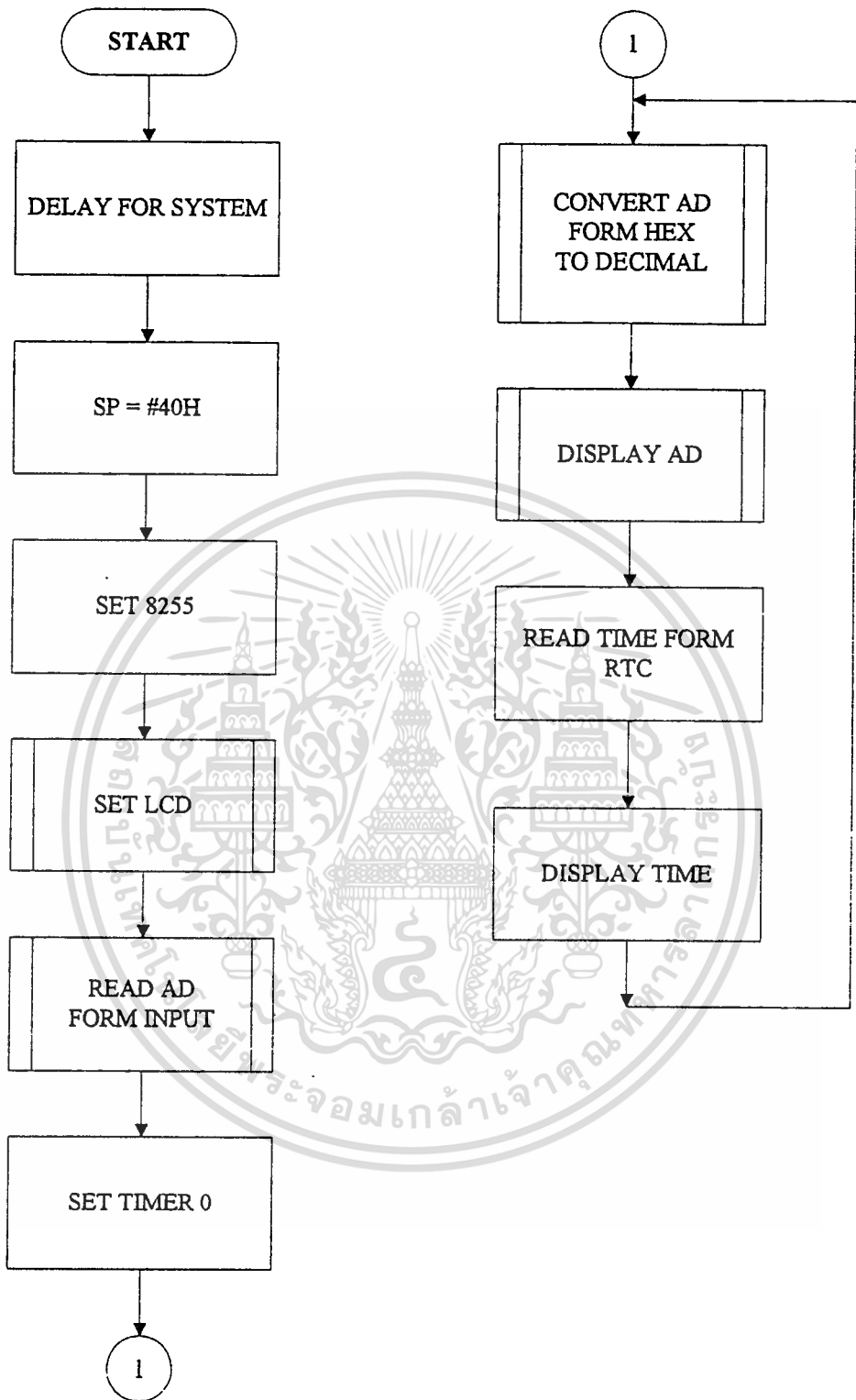
ดังที่กล่าวไว้คร่าว ๆ แล้วเกี่ยวกับหัวก๊าซเซ็นเซอร์เบอร์ NAP-11AS ซึ่งเป็น Indoor Odor Sensor ในการใช้งานจริง ๆ นั้นเราจำเป็นที่จะต้องมีส่วนประกอบอีก ซึ่งจะแสดงได้จากรูปที่ 6.4 เราเห็นว่าจากวงจรค่าของ R1 คือส่วนของหัวก๊าซเซ็นเซอร์

แนวคิดในการออกแบบวงจรนี้ คือว่าเมื่อเริ่มใช้งานเราจำเป็นที่ต้องอุ่นหัวเซ็นเซอร์ด้วยแรงดันประมาณ 5 โวลต์ ก่อนเพื่อจะช่วยให้หัวก๊าซสะอาด พร้อมทั้งจะรับค่าของก๊าซใหม่ได้ ซึ่งจากรูป 6.4 ก็คือส่วนของ Q_2 และ U_1 จะต่ออยู่ในลักษณะวงจรจ่ายไฟ (Regulator) ซึ่งเวลาในการอุ่นหัวเซ็นเซอร์ก๊าซจะขึ้นอยู่กับค่าช่วงเวลาคงที่ของชุดโมโนสเตเบิล (Monostable) ของไอซีเบอร์ NE 555 สามารถคำนวณค่าได้จากค่าของ R_2 และ C_3 ตามสูตร



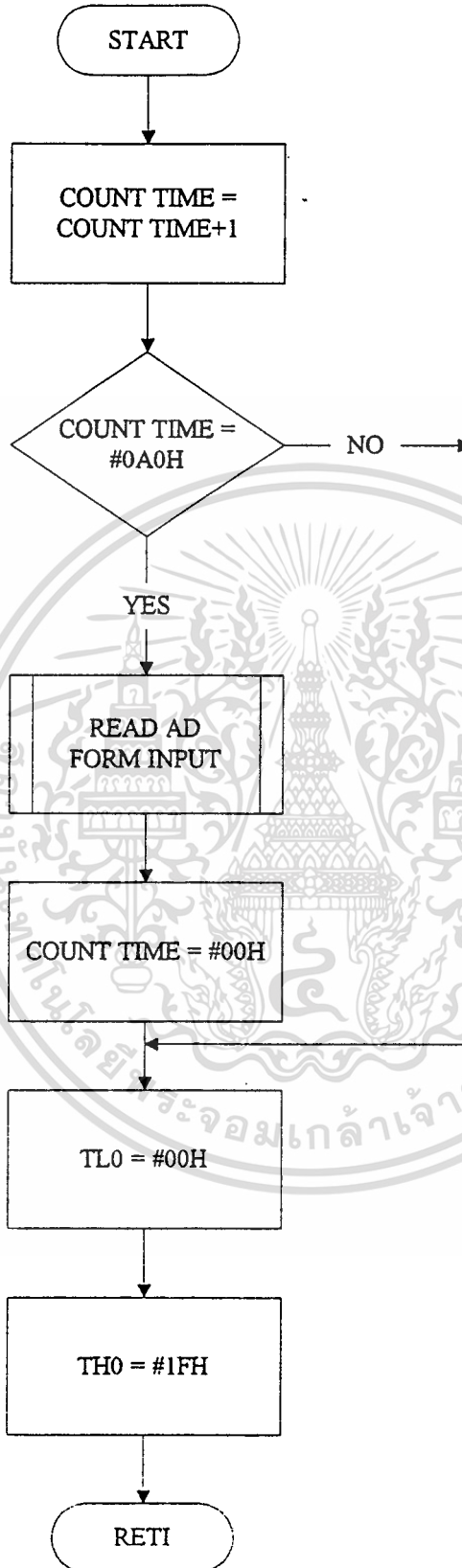
รูปที่ 8.1 8031 system decode and memory

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



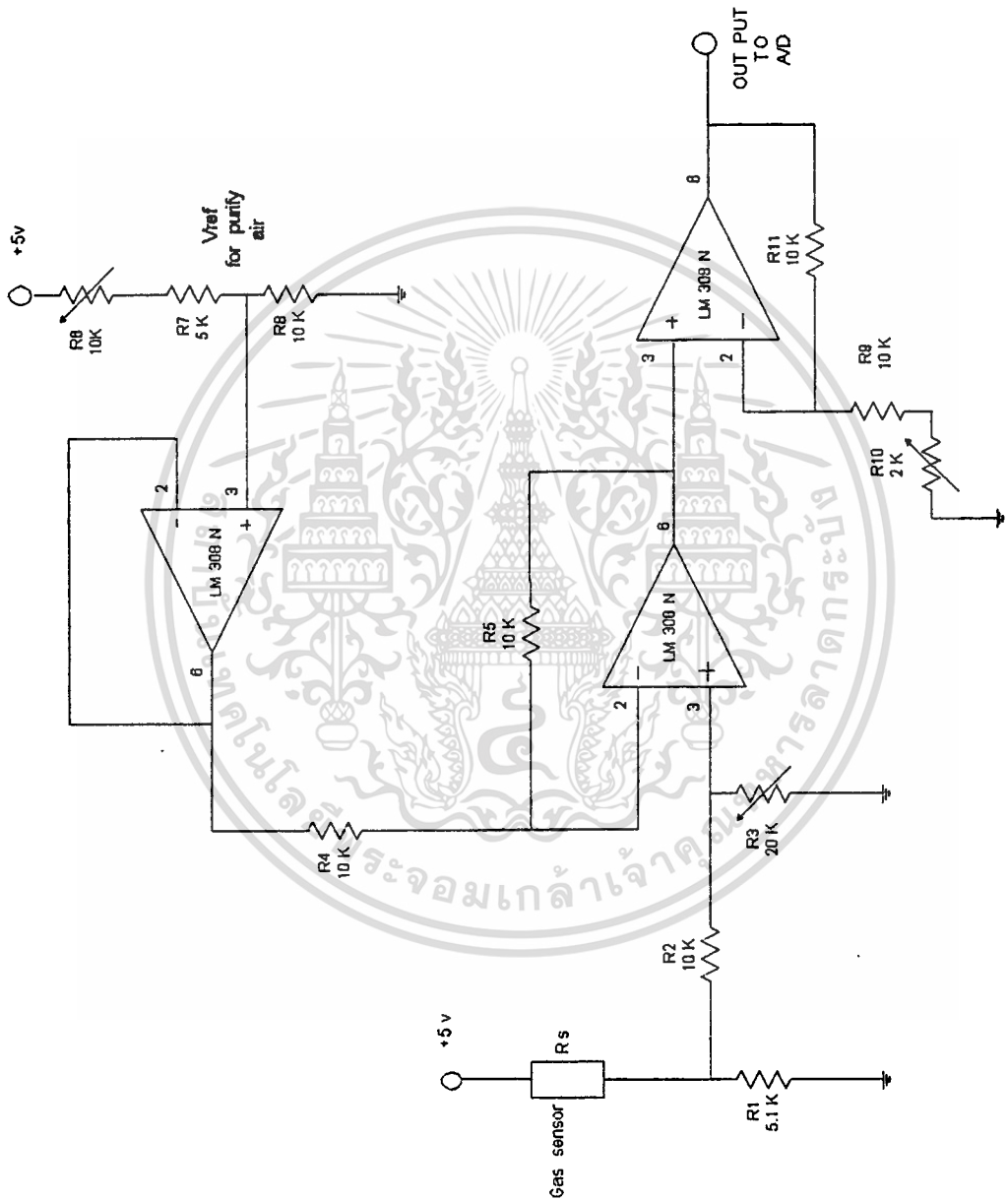
รูปที่ 6.2 Flowchart ของชุดแสดงผล A/D

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



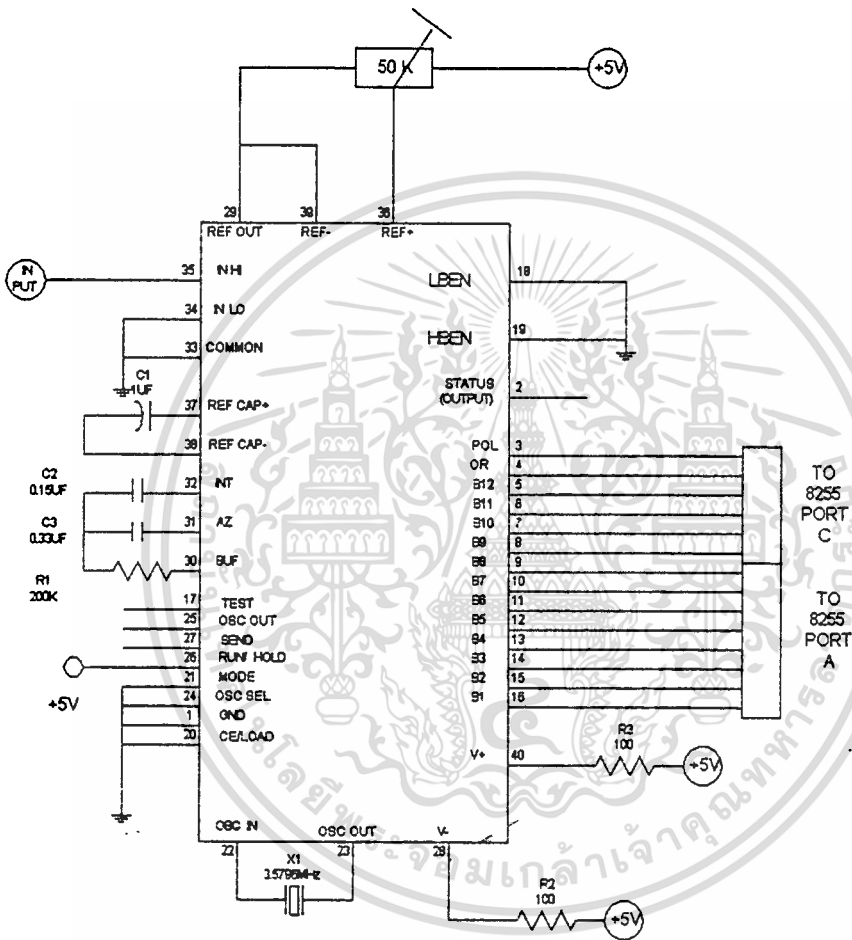
รูป 6.3 Flowchart ของชุดแสดงผล RTC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 6.4 วงจร Gas Sensor Circuit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.5 วงจร A/D Converter 7109

ค่าเวลาคงที่ (Time Constant) = $1.1 R_2 C_3$

เมื่อเราอุ่นหัวก๊าซเซ็นเซอร์แล้ว ต่อไปก็จะเริ่มตรวจวัดปริมาณของก๊าซที่เข้ามาที่หัวเซ็นเซอร์ จากคุณสมบัติของหัวก๊าซเซ็นเซอร์ซึ่งจะมีค่าความต้านทานที่เปลี่ยนไปตามปริมาณและชนิดของก๊าซที่เข้ามา ดังนั้น จะเห็นว่าถ้าค่าของปริมาณก๊าซ หรือชนิดของก๊าซเปลี่ยนไป แรงเคลื่อนที่เข้าพุท (Output) ของก๊าซเซ็นเซอร์จะแปรเปลี่ยนไปด้วย ซึ่งเราจะดูคุณสมบัติที่เปลี่ยนไปดัง ตารางที่ 2.2

จากตารางที่ 2.2 เราจะเห็นว่าในการออกแบบชุดเซ็นเซอร์ของเรานั้นต้องมีการเลือกชนิดของก๊าซที่จะเซ็นเซอร์เพื่อจะสามารถกำหนดหาสมการที่จะใช้ในการตั้งค่าปริมาณของก๊าซที่ออกมาในหน่วยของ “หนึ่งต่อล้านส่วน” (Part per Million ; PPM)

ในการทดลองนี้ เราจะทดสอบโดยใช้ควันบุหรี่ (Cigarettes smoke) เป็นก๊าซทดสอบโดยดูจากเส้นกราฟในตารางที่ 2.2 ดังตัวอย่างเช่น

ที่ gas = 7 ppm. จะได้ค่าความต้านทานประมาณ 12 กิโลโอห์ม ($K\Omega$)

ที่ gas = 30 ppm. จะได้ค่าความต้านทานประมาณ 5 กิโลโอห์ม ($K\Omega$)

ที่ gas = 100 ppm. จะได้ค่าความต้านทานประมาณ 0.9 กิโลโอห์ม ($K\Omega$)

จะเห็นว่า เมื่อปริมาณของก๊าซเพิ่มขึ้น ค่าของความต้านทานจะต่ำลง ดังนั้นในการที่เราจะทดลองหาค่าปริมาณก๊าซที่เข้ามา เราจึงต้องปรับส่วนของแรงดันอ้างอิง (referent Voltage) ให้สัมพันธ์กับแรงดันที่ออกจากส่วนหัวก๊าซเซ็นเซอร์ โดยนำค่าของแรงดันทั้งสองมาเข้าวงจร ดิฟเฟอเรนเชียลแอมพลิฟายเออร์ (Differential Amplifier) ซึ่งในที่นี้เราจะออกแบบในขั้นต้นให้เป็นวงจรบัฟเฟอร์ (Buffer) ธรรมดา ก่อน ซึ่งค่าแรงเคลื่อนเข้าพุทนี้จะออกไปเข้าส่วนของวงจรแปลงแรงดันเป็นดิจิตอลอีกครั้งหนึ่งต่อไป

6.1.3 การออกแบบ ชุดการแปลงสัญญาณอะนาล็อกเป็นดิจิตอล (A/D)

จากที่ได้ทราบถึงคุณสมบัติของไอซี 7109 ซึ่งได้กล่าวไว้ที่บทที่ 1 และบทที่ 3 แล้วนั้น ต่อจากนั้นจะมาถึงขั้นการออกแบบเพื่อนำไปใช้งานโดยดูจากตัวอย่างของวงจรทดสอบที่แสดงไว้ในคาต้าชีท (Data sheet) ของวิทยานิพนธ์นี้แล้ว

ในขั้นแรกของการออกแบบเราจำเป็นต้องทราบว่าไอซีตัวนี้จะต้องใช้แหล่งจ่ายไฟสองแหล่ง คือ บวก 5 โวลต์ และ ลบ 5 โวลต์ และเพื่อให้เกิดการประหยัดไฟเราจะลดส่วนของบัฟเฟอร์ออกไป. เพราะเมื่อข้อมูลออกมาครั้งละ 8 bit ซึ่งจะประกอบด้วยบิตข้อมูลค่า 8 บิต และสูง 4 บิต เหมือนของวิทยานิพนธ์ของเดิม จะต้องใช้บัฟเฟอร์ (Buffer) และแลตช์ (latch) เพื่อดึงข้อมูลให้ปรากฏอยู่ชั่วคราว

เพื่อให้ไมโครคอนโทรลเลอร์อ่านทัน โดยแนวคิดในการออกแบบก็จะต้องให้มีข้อมูลออกมาครั้งเดียว 12 บิตเลข โดยค่าของข้อมูล 12 บิตนี้ จะผ่านเข้าไปที่พอร์ตของ 8255 เพื่อให้ไมโครคอนโทรลเลอร์อ่านค่าได้สะดวก และรวดเร็วขึ้น ซึ่งเมื่อดูที่คาส์ชิทจะพบว่าจะต้องมีการตั้งค่าของ LBEN และ HBEN ให้ทำงานในช่วงแอกทีฟต่ำ (Active low) จึงจะทำให้มีค่าของสัญญาณดิจิตอล 12 บิต ที่ผ่านการแปลงแล้วปรากฏออกมาได้

นอกจากนี้ในการออกแบบ A/D ชุดนี้เราจำเป็นต้องหาค่าความต้านทานอินทิเกรตติ้ง (Integrating Resistance ; Rint) และค่าความจุอินทิเกรตติ้ง (Integrating Capacitor ; Cint) ซึ่งค่าทั้งสองสามารถหาได้จากสูตรดังนี้

$$R_{int} = \frac{\text{full scale voltage}}{20 \mu A}$$

ซึ่งเมื่อกำหนดให้ค่า full scale voltage = 4.096 โวลต์เราจะได้ค่าของ

$$R_{int} = \frac{4}{20 \times 10^{-6}} = 200 \text{ กิโลโอห์ม } \#$$

$$C_{int} = \frac{(2048 \times \text{clock period}) 20}{\text{integrator output Voltage Swing}}$$

ซึ่งจากคาส์ชิท เมื่อกำหนดให้ค่าแหล่งจ่ายอยู่ในช่วง ± 5 โวลต์ และความถี่ของสัญญาณนาฬิกาเท่ากับ 61.72 กิโลเฮิรต์ เราจะหาค่าของ C_{int} เท่ากับ 0.15 ไมโครฟารัด (μF) และ นอกจากนี้ไอซีตัวนี้ยังมีระบบป้องกันสัญญาณรบกวน (Noise) โดยจะใช้คาปาซิเตอร์ เป็นตัวลดทอนสัญญาณรบกวน ซึ่งจะเรียกว่า “Auto - Zero capacitor” ; CAZ คาปาซิเตอร์ตัวนี้ ควรมีค่าความจุมาก ๆ แต่เนื่องจากผลของ R-C time constant เราจึงต้องมีช่วงจำกัดในการใช้งานโดยใช้สูตรการคิดง่าย ๆ คือ

$$CAZ = 2 C_{int}$$

ดังนั้นเราจะกำหนดค่า CAZ ได้เท่ากับ 0.33 ไมโครฟารัด ซึ่งเมื่อเราออกแบบค่าของอุปกรณ์ประกอบไอซีตัวนี้ได้แล้ว ก็จะได้วงจรที่ใช้งานดังรูปที่ 6.5 จากวงจรใช้งานจะเห็นว่ามีส่วนที่จำเป็นต้องทราบอีก เช่น ที่ขาอินพุตคือ ขา 35 จำเป็นจะต้องมีความต้านทานค่าสูง ๆ ต่ออนุกรมอยู่เพื่อ

วัตถุประสงค์ในการช่วยการลดทอนของขนาดแรงดันที่เข้ามา ซึ่งจากตัวอย่างวงจรทดสอบในคาต้า ไอซีตัวนี้จะใช้ค่าเท่ากับ 1 เมกกะโอมห์ และสิ่งที่ต้องระวังอีกอย่างก็คือค่าของแรงดันอ้างอิง (U_{ref}) ที่ป้อนเข้าขา 36 และ 39 ของไอซี ซึ่งเกิดจากการแบ่งแรงดันมาจากความต้านทาน R_4 ซึ่งเป็นความต้านทานปรับค่าละเอียด จะต้องมีความมากกว่าแรงดันเข้าที่พุดสูงสุด (U_{max}) เท่ากับ $U_{max}/2$ หรือก็คือ $U_{max} \geq 2 U_{ref}$ ไม่เช่นนั้นจะทำให้ข้อมูลจะเกิดการผิดพลาดจนตรวจจับสัญญาณไม่ได้ ซึ่งจะเรียกว่าเกิด “Over flow” หรือ “Over run” ซึ่งเกิดจากการทดลองดู จะใช้ค่าของความต้านทานในช่วง 10 กิโลโอมห์ ก็จะสามารถใช้งานได้

6.2 ผลการทดลองการแปลงอะนาลอกเป็นดิจิตอล (Analog to Digital ; A/D)

ในการทดลอง A/D เบอร์ 7109 นี้ โดยทดสอบป้อนไฟ V_{in} เข้าไปซึ่งเป็นไฟตรงโดยจะตั้งค่าแรงดัน Full scale ไว้ที่ 4.096 Volts โดยตั้งค่าแรงดันอ้างอิงไว้ที่ประมาณ 2.1 Volts [คุณสมบัติของไอซีตัวนี้ได้จาก data sheets ในภาคผนวก]

จากสัญญาณ 12 Bit เราสามารถแยกระดับสัญญาณได้ $= 2^{12} = 4096$ Increments

ผลการทดลองของวงจร A/D

ST	POL	OR	V _{in}	B ₁₂	B ₁₁	B ₁₀	B ₉	B ₈	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1	1	1	1	0	1	0
1	1	0	2	0	1	1	1	1	1	0	1	1	0	0	1
1	1	0	3	1	1	0	0	1	1	0	0	1	0	0	1
1	1	0	3.5	1	1	1	1	0	0	0	0	0	0	1	1
1	1	1	4	1	1	1	1	1	1	1	1	1	1	1	1

ตาราง 6.1 ผลการทดลองวงจรแปลงอนาลอกเป็นดิจิตอล

จากตารางผลการทดลองเราสามารถที่จะจัดหาค่า Voltage จากสัญญาณ Digital ที่ทดลองได้ซึ่งจะเป็นตัวชี้ให้เห็นว่า วงจร A/D ของเรามีความผิดพลาดมากหรือไม่

$$\text{จากค่า Full Scale resolution} = 4096 \text{ divisions}$$

$$\text{Full Scale voltage} = 4.096 \text{ Volts. [ตัวที่ 4 Volte]}$$

เราจะคำนวณหาค่า Analog อินพุตจากค่า Digital ที่วัดได้ดังนี้

$$1) \text{ เมื่อค่า Digital ที่วัดได้} = 010001111010$$

$$\text{ค่า division} = 1024 + 64 + 32 + 16 + 8 + 2 = 1146$$

$$\text{ดังนั้นค่า Analog I/P Voltage} = 4 \times \frac{1146}{4096} = 1.1 \text{ Volt}$$

[จะเห็นว่าค่าที่คำนวณมีค่าใกล้เคียงกับค่าแรงดันที่ป้อนเข้าไปคือ 1 Volt I/P]

$$2) \text{ เมื่อค่า Digital ที่วัดได้} = 011111011001$$

$$\begin{aligned} \text{ค่า division} &= 1024 + 512 + 256 + 128 + 64 + 16 + 8 + 1 \\ &= 2009 \end{aligned}$$

$$\text{ดังนั้นค่า Analog I/P Voltage} = 4 \times \frac{2009}{4096} \approx 2 \text{ Volt}$$

$$3) \text{ เมื่อค่า Digital ที่วัดได้} = 110011001001$$

$$\text{ค่า division} = 3271$$

$$\text{ดังนั้นค่า Analog I/P Voltage} = 4 \times \frac{3271}{4096} = 3.2 \text{ Volt}$$

$$4) \text{ เมื่อค่า Digital ที่วัดได้} = 111100000011$$

$$\text{ค่า division} = 3841$$

$$\text{ดังนั้นค่า Analog I/P Voltage} = 4 \times \frac{3841}{4096} = 3.75 \text{ Volt}$$

$$5) \text{ เมื่อค่า Digital ที่วัดได้} = 111111111111$$

$$\begin{aligned} \text{ค่า division} &= 4 \times \frac{4096}{4096} = 4 \text{ Volt} \end{aligned}$$

สรุป

จะเห็นว่าจากการทดลองป้อนค่าอินพุตโวลต์ตรง เข้าที่อินพุตและการคำนวณหาค่าโวลต์ตรง จากค่าคิจิตอล จะมีค่าใกล้เคียงกันมาก ซึ่งในการทดลองที่ค่าแรงดันต่ำ ๆ ในส่วนของบิทต่ำ ๆ จะมีสัญญาณไม่อยู่นิ่ง แต่ก็ไม่ค่อยมีผลมากนักเนื่องจากมีค่านัยสำคัญต่ำ และช่วงใช้งานของวงจรนี้จะขึ้นอยู่กับแรงดัน เอาท์พุทของส่วนตัวก๊าชเซนเซอร์ ซึ่งมีค่าช่วงแรงดันเอาท์พุทเท่ากับ 2 ถึง 4.6 โวลต์ ซึ่งจะต้องมีการใช้วงจรแบ่งแรงดันที่เอาท์พุทเพื่อไม่ให้เกินค่า Over Range (OR) ของไอซีตัวนี้ ซึ่งตั้งค่าใช้ที่ 4 โวลต์



6.3 ผลการทดลองชุดก๊าซเซ็นเซอร์ (Gas Sensor)

จากวงจรในรูป 6.4 ซึ่งเป็นวงจรที่เป็นต้นแบบในการออกแบบใช้งาน เราจะเห็นว่าจะประกอบไปด้วยส่วนหลักๆ 3 ส่วนคือ

1. ส่วนที่เป็นชุดจ่ายไฟเลี้ยงหัวก๊าซเซ็นเซอร์ โดยเมื่อเราดูจากคุณสมบัติการใช้งานของหัวเซ็นเซอร์ในบทที่ 2 จะพบว่าเราจะใช้ไฟเลี้ยงที่เป็นบวก 5 โวลต์
2. ส่วนของหัวเซ็นเซอร์ก๊าซ ซึ่งมีคุณสมบัติซึ่งมีค่าความต้านทานเปลี่ยนแปลงไปตามปริมาณและชนิดของก๊าซ
3. ส่วนของชุดออฟแอมป์ซึ่งจะเป็นตัวเปรียบเทียบค่าแรงดันในจุดเริ่มต้นขณะที่ยังไม่มีก๊าซเข้ามา ซึ่งตามแนวความคิดจะต้องให้ค่าแรงดันที่จุดนี้เป็น 0 โวลต์

จากวงจรทดสอบการทำงานของหัวก๊าซในรูปที่ 2.2 เราสามารถนำมาออกแบบในการทดสอบได้ โดยจำเป็นที่จะต้องมีการกำหนดมาตรฐานของการตรวจวัดดังนี้

1. เราต้องมีการทดสอบในบริเวณที่มีอาณาเขตคงที่เป็นมาตรฐานซึ่งจะขอเรียกว่า “Test Chamber.”
2. ต้องมีการกำหนดค่าแรงเคลื่อนที่ให้ไบอัส(Bias) แก่หัวเซ็นเซอร์ให้คงที่ตามมาตรฐาน (+5 volt)
3. จะต้องทำการวัดค่าแรงเคลื่อนเข้าที่ทุกหลังจากเริ่มจ่ายแรงเคลื่อนให้แก่หัวอ่านไปแล้ว 1 นาที (ในอากาศบริสุทธิ์) หรือรอให้มีการล้างหัวเซ็นเซอร์เป็นเวลา 3 นาทีในกรณีที่วัดในสภาวะอุณหภูมิห้องปกติ

สิ่งสำคัญที่ต้องพิจารณาขณะทดสอบ

1. ในการทำการทดลองวงจรก๊าซเซ็นเซอร์ขั้นแรกเราจะทำการแยกส่วนการทดสอบโดยแบ่งเป็นส่วนของหัวก๊าซเซ็นเซอร์กับส่วนของวงจรดีเฟอเรนเชียลแอมป์ขณะทำการทดลองตรวจวัดก๊าซเราจำเป็นต้องระวังเรื่องของอุณหภูมิที่เกิดขึ้นที่หัวเซ็นเซอร์ด้วย เพราะมีฉนวนเมื่อหัวก๊าซมีอุณหภูมิสูงค่าที่วัดออกมาได้จะผิดจากค่าจริง

2. ไม่ควรให้มีการเป่าก๊าซเข้าที่หัวเซ็นเซอร์โดยตรง ซึ่งจากข้อมูลของหัวก๊าซเซ็นเซอร์จะกำหนดค่าความเร็วของอากาศที่ผ่านชุดเซ็นเซอร์ควรน้อยกว่า 0.5 เมตรต่อวินาที

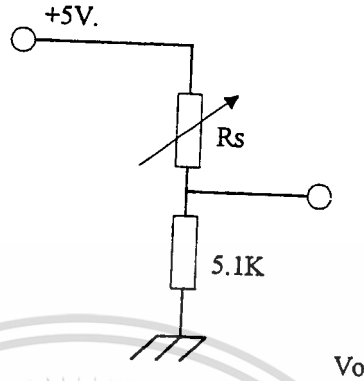
3. การวัดค่าแรงเคลื่อนที่ออกจากหัวก๊าซเซ็นเซอร์เราต้องใช้โวลต์มิเตอร์ที่มีค่าอิมพีแดนซ์สูงกว่า 1 เมกะโอมขึ้นไปตัวอย่างเช่น ดิจิตอลมัลติมิเตอร์เพราะมีฉนวนแล้วมิเตอร์ที่มีค่าอิมพีแดนซ์ต่ำจะแสดงค่าเป็นเสมือนโหลดอีกตัวของวงจรเมื่อนำเข้าไปตรวจวัดจะทำให้เป็นเหมือนวงจรขนานดังนั้นค่าของความต้านทานจะต่ำลงไป และค่าของแรงเคลื่อนที่วัดได้ก็จะผิดไป

โดยหลักการแล้วการวัดค่าแรงเคลื่อนที่ด้วยโวลต์มิเตอร์ที่มีค่าอิมพีแดนซ์ต่ำจะทำให้เกิดผลกระทบต่อการวัดค่าแรงเคลื่อนที่จริงของหัวเซ็นเซอร์ได้ ดังนั้นในการวัดค่าแรงเคลื่อนที่ด้วยโวลต์มิเตอร์ที่มีค่าอิมพีแดนซ์ต่ำ เราจะต้องคำนึงถึงค่าของโวลต์มิเตอร์ที่มีค่าอิมพีแดนซ์ต่ำนี้ด้วย

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการคำนวณโดยประมาณ

จากวงจรทดสอบหาปริมาณก๊าซเราจะคิดวงจรเสมือนของการวัดได้ดังรูป 6.6 ซึ่งก็คือวงจร Divider



รูปที่ 6.6 วงจรทดสอบก๊าซเซ็นเซอร์

ในการหาค่าแรงดันเข้าที่พุทจะขออ้างอิงค่าความต้านทานของหัวตรวจจับ (R_s) มาจากตาราง 2.2 เพื่อหาค่าของปริมาณความเข้มข้นก๊าซต่อไป

เมื่อคิดจากเส้นกราฟของควันนุหรี

- เมื่อค่า $R_s = 10$ กิโลโอมห์ และค่าความเข้มข้นก๊าซ = 9 ppm จะหาค่าแรงดันเข้าที่พุทได้เท่ากับ

$$V_o = V_i R_1 / (R_s + R_1) = 5 * (5.1 * 10^3) / (10 * 10^3 + (5.1 * 10^3)) = 1.688 \text{ โวลต์}$$

- $R_s = 5$ กิโลโอมห์ และค่าความเข้มข้นก๊าซ = 30 ppm

$$V_o = 5 * (5.1 * 10^3) / (5 * 10^3 + (5.1 * 10^3)) = 2.524 \text{ โวลต์}$$

- $R_s = 3$ กิโลโอมห์ จะได้ค่าความเข้มข้นก๊าซ = 70 ppm

$$V_o = 5 * (5.1 * 10^3) / (3 * 10^3 + (5.1 * 10^3)) = 3.148 \text{ โวลต์}$$

- $R_s = 2.2$ กิโลโอมห์ จะได้ค่าความเข้มข้นก๊าซ = 100 ppm

$$V_o = 5 * (5.1 * 10^3) / (2.2 * 10^3 + (5.1 * 10^3)) = 3.49 \text{ โวลต์}$$

6.4 ผลการทดลองชุดไมโครคอนโทรลเลอร์และชุด RTC

การสร้างฐานเวลาให้แก่ไมโครคอนโทรลเลอร์มี 2 วิธี

1. ใช้ชิปที่ทำหน้าที่ฐานเวลา
2. ใช้วงจรกำเนิดความถี่ที่ป้อนให้แก่เคาน์เตอร์ของไทม์เมอร์ 0 หรือ ไทม์เมอร์ 1 อินเทอร์พรีตซีฟิยูในช่วงเวลาตามที่กำหนด

ข้อเสียของไทม์เมอร์หรือเคาน์เตอร์ที่สร้างฐานเวลาแก่ซีฟิยู

1. ไม่เหมาะสมกับการเรียกใช้เวลาจริง
2. ช่วงเวลาที่นอนกเนือจากความสามารถของไทม์เมอร์หรือเคาน์เตอร์จะไม่สามารถใช้ไทม์เมอร์หรือเคาน์เตอร์ได้
3. ต้องมีการตั้งค่าใหม่ทุกครั้งเริ่มทำงาน
4. หากการทำงานของระบบผิดพลาดก็จะทำให้เวลาผิดพลาด

ข้อดีของไทม์เมอร์หรือเคาน์เตอร์ที่สร้างฐานเวลาแก่ซีฟิยู

1. ง่ายต่อการใช้งานเนื่องจากมีอยู่ใน MCS-51
2. ในงานที่ไม่ต้องการใช้ฐานเวลาที่เที่ยงตรงจะใช้ได้ดี
3. การต่อวงจรภายนอกมีน้อยหรือไม่มีก็ได้เมื่อกำหนดฐานเวลาด้วยโปรแกรม

การสร้างฐานเวลาให้แก่ MCS-51 ด้วยชิป

ข้อดี

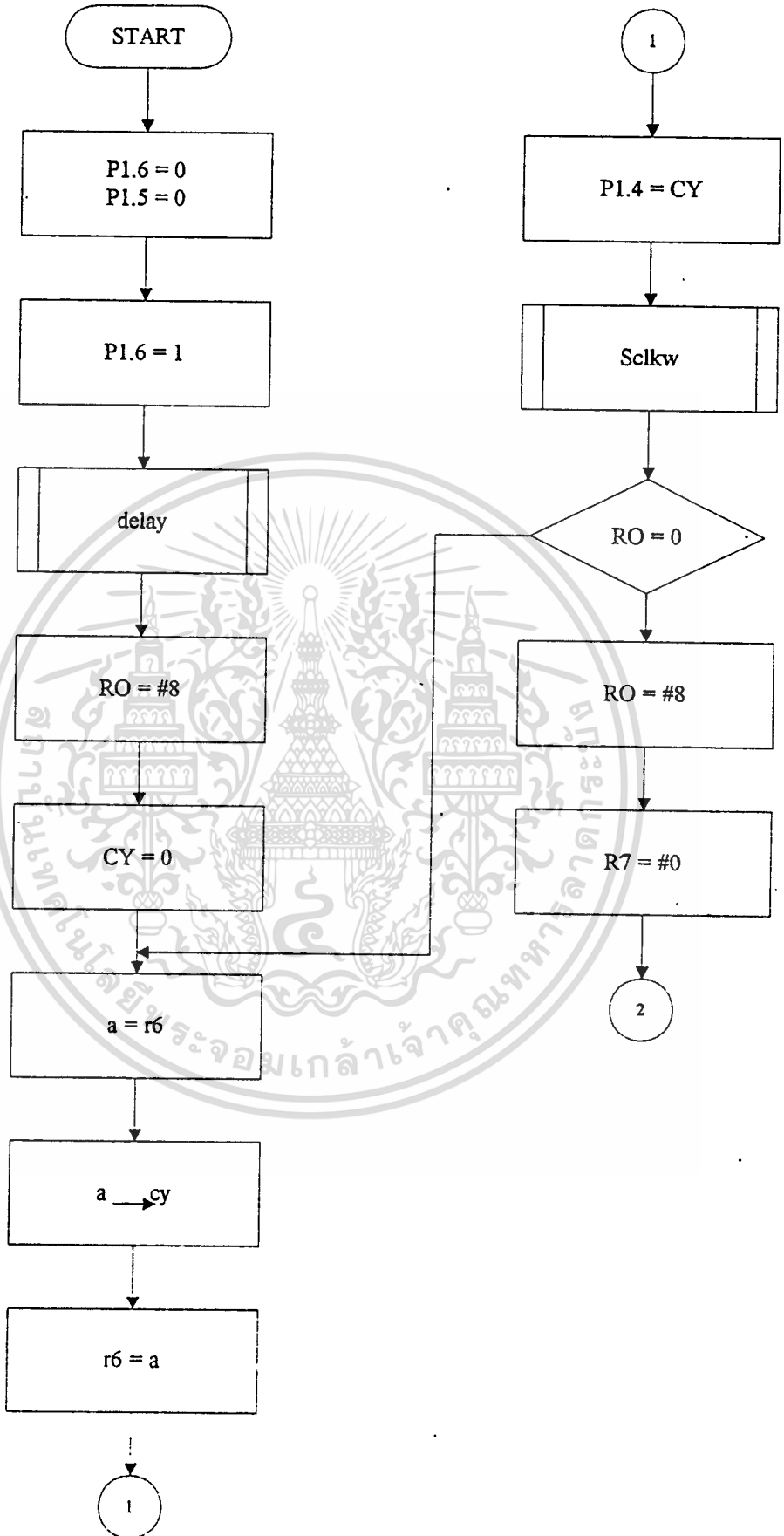
1. การทำงานด้านฐานเวลาไม่ขึ้นกับซีฟิยู
2. ตัวชิปจะทำงานตลอดโดยใช้แบตเตอรี่สำรอง
3. ความเที่ยงตรงในการทำงานสูง
4. ทำให้มีไทม์เมอร์หรือเคาน์เตอร์เหลือไว้ใช้ในงานอื่นๆ

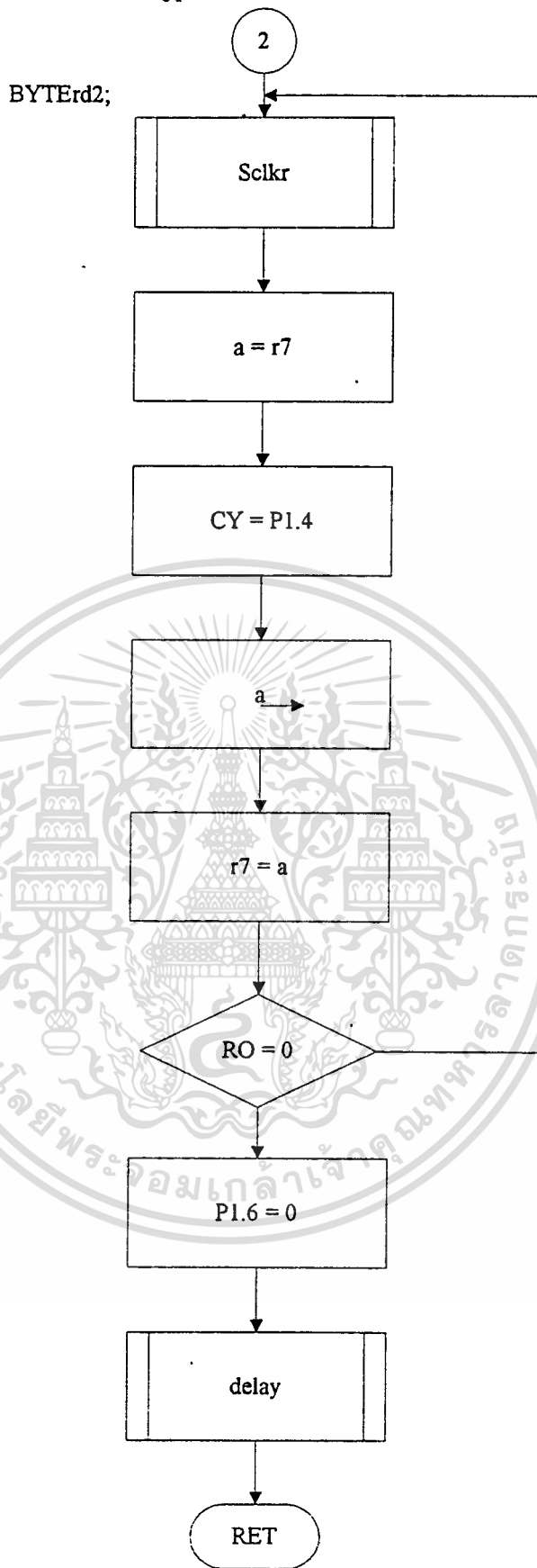
ข้อเสีย

1. มีความยุ่งยากในการต่อวงจร
2. มีขั้นตอนในการเขียนโปรแกรมมาก

byterd;

BYTERd1;



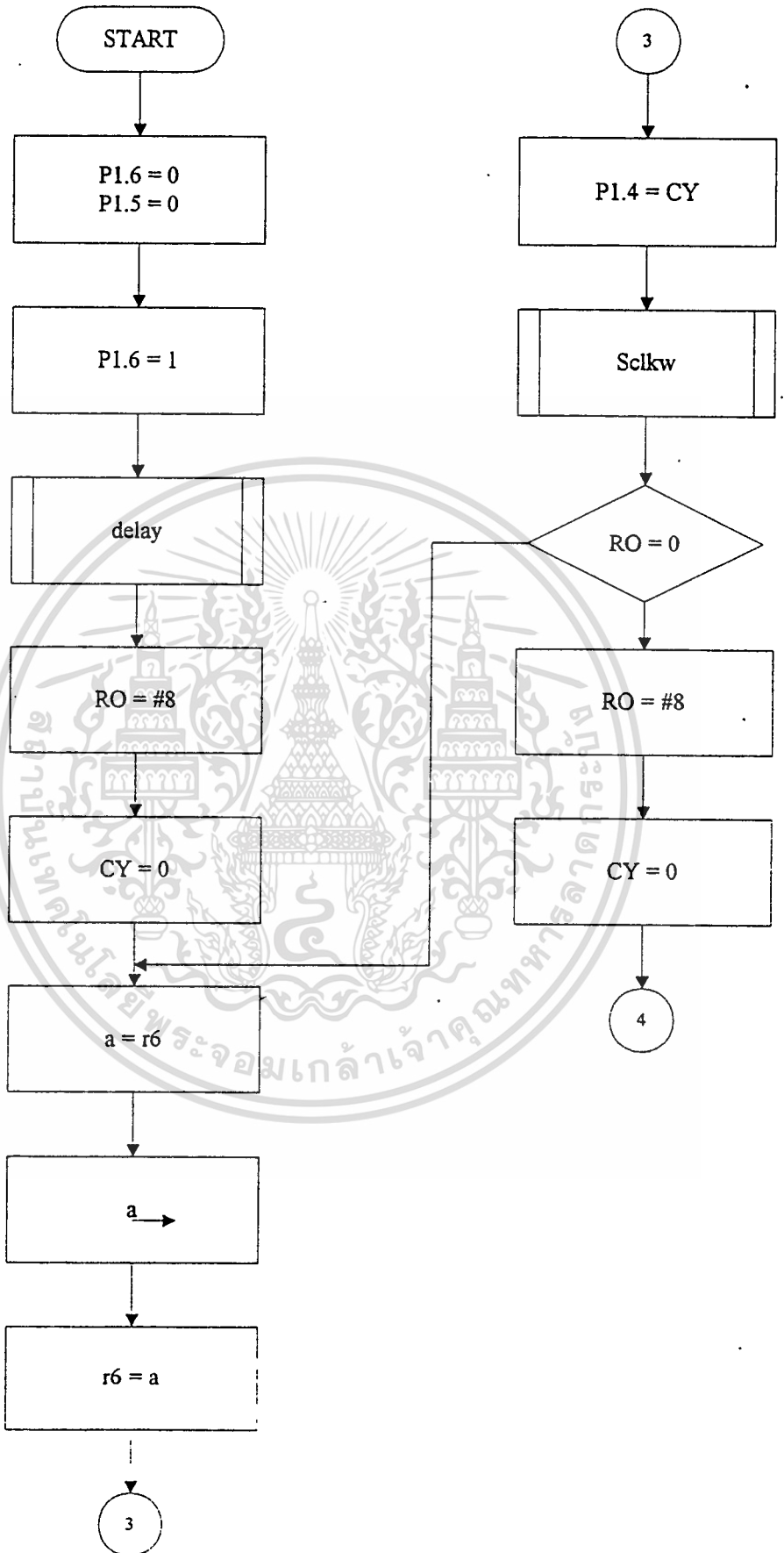


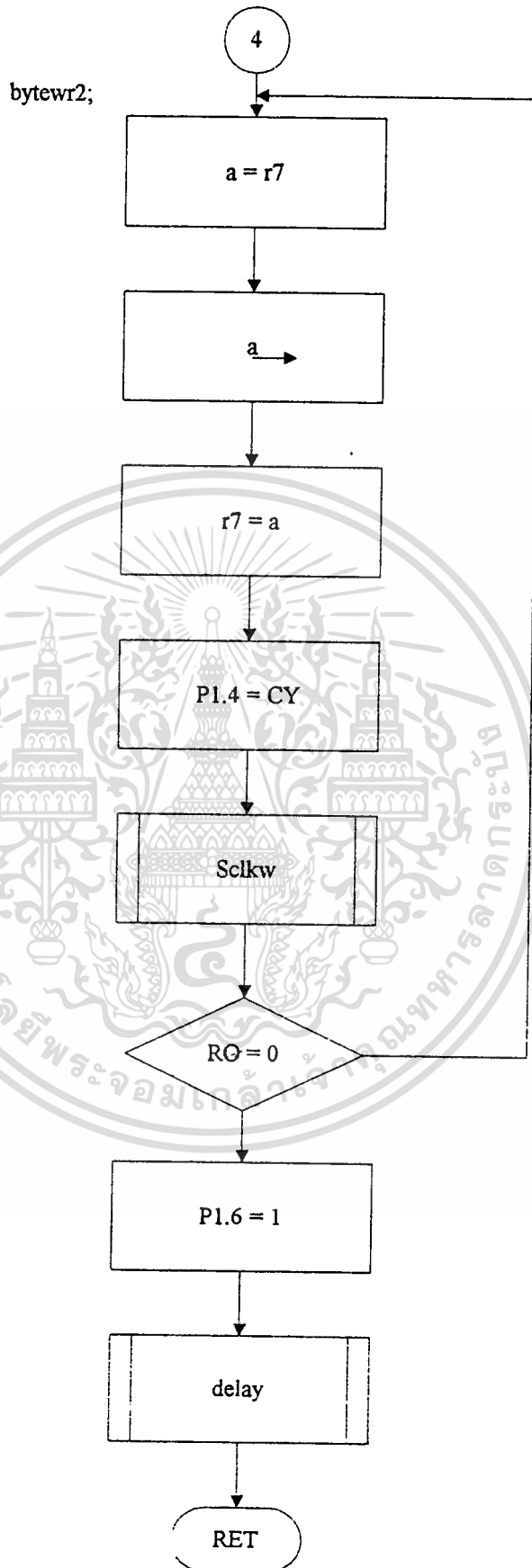
6.7(ข) โฟวาทส่วนที่สองของโปรแกรมชุด RTC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

bytewr;

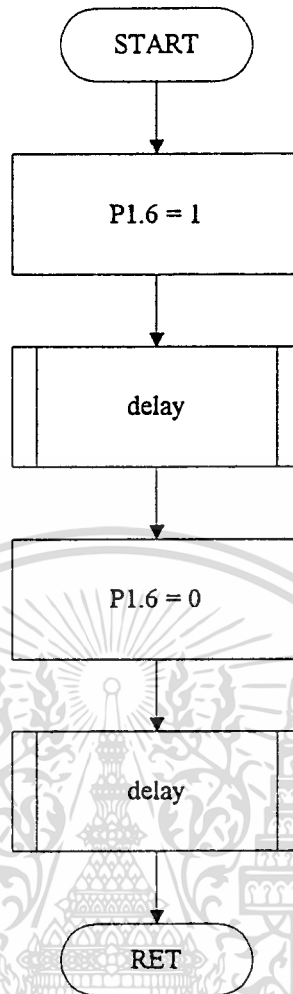
bytewr1;



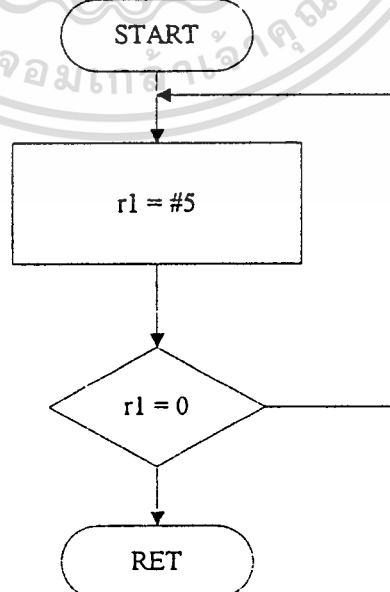


เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับอาจารย์ผู้สอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 6.7(ง) ไฟวาทส่วนที่สี่ของโปรแกรมชุด RTC
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sclr;



DELAY;



โปรแกรมอ่านและเขียนค่าระหว่าง MCS-51 กับ RTC DS1202

```

byterd:  clr          rst\
          clr          sclk
          setb         rst\
          lcall        delay
          mov          b,#8
          clr          c
byterd1:  mov          a,r6
          rcc          a
          mov          r6,a
          mov          i0,a
          lcall        sclkw
          djnz         b,byterd1
          mov          b,#8
          MOV          r7,0
byterd2:  lcall        sclkr
          mov          a,r7
          mov          c,i0
          rrc          a
          mov          r7,a
          djnz         b,byterd2
          clr          rst\
          lcall        delay
          ret
bytewr:   clr          rst\
          clr          sclk
          setb         rst\

```

```

        lcall    delay
        mov     b,#8
        clr     c
bytewr1: mov     a,r6
        rcc     a
        mov     r6,a
        mov     i0,c
        lcall   sclkw
        djnz    b,bytewr1
        mov     b,#8
        clr     c
bytewr2: mov     a,r7
        rcc     a
        mov     r7,a
        mov     i0,c
        lcall   sclkw
        djnz    b,bytewr2
        clr     rst
        lcall   delay
        ret
sclkw:  clr     sclk
        lcall   delay
        setb    sclk
        lcall   delay
        ret
sclkr:  setb    sclk
        lcall   delay
        clr     sclk

```

```

lcall    delay
ret
delay:   mov     r1,#5
        djnz    r1,$
        ret

```

ระบบไมโครคอนโทรลเลอร์ที่มีการอินเตอร์รัปต์ทุกๆ 1 วินาที โดยนำสัญญาณอินเตอร์รัปต์ความถี่ 1 เฮิร์ตซ์ เมื่อเกิดการอินเตอร์รัปต์เกิดขึ้นจึงจะย้ายตำแหน่งไปยังโปรแกรมการ up date เวลา

โปรแกรมตัวอย่างการ up date เวลา

```

:Second EQU 30H ;กำหนดตำแหน่งหน่วยความจำสำหรับเก็บค่า
วินาที
:Minute EQU 31H ;กำหนดตำแหน่งหน่วยความจำสำหรับเก็บค่านาที
:Hour EQU 32H ;กำหนดตำแหน่งหน่วยความจำสำหรับเก็บค่าชั่วโมง
ORG 000BH ;ตำแหน่งเริ่มต้นของโปรแกรมบริการอินเตอร์รัปต์
ของไทม์เมอร์ 0

```

PUSH PSW

PUSH ACC

วินาที

MOV A,Second ;นำค่าวินาทีมาคำนวณ

```

INC      A           ;เพิ่มค่าวินาทีอีกหนึ่ง
MOV      Second,A    ;นำค่าวินาทีไปเก็บไว้ในหน่วยความจำ
CJNZ     A,#60,FINISH ;ตรวจสอบว่าครบ 60 วินาทีหรือยัง
MOV      Second,#0   ;ถ้าครบ 60 วินาที ให้รีเซ็ตวินาทีจาก 0
                        ใหม่

```

นาที

```

MOV      A,Minute    ;นำค่านาทีมาคำนวณ
INC      A           ;เพิ่มค่านาทีอีกหนึ่ง
MOV      Minute,a     ;นำค่านาที ไปเก็บไว้ในหน่วยความจำ
CJNZ     A,#60,FINISH ;ตรวจสอบว่าครบ 60 นาทีหรือยัง
MOV      Minute,#0   ;ถ้าครบ 60 นาทีให้นับค่านาทีจาก 0 ใหม่

```

ชั่วโมง

```

MOV      A,Hour      ;นำค่าชั่วโมงมาคำนวณ
INC      A           ;เพิ่มค่าชั่วโมงอีกหนึ่ง
MOV      Hour,a      ;นำค่าชั่วโมงไปเก็บไว้ในหน่วยความจำ
CJNZ     A,#24,FINISH ;ตรวจสอบว่าครบ 24 หรือยัง
MOV      Hour,#0     ;ถ้าครบ 24 แล้วให้รีเซ็ตชั่วโมงจาก 0
                        ใหม่

```

RETURN FROM INTERRUPT ROUTINE

```

:Finish  POP          ACC
          POP          PSW
          RETI         ;ออกจากโปรแกรมบริการอินเตอร์รัปต์

```

6.5 ผลการทดลองเชื่อมต่อไมโครคอมพิวเตอร์

จากความจำเป็นที่จะต้องมีการส่งข้อมูลจากหน่วยวัดที่ควบคุมได้ ให้กับหน่วยประมวลผลที่มีความสามารถสูงกว่า เพื่อแสดงผลจากเซ็นเซอร์ ในรูปแบบที่มีประสิทธิภาพ

จากเหตุผลข้างต้น

1. ระบบการเชื่อมต่อจะแบ่งเป็นสองส่วนคือ

1.1 Sensor module (ควบคุมได้)

1.2 Pc Display

2. ข้อกำหนดที่ใช้ในการเชื่อมต่อ Sensor กับ Pc Display

2.1 ข้อมูลที่ใช้ในการตรวจสอบ

2.2 มาตรฐานระบบและตัวกลางที่ใช้ในการส่งข้อมูล เช่น RS-232 , RS-485

2.3 รูปแบบการส่งข้อมูลเช่น HALF DUPLEX , FULL DUPLEX

2.4 อัตราการส่งข้อมูล

2.5 Format ของข้อมูลขนาด 1 Byte ที่ใช้ในการส่ง

2.6 วิธีการตรวจสอบความถูกต้องของ ข้อมูลที่ได้รับ เปรียบเทียบกับ ข้อมูลด้านส่ง

2.7 Format ข้อมูลทั้งหมดโดยแบ่งตามหมวดหมู่

ลำดับขั้นตอนการทดลอง

1. ศึกษารูปแบบของการแสดงผลบนเครื่อง PC โดยศึกษาจากตำราเพื่อใช้ข้อมูลที่กำหนด

2. ศึกษาและกำหนดรูปแบบของข้อมูลที่จะส่ง

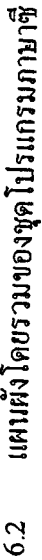
3. เขียน Program ของทั้งสองส่วน ให้ทำงานร่วมกัน ตามข้อกำหนดและข้อมูลที่กำหนด

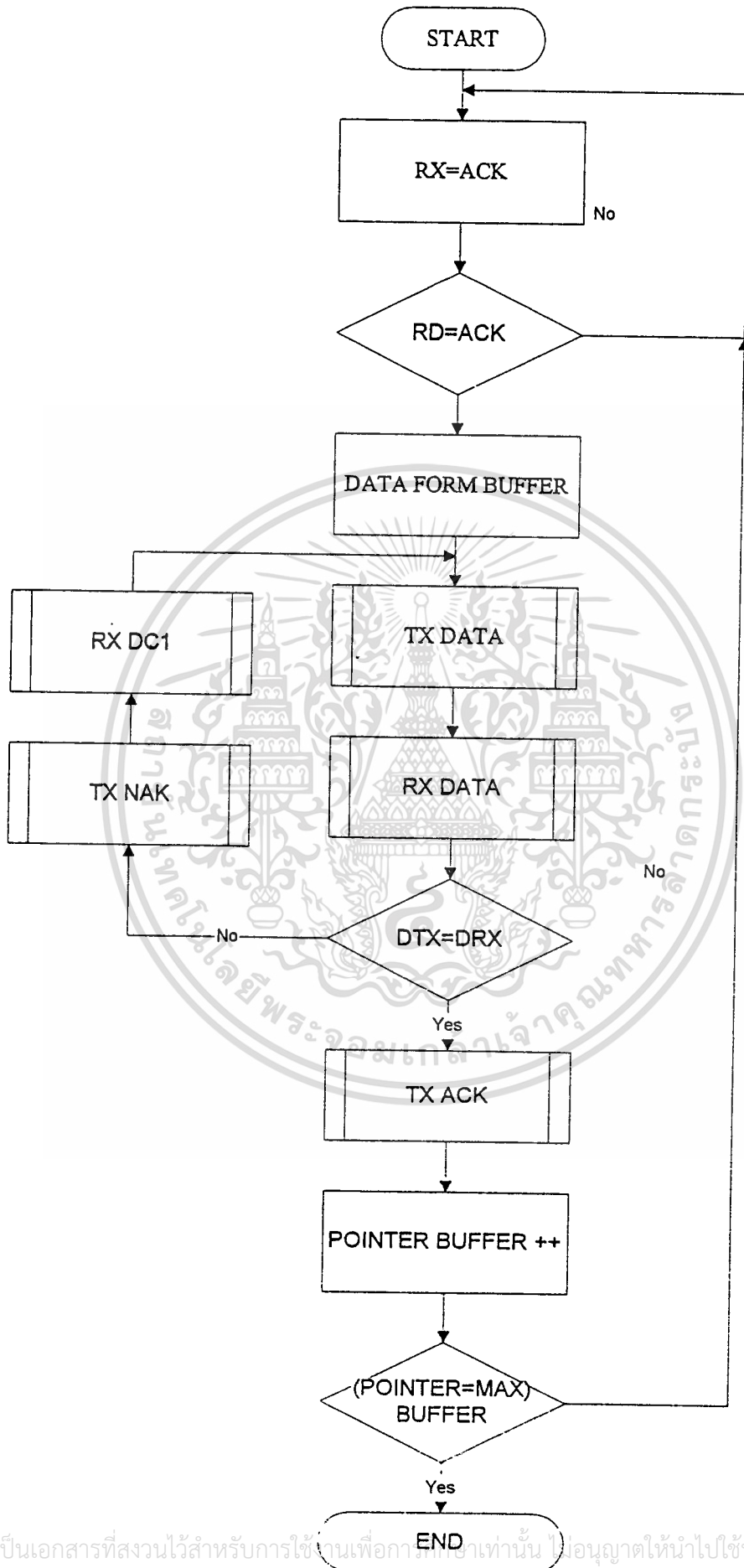
ขึ้น

4. ทำการเปรียบเทียบข้อมูลในขณะที่ส่งและรับข้อมูล

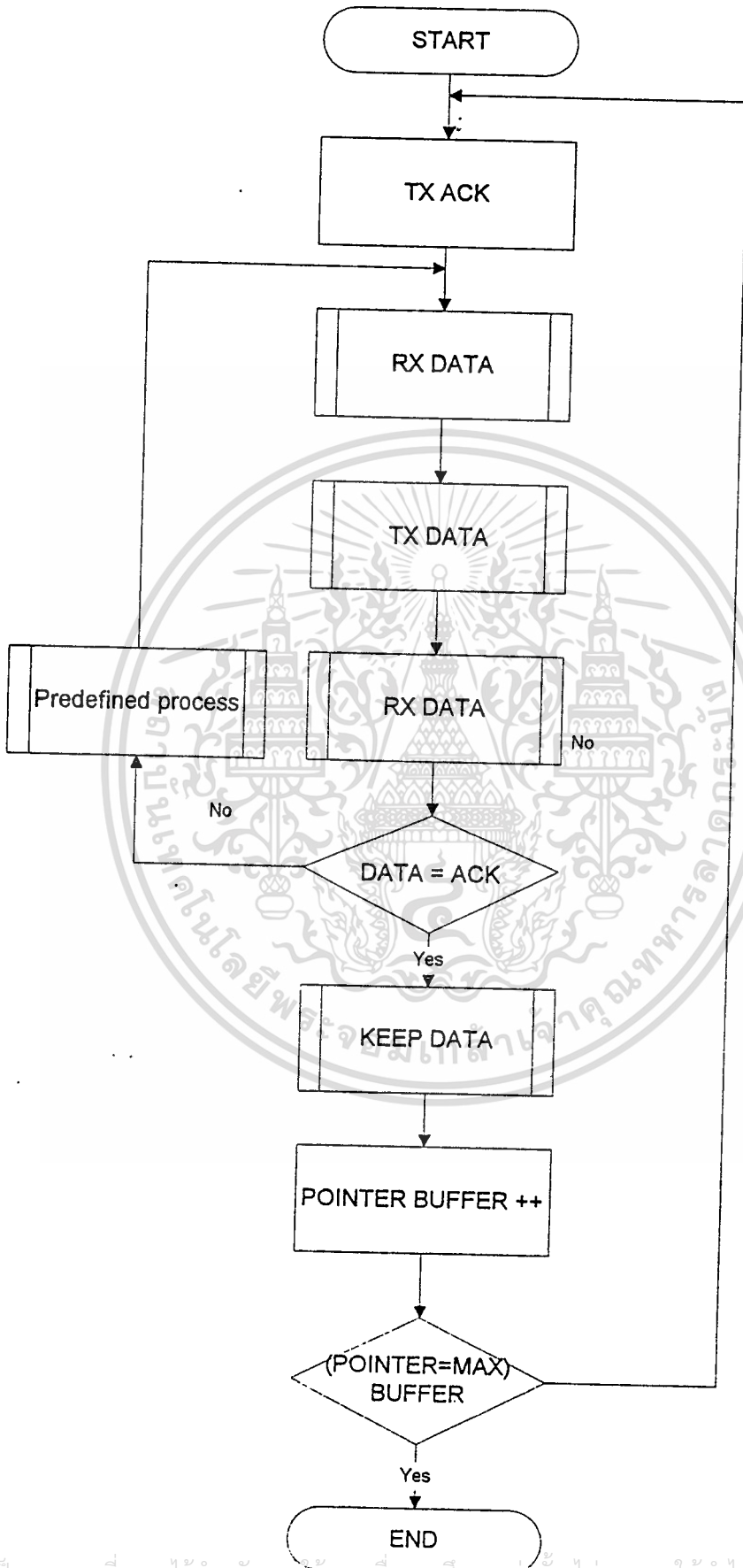
5. ทำการแก้ไขจนข้อมูลระหว่างสองส่วน มีค่าเหมือนกัน ตามข้อกำหนดที่ใช้

6. ข้อกำหนดมาตรฐานที่ใช้ร่วมกัน





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
 6.8(ก) ไฟล์ขาดแสดงการรับส่งข้อมูลระหว่างชุด 8031 กับไมโครคอมพิวเตอร์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

6.8(ข) ไฟฟ้าทาส่งการรับส่งข้อมูลระหว่างชุด 8031 กับไมโครคอมพิวเตอร์ครั้งที่มีการนำไปใช้

โปรแกรมการทดสอบการเชื่อมต่อไมโครคอนโทรลเลอร์กับ PC

```

#define PORT 0

#define MAX_DATA_TRANS 171

#define MAX_DATA 160

#define MAX_HEAD 11

#define MAX_DATA_PAGE 40

#define Max 160

#define Max_p 40


#include <dos.h>
#include <bios.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <stdarg.h>
#include <graphics.h>

unsigned int DaTa0[Max];

unsigned char data_page[MAX_DATA_PAGE];

int Data_num0[Max_p] = {1,0,2,0,3,0,4,0,5,0
                        ,6,0,7,0,8,0,9,0,10,0
                        ,11,0,12,0,13,0,14,0,15,0
                        ,16,0,17,0,18,0,19,0,20,0};

int Data_num1[Max_p];

int Data_level[Max_p];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

unsigned char str_d1[MAX_DATA_TRANS];
unsigned char str_d2[MAX_DATA];
unsigned char str_d3[MAX_HEAD];

```

```

unsigned char ch1[9];
unsigned char ch2[9];
unsigned char ch3[9];
unsigned char rate[2];

```

```

/* System sport */
void wait(int port);
void sport(int port, char c);
void port_init(int port, unsigned char code);
int rport(int port);
int check_stat(int port);
void system_rx();
void set_formx(int num_page);

union comp{
    unsigned char w[2];
    unsigned int b;
};

```

```

union kkp{
    char chad[2];
    int dxp;
};

```

```

union com_b{
    unsigned char nppm[2];

```

```

    unsigned int value;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }

main()
{
    union kkp keyp;
    union comp x;
    int i,j;
    port_init(PORT, 231); /* initialize the serial port */
    system_rx();
    getch();
}

void system_rx()
{
    int Count_da = 0;
    unsigned char ch_buf1; /*control byte*/
    unsigned char ch_buf2; /*data transfer*/
    int Control = 1;

    while(Count_da < MAX_DATA_TRANS){
        ch_buf1 = 0x06;
        sport(PORT,ch_buf1); /* send control start next
data */

        ch_buf1 = 0xff;

        do{
            ch_buf2 = rport(PORT); /* resive data */
            sport(PORT,ch_buf2); /* send for compare data of
source */

            ch_buf1 = rport(PORT); /* if resive ACK = 0x06h
data "T"

/* if resive NAK = 0x15h data
"F" */

```

```

switch(ch_buf1){
    case 0x06:
        str_d1[Count_da] = ch_buf2;
        Control = 0;
        ch_buf1 = 0x06;
        break;

    case 0x15:
        ch_buf1 = 0x11;
        sport(PORT,ch_buf1);
        Control = 1;
    default:
        ch_buf1 = 0x11;
        }while((ch_buf1 == 0x11) && Control);
        Count_da +=1;
        Control = 1;
    }
    return;
}

```

/* Wait for a response. */

```
void wait(int port)
```

```

{
    if(rport(port) != '.') {
        printf("Communication error\n");
        exit(1);
    }
}

```

/* Send a character out the serial port. */

```
void sport(int port, /* i/o port */
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

char c) /* character to send */
{
    union REGS r;

    r.x.dx = port; /* serial port */
    r.h.al = c; /* char to send */
    r.h.ah = 1; /* send character function */
    int86(0x14, &r, &r);

    if(r.h.ah & 128) {
        printf("Send error detected in serial port");
        exit(1);
    }
}

/* Read a character from a port. */
rport(int port) /* i/o port */
{
    union REGS r;

    /* wait for a character */
    while(!(check_stat(port)&256));

    r.x.dx = port; /* serial port */
    r.h.ah = 2; /* read character function */
    int86(0x14, &r, &r);

    if(r.h.ah & 128)
        printf("Read error detected in serial port");

    return r.h.al;
}

```

```

/* Check the status of the serial port. */

```

```

check_stat(int port) /* i/o port */
{
    union REGS r;

    r.x.dx = port; /* serial port */
    r.h.ah = 3; /* read status */
    int86(0x14, &r, &r);
    return r.x.ax;
}

```

```

/* Initialize the port. */
void port_init(int port, unsigned char code)
{
    union REGS r;

    r.x.dx = port; /* serial port */
    r.h.ah = 0; /* initialize port function */
    r.h.al = code; /* initialization code - see text for details */
    int86(0x14, &r, &r);
}

```

โปรแกรมการทดสอบชุด LCO-Display

```
org 0000h
```

```
mov sp,#40h
```

```
lcall set_system
```

```
setb tr1
```

```
mov dptr,#table_x
```

```
mov r0,#0ABh
```

```
lcall up_tx_data
```

```
sjmp $
```

```
set_system:
```

```
mov scon,#01010010b
```

```
mov tmod,#00100000b
```

```
mov th1,#11111101b
```

```
ret
```

```
up_tx_data:
```

```
lcall rx_data
```

```
cjne a,#06h,up_tx_data
```

```
jc up_tx_data
```

```
back:
```

```
mov a,#00h
```

```
movc a,@a+dptr
```

```
mov r3,a
```

```
lcall tx_data
```

```
lcall rx_data
```

```
cjne a,03h,error
```

```
jc error
```

```
mov a,#06h
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
lcall tx_data
jmp exit_data
```

error:

```
mov a,#15h
lcall tx_data
lcall rx_data
cjne a,#11h,back
jc back
```

exit_data:

```
inc dptr
djnz r0,up_tx_data
ret
```

rx_data:

```
jnb ri,$
clr ri
mov a,sbuf
ret
```

tx_data:

```
jnb ti,$
clr ti
mov sbuf,a
ret
```

delay:

```
mov r1,#0ffh
```

delay1:

```
mov r2,#400
```

```

de:  nop
      nop
      djnz r2,de
      djnz r1,delay1
      ret

```

;12345678901234567890123456789012345678901234567890

table_x: DB

23H,47H,50H,26H,07H,95H,01H,30H,25H,10H,0A0H

DB

03H,7FH,05H,7CH,09H,38H,0AH,77H,0CH,7DH,01

H,9FH,0AH,0E3H

DB

0FH,0B0H,0DH,65H,0CH,91H,0EH,0A3H,08H,38H,0BH,8BH,06H,2F

H

DB 09H,64H,07H,0D5H,0BH,27H,08

H,1AH,0EH,0B0H,04H,0E1H

; DATA PAGE 2

DB

02H,24H,0FH,5DH,08H,98H,0BH,30H,01H,17H,0AH,0F0H,09H,7CH

DB

07H,0DDH,0EH,81H,02H,02H,0DH,35H,05H,8DH,08

H,0A0H,01H,92H

DB 00H,50H,09H,0E3H,0CH,0B4H,05

H,4CH,07H,7FH,06H,0F0H

; DATA PAGE 3

DB

00H,25H,04H,4BH,0AH,32H,07H,9CH,05H,20H,0FH,11H,0AH,37H

DB 0BH,09H,0CH,0DH,07H,0ACH,08

H,59H,02H,70H,0AH,12H,09H,55H

DB 0CH,0BCH,08H,3EH,0BH,0FH,07
H,0D8H,0DH,92H,01H,01H

; DATA PAGE 4

DB 0AH,00H,01H,0B7H,0BH,72H,03
H,19H,0DH,3EH,0EH,0B0H,07H,2CH

DB 03H,19H,0CH,81H,06H,2DH,0FH,0
E0H,08H,01H,01H,0B3H,0CH,8CH

DB

02H,0E1H,0BH,29H,0AH,36H,08H,16H,04H,0FH,01H,7CH

;DB

23H,47H,50H,26H,07H,95H,01H,30H,25H,10H,0A0H

;DB

'ABCDEFGHIJKLMNOPQRSTUVWXYZ0ABCDEFGHIJKLMNOPQRSTUVWXYZ
TWXYZ0'

;DB

'ABCDEFGHIJKLMNOPQRSTUVWXYZ0ABCDEFGHIJKLMNOPQRSTUVWXYZ
TWXYZ0'

;DB

'ABCDEFGHIJKLMNOPQRSTUVWXYZ0ABCDEFGHIJKLMNOPQRSTUVWXYZ
TWXYZ0'

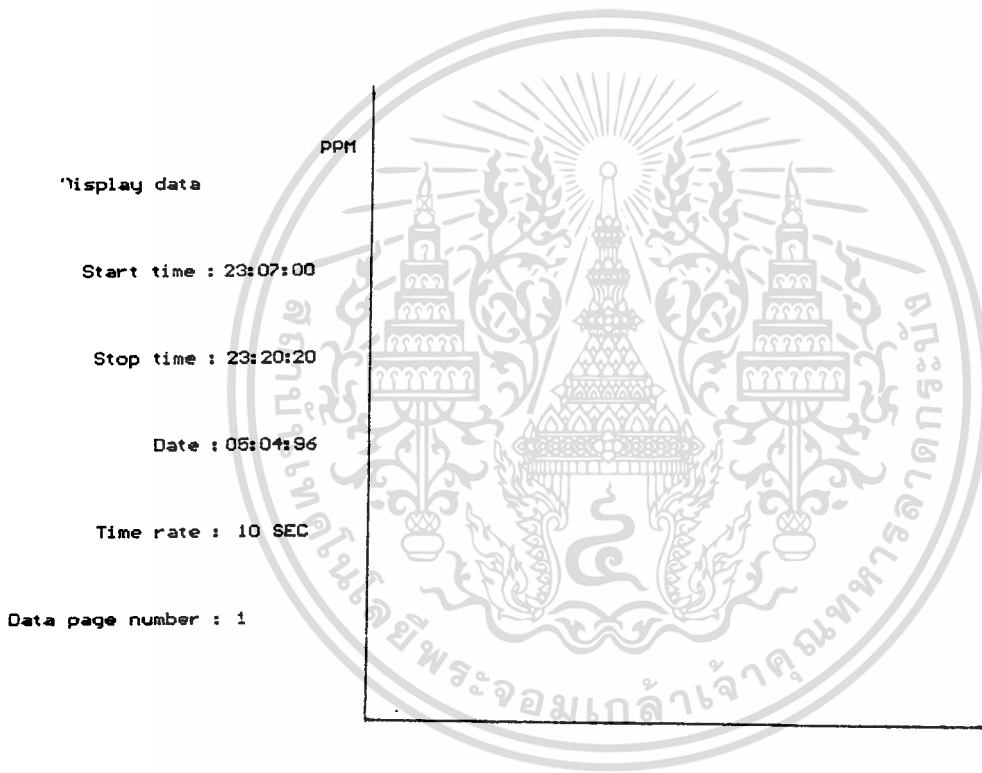
;DB '1234567890'

end

6.6 ผลการทดลองระบบโดยรวม

การทดลอง

ต่อเครื่อง Data Logger กับเครื่อง PC โดยผ่านสายส่ง RS232C เข้าด้วยกัน ON เครื่องพร้อมกัน เริ่มทำการเก็บข้อมูล โดยทำการ Set Up ระบบทั้งสองเครื่องก่อน เราจะเรียกโปรแกรมชื่อ NA3.P.C. ซึ่งเป็นโปรแกรมภาษาซีมาทำการ RUN เพื่อแสดงผลการทดลองเก็บข้อมูลจริง (Real Time) ก็จะปรากฏภาพดังนี้



ซึ่งจากภาพที่ปรากฏจะมีค่าความหมายต่าง ๆ ดังนี้

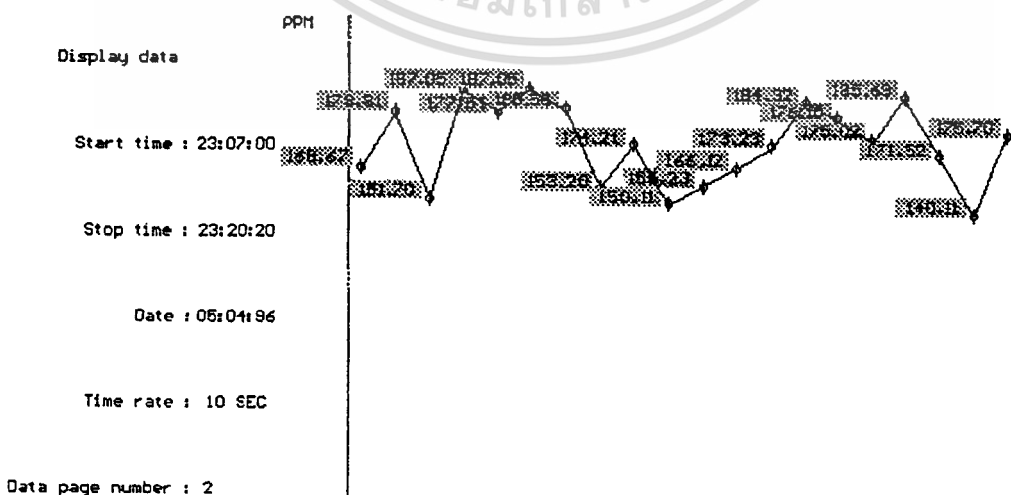
- แกน Y จะเป็นค่า PPM ซึ่งเป็นปริมาณความเข้มข้นของก๊าซ
- แกน X จะเป็นค่าเวลาที่ใช้ในการเก็บข้อมูล
- Start time : 23:07:00 เป็น ค่าเวลาเริ่มต้นในการเก็บข้อมูล
- Stop time : 23:20:20 เป็นค่าเวลาสิ้นสุดการเก็บข้อมูล
- Date : 05:04:96 เป็นค่าวัน เดือน ปี ปัจจุบัน
- Time rate : 10 Sec เป็นค่าอัตราเวลาในการสุ่มเก็บข้อมูล
- Data page number : 1 เป็นหมายเลขแผ่นข้อมูลหน้าที่... ในท่อนี้คือ หน้าที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโปรแกรมภาษาซีเราจะแบ่งการเก็บข้อมูลเป็น 4 Page และขนาดการเก็บข้อมูล Page ละ 20 ค่า ซึ่งเราจะใช้ 4 ปุ่มในการควบคุมที่เป็นคีย์บอร์ด PC ดังนี้

- F1 = การแสดงผลการเก็บข้อมูล
- Page Up = เพิ่ม Data page
- Page Down = ลด Data page
- F10 = Exit (ออกจากระบบ)

ในการ Set Up เครื่อง Data Logger เมื่อ ON POWER ที่ Display จะ Show คำว่า System Busy ซึ่งระบบยังไม่พร้อมเก็บข้อมูล จะต้องทำการล้างตัวก๊าซเซนเซอร์ก่อนเป็นเวลาประมาณ 4 นาที (Delay) เมื่อล้างหัวก๊าซเสร็จ Display จะ Show Set system ซึ่งจากหน้าปัดเครื่องจะมี 3 ปุ่ม คือ 1, 2, Enter ให้กดปุ่ม 1 เพื่อ Set ค่าต่าง ๆ ดังนี้ Set Time, Set Date, Set Rate, Up Load Data to PC, Clear Data Buffer โดยกดปุ่ม 1 ซ้ำ ๆ กันไป ค่าที่จะวนเวียนตามนี้ ถ้าจะ Set Time ก็ให้กดปุ่ม 1 มา Show ที่ Display จากนั้นก็กดปุ่ม 2 เมื่อเลือก Set Hour, Set Min, Set Sec แล้วมากดปุ่ม Enter เพื่อเลือกค่า จากนั้นปุ่ม 1 หรือปุ่ม 2 จะเป็นการเลือกค่าเพิ่มขึ้นหรือลดลงตามลำดับ ซึ่งในการ Set Date, Set Rate ก็ทำในลักษณะเดียวกัน ส่วนการ Set Up Load Data to PC จะหมายถึง การทดสอบการ Ling กันระหว่างเครื่อง Data Logger กับเครื่อง PC โดยเราจะโปรแกรมข้อมูลสมมุติไว้ใน EPROM แล้ว การทดสอบก็ทำได้โดย กดปุ่ม 1 เลือก Up Load Data to PC จากนั้นกดปุ่ม 2 เพื่อยืนยันการ Load ข้อมูล และกดปุ่ม Enter ตามเพื่อส่งข้อมูลสมมุติขึ้น หน้าจอที่เครื่อง PC ก็จ่าปรากฏภาพดังรูปข้างล่างนี้ (กด F1 ที่เป็นพิมพ์ของ PC)



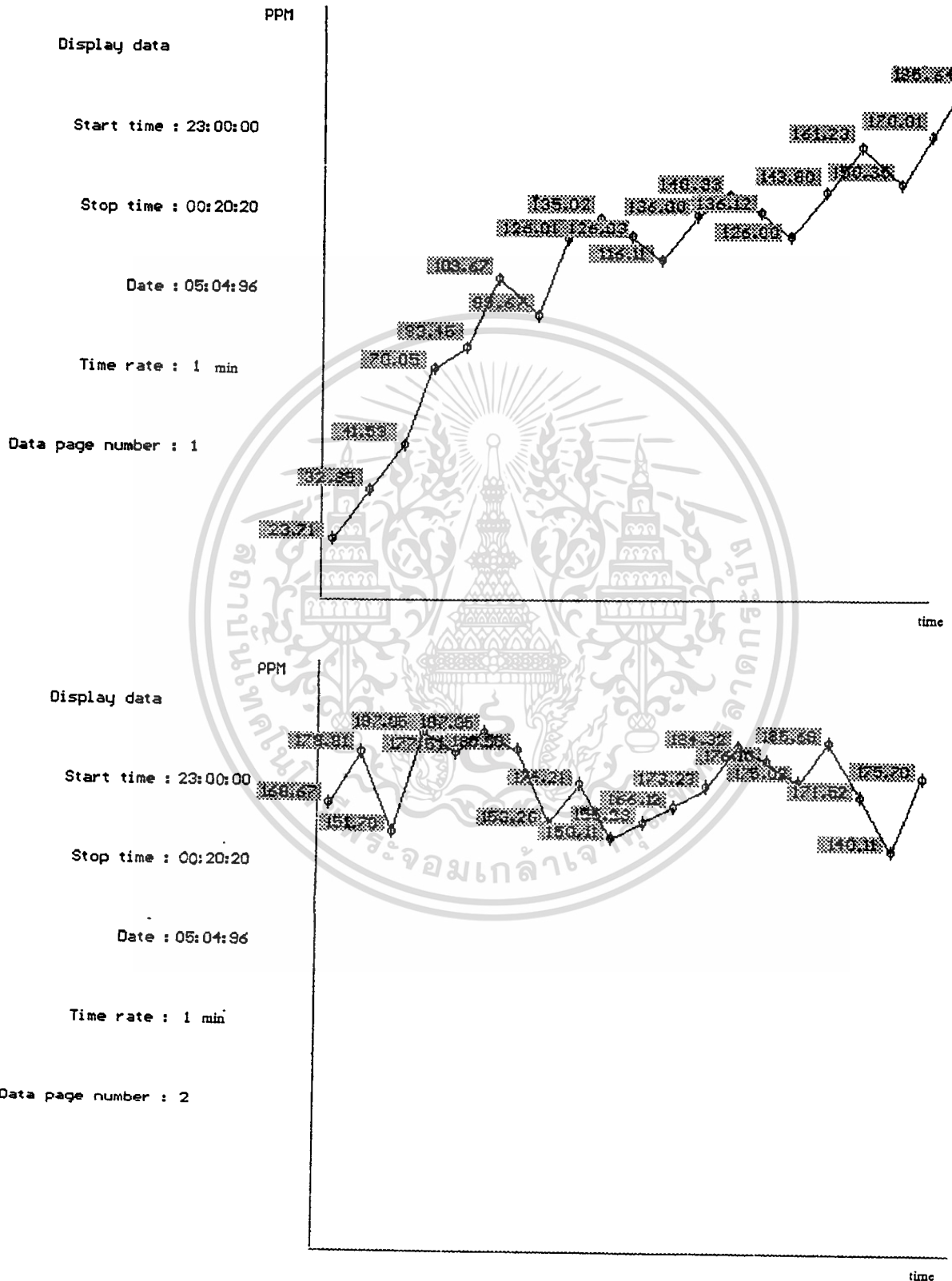
จากภาพที่ปรากฏแสดงให้เห็นว่า เราสามารถส่งข้อมูลไปแสดงผลบน PC ได้ ดังนั้นเราจะทำการ Clear ข้อมูลเก่าทิ้ง โดยกดปุ่ม 1 เลือก Clear Data Buffer และกดปุ่ม 2 เพื่อยืนยันการ Clear ข้อมูล แล้วกดปุ่ม Enter ข้อมูลเก่าก็จะหายไป และเมื่อกดปุ่ม Enter อีกครั้ง ก็จะเป็นการเก็บข้อมูลจริง ซึ่งขั้นตอนการ Set Up เครื่องก็มีเพียงเท่านี้

สำหรับในการทดลองจริง เนื่องจาก Data ที่เราจะเก็บเป็น Cigarette's smoke ดังนั้นเราต้องนำเครื่องไปไว้ในสถานที่ต่าง ๆ เช่น บาร์, คาเฟ่, เซ็คต่าง ๆ ฯลฯ ซึ่งเป็นการไม่สะดวกกับการทดลอง จึงสมมุติสถานที่จริง โดยใช้ห้องทดลองขนาด 3x6 m. วางตำแหน่งเครื่องไว้กลางห้อง จากนั้นก็ทดสอบจุดนุหรี่ 10 ม้วน ทัวบริเวณต่าง ๆ ในห้อง และ Set Up เครื่องเก็บข้อมูล ซึ่งในการเก็บข้อมูล เราจะพิจารณา 2 กรณี คือ

1. แบบใช้พัดลมช่วยดูดอากาศ
2. แบบไม่ใช้พัดลมช่วยดูดอากาศ

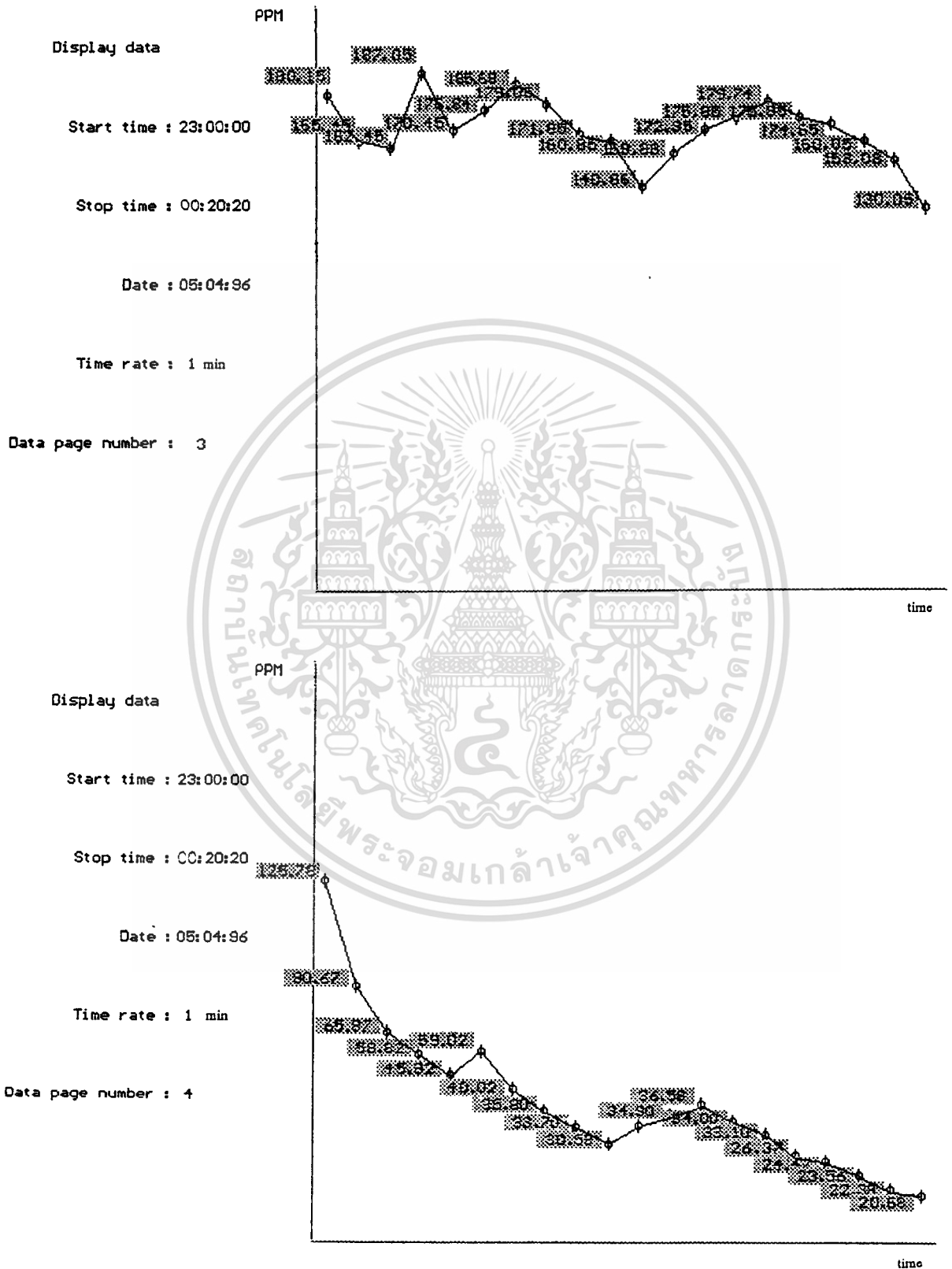


ผลการทดลอง (แบบใช้พัดลมช่วยดูดอากาศ)



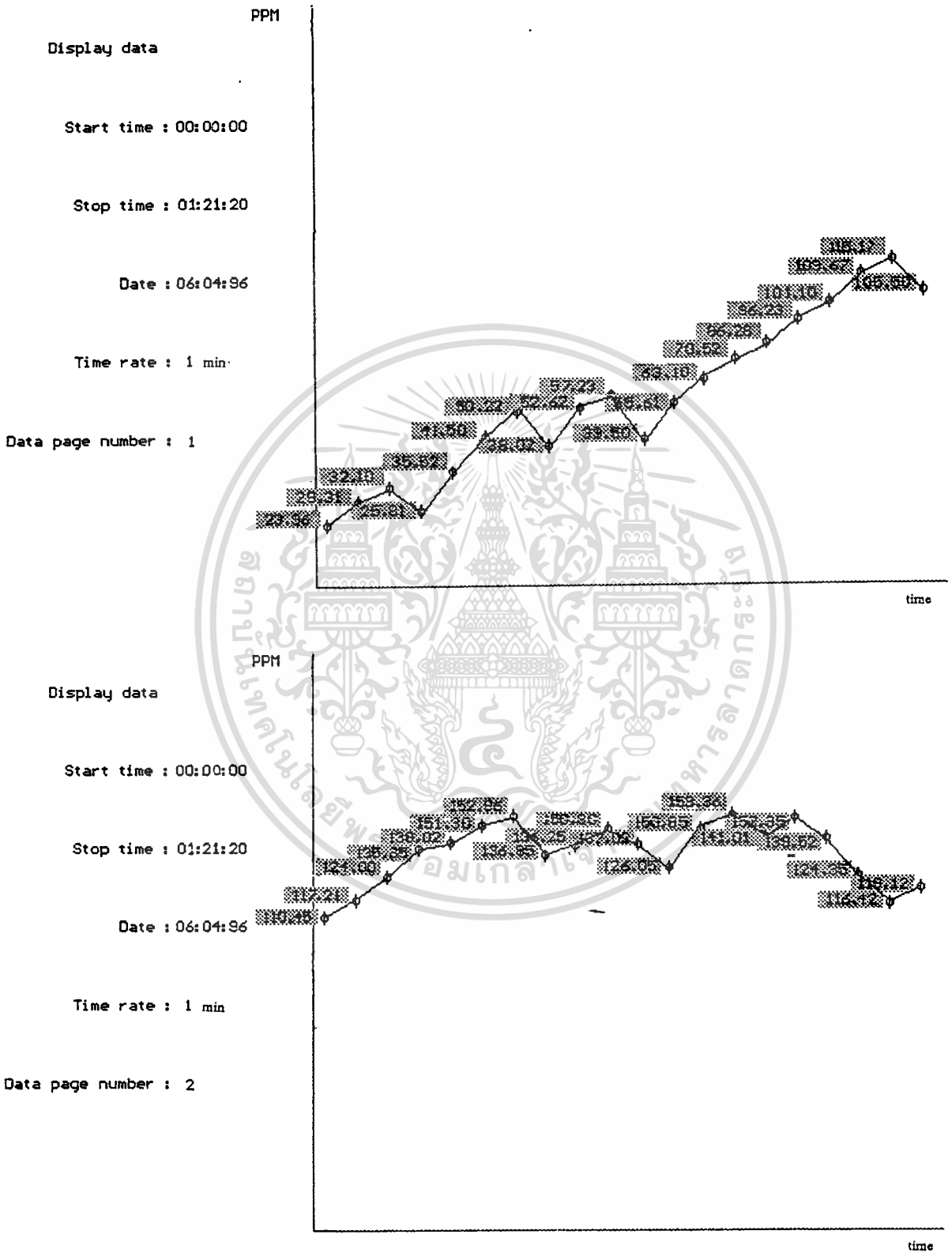
6.9(ก) แสดงกราฟการตรวจวัดข้อมูลจริงแบบใช้พัดลมดูดอากาศชุด 1 และ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารการตรวจวัดข้อมูลจริงแบบใช้พัดลมดูดอากาศชุด 3 และ 4
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

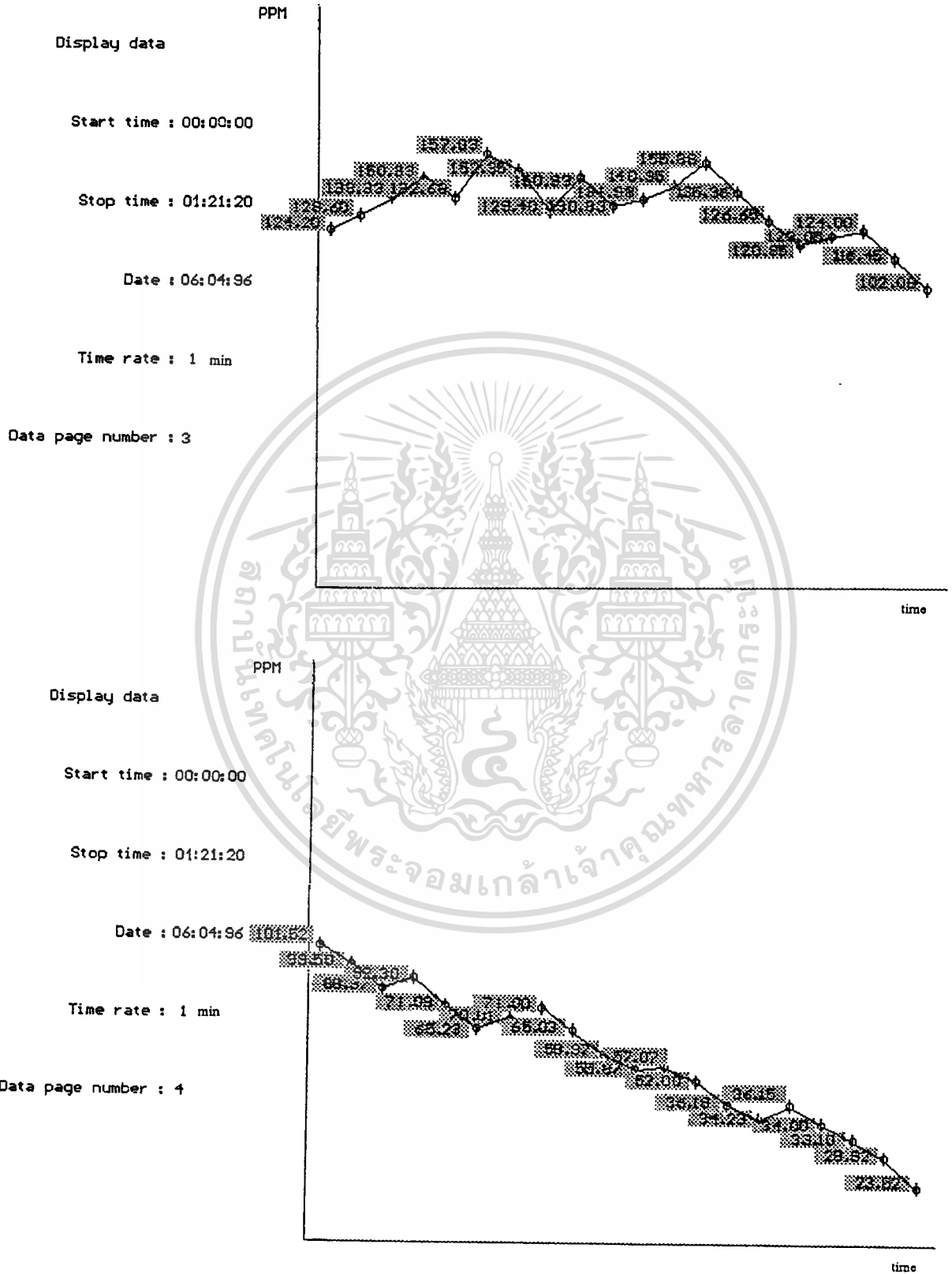
ผลการทดลอง (แบบไม่ใช้พัดลมช่วยดูดอากาศ)



6.10(ก) แสดงกราฟการตรวจวัดข้อมูลจริงแบบไม่ใช้พัดลมดูดอากาศชุด 1 และ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



6.10(ข) แสดงกราฟการตรวจวัดข้อมูลจริงแบบไม่ใช่พัดลมดูดอากาศชุด 3 และ 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากผลการทดลองจะ เห็นว่า ค่าสูงสุดที่ได้จากการใช้พัคลมช่วยมีค่า PPM มากกว่า แบบ
 ไม่ใช่พัคลมอยู่ไม่มากนัก ประมาณ 20 กว่าค่า ซึ่งอาจสรุปได้ว่า การใช้พัคลมช่วย กับไม่ใช่พัคลม
 มีค่าไม่แตกต่างกัน เมื่อขณะก๊าซอยู่ในช่วงกระจายตัวที่ ภาชนะ (ห้อง)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

สรุปผลและข้อเสนอแนะ

จากโครงการนี้ทางกลุ่มนักศึกษาที่จัดทำได้พยายามปรับปรุงโครงการโดยยึดหลักให้เป็นไปตามโครงการที่ตั้งไว้ คือการนำก๊าซเข้ามาแปลงค่าแล้วแสดงผลเป็นค่าปริมาณก๊าซออกทางจอ LCD พร้อมกับกับแสดงเวลาขณะปัจจุบัน (Real Time Clock) และนำข้อมูลเก็บไว้ในหน่วยความจำ เพื่อนำค่าไปเขียนกราฟต่อไปซึ่งในจุดนี้ทางกลุ่มนักศึกษาที่ได้จัดทำสามารถที่จะทำการทดลองจนได้ผลออกมาตามจุดประสงค์ที่ตั้งไว้คือจะมีชุดของ RTC และคีย์บอร์ด (Keyboard) ที่สามารถจะตั้งเวลาในการจะตรวจเก็บข้อมูลในแต่ละครั้งได้ตามเวลาจริงโดยในที่นี้จะตั้งเวลาเก็บข้อมูลได้ 3 ค่าคือ ทุก 10 วินาที, 30 วินาที และ ทุก 1 นาที เพื่อความสะดวกในการอ่านของปริมาณก๊าซที่เข้ามา ซึ่งถ้าเราทำการเก็บข้อมูลทุก 10 วินาที โดยมีค่าของหน่วยความจำเท่ากับ 32 กิโลไบต์ เราจะเก็บข้อมูลต่อเนื่องได้ประมาณ 2 วัน และผลของการเชื่อมต่อของชุดไมโครคอนโทรลเลอร์กับชุดไมโครคอมพิวเตอร์ ในการส่งค่าข้อมูลจากหน่วยความจำเพื่อออกไปแสดงผลที่จอมอนิเตอร์ (monitor) เราจะแสดงผลของข้อมูลออกมาในรูปของกราฟที่เป็นเชิงเส้น เพื่อความสะดวกในการวิเคราะห์และการประมวลผลทางสถิติได้ โดยการเลือกใช้รูปแบบของการแสดงผลบนจอมอนิเตอร์เราจะแสดงออกมาในรูปแบบวินโดว์ (windows) ทั่วๆ ไปซึ่งเหมาะกับการใช้งานได้ง่าย แต่จากโครงการนี้ในส่วนของก๊าซเซ็นเซอร์ซึ่งในการตรวจวัดนั้นเรายังได้ค่าที่ไม่ค่อยมาตรฐานอยู่ซึ่งเกิดจากเหตุผลหลายประการ เช่นค่าความผิดพลาดที่เกิดจากตัวอุปกรณ์ก๊าซเซ็นเซอร์เองซึ่งมีพารามิเตอร์ (Parameter) ที่ไม่สามารถควบคุมได้เช่นค่าความต้านทานในอุปกรณ์ที่เปลี่ยนแปลงตามอุณหภูมิ อีกทั้งปัญหาการระบายก๊าซที่ผ่านเข้ามาที่ตัวตรวจจับก๊าซไม่ให้มีการค้างนานเกินไปซึ่งจะทำให้ค่าที่ได้ผิดพลาด ดังนั้นขั้นตอนของแนวทางในการแก้ปัญหาทำโดยใช้พัดลมดูดอากาศเพื่อช่วยในการถ่ายเทก๊าซให้เป็นไปอย่างรวดเร็ว อีกเหตุผลหนึ่งคือความผิดพลาดของวงจรขยายความแตกต่างเกิดค่าแรงดันออฟเซต (Off set Voltage) ทำให้ค่าแรงดันของก๊าซเซ็นเซอร์ที่เข้าที่ทุกไม่สามารถกำหนดให้เท่ากับ 0 โวลต์ได้ทำให้ข้อมูลต้นๆเกิดผิดพลาดและในส่วนของโปรแกรมภาษาซี การสร้างสมการเพื่อหาค่าเปรียบเทียบกับมาตรฐานที่มีมาในคู่มือของก๊าซเซ็นเซอร์นั้นค่าที่คำนวณได้ยังไม่ตรงกับค่าที่ตรวจวัดได้จริง เนื่องจากสมการการคำนวณที่สร้างมาจากตารางที่เป็นค่าลึกลับกาลิเทียม ซึ่งไม่เป็นเชิงเส้น เราจึงใช้หลักการโดยการเทียบค่าอินพุต

สำหรับแนวทางการพัฒนาต่อนั้นอาจจะทำการปรับปรุงในส่วนของการตรวจจับเพื่อ
วัดค่าของก๊าซชนิดอื่นๆ ได้อีกในวงจรเดียวกันโดยการเขียนโปรแกรมกำหนดสมการของก๊าซ
แต่ละชนิดนั้นขึ้นมา และเขียนเมนูในการเลือกใช้บนจอมอนิเตอร์เพิ่มอีกเราก็จะสามารถนำ
โครงการนี้ไปประยุกต์ใช้ในงานอื่นๆ ได้อีกต่อไป





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรม .

```

#ifdef __TINY__

#error BGIDEMO will not run in the tiny model.

#endif

#define PORT 0

#define MAX_DATA_TRANS 171

#define MAX_DATA 160

#define MAX_HEAD 11

#define MAX_DATA_PAGE 40

#define Max 160

#define Max_p 40

#define ESC 0x1b /* Define the escape key
*/

#define TRUE 1 /* Define some handy
constants */

#define FALSE 0 /* Define some handy
constants */

#define PI 3.14159 /* Define a value for PI */

#define ON 1 /* Define some handy
constants */

#define OFF 0 /* Define some handy
constants */

#include <dos.h>

#include <bios.h>

#include <stdio.h>

```

```
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <stdarg.h>
#include <graphics.h>
```

```
char *Fonts[] = {
    "DefaultFont", "TriplexFont", "SmallFont",
    "SansSerifFont", "GothicFont"
};
```

```
char *LineStyle[] = {
    "SolidLn", "DottedLn", "CenterLn", "DashedLn", "UserBitLn"
};
```

```
char *FillStyles[] = {
    "EmptyFill", "SolidFill", "LineFill", "LtSlashFill",
    "SlashFill", "BkSlashFill", "LtBkSlashFill", "HatchFill",
    "XHatchFill", "InterleaveFill", "WideDotFill", "CloseDotFill"
};
```

```
char *TextDirect[] = {
    "HorizDir", "VertDir"
};
```

```
char *HorizJust[] = {
    "LeftText", "CenterText", "RightText"
};
```

```
char *VertJust[] = {
    "BottomText", "CenterText", "TopText"
```

```
};
```

```
struct PTS {
```

```
    int x, y;
```

```
};    /* Structure to hold vertex points    */
```

```
int    GraphDriver;    /* The Graphics device driver
    */
```

```
int    GraphMode;    /* The Graphics mode value
    */
```

```
double AspectRatio;    /* Aspect ratio of a pixel on the
screen*/
```

```
int    MaxX, MaxY;    /* The maximum resolution of the
screen */
```

```
int    MaxColors;    /* The maximum # of colors
available    */
```

```
int    ErrorCode;    /* Reports any graphics errors
    */
```

```
struct palettetype palette;    /* Used to read palette info
    */
```

```
unsigned int DaTa0[Max];
```

```
unsigned char data_page[MAX_DATA_PAGE];
```

```
int Data_num0[Max_p] = {1,0,2,0,3,0,4,0,5,0
                        ,6,0,7,0,8,0,9,0,10,0
                        ,11,0,12,0,13,0,14,0,15,0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

,16,0,17,0,18,0,19,0,20,0};

int Data_num1[Max_p];

int Data_level[Max_p];

unsigned char str_d1[MAX_DATA_TRANS];

unsigned char str_d2[MAX_DATA];

unsigned char str_d3[MAX_HEAD];

unsigned char ch1[9];

unsigned char ch2[9];

unsigned char ch3[9];

unsigned char rate[2];

void Initialize(void);

void ReportStatus(void);

void LineRelDemo();

void SayGoodbye(void);

void Pause(void);

void MainWindow(char *header);

void StatusLine(char *msg);

void DrawBorder(void);

void changetextstyle(int font, int direction, int charsize);

int gprintf(int *xloc, int *yloc, char *fmt, ...);

void make_page(int bxc);

void grap_d();

void cal_data();

void show_head();

/* System sport */

void wait(int port);

void sport(int port, char c);

```

void port_init(int port, unsigned char code);
int rport(int port);
int check_stat(int port);
void system_rx();
void set_formx(int num_page);

```

```

union comp{
    unsigned char w[2];
    unsigned int b;
};

```

```

union kkp{
    char chad[2];
    int dxp;
};

union com_b{
    unsigned char nppm[2];
    unsigned int value;
}

```

```

main()
{
    union kkp keyp;
    union comp x;
    int i,j;

    /*struct viewporttype vp;*/

    int count_page_d = 1; /* start show page 1 of data */
    char cha;

    port_init(PORT, 231); /* initialize the serial port */

```

```

clrscr();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้ทำงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Initialize();          /* Set system into Graphics mode      */
/* ReportStatus();*/    /* Report results of the initialization */
*/

```

```

/* ColorDemo();*/      /* Begin actual
demonstration          */

```

```

if( GraphDriver==EGA || GraphDriver==EGALO ||
GraphDriver==VGA )
/* PaletteDemo();*/

```

```

LineRelDemo(1);
make_page(1);
for(;;){
while(bioskey(1)==0);
keyp.dxp = bioskey(0);
if(keyp.chad[0] ==0)
switch(keyp.chad[1]){
case 0x3b:

```

```

keyp.chad[0] = 1;

```

```

system_rx();

```

```

for(i = 0; i<MAX_HEAD; i++)

```

```

str_d3[i] = str_d1[i];

```

```

for(i =11; i<MAX_DATA_TRANS; i++)

```

```

str_d2[i-11] = str_d1[i];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีกรนำไปใช้

```

for(i = 0; i < Max_p; i++)
    Data_num1[i] = Data_num0[i];

count_page_d = 1;

set_formx(count_page_d);

make_page(count_page_d);
grap_d();
cal_data();
show_head();
setcolor(YELLOW);
outtextxy(135,100,ch1);
outtextxy(135,150,ch2);
outtextxy(135,200,ch3);
outtextxy(135,250,rate);
break;

case 0x49:
    keyp.chad[0] = 1;
    if(count_page_d != 4)
        count_page_d += 1;
    for(i = 0; i <Max_p; i++)
        Data_num1[i] = Data_num0[i];
    set_formx(count_page_d);

    make_page(count_page_d);
    grap_d();
    cal_data();
    break;

case 0x51:

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

keyp.chad[0] = 1;

if(count_page_d != 1)
    count_page_d -=1;

for(i = 0; i < Max_p; i++)
    Data_num1[i] = Data_num0[i];

set_formx(count_page_d);

make_page(count_page_d);

grap_d();
cal_data();
break;
case 0x44:
    keyp.chad[0] = 1;
    closegraph(); /* Return the system to text
mode */
    exit(0);
default:
    keyp.chad[0] = 1;
}

}

}

void system_rx()
{

    int Count_da = 0;

    unsigned char ch_buf1; /*control byte*/
    unsigned char ch_buf2; /*data transfer*/

    int Control = 1;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

while(Count_da < MAX_DATA_TRANS){
    ch_buf1 = 0x06;
    sport(PORT,ch_buf1);          /* send control start next
data */
    ch_buf1 = 0xff;
    do{
        ch_buf2 = rport(PORT);    /* resive data */
        sport(PORT,ch_buf2);      /* send for compare data of
source */
        ch_buf1 = rport(PORT);    /* if resive ACK = 0x06h
data "T"
                                /* if resive NAK = 0x15h data
"F" */
        switch(ch_buf1){
            case 0x06:
                str_d1[Count_da] = ch_buf2;
                Control = 0;
                ch_buf1 = 0x06;
                break;
            case 0x15:
                ch_buf1 = 0x11;
                sport(PORT,ch_buf1);
                Control = 1;
            default:
                ch_buf1 = 0x11;
        }
    }while((ch_buf1 == 0x11) && Control);
    Count_da +=1;
    Control = 1;
}

```

```

        return;
    }

    /* Wait for a response. */
    void wait(int port)
    {
        if(rport(port) != '.') {
            printf("Communication error\n");
            exit(1);
        }
    }

    /* Send a character out the serial port. */
    void sport(int port, /* i/o port */
               char c) /* character to send */
    {
        union REGS r;

        r.x.dx = port; /* serial port */
        r.h.al = c; /* char to send */
        r.h.ah = 1; /* send character function */
        int86(0x14, &r, &r);

        if(r.h.ah & 128) {
            printf("Send error detected in serial port");
            exit(1);
        }
    }

    /* Read a character from a port. */
    rport(int port) /* i/o port */
    {
        union REGS r;

```

```

/* wait for a character */
while(!(check_stat(port)&256));

r.x.dx = port; /* serial port */
r.h.ah = 2; /* read character function */
int86(0x14, &r, &r);
if(r.h.ah & 128)
    printf("Read error detected in serial port");
return r.h.al;
}

```

```

/* Check the status of the serial port. */
check_stat(int port) /* i/o port */
{
    union REGS r;

    r.x.dx = port; /* serial port */
    r.h.ah = 3; /* read status */
    int86(0x14, &r, &r);
    return r.x.ax;
}

```

```

/* Initialize the port. */

```

```

void port_init(int port, unsigned char code)

```

```

{
    union REGS r;

    r.x.dx = port; /* serial port */
    r.h.ah = 0; /* initialize port function */
    r.h.al = code; /* initialization code - see text for details */
    int86(0x14, &r, &r);
}

```

}

```
void set_formx(int num_page1)
```

{

```
    int begin_p_data, end_p_data;
```

```
    int lindex, jindex;
```

```
    int Count_arr;
```

```
    int Temp;
```

```
    int i;
```

```
    int j,k,l;
```

```
    switch(num_page1){
```

```
        case 1:
```

```
            begin_p_data = 0; /*begin data of page1*/
```

```
            end_p_data = 39; /*end data of page2*/
```

```
            i = 1;
```

```
            break;
```

```
        case 2:
```

```
            begin_p_data = 40; /*begin data of page2*/
```

```
            end_p_data = 79; /*end data of page2*/
```

```
            i = 6;
```

```
            break;
```

```
        case 3:
```

```
            begin_p_data = 80; /*begin data of page3*/
```

```
            end_p_data = 119; /*end data of page3*/
```

```
            i = 10;
```

```
            break;
```

```
        case 4:
```

```
            begin_p_data = 120; /*begin data of page4*/
```

```
            end_p_data = 159; /*end data of page4*/
```

```
            i = 16;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

for(Count_arr= begin_p_data; Count_arr <=end_p_data;
Count_arr++)

    data_page[Count_arr-begin_p_data] = str_d2[Count_arr];

for(Iindex=0; Iindex<=36; Iindex+=2)
    for(Jindex=Iindex+2; Jindex<=38; Jindex+=2){
        if(data_page[Iindex] < data_page[Jindex])
        {
            Temp = data_page[Iindex];
            data_page[Iindex] = data_page[Jindex];
            data_page[Jindex] = Temp;

            Temp = Data_num1[Iindex];
            Data_num1[Iindex] = Data_num1[Jindex];
            Data_num1[Jindex] = Temp;

            Temp = data_page[Iindex+1];
            data_page[Iindex+1] = data_page[Jindex+1];
            data_page[Jindex+1] = Temp;
        }
        else if(data_page[Iindex] == data_page[Jindex])
        {

            Temp = data_page[Iindex+1];
            data_page[Iindex+1] = data_page[Jindex+1];
            data_page[Jindex+1] = Temp;

            Temp = Data_num1[Iindex];
            Data_num1[Iindex] = Data_num1[Jindex];
            Data_num1[Jindex] = Temp;

        }
    }

```

```

        for(j = 1; j <= 20; j++)
            for(k = 0; k <= 20; k++)
                if(Data_num1[k*2] == j)

                    Data_level[j-1] =k+1;

        return;
    }

void grap_d()
{
    int c_o,c_x;
    int num_l,num_l1;
    c_x = 21;
    for(c_o=0; c_o<=38; c_o+=2){
        c_x -=1;
        num_l = Data_num1[c_o];
        setcolor(YELLOW);
        circle(200+ (num_l*20)-1,400 -(c_x*18)+5,2);
    }

    c_x = 1;
    setcolor(WHITE);
    for(c_o = 0; c_o <=18;c_o++){
        num_l = Data_level[c_o];
        num_l1 = Data_level[c_o+1];
        line(200+(c_x*20)-1,40+(num_l*18)-13,200+((c_x+1)*20)-
1,40 +(num_l1*18)-13);
        c_x +=1;
    }

    return;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
```

```
void cal_data()
```

```
{
```

```
    double value_ppm,value_ppmc;
```

```
    double value_ppmdd = 1;
```

```
    char ch_ppm[5];
```

```
    union com_b.compx;
```

```
    int ndig = 5;
```

```
    int c_dx = 21;
```

```
    int num_a;
```

```
    int c_i;
```

```
    setcolor(YELLOW);
```

```
    for(c_i=0; c_i<=38; c_i+=2){
```

```
        c_dx -=1;
```

```
        num_a = Data_num1[c_i];
```

```
        setfillstyle(SOLID_FILL,RED);
```

```
        bar(200+(num_a*20)-21,391-(c_dx*18)
```

```
            ,216+(num_a*20),401-(c_dx*18));
```

```
        ch_ppm[0] = 0;
```

```
        ch_ppm[1] = 0;
```

```
        ch_ppm[2] = 0;
```

```
        ch_ppm[3] = 0;
```

```
        compx.nppm[0] = data_page[c_i+1];
```

```
        compx.nppm[1] = data_page[c_i];
```

```
        value_ppm = compx.value*value_ppmdd;
```

```
        value_ppmc = (value_ppm*200)/4095;
```

```

        gcvt(value_ppmc, ndig, &ch_ppm);

        outtextxy(200+(num_a*20)-3, 400-(c_dx*18)-9, ch_ppm);
    }

    return;
}

```

```

void show_head()

```

```

{
    unsigned a,b,c,d;
    int i;

    ch1[8] = '\0';
    ch2[8] = '\0';
    ch3[8] = '\0';
    rate[2] = '\0';

    ch1[2] = ':';
    ch1[5] = ':';
    ch2[2] = ':';
    ch2[5] = ':';

    ch3[2] = '/';
    ch3[5] = '/';

    for(i = 0; i < 3; i++){
        if(i < 1){
            a = str_d3[i];
            b = a;
            c = a >> 4;
            c = c + 0x30;
            ch1[0] = c;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    c = 0;

    c = 0x0f & b;
    c = c+0x30;
    ch1[1] = c;
}

if((i==1)&&(i<2)){
    a = str_d3[i];
    b = a;
    c = a>> 4;
    c = c+0x30;
    ch1[3] = c;
    c = 0;
    c = 0x0f & b;
    c = c+0x30;
    ch1[4] = c;
}
if(i == 2){
    a = str_d3[i];
    b = a;
    c = a>> 4;
    c = c+0x30;
    ch1[6] = c;
    c = 0;
    c = 0x0f & b;
    c = c+0x30;
    ch1[7] = c;
}

}

for(i = 3; i < 6; i++){
    if(i <4){

```

```

        a = str_d3[i];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

b = a;
c = a >> 4;
c = c + 0x30;
ch2[0] = c;
c = 0;
c = 0x0f & b;
c = c + 0x30;
ch2[1] = c;
}

if((i==4)&&(i<5)){
a = str_d3[i];
b = a;
c = a >> 4;
c = c + 0x30;
ch2[3] = c;
c = 0;
c = 0x0f & b;
c = c + 0x30;
ch2[4] = c;
}

if(i == 5){
a = str_d3[i];
b = a;
c = a >> 4;
c = c + 0x30;
ch2[6] = c;
c = 0;
c = 0x0f & b;
c = c + 0x30;
ch2[7] = c;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

for(i =6; i < 9; i++){
    if(i <7){
        a = str_d3[i];
        b = a;
        c = a>> 4;
        c = c+0x30;
        ch3[0] = c;
        c = 0;
        c = 0x0f & b;
        c = c+0x30;
        ch3[1] = c;
    }
    if((i==7)&&(i<8)){
        a = str_d3[i];
        b = a;
        c = a>> 4;
        c = c+0x30;
        ch3[3] = c;
        c = 0;
        c = 0x0f & b;
        c = c+0x30;
        ch3[4] = c;
    }
    if(i == 8){
        a = str_d3[i];
        b = a;
        c = a>> 4;
        c = c+0x30;
        ch3[6] = c;
        c = 0;
        c = 0x0f & b;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

        ErrorCode = graphresult();          /* Read result of
initialization*/

        if( ErrorCode != grOk ){            /* Error occurred during init
        */

            printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode
        ));

            exit( 1 );

        }

```

```

        getpalette( &palette );            /* Read the palette from board */
        MaxColors = getmaxcolor() + 1;      /* Read maximum number
of colors*/

        MaxX = getmaxx();
        MaxY = getmaxy();                   /* Read size of screen
        */

        getspectratio( &xasp, &yasp );      /* read the hardware aspect
        */

        AspectRatio = (double)xasp / (double)yasp; /* Get correction factor
        */

    }

```

```

/*

    */

/*    REPORTSTATUS: Report the current configuration of the
system */

/*    after the auto-detect initialization.

    */

/*

```

```

void ReportStatus(void)
{
    struct viewporttype    viewinfo;    /* Params for inquiry
procedures*/

    struct linesettingstype lineinfo;
    struct fillsettingstype fillinfo;
    struct textsettingstype textinfo;
    struct palettetype     palette;

    char *driver, *mode;                /* Strings for driver and
mode */
    int x, y;

    getviewsettings( &viewinfo );
    getlinesettings( &lineinfo );
    getfillsettings( &fillinfo );
    gettextsettings( &textinfo );
    getpalette( &palette );

    x = 10;
    y = 4;

    MainWindow( "Status report after InitGraph" );
    settxtjustify( LEFT_TEXT, TOP_TEXT );

    driver = getdrivername();
    mode = getmodename(GraphMode);    /* get current setting
*/

    printf( &x, &y, "Graphics device    : %-20s (%d)", driver,
GraphDriver );

```

```

    gprintf( &x, &y, "Graphics mode      : %-20s (%d)", mode,
    GraphMode );

    gprintf( &x, &y, "Screen resolution : ( 0, 0, %d, %d )", getmaxx(),
    getmaxy() );

```

```

    gprintf( &x, &y, "Current view port : ( %d, %d, %d, %d )",
    viewinfo.left, viewinfo.top, viewinfo.right, viewinfo.bottom );

    gprintf( &x, &y, "Clipping          : %s", viewinfo.clip ? "ON" : "
    OFF" );

```

```

    gprintf( &x, &y, "Current position : ( %d, %d )", getx(), gety() );
    gprintf( &x, &y, "Colors available : %d", MaxColors );
    gprintf( &x, &y, "Current color    : %d", getcolor() );

    gprintf( &x, &y, "Line style       : %s", LineStyles[
    lineinfo.linestyle ] );
    gprintf( &x, &y, "Line thickness  : %d", lineinfo.thickness );

    gprintf( &x, &y, "Current fill style : %s", FillStyles[ fillinfo.pattern
    ] );
    gprintf( &x, &y, "Current fill color : %d", fillinfo.color );

```

```

    gprintf( &x, &y, "Current font      : %s", Fonts[ textinfo.font ] );
    gprintf( &x, &y, "Text direction   : %s", TextDirect[
    textinfo.direction ] );

    gprintf( &x, &y, "Character size   : %d", textinfo.charsize );
    gprintf( &x, &y, "Horizontal justify : %s", HorizJust[ textinfo.horiz
    ] );

    gprintf( &x, &y, "Vertical justify  : %s", VertJust[ textinfo.vert ] );

```

```

    Pause();                               /* Pause for user to read

```

```

    screen*/

```

```

    }

/*
    */

/*    LINERELDEMO: Display pattern using moverel and linerel
cmds. */
/*

    */

void LineRelDemo()
{
    struct viewporttype vp;
    int h, w, dx, dy, cx, cy;
    struct PTS outs[7];
    int i;
    int xx;
    int x,y;
    unsigned char cdv;

    getviewsettings( &vp );

    cleardevice();
    setcolor( BLACK );
    MainWindow("Pawin Komonvipaht");

    setfillstyle( SOLID_FILL, BLUE );
    bar( 0, 0, vp.right-vp.left, vp.bottom-vp.top ); /* Draw backgnd */

    setfillstyle( SOLID_FILL, LIGHTGRAY);
    bar(0, 438, vp.right-vp.left, vp.bottom-vp.top);

```

```

setcolor(MAGENTA);
outtextxy(30,440,"Help");
setcolor(RED);

outtextxy(140,440,"F1");
setcolor(BLACK);
outtextxy(173,440,"-UpLoad");
setcolor(RED);
outtextxy(291,440,"P");
setcolor(BLACK);
outtextxy(314,440,"ageUp");
setcolor(RED);
outtextxy(425,440,"P");
setcolor(BLACK);
outtextxy(456,440,"ageDown");
setcolor(RED);
outtextxy(573,440,"F10");
setcolor(BLACK);
outtextxy(604,440,"-Quit");

setcolor(YELLOW);
outtextxy(60,50,"  Display data  ");
setcolor(YELLOW);
outtextxy(60,100,"  Start time : ");
setcolor(YELLOW);
outtextxy(60,150,"  Stop time : ");
setcolor(YELLOW);
outtextxy(60,200,"  Date : ");
setcolor(YELLOW);
outtextxy(60,250,"  Time rate : ");
setcolor(YELLOW);

```

```

outtextxy(60,300,"Data page number : ");
outtextxy(170,30,"PPM");
setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(200,30,200,400);
setfillstyle( SOLID_FILL, LIGHTGRAY);
bar(200,400,600,400);

outtextxy(590,420,"TIME");

/* for(x = 382; x>= 40; x-=18){
    setcolor(YELLOW);
    outtextxy(200,x,"-");
}
for(x = 220; x<= 600; x+=20){
    setcolor(YELLOW);
    outtextxy(x,394,"I");
}*/
show_head();
}

void make_page(int bxc)
{
    int x;

    setfillstyle(SOLID_FILL,BLUE);
    bar(120,285,140,320);

    switch(bxc){
        case 1:
            outtextxy(123,300,"1");
            setfillstyle(SOLID_FILL,BLUE);
            bar(180,20,650,420);

```

```

setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(200,30,200,400);

setfillstyle( SOLID_FILL, LIGHTGRAY);
bar(200,400,600,400);

/* for(x = 382; x>= 40; x-=18){
    setcolor(YELLOW);
    outtextxy(200,x,"-");
}

for(x = 220; x<= 600; x+=20){
    setcolor(YELLOW);
    outtextxy(x,394,"I");
}*/

break;
case 2:
    outtextxy(123,300,"2");
    setfillstyle(SOLID_FILL,BLUE);
    bar(180,20,650,420);

    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(200,30,200,400);

    setfillstyle( SOLID_FILL, LIGHTGRAY);
    bar(200,400,600,400);

/*    for(x = 382; x>= 40; x-=18){
        setcolor(YELLOW);
        outtextxy(200,x,"-");
    }

    for(x = 220; x<= 600; x+=20){
        setcolor(YELLOW);

```

```

        outtextxy(x,394,"I");
    } */

    break;
case 3:
    outtextxy(123,300,"3");
    setfillstyle(SOLID_FILL,BLUE);
    bar(180,20,650,420);

    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(200,30,200,400);
    setfillstyle( SOLID_FILL, LIGHTGRAY);
    bar(200,400,600,400);

    /* for(x = 382; x>= 40; x-=18){
        setcolor(YELLOW);
        outtextxy(200,x,"-");
    }
    for(x = 220; x<= 600; x+=20){
        setcolor(YELLOW);
        outtextxy(x,394,"I");
    } */

    break;
case 4:
    outtextxy(123,300,"4");
    setfillstyle(SOLID_FILL,BLUE);
    bar(180,20,650,420);

    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(200,30,200,400);
    setfillstyle( SOLID_FILL, LIGHTGRAY);
    bar(200,400,600,400);

```

```

bar(200,400,600,400);

/*    for(x = 382; x>= 40; x-=18){
        setcolor(YELLOW);
        outtextxy(200,x,"-");
    }

    for(x = 220; x<= 600; x+=20){
        setcolor(YELLOW);
        outtextxy(x,394,"|");
    } */

break;
}

/*
    */
/*    SAYGOODBYE: Give a closing screen to the user before
leaving.*/
/*
    */

void SayGoodbye(void)
{
    struct viewporttype viewinfo; /* Structure to read viewport */
    int h, w;

    MainWindow( "== Finale ==" );

    getviewsettings( &viewinfo ); /* Read viewport settings */

```

```
settextjustify( CENTER_TEXT, CENTER_TEXT );
```

```
h = viewinfo.bottom - viewinfo.top;
```

```
w = viewinfo.right - viewinfo.left;
```

```
outtextxy( w/2, h/2, "That's all, folks!" );
```

```
StatusLine( "Press any key to EXIT" );
```

```
getch();
```

```
cleardevice(); /* Clear the graphics screen */
```

```
}
```

```
/*
```

```
*/
```

```
/* PAUSE: Pause until the user enters a keystroke. If the
```

```
*/
```

```
/* key is an ESC, then exit program, else simply return.
```

```
*/
```

```
/*
```

```
*/
```

```
void Pause(void)
```

```
{
```

```
static char msg[] = "Esc aborts or press a key...";
```

```
int c;
```

```
StatusLine( msg ); /* Put msg at bottom of
```

```
screen */
```

```
c = getch(); /* Read a character from
```

```
kbd */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if( ESC == c ){                                /* Does user wish to leave?
    */
    closegraph();                               /* Change to text mode    */
    exit( 1 );                                  /* Return to OS
    */
}

if( 0 == c ){                                  /* Did use hit a non-ASCII key? */
    c = getch();                               /* Read scan code for keyboard */
}

cleardevice();                                 /* Clear the screen
*/
}
/*
*/
/* MAINWINDOW: Establish the main window for the demo
and set */
/* a viewport for the demo code.
*/
/*
*/
*/

```

```

void MainWindow( char *header )

```

```

{

```

```

    int height;

```

```

    cleardevice();                               /* Clear graphics screen */

```

```

    setcolor( MaxColors - 1 );                   /* Set current color to white

```

```

    */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

setviewport( 0, 0, MaxX, MaxY, 1 ); /* Open port to full screen
*/

height = textheight( "H" );          /* Get basic text height */

/* changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );*/
changetextstyle( SMALL_FONT, HORIZ_DIR, 4);
settextjustify( CENTER_TEXT, TOP_TEXT );
outtextxy( MaxX/2, 2, header );
setviewport( 0, height+4, MaxX, MaxY-(height+4), 1 );
DrawBorder();
setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
}

/*
*/
/* STATUSLINE: Display a status line at the bottom of the
screen. */
/*
*/

void StatusLine( char *msg )
{
    int height;

    setviewport( 0, 0, MaxX, MaxY, 1 ); /* Open port to full screen
*/

    setcolor( MaxColors - 1 );          /* Set current color to white
*/

    changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );

```

```

settextjustify( CENTER_TEXT, TOP_TEXT );
setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
setfillstyle( EMPTY_FILL, 0 );

height = textheight( "H" );      /* Determine current height
*/

bar( 0, MaxY-(height+4), MaxX, MaxY );
rectangle( 0, MaxY-(height+4), MaxX, MaxY );
outtextxy( MaxX/2, MaxY-(height+2), msg );
setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );

}

/*
*/
/* DRAWBORDER: Draw a solid single line around the current
*/
/* viewport.
*/
*/

void DrawBorder(void)
{
    struct viewporttype vp;

    setcolor( MaxColors - 1 );      /* Set current color to white
    */

    setlinestyle( SOLID_LINE, 0, NORM_WIDTH );

    getviewsettings( &vp );

```

```
rectangle( 0, 0, vp.right-vp.left, vp.bottom-vp.top );
```

```
}
```

```
/*
```

```
*/
```

```
/*  CHANGETEXTSTYLE: similar to settextstyle, but checks
```

```
for  */
```

```
/*  errors that might occur while loading the font file.  */
```

```
/*
```

```
*/
```

```
void changetextstyle(int font, int direction, int charsize)
```

```
{
```

```
int ErrorCode;
```

```
graphresult(); /* clear error code */
```

```
settextstyle(font, direction, charsize);
```

```
ErrorCode = graphresult(); /* check result */
```

```
if( ErrorCode != grOk ){ /* if error occurred
```

```
*/
```

```
closegraph();
```

```
printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode
));
```

```
exit( 1 );
```

```
}
```

```
}
```

```
/*
```

```
*/
```

```
/*  GPRINTF: Used like PRINTF except the output is sent to the
```

```
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/*      screen in graphics mode at the specified co-ordinate.
      */
/*
      */

int gprintf( int *xloc, int *yloc, char *fmt, ... )
{
    va_list argptr;                /* Argument list pointer */
    char str[140];                  /* Buffer to build sting into */
    int cnt;                        /* Result of SPRINTF for
return */

    va_start( argptr, format );     /* Initialize va_ functions
    */

    cnt = vsprintf( str, fmt, argptr ); /* prints string to buffer */
    outtextxy( *xloc, *yloc, str ); /* Send string in graphics mode */
    *yloc += textheight( "H" ) + 2; /* Advance to next line
    */

    va_end( argptr );               /* Close va_ functions
    */

    return( cnt );                  /* Return the conversion count */
}
~

```

ภาคผนวก ข

ภาค LCD Display

ในการแสดงผลของค่าสัญญาณที่ได้จาก Gas Sensor เราจะต่อ LCD MODULE เข้ากับระบบไมโครคอนโทรลเลอร์ 8031 ซึ่งจะต่อได้ 2 ลักษณะคือ แบบ MEMORY MAP โดยผ่าน LCD BUS ขนาด 20 PIN และแบบ I/O PORT โดยผ่าน 8255 BUS ขนาด 26 PIN ซึ่งทั้งสองแบบนี้จะมีข้อดีและข้อด้อยแตกต่างกันไป โดยแต่ละแบบจะมีหลักการดังนี้

การต่อแบบ MEMORY MAP

1. สามารถต่อเข้ากับ CHIP เบอร์ทั่ว ๆ ไปได้ เช่น 8051 หรือ Z80 โดยจะให้ระบบไมโครมองเห็น LCD MODULE ในลักษณะของ MEMORY ได้ทันที
2. ผู้ใช้สามารถเขียนและอ่านข้อมูลได้จาก LCD MODULE ได้ ทำให้ง่ายเหมือนว่าเป็น MEMORY BUFFER ไปในตัว
3. เนื่องจากสามารถอ่านข้อมูลกลับได้ จึงทำให้สามารถตรวจสอบ FLAG ความพร้อมในขณะที่ LCD MODULE กำลังทำงานได้
4. ใช้ได้กับบอร์ดที่มี LCD BUS มาให้พร้อมเท่านั้น
5. ทำให้กินพื้นที่ของหน่วยความจำไปส่วนหนึ่ง และต้องมีการ DECODE ละเอียดพอสมควร
6. การจัดขาสัญญาณจะต้องเป็นไปตามแบบของ CHIP แต่ละเบอร์ด้วย

การต่อแบบ I/O PORT

1. สามารถต่อเข้ากับ I/O PORT ใด ๆ ก็ได้ โดยใช้สายสัญญาณจำนวน 11 เส้น และใช้โปรแกรมเป็นตัวสร้างสัญญาณขึ้นมาให้ตรงกับข้อกำหนดของ LCD MODULE
2. ผู้ใช้จะเขียนข้อมูลให้ LCD MODULE ได้อย่างเดียว ซึ่งผู้ใช้ควรจะกำหนด MEMORY ส่วนหนึ่งให้เป็นเสมือน BUFFER ให้กับ LCD MODULE อีกที
3. เนื่องจากไม่สามารถอ่านข้อมูลกลับได้ จึงต้องใช้การหน่วงเวลาของระบบไมโครเอง เพื่รอให้ LCD MODULE กระทำขบวนการต่าง ๆ

4. ใช้ได้กับบอร์ดทั่ว ๆ ไปที่มี PORT
5. ไม่เปลืองส่วนของ MEMORY ในการใช้งาน
6. การจัดขาสัญญาณกระทำได้อย่างอิสระ

ในการใช้งานของ Project ได้เลือกใช้ชุด LCD MODULE เบอร์ DMC202 ซึ่งเป็น LCD 2 บรรทัด 20 ตัวอักษร แบบ Dot Matrix ขนาด 5*8 Dot โดยหลักการแล้วเราควรที่จะเข้าใจหลักการพื้นฐานของการเขียนชุดคำสั่งเพื่อใช้งานต่อไปซึ่งมีดังนี้

1. การเขียนข้อมูลให้กับ LCD MODULE จะแบ่งเป็น 2 ลักษณะคือ INSTRUCTION และ DATA โดยจะกำหนดด้วยขาสัญญาณ RS ก็คือถ้า RS=0 จะหมายถึงส่งสัญญาณควบคุม (INSTRUCTION) หรืออ่านค่า FLAG สภาพการทำงานของ LCD MODULE และถ้า RS=1 จะหมายถึงการเขียนหรืออ่าน DATA กับ LCD MODULE

2. หลักการในการเขียนข้อมูลให้ LCD MODULE นี้ คือ เมื่อมีการเขียนข้อมูลไปแล้ว ตัว LCD MODULE จะต้องใช้เวลาในการทำงานชั่วขณะหนึ่ง (ตามค่า EXECUTE TIME ในตาราง) ซึ่งระบบไมโครสามารถตรวจสอบได้จาก BUSY FLAG (BF) และถ้าเรียบร้อยแล้ว จึงจะสามารถเขียนข้อมูลอันต่อไปได้ ในกรณีที่การต่อวงจรเป็นแบบ I/O PORT คือไม่สามารถอ่านข้อมูลย้อนกลับได้ ระบบไมโครก็จะต้องใช้วิธีการหน่วงเวลาแทน

3. การเขียนข้อมูลให้กับ LCD MODULE นี้ สามารถทำได้ทั้งแบบ 8 BIT และ 4 BIT โดยกรณี 4 BIT จะใช้สายสัญญาณ DATA เพียง 4 เส้น คือ DB4 - DB7 (ใช้สำหรับระบบไมโครแบบ 4 BIT หรือเพื่อการประหยัดสาย) การเขียนข้อมูลจะกระทำเหมือนกับ 8 BIT เพียงแต่ให้เขียน 2 ครั้ง คือ DB4 - DB7 ก่อน แล้วตามด้วย DB0 - DB3 และจะต้องกำหนดคุณสมบัติตามค่า DL ในคำสั่ง FUNCTION SET ด้วย

4. DDRAM (DISPLAY DATA RAM) คือหน่วยความจำภายในตัว LCD MODULE ที่เป็น BUFFER ของข้อมูล โดยถ้าเขียนรหัส ASCII ใด ๆ ลงไปในหน่วยความจำนี้ ก็จะปรากฏเป็นตัวอักษรที่แสดงผลทันที

5. CGRAM (CHARACTER GENERATOR RAM) คือหน่วยความจำภายในตัว LCD MODULE สำหรับเก็บภาพตัวอักษรที่ผู้ใช้สามารถสร้างได้เอง (8 ตัว) โดยจะอ้าง ADDRESS ได้ทั้งหมด 64 BYTE คือ 8 ตัวอักษร คูณกับ 8 ROW

แนวทางการเขียนโปรแกรมควบคุม

1. เมื่อจ่ายไฟเลี้ยงให้กับ LCD MODULE ครั้งแรก ภายในจะมีการ RESET ระบบ โดยอัตโนมัติ ซึ่งจะใช้เวลา 10 ms หลังจากทีระดับแรงไฟขึ้นถึง 4.5 V แล้ว ทั้งนี้ระบบ RESET ดังกล่าวจะกระทำสิ่งต่าง ๆ ดังต่อไปนี้

- ทำการ CLEAR จอภาพทั้งหมด (CLEAR DISPLAY)
- กำหนดคุณสมบัติด้วยคำสั่ง FUNCTION SET คือ DL = 1 (ติดต่อกับระบบ

ไมโครในแบบ 8

BIT), N = 0 (แสดงข้อมูล 1 บรรทัด), F = 0 (กำหนดตัวอักษรแบบ 5*7 DOTS)

- กำหนดคุณสมบัติด้วยคำสั่ง DISPLAY ON/OFF คือ D = 0 (ไม่แสดงข้อมูล), C =

0

(CURSOR OFF), B = 0 (BLINK OFF)

- กำหนดคุณสมบัติด้วยคำสั่ง ENTER MODE SET คือ I/O = 1 (INCREMENT), S

= 0 (NO

SHIFT)

การใช้งาน LCD MODULE ต้องรอให้ขบวนการ RESET ภายในทำงานเรียบร้อย ก่อน ซึ่งจะตรวจสอบได้ด้วย BF (BUSY FLAG) หรืออาจจะใช้การหน่วงเวลาก็ได้

2. การใช้งาน LCD MODULE จะเกี่ยวข้องกับทางด้านโปรแกรมเป็นส่วนใหญ่ ชุดคำสั่งต่าง ๆ รวมทั้งการอ่านหรือการเขียนข้อมูลนั้น จะถูกกำหนดด้วยขาสัญญาณทั้งหมดที่มีอยู่ ปกติโปรแกรมจะต้องกำหนดคุณสมบัติต่าง ๆ ที่ต้องการไว้ที่ส่วนต้น และจากนั้นก็จะเป็นการอ่านและเขียนข้อมูลลงใน DDRAM ซึ่งก็คือข้อความที่จะให้แสดงนั่นเอง รายละเอียดตัวโปรแกรมขอให้ศึกษาได้จากตัวอย่าง

ภาคผนวก ค

พอร์ตอนุกรมและมาตรฐาน RS-232

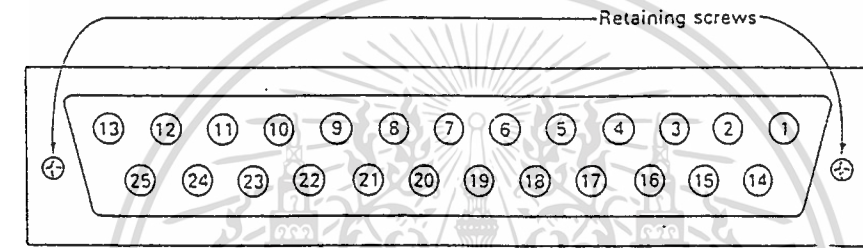
(SERIAL PORTS AND THE RS-232 STANDARD)

พอร์ตอนุกรม

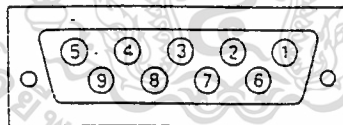
พอร์ตอนุกรมส่วนใหญ่จะมีรูปร่างขึ้นอยู่กัมาตรฐาน RS-232 คือ มี 25 ขา แต่พอร์

ตอนุกรม

ของ IBM-PC AT จะมีเพียง 9 ขา เพราะว่าสัญญาณบางขาไม่จำเป็นต้องใช้



RS-232 "D-type" connector (Socket)



Front view

Pin	1	DCD	(8)
	2	RD	(3)
	3	TD	(2)
	4	DTR	(20)
	5	SG	(7)
	6	DSR	(6)
	7	RTS	(4)
	8	CTS	(5)
	9	RI	(22)

IBM-PC/AT 9-pin RS-232 (Socket)

รูปที่ ค.1 แสดงขาของพอร์ตอนุกรมแบบ 25 ขาและ 9 ขา

D8 - 9 Pin	D8 - 25 Pin	Assignment/function
1	8	Carrier detect
2	3	Receive data
3	2	Transmit data
4	20	Data terminal ready
5	7	Signal ground
6	6	Data set ready
7	4	Request to send
8	5	Clear to send
9	22	Ring indicator

มาตรฐาน RS-232

ตารางที่ ค.1

RS-232 (Recommended Standard 232) มีชื่อเต็มว่า "Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interface" มาตรฐาน RS-232 ถูกกำหนดขึ้นโดย EIA (Electronic Industries Association) เพื่อความสะดวกในความเข้ากันได้ (compatibility) และการติดต่อกัน (inchangeability) ระหว่างคอมพิวเตอร์ และอุปกรณ์รอบข้างต่าง ๆ (peripheral devices) โดยผ่านทางพอร์ตอนุกรม

ตารางที่ ค.2

Voltage	Logic	Control	Signal Condition
+ 3 V to + 25 V	0	On	Space
- 3 V to - 25 V	1	Off	Mark

RS-232 จะใช้ระดับสัญญาณแบบ negative-logic levels หมายถึงศักดาไฟฟ้าค่าบวก (+3 V ถึง +25 V) ที่มากกว่าในวงจรเป็นลอจิก "0" และศักดาไฟฟ้าลบ (-3 V ถึง -25 V) เป็นลอจิก "1" ซึ่งตรงข้ามกับสัญญาณทางลอจิกแบบ TTL จะถือว่าศักดาไฟฟ้าที่เป็นบวกมากกว่ามีค่าทางลอจิกเป็น "1" และศักดาไฟฟ้าที่เป็นลบมากกว่ามีค่าทางลอจิกเป็น "0" ค่าแรงดันไฟฟ้าของ RS-232 ที่อยู่ระหว่างช่วง +3V และ -3 V จะมีความหมายเหมือนกับสัญญาณไฟฟ้าแบบ TTL ที่อยู่ในช่วง 0.8 V และ 2 V แรงดันไฟฟ้า +12 V และ -12 V เป็นระดับแรงดันไฟฟ้าที่ใช้กันทั่วไปในมาตรฐาน RS-232 ในขณะที่ RS-232 ไม่ได้ส่งข้อมูลสายส่ง (ขา Transmitted Data) จะมีค่าทางลอจิกเป็น "1"

ภาคผนวก ง

DATA SHEET

ICL7109

- 12 bit binary (plus polarity and overrange) dual slope integrating analog-to-digital converter.
- Byte-organized TTL-compatible three-state outputs and UART handshake mode for simple parallel or serial interfacing to microprocessor systems.
- RUN/HOLD input and STATUS output can be used to monitor and control conversion timing.
- True differential input and differential reference.
- Low noise-typically $15\mu\text{V}$ peak-to-peak.
- 1pA typical input current.
- Operates at up to 30 conversions per second.
- On-chip oscillator operates with inexpensive 3.58MHz TV crystal giving 7.5 conversions per second for 60Hz rejection, or may be operated as an RC oscillator for other clock frequencies.
- Fabricated using MAX-CMOS™ technology combining analog and digital functions on a single low power LSI CMOS chip.
- All inputs fully protected against static discharge; no special handling precautions necessary.

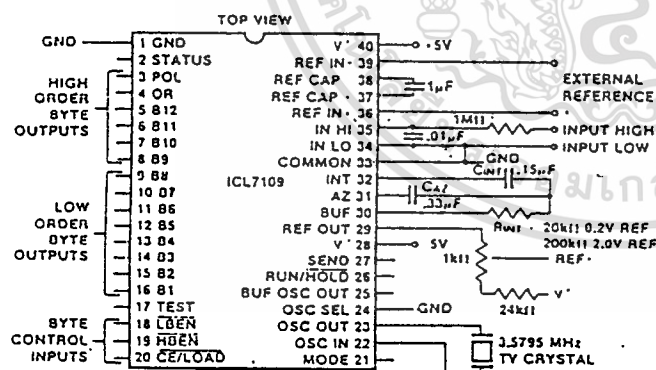
The ICL7109 is a high performance, low power integrating A/D converter designed to easily interface to microprocessors.

The output data (12 bits, polarity and overrange) may be directly accessed under control of two byte enable inputs and a chip select input for a simple parallel bus interface. A UART handshake mode is provided which allows the ICL7109 to work with industry-standard UARTs to provide serial data transmission, ideal for remote data logging applications. The RUN/HOLD input and STATUS output allow monitoring and control of conversion timing.

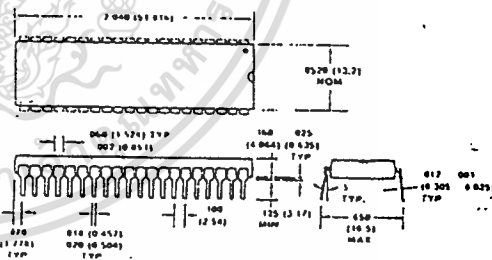
The ICL7109 provides the user the high-accuracy, low noise, low drift, versatility and economy of the dual-slope integrating A/D converter. Features like true differential input and reference, zero drift of less than $1\mu\text{V}/^\circ\text{C}$ max., input bias current of 10pA max., and typical power consumption of 20mW make the ICL7109 an attractive per-channel alternative to analog multiplexing for many data acquisition applications.

Note: A MIL-STD (883 processing) version of the ICL7109 will be available July 1979.

*See Figure 1 for typical connection to a UART or Microcomputer.



40 Pin Plastic Dual-in-Line Package



Part	Package	Temp. Range	Order Part #
7109	40 pin ceramic DIP	25°C to +85°C	ICL7109IDL
7109	40 pin plastic DIP	0°C to +70°C	ICL7109CPL

ICL7109

ABSOLUTE MAXIMUM RATINGS

Positive Supply Voltage (GND to V^+)	+6.2V
Negative Supply Voltage (GND to V^-)	-9V
Analog Input Voltage (Lo or Hi) (Note 1)	V^+ to V^-
Reference Input Voltage (Lo or Hi) (Note 1)	V^+ to V^-
Digital Input Voltage	$V^+ \pm 0.3V$
(Pins 2-27) (Note 2)	GND-0.3V
Power Dissipation (Note 3)	
Ceramic or Cerdip Package	1W @ 85°C
Plastic Package	500mW @ 70°C
Operating Temperature	
Ceramic or Cerdip Package	25°C $\leq T_A \leq$ 85°C
Plastic Package	0°C $\leq T_A \leq$ 70°C
Storage Temperature	55°C $\leq T_A \leq$ 125°C
Lead Temperature (soldering, 60 sec)	300°C

Absolute maximum ratings define stress limitations which if exceeded may permanently damage the device. These ratings are not continuous duty ratings. For continuous operation these devices must be operated under the conditions defined under "Operating Characteristics."

TABLE I OPERATING CHARACTERISTICS

All parameters with $V^+ = +5V$, $V^- = 0V$, $T_A = 25^\circ C$, unless otherwise indicated.
Test circuit as shown on page 1.

ANALOG SECTION

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Zero Input Reading		$V_{IN} = 0.0V$ Full scale = 409.6mV	0000 _L	10000 _g	+0000 _g	Octal Reading
Ratiometric Reading		$V_{IN} = V_{REF}$ $V_{REF} = 204.8mV$	3777 _g	3777 _g 4000 _g	4000 _g	Octal Reading
Non-Linearity (Max deviation from best straight line fit)		Full scale = 409.6mV or 4.096V	-1	± 2	+1	Counts
Roll-over Error (disturbance in reading for equal pos. and neg. inputs near full scale.)			1	± 2	+1	Counts
Common Mode Rejection Ratio		$V_{CM} \pm 1V$ $V_{IN} = 0V$ Full Scale = 409.6mV		50		$\mu V/V$
Noise (p-p value not exceeded 95% of time)		$V_{IN} = 0V$ Full Scale = 409.6mV		15		μV
Leakage Current at Input		$V_{IN} = 0V$		1	10	pA
Zero Reading Drift		$V_{IN} = 0V$		0.2	1	$\mu V/^\circ C$
Scale Factor Temperature Coefficient		$V_{IN} = 408.9mV$ 7770 _g reading Ext. Ref. 0 ppm/ $^\circ C$		1	5	ppm/ $^\circ C$
Supply Current V^+ to GND	I_{OL}	$V_{IN} = 0$, Crystal Osc. 3.58MHz test circuit Pins 2-21, 25, 26, 27, 29, open		700	1500	μA
Supply Current V^+ to V^-	I_{OA}			700	1500	μA
Ref Out Voltage		Referred to V^+ , 25k Ω between V^+ and REF OUT	-2.4	-2.8	-3.2	V
Ref Out Temp. Coefficient		25k Ω between V^+ and REF OUT		80		ppm/ $^\circ C$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ICL7109

INTERSI

DIGITAL SECTION

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Output High Voltage	V_{OH}	$I_{OUT} = 100\mu A$ Pins 2-16, 18, 19, 20	3.5	4.3		V
Output Low Voltage	V_{OL}	$I_{OUT} = 1.6mA$		0.2	0.4	V
Output Leakage Current		Pins 3-16 high impedance		± 0.1	± 1	μA
Control I/O Pullup Current		Pins 18, 19, 20 $V_{OUT} = V^- - 3V$ MODE input at GND		5		μA
Control I/O Loading		HBEN Pin 19 LBEN Pin 18			50	pF
Input High Voltage	V_{IH}	Pins 18-21, 26, 27 referred to GND	2.5			V
Input Low Voltage	V_{IL}	Pins 18-21, 26, 27 referred to GND			1	V
Input Pull-up Current		Pins 26, 27 $V_{OUT} = V^- - 3V$		5		μA
Input Pull-up Current		Pins 17, 24 $V_{OUT} = V^- - 3V$		25		μA
Input Pull-down Current		Pin 21 $V_{OUT} = GND$ 13V		5		μA
Oscillator Output Current	High	O_{OH} $V_{OUT} = 2.5V$		1		mA
	Low	O_{OL} $V_{OUT} = 2.5V$		1.5		mA
Buffered Oscillator Output Current	High	BO_{OH} $V_{OUT} = 2.5V$		2		mA
	Low	BO_{OL} $V_{OUT} = 2.5V$		5		mA
MODE Input Pulse Width			50			ns

Note 1: Input voltages may exceed the supply voltages provided the input current is limited to $100\mu A$

Note 2: Due to the SCR structure inherent in the process used to fabricate these devices, connecting any digital inputs or outputs to voltages greater than V^+ or less than GND may cause destructive device latchup. For this reason it is recommended that no inputs from sources not on the same power supply be applied to the ICL7109 before its power supply is established, and that in multiple supply systems the supply to the ICL7109 be activated first.

Note 3: This limit refers to that of the package and will not be obtained during normal operation.

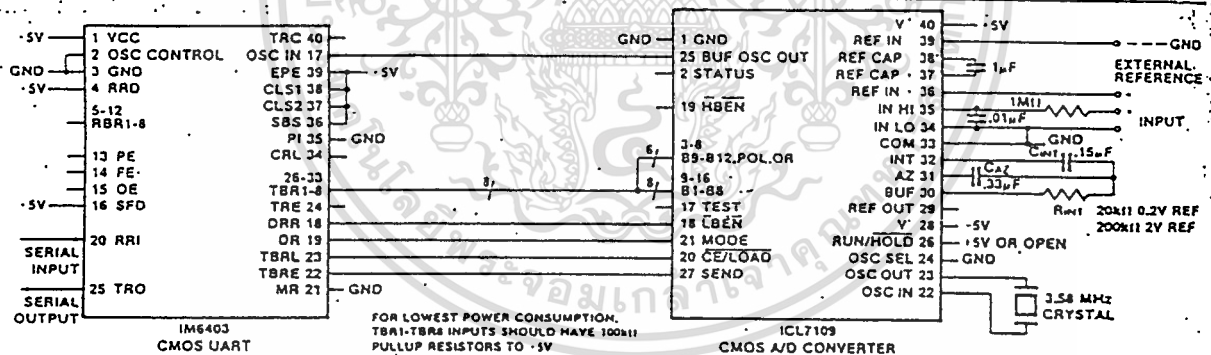


Figure 1A. To transmit latest result, send any word to UART.

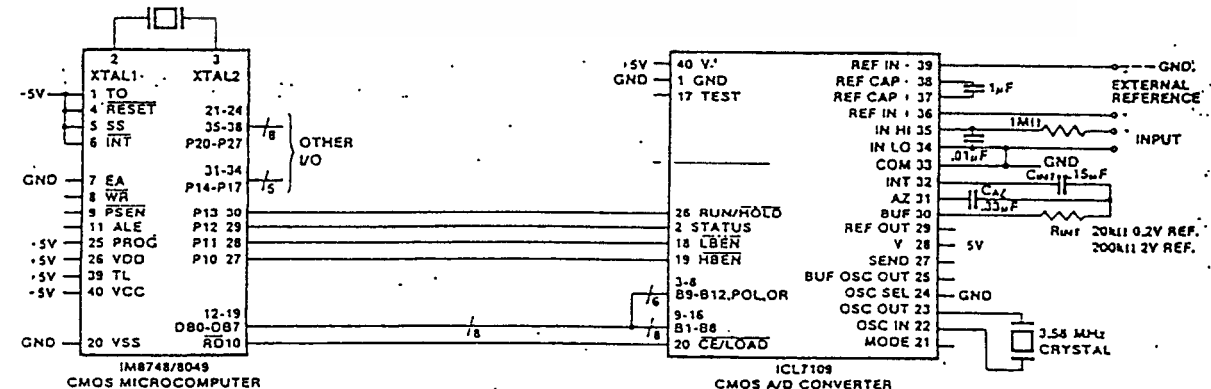


Figure 1B. Typical Connection Diagram Parallel Interface With 8748/8048 Microcomputer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ICL7109

INTERSiL

TABLE 2 - Pin Assignment and Function Description

PIN	SYMBOL	DESCRIPTION
1	GND	Digital Ground, 0V, Ground return for all digital logic
2	STATUS	Output - High during integrate and deintegrate until data is latched. - Low when analog section is in Auto-Zero configuration.
3	POL	Polarity, Three-State Output
4	OR	Over-range, Three-State Output
5	B12	Bit 12 (Most Significant Bit)
6	B11	Bit 11
7	B10	Bit 10
8	B9	Bit 9
9	B8	Bit 8
10	B7	Bit 7
11	B6	Bit 6
12	B5	Bit 5
13	B4	Bit 4
14	B3	Bit 3
15	B2	Bit 2
16	B1	Bit 1 (Least Significant Bit)
17	TEST	Input High - Normal Operation. Input Low - Forces all bit outputs high. Note: This input is used for test purposes only.
18	LBEN	Low Byte Enable - With Mode (Pin 21) low, and CE/LOAD (Pin 20) low, taking this pin low activates low order byte outputs B1-B8. - With Mode (Pin 21) high, this pin serves as a low byte flag output used in handshake mode. See Figures 7, 8, 9.
19	HBEN	High Byte Enable - With Mode (Pin 21) low, and CE/LOAD (Pin 20) low, taking this pin low activates high order byte outputs B9-B12, POL, OR. - With Mode (Pin 21) high, this pin serves as a high byte flag output used in handshake mode. See Figures 7, 8, 9.
20	CE/LOAD	Chip Enable Load - With Mode (Pin 21) low, CE/LOAD serves as a master output enable. When high, B1-B12, POL, OR outputs are disabled. - With Mode (Pin 21) high, this pin serves as a load strobe used in handshake mode. See Figures 7, 8, 9.

PIN	SYMBOL	DESCRIPTION
21	MODE	Input Low - Direct output mode where CE/LOAD (Pin 20), HBEN (Pin 19) and LBEN (Pin 18) act as inputs directly controlling byte outputs. Input Pulsed High - Causes immediate entry into handshake mode and output of data as in Figure 9. Input High - Enables CE/LOAD (Pin 20), HBEN (Pin 19), and LBEN (Pin 18) as outputs, handshake mode will be entered and data output as in Figures 7 and 8 at conversion completion.
22	OSC IN	Oscillator Input
23	OSC OUT	Oscillator Output
24	OSC SEL	Oscillator Select - Input high configures OSC IN, OSC OUT, BUF OSC OUT as RC oscillator - clock will be same phase and duty cycle as BUF OSC OUT. - Input low configures OSC IN, OSC OUT for crystal oscillator - clock frequency will be 1/58 of frequency at BUF OSC OUT.
25	BUF OSC OUT	Buffered Oscillator Output
26	RUN/HOLD	Input High - Conversions continuously performed every 8192 clock pulses Input Low - Conversion in progress completed, converter will stop in Auto-Zero 7 counts before integrate.
27	SEND	Input - Used in handshake mode to indicate ability of an external device to accept data.
28	V	Analog Negative Supply - Nominally -5V with respect to GND (Pin 1).
29	REF OUT	Reference Voltage Output - Nominally 2.8V down from V ⁺ (Pin 40).
30	BUFFER	Buffer Amplifier Output
31	AUTO-ZERO	Auto-Zero Node - Inside foil of C _{AZ}
32	INTEGRATOR	Integrator Output - Outside foil of C _{INT}
33	COMMON	Analog Common - System is Auto-Zeroed to COMMON
34	INPUT LO	Differential Input Low Side
35	INPUT HI	Differential Input High Side
36	REF IN	Differential Reference Input Positive
37	REF CAP	Reference Capacitor Positive
38	REF CAP	Reference Capacitor Negative
39	REF IN	Differential Reference Input Negative
40	V	Positive Supply Voltage - Nominally +5V with respect to GND (Pin 1).

DETAILED DESCRIPTION

Analog Section

Figure 2 shows the equivalent circuit of the Analog Section of the ICL7109. When the RUN/HOLD input is left open or connected to V⁺, the circuit will perform conversions at a rate determined by the clock frequency (8192 clock periods per cycle). Each measurement cycle is divided into three phases as shown in Figure 3. They are (1) Auto-Zero (AZ), (2) Signal Integrate (INT) and (3) Deintegrate (DE).

Auto-Zero Phase

During auto-zero three things happen. First, input high and low are disconnected from their pins and internally shorted to analog common. Second, the reference capacitor is charged to the reference voltage. Third, a feedback loop is closed around the system to charge the auto-

zero capacitor C_{AZ} to compensate for offset voltages in the buffer amplifier, integrator, and comparator. Since the comparator is included in the loop, the AZ accuracy is limited only by the noise of the system. In any case, the offset referred to the input is less than 10μV.

2. Signal Integrate Phase

During signal integrate the auto-zero loop is opened, the internal short is removed and the internal input high and low are connected to the external pins. The converter then integrates the differential voltage between input high and input low for a fixed time of 2048 clock periods. At the end of this phase, the polarity of the integrated signal is determined.

ICL7109

INTERMIL

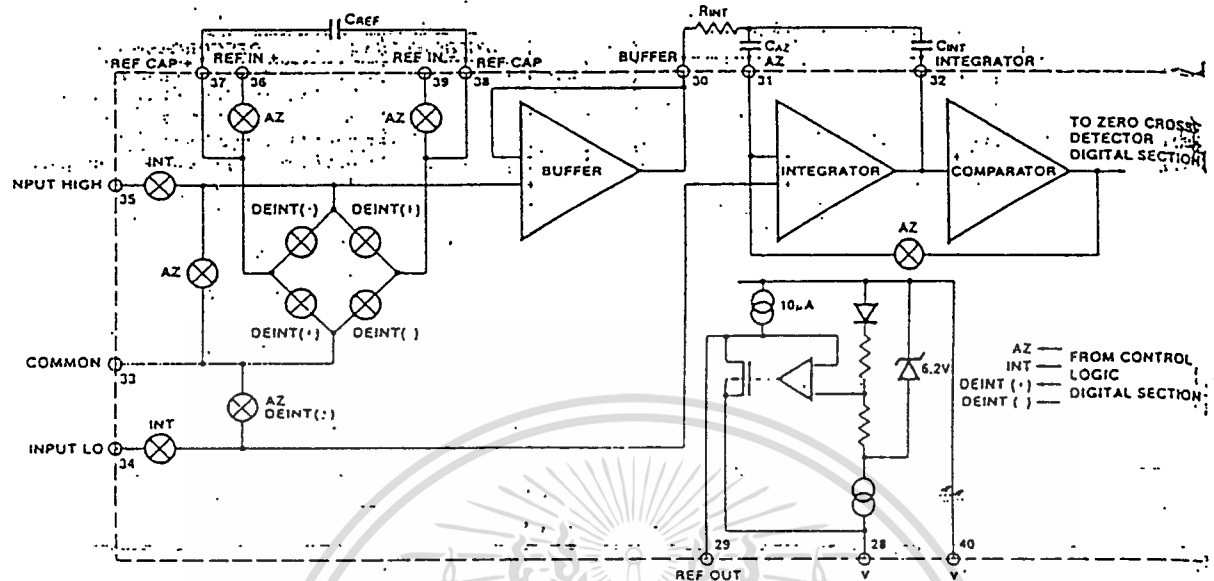


Figure 2: Analog Section

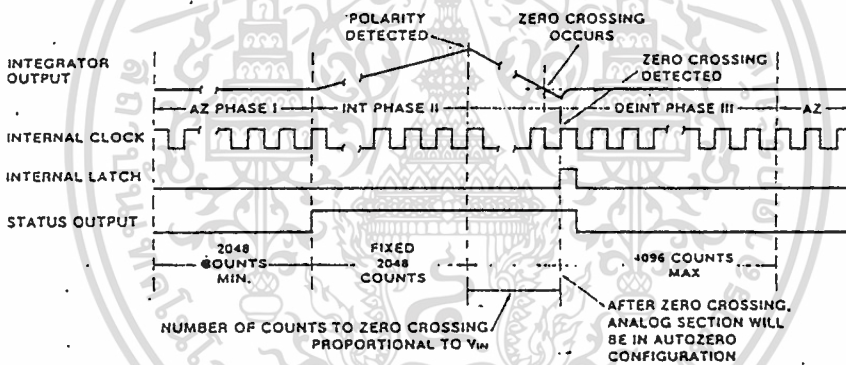


Figure 3: Conversion Timing

3. Deintegrate Phase

The final phase is deintegrate, or reference integrate. Input low is internally connected to analog common and input high is connected across the previously charged (during auto-zero) reference capacitor. Circuitry within the chip ensures that the capacitor will be connected with the correct polarity to cause the integrator output to return to the zero crossing (established in Auto Zero) with a fixed slope. Thus the time for the output to return to zero (represented by the number of clock periods counted) is proportional to the input signal.

Differential Input

The input can accept differential voltages anywhere within the common mode range of the input amplifier; or specifically from 0.5 volts below the positive supply to 1.0 volt above the negative supply. In this range the system has a CMRR of 86dB typical. However, since the integrator also swings with the common mode voltage, care must be exercised to assure the integrator output does not saturate. A worst case condition would be a large positive common mode voltage with a near full-scale negative differential input voltage. The negative input signal drives the integrator

positive when most of its swing has been used up by the positive common mode voltage. For these critical applications the integrator swing can be reduced to less than the recommended 4V full scale with some loss of accuracy. The integrator output can swing within 0.3 volts of either supply without loss of linearity.

The ICL7109 has, however, been optimized for operation with analog common near digital ground. With power supplies of +5V and -5V, this allows a 4V full scale integrator swing positive or negative maximizing the performance of the analog section.

Differential Reference

The reference voltage can be generated anywhere within the power supply voltage of the converter. The main source of common mode error is a roll-over voltage caused by the reference capacitor losing or gaining charge to stray capacity on its nodes. If there is a large common mode voltage, the reference capacitor can gain charge (increase voltage) when called up to deintegrate a positive signal but lose charge (decrease voltage) when called up to deintegrate a negative input signal. This difference in reference for (+) or (-) input voltage will give a roll-over error. However, by

ICL7109

selecting the reference capacitor large enough in comparison to the stray capacitance, this error can be held to less than 0.5 count for the worst case condition (see Component Values Selection below).

The roll-over error from these sources is minimized by having the reference common mode voltage near or at analog common.

Component Value Selection

For optimum performance of the analog section, care must be taken in the selection of values for the integrator capacitor and resistor, auto-zero capacitor, reference voltage, and conversion rate. These values must be chosen to suit the particular application.

The most important consideration is that the integrator output swing (for full-scale input) be as large as possible. For example, with $\pm 5V$ supplies and COMMON connected to GND, the nominal integrator output swing at full scale is $\pm 4V$. Since the integrator output can go to 0.3V from either supply without significantly affecting linearity, a 4V integrator output swing allows 0.7V for variations in output swing due to component value and oscillator tolerances. With $\pm 5V$ supplies and a common mode range of $\pm 1V$ required, the component values should be selected to provide $\pm 3V$ integrator output swing. Noise and rollover errors will be slightly worse than in the $\pm 4V$ case. For larger common mode voltage ranges, the integrator output swing must be reduced further. This will increase both noise and rollover errors. To improve the performance, supplies of $\pm 6V$ may be used.

4. Integrating Resistor

Both the buffer amplifier and the integrator have a class A output stage with $100\mu A$ of quiescent current. They supply $20\mu A$ of drive current with negligible non-linearity. The integrating resistor should be large enough to remain in this very linear region over the input voltage range, but small enough that undue leakage requirements are not placed on the PC board. For 4.096V full scale, $200k\Omega$ is near optimum and similarly a $20k\Omega$ for a 409.6mV scale. For other values of full scale voltage, R_{INT} should be chosen by the relation $R_{INT} = \frac{\text{full scale voltage}}{20\mu A}$.

2. Integrating Capacitor

The integrating capacitor C_{INT} should be selected to give the maximum integrator output voltage swing without saturating the integrator (approximately 0.3 volt from either supply). For the ICL7109 with $\pm 5V$ supplies and analog common connected to GND, a ± 3.5 to ± 4 volt integrator output swing is nominal. For 7-1/2 conversions per second (61.72KHz clock frequency) as provided by the crystal oscillator, nominal values for C_{INT} and C_{AZ} are $0.15\mu F$ and $0.33\mu F$, respectively. If different clock frequencies are used, these values should be changed to maintain the integrator output voltage swing. In general, the value of C_{INT} is given by

$$C_{INT} = \frac{(2048 \times \text{clock period}) (20\mu A)}{\text{integrator output voltage swing}}$$

An additional requirement of the integrating capacitor is that it have low dielectric absorption to prevent roll-over errors. While other types of capacitors are adequate for this application, polypropylene capacitors give undetectable errors at reasonable cost.

3. Auto-Zero Capacitor

The size of the auto-zero capacitor has some influence on the noise of the system; a big capacitor giving less noise. However, it cannot be increased without limits since it, in parallel with the integrating capacitor forms an R-C time constant that determines the speed of recovery from overloads and more important the error that exists at the end of an auto-zero cycle. For 409.6mV full scale where noise is very important and the integrating resistor small, a value of C_{AZ} twice C_{INT} is optimum. Similarly for 4.096V full scale where recovery is more important than noise, a value of C_{AZ} equal to half of C_{INT} is recommended.

For optimal rejection of stray pickup, the outer foil of C_{AZ} should be connected to the R-C summing junction and the inner foil to pin 31. Similarly the outer foil of C_{INT} should be connected to pin 32 and the inner foil to the R-C summing junction.

4. Reference Capacitor

A $1\mu F$ capacitor gives good results in most applications. However, where a large common mode voltage exists (i.e. the reference low is not at analog common) and a 409.6mV scale is used, a larger value is required to prevent roll-over error. Generally $10\mu F$ will hold the roll-over error to 0.5 count in this instance.

5. Reference Voltage

The analog input required to generate a full scale output of 4096 counts is $V_{IN} = 2V_{REF}$. Thus for a normalized scale, a reference of 2.048V should be used for a 4.096V full scale, and 204.8mV should be used for a 0.4096V full scale. However, in many applications where the A/D is sensing the output of a transducer, there will exist a scale factor other than unity between the absolute output voltage to be measured and a desired digital output. For instance, in a weighing system, the designer might like to have a full scale reading when the voltage from the transducer is 0.682V. Instead of dividing the input down to 409.6mV, the input voltage should be measured directly and a reference voltage of 0.341V should be used. Suitable values for integrating resistor and capacitor are $34k\Omega$ and $0.15\mu F$. This avoids a divider on the input. Another advantage of this system occurs when a zero reading is desired for non-zero input. Temperature and weight measurements with an offset or tare are examples. The offset may be introduced by connecting the voltage output of the transducer between common and analog high, and the offset voltage between common and analog low, observing polarities carefully. However, in processor-based systems using the ICL7109, it may be more efficient to perform this type of scaling or tare subtraction digitally using software.

6. Reference Sources

The stability of the reference voltage is a major factor in the overall absolute accuracy of the converter. The resolution of the ICL7109 at 12 bits is one part in 4096, or 244ppm. Thus if the reference has a temperature coefficient of $80\text{ppm}/^\circ\text{C}$ (onboard reference) a temperature difference of 3°C will introduce a one-bit absolute error. For this reason, it is recommended that an external high-quality reference be used where the ambient temperature is not controlled or where high-accuracy absolute measurements are being made.

ICL7109

The ICL7109 provides a Reference Output (pin 29) which may be used with a resistive divider to generate a suitable reference voltage. This output will sink up to about 20mA without significant variation in output voltage, and is provided with a pullup bias device which sources about 10 μ A. The output voltage is nominally 2.8V below V^+ , and has a temperature coefficient of ± 80 ppm/ $^{\circ}$ C typ. When using the onboard reference, Ref Out (Pin 29) should be connected to Ref- (pin 39), and Ref+ should be connected to the wiper of a precision potentiometer between Ref Out and V^+ . The circuit for a 204.8mV reference is shown in the test circuit. For a 2.048V reference, the fixed resistor should be removed, and a 25k Ω precision potentiometer between Ref Out and V^+ should be used.

DETAILED DESCRIPTION

Digital Section

The digital section includes the clock oscillator and scaling circuit, a 12-bit binary counter with output latches and TTL-compatible three-state output drivers, polarity, over-range and control logic, and UART handshake logic, as shown in the Block Diagram Figure 4.

Throughout this description, logic levels will be referred to as "low" or "high". The actual logic levels are defined in Table 1 "Operating Characteristics". For minimum power consumption, all inputs should swing from GND (low) to V^+ (high). Inputs driven from TTL gates should have 3-5k Ω pullup resistors added for maximum noise immunity.

MODE Input

The MODE input is used to control the output mode of the converter. When the MODE pin is connected to GND or left open (this input is provided with a pulldown resistor to ensure a low level when the pin is left open), the converter is in its "Direct" output mode, where the output data is directly accessible under the control of the chip and byte enable

inputs. When the MODE input is pulsed high, the converter enters the UART handshake mode and outputs the data in two bytes, then returns to "direct" mode. When the MODE input is left high, the converter will output data in the handshake mode at the end of every conversion cycle. (See section entitled "Handshake Mode" for further details).

STATUS Output

During a conversion cycle, the STATUS output goes high at the beginning of Signal Integrate (Phase II), and goes low one-half clock period after new data from the conversion has been stored in the output latches. See Figure 3 for details of this timing. This signal may be used as a "data valid" flag (data never changes while STATUS is low) to drive interrupts, or for monitoring the status of the converter.

RUN/HOLD Input

When the RUN/HOLD input is connected to V^+ or left open (this input has a pullup resistor to ensure a high level when the pin is left open), the circuit will continuously perform conversion cycles, updating the output latches at the end of every Deintegrate (Phase III) portion of the conversion cycle (See Figure 3). In this mode of operation, the conversion cycle will be performed in 8192 clock periods, regardless of the resulting value.

If the RUN/HOLD input goes low (and stays there) during Integrate (Phase II) or Deintegrate (Phase III) before the zero crossing is detected, the converter will complete the conversion in progress, update the output latches, and then terminate Phase III, jumping to Auto-Zero (Phase I). If RUN/HOLD stays low, the converter will ensure a minimum Auto-Zero time, and wait in Auto-Zero until the RUN/HOLD input goes high. The converter will begin the Integrate (Phase II) portion of the next conversion (and the STATUS output will go high) seven clock periods after the high level is detected at RUN/HOLD. See Figure 5 for details.

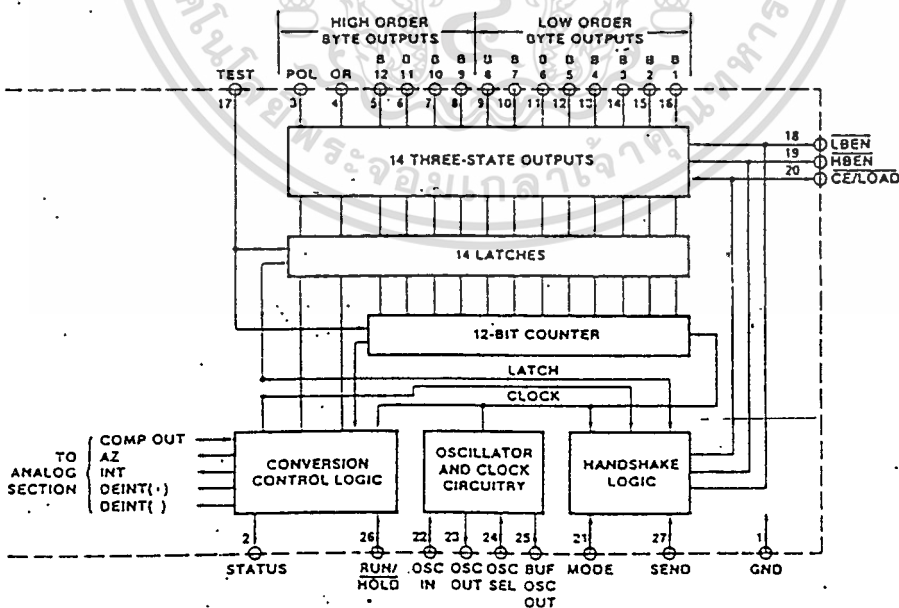


Figure 4: Digital Section

ICL7109

INTERSiL

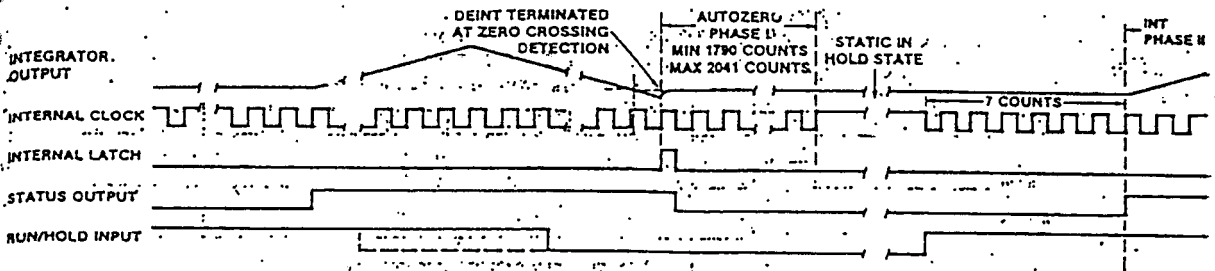


Figure 5: Run/Hold Operation

Using the RUN/HOLD input in this manner allows an easy "convert on demand" interface to be used. The converter may be held at idle in auto-zero with RUN/HOLD low. When RUN/HOLD goes high the conversion is started, and when the STATUS output goes low the new data is valid for transfer to the UART - see Handshake Model. RUN/HOLD may now go low terminating Deintegrate and ensuring a minimum Auto-Zero time before stopping to wait for the next conversion.

If RUN/HOLD goes low at any time during Deintegrate (Phase III) after the zero crossing has occurred, the circuit will immediately terminate Deintegrate and jump to Auto-Zero. This feature can be used to "short-cycle" the converter by eliminating the time spent in Deintegrate after the zero crossing. The required activity on the RUN/HOLD input can be provided by connecting it to the Buffered Oscillator Output. In this mode the conversion time is dependent on the input value measured. Also refer to Intersil Application Bulletin A030 for a discussion of the effects this will have on Auto-Zero performance.

If the RUN/HOLD input goes low and stays low during Auto-Zero (Phase II), the converter will simply stop at the end of Auto-Zero and wait for RUN/HOLD to go high. As above, Deintegrate (Phase III) begins seven clock periods after the high level is detected.

Direct Mode

When the MODE pin is left at a low level, the data outputs (bits 1 through 8 low order byte, bits 9 through 12, polarity and over-range high order byte) are accessible under control of the byte and chip enable terminals as inputs. These three inputs are all active low, and are provided with pullup resistors to ensure an inactive high level when left open. When the chip enable input is low, taking a byte enable input low will allow the outputs of that byte to become active (three-stated on). This allows a variety of parallel data accessing techniques to be used, as shown in the section entitled "Interfacing." The timing requirements for these outputs are shown in Figure 6 and Table 3.

Table 3 - Direct Mode Timing Requirements

SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
tBEA	Byte Enable Width	200	500		ns
tOAB	Data Access Time from Byte Enable		150	300	ns
tOHB	Data Hold Time from Byte Enable		150	300	ns
tCEA	Chip Enable Width	300	500		ns
tOAC	Data Access Time from Chip Enable		200	400	ns
tOHC	Data Hold Time from Chip Enable		200	400	ns

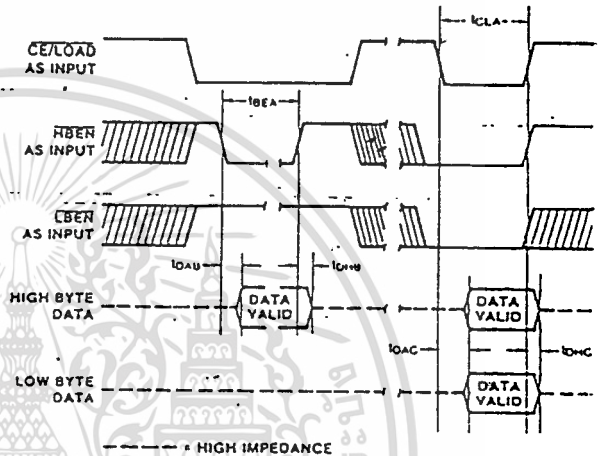


Figure 6: Direct Mode Output Timing

It should be noted that these control inputs are asynchronous with respect to the converter clock - the data may be accessed at any time. Thus it is possible to access the data while it is being updated, which could lead to scrambled data. Synchronizing the access of data with the conversion cycle by monitoring the STATUS output will prevent this. Data is never updated while STATUS is low.

Handshake Mode

The handshake output mode is provided as an alternative means of interfacing the ICL7109 to digital systems, where the A/D converter becomes active in controlling the flow of data instead of passively responding to chip and byte enable inputs. This mode is specifically designed to allow a direct interface between the ICL7109 and industry-standard UARTs (such as the Intersil CMOS UARTs, IM6402/3) with no external logic required. When triggered into the handshake mode, the ICL7109 provides all the control and flag signals necessary to sequence the two bytes of data into the UART and initiate their transmission in serial form. This greatly eases the task and reduces the cost of designing remote data acquisition stations using serial data transmission to minimize the number of lines to the central controlling processor.

Entry into the handshake mode is controlled by the MODE pin. When the MODE terminal is held high, the ICL7109 will enter the handshake mode after new data has been stored in the output latches at the end of every conversion performed (See Figures 7 and 8). The MODE terminal may also be used to trigger entry into the handshake mode on demand. At any time during the conversion cycle, the low to high transition of a short pulse at the MODE input will cause immediate entry

ICL7109

INTERSiC

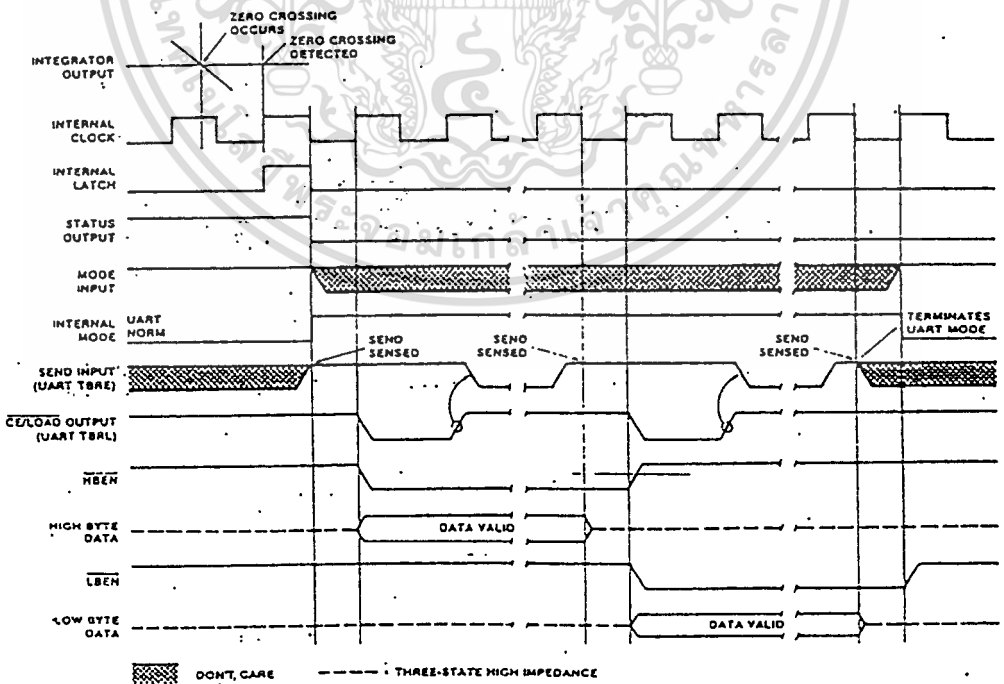
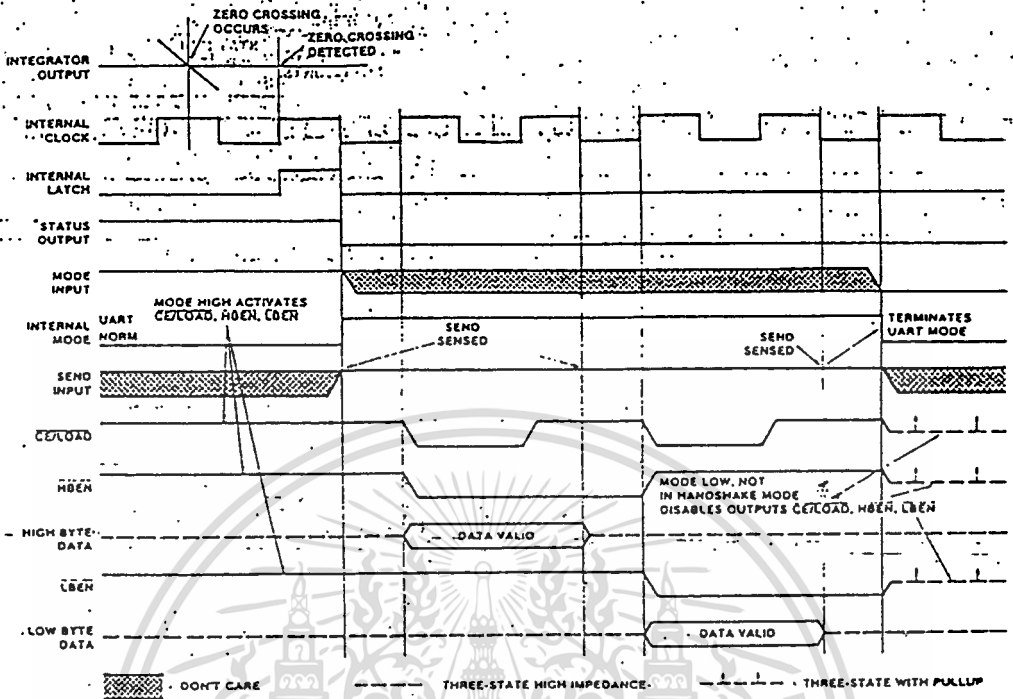


Figure 8: Handshake - Typical UART Interface Timing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ICL7109

into the handshake mode. If this pulse occurs while new data is being stored, the entry into handshake mode is delayed until the data is stable. While the converter is in the handshake mode, the MODE input is ignored, and although conversions will still be performed, data updating will be inhibited (See Figure 9) until the converter completes the output cycle and clears the handshake mode.

When the converter enters the handshake mode, or when the MODE input is high, the chip and byte enable terminals become TTL-compatible outputs which provide the control signals for the output cycle (See Figures 7, 8, and 9).

In handshake mode, the SEND input is used by the converter as an indication of the ability of the receiving device (such as a UART) to accept data.

Figure 7 shows the sequence of the output cycle with SEND held high. The handshake mode (Internal MODE high) is entered after the data latch pulse (since MODE remains high the $\overline{\text{CE/LOAD}}$, $\overline{\text{LBEN}}$ and $\overline{\text{HBEN}}$ terminals are active as outputs). The high level at the SEND input is sensed on the same high to low internal clock edge. On the next low to high internal clock edge, the $\overline{\text{CE/LOAD}}$ and the $\overline{\text{HBEN}}$ outputs assume a low level, and the high-order byte (bits 9 through 12, POL, and OR) outputs are enabled. The $\overline{\text{CE/LOAD}}$ output remains low for one full internal clock period only, the data outputs remain active for 1-1/2 internal clock periods, and the high byte enable remains low for two clock periods. Thus the $\overline{\text{CE/LOAD}}$ output low level or low to high edge may be used as a synchronizing signal to ensure valid data, and the

byte enable as an output may be used as a byte identification flag. With SEND remaining high, the converter completes the output cycle using $\overline{\text{CE/LOAD}}$ and $\overline{\text{LBEN}}$ while the low order byte outputs (bits 1 through 8) are activated. The handshake mode is terminated when both bytes are sent.

Figure 8 shows an output sequence where the SEND input is used to delay portions of the sequence, or handshake, to ensure correct data transfer. This timing diagram shows the relationships that occur using an industry-standard IM6402/3 CMOS UART to interface to serial data channels. In this interface, the SEND input to the ICL7109 is driven by the TBRE (Transmitter Buffer Register Empty) output of the UART, and the $\overline{\text{CE/LOAD}}$ terminal of the ICL7109 drives the TBRL (Transmitter Buffer Register Load) input to the UART. The data outputs are paralleled into the eight Transmitter Buffer Register inputs.

Assuming the UART Transmitter Buffer Register is empty, the SEND input will be high when the handshake mode is entered after new data is stored. The $\overline{\text{CE/LOAD}}$ and $\overline{\text{HBEN}}$ terminals will go low after SEND is sensed, and the high order byte outputs become active. When $\overline{\text{CE/LOAD}}$ goes high at the end of one clock period, the high order byte data is clocked into the UART Transmitter Buffer Register. The UART TBRE output will now go low, which halts the output cycle with the $\overline{\text{HBEN}}$ output low, and the high order byte outputs active. When the UART has transferred the data to the Transmitter Register and cleared the Transmitter Buffer Register, the TBRE returns high. On the next ICL7109

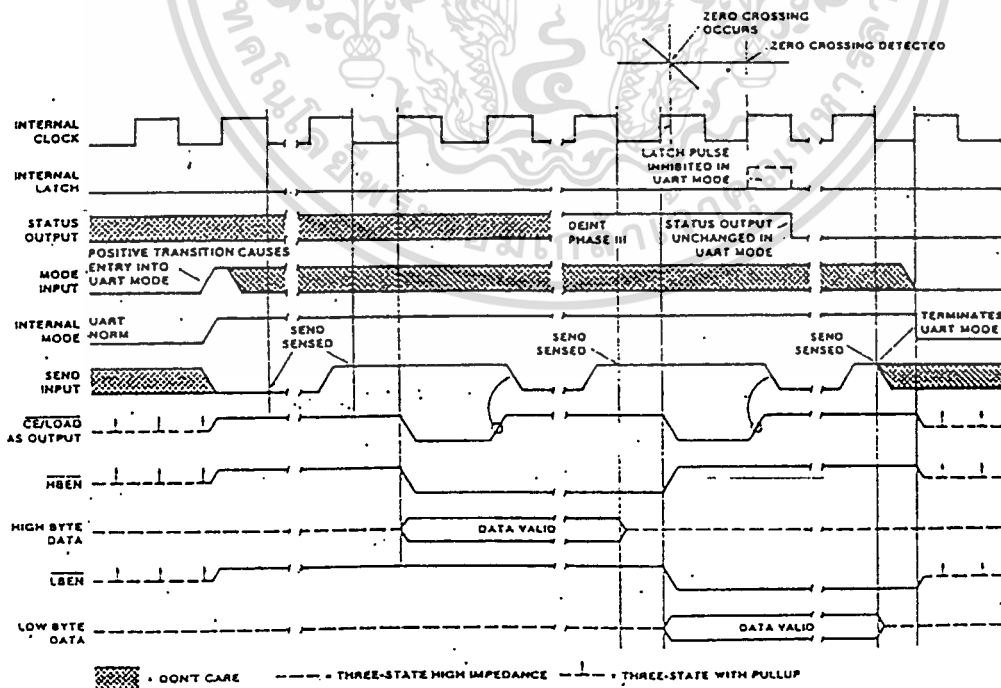


Figure 9: Handshake Triggered By Mode

ICL7109

internal clock high to low edge, the high order byte outputs are disabled, and one-half internal clock later, the HBEN output returns high. At the same time, the CE/LOAD and LBEN outputs go low, and the low order byte outputs become active. Similarly, when the CE/LOAD returns high at the end of one clock period, the low order data is clocked into the UART Transmitter Buffer Register, and TBRE again goes low. When TBRE returns to a high it will be sensed on the next ICL7109 internal clock high to low edge, disabling the data outputs. One-half internal clock later, the handshake mode will be cleared, and the CE/LOAD, HBEN, and LBEN terminals return high and stay active as long as MODE stays high.

With the MODE input remaining high as in these examples, the converter will output the results of every conversion except those completed during a handshake operation. By triggering the converter into handshake mode with a low to high edge on the MODE input, handshake output sequences may be performed on demand. Figure 9 shows a handshake output sequence triggered by such an edge. In addition, the SEND input is shown as being low when the converter enters handshake mode. In this case, the whole output sequence is controlled by the SEND input, and the sequence for the first (high order) byte is similar to the sequence for the second byte. This diagram also shows the output sequence taking longer than a conversion cycle. Note that the converter still makes conversions, with the STATUS output and RUN/HOLD input functioning normally. The only difference is that new data will not be latched when in handshake mode, and is therefore lost.

Oscillator

The ICL7109 is provided with a versatile three terminal oscillator to generate the internal clock. The oscillator may be overdriven, or may be operated as an RC or crystal oscillator. The OSCILLATOR SELECT input changes the internal configuration of the oscillator to optimize it for RC or crystal operation.

When the OSCILLATOR SELECT input is high or left open (the input is provided with a pullup resistor), the oscillator is configured for RC operation, and the internal clock will be of the same frequency and phase as the signal at the BUFFERED OSCILLATOR OUTPUT. The resistor and capacitor should be connected as in Figure 10. The circuit will oscillate at a frequency given by $f = .45/RC$. A 100k Ω resistor is recommended for useful ranges of frequency. For optimum 60Hz line rejection, the capacitor value should be chosen such that 2048 clock periods is close to an integral multiple of the 60Hz period.

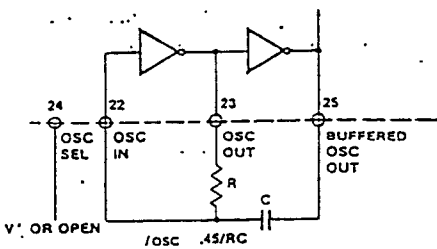


Figure 10: RC Oscillator

When the OSCILLATOR SELECT input is low a feedback device and output and input capacitors are added to the oscillator. In this configuration, as shown in Figure 11, the

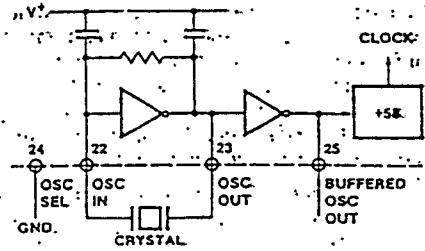


Figure 11: Crystal Oscillator

oscillator will operate with most crystals in the 1 to 5MHz range with no external components. Taking the OSCILLATOR SELECT input low also inserts a fixed +58 divider circuit between the BUFFERED OSCILLATOR OUTPUT and the internal clock. Using an inexpensive 3.58MHz TV crystal, this division ratio provides an integration time given by:

$$T = (2048 \text{ clock periods}) \times \left(\frac{58}{3.58\text{MHz}} \right) = 33.18\text{ms}$$

This time is very close to two 60Hz periods or 33.33ms. The error is less than one percent, which will give better than 40dB 60Hz rejection. The converter will operate reliably at conversion rates of up to 30 per second, which corresponds to a clock frequency of 245.8kHz.

If at any time the oscillator is to be overdriven, the overdriving signal should be applied at the OSCILLATOR INPUT, and the OSCILLATOR OUTPUT should be left open. The internal clock will be of the same frequency, duty cycle, and phase as the input signal when OSCILLATOR SELECT is left open. When OSCILLATOR SELECT is at GND, the clock will be a factor of 58 below the input frequency.

When using the ICL7109 with the IM6403 UART, it is possible to use one 3.58MHz crystal for both devices. The BUFFERED OSCILLATOR OUTPUT of the ICL7109 may be used to drive the OSCILLATOR INPUT of the UART, saving the need for a second crystal. However, the BUFFERED OSCILLATOR OUTPUT does not have a great deal of drive, and when driving more than one slave device, external buffering should be used.

Test Input

When the TEST input is taken to a level halfway between V^+ and GND, the counter output latches are enabled, allowing the counter contents to be examined anytime.

When the TEST input is connected to GND, the counter outputs are all forced into the high state, and the internal clock is disabled. When the input returns to the $1/2(V^+ - \text{GND})$ voltage or to V^+ and one clock is input, the counter outputs will all be clocked to the negative state. This allows easy testing of the counter and its outputs.

INTERFACING

Direct Mode

Figure 12 shows some of the combinations of chip enable and byte enable control signals which may be used when interfacing the ICL7109 to parallel data lines. The CE/LOAD input may be tied low, allowing either byte to be controlled by its own enable as in Figure 12A. Figure 12B shows a configuration where the two byte enables are connected together. In this configuration, the CE/LOAD serves as a chip enable, and the HBEN and LBEN may be connected to GND or serve as a second chip enable. The 14 data outputs will all be enabled simultaneously. Figure 12C shows the HBEN and LBEN as flag inputs, and CE/LOAD as a master enable, which could be the READ strobe available from most microprocessors.

กิตติกรรมประกาศ

- ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี ผู้จัดทำขอขอบคุณดร. กิตติพล ชิตสกุล อาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำและปรึกษาจนสำเร็จลุล่วงด้วยดี
- ขอขอบคุณ คุณสาร อาษา ที่ได้ให้คำแนะนำการเขียน โปรแกรมภาษาซี ตลอดจนคู่มืออ้างอิงต่าง ๆ
- ขอขอบคุณเพื่อน ๆ ที่ให้กำลังใจและคำปรึกษา จนทำให้ปริญญานิพนธ์สำเร็จด้วยดี



หนังสืออ้างอิง

- 1 กฤษณ์ อุ่นพิกุล, “ระบบเก็บข้อมูลปริมาณมลพิษในอากาศ”, ปรินูญานิพนธ์ ปี 2537.
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์ สถาบันเทคโนโลยีเจ้าคุณทหารลาดกระบัง
- 2 สุนทร วิทสุรพจน์, “การใช้งานไมโครคอนโทรเลอร์ตระกูล 8051”,
บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2537
- 3 “เอกสารคู่มือการใช้ INDOOR ODOR SENSOR” บริษัท แอนนาติจิกกรุ๊ป จำกัด
- 4 ยืน ภู่วรรณ, “เทคโนโลยีฮาร์ดแวร์ IBM PC”, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน),
หน้า 127-134.
- 5 ศิววัฒน์ ศิวะบรร, พรชัย จักรธำวงศ์, “การประยุกต์ใช้งานภาษาซี”
บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 608 หน้า
- 6 Motorola, “Linear/Interface ICs devices data Vol. II, 1993
- 7 Intel, “MCS-51 Microcontroller”