ระบบเก็บข้อมูลแบบหลายช่อง

MULTI – CHANNEL DATA LOGGER

โดย

นาย คำผอง คงสมบูน

นางสาว ลัดสะหมี ไชสุลิยง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวะกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถานบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ระบบเก็บข้อมูลแบบหลายช่อง

MULTI – CHANNEL DATA LOGGER

โดย

นาย คำผอง กงสมบูน 42015720

นางสาว ลัดสะหมี ไชสุลิยง 42015728

อาจารย์ที่ปรึกษา

ดร.กิติพล ชิตสกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวะกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถานบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์เรื่อง ระบบเก็บข้อมูลอัตโนมัติ

จัดทำโดย นาย คำผอง คงสมบูน รหัส 42015720

นางสาว ลัดสะหมี ไชสุลิยง รหัส 42015728

อาจารย์ที่ปรึกษา ดร. กิติพล ชิตสกุล

ปริญญานิพนธ์ฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ดร. กิติพล ชิตสกุล)

วันที่ 23 / 11 / 44

กิตติกรรมประกาศ

โครงงานนี้สำเร็จได้เนื่องจากได้รับคำแนะนำจากอาจารย์ ดร.กิติพล ชิตสกุล และอาจารย์ทุกท่านของภาควิชาอิเล็กทรอนิกส์ จึงขอกราบขอบพระคุณทุกท่านมา ณ โอกาสนี้ ขอขอบคุณเพื่อน ๆ ทุกคนที่คอยให้กำลังใจและให้คำแนะนำที่เป็นประโยชน์ในการทำโครงงานจนสำเร็จลุล่วง ขอขอบคุณมหาวิทยาลัยแห่งชาติลาว สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง องค์การ JICA ประจำประเทศลาว และ องค์การ JICA ประจำประเทศไทย ที่สนับสนุนทุน อุปกรณ์เครื่องมือวัดพื้นฐานทางอิเล็กทรอนิกส์ ที่ใช้ทั้งเพื่อ การทดลอง และการวัด อีกทั้งคอมพิวเตอร์เพื่อการออกแบบ และพิมพ์รายงานมาตลอด และหากปริญญานิพนธ์นี้มีข้อผิดพลาดประการใด ทางผู้จัดทำขอน้อมรับและขออภัยมา ณ โอกาสนี้ด้วย

.....

(นาย คำผอง คงสมบูน)

.....

(นางสาว ลัดสะหมี ไชสุลิยง)

ผู้จัดทำ

ระบบเก็บข้อมูลอัตโนมัติ

นาย คำผอง คงสมบูน
นางสาว ลัดสะหมี ไชสุลิยง
ดร.กิติพล ชิตสกุล(อาจารย์ที่ปรึกษา)
ปีการศึกษา 2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอการออกแบบระบบเก็บข้อมูลแบบหลายช่องสัญญาณ เพื่อใช้ในการเก็บข้อมูลต่าง ๆ ในรูปแบบสัญญาณดิจิตอล การทำงานของระบบทั้งหมดจะถูกควบคุมด้วยไมโครคอนโทรลเลอร์ ตระกูลMCS-51 ในการสุ่มเก็บข้อมูลระบบนี้ถูกออกแบบให้วัด และเก็บข้อมูลได้ถึง 4 ช่องสัญญาณ สัญญาณที่ถูกเก็บจะอยู่ในรูปของสัญญาณ แรงดันไฟฟ้าขนาดไม่เกิน 5 โวลต์ความถี่ตั้งแต่ 0ถึง xx เฮิร์ตซ สัญญาณอนาลอกอินพุทจะถูกแปลงเป็นสัญญาณดิจิตอลความละเอียด 8 บิตที่อัตราการสุ่มกำหนดได้ ข้อมูลที่ได้จะถูกเก็บไว้ในหน่วยความจำที่มีขนาด 32 กิโลไบต์และส่งออกไปยังไมโครคอมพิวเตอร์ด้วยอัตราเร็ว 9600 บิต ต่อวินาทีโดยผ่านพอร์ตอ นุกรม RS 232 เพื่อแสดงผลออกมาทางหน้าจอคอมพิวเตอร์ ในโครงงานนี้ใช้อุณหภูมิเป็นข้อมูลในการวัดและบันทึกผล โดยใช้ LM335 เป็นเซ็นเซอร์

MULTICHANNEL DATA LOGGER SYSTEM

Mr.Khamphong KHONGSOMBOUN

Ms.Latsamy SAYSOURINHONG

Dr.Kitiphol CHITSAKUL (Advisor)

Academic year 2001

Abstract

This thesis prcsents a design of Multi-Channel Data Logger System. This system is designed to acquire and digitally record any analogue signals. A Micro-controller family MCS-51 plays the grand role in controlling the whole system. Analogue signal, amplitude up to 5 volts, can be as channel input up to 4 channels. Depend on the channel sampling rate defined by user, frequency of a signal input could be 0 to xx cycle per second. The converted 8 bits digital data are stored in RAM of 32 KBytes before being sent to a microcomputer via a serial port at board rate 9600 bps. The record data could be displayed on monitor in graphic and text mode. As an example, a temperature scnsing board, employing a LM335 as the sensor, has been implemented to serve as input of the system.

สารบัญ

สารบัญรูปภาพ

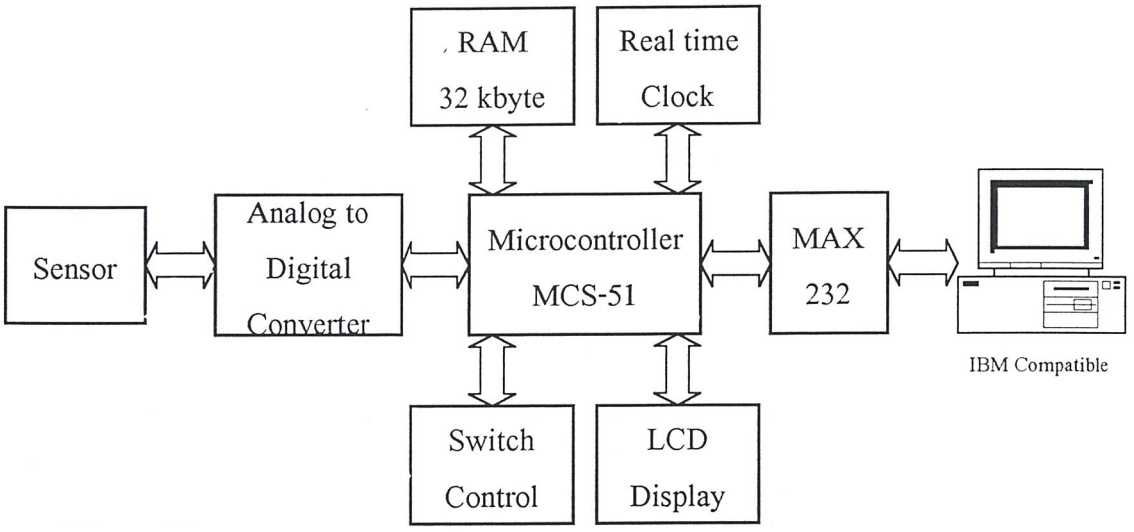
สารบัญตาราง

บทที่ 1
บทนำ

ดาต้าล็อกเกอร์ (data logger) คืออุปกรณ์ที่ใช้ในการเก็บบันทึกข้อมูลในช่วงเวลาที่กำหนดเพื่อนำมาแสดงผลหรือนำข้อมูลเหล่านั้นมาประมวลผลภายหลัง การเก็บข้อมูลในลักษณะดังกล่าวอาจจะเป็นการบันทึกแบบอนาลอก เช่นการบันทึกเสียงและภาพแบบเก่า การเก็บบันทึกข้อมูลอย่างต่อเนื่องบนกระดาษกราฟ อย่างไรก็ดีการบันทึกแบบดิจิตอลจะมีข้อดีในแง่การของจัดเก็บที่สามารถเก็บปริมาณได้มากในขนาดทางกายภาพที่เล็กกว่า นอกจากนี้สามารถนำมาประมวลผลด้วยดิจิตอลคอมพิวเตอร์ได้โดยตรง ข้อดีประการแรกดังกล่าวของดิจิตอลดาต้าล็อกเกอร์นั้น เหมาะกับการเก็บข้อมูลในบริเวณที่ห่างไกล ยากต่อการตรวจสอบ หรือ บันทึกการ เปลี่ยนแปลงนั้นเป็นเวลานาน ๆ เนื่องจากขนาดเล็กและเก็บข้อมูลได้มากกว่าและการกินกำลังงานน้อยกว่าจากการใช้เทคโนโลยีของอุปกรณ์สารกึ่งตัวนำในปัจจุบัน แต่เนื่องจากปริมาณทางกายภาพที่ใช้เซ็นเซอร์ (sensor) แปลงให้อยู่ในรูปของปริมาณไฟฟ้าแล้วเกือบทั้งหมดจะอยู่ในรูปแบบอนาลอกที่ต่อเนื่องในแกนเวลา การเก็บแบบดิจิตอลนั้นต้องมีการสุ่มสัญญาณนั้นในเวลาใด ๆ แล้วนำมาเข้ารหัสดิจิตอลเพื่อให้เหมาะกับการจัดเก็บและประมวลผล ดังนั้นการจัดเก็บแบบดิจิตอลอาจจะขาดความต่อเนื่องหากการสุ่มค่ามีช่วงห่างเมื่อเทียบกับการเปลี่ยนแปลงของปริมาณทางกายภาพนั้น ๆ ดังนั้นดิจิตอลดาต้าล็อกเกอร์มักจะสามารถกำหนดช่วงการสุ่มได้ การนำไมโครคอนโทรลเลอร์จะช่วยลดความซับซ้อนของวงจรอิเล็กทรอนิกส์ที่ใช้ในควบคุมกระบวนการแปลงสัญญาณ การสุ่มและการจัดเก็บข้อมูลแบบอัตโนมัติโปรแกรมได้ โครงงานนี้ได้ทำการศึกษาออกแบบระบบดิจิตอลดาต้าล็อกเกอร์แบบ 8 บิต สี่ช่องสัญญาณ ควบคุมด้วยไมโครคอนโทรลเลอร์ โดยใช้อุณหภูมิเป็นตัวอย่างของข้อมูลในการเก็บบันทึก

1.1 แนวความคิดและโครงสร้างของโครงงาน

แนวความคิดของโครงงานที่ต้องการสร้างคือ ดาต้าล็อกเกอร์ ที่สามารถวัดได้หลาย ช่องสัญญาณ โดยมีไมโครคอนโทรลเลอร์MCS51เป็นอุปกรณ์ที่ใช้ในการควบคุมระบบโดยเราสามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระและมีการส่งข้อมูลที่วัดได้นั้นมาบันทึกบนไมโครคอมพิวเตอร์ซึ่งอยู่ระยะไกลได้จากแนวความคิดของโครงงานดังกล่าวทำให้เราได้บล็อกไดอะแกรมของโครงงานดังรูปที่ 1-1



รูปที่ 1-1 โครงสร้างของโครงงาน

1.2 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจถึงการออกแบบ การทำงาน และโครงสร้างภายในของไมโครคอนโทรลเลอร์มากยิ่งขึ้น
2. มี ความเข้าใจและสามารถออกแบวงจร ทั้งทางฮาร์ดแวร์และซอฟท์แวร์ ที่ใช้ไมโครคอนโทรล-เลอร์เป็นตัวควบคุม
3. ได้รู้จักขั้นตอนการทำงานการวางแผนการตรวจสอบการทำงานและแนวทางการแก้ไขปัญหาที่เกิดขึ้น
4. สามารถนำแนวคิดไปประยุกต์ใช้ให้เกิดประโยชน์ได้กว้างขวางขึ้นต่อไป

1.3 ขอบเขตของโครงงาน

จากแนวความคิดในการออกแบบระบบ ซึ่งกำหนดว่าสามารถใช้ในการเก็บข้อมูลจากเซ็นเซอร์ ได้ทั้งหมด 4 ช่อง ระดับแรงดันอินพุทตั้งแต่ 0-5 โวลต์ แต่ในข้อจำกัดของเวลาในการดำเนินโครงงาน จะทำการวัดเฉพาะอุณหภูมิเพียงอย่างเดียว โดยใช้เซมิคอนดัคเตอร์เซ็นเซอร์ วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลใช้ขนาด 8บิต ข้อมูลดิจิตอลนี้จะถูกเก็บบันทึกไว้ในหน่วยความจำแบบแรม(RAM) ขนาด 32 กิโลไบต์ตลอดในช่วงการเก็บบันทึก ก่อนที่จะถูกเรียกอ่านหรือส่งต่อไปยังไมโครอมพิวเตอร์ ผ่านทางพอร์ตอนุกรม (RS-232) การกำหนดรายละเอียดในการสุ่มเก็บข้อมูลนั้น สามารถกำหนดได้ว่าจะให้เก็บอย่างไรหรือเก็บทุก ๆ กี่นาทีและสามารถตั้งเวลาในการเก็บข้อมูลหรือหยุดเก็บข้อมูลได้โดยการโปรแกรมบนไมโครคอมพิวเตอร์ หรือโปรแกรมไมโครคอนโทรเลอร์โดยตรง ข้อมูลที่ได้จากดาต้า ล็อกเกอร์จะถูกเก็บในลักษณะไฟล์ตัวอักษร

1.4 เนื้อหาของรายงาน

ปริญญานิพนธ์ฉบับนี้ปริญญานิพนธ์เล่มนี้เป็นผลจากการศึกษาและทดลองตลอดสองภาคการศึกษา เพื่อออกแบบสร้างดิจิตอลดาต้าล็อกเกอร์ขนาด 8 บิต 4 ช่องสัญญาณ โดยเริ่มจากการรวบรวมรายละเอียดเกี่ยวกับทฤษฎีที่เกี่ยวข้องกับการวัดอุณหภูมิการแปลงสัญญาณเพื่อที่จะเป็นแนวทางในการศึกษาเพิ่มเติมหรือออกแบบเพื่อให้ได้ระบบที่มีประสิทธิภาพที่ดีตามวัตถุประสงค์ ซึ่งเนื้อหาของรายงานเล่มนี้นั้นจะประกอบด้วยส่วนต่าง ๆ แยกเป็นบท ๆ ไปดังนี้

บทที่ 1 บทนำกล่าวถึงแนวคิดของระบบที่จะสร้างขึ้นมา

บทที่ 2

กล่าวถึงอุปกรณ์ที่ใช้ในการตรวจวัดอุณหภูมิซึ่งก็คือตัวเซ็นเซอร์ในบทนี้ยังจะได้กล่าวถึงวงจรที่ใช้ในการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลเพื่อทำการสุ่มแรงดันเอาท์พุตที่ได้จากตัวอุปกรณ์เซ็นเซอร์ในที่นี้จะใช้การเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอลแบบประมาณรวบยอดซึ่งเป็นที่นิยมกันมาก เนื่องจากมีความเร็วปานกลางและค่อนข้างสู ง อีกทั้ง ยังมีความละเอียด พอสมควร

บทที่ 3

กล่าวถึงทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ซึ่งเป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่บรรจุความสามารถมากมาย โดยในบทนี้จะกล่าวถึงคุณสมบัติ โครงสร้างสถาปัตยกรรม การจัดหน่วยความจำและการใช้งานพอร์ต อินพุต / เอาท์พุตนอกจากนั้นยังจะกล่าวถึงการเชื่อมต่อไมโครคอนโทรลเลอร์กับคอมพิวเตอร์โดยผ่านการอินเตอร์เฟสตามมาตรฐาน RS-232

บทที่ 4

กล่าวถึงหลักการออกแบบและการทำงานของระบบเก็บข้อมูล(DataLogger)ในบทนี้จะได้กล่าวถึงรายละเอียดในการออกแบบส่วนต่างๆ ไม่ว่าจะเป็นวงจรภาคเซ็นเซอร์ และการออกแบบภาคควบคุมซึ่งจะพูดถึงการเชื่อมต่อไมโครคอนโทรลเลอร์กับไอซี ADC0808 การเชื่อมกับหน่วยความจำภายนอก(ไอซี62256)การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์การออกแบบภาคแสดงผลและการออกแบบหน้าจอ

บทที่ 5

กล่าวถึงการทดสอบคุณสมบัติและผลการทดลองที่ได้จากส่วนต่างๆไม่ว่า จะเป็น เซ็นเซอร์ และส่วนที่ทำการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

บทที่ 6 สรุปและวิจารณ์

บทที่ 2
อุปกรณ์เซนเซอร์ และการแปลงสัญญาณอนาลอกเป็นดิจิตอล

เซ็นเซอร์เป็นองค์ประกอบที่สำคัญในการวัดสภาวะหรือปริมาณทางกายภาพของตัวกลางหรือวัตถุใด ๆ ในหัวข้อต่อไปนี้จะกล่าวถึงอุปกรณ์เซ็นเซอร์อุณหภูมิพอสังเขปเพื่อให้ความเข้าใจถึงแนวคิดของระบบที่ออกแบบขึ้นมา ในหัวข้อถัดไปจะกล่าวถึงพื้นฐานการแปลงสัญญาณเบื้องต้นที่เน้นเฉพาะแบบที่นำเอามาใช้งาน

2.1 อุปกรณ์เซ็นเซอร์อุณหภูมิ

อุปกรณ์ที่ใช้ในการตรวจวัดอุณหภูมิ มีอยู่ หลาย ชนิด ด้วยกันโดยอุปกรณ์ แต่ละชนิดจะมี หลักการเปลี่ยนแปลงคุณสมบัติเฉพาะของสาร คือจะต้องมีการเปลี่ยนแปลงค่าเมื่ออุณหภูมิที่วัดเปลี่ยนแปลงไป และ การเปลี่ยนแปลงที่วัดได้จะต้องคงที่แน่นอน ซึ่งหลักการที่ใช้ในการวัดอุณหภูมิ โดยทั่วไป สามารถแบ่งออกได้ดังต่อไปนี้

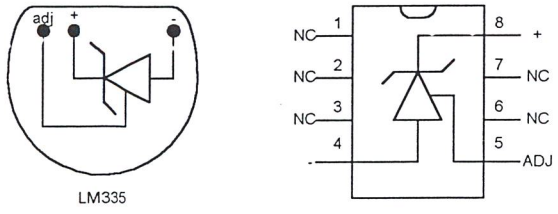
- อาศัยการเปลี่ยนแปลงคุณสมบัติเชิงกล
- อาศัยการเปลี่ยนแปลงความดันก๊าซหรือไอ
- อาศัยการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้า
- อาศัยการเปลี่ยนแปลงคุณสมบัติทางแสงและการแผ่รังสี
- อาศัยหลักการโดยวิธีทางเคมี

ในโครงงานนี้การตรวจวัดอุณหภูมิจะใช้หลักการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้า เนื่องจากสัญญาณที่ได้จากอุปกรณ์นี้สามารถนำไปต่อใช้งานร่วมกับวงจรอิเล็กทรอนิกส์เพื่อวัดผลได้สดวก อุปกรณ์กลุ่มนี้โดยทั่วไปจะอยู่ในรูปของตัวต้านทาน ไดโอด หรือในรูปของไอซี ซึ่งอุปกรณ์ดังกล่าวจะใช้วัดอุณหภูมิในช่วง –50 องศาเซลเซียส ถึง 150 องศาเซลเซียส และโครงสร้างภายนอกจะมีขนาดที่เล็ก

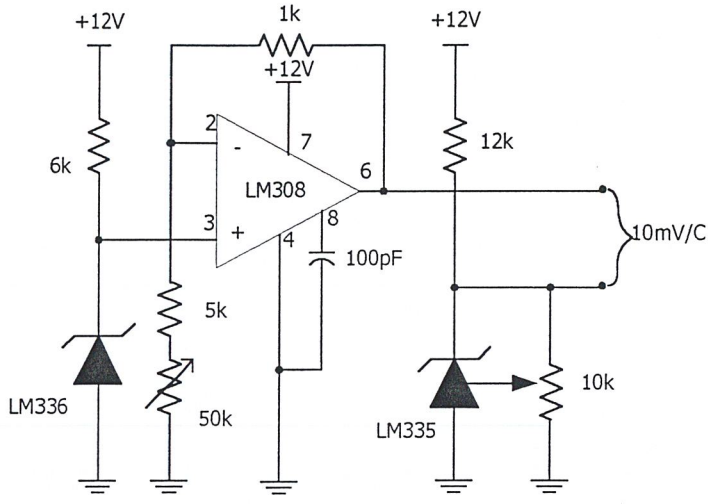
เซ็นเซอร์ที่มีโครงสร้างภายในเป็นตัวต้านทาน ค่าความต้านทานจะเปลี่ยนค่าตามอุณหภูมิ หรือกรณีเป็นซีเนอร์ไดโอด สามารถเปลี่ยนแปลงแรงดันพังทะลายตามระดับอุณหภูมิ ดังนั้นเมื่อไบอัสย้อนกลับแล้วอุณหภูมิโดยรอบมีการเปลี่ยนแปลง แรงดันที่ตกคร่อมอุปกรณ์เซ็นเซอร์ดังกล่าวจะเปลี่ยนแปลงตามไปด้วย จึงสามารถนำการเปลี่ยนแปลงดังกล่าวมาประมวลผล เพื่อวัดค่าอุณหภูมิออกมาได้

ในโครงงานนี้จะใช้เซ็นเซอร์แบบสารกึ่งตัวนำเบอร์ LM 335 (Precision Temperature Sensor) ซึ่งเป็นวงจรรวมที่มีคุณสมบัติทางอุณหภูมิคล้ายซีเนอร์ไดโอด สามารถใช้ทำงานในย่าน -40 องศาเซลเซียส ถึง100องศาเซลเซียส มีความเที่ยงตรงสูงในย่านที่ใช้งาน เมื่อต่อเป็นวงจรใช้งานร่วมกับอุปกรณ์ภายนอกดังแสดงในรูปที่2-2 จะสามารถให้แรงดันเอาท์พุทเป็นปฏิภาคโดยตรงกับอุณหภูมิโดยรอบ ในวงจรจะใช้ LM336 จ่ายแรงดันอ้างอิงเปรียบเทียบกับเอาท์พุทของ LM335 จะได้แรงดันเอาต์พุตที่เป็นผลต่าง

นี้ในอัตรา +10 mV / °C ซึ่งจะขยายแรงดันผลต่างด้วยวงจรขยายความแตกต่าง (Differential amplifier) เพื่อให้มีค่าแรงดันเหมาะสมกับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล



รูปที่ 2-1 แสดงตำแหน่งขาสัญญาณของ LM335



รูปที่ 2-2 แสดงการต่อวงจร Centigrade Thermometer

ตารางที่ 2-1 แสดงความสัมพันธ์ระหว่างค่าอุณหภูมิหน่วย °C ,°K และ Vout

อุณหภูมิ		
องศาเซลเซียส	องศาเคลวิน	Vout
0	273	2.732
25	298	2.982
50	323	3.232
75	348	3.482
100	373	3.732

โดยค่าแรงดันที่ได้ทางเอาท์พุตของตัวอุปกรณ์เซ็นเซอร์ ได้มาจากการคำนวณด้วยสมการ

$$V_{OUT_T} = V_{OUT_{T_O}} \times \frac{T}{T_O} \qquad (2.1)$$

เมื่อ T คืออุณหภูมิที่เราสามารถรู้ได้ ($^{\circ}K$)

T_O คืออุณหภูมิอ้างอิง ($298^{\circ}K$)

$V_{OUT_{T_O}}$ คือค่าแรงดันเอาท์พุตที่อุณหภูมิอ้างอิง (2.982V ที่ $25^{\circ}C$)

V_{OUT_T} คือแรงดันเอาท์พุตที่ LM335

2.2 การแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog to Digital Converter : ADC)

การนำสัญญาณอนาล็อกมาประมวลผลในวงจร ดิจิตอล จำเป็นต้อง อาศัยตัวแปลง สัญญาณอนาล็อกเป็นสัญญาณดิจิตอลหรือเรียกว่า ADC ซึ่งความละเอียดของการแปลงสัญญาณขึ้นอยู่กับ จำนวนบิตของ ADC ในการแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้นมีขั้นตอนต่างๆที่ซับซ้อนอยู่พอสมควร กว่าจะได้สัญญาณเป็นดิจิตอล ที่มีค่าผิดพลาดต่ำ ซึ่งโดยส่วนใหญ่แล้ว จะประกอบด้วย ขั้นตอนที่สำคัญอยู่สามขั้นตอนคือ การสุ่ม (Sampling) การจัดระดับ (Quantizing) และการเข้ารหัส (Encoder) โดยทั้งหมดนั้นถือเป็นทฤษฎีพื้นฐานในการแปลงสัญญาณ จากอนาล็อกเป็นดิจิตอล

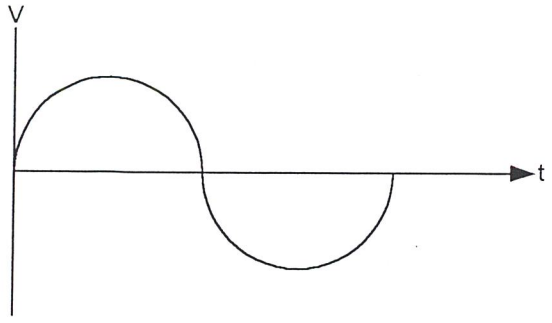
2.2.1 ทฤษฎีการสุ่ม (Sampling Theory)

ในการแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้น ADC จำเป็นต้องใช้เวลาช่วงหนึ่งในการจัดการ ซึ่งช่วงเวลาดังกล่าวนั้นขึ้นอยู่กับหลายปัจจัยต่าง ๆ เช่นความละเอียดของการแปลงสัญญาณ เทคนิคของการแปลงสัญญาณและความเร็วในการทำงานของอุปกรณ์ร่วมอื่น ๆ การกำหนดความเร็วของการแปลงสัญญาณขึ้นอยู่กับการประยุกต์ใช้งานเฉพาะอย่างและความแม่นยำที่ต้องการ ช่วงเวลาในการแปลงสัญญาณบางครั้งอาจเรียกว่าApertureTimeซึ่งหมายถึงช่วงเวลาที่เกิดความไม่แน่นอนขึ้นในการวัดและผลก็คือความผิดพลาด (error) ต่อค่าที่วัดได้

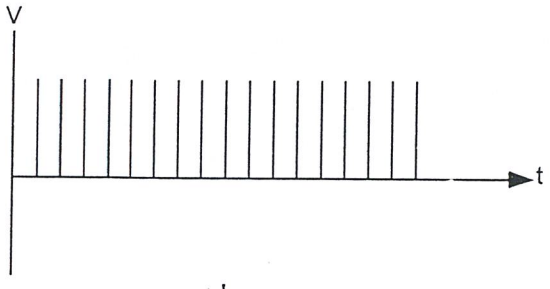
วงจร Sampling & Hold จะทำการสุ่มสัญญาณอินพุต และนำสัญญาณที่สุ่มนั้นมาเก็บ (hold) ไว้ในช่วงเวลาใดหนึ่ง ซึ่งส่วนใหญ่จะใช้การประจุแรงดันนั้นไว้ในตัวเก็บประจุที่รั่วไหลต่ำ Aperture time ของ Sampling & Hold คือเวลาตั้งแต่เริ่มสุ่มสัญญาณจนเก็บค่าประจุแรงดันจนถึงค่าที่สุ่ม ซึ่งค่า Aperture time สำหรับ ค่าความถี่ ในการสุ่ม นั้น จะขึ้นอยู่กับแบนด์วิดธ์ของสัญญาณ

ในการสุ่มสัญญาณอนาลอกจะถูกสุ่มเป็นระยะๆคงที่ตาม รูป 2-3 ค การสุ่มจะเป็นการตัดต่อ สัญญาณอนาลอกในช่วงเวลาอันสั้นด้วยสวิทช์ที่ทำงานด้วยความเร็วสูง ผลของการสุ่มสัญญาณด้วย ความเร็วจะเสมือนการคูณขบวนสัญญาณพัลส์แคบๆกับสัญญาณอนาลอก ซึ่งจะได้เป็นสัญญาณที่มอดดูเลทระหว่างขบวนพัลส์กับสัญญาณอนาลอกโดยเสมือนว่าสัญญาณอนาลอกจะซ้อนทับมาบนขบวนพัลส์ ถ้าหากสัญญาณอนาลอกที่ถูกสุ่ม ถูก hold จนกว่าสัญญาณค่าใหม่จะถูกสุ่มเข้ามาซึ่งจะได้ลักษณะ ของ

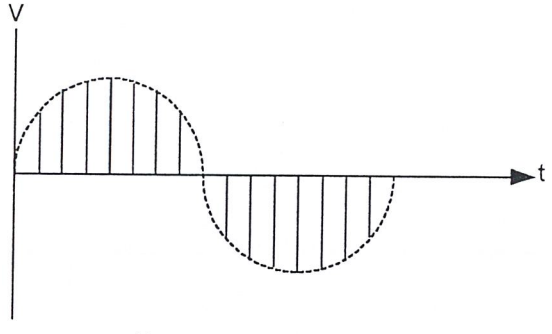
เอาท์พุตที่แสดงในรูป 2-3.ง



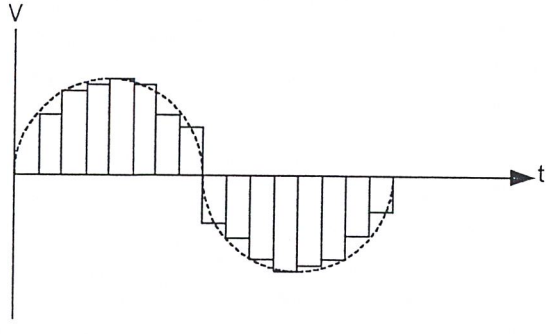
ก. สัญญาณอนาลอกอินพุต



ข..พัลส์ที่นำมาสุ่มสัญญาณ



ค..สัญญาณอนาลอกหลังการสุ่ม



ง..สัญญาณอนาลอกหลังการสุ่มและ Hold ไว้

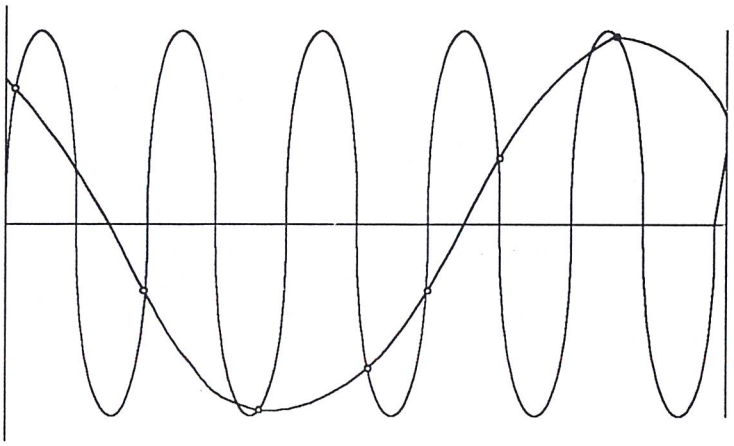
รูปที่ 2-3 การสุ่มสัญญาณ

ถ้าสัญญาณต่อเนื่อง ซึ่งมีความถี่และฮาร์โมนิคส์ไม่เกิน f_c ถูกสุ่มด้วยอัตราการสุ่มไม่น้อยกว่า $2f_c$ แล้วสัญญาณดังกล่าวจะสามารถเปลี่ยนกลับมาได้อย่างเดิมโดยไม่สูญเสียรายละเอียดหรือผิดเพี้ยนไป

ผลของการใช้อัตราการสุ่มที่ไม่เหมาะสมจะทำให้เกิดสัญญาณรบกวนความถี่ต่ำกว่าเรียกว่า Aliasing frequency เมื่อสัญญาณถูกเปลี่ยนกลับมามาเช่นเดิมหลังจากถูกสุ่มแล้วดังรูปที่ 2.4 จะเห็นว่าความถี่ Aliasing จะแตกต่างจากความถี่เดิมไปมาก Anti aliasing filter เป็นวงจรกรองผ่านความถี่ต่ำที่จะช่วยลดสัญญาณในแถบความถี่ที่ทำให้เกิด Aliasing frequency ในขณะที่ ไม่ต้องการให้เกิดความผิดพลาดของสัญญาณในแบนด์ที่ใช้งานและไม่ลดความแม่นยำในการวัดโดยรวมอีกด้วยนอกจากการใช้ฟิลเตอร์ฮาร์โมนิคส์ที่สูงๆแล้วพยายามให้การสุ่มเป็นไปอย่างรวดเร็วมากที่สุดซึ่งปกติจะสูงกว่า ความถี่ต่ำสุดตามทฤษฎี Sampling คือ $2 f_c$

การใช้ Anti aliasing filter ขึ้นอยู่กับ

- ความถี่สูงสุดที่สนใจ
- อัตราการสุ่ม
- ความละเอียดของการแปลงสัญญาณ



รูปที่ 2-4 การเกิดความถี่ Aliasing

จากทฤษฎีการสุ่มที่ว่าให้ $f_s > 2f_c$ นั้นก็เพื่อขจัดการซ้อนกันของสเปคตรัม (Aliasing Effect) เพื่อจำกัด แบนวิดธ์ของ สัญญาณที่จะถูกแปลงไม่ให้เกินไปกว่า $(f_s/2)$ ซึ่งในทางปฏิบัติแล้วจะยังคงเกิด frequency folding ได้เสมอจากส่วนฮาร์โมนิคส์ของสัญญาณและสเปคตรัมของสัญญาณรบกวนที่ยังคงมีอยู่แม้ว่าจะทำการฟิลเตอร์แล้วก็ตาม

2.2.2 การจัดระดับสัญญาณ (Quantizing)

อีกขั้นตอนหนึ่งที่สำคัญหลังจากที่สัญญาณผ่านการสุ่มมาแล้วก็คือขั้นตอนการจัดระดับของสัญญาณโดยสัญญาณที่ผ่านการสุ่มมาแล้วจะมีลักษณะที่ไม่ต่อเนื่อง (Discrete Signal) และเนื่องจากสัญญาณอนาล็อกส่วนใหญ่จะมีสัญญาณรบกวนปะปนมาด้วยเสมอจึงจำเป็นต้องกำจัดสัญญาณเหล่านี้ออกไป เพื่อ ไม่ให้เกิดความผิดพลาดในการประมวลผลแบบดิจิตอล และ สามารถนำไปเข้ารหัสได้ง่าย

เมื่อผ่านการสุ่มการจัดระดับแล้ว ข้อมูลที่เป็นสัญญาณอนาล็อกจะถูกเปลี่ยนเป็นระดับ สัญญาณที่แน่นอน และ มีลักษณะไม่ต่อเนื่องเป็นระดับต่างๆกัน ในแต่ละสถานะของสัญญาณดิจิตอลเอาท์พุตจะแทนขนาดของสัญญาณอนาลอกค่าใดค่าหนึ่งในช่วงแคบๆระหว่างจุดแบ่งระดับ เรียกช่วงเล็กๆนี้ว่า Analog Quantization หรือหนึ่งควันตัม (Quantum) หรือ 1 LSB (Least Significant Bit) ของการแปลงสัญญาณ คลาดเคลื่อน(Error) ในการแทนค่าสัญญาณอนาลอกในช่วงนั้น ๆ อยู่ด้วย ค่าความคลาดเคลื่อนนี้เป็นธรรมชาติของ Quantizing ซึ่งทำการแก้ไขไม่ได้นอกจากการเพิ่มจำนวนบิตของ Quantizer ให้มากขึ้นและอีกอย่างหนึ่งที่สำคัญมากก็คือความละเอียด(Resolution)ของตัวแปลงว่ามีกี่บิตเพราะว่าจำนวนบิตของตัว แปลงจะบอกถึงจำนวนระดับสัญญาณคือ

$$\text{จำนวนระดับสัญญาณ}(Q) = 2^n \text{ ; n คือจำนวนบิต}$$

ถ้าเราใช้ตัวแปลงขนาด 8 บิต กับขนาดสัญญาณอนาล็อกที่สูงสุด 10 โวลต์จะได้จำนวนระ ดับสัญญาณเท่ากับ 256 ระดับ ดังนั้นค่าของบิตขวาสุด (Least Significant Bit : LSB) จะมีค่า เป็น

$$\text{ค่าหนึ่งบิตด้านต่ำ} = \text{แรงดันสัญญาณด้านสูง/จำนวนระดับสัญญาณ}$$

ในบางครั้งเราเรียก LSB ว่า " Step Size " โดยใช้สัญญาณจากตัวแปลง 8 บิต และ ขนาดสัญญาณ 10 โวลต์ จะมีค่า $10/2^8 = 10/256 = 0.039$ V .แล้วอาจเขียนสมการในการหาค่า (Δ) ใหม่ได้ว่า

$$\Delta = FSR / Q$$

เมื่อ Q คือจำนวนระดับสัญญาณ

FSR คือช่วงเต็มสเกลของแรงดันอนาลอก (Full Scale Range)

Δ คือ ค่าความผิดพลาดจากการจัดระดับสัญญาณ

จะเห็นว่าจำนวนบิตยิ่งมากค่า Δ จะยิ่งลดลงจะทำให้ค่าความผิดพลาดลดลงไปด้วย โดยค่าความผิดพลาดจะอยู่ระหว่าง 0 ถึง $\Delta/2$ ซึ่งค่าความผิดพลาดอาจจะเป็นศูนย์ถ้าสัญญาณอนาลอกมีค่าที่กึ่งกลางของควันตัมพอดี

2.2.3 การเข้ารหัส (Coding)

ในการเข้ารหัสสัญญาณที่ผ่านการสุ่ม และการจัดระดับมาแล้วนั้นส่วนใหญ่จะแปลงให้อยู่ในรูปของรหัสตัวเลขฐานสอง (Binary code) แล้ว เปลี่ยนจากข้อมูลแบบขนานให้เป็นแบบอนุกรม เพื่อสามารถลดช่องสัญญาณในการส่งข้อมูลให้เหรือเพียงช่องสัญญาณเดียวได้

จากตารางที่ 2.2 จะแสดงการเข้ารหัสของระดับแรงดันจากการสุ่มสัญญาณขนาด 0 ถึง 5 โวลท์ และผ่านการจัดระดับ สัญญาณมาแล้ว ซึ่งที่ระดับแรงดันต่ำที่สุดก็จะมีขนาด 8 บิตเลขฐานสองเป็น 0000 0000 ส่วนระดับแรงดันสูงที่ สุดก็จะมีรหัสเลขฐานสองเป็น 1111 1111 เป็นต้น

ตารางที่ 2-2 แสดงการเข้ารหัสสัญญาณ

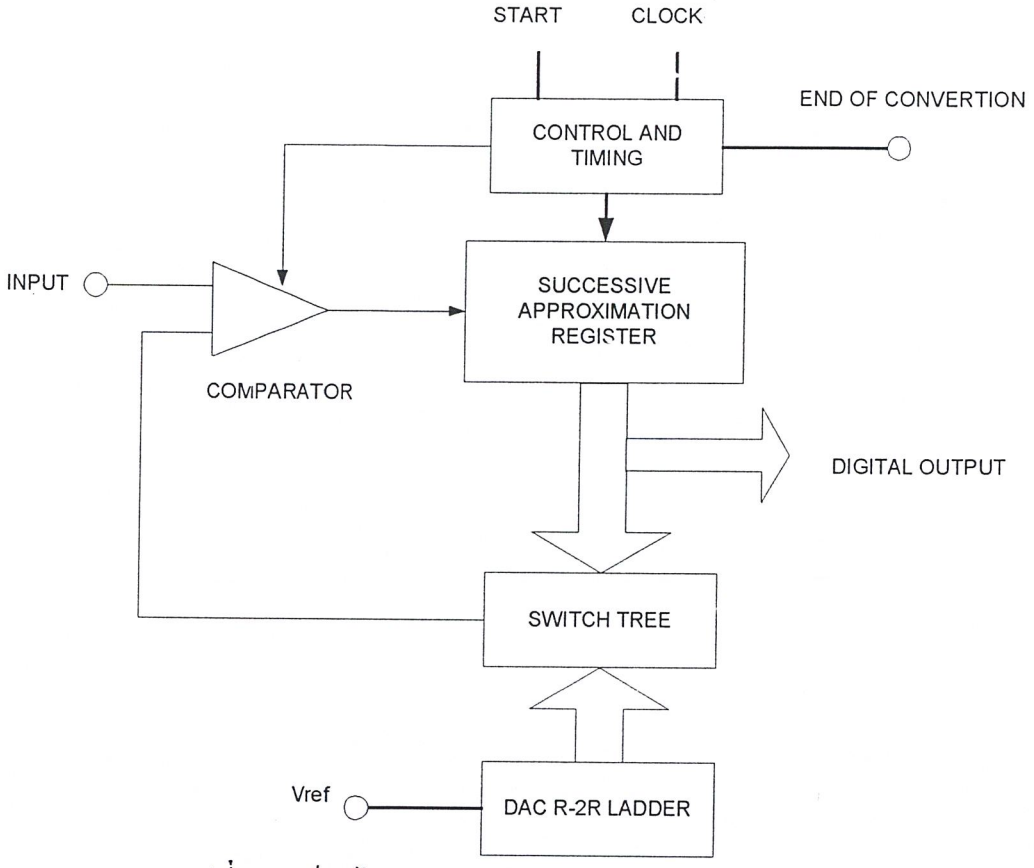
Vin	Binary code
0	0000 0000
0.5	0001 1101
1.0	0011 0011
1.5	0100 1011
2.0	0110 0111
2.5	1000 0000
3.0	1001 1100
3.5	1011 0010
4.0	1100 1111
4.5	1110 1001
5.0	1111 1111

จากขั้นตอนทั้งหมดของการแปลงสัญญาณอนาล็อก เป็น ดิจิตอล นั้นเราสามารถกำหนดความ สามารถ หรือ คุณภาพในการแปลงได้ ด้วยปัจจัยหนึ่งที่สำคัญมากก็คือ ความละเอียด (Resolution) ซึ่งขึ้นอยู่กับจำนวนบิตนั้นเอง แต่ในการใช้งานจริงแล้วยังมีเรื่องความ เร็วในการแปลงอีกด้วย

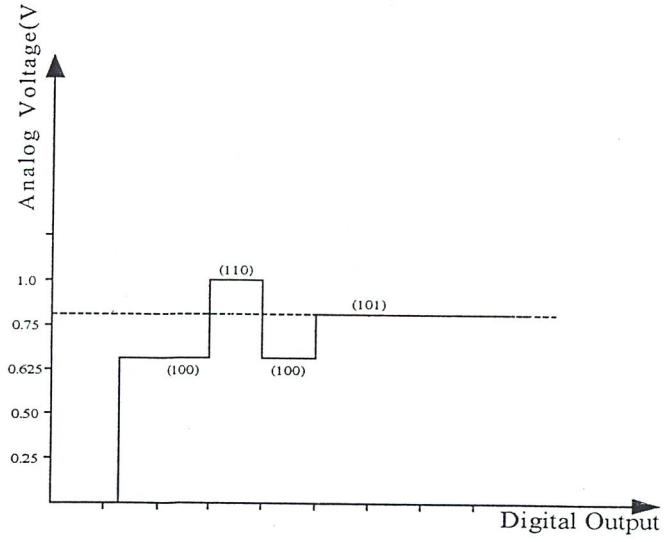
2.3 การเปลี่ยนสัญญาณอนาลอกเป็นสัญญาณเชิงเลขแบบประมาณรวบยอด (Successive Approximation)

เทคนิคการประมาณค่าหลายๆครั้ง (ประมาณรวบยอด) เป็นวงจร ADC ที่ได้รับความนิยมและเหมาะต่อการเลือกใช้เนื่องจากราคาถูกความเร็วปานกลางและค่อนข้างสูงมีความละเอียดพอสมควร ซึ่งสามารถใช้ในการแปลง สัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลได้ดี และมีประสิทธิภาพเพราะไม่

เกิดการออสซิลเลตแต่ว่ากระบวนการของเทคนิคแบบนี้จะเข้าใจได้ยากกว่าเทคนิคแบบอื่นๆซึ่งหัวใจของ SA Converter ก็คือ อุปกรณ์ " Successive Approximation Register : SAR" ซึ่งเป็นอุปกรณ์ที่มีจุดประสงค์แตกต่างจากวงจรนับทั่วไปอย่างมาก เพราะเป็นไอซี MSI (Medium Scale Integrated Circuit) ที่ได้รับการออกแบบเป็นพิเศษเพื่อทำหน้าที่นี้โดยเฉพาะและการจัดวงจรคล้ายกับแบบเคาน์เตอร์ที่ทำงานในลักษณะการป้อนกลับ ซึ่งบล็อกไดอะแกรมได้แสดงไว้ในรูปที่ 2-5



รูปที่2-5 บล็อกไดอะแกรมของ Successive Approximation ADC



รูปที่2-6 แสดงไดอะแกรมเวลาของการทำงาน

จากรูปที่2-6 แสดงไทม์มิ่งไดอะแกรมของ ADC ที่มีระดับอนาล็อก 0.625 V เมื่อป้อนสัญญาณนาฬิกา เข้าไป 1 ลูกจะ ทำให้ MSB บิตที่ 4 เป็น " 1 " ทุกบิตอื่นยังคงเป็น " 0 " จะเปลี่ยนเอาต์พุต ของ SAR เป็นอนาล็อกเปรียบเทียบกับสัญญาณอนาล็อกอินพุตถ้าผลการเปรียบเทียบ แรงดันบอกว่าน้อยกว่าอินพุตก็ให้คงบิตนั้นเป็น " 1 " ไว้ แต่ถ้าว่ามากกว่าจะให้บิตนั้นเป็น " 0 " จากนั้นทำการทดสอบบิตถัดไปโดยทำให้เป็น " 1 " หากผลรวมของสองบิตหรือบิตหลังมากกว่าก็ทำ ให้บิตนั้นเป็น " 0 " แต่ถ้า น้อยกว่าให้คง " 1 " ไว้แล้วทดสอบบิตถัดไปตามกรรมวิธีดังกล่าวจนครบ ทุกบิตหรือจนกว่าเอาต์พุตจะต่างจาก Vin ไม่เกิน 1 LSB

มีข้อจำกัดประการหนึ่ง สำหรับการใช้งานคือ สัญญาณอนาล็อกอินพุต จะต้องคงที่ ในช่วงเวลาที่ทำการเปลี่ยนแปลงสัญญาณ โดยเปลี่ยนได้ไม่เกิน ½ LSB วงจร ADC แบบนี้ สามารถทำงานได้สองโหมด คือ โหมดที่ทำงานโดยอิสระ (Free running) และโหมดที่รอคำสั่ง Start conversion จากภายนอก เวลาที่ใช้ในการเปลี่ยนแปลงสัญญาณใช้ (n+1) ลูกของพัลส์ clock โดย clock ลูกแรกจะใช้ในการรีเซ็ทรีจิสเตอร์ภายในสุดท้ายคุณภาพของระบบนี้จะขึ้นอยู่กับคุณภาพของวงจรเปลี่ยนสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (DAC) ด้วย

2.4 ไอซีสำหรับการเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล ADC0808 :8 bit A/D

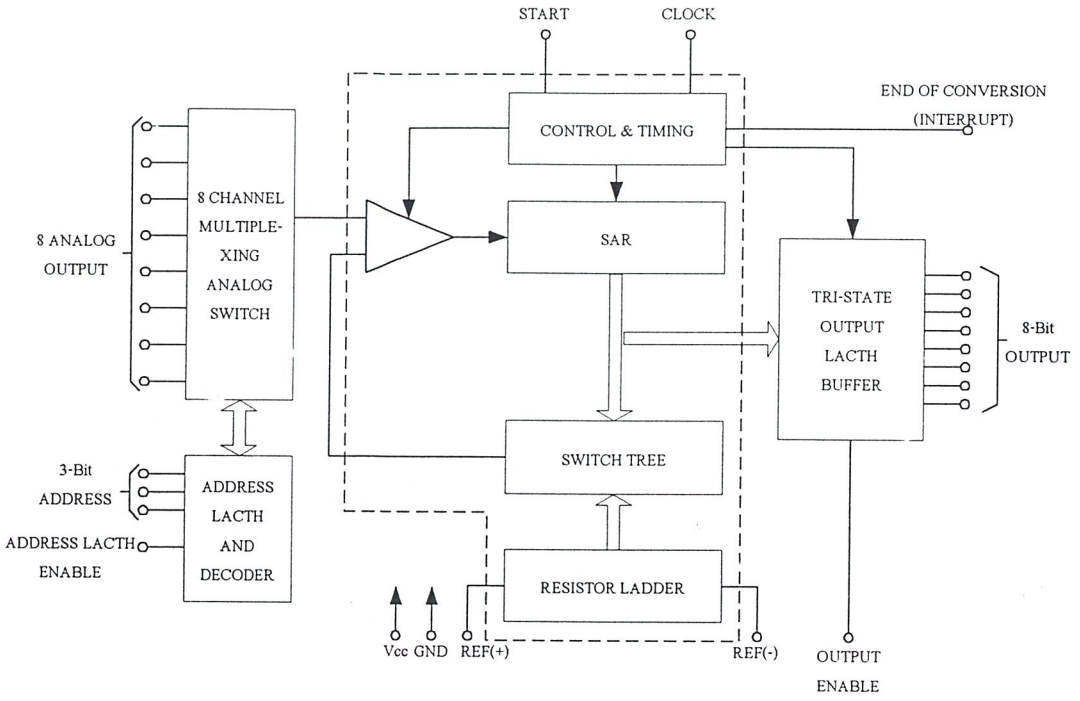
ไอซีเบอร์ ADC0808 นี้เป็น อุปกรณ์ที่ สามารถแปลงสัญญาณอนาล็อก เป็น ดิจิตอลขนาด 8 บิต สามารถรับอินพุตได้ 8 ช่องและมัลติเพล็กซ์สัญญาณได้ด้วยสามารถเชื่อมต่อกับไมโครโปรเซสเซอร์ ได้ง่ายใช้วิธีการแปลงแบบ Successive Approximation สามมารถต่อกับอุปกรณ์ TTL และทำงานในลักษณะ Tri-Stateได้

ลักษณะที่สำคัญของ ADC0808

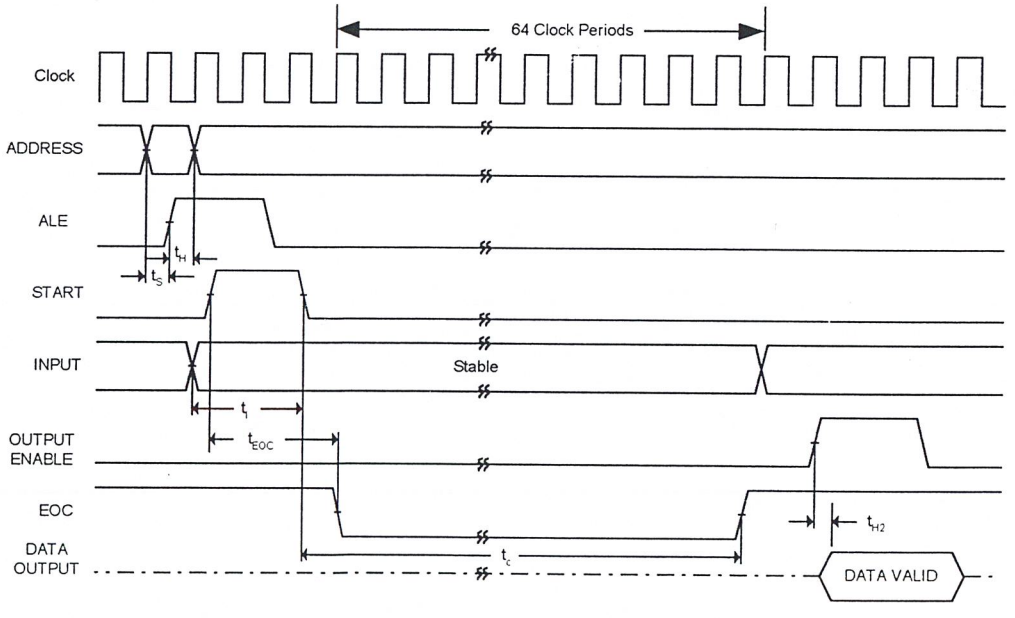
- เชื่อมต่อกับไมโครคอนโทรลเลอร์ทุกเบอร์ได้ง่าย
- สามารถปรับแรงดันอ้างอิงได้ในช่วง 0 ถึง + 5Vdc
- มัลติเพล็กซ์ได้ถึง 8 ช่องด้วย address logic
- แรงดันเอาท์พุตเป็น TTL
- อยู่ในรูปของไอซี 28 ขาแบบ DIP

คุณสมบัติที่สำคัญของ ADC0808

- ความละเอียด (Resolution) 8 บิต
- ค่าผิดพลาดสูงสุดอยู่ระหว่าง ± ½ LSB
- แหล่งจ่ายเดียว 5 Vdc
- กินกำลังไฟฟ้าต่ำที่ 15mW
- Convertion time 100 μS



รูปที่ 2-7 บล็อกไดอะแกรมของ ADC0808



รูปที่ 2-8 ไทม์มิ่งไดอะแกรมของ ADC0808

จากรูปที่ 2-8 แสดงถึงไทม์มิ่งไดอะแกรมการทำงานของ ADC0808 เพื่อที่จะเริ่มการแปลงสัญญาณนั้นเราจะต้องป้อนสัญญาณพัลส์ (สัญญาณ Clock) ที่ขา START โดยที่สัญญาณขอบขาขึ้นจะเป็นการเครียร์รีจิสเตอร์ภายใน และที่สัญญาณขาลงจะเป็นการเริ่มต้นการแปลงสัญญาณ

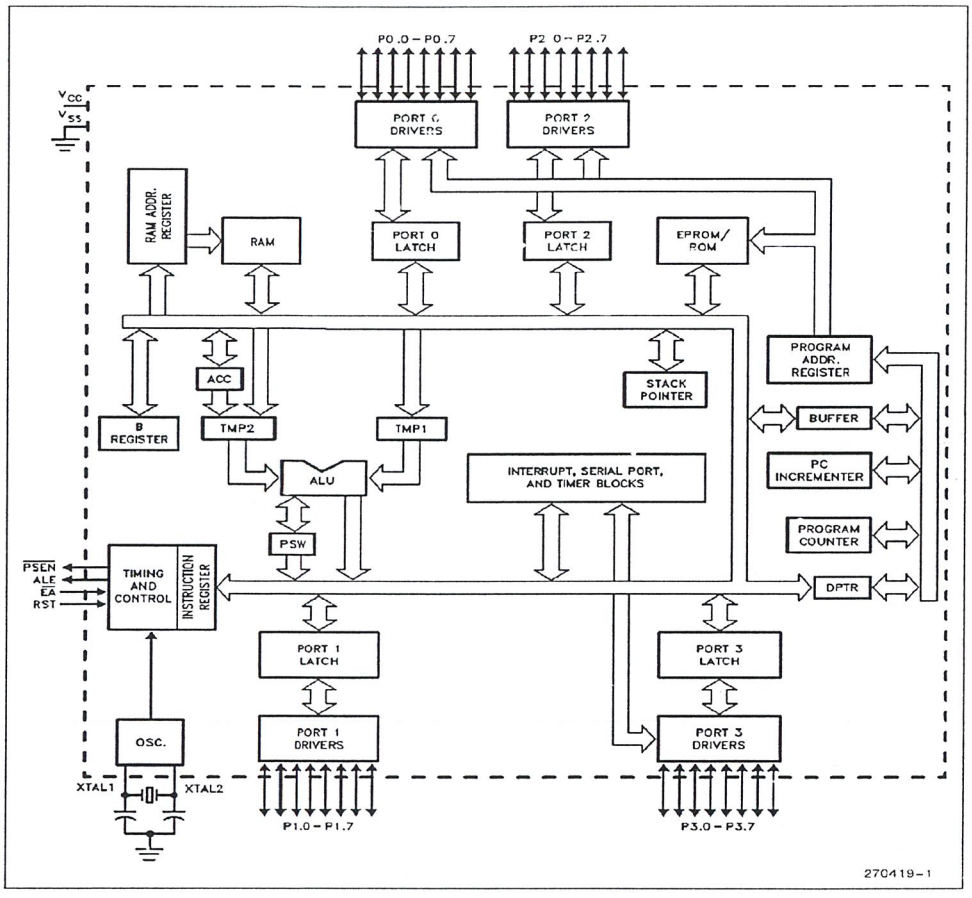
บทที่ 3
ทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่บรรจุความสามารถมากมาย ไม่ว่าจะเป็นหน่วยประมวลผลหน่วยคำนวณทางคณิตศาสตร์และลอจิกวงจรรับสัญญาณอินพุตวงจรขับสัญญาณออกทางเอาต์พุตหน่วยความจำวงจรกำเนิดสัญญาณนาฬิกาทำให้ไมโครคอนโทรลเลอร์สามารถนำไปประยุกต์ใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อน ได้เป็นอย่างดี โดยช่วยลดจำนวนของอุปกรณ์และขนาดของระบบลงในขณะที่มีขีดความสามารถสูงขึ้นดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้นิยมในการควบคุม โดยสามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

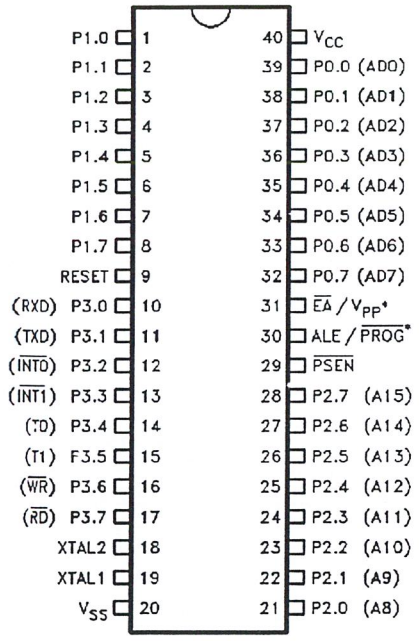
3.1 คุณสมบัติไมโครคอนโทรลเลอร์ MCS-51

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียู 8 บิต
- ภายในมีหน่วยความจำเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- สายอินพุตและเอาต์พุตมีจำนวน 32 เส้นใช้เลือกแอดเดรสแยกต่างหากจากกันได้
- มีแรมบรรจุภายในขนาด 128 ไบต์ หรือ 256 บิต
- วงจรตั้งเวลา/วงจรนับมีขนาด 16 บิต อย่างน้อย 2 ตัว
- กำหนดเป็น UART (Universal Synchronous Asynchronous Receiver Transmitter) ส่งข้อมูลอนุกรมได้สองทิศทาง
- อินเตอร์รัปต์ แบ่งเป็นสองระดับจาก 5 หรือ 6 แหล่ง
- มีสัญญาณนาฬิกา อยู่ภายในตัว
- มีหน่วยความจำสำหรับเก็บข้อมูลภายในขนาด 4 หรือ 8 กิโลไบต์
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- คำสั่งทั้งหมดมี 111 คำสั่ง
- ทำงานด้วยเลขฐานสิบ และเลขฐานสิบหก
- ในเบอร์ AT89C2051, AT89C1051, จะลดขนาดลงเหลือแค่ 20 ขาโดยมีROM ภายในจำนวน 2 กิโลไบต์และ 1 กิโลไบต์ ตามลำดับ โดยจะไม่มี Port 0, Port 2, ALE, PSEN, EA และ RD เนื่องจากได้อัดโปรแกรมไว้ในตัวแล้วขาเหล่านี้จะไม่ได้ใช้งาน

3.2 โครงสร้างสถาปัตยกรรมของ MCS-51



รูปที่ 3-1 โครงสร้างภายในของ MCS-51



รูปที่ 3-2 ตำแหน่ง ขาของชิป ไมโครคอนโทรเลอร์ MCS-51

หน้าที่การใช้งานแต่ละขาของชิปไมโครคอนโทรลเลอร์ MCS-51 มีดังนี้

- ขา Vss (ขา 20)สำหรับต่อลงกราวด์
- ขา Vcc (ขา 40)สำหรับต่อแหล่งจ่ายแรงดันกระแสไฟตรงขนาด 5 โวลต์
- ขาพอร์ต 0 (ขา 32-39) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต(P0.0-P0.7) แบบ Open drain Bi-directional พอร์ตนี้สามารถใช้งานเป็นพอร์ตอินพุต เอาต์พุตทั่ว ไปได้ และใช้ในการติดต่อ หน่วย ความจำสำหรับโปรแกรมและข้อมูลภายนอกชิปด้วย
- ขาพอร์ต 1 (ขา 1-8) มี 8 ขาใช้สำหรับพอร์ต 1 (P1.0-P1.7) สามารถใช้เป็นอินพุต หรือเอาต์พุตพอร์ต ทั่วไป ได้ หากต้องการใช้งานเป็นพอร์ตอินพุต ต้องโหลดค่า ไปยังแต่ละ บิตของพอร์ตนี้
- ขาพอร์ต 2 (ขา 21-28) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 2 ขนาด 8 (P2.0-P2.7) มีโครงสร้างคล้ายพอร์ต 0 โดยมี FET ตัวล่างตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน พอร์ตนี้ทำงานได้ สองหน้าที่ คือ สามารถใช้เป็น แอดเดรสบัส ขนาด 8 บิต (A8-A15) และเป็นพอร์ตอินพุต เอาต์พุตใช้งานทั่วไป เมื่อจะใช้งานเป็นพอร์ตอินพุต ต้องส่ง ลอจิก “1” มาที่พอร์ตนี้ก่อนเพื่อบังคับให้ FET อยู่ในสภาวะ off
- ขาพอร์ต 3 (ขา 10-17) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 3 (P3.0-P3.7) ขนาด 8 บิต พอร์ตนี้สามารถ ใช้งาน เป็น อินพุต เอาต์พุตพอร์ตทั่วไปได้ นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆ ดังนี้

ขา P3.0 ใช้รับข้อมูลจากภายนอกแบบอนุกรม

ขา P3.1 ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม

ขา P3.2 ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิดที่ 0

ขา P3.3 ใช้เป็นอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์ชนิดที่ 1

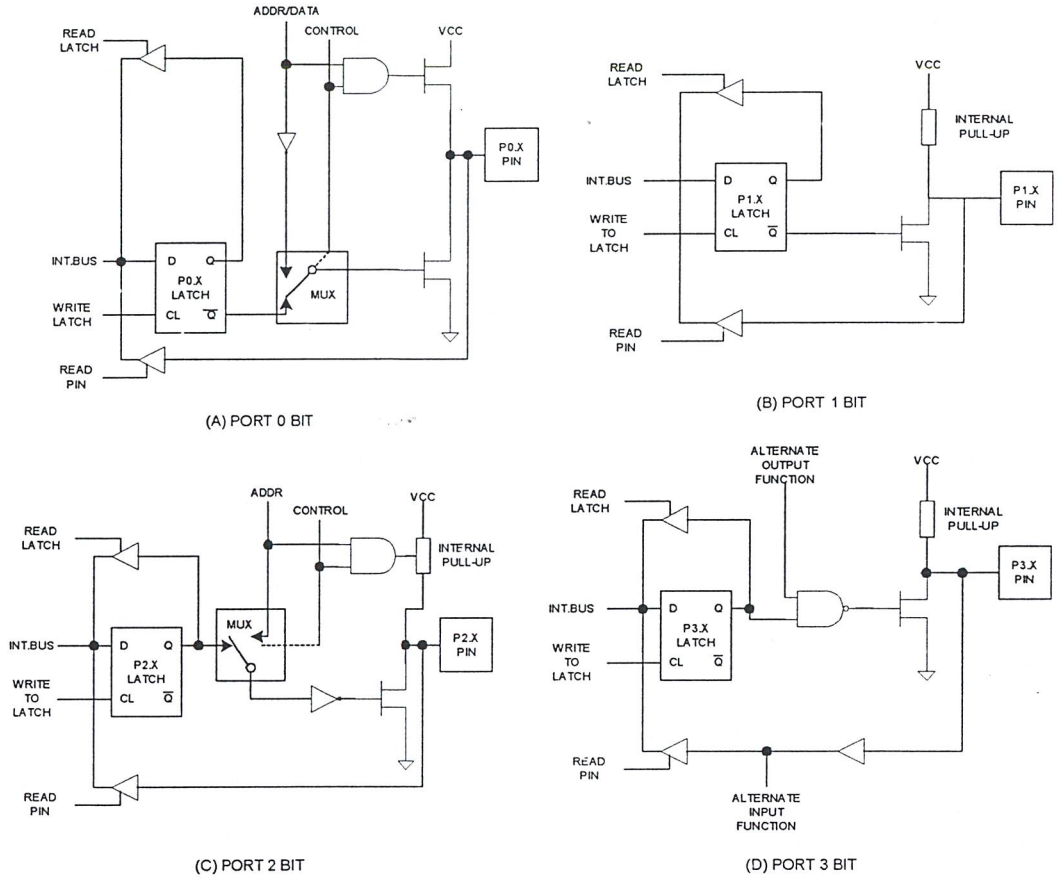
ขา P3.4 สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 0

ขา P3.5 สัญญาณอินพุต ให้เคาน์ เตอร์ของ ไทม์เมอร์ 1

ขา P3.6 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บภายนอกชิป

ขา P3.7 ใช้เป็น สัญญาณ ควบคุมการเขียน ข้อมูลจากนหน่วย ความสำหรับ เก็บข้อมูล ภาย นอกชิป

- ขา RT (ขา 9) ใช้สำหรับการรีเซ็ตวงจรทุกอย่างภายในชิป เพื่อเริ่มต้นการทำงานใหม่
- ขา ALE / PROG (ขา 30) เป็นขาสำหรับใช้ส่งสัญญาณออกไปภายนอก เพื่อควบ คุมการแลตช์ค่า แอดเดรสไบต์ต่ำ (Address Latch Enable) พอร์ต 0 ในการติดต่อ หน่วยความจำสำหรับโปร แกรม หรือข้อมูลภายนอก สัญญาณนี้จะแอตทีฟทุกๆ 2 ครั้งใน 1 แมชชีนไซเคิล
- ขา PSEN (ขา 29) ใช้ส่งสัญญาณสโตรบ เพื่ออ่านคำสั่งจากโปรแกรมที่ เก็บไว้ในหน่วยความ จำภายนอกชิป (Program Strobe Enable)
  สัญญาณนี้จะส่งออกมาสองครั้งในแต่ละแมชชีนไซเคิลแต่ถ้าเป็นการอ่าน
  คำสั่งจากหน่วยความจำโปรแกรมภายในจะไม่มีสัญญาณออกที่ขานี้
- ขา EA/Vpp (ขา 31) เป็นขาสำหรับใช้เลือกให้ MCS-51 ทำงานจากโปรแกรมที่อยู่ภายในชิป หรือ ภายนอกชิป หากขานี้มีสถานะเป็น 0 หมายถึงให้ใช้โปรแกรมที่อยู่ภายนอก หากขานี้มีสถา นะเป็น 1 หมายถึงบังคับให้ MCS-51 ใช้โปรแกรมจากหน่วยความจำสำหรับ โปรแกรมภายในชิป ส่วน MSC-51 ที่ไม่มีหน่วยความจำโปรแกรมสำหรับเก็บโปรแกรมภายในชิป ให้ต่อขานี้ ลงกราวด์ เสมอ
- ขา XTAL1 (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิลเลเตอร์
- ขา XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอก โดยเป็นเอาต์พุตออกจากวงจรออสซิลเลเตอร์



รูปที่ 3-3 แสดงโครงสร้างแต่ละบิตภายในพอร์ตอินพุท/เอาท์พุทของ 8051

3.3 การจัดหน่วยความจำ

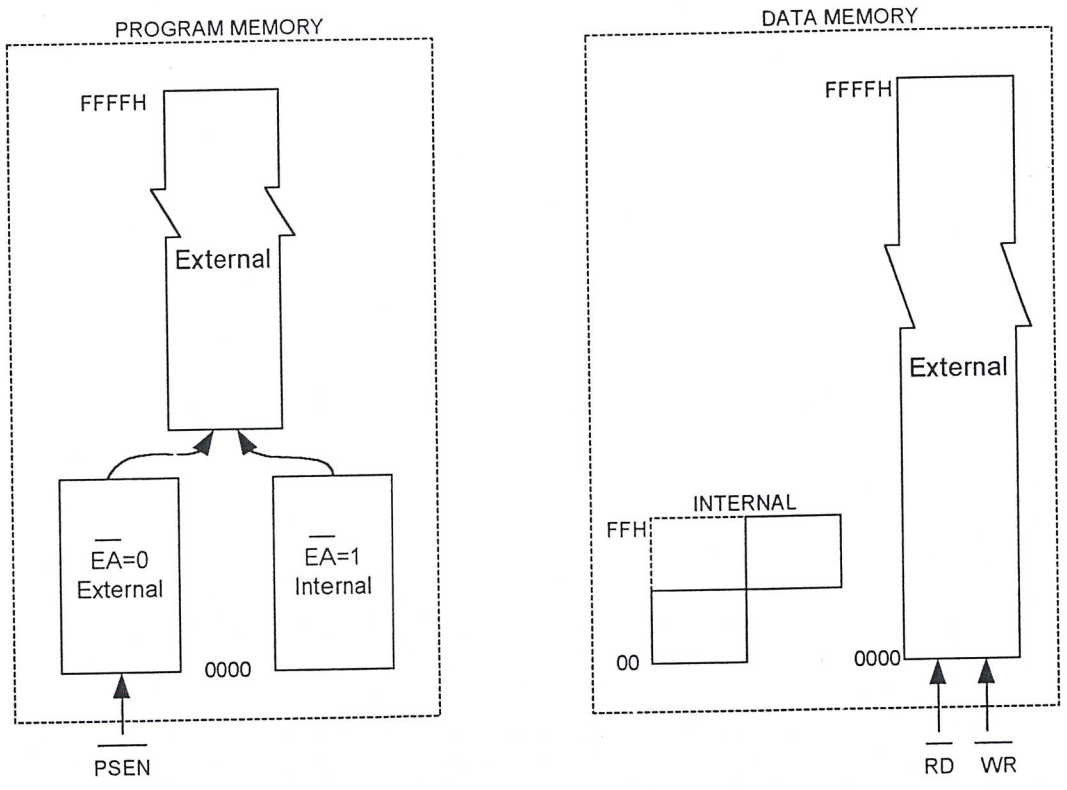
หน่วยความจำที่ใช้กับ MCS –51 มีอยู่ด้วยกัน 2 แบบคือ

- หน่วยความจำสำหรับโปรแกรม(Program Memory)
- หน่วยความจำสำหรับเก็บข้อมูล (Data Memory)

3.3.1 Program Memory

เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง (Machine Language) ซึ่ง ต้องการ ให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็ต้องอ่านข้อมูล ที่เก็บในหน่วยความจำประเภทนี้ เข้าไปถอดรหัส แล้วสร้าง สัญญาณควบคุมส่วนอื่น ๆ ตามการทำงานของแต่ละคำสั่งหน่วยความจำแบบนี้ จะต้องเป็นแบบ Read Only Memory (ROM) และ ผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของ หน่วยความจำเป็น รหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ การเขียนข้อมูลลงไปบน ROMจะต้องใช้เครื่องมือพิเศษ ในระหว่างการทำงานของ 8051 ผู้ใช้จะไม่สามารถใช้คำสั่ง ทำงานเขียนข้อมูลลงในหน่วยความจำแบบนี้ได้จำนวนตำแหน่งสูงสุดของหน่วยความจำแบบนี้ที่ 8051 จะใช้งานได้คือ 65536 ตำแหน่ง ค่าของ

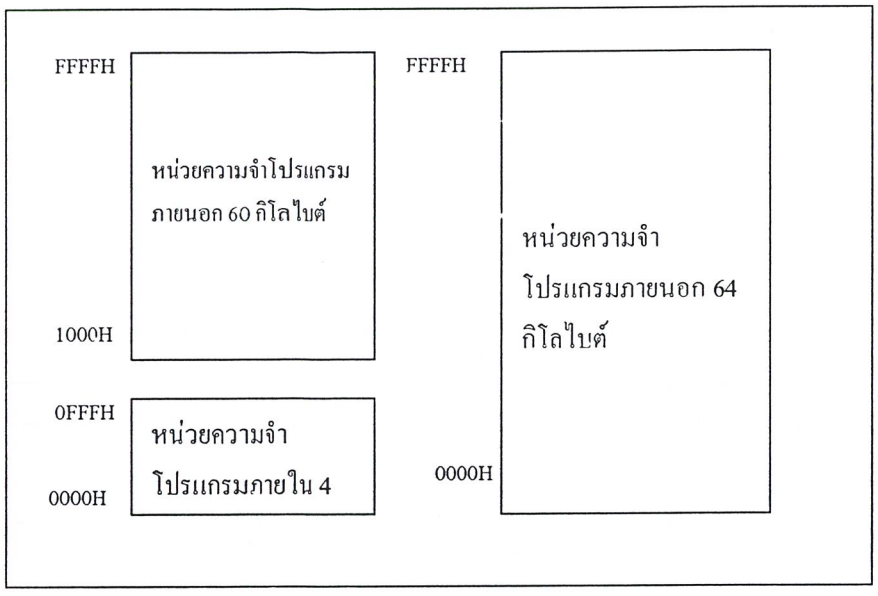
ตำแหน่ง (Address) จะเขียนเป็นเลขฐาน 16 ได้ตั้ง แต่ 0000-FFFFh หน่วยความจำตำแหน่ง 0000-0FFFh จำนวน 4 กิโลไบต์ นั้นผู้ใช้จะเลือกได้ว่าเป็นตำแหน่งของ ROM ที่ภายในหรือภายนอก 8051 ถ้าต้องการให้ 8051 ทำงานตามคำสั่งที่เก็บไว้ใน ROM ภายใน 8051 ก็ให้ป้อนสัญญาณสภาวะลอจิก High (1) เข้าที่ขา EA ของ 8051 แต่ถ้าต้องการให้ทำงานในโปรแกรมที่เก็บไว้ใน ROM ภายนอก 8051 ก็ให้ต่อลอจิก Low (0) เข้าที่ขา EA ของ 8051 ส่วนหน่วยความ จำตำแหน่ง 1FFFh ถึง FFFFh จะต้องต่ออยู่ภายนอก 8051 เสมอดังแสดงใน แผนภูมิหน่วยความจำ (Memory Map) ในรูปที่ 3-4



รูปที่ 3-4 แผนภูมิหน่วยความจำ

ในพื้นที่ ของหน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายใน หรือภายนอกก็ตาม ต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับการบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

- พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 0003H
- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH
- พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H
- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH
- พื้นที่สำหรับบริการอินเตอร์รัปตของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H
- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

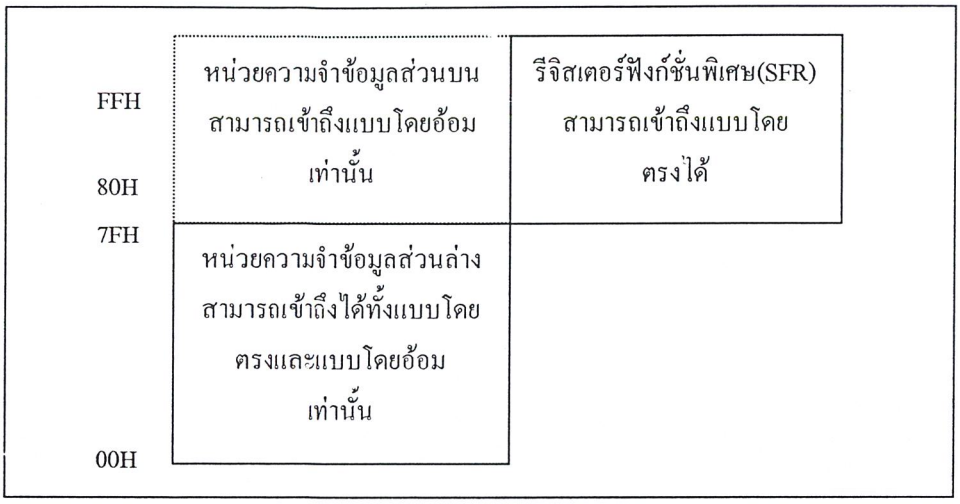


รูปที่ 3-5 การจัดสรรหน่วยความจำโปรแกรม ของไมโครคอนโทรลเลอร์ MCS-51

Internal Memory หมายถึงหน่วยความจำนั้นจะอยู่ภายใน 8051 ส่วน External Memory หมายถึง หน่วยความจำนั้นอยู่ภายนอก 8051

3.3.2 Data Memory

เป็นหน่วยความจำที่ 8051 จะใช้สำหรับพัก, เก็บข้อมูล แล้วเรียก มาใช้ใหม่ ในระ หว่าง การทำงานของ 8051 การอ่านหรือการ เขียนข้อมูลจากหน่วย ความจำจะกระทำ โดยคำสั่ง ที่เก็บ ไว้ใน Program Memory หน่วยความจำประเภทนี้เป็นประเภท Random Access Memory ถ้ามีไฟเลี้ยงอยู่ ข้อมูลที่เก็บไว้จะไม่สูญหาย หน่วยความจำแบบ Data Memory ของ 8051 จะมีอยู่ 2 ชุด ชุด หนึ่ง อยู่ภายใน 8051 จำนวน 128 ไบต์ ที่ตำแหน่ง 00h-7Fh และอีกชุดหนึ่งจะต้องต่ออยู่ภายนอกของ 8051 มีได้สูงสุด 65536 ไบต์ อยู่ที่ตำแหน่ง 0000-FFFFh ดังแสดงในรูปที่ 3-5 โดยหน่วยความจำแบบ Data Memory ใน 8051 ที่ตำแหน่ง 80h-FFh นั้นไม่ได้มี อยู่ทุกตำแหน่ง จะมีอยู่ เฉพาะ ในบางตำแหน่ง ซึ่งเรียกหน่วยความจำบางตำแหน่งนี้ว่า Special Function Register (SFR) เพราะจะใช้หน่วย ความจำเหล่านี้สำหรับงานพิเศษเท่านั้น อาจเป็น RAM หรือวงจรนับ, วงจรตั้งเวลาก็ได้



รูปที่3-6 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51

จะเห็นได้ว่าหน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งทับซ้อนกัน แต่จะใช้การติดต่อที่แตกต่างกัน โดยในหน่วยความจำข้อมูลส่วนล่างขนาด 128 ไบต์มีแอดเดรส อยู่ที่ 00H-7FH สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128 ไบต์เช่นกัน มีแอดเดรสอยู่ที่ 80H-FFH แต่สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ ฟังก์ชั่นพิเศษมีแอดเดรสอยู่ที่ 80H-FFH เช่นเดียวกันกับหน่วยความจำข้อมูลส่วนบน แต่สำหรับรีจิส เตอร์ฟังก์ชั่นพิเศษใช้การเข้าถึงแบบโดยตรง

ตารางที่ 3-1 แสดงตำแหน่งแอดเดรสของรีจิสเตอร์ใช้งานทั่วไป

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

ตารางที่ 3-2 แสดงการเลือก รีจิส เตอร์แบงค์ของรีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์	บิต RS0	บิต RS1	ตำแหน่งหน่วยความจำ
แบงค์ 0	0	0	0000H
แบงค์1	0	1	0008H
แบงค์ 2	1	0	0010H
แบงค์ 3	1	1	0018H

3.4 พอร์ตอินพุท/เอาท์พุทของ 8051

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีโครงสร้างของพอร์ตที่สามารถ ใช้งานแบบขนานได้จำนวนทั้งหมด 4 พอร์ต เรียกชื่อเรียงตามลำดับว่า พอร์ต 0, 1, 2 และ 3 ตามลำดับ เป็นพอร์ตขนาด 8 บิททั้งหมดการใช้งานพอร์ตสามารถทำได้ทั้งในลักษณะของเส้นสัญญาณเดี่ยวๆ หรือกลุ่มของ สัญญาณก็ได้ นอกจากนี้พอร์ต 0-3 ยังสามารถนำไป ใช้งานอื่นๆที่ไม่ใช่พอร์ตอิน พุท/เอาท์พุทได้ โดยพอร์ต 0 ทำหน้าที่มัลติเพลกซ์ระหว่างบัสแอดเดรสไบต์ต่ำ และบัสข้อมูล

ซึ่งพอร์ต 0 นั้นไม่สามารถที่จะใช้งานในการอ้างแอดเดรสไบต์ต่ำ และอ่านเขียนข้อมูลไปยัง RAM ภายนอกได้พร้อมๆกัน ด้วยเหตุนี้จึงมีการต่อ ไอซี 74HC573 ร่วมเข้าไปเพื่อช่วยในการ แลตซ์แอดเดรสไบต์ต่ำให้กับหน่วยความจำภายนอก โดยขา ALE ของ MCU(Microcontroller Unit) จะเป็นตัว ควบคุมจังหวะการ แลตซ์ ให้กับ ไอซี 74HC573 และขณะที่ ไอซี 74HC573 แลตซ์แอดเดรส ไบต์ ต่ำอยู่ นั้นพอร์ต 0 จะทำการอ่านเขียน ข้อมูลกับหน่วยความจำภายนอกได้ นั้น เอง ซึ่งจากวงจรจะเห็นว่าหน่วยความจำภายนอกนั้น จะมีสัญญาณในการอ่าน เขียนข้อมูล โดยใช้ขาสัญญาณ $\overline{RD}$ และ $\overline{WR}$ ของ MCU เป็นตัวควบคุม ในพอร์ต2นั้น ไม่มีปัญหา แต่อย่างใด เพราะ ใช้ในการ อ้างแอดเดรส ไบต์สูงเพียง หน้าที่เดียวเท่านั้น

สำหรับพอร์ต 3 นั้นนอกเหนือจากความสามารถเช่นพอร์ตปกติแล้ว สามารถนำไปเป็นขาสัญญาณ ของการอินเตอร์รัพท์ต่างๆ รวมทั้งสร้างสัญญาณควบคุม $\overline{RD}$ และ $\overline{WR}$ เพื่อทำหน้าที่อ่าน หรือเขียนหน่วยความจำข้อมูลภายนอกด้วย การใช้งานพอร์ต ลักษณะงานแบบอื่นๆ ที่ไม่ใช้เป็นพอร์ตแบบอินพุท/เอาท์พุตนี้จะดำเนินการโดย ซีพียูโดยอัตโนมัติ

3.4.1 การใช้งานเป็นพอร์ตอินพุท

การใช้งานพอร์ตเป็นการอินพุทข้อมูลนั้น ก่อนอื่นจะต้องส่งข้อมูลค่า " 1 " ออกมาที่บต ของพอร์ตก่อนเพื่อหยุดการทำงานของทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทของบิตนั้น ทำให้ ขาสัญญาณของบิตถูกต่อเข้ากับ ตัวต้านทาน ซึ่งทำหน้าที่ Pull-up ภายใน ซึ่งมีผลให้บิตนั้นๆ ของ พอร์ต 1, 2 และ 3 มีสภาวะลอจิกสูง ตัวต้านทานเหล่านี้มีค่าประมาณ 50K ซึ่งเป็นค่าที่สูงมาก ทำให้อุปกรณ์ภายนอกสามารถขับสัญญาณของพอร์ตเหล่านี้เป็นลอจิกต่ำได้ง่าย สำหรับบิทของพอร์ต 0 นั้นแม้ว่าจะมีการทำงานที่ คล้ายคลึง กับ บิตของพอร์ตอื่นๆ แต่เนื่องจากการที่ไม่มีตัวต้านทาน ทำหน้าที่ Pull-up ภายในไว้ทำให้เมื่อทรานซิสเตอร์ที่ทำหน้าที่ ขับสัญญาณเอาท์พุทนั้น หยุดทำงาน ก็จะเป็นผลให้ ขา สัญญาณนี้อยู่ในสภาวะ High Impedance แทน

3.4.2 การใช้งานเป็นพอร์ตเอาท์พุท

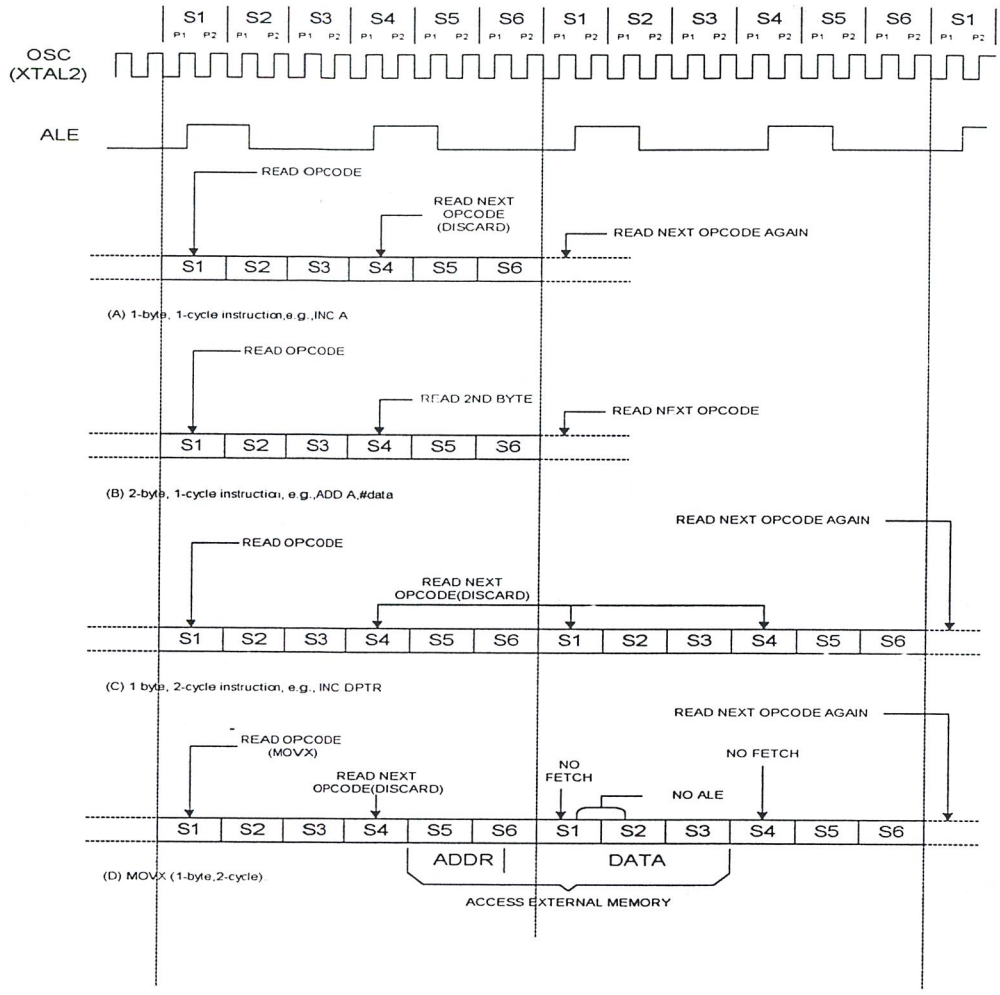
เมื่อมีการส่งข้อมูลที่มีค่าเป็น 0 ให้กับแต่ละบิตของพอร์ตทุกพอร์ต ข้อมูลนี้จะถูกส่งให้กับ ฟลิป-ฟลอป ซึ่งจะค้างค่านี้เอาไว้ และ มีผ ทำให้ ทรานซิสเตอร์ ที่ ทำหน้าที่ขับ สัญญาณเอาท์พุท นั้นทำ งาน ดังนั้นขาสัญญาณก็จะมีสภาวะเป็นลอจิกต่ำด้วย ส่วนการส่งข้อมูลที่มีค่ เป็น 1 ออกมานั้น ใน กรณี ที่เป็นการทำงาน ในแต่ละบิตของ พอร์ต 1 , 2 หรือ 3 จะทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณ เอาท์พุทนั้นหยุดท่างาน มีผลทำให้ขาของสัญญาณเป็น ลอจิกสูงด้วยตัวต้านทานที่ Pull-up อยู่ภายใน นั้น แต่สำหรับการทำงานในแต่ละบิตของพอร์ต 0 จะมีผลที่แตกต่างออกไป โดยขาสัญญาณจะเป็น สภาวะอิมพีแดนซ์สูงแทน เนื่องจากไม่มีตัวต้าน ทานภายในเชื่อมต่ออยู่นั่นเอง ดังนั้นในการใช้งาน พอร์ต 0 เป็นการเอาท์พุทข้อมูล จึงจำเป็นต้องใช้ตัวต้านทานภายนอก Pull-up สัญญาณไว้กับไฟเลี้ยงแทน

3.5 ฐานเวลาในการทำงานของซีพียู 8051

ซีพียูตระกูล MCS-51 มีวงจรออสซิเลเตอร์อยู่ภายใน สำหรับการสร้างพัลซ์ของสัญญาณนาฬิกาซึ่งจะนำไปเป็นฐานเวลาหรือ การกำหนดจังหวะการทำงานของหน่วย การทำงาน ทั้งหมดให้สอดคล้องกัน (Synchronization) โดยปกติแล้วก็มักจะทำโดยการใช้คริสตอลเชื่อมต่อ เข้ากับขาสัญญาณ XTAL1 และ XTAL2 หรืออาจจะใช้สัญญาณนาฬิกาจากภายนอกก็ได้

พัลซ์ความถี่ของสัญญาณนาฬิกาจะเรียกว่า Pulse และคาบของสัญญาณนาฬิกานี้เรียกว่า คาบ เวลาออสซิเลเตอร์ (Oscillator Period) คาบเวลาออสซิเลเตอร์จำนวน 2 คาบเรียกว่า State ซึ่งจะนำไปใช้เป็นช่วงเวลาพื้นฐานการทำงานย่อยของไมโครคอนโทรลเลอร์ ช่วงเวลาของ State จำนวน 6 ครั้งเรียกว่า แมชชีนไซเคิล (Machine Cycle) ดังนั้นค่าเวลาหนึ่งแมชชีนไซเคิลจะใช้เวลา 12 คาบเวลาออสซิเลเตอร์ ค่าของแมชชีนไซเคิลนี้จัดเป็นช่วง เวลาที่น้อยที่สุด ในการทำคำสั่งใดคำสั่งหนึ่ง ซึ่งหากว่าเป็นคำสั่งที่ซับ ซ้อนมาก ก็ต้องใช้เวลานาน สองถึงสาม แมชชีนไซเคิล

จากรูปที่ 3-7 จะเห็นว่าหากทำคำสั่งที่ใช้เวลา 1 แมชชีนไซเคิลแล้ว จะมีการอ่านคำสั่งที่สเตท 1 หากคำสั่งมีขนาด 2 ไบท์ ก็จะมีการ Fetch ไบท์ที่สองของคำสั่งที่สเตทที่ 4 ของคำสั่งส่วนคำสั่งที่ ใช้เวลา 2 แมชชีนไซเคิลนั้น จะมีการอ่าน Opcode ที่สเตทแรก และระหว่างนั้นจะไม่มีการ Fetch คำสั่งถัดไปเข้ามา จนกว่าจะทำคำสั่งนั้นเสร็จสิ้น จึงจะเริ่ม Fetch คำสั่งถัดไป



รูปที่ 3-7 ไซเคิลการทำงานของไมโครคอนโทรลเลอร์ MCS-51

คำว่า 1 แมชชีนไซเคิล คือช่วงเวลาการทำงาน ตั้งแต่ S1 จนถึง S6 การทำงานในหนึ่งคำสั่งจะใช้เวลาเพียง 1 μs ปรกติแล้วซีพียูจะ Run ด้วยความเร็วเท่ากับ 12 MHz ดังนั้นคล็อค 12 ลูกจะใช้เวลาเท่ากับ 1 μs

3.6 การอินเตอร์เฟสโดยพอร์ต RS-232

ลักษณะของการส่งข้อมูลแบบอนุกรมนั้น ข้อมูลจะถูกส่งออกมาทีละบิต จากตัวอุปกรณ์ส่งไปยังอุปกรณ์รับ ช่องสัญญาณในการส่งข้อมูลอาจจะใช้เพียง 1 หรือ 2 ช่องสัญญาณเท่านั้น ทำให้ค่าใช้จ่ายในการสื่อสารถูกกว่าแบบขนาน แต่อัตราการส่งข้อมูลจะช้ากว่าการส่งแบบขนาน ในการส่งข้อมูลแบบอนุกรม ข้อมูลที่ต้องการส่งจะอยู่ในลักษณะเป็นไบต์ จะถูกทยอยส่งทีละบิต และทาง อุปกรณ์รับจะต้องรับข้อมูลเข้ามาทีละบิต แล้วมารวมกันเป็นไบต์ซึ่งทางด้านอุปกรณ์รับจะต้องคอย ตรวจสอบว่าบิตใดเป็นบิตเริ่มแรกของไบต์นั้น การตรวจสอบจะขึ้น อยู่กับรูปแบบของรหัสของ บิตที่ใช้ในการรับส่ง

ข้อมูลแบบอนุกรม ระหว่างไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกนั้น จำเป็น ต้องมีมาตรฐานในการรับส่ง ซึ่งมาตรฐานที่นิยมมากที่สุดก็คือมาตรฐาน RS-232

3.6.1 มาตรฐาน RS-232

เพื่อที่จะให้อุปกรณ์ จากผู้ผลิตต่างกันทำงานร่วมกันได้ มาตรฐานหลายชนิดจึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันอย่างกว้างขวางที่สุดคือ RS-232C ซึ่งโดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรมอยู่ในตัวอยู่แล้วและจะทำหน้าที่รับส่งข้อมูลในแบบอนุกรม ตามจุดประสงค์ของมาตรฐาน RS-232C นั้น เพื่อจะสามารถเชื่อมต่อกันระหว่างอุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment : DTE) เช่น พอร์ตของคอมพิวเตอร์หลักหรืออุปกรณ์ ปลายทาง กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment : DCE) หมายถึงอุปกรณ์ที่สามารถแปลงรูปคลื่นดิจิตอล ไปเป็นสัญญาณที่เหมาะสมสำหรับการส่งผ่านสายโทรศัพท์ หรือตัวกลางการสื่อสารอย่างอื่นโดยกระบวน การผสมสัญญาณ (Modulation)

3.6.2 การอินเตอร์เฟสตามมาตรฐาน RS-232C

มาตรฐาน RS-232C ใช้สัญญาณเพียงเส้นเดียว ในการส่งสัญญาณ โดยสัญญาณ ที่ส่งไปได้ทิศทางเดียว สำหรับความเร็วและระยะทางของการเชื่อมต่อ RS-232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้ถึง 200 Kbps ซึ่งเพียงพอสำหรับไมโครคอนโทรลเลอร์ที่มีอัตรา 110-9600 บอด ความยาวของสายเชื่อมสัญญาณระหว่าง DTE และ DCE โดยสัญญาณตามมาตรฐานของ RS-232 จำกัดอยู่ที่ 50 ฟุตหรือประมาณ 15 เมตร ซึ่งเพียงพอสำหรับการสื่อสารไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกสำหรับแรง ดันของระดับสัญญาณจะถูกกำหนด ลงใน 2 บริเวณ คือ

- แรงดันไฟบวก (สถานะ Space) อยู่ระหว่าง +5 ถึง +15 โวลท์สำหรับเอาต์พุต และระหว่าง +3 ถึง +15โวลท์สำหรับอินพุตความแตกต่างมีไว้เผื่อกรณีที่แรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณ
- แรงดันไฟลบ (สถานะ Mark) ถูกกำหนดไว้ระหว่าง –5 ถึง –15 โวลต์ สำหรับเอาต์พุต และระหว่าง -3 ถึง -15 โวลท์สำหรับอินพุต

ซึ่งจะเห็นได้ว่าถ้าให้สายสัญญาณยาวเกินไป ระดับแรงดันไฟฟ้าจะตกลงเกินขอบเขตที่กำหนด นอกจากนี้ค่าความจุไฟฟ้าที่เกิดขึ้น จะมีผลกับคุณภาพของสัญญาณ โดยทำให้การเปลี่ยนสถานะจากแรงดันไฟบวกไปเป็นแรงดันไฟลบนั้นไม่ชัดเจน เนื่องจาก RS-232C ไม่ได้ออกแบบให้นำไปใช้กับระยะทางไกล ดังนั้นถ้าอุปกรณ์อยู่ห่างกันมาก อาจจำเป็นต้องใช้โมเด็ม หรือวิธีการอื่น ๆ แทน

บทที่ 4

การออกแบบและหลักการทำงาน

ระบบเก็บข้อมูลในโครงงานนี้ ประกอบด้วย 3 ส่วนใหญ่ๆ คือ ส่วนเซ็นเซอร์ ส่วนควบคุม และส่วนประมวลผล ซึ่งจะมีการประมวลผลทั้งบนไมโครคอนโทรลเลอร์ และบนคอมพิวเตอร์

- ส่วนของตัวอุปกรณ์เซ็นเซอร์นั้นจะใช้ LM335ซึ่งจะมีการเปลี่ยนแปลงแรงดันทางด้านเอาท์พุตที่สัมพันธ์กับค่าของอุณหภูมิที่เปลี่ยนแปลงค่อนข้างเป็นเชิงเส้น
- ส่วนควบคุมระบบในที่นี้จะใช้ ไมโครคอนโทรลเลอร์ MCS-51 เป็น ตัวควบคุม การทำงานทั้งหมดของระบบ (ไม่รวมส่วนที่เป็นอุปกรณ์เซ็นเซอร์) ในส่วนควบคุมนี้จะรวมทั้งส่วนประมวลผลโดยไมโครคอนโทรลเลอร์ การแปลงสัญญาณอนาลอกเป็นดิจิตอลและข้อมูลที่ได้จากการสุ่มจะถูกพักไว้ที่หน่วยความจำขนาด 32 กิโลไบต์เพื่อส่งต่อไปประมวลผลยังคอมพิวเตอร์ต่อไป
- ส่วนประมวลผลบนคอมพิวเตอร์ จะเป็นการนำเอาข้อมูลที่ได้จากอุปกรณ์ดาต้าล็อกเกอร์มาทำการประมวลผล และ พล็อตกราฟความสัมพันธ์ของค่าแรงดันกับอุณหภูมิ โดยสามารถเลือกดูเส้นกราฟของแต่ละช่องได้ด้วยการป้อนข้อมูลที่คีย์บอร์ดของคอมพิวเตอร์

ซึ่งรายละเอียดของหลักการออกแบบ และ หลักการทำงานของ แต่ละส่วนนั้น จะได้กล่าว โดยละเอียดต่อไป

4.1 วงจรทางด้านภาคเซ็นเซอร์

ในการตรวจวัดอุณหภูมิเราจะใช้ LM335 เป็นตัวอุปกรณ์เซ็นเซอร์ เนื่องจากว่าง่ายต่อการ ปรับแต่ง (Calibrated) สามารถทำงานได้อย่างต่อเนื่องในช่วงอุณหภูมิ -40°C ถึง +100 °C มีอินพุตอิมพีแดนซ์ต่ำ และให้แรงดันทางขาออกที่เป็นแบบเชิงเส้น โดยมีค่าผิดพลาด 0.5% ที่อุณหภูมิห้อง (25°C) แรงดันเอาท์พุตของตัวเซ็นเซอร์สามารถคำนวณได้จาก

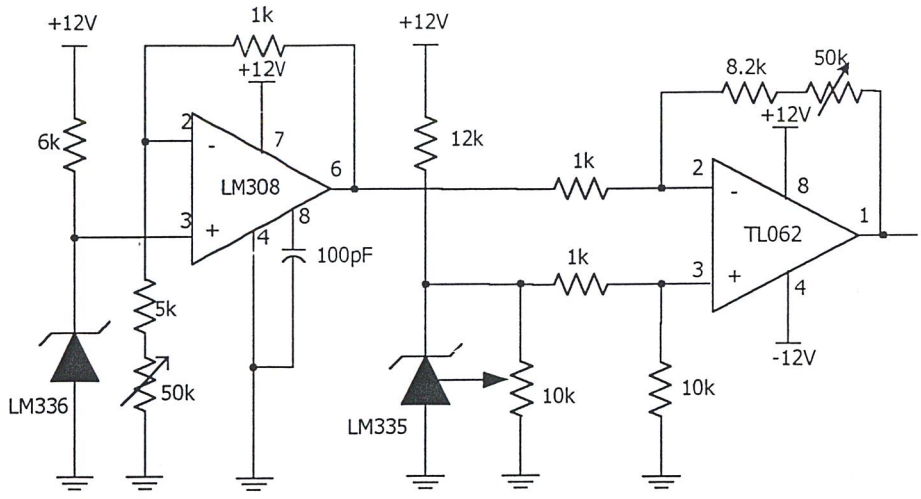
$$V_{OUT_T} = V_{OUT_{T_O}} \times \frac{T}{T_O} \qquad (4.1)$$

เมื่อ T : คืออุณหภูมิที่วัดได้ (°K)

T_O: คืออุณหภูมิอ้างอิง (298°K)

$V_{OUT_{T_O}}$ คือค่าแรงดันเอาท์พุตที่อุณหภูมิอ้างอิง (2.982V ที่ 25°C)

V_{OUT_T} คือแรงดันเอาท์พุตที่ LM335



รูปที่4-1 วงจรตรวจวัดอุณหภูมิ

จากวงจร รูปที่ 4-1 จะใช้ LM336 เป็นตัวสร้างแรงดันอ้างอิงให้กับวงจร ในที่นี่คือ 2.5 โวลท์ หลังจากนั้นก็จะไปผ่าน LM308 ซึ่งเป็นวงจรขยายที่สามารถปรับค่าอัตราการขยายเพื่อให้เราได้แรงดันเอาท์พุต (ที่ขา 6) ตามที่เราต้องการ โดยที่เอาต์พุตของ LM308 เราจะปรับค่าความต้านทาน R1 จนทำให้ได้แรงดันที่เอาต์พุตเท่ากับ 2.731 โวลท์ ที่อุณหภูมิห้อง ค่าที่ได้ นี้ จะคงที่ตลอดเวลา ไม่มีการเปลี่ยนแปลงตามค่าอุณหภูมิแต่อย่างใด และที่เอาต์พุตของตัวอุปกรณ์เซ็นเซอร์ (LM335) เราจะปรับค่า R2 เพื่อให้ได้แรงดันที่เอาต์พุตมีค่า 2.982 โวลท์ ค่าที่ได้นี้จะเป็นค่าอ้างอิงที่อุณหภูมิห้องแต่เมื่อใดที่อุณหภูมิมีการเปลี่ยนแปลงจะทำให้ค่าแรงดันเอาท์พุตที่ตัวเซ็นเซอร์มีการเปลี่ยนแปลงที่สัมพันธ์กับค่าอุณหภูมิที่เปลี่ยนไปนั้น

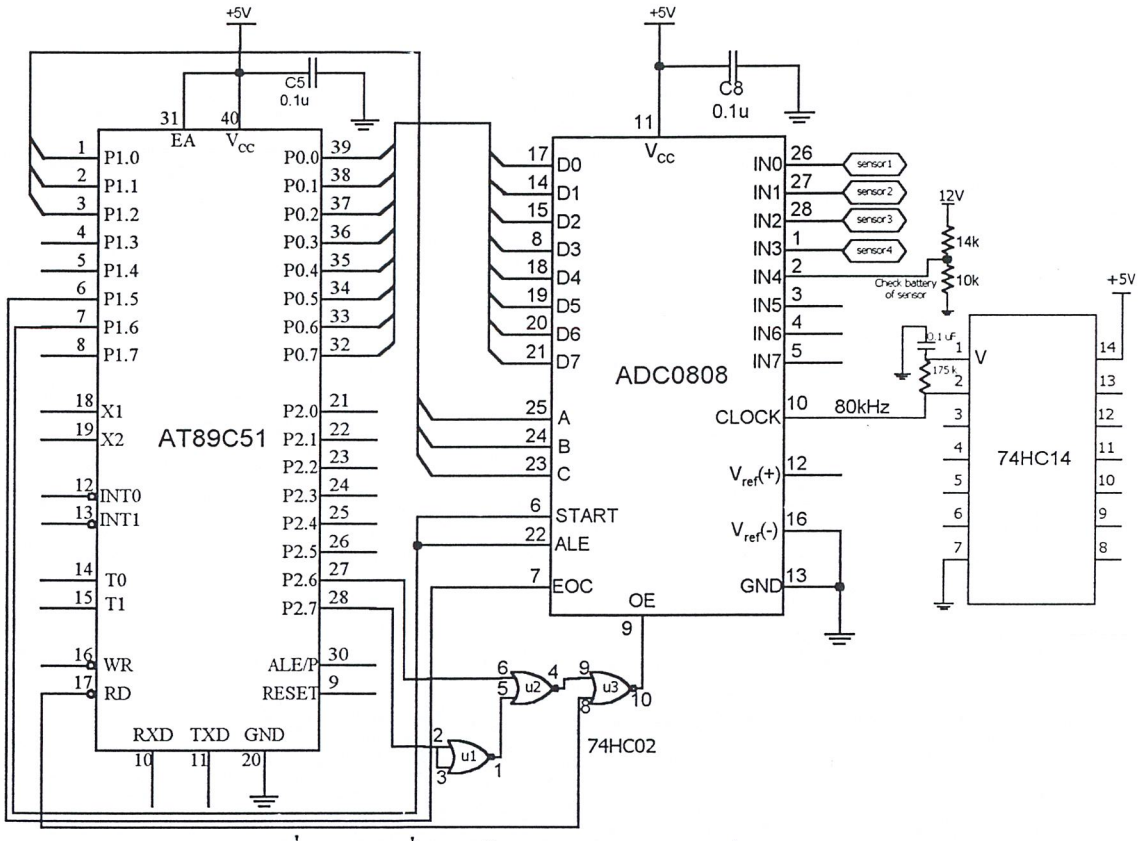
เนื่องจากแรงดันผลต่างที่ได้จากวงจรนี้มีการเปลี่ยนแปลงเท่ากับ 10 mV/$^{\circ}$C ดังนั้นจึงได้มีการนำไปผ่านวงจรขยายความแตกต่างเพื่อให้ค่าผลต่างที่ได้มีการเปลี่ยนแปลงเท่ากับ 100 mV/$^{\circ}$C เนื่องจากความสามารถอุปกรณ์แปลงสัญญาณอนาลอกเป็นดิจิตอล สามารถแปลงสัญญาณได้ตั้งแต่ 0 ถึง 5 โวลท์ และมีการเปลี่ยนแปลงเท่ากับ 20 mV/$^{\circ}$C ดังจะได้กล่าวโดยละเอียดถึงการออกแบบ และหลักการทำงานของอุปกรณ์ดังกล่าวในภาคการออกแบบส่วนควบคุมในหัวข้อต่อไป

4.2 การออกแบบภาคควบคุม

4.2 1 การเชื่อมต่อ ADC0808 กับไมโครคอนโทรลเลอร์ MCS-51

ในการแปลงสัญญาณอนาลอกเป็นดิจิตอลนั้นเราสามารถเขียนโปรแกรมเพื่อใช้งานไอซีแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลที่ใช้การเชื่อมต่อแบบ I^2C (Inter-IC Communication) เบอร์ ADC0808

- ขาข้อมูล D0-D7 จะต่อเข้ากับขาพอร์ต P0.0-P0.7 ซึ่งเป็นดาต้าบัส
- ขา Start และ ALE ต่อเข้าขาพอร์ต P1.6
- ขา EOC (End of Conversion) ต่อเข้ากับขาพอร์ต P1.5
- ขา A, B, C ต่อเข้ากับขาพอร์ต P1.0 ถึง P1.3 ตามลำดับ



รูปที่ 4-2 การเชื่อมต่อไมโครคอนโทรลเลอร์กับ ADC0808

การทำงาน

การแปลงสัญญาณอนาลอกเป็นดิจิตอลจำเป็นที่จะต้องส่งสัญญาณเริ่มต้นการทำงาน (Start) มายัง ADC ก่อน ความเร็วในการแปลงสัญญาณเท่าไรนั้นขึ้นอยู่กับชนิดของไอซีที่ใช้ เช่นเบอร์ ADC0808 จะใช้เวลาในการแปลง 100μS เมื่อแปลงเสร็จแล้วก็ จะมีสัญญาณออกมาที่ขา EOC เป็นลอจิก "1" ดังนั้นในการเขียนโปรแกรมต้องทำการเขียนพอร์ต ADC ก่อนหลังจากนั้นก็รอให้ขา EOC เป็นลอจิก "1" แล้วจึงอ่านเข้าไปเป็นอันว่าสิ้นสุดการทำงาน 1 รอบหรืออีกวิธีหนึ่ง เมื่อทำการ Start ADC แล้วให้วนลูปรอจนกว่าจะกินเวลาครบ 100μS แล้วจึงมาอ่านค่าจาก ADC ไปเก็บ

ช่วงเวลาของการแปลงสัญญาณจะเริ่มต้นขึ้นตั้งแต่สัญญาณนาฬิกาลูกแรกถูกส่งเข้าไปเตรียมระบบไปจนถึง เมื่อสภาวะของรีจิสเตอร์กลับมาพร้อมทำงาน (Ready) อีกครั้งหนึ่ง ซึ่งจะต้องใช้จำนวนสัญญาณนาฬิกาเท่ากับ n+1 เท่ากับจำนวนบิตของรีจิสเตอร์ดังนั้นถ้าวงจรแปลงสัญญาณอนาลอกเป็น

ดิจิตอลแบบประมาณค่ารวบยอด 8 บิต ที่ใช้สัญญาณนาฬิกาความถี่ 80 kHz เวลาที่ใช้ในการแปลงสัญญาณจะคำนวณได้ดังนี้

คำนวณคาบเวลาของสัญญาณนาฬิกา

$F_{clk} = 80kHz$

$$T = \frac{1}{F_{clk}} = T_1 + T_2 \cong \frac{1}{80 \times 10^3} \cong 12.5\mu S$$

$$T_1 = (R_a + R_b)C \times \log\left(\frac{V_{cc} - \frac{2}{3}V_{cc}}{V_{cc} - \frac{1}{3}V_{cc}}\right)$$

$$T_2 = R_aC \times \log\left(\frac{V_{cc} - \frac{2}{3}V_{cc}}{V_{cc} - \frac{1}{3}V_{cc}}\right)$$

จำนวนสัญญาณนาฬิกาทั้งหมดที่ใช้ในการแปลงเท่ากับ

$n+1 = 8+1 = 9$

เวลาทั้งหมดที่ใช้เท่ากับ

$9 \times 12.5\mu s = 112.5\mu s$

ความเที่ยงตรงของวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล เป็นการเปรียบเทียบแรงดันอนาลอกของวงจรแปลงสัญญาณอนาลอก กับแรงดันที่ควรจะเกิดขึ้นจริงเช่นข้อมูลดิจิตอลสูงสุดของวงจร ADC ขนาด 8 บิตเมื่อเทียบกับแรงดันอนาลอกควรมีขนาดเท่ากับ 5.0000 โวลท์แต่จากการคำนวณจะได้ค่าแรงดัน 4.9804 โวลท์ นั่นคือความผิดพลาดที่เกิดขึ้น ดังนั้นในวงจร ADC ขนาด 8 บิต จะมีความผิดพลาดเป็น $\pm \frac{1}{2}$ LSB

ค่าเวลาในการแปลงสัญญาณเป็นค่าเวลาทั้งหมดที่วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลใช้ ในการแปลงสัญญาณจรเสร็จสิ้น พารามิเตอร์นี้มักจะปรากฏในคุณสมบัติของไอซีที่ทำงานเป็นวงจร ADC เมื่อไอซีทำการแปลงสัญญาณเสร็จสิ้นลง จะส่งสัญญาณที่เรียกว่า EOC ออกมา

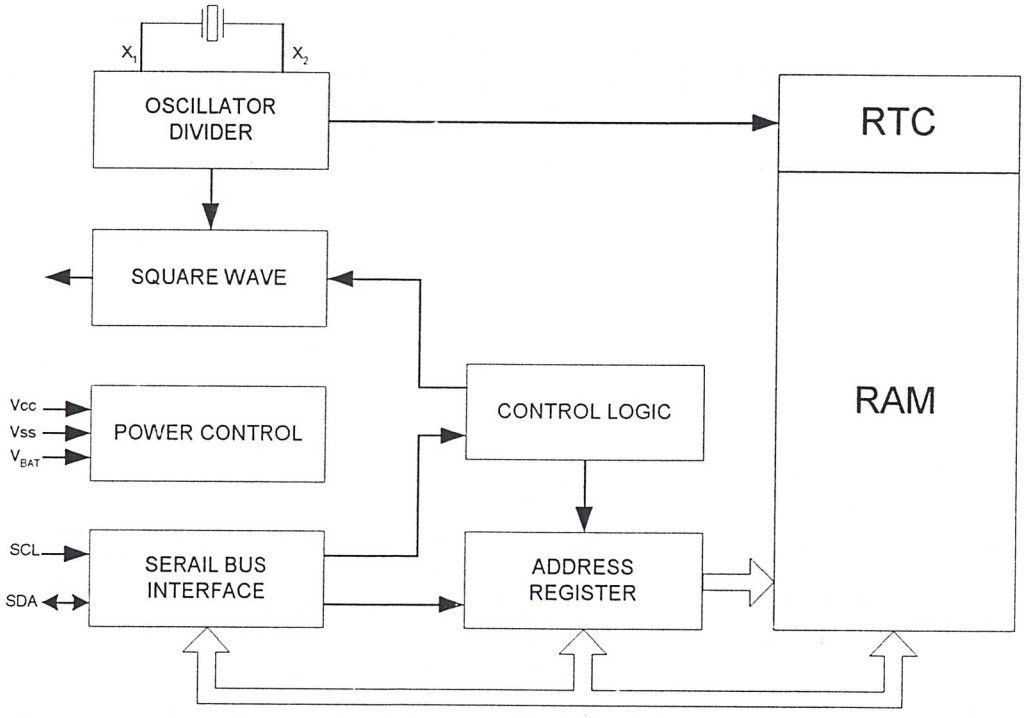
4.2.2 การสร้างฐานเวลาให้กับไมโครคอนโทรลเลอร์

ในการสร้างฐานเวลาให้กับไมโครคอนโทรลเลอร์จะใช้ ไอซี DS1307 โดยจะให้ข้อมูลเกี่ยวกับเวลาจริงทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลัก วินาที นาที ชั่วโมง วันที่ วันในสัปดาห์ เดือน และ ปี ด้วยคุณสมบัติทางเทคนิคที่สำคัญดังนี้

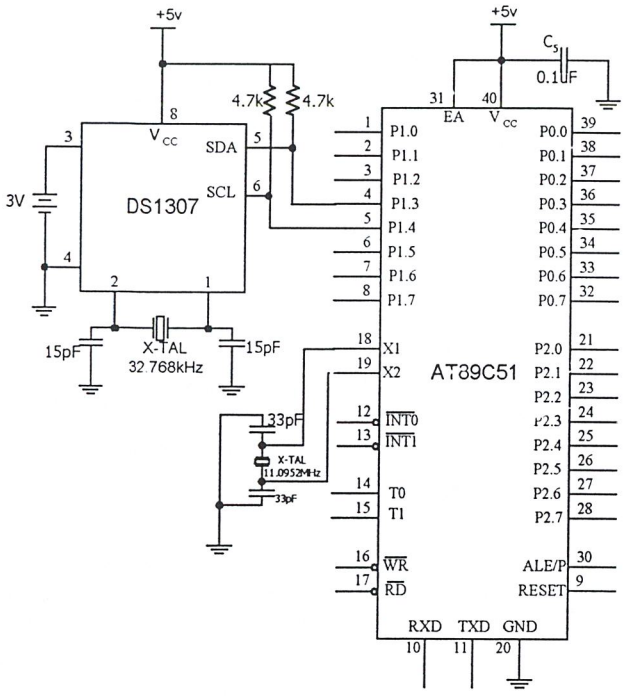
- เป็นไอซีรีลไทม์คล็อกให้ข้อมูลตั้งแต่วินาทีจนถึงปี
- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายใน สามารถใช้เก็บข้อมูลทั่วไปได้

- ใช้การเชื่อมต่อแบบระบบ I^2C
- มีวงจรตรวจจับไฟเลี้ยง ต่ำ หรือ ขาดหาย ไปอย่าง อัตโนมัติ และ สามารถรักษา ข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี

ขา 1 และ 2 ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768 kHz เพื่อใช้เป็นฐานเวลาในการสร้าง ค่าเวลาจริงใน การใช้งาน ต้องต่อคริสตอลเข้า กับ ขาทั้งสองนี้และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15 pF คร่อมกับขากราวด์ด้วย



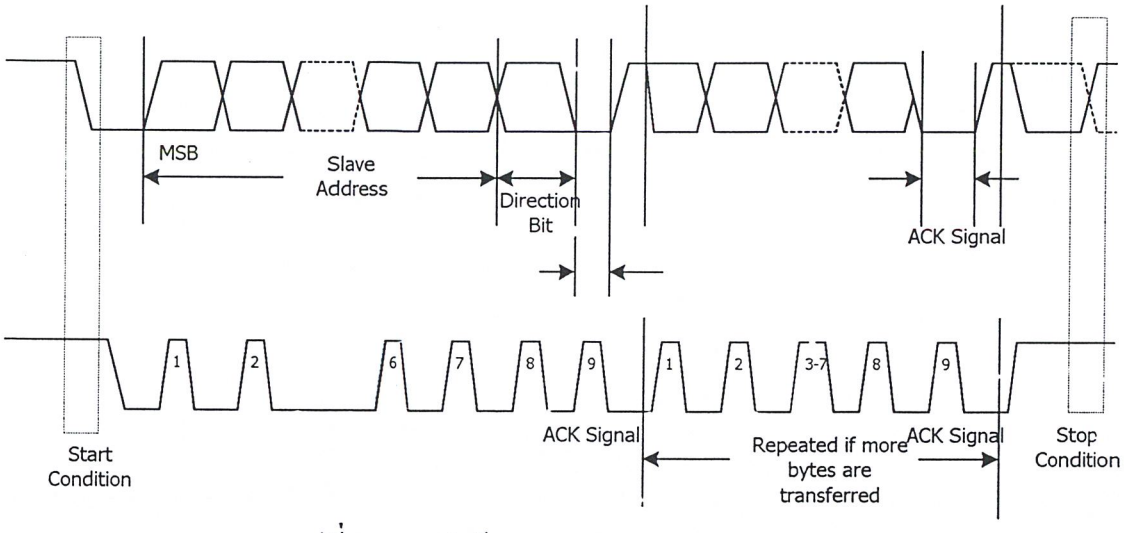
รูปที่4-3 แสดงถึงส่วนประกอบหลักที่สำคัญของ DS1307



รูปที่ 4-4 การเชื่อมต่อไมโครคอนโทรลเลอร์กับไอซีรีลไทม์คล็อก

การทำงานของ DS1307

วงจรออสซิลเลทเตอร์ ถือเป็นหัวใจหลักของไอซีกำหนดสัญญาณนาฬิกาจริง (Real time clock) เนื่องจากเป็น จุดเริ่มต้นของ การสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงานที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลาในกรณีที่มีการอีเนเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิส เตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่าคือ 1 Hz , 4.096kHz, 8.192kHz และ 32kHz โดยการกำหนดค่า RS1 และ RS0 ดังแสดงในตาราง 4-1 พร้อมกันนั้นจะมีการเก็บค่าของเวลาไว้ ในหน่วยความจำซึ่งมีขนาด 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปอีก 56 ไบต์



รูปที่ 4-5 การส่งข้อมูลบน Serial Bus ของ DS1308

รีจิสเตอร์ควบคุม (Control Register) ที่ใช้สำหรับควบคุมการทำงานของขา SQW/OUT

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
OUT	X	X	SQWE	X	X	RS1	RS0

- OUT (Output control): บิตนี้ใช้ควบคุมระดับเอาท์พุตของขา SQW/OUT ในกรณีที่มีการดิสเอเบิลการกำเนิดสัญญาณพัลส์ ถ้า SQWE = 0 ระดับสัญญาณลอจิกที่ขา SQW/OUT จะเท่ากับ"0" และจะเท่ากับ "1" เมื่อขา SQWE เท่ากับ "1"
- SQWE (Square Wave Enable): บิตนี้เมื่อเซ็ตค่าลอจิกเป็น "1" จะเป็นการ Enable เอาท์พุตออสซิลเลทเตอร์ ความถี่ของสัญญาณพัลส์จะขึ้นอยู่กับค่าของบิต RS0 และ RS1
- RS (Rate Select): บิตเหล่านี้ใช้ในการควบคุมความถี่เอาท์พุตพัลส์ ตารางที่4-1 แสดงถึงรายละเอียดของการเลือกความถี่คลื่นพัลส์โดยบิต RS ทั้งสองตัว
- SDA และ SCL (ขา 5 และ 6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบ I^2C ซึ่งในที่นี้จะต่อกับ P1.3 และ P1.4 ตามลำดับ

ตารางที่ 4-1 การเลือกความถี่ของสัญญาณพัลส์

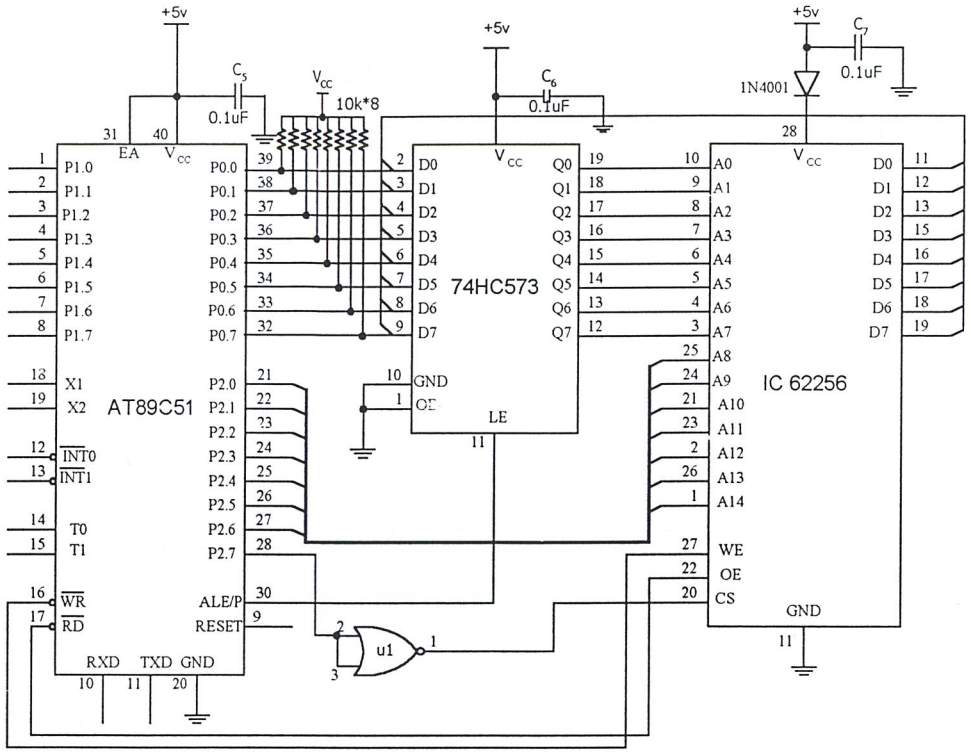
RS1	RS0	SQW/OUT Frequency
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

ในการส่งข้อมูลทุกๆครั้งจะเริ่มต้นด้วยสภาวะเริ่มต้น (Start Condition) และสิ้นสุดการส่งข้อมูลด้วยสภาวะหยุด (Stop Condition) จำนวนไบต์ของข้อมูลที่ทำการส่งในระหว่างสภาวะเริ่มต้นจนถึงสภาวะหยุดการส่งนั้นจะไม่มีขีดจำกัด แต่จะถูกกำหนดโดยอุปกรณ์ควบคุม ในการส่งไบต์ข้อมูลนั้นอยู่ที่อุปกรณ์รับจะมีการส่งสัญญาณเพื่อตอบรับการส่งข้อมูล (Acknowledge Signal) ไปที่ตัวส่งเพื่อบอกให้รู้ว่าสามารถส่งข้อมูลได้หรือข้อมูลที่ส่งไปได้รับเรียบร้อยแล้วโดยอุปกรณ์ตัวรับจะต้องส่งสัญญาณดังกล่าวไปพร้อมกับบิตที่ 9 หลังจากที่มีการรับข้อมูลทุกครั้ง

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสถานะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า $1.25V_{BAT}$ ก็จะควบคุมให้ DS1307 หยุดการทำงานดังนั้นในการใช้งานต้องระมัด ระวังอย่าให้ไฟเลี้ยง ต่ำกว่า $1.25V_{BAT}$ หรือประมาณ 3.75 โวลท์ ในกรณีที่ V_{BAT} เท่ากับ 3โวลท์ หากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซีจะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยง ปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่า ของเวลาที่เป็นจริงต่อไป

4.2.3 การเชื่อมต่อไมโครคอนโทรลเลอร์ กับ หน่วย ความจำภายนอก

Static RAM เป็นหน่วยความจำชนิดที่ง่ายต่อการใช้งานโดยไอซีที่นำมาต่อเป็นหน่วยความจำภายนอกได้แก่ไอซีเบอร์ 62256 โดยมีหน่วยความจำ 32 กิโลไบต์ ในการคิดค่าขนาดของหน่วยความจำนั้นสามารถคำนวณได้จากขาแอดเดรสของแรม จะเห็นว่าแรมเบอร์ 62256 มีขาแอดเดรสคือ A0-A14 หรือ 15 บิต ขนาดหน่วยความจำก็จะเท่ากับ 2^{15} ซึ่งเท่ากับ 32 กิโลไบต์



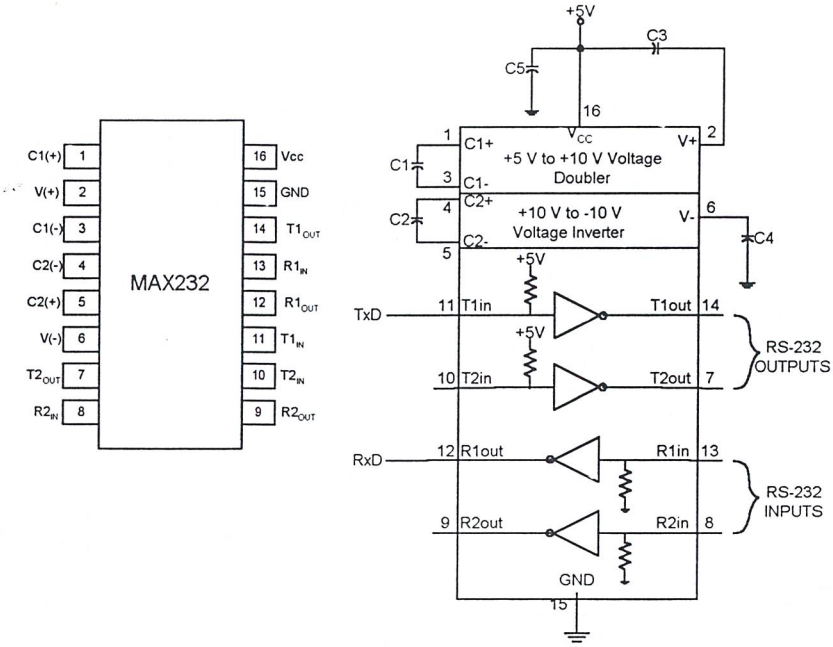
รูปที่ 4-6 วงจรใช้งานแรมภายนอก

ในการใช้งานหน่วยความจำภายนอกโดยจากวงจรในรูปที่ 4-6 จะใช้พอร์ต 0 ของไมโครคอนโทรลเลอร์ในการอ้างแอดเดรสไบต์ต่ำ และใช้พอร์ต 2 ในการอ้างแอดเดรสไบต์สูงให้กับแรมภายนอก โดยจะเห็นว่าพอร์ต 0 ต่อเข้ากับพอร์ต Data Bus ของแรมภายนอกซึ่งพอร์ต 0 นั้นไม่สามารถที่จะใช้งานในการอ้างแอดเดรสไบต์ต่ำ และ อ่านเขียนข้อมูลไปยังแรมได้พร้อมกัน ด้วยเหตุนี้ จึงได้มีการต่อไอซี 74HC573ร่วมเข้าไปเพื่อช่วยในการแลตช์แอดเดรสไบต์ต่ำให้กับแรมภายนอกโดยขาALE ของไมโครคอนโทรลเลอร์จะเป็นตัวควบคุมจังหวะการแลตช์ให้กับไอซี 74HC573 และในขณะที่ไอซี 74HC573 แลตช์แอดเดรสไบต์ต่ำอยู่นั้น พอร์ต 0 จะทำการอ่านเขียนข้อมูลกับแรมภายนอกได้นั้นเอง จากวงจรจะเห็นว่าแรมภายนอกนั้นจะมีสัญญาณควบคุมในการอ่านเขียนข้อมูล โดยใช้ขาสัญญาณ RD และ WR ไมโครคอนโทรลเลอร์เป็นตัวควบคุมในพอร์ต2นั้นไม่มีปัญหาแต่อย่างใดเพราะใช้ใน การอ้างแอดเดรสไบต์สูงเพียงหน้าที่เดียวเท่านั้น

4.2.3 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานจงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้การติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมารตฐาน RS-232 เป็นส่วนใหญ่แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ± 3 V ถึง ± 12 V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์จะอยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อกันโดยตรงได้ต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ โครงงานนี้เราจะใช้ไอซี MAX232

ซึ่งทำหน้าที่ในการแปลงระดับสัญญาณข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอล (TTL) ไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ได้ รูปที่ 4-6 แสดงให้เห็นถึงการจัดวางตำแหน่งขาสัญญาณและโครงสร้างภายในของไอซี MAX232 ที่จะใช้เชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์



รูปที่ 4-7 การจัดขา และโครงสร้างภายในของ MAX232 สำหรับต่อกับพอร์ตของคอมพิวเตอร์และไมโครคอนโทรลเลอร์

4.2.4 การเชื่อมต่อกับผู้ใช้งาน

ในโครงงานนี้ใช้แผง LCD แสดงผลเมื่อมีการ

4.2.4.1 การเชื่อมต่อ LCD กับไมโครคอนโทรลเลอร์

โมดูล LCD จะมีส่วนประกอบหลักๆ 3 ส่วนดังนี้

ตัวแสดง (Display) ผลภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

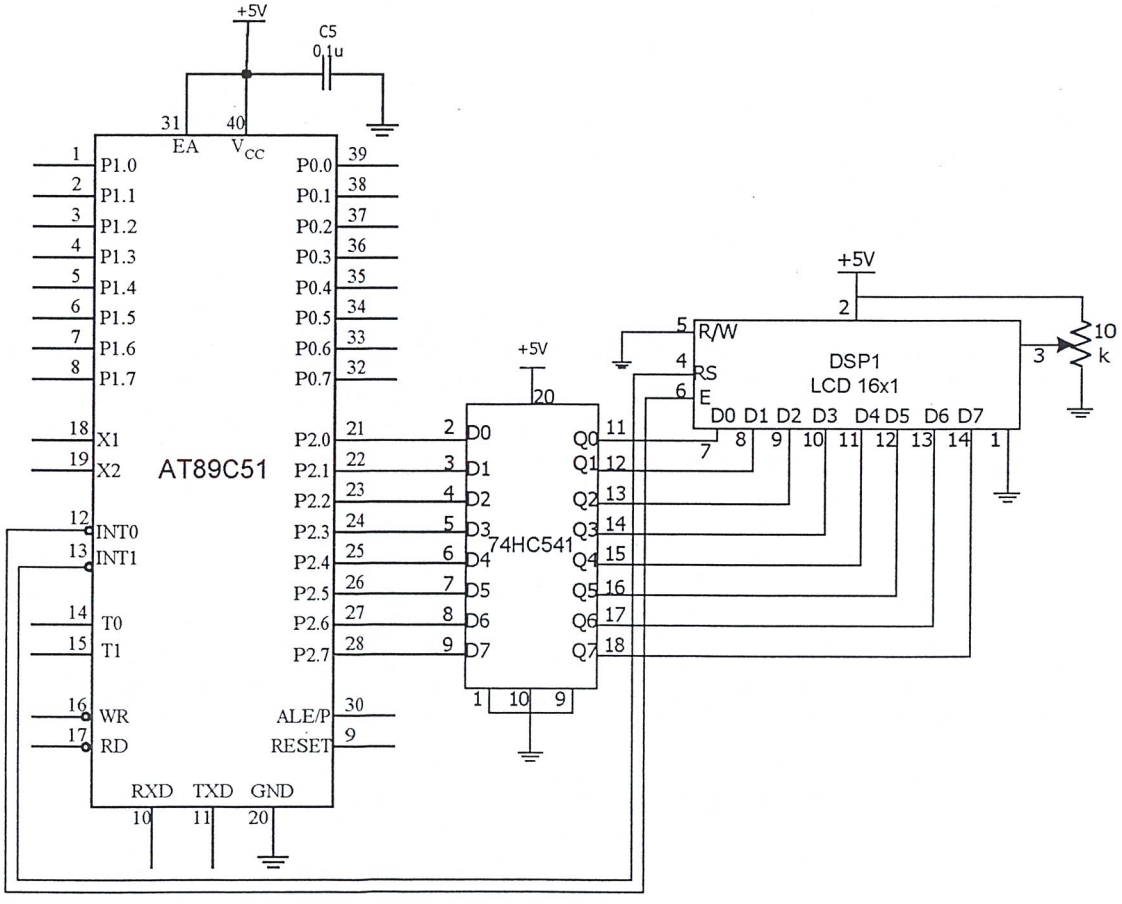
ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์เป็นต้น

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด

ในการเชื่อมต่อโมดูลแสดงผลแบบผลึกเหลว (LCD) กับไมโครคอนโทรลเลอร์นั้นเราจะใช้ไอซีเบอร์ 74HC541 ทำหน้าที่เป็นไอซีบัฟเฟอร์เพื่อเพิ่มกระแสจากพอร์ตของไมโครคอนโทรลเลอร์ที่จะไปขับ LCD นั้นมีค่าสูงขึ้น เนื่องจากว่ากระแสที่ขาพอร์ตของไมโครคอนโทรลเลอร์นั้นมีค่าต่ำ ซึ่งไม่สามารถที่จะขับโมดูล LCD ให้สว่างได้นั่นเอง.

โมดูล LCD ขนาด 16x1 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังรูปที่4-8 รายละเอียดการทำงานของแต่ละขามีดังนี้

- V_{SS} (ขา 1) ต่อกราวด์
- V_{DD}(ขา 2) ต่อไฟเลี้ยง +5 โวลท์
- V_O (ขา 3) เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล
- RS (ขา 4) เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR (Intruction Registe) หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR (Data Register) โดยถ้าขานี้เป็น " 0 " ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น " 1 " ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล
- R/W (ขา 5) เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น " 0 " เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น " 1 " จะเป็นการอ่านข้อมูล
- E (ขา 6) เป็นขาอีเนเบิล LCD ให้ทำงาน
- D0-D7 (ขา 7-14) เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต



รูปที่ 4-8 การเชื่อมต่อโมดูล LCD กับไมโครคอนโทรลเลอร์

4.2.5 สวิทช์ส่งงานคุมระบบ

สวิทช์ที่ออกแบบไว้ในระบบดาต้าล็อกเกอร์มีทั้งหมด 3 ตัวด้วยกันซึ่งแต่ละตัวจะทำหน้าที่หลายอย่างด้วยกัน (ภาคผนวก) ได้แก่

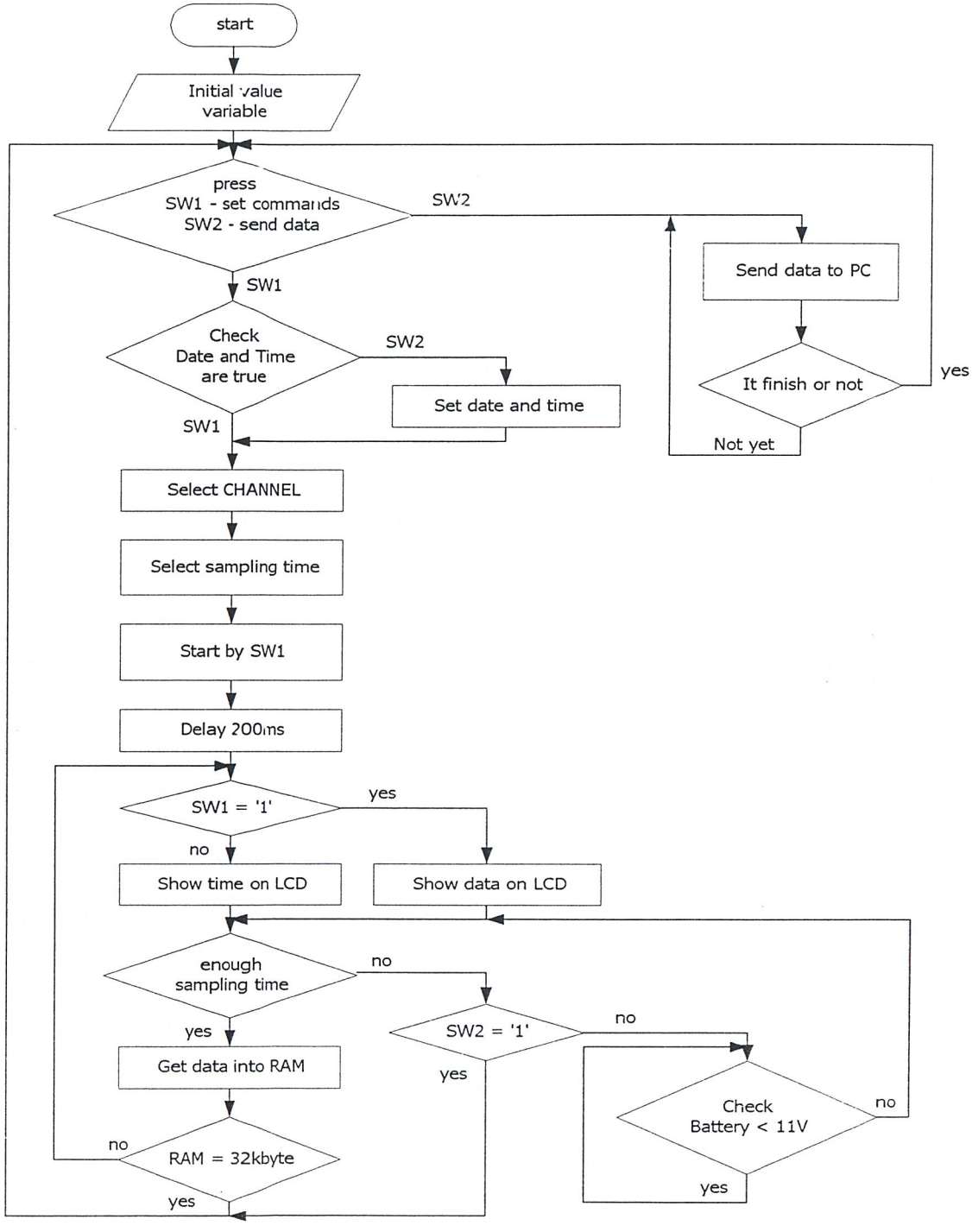
- Switch 1 (SW1)
- Switch 2 (SW2)
- Switch 3 (SW3)

4.3 ซอฟต์แวร์

ซอฟต์แวร์ของระบบประกอบด้วยสองส่วน: ส่วนแรกใช้ ในการโปรแกรม การทำงานของฮาร์ดแวร์และเชื่อมต่อกับไมโครคอมพิวเตอร์และกับผู้ใช้ ในส่วนนี้พัฒนาบน ET – AFP V1.0 ในส่วนที่สองเป็นซอฟต์แวร์บนไมโครคอมพิวเตอร์ ใช้ในการรับข้อมูลเพื่อนำมาประมวลและแสดงผล ในส่วนนี้เขียนด้วยภาษา C (Turbo C^{TM})

4.3.1 การไหลของโปรแกรมควบคุมระบบ

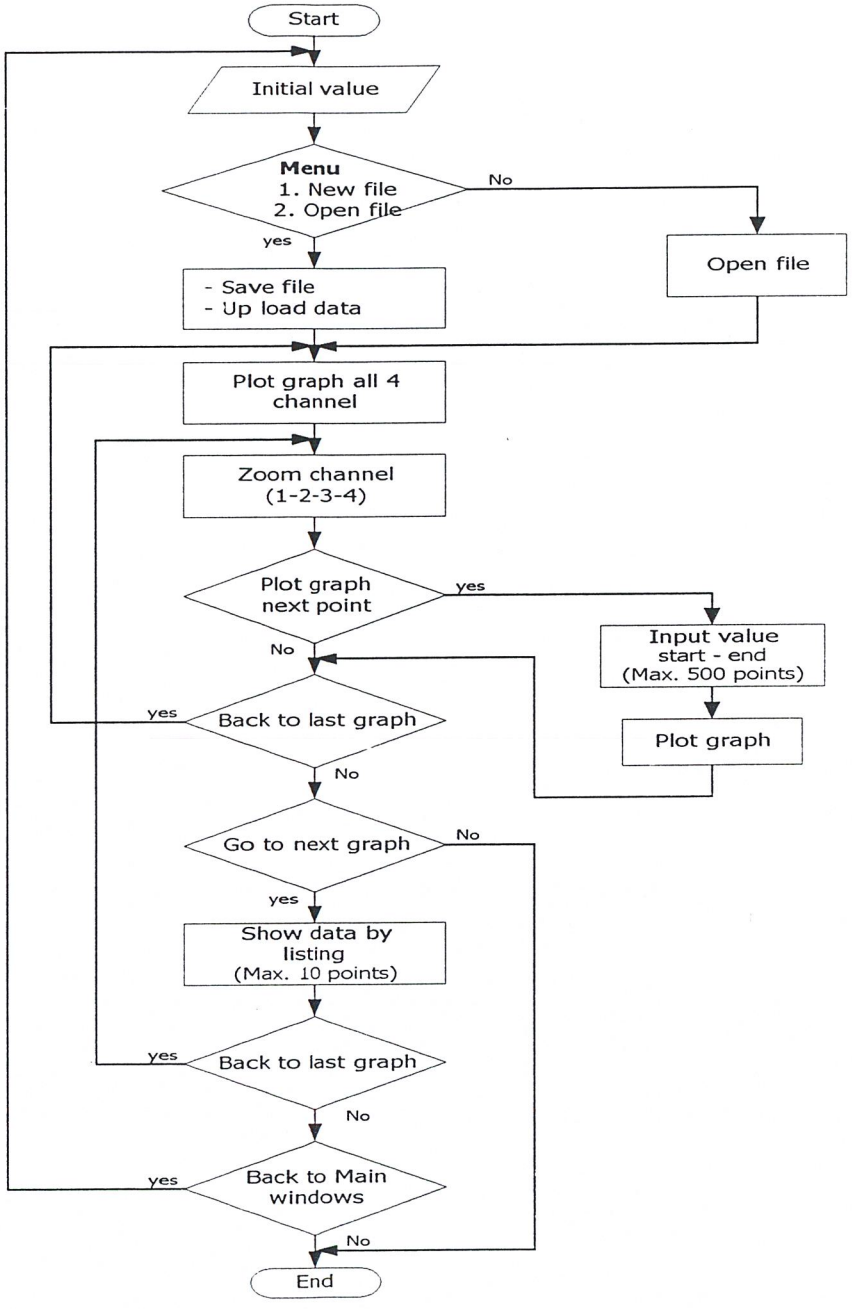
การทำงานแสดงเป็นโฟลวชาร์ตดังรูปที่4-9



รูปที่ 4-9 การไหลของโปรแกรมควบคุมระบบ

4.3.2 การไหลของโปรแกรมบนไมโครคอมพิวเตอร์

การทำงานแสดงเป็นโฟลวชาร์ตดังรูปที่ 4-10

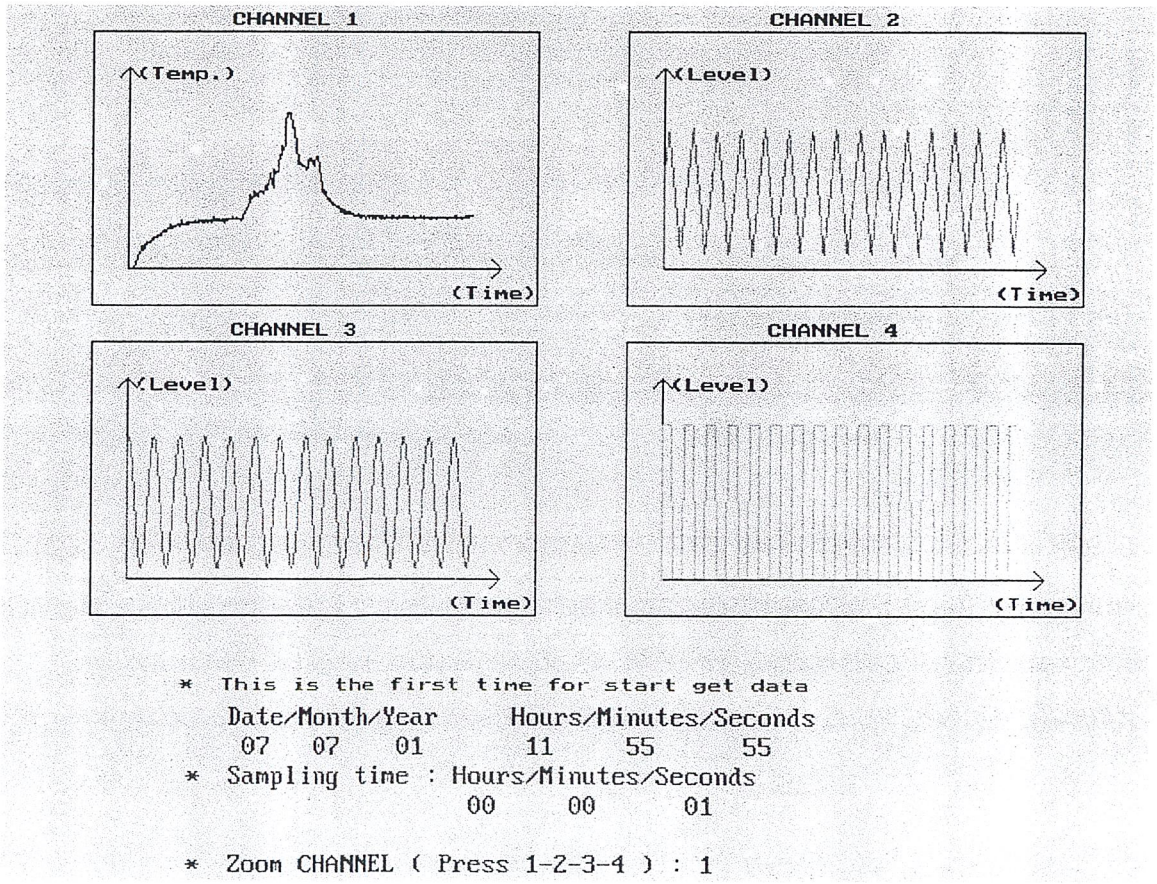


รูปที่ 4-10 การไหลของโปรแกรมบนไมโครคอมพิวเตอร์

4.3.3 การออกแบบภาคแสดงผลและการออกแบบหน้าจอ

ในภาคนี้เราจะทำการเขียนโปรแกรมเพื่อลิงค์กับคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม RS-232 รายละเอียดของตัวโปรแกรมนั้นจะอยู่ในภาคผนวก (ข)

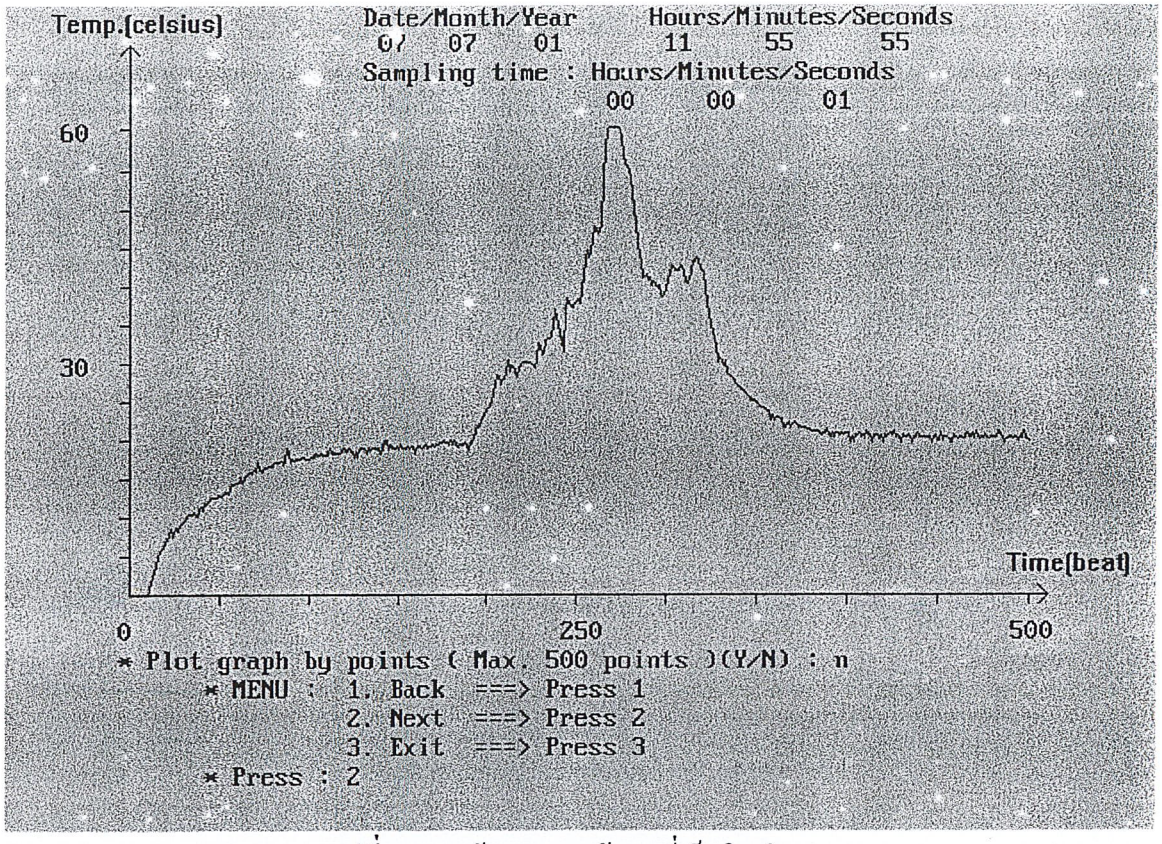
เพื่อนำเอาข้อมูลที่ได้จากการสุ่มซึ่งเก็บไว้ ในแรม ไปทำการประมวลผล และ แสดงบนจอคอมพิวเตอร์นั้นสามารถทำได้โดยการเขียนโปรแกรมเพื่อทำการอ่านค่าจากหน่วยความจำไปยังคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม RS-232 ซึ่งในการแสดงผลบนจอนั้นก็ยังสามารถเลือกได้ว่าจะให้แสดงข้อมูลของช่องไหนโดยการกดที่คีย์บอร์ดของคอมพิวเตอร์ ข้อมูลที่ได้จากดาต้าล็อกเกอร์นี้จะถูกนำไปพล็อตกราฟแล้วแสดงผลที่จอคอมพิวเตอร์ ในการพล็อตกราฟ นี้เราก็สามารถเลือกได้ว่าจะให้แสดงทีละช่องหรือว่าทั้ง 4 ช่องพร้อมกันก็ได้ที่หน้าจอคอมพิวเตอร์จะปรากฏเป็นหน้าต่างดังรูปที่ 4-10 ซึ่งจะแสดงให้เห็นข้อมูลของทั้ง 4 ช่องพร้อมกันจากนั้นเราก็สามารถเลือกดูได้ว่าจะดูข้อมูลที่ช่องไหนโดยการ กดคีย์บอร์ด โดยช่องที่หนึ่งจะแสดงผลข้อมูลเป็นค่าอุณหภูมิ และอีก 3 ช่องจะแสดงข้อมูลเป็นแบบระดับคือจะแสดงตั้งแต่ 0 ถึง 255 ระดับ มีไว้สำหรับการใช้งานทั่วไป



รูปที่ 4-11 หน้าจอแสดงผล

ที่หน้าจอแสดงผลจะบอกเวลาที่เราเริ่มเก็บข้อมูล ค่า Sampling time และจำนวนช่องที่เราใช้เก็บข้อมูล

สมมุติว่าเราต้องการดูข้อมูลที่เก็บได้จากช่อง 1 เราก็กดเลข 1 บนคีย์บอร์ดของคอมพิวเตอร์จากนั้นที่หน้าจอจะปรากฏข้อมูลเฉพาะที่ช่อง 1 ดังรูปที่ 4-12



รูปที่ 4-12 หน้าจอแสดงข้อมูลที่เก็บในช่อง 1

4.4 การคำนวณระยะเวลาของการเก็บข้อมูล

เนื่องจากว่าเวลาของการสุ่มเก็บข้อมูลของระบบเรานั้นสามารถที่จะเลือก Sampling time ได้ตามที่เราต้องการโดยการกดสวิทช์ที่บอร์ตของดาต้าล็อกเกอร์ดังนั้นระยะเวลาที่หน่วยความจำในตัวดาต้า ล็อกเกอร์ จะสามารถเก็บข้อมูลได้ในแต่ละช่องเมื่อเราเลือก Sampling time ค่าต่างๆดังนี้

$$\text{Duration} = \frac{32768}{\text{Number of Channel}} \times \text{Sampling time}$$

กำหนดให้

Duration : ระยะเวลาที่ดาต้าล็อกเกอร์สามารถเก็บข้อมูลได้

Sampling time : ช่วงระยะเวลาที่จะทำการสุ่มเก็บข้อมูล

Number of Channel :จำนวนช่องที่ ต้อง การเก็บ

ตัวอย่างการคำนวณระยะเวลาที่หน่วยความจำจะสามารถเก็บข้อมูลได้ เมื่อใช้ Sampling time ค่าต่างๆ โดยใช้สมการข้างบนเราจะคำนวณได้ดังนี้

4.4.1 เก็บข้อมูลทุกๆ 1 วินาที

ถ้าเราให้ดาต้าล็อกเกอร์เก็บข้อมูลทุกๆ 1 วินาที จะได้ระยะเวลาของการสุ่มเก็บดังนี้

- เก็บ 1 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{1} \times 1\,\text{Sec} = 32768\,\text{Sec} = 9\,\text{Hours}\,6\,\text{Min}$$

- เก็บ 2 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{2} \times 1\,\text{Sec} = 16394\,\text{Sec} = 4\,\text{Hours}\,33\,\text{Min}$$

- เก็บทั้ง 3 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{3} \times 1\,\text{Sec} = 10922\,\text{Sec} = 3\,\text{Hours}\,2\,\text{Min}$$

- เก็บทั้ง 4 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{4} \times 1\,\text{Sec} = 80692\,\text{Sec} = 2\,\text{Hours}\,16\,\text{Min}$$

4.4.2 เก็บข้อมูลทุกๆ 1 นาที

- เก็บ 1 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{1} \times 60\,\text{Sec} = 1966080\,\text{Sec} = 546\,\text{Hours}\,8\,\text{Min}$$

- เก็บ 2 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{2} \times 60\,\text{Sec} = 983040\,\text{Sec} = 273\,\text{Hours}\,4\,\text{Min}$$

- เก็บ 3 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{3} \times 60\,\text{Sec} = 819200\,\text{Sec} = 182\,\text{Hours}\,2\,\text{Min}$$

- เก็บทั้ง 4 ช่องจะได้เท่ากับ

$$\text{Duration} = \frac{32768}{4} \times 60\,\text{Sec} = 491520\,\text{Sec} = 136\,\text{Hours}\,32\,\text{Min}$$

ในการคำนวณระยะเวลาที่ค่า Sampling time อื่นๆ นั้นก็ใช้หลักในการคำนวณเหมือนกับ การคำนวณด้านบน ซึ่งจะสรุปได้ดังตารางที่ 4-2 แสดงถึงระยะเวลาที่สามารถเก็บข้อมูลได้เมื่อใช้ Sampling time ค่าต่างๆ

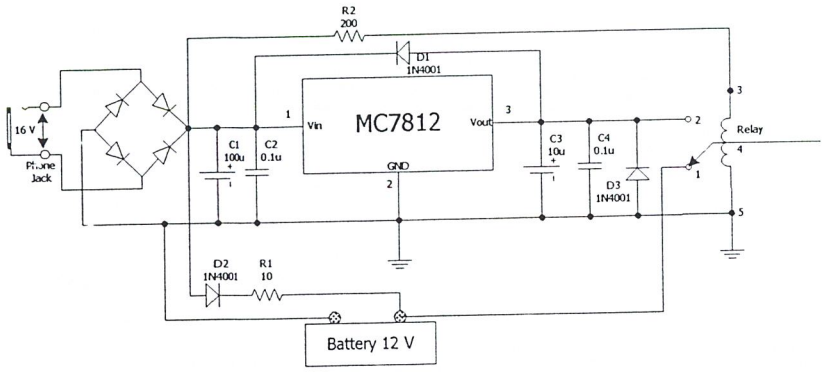
ตารางที่ 4-2 ความสามารถในการเก็บข้อมูลของหน่วยความจำ

Sampling time	ระยะเวลาที่สามารถเก็บข้อมูลในหน่วยความจำ (ชั่วโมง : นาที : วินาที)			
	1 ช่อง	2 ช่อง	3 ช่อง	4 ช่อง
1 วินาที	09 : 06 : 08	04 : 33 : 04	03 : 02 : 02	02 : 16 : 32
5 วินาที	45 : 30 :40	22 : 45 : 20	15 : 10 : 13	11 : 22 : 40
10 วินาที	91 : 01 : 40	45 : 30 : 40	30 : 20 : 26	22 : 45 : 20
30 วินาที	273 : 04 : 00	136 : 32 : 00	91 : 01 : 20	68 : 16 : 00
1 นาที	546 : 08 : 00	273 : 04 : 00	182 : 02 : 40	136 : 32 : 00
10 นาที	5461 : 20 : 00	2730 : 40 : 00	1820 : 26 : 40	1365 : 20 : 00
30 นาที	16384 : 00 : 00	8192 : 00 : 00	5461 : 20 : 00	4096 : 00 : 00
1 ชั่วโมง	32768 : 00 : 00	16384 : 00 : 00	10922 : 40 : 00	8192 : 00 : 00

4.5 แหล่งจ่ายไฟของระบบ (Power Supply)

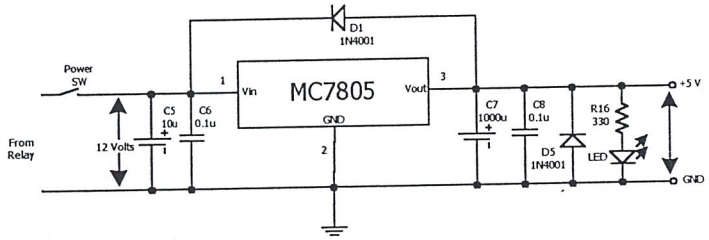
เนื่องจากว่าในระบบของเรานั้นต้องการแหล่งจ่ายแรงดันที่แตกต่างกันเช่น ส่วนที่เป็นวงจรเซ็นเซอร์จะต้องการแรงดันทั้งบวกและลบ ($\pm$ 12 โวลท์) และดาต้าล็อกเกอร์ที่ต้องการแรงดัน +5 โวลท์ ดังนั้นในส่วนนี้จะเป็นการออกแบบแหล่งจ่ายไฟเลี้ยงที่จะให้กับระบบ เพื่อที่จะทำให้ระบบของเราสามารถใช้ได้ทั้งไฟบ้าน คือใช้หม้อแปลง 16 โวลท์ ผ่านวงจรเรกกูเลเตอร์ เพื่อให้ได้แรงดัน 12 โวลท์ และใช้ได้กับแบตเตอร์รี่ 12 โวลท์ โดยในขณะที่เราใช้ไฟโดยผ่านหม้อแปลงนั้น ระบบก็จะทำการชาร์จแบตเตอร์รี่ไปในตัวด้วยและเมื่อไม่มีไฟเลี้ยงอินพุตของวงจรเรกกูเลเตอร์ สวิชท์รีเลย์ก็จะเปลี่ยนมาใช้ไฟจากแบตเตอร์รี่แทน โดยในส่วนของการออกแบบภาคไฟเลี้ยงจะมีอยู่ด้วยกัน 3 ส่วนคือ

- แรงดันที่ได้จากหม้อแปลง 16 โวลท์จะผ่านวงจรเรกกูเลเตอร์ เพื่อให้ได้ แรงดันเอาท์พุต ที่เราต้องการคือ 12 โวลท์ รูปที่ 4-13 โดยเราจะใช้ไอซีเรกกูเลเตอร์เบอร์ MC7812 และแรงดันที่ได้จากหม้อแปลงนี้ก็จะถูกนำไปใช้ในการชาร์จแบตเตอร์รี่ด้วย ซึ่งกระแสที่ใช้ในการชาร์จนั้นประมาณ 0.5 แอมป์



รูปที่ 4-13 วงจรเรกกูเรเตอร์ 12 โวลท์

วงจรเรกกูเลเตอร์อีกตัวหนึ่งจะเป็นส่วนที่จะทำให้เราได้ไฟ +5 โวลท์ที่เพื่อเป็นแหล่งจ่ายให้กับระบบ ดังรูปที่ 4-14 ในส่วนนี้เราใช้ไอซีเรกกูเรเตอร์เบอร์ MC 7805 ซึ่งจะให้แรงดันทางเอาท์พุต +5 โวลท์



รูปที่ 4-14 วงจรเรกกูเรเตอร์ 5 โวลท์

- DC to DC Converter หน้าที่หลักของวงจร DC/DC Converter เป็นวงจรที่สร้างแหล่งจ่ายไฟลบให้กับ Op_ampทั้งหมดในระบบ

โดยหลักการออกแบบ DC to DC Converter นั้นเป็นการนำเอาวงจร อินเวอร์เตอร์แบบ Indirect มาใช้ ในรูปที่ 4-15 โดยที่ขาเบสของทรานซิสเตอร์นั้นจะเป็นสัญญาณรูปสี่เหลี่ยม ซึ่งเราได้ จากขาที่ 8 และ 11 ของไอซีเบอร์ TL 494 ด้วยเหตุนี้จะทำให้ทรานซิสเตอร์ มีการสวิทช์ และเมื่อทรานซิสเตอร์ "ON" กระแสจากขา Collector จะทำให้ขดลวดที่ 1 มีการนำกระแส และในขณะเดียวกันนี้ไดโอดถูกป้อนแรงดันย้อนกลับ แต่เมื่อทรานซิสเตอร์ "OFF" แรงดันที่ตกคร่อมขดลวดจะกลับขั้วหลังจากนั้นก็จะจ่ายให้กับโหลดในขณะนี้ไดโอดจะถูกป้อนแรงดันตรง ฉะนั้นจุดที่สำคัญของวงจรนี้คือการออกแบบค่าขดลวดที่ 1 และคาปาซีเตอร์ 1 ให้เหมาะสมเพราะว่าทั้งสองตัวนี้คือตัวที่กำหนด Time Constant ของวงจร ดังนั้น เพื่อ ให้ L และ C ชุดนี้มีการตอบสนองต่อความถี่ของสวิทช์ได้ที่เราสามารถคำนวณค่า L และ C ได้ดังนี้

ให้

$$|V_O| \approx 12\,V$$

$$|I_O| \approx 0.25\,A$$

$$Eff \approx 60$$

$$|I_O| = \frac{0.25A}{0.6} = 0.42A$$

จากสมการ $$T_{on(MAX)} = \frac{0.8V_O T}{N_{IN(MIN)} + V_O}$$

ให้ $$f = 25kHz; T = \frac{1}{25kHz} = 40\mu S$$

ให้ $$V_{IN(MIN)} = 8V$$

ได้ $$T_{on(MIN)} = \frac{0.8 \times 12 \times 40\mu s}{8+12} = 19.2\mu s$$

จากสมการ

$$V_O = V_{in(MIN)} T_{on} \sqrt{\frac{R_L}{2TL}}$$

ได้ $$L = \frac{R_L}{2T} \frac{V_{IN}^2 T_{on(MAX)}^2}{V_O^2}$$

$$R_L = \frac{|V_O|}{|I_O|} = \frac{12V}{0.42A} = 28.57\Omega \approx 28\Omega$$

ได้ $$L = \frac{28}{2 \times 40\mu s} \times \frac{8^2 \times (19.2\mu s)^2}{12^2} = 57.34\mu H$$

ใช้ $$L \approx 60\mu H$$

จากสมการ $$I_{Pk} = \frac{V_{IN(MIN)} T_{on(MAX)}}{L}$$

$$= \frac{8 \times 19.2\mu}{57\mu} = 2.7A$$

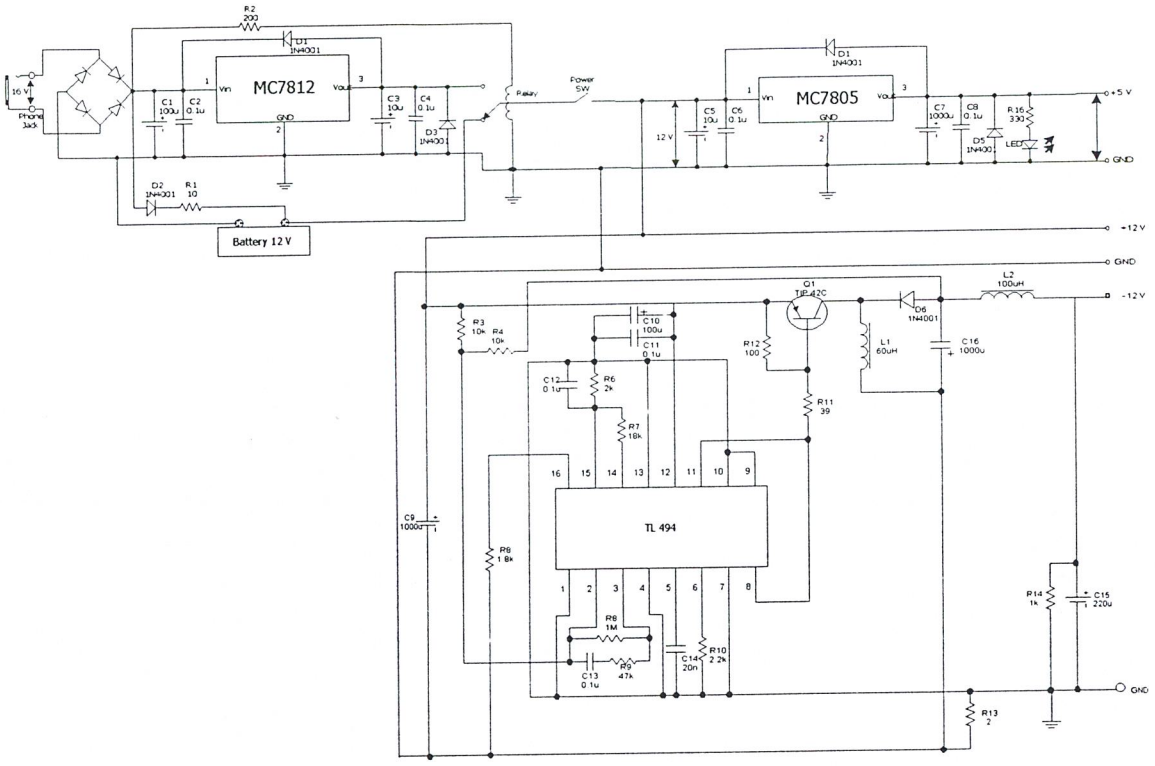
หาค่า C จากตาราง1.4 ใน DATA MC 33063 หรือ 78S40

$$C \geq \frac{9 I_O t_{on}}{V_{ripple}}$$

ให้ $$V_{ripple} = 50mV_{PP}$$

$$C \geq \frac{9 \times 0.25 \times 19.2\mu}{50mV_{PP}} \geq 864\mu F$$

ใช้ $$C = 1000\mu F$$



รูปที่ 4-15 วงจร dc to dc converter

ระบบที่ออกแบบมาทั้งหมดจะใช้ไฟประมาณ 120 mA ฉะนั้นถ้าเราใช้แบตเตอร์รี่ที่มี ขนาด 12 โวลท์ 1.2 Ah ระบบก็จะสามารถเก็บข้อมูลได้นานประมาณ 10 ชั่วโมง หรือหากต้องการให้ระยะเวลาในการเก็บข้อมูลเพิ่มขึ้น ก็สามารถเลือกใช้แบตเตอร์รี่ที่มีขนาด 12 โวลท์ 40Ah ได้เพราะจะทำให้เก็บข้อมูลได้ 13 ถึง 15 วัน แต่ก็อาจจะมีปัญหาว่าถ้าขนาดของแบตเตอร์รี่ใหญ่ขึ้นการเคลื่อนย้ายก็จะมีความลำบากมากขึ้นไปด้วย

4.6 การตรวจสอบแรงดันแบตเตอร์รี่

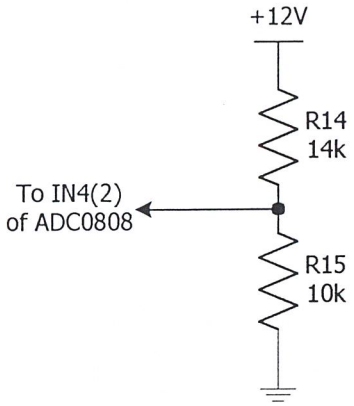
เนื่องจากขนาดของแรงดันไฟเลี้ยงมีผลอย่างมากกับการวัดอุณหภูมิ จึงได้ออกแบบให้ระบบมีการตรวจสอบแรงดันแบตเตอร์รี่ก่อนการวัดทุกครั้งโดยใช้ตัวต้านทานแบ่งแรงดันดังรูป 4-15 ซึ่งจะมีการทำงานดังนี้

- เมื่อแรงดันไฟเลี้ยงเท่ากับ 12 โวลท์

$$V_{O(MAX)} = \frac{(10k)(12V)}{(14k + 10k)} = 5V$$

- เมื่อแรงดันไฟเลี้ยงเท่ากับ 11 โวลท์

$$V_{O(MIN)} = \frac{(10k)(11V)}{(14k + 10k)} = 4.58V$$



รูปที่ 4-16 วงจรตรวจสอบแรงดันแบตเตอร์รี่

จากนั้นค่า $V_{O(MIN)}$ที่ได้จะถูกนำไปผ่านวงจร ADC เพื่อเปลี่ยนเป็นรหัสใน 256 ระดับจะได้ 204 เท่ากับ CCh (เลขฐานสิบหก) แล้วนำเอารหัสดังกล่าวมาเป็นตัวตรวจสอบแบตเตอร์รี่

บทที่ 5

การทดลองและผลการทดลอง

จากการทดสอบวงจรนั้น ในเบื้องต้นได้แยกทดสอบเป็นส่วน ๆ เพื่อที่จะสามารถตรวจสอบการทำงานของวงจรเหล่านั้นได้สะดวกโดยมีผล การทดลองดังนี้

5.1 ภาคเซ็นเซอร์อุณหภูมิ

5.1.1 ผลที่อ้างอิงจาก data sheet

เราจะสามารถคำนวณหาค่าแรงดันผลต่างที่เอาท์พุตได้ โดยใช้สมการข้างล่างนี้

$$V_{sensor} = V_{OUT_{T_O}} \times \frac{T}{T_O} \qquad (5.1)$$

$$V_{diff} = \left(V_{sensor} - V_{ref}\right) \qquad (5.2)$$

$$V_{OUT} = A_V \times V_{diff} \qquad (5.3)$$

กำหนดให้

T : คืออุณหภูมิที่เราสามารถรู้ได้ ($^{\circ}K$)

T_O : คืออุณหภูมิอ้างอิง ($298^{\circ}K$)

$V_{OUT_{T_O}}$: คือค่าแรงดันเอาท์พุตที่อุณหภูมิอ้างอิง (2.982V ที่ $25^{\circ}C$)

V_{sensor} : คือแรงดันเอาท์พุตที่ LM335

V_{diff} : แรงดันผลต่าง

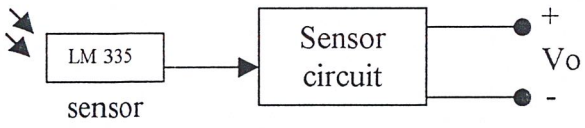
V_{ref} : ค่าแรงดันอ้างอิงเท่ากับ 2.731 โวลท์

$A_V = 10$: ค่าอัตราการขยายของวงจรขยายแรงดันผลต่าง (Differentail Amplifier)

ตารางที่ 5-1 แสดงความสัมพันธ์ระหว่างค่าอุณหภูมิ หน่วย °C, °K และ แรงดันเอาต์พุต (จากการคำนวณ)

อุณหภูมิ		V_{diff} (V)	V_{out} (V)
องศาเซลเซียส	องศาเคลวิน		
0	273	0.00	0.00
5	278	0.05	0.50
10	283	0.10	1.00
15	288	0.15	1.50
20	293	0.20	2.00
25	298	0.25	2.50
30	303	0.301	3.01
35	308	0.351	3.51
40	313	0.401	4.01
45	318	0.451	4.51
50	323	0.501	5.01

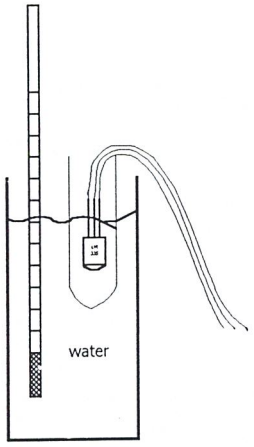
5.1.2 ผลการทดลองที่ได้จากวงจรตรวจวัดอุณหภูมิ



รูปที่ 5-1 วิธีการทดลองวงจรเซ็นเซอร์อุณหภูมิ

วิธีการทดลอง

ในการทดสอบหาคุณสมบัติของภาคเซ็นเซอร์ เราจะทำการทดลองโดยการวัดอุณหภูมิของน้ำ ซึ่งในการวัดนั้นเราจะใช้การวัดจากเซ็นเซอร์แล้วเปรียบเทียบผลที่ได้กับการวัดโดยใช้เทอร์โมมิเตอร์ โดยในการทดลองนั้นจะทำการวัดที่เอาท์พุตของเซ็นเซอร์โดยมีการเปลี่ยนแปลงค่าแรงดันที่จ่ายให้กับตัวเซ็นเซอร์ ดังรูปที่ 5-2

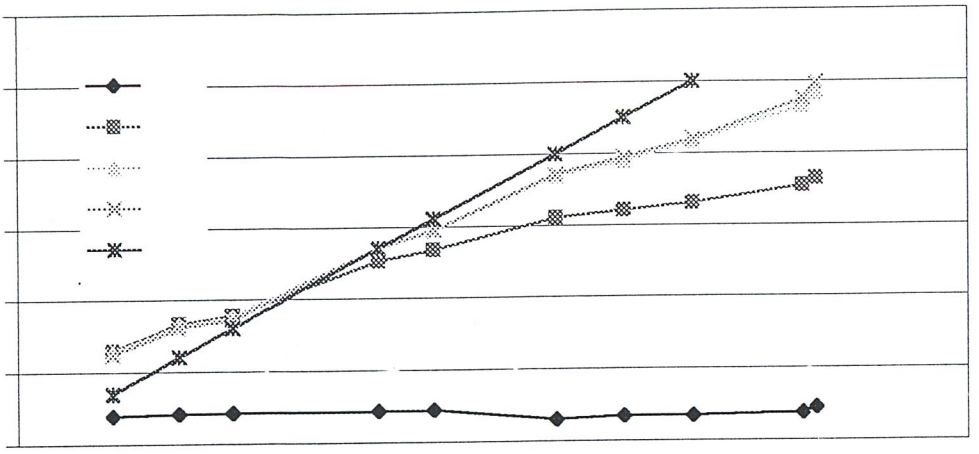


รูปที่ 5-2 การทดสอบคุณสมบัติของภาคเซ็นเซอร์

ซึ่งจะเห็นได้ว่าผลที่ได้จากการทดลองเมื่อนำมาเปรียบเทียบกับผลที่ได้จากการคำนวณจะเห็นว่ามีความแตกต่างกันค่อนข้างมาก เนื่องจากว่าขณะที่ทำการทดลองนั้นได้ใช้พลาสติกหุ้มตัวเซ็นเซอร์เอาไว้เพื่อไม่ให้เสียหายเมื่อโดนน้ำนั่นเอง อีกประการณ์หนึ่งแรงดันไฟเลี้ยงที่จ่ายให้กับภาคเซ็นเซอร์นั่นเมื่อมีค่าลดลงมากก็จะทำให้ความสามารถในการทำงานของตัวเซ็นเซอร์เองไม่เป็นไปตามที่เราต้องการด้วย ซึ่งจะได้ผลการทดลองดังตารางที่ 5-2

ตารางที่ 5-2 แสดงความสัมพันธ์ระหว่างค่าอุณหภูมิ หน่วย องศาเซลเซียส และแรงดันเอาต์พุต (จากการทดลอง)

อุณหภูมิ (เซลเซียส)	แรงดันเอาท์พุต (โวลท์)			
	V_{CC}= 9 โวลท์	V_{CC}= 10 โวลท์	V_{CC}= 11 โวลท์	V_{CC}= 12 โวลท์
7	0.385	1.29	1.27	1.24
12	0.417	1.66	1.63	1.61
16	0.423	1.77	1.75	1.71
27	0.43	2.515	2.697	2.695
31	0.44	2.67	2.95	2.945
40	0.297	3.1	3.69	3.721
45	0.34	3.2	3.9	3.94
50	0.345	3.3	4.16	4.2
58	0.38	3.64	4.65	4.75
59	0.45	3.54	4.84	4.97



รูปที่ 5-3 กราฟแสดงความสัมพันธ์ระหว่างอุณหภูมิ และแรงดันเอาท์พุตเมื่อจ่ายไฟเลี้ยงค่าต่างๆ

5.2 ภาคการแปลงสัญญาณอนาลอกเป็นดิจิตอล (Analog to Digital converter)

ตารางที่ 5-3 แสดงแรงดันเอาต์พุตและค่าข้อมูลไบนารีที่ได้จากการแปลงสัญญาณอนาลอกเป็นดิจิตอล

อุณหภูมิ		Vout (V)	Binary code
องศาเซลเซียส	องศาเคลวิน		
0	273	0.012	0000 0001
5	278	0.144	0000 0111
10	283	0.597	0001 1111
15	288	1.014	0011 0100
20	293	1.605	0101 0011
25	298	1.991	0110 0110
30	303	2.353	0111 1000
35	308	2.832	1001 0001
40	313	3.223	1010 0111
45	318	3.763	1100 0011
50	323	4.261	1101 1101
55	328	4.763	1111 0110
58	331	5.057	1111 1111

เส้นกราฟข้างบนนี้ได้มา จากการสุ่มเก็บข้อมูล โดยการวัดอุณหภูมิ จากน้ำร้อนที่ 50 องศา เซล -

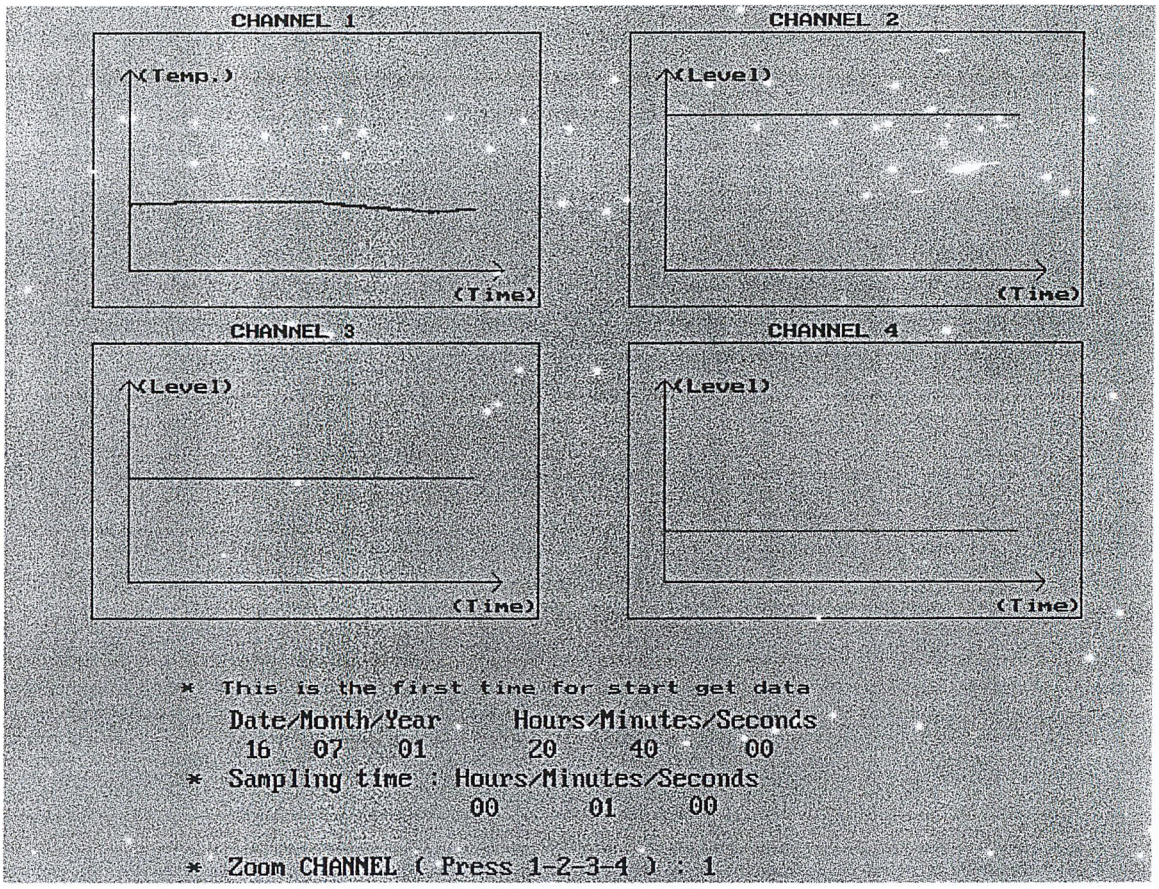
เซียส แล้งค่อยๆปล่อยให้น้ำเย็นลง เราสามารถรู้ค่าอุณหภูมิที่แรงดันจุดต่างๆ โดยการเปรียบเทียบกับตารางที่ 5-3 ซึ่งตารางดังกล่าวนี้ได้มาจากการคำนวณ

ตารางที่ 5-4 แสดงความสัมพันธ์ระหว่างแรงดันเอาท์พุตและอุณหภูมิ (จากการคำนวณ)

อุณหภูมิ (องศาเซลเซียส)	แรงดันผลต่าง (V)	แรงดันเอาท์พุต (V)	อุณหภูมิ (องศาเซลเซียส)	แรงดันผลต่าง (V)	แรงดันเอาท์พุต (V)
0	0.000	0.000	26	0.260	2.600
1	0.010	0.100	27	0.270	2.700
2	0.020	0.200	28	0.280	2.800
3	0.030	0.300	29	0.290	2.900
4	0.040	0.400	30	0.300	3.000
5	0.050	0.500	31	0.310	3.100
6	0.060	0.600	32	0.320	3.200
7	0.070	0.700	33	0.330	3.300
8	0.080	0.800	34	0.340	3.400
9	0.090	0.900	35	0.350	3.500
10	0.100	1.000	36	0.360	3.600
11	0.110	1.100	37	0.370	3.700
12	0.120	1.200	38	0.380	3.800
13	0.130	1.300	39	0.390	3.900
14	0.140	1.400	40	0.400	4.000
15	0.150	1.500	41	0.410	4.100
16	0.160	1.600	42	0.420	4.200
17	0.170	1.700	43	0.430	4.300
18	0.180	1.800	44	0.440	4.400
19	0.190	1.900	45	0.450	4.500
20	0.200	2.000	46	0.460	4.600
21	0.210	2.100	47	0.470	4.700
22	0.220	2.200	48	0.480	4.800
23	0.230	2.300	49	0.490	4.900
24	0.240	2.400	50	0.500	5.000
25	0.250	2.500			

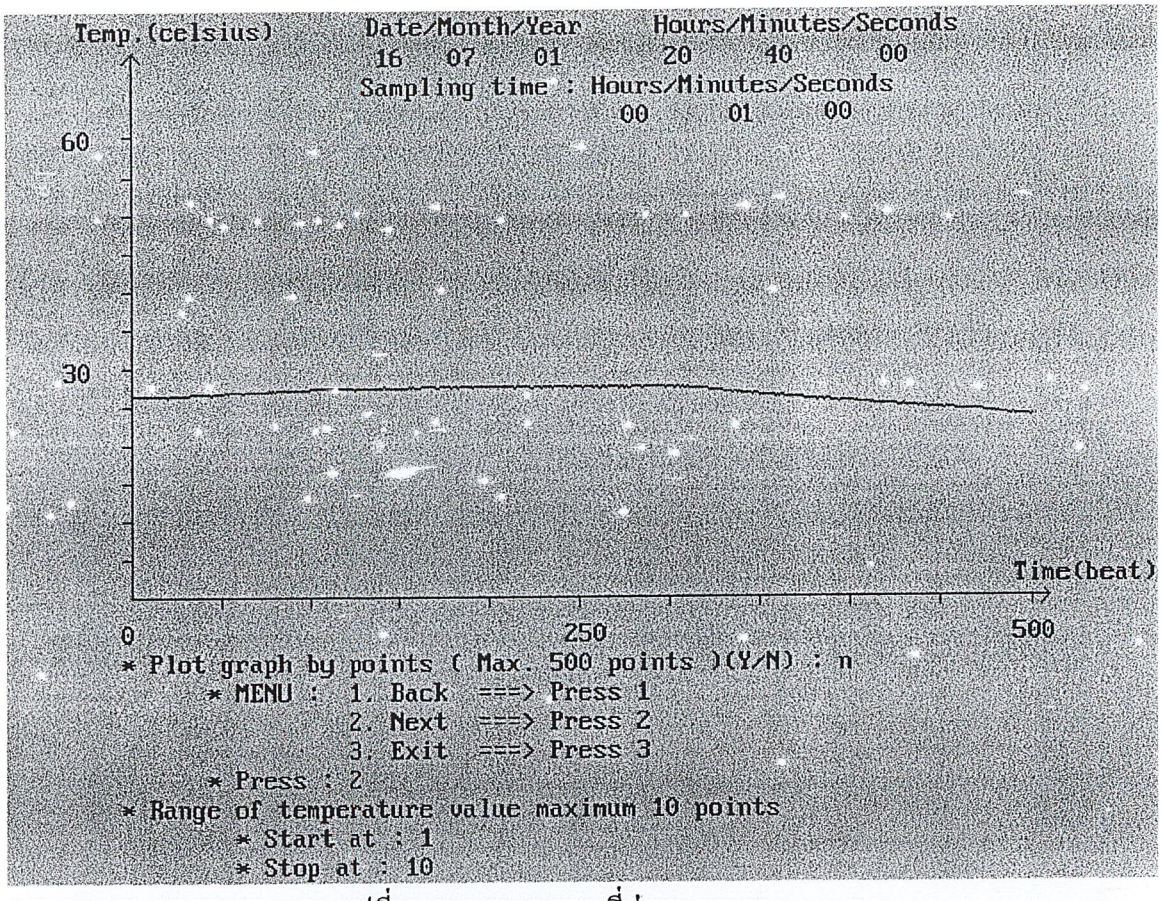
5.3 ผลการทดลองเก็บข้อมูลจริง

จากกราฟด้านล่างนี้เป็นผลที่ได้จากการทดลองโดยการวัดค่าอุณหภูมิที่ห้องทำงาน และ อีกสามช่องเป็นการวัดโดยการให้เก็บค่าแรงดันไฟตรง ซึ่งสามช่องที่ว่านี้สามารถที่จะนำไปใช้ได้ตามความต้องการ โดยสามารถที่จะเป็นอะไรก็ได้แล้วแต่ชนิดของเซ็นเซอร์

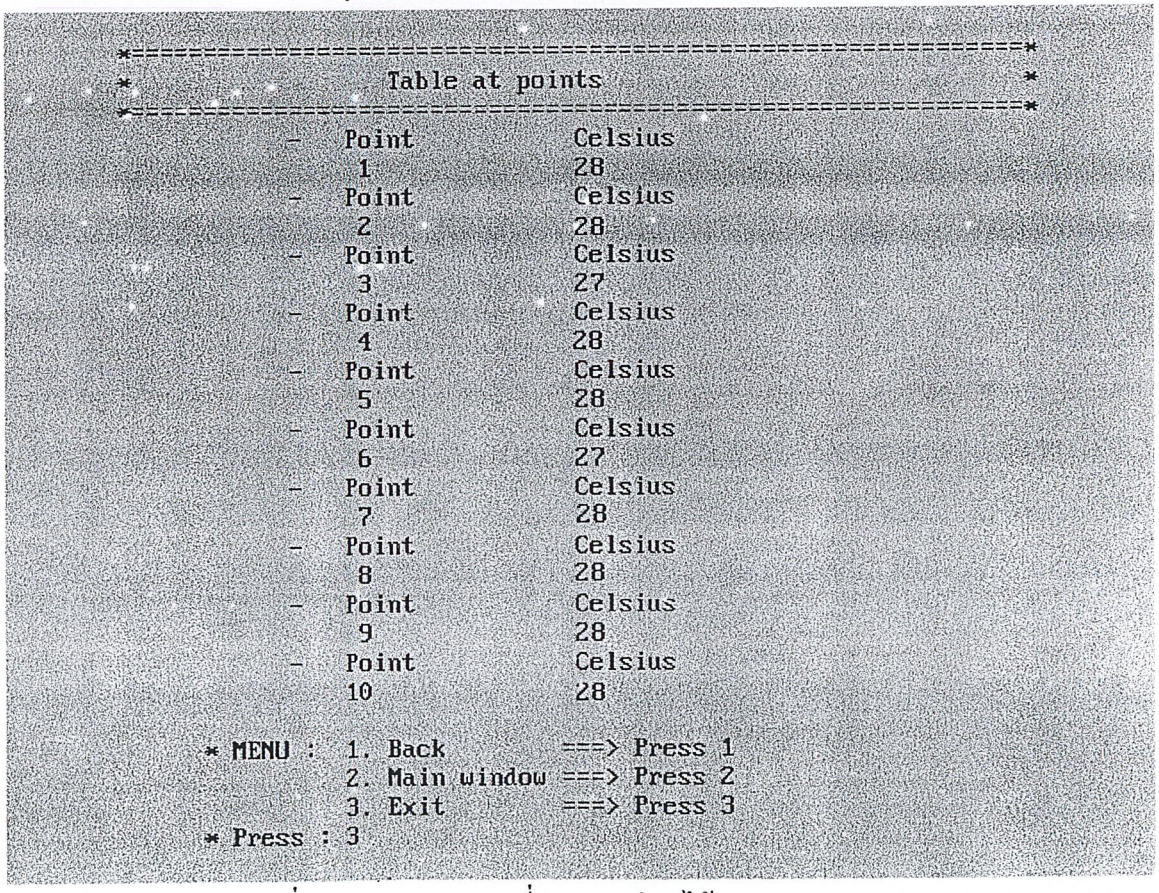


รูปที่ 5-4 ผลที่ได้จากวัดทั้ง 4 ช่อง

กราฟที่เห็นนี้เป็นกราฟที่แสดงบนหน้าจอคอมพิวเตอร์ โดยจะแสดงทั้ง 4 ช่องพร้อมกันและนอกจากนั้นก็ยังสามารถตรวจดูข้อมูลที่เก็บไว้ในแต่ละช่องได้ ดังรูปที่ 5-5 เป็นข้อมูลอุณหภูมิที่เก็บไว้ที่ช่อง 1 และก็ยังสามารถตรวจดูค่าอุณหภูมิที่จุดต่างๆโดยจะแสดงค่าเป็นตารางได้ครั้งละ 10 จุด



รูปที่ 5-5 ผลการทดลองที่ช่อง 1



รูปที่ 5-6 ผลการทดลองที่สามารถสุ่มดูได้ 10 จุด

- **เปรียบเทียบข้อมูลจริงกับข้อมูลที่แสดงบน LCD ของ Board และ ข้อมูลที่แสดงบน PC (Personal Computer)**

ไรยะเวลาในการเก็บข้อมูลต่อหนึ่งครั้ง (นาที)	ข้อมูลจึงที่ได้จากบาหรอด (องศา C)	ข้อมูลที่ได้จาก LCD บน Board (องศา C)	ข้อมูลที่ได้จากการแสดงบน Computer (องศา C)
5	24	24	23
10	22	22	21
15	22	20	20
20	21	20	19
25	21	20	19
30	21	20	19
35	21	20	19
40	21	20	19
45	21	20	19
50	21	19	18

จากตาราง เห็นว่าค่าอุณหภูมิที่ได้ไม่ว่าจะเป็นการแสดงข้อมูลบน LCD ของ Board data logger หรือการแสดงข้อมูลบน Computer จะมีค่าที่ใกล้เคียงกัน โดยมีค่าความผิดพลาดน้อยกว่า 3 % จากข้อมูลจึง

บทที่ 6
สรุปและวิจารณ์

จากแนวคิดที่วางไว้ในการสร้างระบบเก็บข้อมูลแบบหลายช่องในปริญญานิพนธ์ฉบับนี้ได้กล่าวถึงความเป็นมาของการทำปริญญานิพนธ์ระบบเก็บข้อมูลแบบหลายช่องโดยได้ให้รายละเอียดทฤษฎีที่จำเป็นสำหรับการสร้างโครงงานได้แก่ ทฤษฎีไมโครคอนโทรลเลอร์ การแปลงสัญญาณอนาลอกเป็นดิจิตอล ตลอดจนเซ็นเซอร์อุณหภูมิได้ทำการทดสอบส่วนที่เป็นเซ็นเซอร์สำหรับวัดอุณหภูมิโดยการวัดแรงดันเอาท์พุตที่อุณหภูมิค่าต่างๆ ดังแสดงในตารางที่ 5-2 จากแนวความคิดของโครงงานที่ว่าเราสามารถเก็บ ข้อมูลได้ทั้งหมด 4 ช่องด้วยกันแต่เนื่องจากข้อจำกัดในการดำเนินโครงงานดังนั้นในการทดลองจึงได้ ทำการวัดเฉพาะอุณหภูมิเพียงอย่างเดียว แต่ว่าในการนำไปใช้งานจริงก็จะสามารถประยุกต์ใช้งานได้โดยการเลือกอุปกรณ์เซ็นเซอร์ที่ความเหมาะสมเพื่อการใช้งานต่อไป

จากการดำเนินโครงงานมาทั้งหมด สามารถสรุปความก้าวหน้าของโครงงานได้ ดังนี้

- ระบบสามารถที่จะเก็บข้อมูล ณ ที่เวลาใดก็ได้ตามที่ผู้ใช้งานต้องการ โดยการกดสวิชท์เพื่อเลือกเวลา ซึ่งจะเป็น ชั่วโมง นาที และวินาที
- ข้อมูลที่เก็บได้จะถูกพักไว้ที่หน่วยความจำขนาด 32 กิโลไบต์ เพื่อที่จะมาสารถนำไปแสดงผลที่คอมพิวเตอร์ได้
- ข้อมูลที่แสดงบนจอคอมพิวเตอร์จะเป็นเส้นกราฟ แสดงทั้ง 4 ช่องพร้อมกัน สามารถเลือกดูข้อมูลแต่ละช่องได้ โดยที่หน้าจอจะแสดงเวลาที่เริ่มต้นเก็บข้อมูล และแสดงค่า Sampling time ที่เราใช้ในการเก็บข้อมูล
- ข้อมูลช่องที่ 1 จะเป็นค่าอุณหภูมิ และอีก 3 ช่องจะเอาไว้สำหรับใช้งานทั่วไป

ในปริญญานิพนธ์นี้จะมีบางส่วนที่ยังจะต้องมีการปรับปรุง นั่นก็คือในส่วนของไฟเลี้ยงที่จ่ายให้กับเซ็นเซอร์ที่เป็นวงจรไฟลบ 12 โวลท์ แบบสวิชท์ชิ่งเรกกูเลเตอร์ เนื่องจากวงจรนี้ดึงกระแสประมาณ 70 mA ของกระแสทั้งหมดในวงจร (120 mA) และปัญหาจากเอาท์พุตของเซ็นเซอร์ซึ่งจะไม่เที่ยงตรงเมื่อไฟเลี้ยงต่ำกว่า 11 โวลท์

ภาคผนวก ก. วงจรรวม

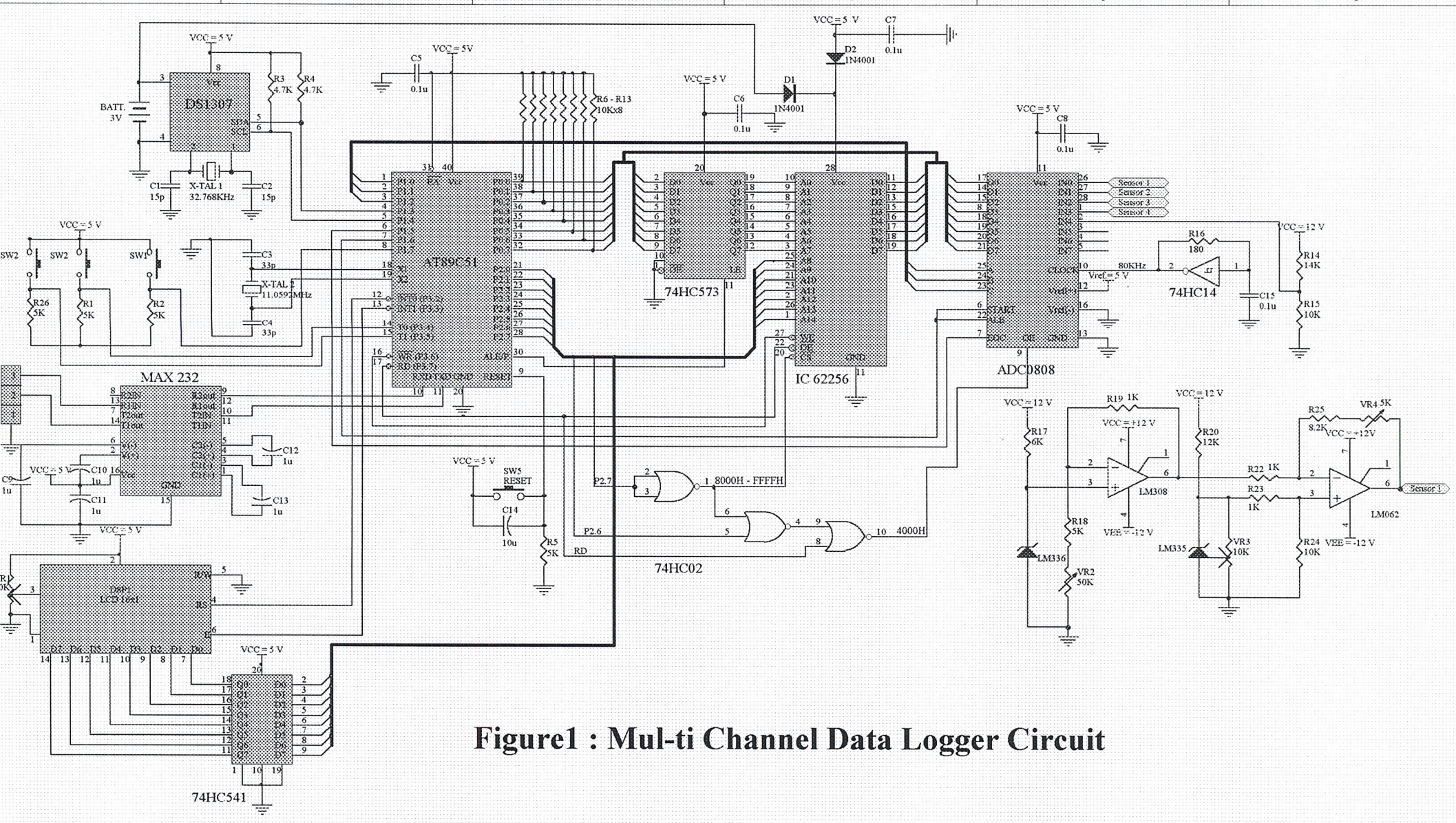


Figure1 : Mul-ti Channel Data Logger Circuit

Title		
Size: B	Number	Revision
Date: 24-Jul-2001		Sheet of
File: C:\logger\Logger1.DDB		Drawn By:

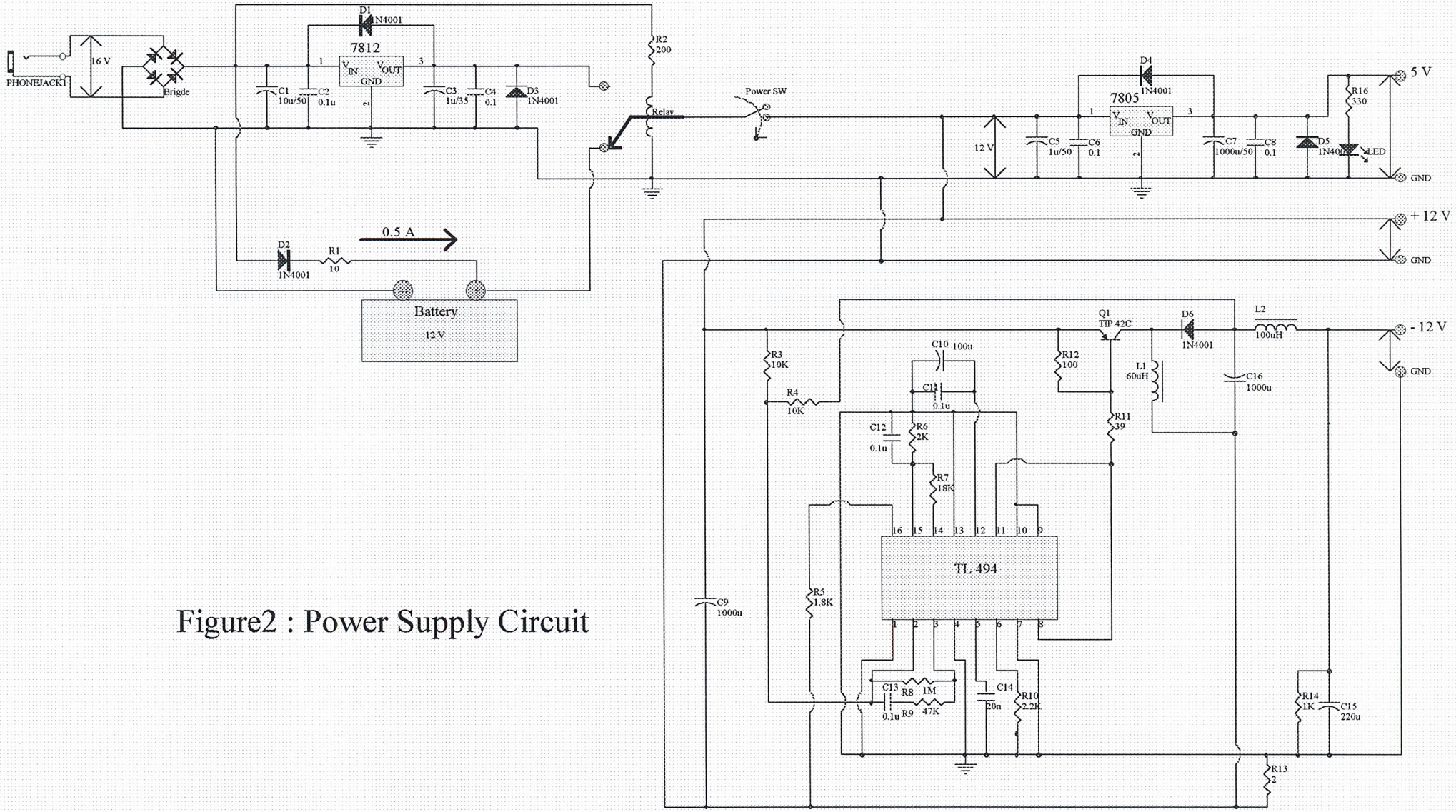


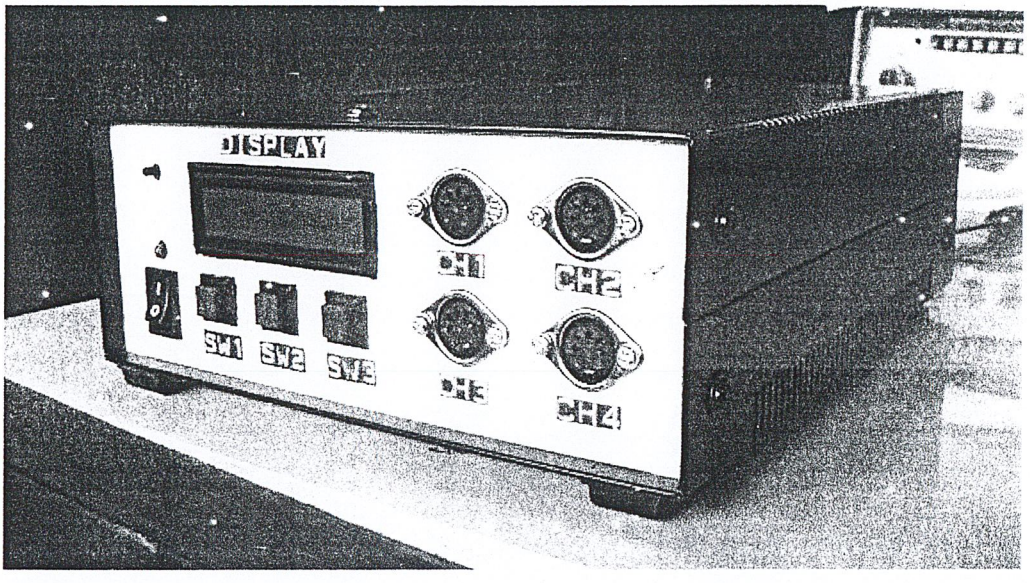
Figure2 : Power Supply Circuit

Title		
Size B	Number	Revision
Date: 24-Jul-2001		Sheet of
File: C:\logger\Logger1.DDB		Drawn By:

ภาคผนวก ข. คู่มือการใช้งาน

คู่มือการใช้งาน

เครื่องเก็บข้อมูลแบบ 4 ช่องสัญญาณ



จัดทำโดย : นาย คำผอง คงสมบูน

นางสาว ลัดสะหมี ไชสุลิยง

ดร.กิติพล ชิตสกุล (อาจารย์ที่ปรึกษา)

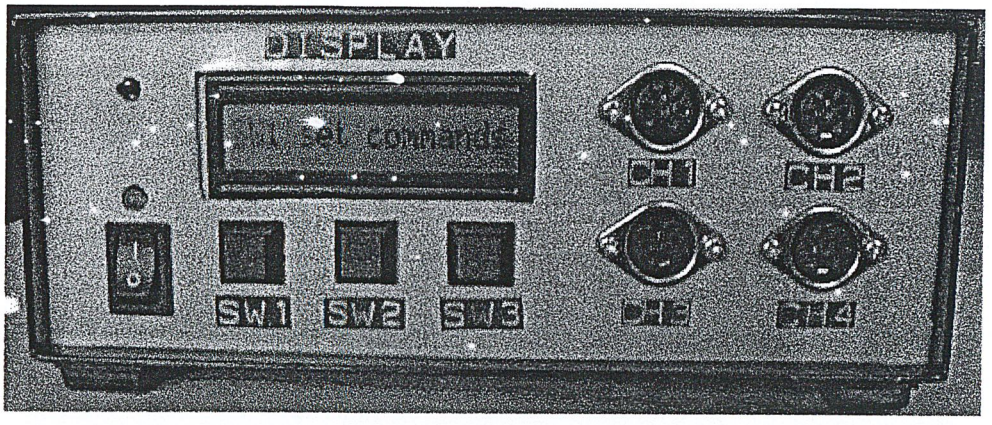
ปีการศึกษา 2544

Supported by NUOL – KMITL - JICA

คู่มือการใช้งาน

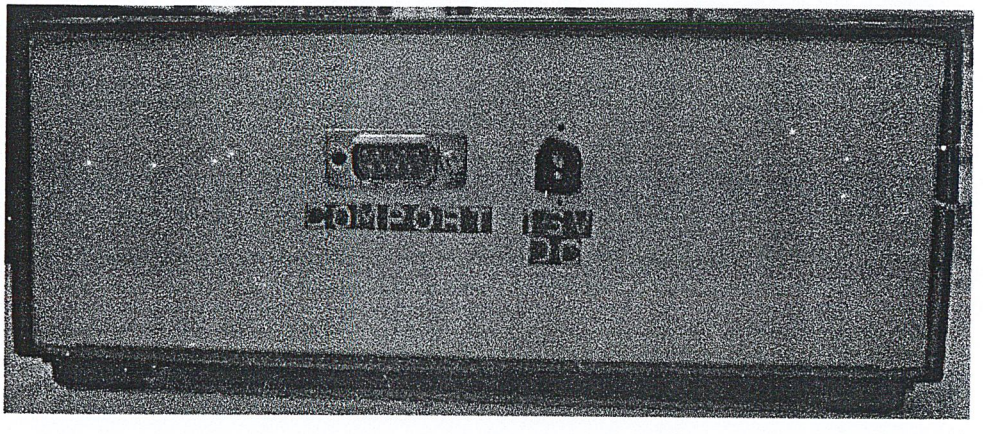
1. โครงสร้างของกล่อง

ด้านหน้า



รูปที่ 1 ภาพถ่ายด้านหน้าของตัวเก็บข้อมูล

ด้านหลัง

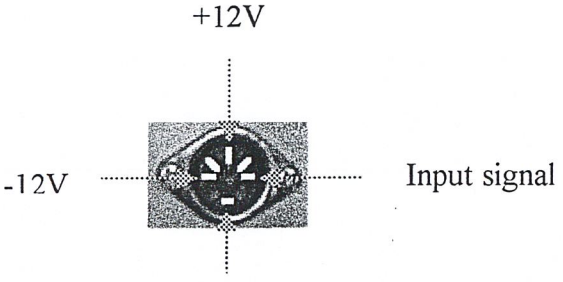


รูปที่ 2 ภาพถ่ายด้านหลังของตัวเก็บข้อมูล

2. หน้าที่ของปุ่มต่างๆ

- POWER SWITCH : ทำหน้าที่ เปิด-ปิด ไฟเลี้ยงให้กับวงจรทั้งหมด
- LED :เป็นตัวแสดงสภาวะ เปิด-ปิด ของเครื่อง
- RESET SWITCH : มีหน้าที่สำหรับ Reset ระบบใหม่เมื่อระบบทำงานผิดปกติ

- LCD : หน้าที่หลักคือแสดงสถานะต่างๆ ของ Board ไม่ว่าจะอยู่ในช่วงรอการตั้งคำสั่งเพื่อให้ Board ทำงาน หรือ รอการส่งข้อมูลขึ้นไปยังเครื่องไมโครคอมพิวเตอร์ รวมถึงเป็นนาฬิกาอีกด้วย
- Ch1 , Ch2 , Ch3 และ Ch4 : เป็นอินพุทสำหรับรับข้อมูลจาก Sensor โดยโครงสร้างแสดงดังรูปที่1. . จะเห็นว่า Jack มีสายสัญญาณ ดั่งนี้คือ :



รูปที่ 3 แสดงส่วนประกรอบของ Channel Jack

- สายสัญญาณที่เป็น Input ของ Boardโดยได้จาก Output ของ Sensor
- สายไฟ ±12 V เป็นไฟที่ออกมาจาก Board เพื่อนำไปเป็นไฟเลี้ยงให้ กับ Sensor
- GND : กราว

- SW1 , SW2 , SW3 : เป็นส่วนควบคุมการทำงานของระบบ และจะขออธิบายการใช้งานในขั้นต่อไป.
- Comport หรือ RS232 port : เป็นพอร์ทที่ใช้ส่งข้อมูลระหว่างตัวเก็บข้อมูลไปยังเครื่องคอมพิวเตอร์ โดยผ่านทางพอร์ท COM1 ของ Computer .

3. ขั้นตอนการตั้งคำสั่งต่างๆเพื่อให้ Board ทำการเก็บข้อมูล.

เมื่อเริ่มต้นกด SW Power เพื่อจ่ายแรงดันให้กับระบบแล้ว LCD จะเป็นตัวแสดงสถานะต่างๆดั่งนี้ :

- **ขั้นตอนที่ 1 .**

LCD Display จะแสดงข้อความออกมาเพื่อให้เลือกแบบการทำงานของเครื่องเก็บข้อมูลด้วยการใช้ SW1 และ SW2 ดั่งนี้

LCD Display

SW1 set commands

- เมื่อกด SW1 ระบบก็จะเข้าสู่การตั้งค่าคำสั่งต่างๆ เพื่อให้ Board ทำงาน .

LCD Display

SW2 send data

- เมื่อกด SW2 ก็จะเป็นการส่งข้อมูลขึ้นไปบน Computer แต่ Computer ต้องอยู่ในสถานะรอรับข้อมูลอยู่แล้ว.

สมมุติว่าเรากด SW1 แล้ว LCD จะแสดงค่าเวลาต่างๆ ออกมาแต่ถ้าค่าเวลาไม่ถูกต้องก็สามารถแก้ไขได้ด้วยการกด SW2 ซึ่งเป็นการตั้งค่าเวลาใหม่ จากนั้นสามารถควบคุมการตั้งค่าได้ด้วย SW1 และ SW2

LCD Display

T:01/01/01 00 : 00

- SW1 : ทำหน้าที่เป็น Enter
- SW2 : ทำหน้าที่เลื่อนตัวเลข

แต่ถ้าเวลาถูกต้องแล้วก็สามารถข้ามขั้นตอนนี้ไปได้ด้วยการกด SW1

- **ขั้นตอนที่ 2 .**

เป็นการตั้งช่องในการเก็บข้อมูลซึ่งทั้งหมดมีอยู่ 4 ช่องสัญญาณคือ : Ch1 , Ch2 , Ch3 และ Ch4 สามารถควบคุมได้จาก SW1 และ SW2 เช่นกัน

LCD Display

Channel : (1-2-3-4)

- SW1 : ทำหน้าที่เป็น Enter
- SW2 : ทำหน้าที่เลื่อนจำนวนช่องสัญญาณ

- **ขั้นตอนที่ 3 .**

เป็นการตั้งเวลาสุ่มข้อมูลให้กับระบบว่าจะให้ Board เรานานเท่าไรจึงจะเก็บข้อมูลหนึ่งครั้ง การตั้งค่านี้ทำเหมือนกันกับการตั้งเวลาโดยสามารถกำหนดเวลาในการสุ่มข้อมูลต่ำสุด 1 วินาที

LCD Display	**กด SW2**	**LCD Display**
Sampling time	⇨	Time : 01/30/10

- SW1 : ทำหน้าที่เป็น Enter
- SW2 : ทำหน้าที่เลื่อนตัวเลข

- **ขั้นตอนที่ 4 .**

เมื่อตั้งค่าต่างๆ ให้กับ Board แล้วจากนั้นก็เริ่ม Start โดยการกด SW1 ขณะเดียวกันก็สามารถหยุดระบบได้ด้วยการกด SW2 .

LCD Display

Start by SW1

- SW1 : Start ให้ Board เริ่มเก็บข้อมูล
- SW2 : Stop เก็บข้อมูล

ในขณะเก็บข้อมูลอยู่ LCD จะแสดงวันและเวลาออกมา แต่ถ้าอยากดูข้อมูลก็ให้กด SW1 แล้ว LCD ก็จะแสดงข้อมูลในแต่ละช่องสัญญาณออกมาเป็นค่าตัวเลข.

เมื่อกด SW2 จะเป็นการหยุดเก็บข้อมูล แล้วระบบจะไปอยู่ในขั้นตอนที่ 1 เพื่อรอการส่งข้อมูลขึ้น PC

ส่วน SW3 จะถูกใช้งานก็ต่อเมื่อระบบมีระดับแรงดันไฟเลี้ยงต่ำกว่า 11 V ในขณะเดียวกัน LCD จะแสดงข้อความว่า LOW BATTERY แต่เมื่อเรา Charge Bat. ให้มีค่ามากกว่า 11 โวลท์ LCD ก็จะแสดงข้อความว่า BATTERY PASS ตอนนี้ ระบบจะรอการกด SW3 อยู่เพื่อเริ่มเก็บข้อมูลต่อไป

.

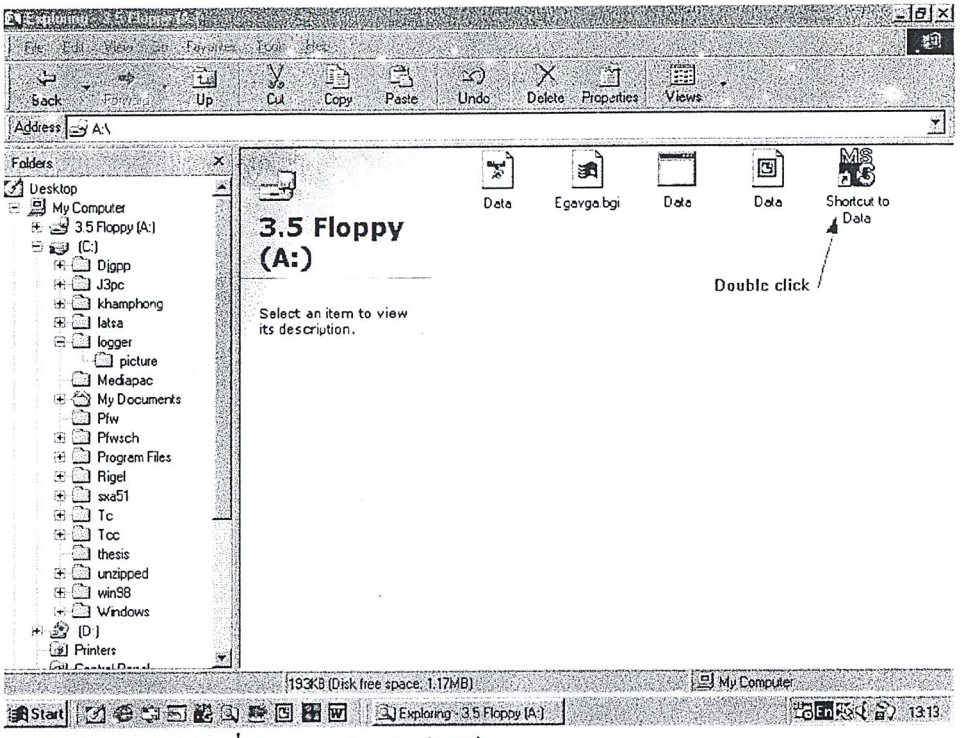
การแสดงข้อมูลบน PC

- **ขั้นตอนที่ 1.**

ต่อ Comport หรือ RS232 port เข้ากับ COM1 ของ Computer .

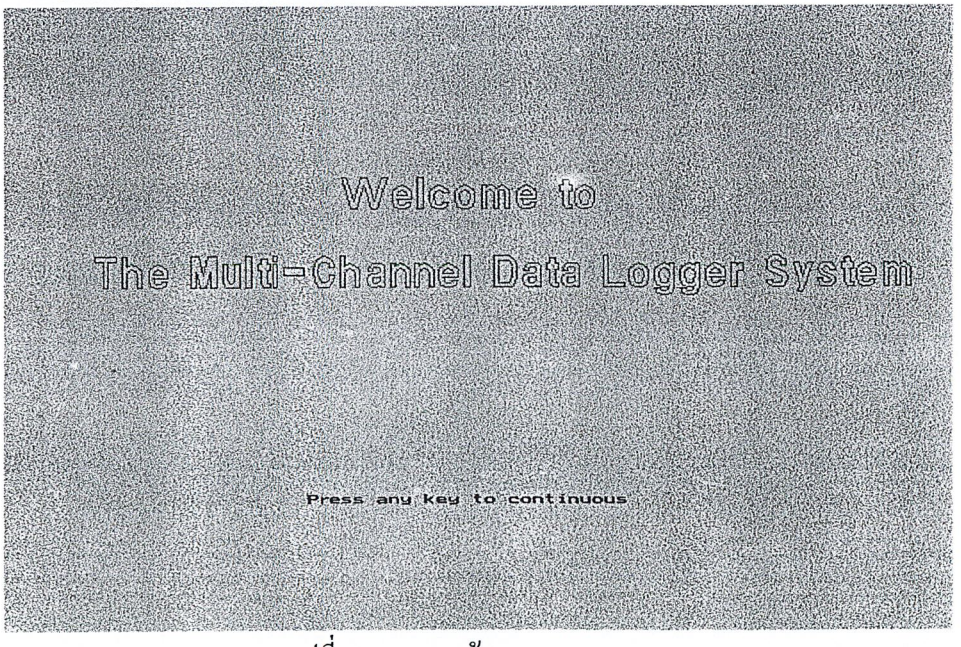
- **ขั้นตอนที่ 2 .**

เปิด File จาก Floppy disk ที่มีชื่อว่า : data โดยการใช้ Mouse click สองครั้ง ดูรูปที่ 4



รูปที่ 4: แสดงการเปิดไฟล์จาก Floppy disk

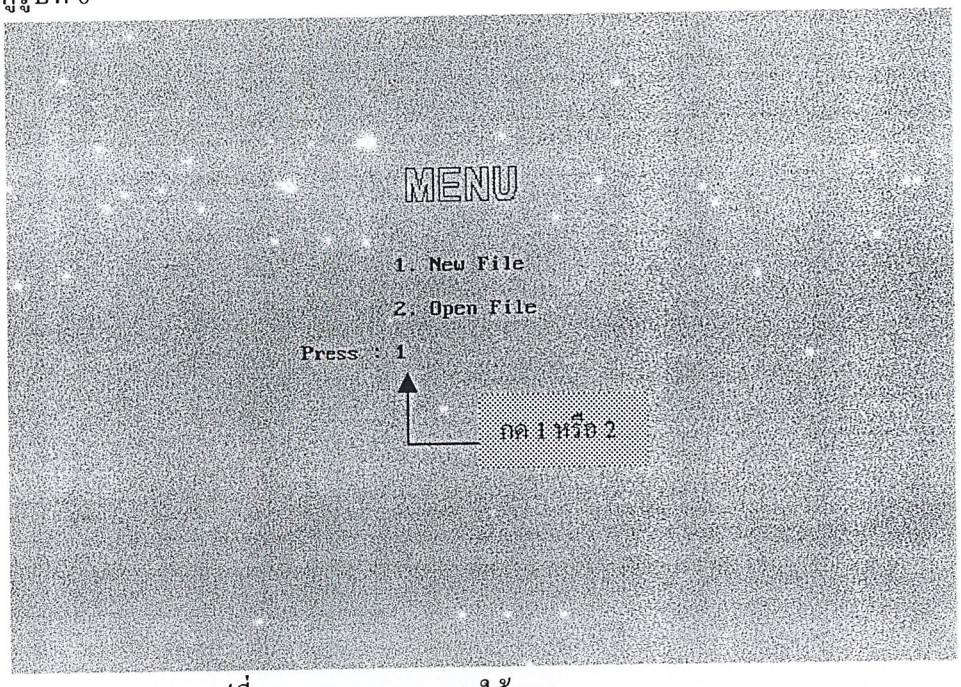
จากนั้นจะได้รูปที่ 5 แล้วให้กด Enter



รูปที่ 5 : เสนอหน้าจอ

- **ขั้นตอนที่ 3 .**

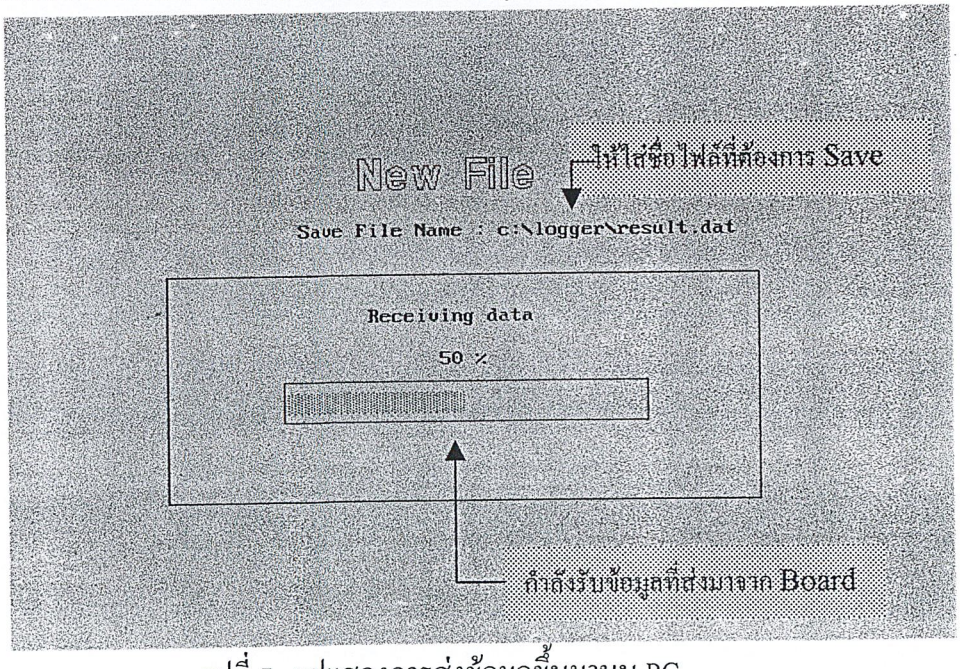
หน้าจอ Computer จะแสดง Menu ออกมา เพื่อให้เลือกว่า จะเปิดไฟล์ใหม่หรือเปิดไฟล์เก่าที่เก็บไว้ดูรูปที่ 6



รูปที่ 6 : แสดง Menu การใช้งาน

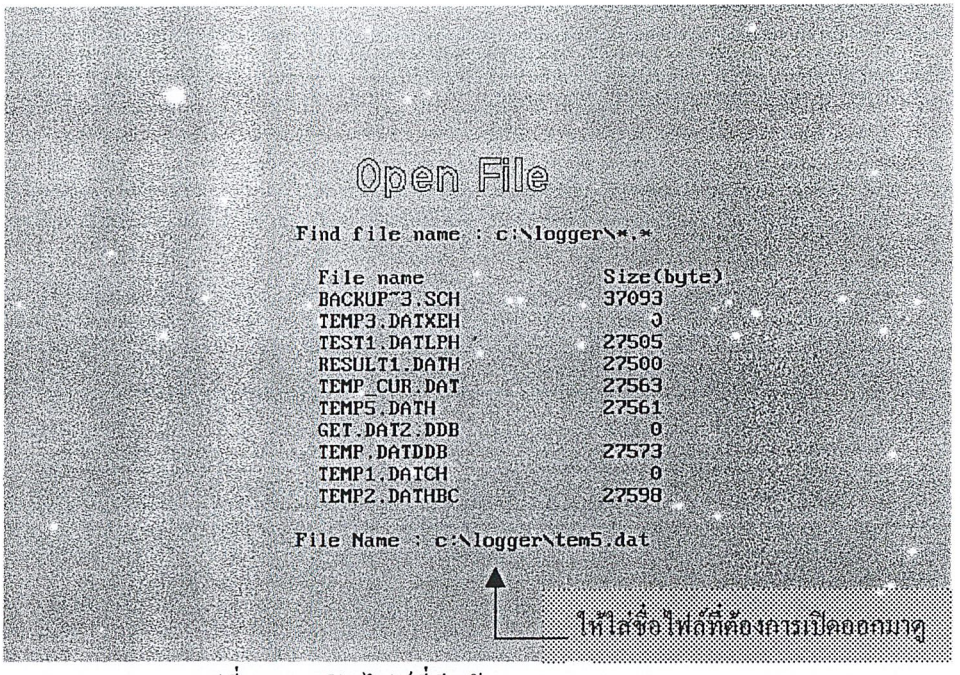
เมื่อกด 1 แสดงว่าท่านเลือกที่จะเปิดไฟล์ใหม่ (New File) ดูรูปที่ 7

ในขั้นตอนนี้ให้ใส่ชื่อไฟล์เพื่อจะ Save เก็บไว้ต่อจากนั้น Program จะรอการส่งข้อมูลจาก Board เวลานี้ให้กด SW2 ที่ Board เพื่อเป็นการส่งข้อมูลขึ้นมา .



รูปที่ 7 : รูปแสดงการส่งข้อมูลขึ้นมาบน PC

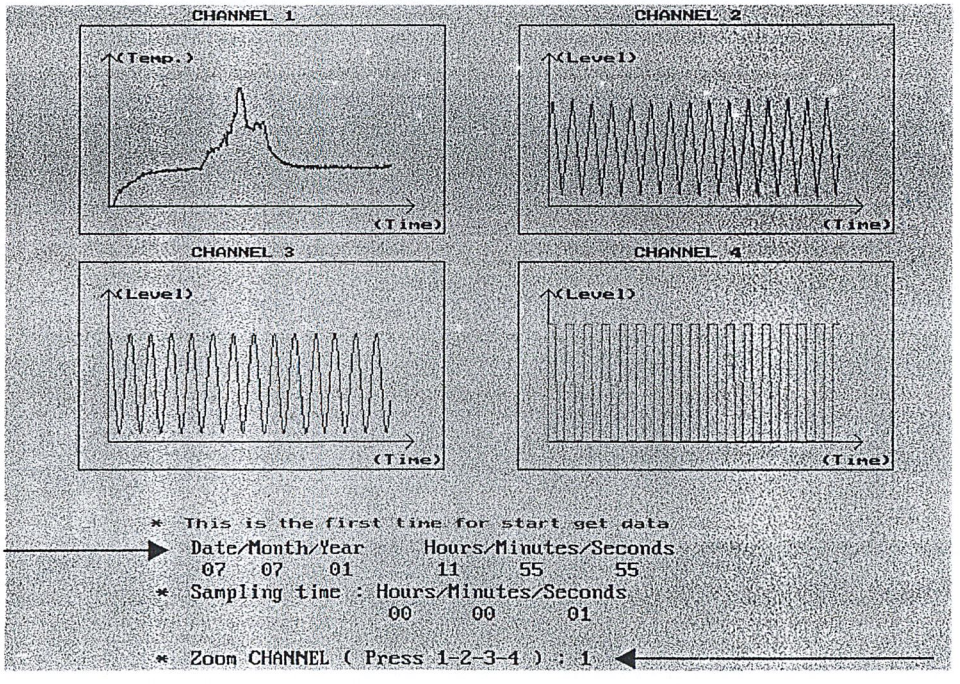
แต่ถ้ากด 2 แสดงว่าเป็นการเลือกเปิดไฟล์ที่เก็บไว้เอาออกมา Plot graph จากนั้นให้ใส่ชื่อไฟล์ที่ต้องการแสดงผลดูรูปที่ 8.



รูปที่ 8: การเปิดไฟล์ที่มีแล้ว

- **ขั้นตอนที่ 4 .**

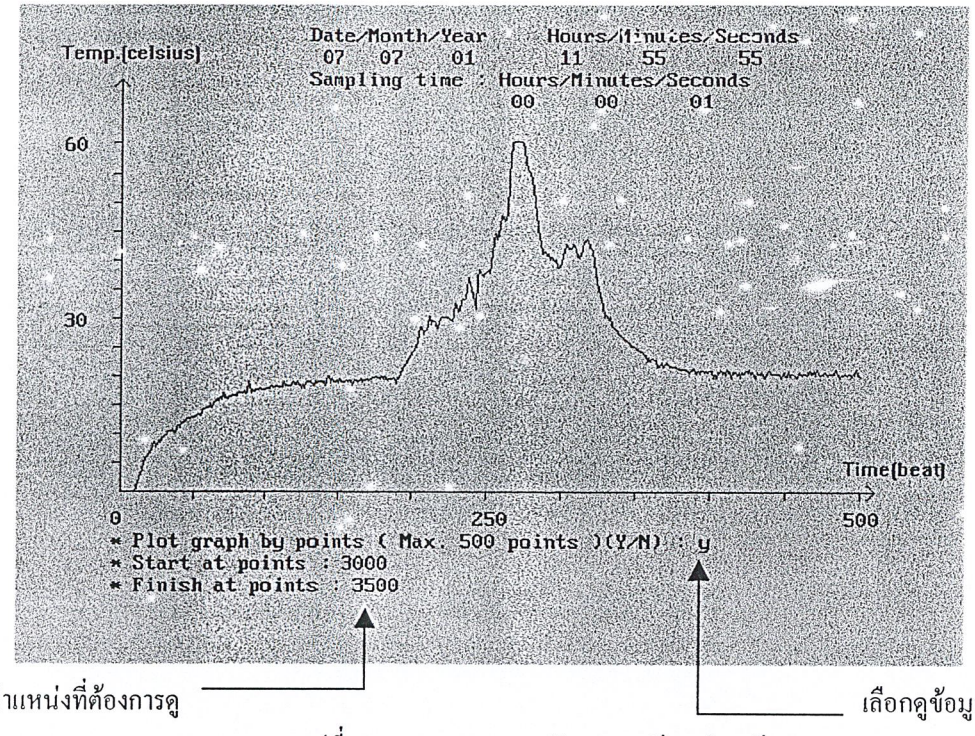
ข้อมูลที่ได้มาจะนำมา Plot เป็น Graph ในตอนนี้ Computer จะ Plot graph ตามจำนวนช่องที่ได้เลือกไว้ ดูรูปที่ 9 ถ้าช่องไหนไม่ถูกเลือกให้เก็บข้อมูลก็จะไม่มี Graph ให้เห็น.



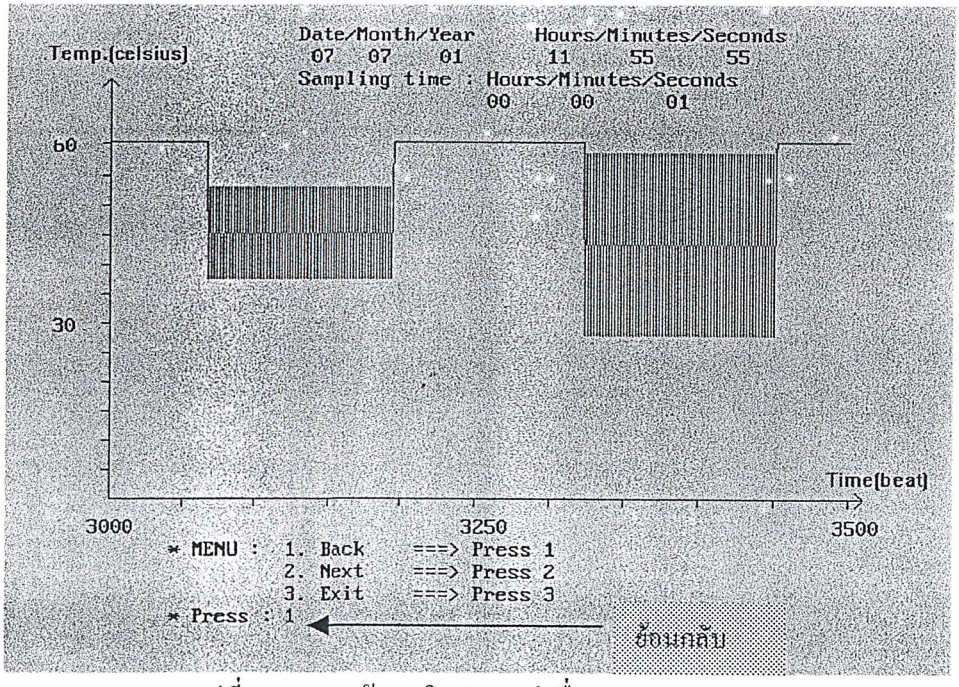
รูปที่ 9 : การ Plot graph ทั้ง 4 ช่องสัญญาณ

- <u>ขั้นตอนที่ 5.</u>

จากนั้นเราสามารถ Zoom ดูข้อมูลทีละช่องสัญญาณได้ และยังสามารถดูข้อมูลในช่วงเวลาต่างๆ ที่เก็บได้ ดูรูปที่ 10 และ รูปที่ 11

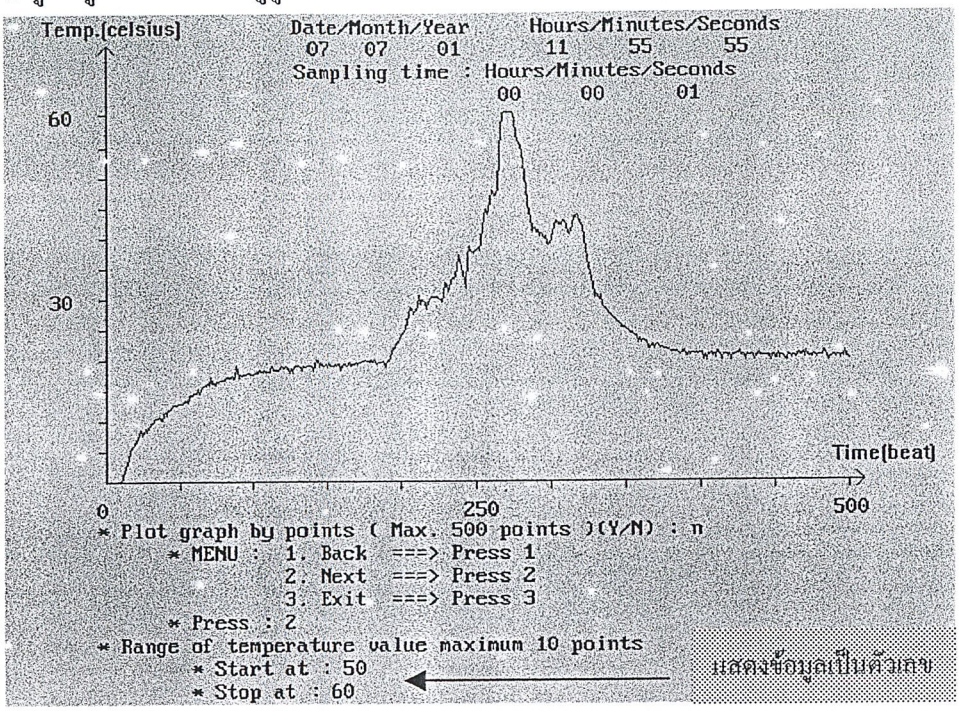


รูปที่ 10 : การ Zoom ดูสัญญาณเพียงช่องเดียว



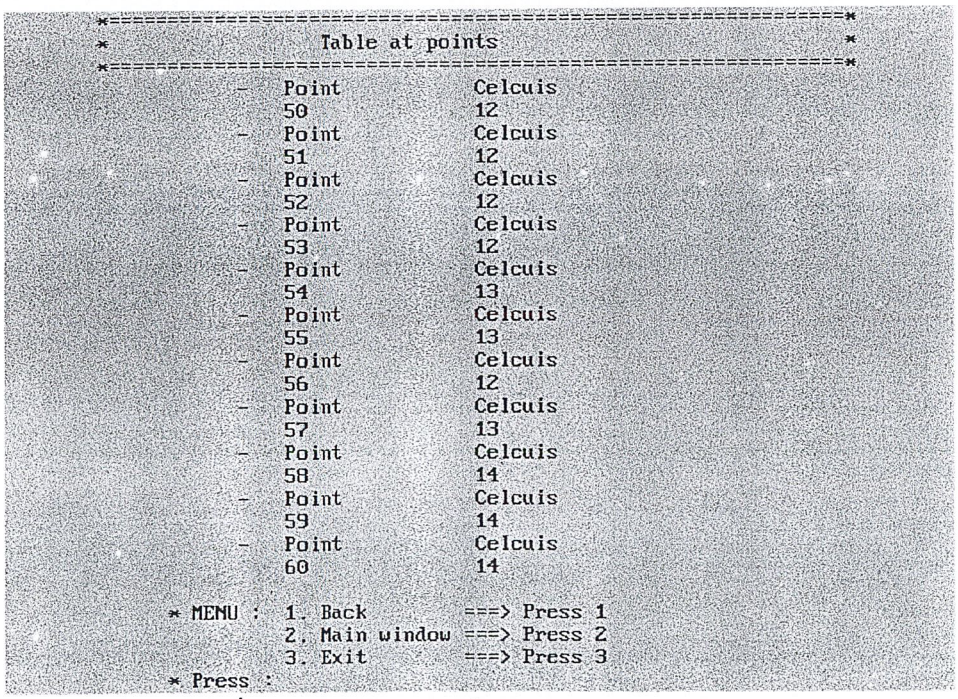
รูปที่ 11 : การดูข้อมูลในตำแหน่งอื่นๆ

ถ้าไม่อยากดูข้อมูลที่ช่องต่อไป ดูรูปที่ 12.



รูปที่ 12 : แสดงการกำหนดค่าที่จุดต่างๆ เพื่อดูข้อมูลเป็นค่าตัวเลข

จากรูปที่ 12. เรายังสามารถแสดงข้อมูลในแต่ละช่วงเวลาได้ตามที่ต้องการ แต่ต้องไม่เกิน 10 จุด ดูรูปที่ 13.



รูปที่ 16 : แสดงค่าต่างๆเป็นตัวเลข

ในการดูข้อมูลต่างๆสามารถใช้ Keyboard ควบคุมหรือป้อนคำสั่งได้เพียงอย่างเดียว

ภาคผนวก ค. โปรแกรม

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <io.h>
#include <math.h>
#include <bios.h>
#include <dir.h>
#include <ctype.h>
#define T 32700
#define SETTINGS ( 0xE0 | 0x03 | 0x00 | 0x00 )

// Prototype
void window_channel(void);
void plot_4channel(void);
void window_temp_channel(void);
void window_a_channel(void);
void plot_a_channel(void);
unsigned new_file(void);
unsigned open_file(void);

FILE *buffer;
char file_name[15];
int  gdriver=DETECT,gmode=VGAHI;
int  data1,drive=3,mode=1,x,y,i,j,k,l,input=010,count=0;
unsigned char  data2,data3[T];
int  data4=500,channel,start=1,finish=1,t=0,first_data=10;

//Read serial port
char readcom(void)
{
  int  data1;
  unsigned char    data2;

  data1=0;

  while((data1&0x0100)==0)
  {
    if(kbhit()!=0)
    return(-1);
    data1=bioscom(3,0,0);
  }
  data2=bioscom(2,0,0);
  return(data2);
}

void main()
{
  int  temp,temp1,temp2,back,x4=570,x8=70,x9,y11=355;
  char   se_menu[2],condition;

win:            initgraph(&gdriver,&gmode,"c:\\tc");

     setcolor(13);
     settextstyle(BOLD_FONT,HORIZ_DIR,1);
     outtextxy(220,120," Welcome to ");
     outtextxy(50,180," The Multi-Channel Data Logger System ");
     setcolor(15);
     settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
     outtextxy(200,375," Press any key to continuous ");
     getch();
     closegraph();

choise:   initgraph(&gdriver,&gmode,"c:\\tc");
     setcolor(13);
     settextstyle(BOLD_FONT,HORIZ_DIR,1);
     outtextxy(260,100," MENU ");
     gotoxy(24,12);
     printf("       1. New File "
            "\n\n                2. Open File "
            "\n\n             Press : ");
     gets(se_menu);
     if(se_menu[0]=='1')
     {
       new_file();
     }
     else if(se_menu[0]=='2')
     {
       open_file();
```

```
      }
      else
      {
        printf(" \n Error , please select again ");
        delay(1000);
        delay(1000);
        delay(1000);
        clrscr();
        goto choise;
      }

cha_nel:  initgraph(&gdriver,&gmode,"c:\\tc");

      window_channel();
      plot_4channel();

      setcolor(13);
      printf("\n\n           * Select CHANNEL ( Press 1-2-3-4 ) : ");
      scanf("%d",&channel);

      if(channel>0&&channel<5)
      {
        if(channel==1)
        {
          closegraph();
          goto temp_graph;
        }
        else
        {
          closegraph();
          goto graph;
        }
      }
      else
      {
      closegraph();
      goto cha_nel;
      }

temp_graph:  initgraph(&gdriver,&gmode,"c:\\tc");

      window_temp_channel();
      plot_a_channel();

      gotoxy(5,23);
      printf("\n\n   * Plot graph by points ( Max. 500 points ) (Y/N) : ");
      scanf("%s",&condition);
      if(condition=='Y'||condition=='y')
      {
        printf("     * Start at points : ");
        scanf("%d",&start);
        printf("     * Finish at points : ");
        scanf("%d",&finish);
        if(start<=0||finish>32766||start>finish||(finish-start)>500)
        {
          closegraph();
          goto temp_graph;
        }
        else
        {
          closegraph();
          goto temp_graph_1;
        }
      }
      else if(condition=='N'||condition=='n')
      {
        printf("         *MENU : 1. Back  ====> Press 1 "
               "\n                 2. Next  ====> Press 2 "
               "\n                 3. Exit  ====> Press 3 ");
        printf("\n          * Press : ");
        scanf("%d",&back);
        if(back==1)
        {
          closegraph();
          goto cha_nel;
        }
        else if(back==2)
        {
          printf("        * Range of temperature value maximum 10 points ");
          printf("\n          * Start at : " );
```

```
            scanf("%d",&temp1);
            printf("          * Stop at : " );
            scanf("%d",&temp2);
            if(temp1<=0||temp2>=32700||(temp2-temp1)>10||temp1>temp2)
            {
               closegraph();
               goto temp_graph;
            }
            else
            {
               closegraph();
               goto graph_2;
            }
         }
         else if(back==3)
         {
            closegraph();
            goto graph_3;
         }
      }
      else
      {
         closegraph();
         goto temp_graph;
      }

graph:    initgraph(&gdriver,&gmode,"c:\\tc");

      window_a_channel();
      plot_a_channel();

      gotoxy(5,23);
      printf("\n\n    * Plot graph by points : Max. 500 points !(Y/N) : ");
      scanf("%s",&condition);
      if(condition=='Y'||condition=='y')
      {
         printf("     * Start at points : ");
         scanf("%d",&start);
         printf("     * Finish at points : ");
         scanf("%d",&finish);
         if(start<=0||finish>32766||start>finish||(finish-start)>500)
         {
            closegraph();
            goto graph,
         }
         else
         {
            closegraph();
            goto graph_1;
         }
      }
      else if(condition=='N'||condition=='n')
      {
         printf("         * MENU :  1. Back  ====> Press 1 "
                "\n                   2. Next  ====> Press 2 "
                "\n                   3. Exit  ====> Press 3 ");
         printf("\n                    * Press : ");
         scanf("%d",&back);
         if(back==1)
         {
            closegraph();
            goto cha_nel;
         }
         else if(back==2)
         {
            printf("        * Range of temperature value maximum 10 points ");
            printf("\n                * Start at : " );
            scanf("%d",&temp1);
            printf("          * Stop at : " );
            scanf("%d",&temp2);
            if(temp1<=0||temp2>=32700||(temp2-temp1)>10||temp1>temp2)
            {
               closegraph();
               goto graph;
            }
            else
            {
               closegraph();
               goto graph_2;
            }
```

```
        }
        else if(back==3)
        {
           closegraph();
           goto graph_3;
        }
      }
      else
      {
        closegraph();
        goto graph;
      }

temp_graph_1:    initgraph(&gdriver,&gmode,"c:\\tc");

       t=0;

check_code_2:  if(data3[t]==0x01||data3[t]==0x02||data3[t]==0x03||
                  data3[t]==0x04||data3[t]==0x05||data3[t]==0x06||
                  data3[t]==0x07||data3[t]==0x08||data3[t]==0x09||
                  data3[t]==0x0A||data3[t]==0x0B||data3[t]==0x0C||
                  data3[t]==0x0D||data3[t]==0x0E||data3[t]==0x0F)
        {
        t=t++;
        goto ting_2;
        }
       else
        {
        t=t++;
        goto check_code_2;
        }

ting_2:   gotoxy(25,2);
     printf(" Date/Month/Year      Hours/Minutes/Seconds ");
     gotoxy(25,3);
     printf("  %c%c    %c%c     %c%c        %c%c      %c%c       %c%c    "
            ,data3[t+1],data3[t+2],data3[t+3],data3[t+4],data3[t+5],data3[t+6]
            ,data3[t+7],data3[t+8],data3[t+9],data3[t+10],data3[t+11],data3[t+12]);
     gotoxy(25,4);
     printf(" Sampling time : Hours/Minutes/Seconds ");
     gotoxy(25,5);
     printf("             %c%c      %c%c       %c%c   ",data3[t+13],data3[t+14]
            ,data3[t+15],data3[t+16],data3[t+17],data3[t+18]);

     window_temp_channel();

      gotoxy(8,24);
      printf("%d",start);
      gotoxy(70,24);
      printf(" %d ",(start+500));//x8+data4);
      gotoxy(39,24);
      printf(" %d ",(start+250));//(x8+data4)/2);

     setcolor(9);
       if(data3[t]==0x01)
       {
          x9=x8;
          moveto(x9,y11-data3[(t+first_data+start)]);
          for(j=(t+first_data+start);j<=(t+first_data+finish);j++)
          {
            if(x9<=x4)
            {
            lineto(x9,y11-data3[j]);
            x9=x9++;
            }
          }
       }
       else if(data3[t]==0x03||data3[t]==0x05||data3[t]==0x09)
       {
          x9=x8;
          moveto(x9,y11-data3[(t+first_data+(2*start)-2)]);
          for(j=(t+first_data+(2*start)-2);j<=(t+first_data+(2*finish)-2);j++)
          {
            if(x9<=x4)
            {
            lineto(x9,y11-data3[j]);
            x9=x9++;
            j=j+1;
            }
          }
```

```
        }
        else if(data3[t]==0x07||data3[t]==0x08||data3[t]==0x0D)
        {
          x9=x8;
          moveto(x9,y11-data3[(t+first_data+(3*start)-3)]);
          for(j=(t+first_data+(3*start)-3);j<=(t+first_data+(3*finish)-3);j++)
          {
            if(x9<=x4)
            {
            lineto(x9,y11-data3[j]);
            x9=x9++;
            j=j+2;
            }
          }
        }
        else if(data3[t]==0x0F)
        {
          x9=x8;
          moveto(x9,y11-data3[(t+first_data+(4*start)-4)]);
          for(j=(t+first_data+(4*start)-4);j<=(t+first_data+(4*finish)-4);j++)
          {
            if(x9<=x4)
            {
            lineto(x9,y11-data3[j]);
            x9=x9++;
            j=j+3;
            }
          }
        }
        else
        {
        }
        setcolor(15);

      printf("\n          * MENU :  1. Back     ===> Press 1 "
        "\n                          2. Next     ===> Press 2 "
        "\n                3. Exit     ===> Press 3 ");
      printf("\n          * Press : ");
      scanf("%d",&back);

      if(back==1)
      {
        closegraph();
        goto temp_graph;
      }
      else if(back==2)
      {
        printf("        * Range of temperature value maximum 10 points ");
        printf("\n            * Start at : " );
        scanf("%d",&temp1);
        printf("          * Stop at : " );
        scanf("%d",&temp2);
        if(temp1<=0||temp2>=32700||(temp2-temp1)>10||temp1>temp2)
        {
          closegraph();
          goto temp_graph_1;
        }
        else
        {
          closegraph();
          goto graph_2;
        }
      }
      else if(back==3)
      {
        closegraph();
        goto graph_3;
      }
      else
      {
        closegraph();
        goto temp_graph_1;
      }

graph_1:  initgraph(&gdriver,&gmode,"c:\\tc");
      t=0;
check_code_3:  if(data3[t]==0x01||data3[t]==0x02||data3[t]==0x03||
                  data3[t]==0x04||data3[t]==0x05||data3[t]==0x06||
                  data3[t]==0x07||data3[t]==0x08||data3[t]==0x09||
                  data3[t]==0x0A||data3[t]==0x0B||data3[t]==0x0C||
```

```
                data3[t]==0x0D||data3[t]==0x0E||data3[t]==0x0F)
        {
        t=t++;
        goto ting_3;
        }
       else
        {
        t=t++;
        goto check_code_3;
        }

ting_3:   gotoxy(25,2);
     printf(" Date/Month/Year:     Hours/Minutes/Sec only ");
     gotoxy(25,3);
     printf("  %c%c    %c%c    %c%c       %c%c      %c%c      %c%c   ",
              data3[t+1],data3[t+2],data3[t+3],data3[t+4],data3[t+5],data3[t+6],
              data3[t+7],data3[t+8],data3[t+9],data3[t+10],data3[t+11],data3[t+12]);
     gotoxy(25,4);
     printf(" Sampling time : Hours/Minutes/Sec only ");
     gotoxy(25,5);
     printf("           %c%c     %c%c      %c%c   ",data3[t+13],data3[t+14],data3[t+15]
,
                        data3[t+16],data3[t+17],data3[t+18]);
     gotoxy(5,24);

      window_a_channel();

      gotoxy(8,24);
      printf("%d",start);
      gotoxy(70,24);
      printf(" %d ",(start+500));//x8+data4);
      gotoxy(39,24);
      printf(" %d ",(start+250));//(x8+data4)/2);

     if(channel==2)
     {
       setcolor(10);
       if(data3[t]==0x02)
       {
         x9=x8;
         moveto(x9,y11-data3[(t+first_data+start-1)]);
         for(j=(t+first_data+start-1);j<=(t+first_data+finish-1);j++)
         {
           if(x9<=x4)
           {
           lineto(x9,y11-data3[j]);
           x9=x9++;
           }
         }
       }
       else if(data3[t]==0x03)
       {
         x9=x8;
         moveto(x9,y11-data3[(t+first_data+(2*start)-1)]);
         for(j=(t+first_data+(2*start)-1);j<=(t+first_data+(2*finish)-1);j++)
         {
           if(x9<=x4)
           {
           lineto(x9,y11-data3[j]);
           x9=x9++;
           j=j+1;
           }
         }
       }
       else if(data3[t]==0x06||data3[t]==0x0A)
       {
         x9=x8;
         moveto(x9,y11-data3[(t+first_data+(2*start)-2)]);
         for(j=(t+first_data+(2*start)-2);j<=(t+first_data+(2*finish)-2);j++)
         {
           if(x9<=x4)
           {
           lineto(x9,y11-data3[j]);
           x9=x9++;
           j=j+1;
           }
         }
       }
       else if(data3[t]==0x07||data3[t]==0x0B)
       {
```

```
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(3*start)-2)]);
      for(j=(t+first_data+(3*start)-2);j<=(t+first_data+(3*finish)-2);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0E)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(3*start)-3)]);
      for(j=(t+first_data+(3*start)-3);j<=(t+first_data+(3*finish)-3);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(4*start)-3)]);
      for(j=(t+first_data+(4*start)-3);j<=(t+first_data+(4*finish)-3);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
  }
  if(channel==3)
  {
    setcolor(11);
    if(data3[t]==0x04)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+start-1)]);
      for(j=(t+first_data+start-1);j<=(t+first_data+finish-1);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        }
      }
    }
    else if(data3[t]==0x0C)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(2*start)-2)]);
      for(j=(t+first_data+(2*start)-2);j<=(t+first_data+(2*finish)-2);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x05||data3[t]==0x06)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(2*start)-1)]);
      for(j=(t+first_data+(2*start)-1);j<=(t+first_data+(2*finish)-1);j++)
```

```
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x0D||data3[t]==0x0E)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(3*start)-2)]);
      for(j=(t+first_data+(3*start)-2);j<=(t+first_data+(3*finish)-2);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x07)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(3*start)-1)]);
      for(j=(t+first_data+(3*start)-1);j<=(t+first_data+(3*finish)-1);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(4*start)-2)]);
      for(j=(t+first_data+(4*start)-2);j<=(t+first_data+(4*finish)-2);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
  }
  if(channel==4)
  {
    setcolor(11);
    if(data3[t]==0x08)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+start)]);
      for(j=(t+first_data+start);j<=(t+first_data+finish);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        }
      }
    }
    else if(data3[t]==0x09||data3[t]==0x0A||data3[t]==0x0C)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(2*start)-1)]);
      for(j=(t+first_data+(2*start)-1);j<=(t+first_data+(2*finish)-1);j++)
      {
        if(x9<=x4)
        {
```

```
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x0B||data3[t]==0x0D||data3[t]==0x0E)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(3*start)-1)]);
      for(j=(t+first_data+(3*start)-1);j<=(t+first_data+(3*finish)-1);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8;
      moveto(x9,y11-data3[(t+first_data+(4*start)-1)]);
      for(j=(t+first_data+(4*start)-1);j<=(t+first_data+(4*finish)-1);j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
}
else
{
}
printf("\n          * MENU :  1. Back     ====> Press 1 "
   "\n                    2. Next     ====> Press 2 "
   "\n                3. Exit     ====> Press 3 ");
printf("\n          * Press : ");
scanf("%d",&back);

if(back==1)
{
   closegraph();
   goto graph;
}
else if(back==2)
{
   printf("       * Range of temperature value maximum 10 points ");
   printf("\n            * Start at : " );
   scanf("%d",&temp1);
   printf("          * Stop at : " );
   scanf("%d",&temp2);
   if(temp1<=0||temp2>=32700||(temp2-temp1)>10||temp1>temp2)
   {
      closegraph();
      goto graph_1;
   }
   else
   {
      closegraph();
      goto graph_2;
   }
}
else if(back==3)
{
   closegraph();
   goto graph_3;
}
else
{
   closegraph();
   goto graph_1;
```

```
    }

graph_2:  initgraph(&gdriver,&gmode,"c:\\tc");
     t=0;
     printf("\n  *-------------------------------------------------------------------------------* "
            "\n  *       Table at points                                         * "
            "\n  *-------------------------------------------------------------------------------* ");
check_code_4:  if(data3[t]==0x01||data3[t]==0x02||data3[t]==0x03||
                  data3[t]==0x04||data3[t]==0x05||data3[t]==0x06||
                  data3[t]==0x07||data3[t]==0x08||data3[t]==0x09||
                  data3[t]==0x0A||data3[t]==0x0B||data3[t]==0x0C||
                  data3[t]==0x0D||data3[t]==0x0E||data3[t]==0x0F)
       {
       t=t++;
       goto ting_4;
       }
      else
       {
       t=t++;
       goto check_code_4;
       }

ting_4:   if(channel==1)
     {
       if(data3[t]==0x01)
       {
          for(j=(t+19+temp1-1);j<=(t+19+temp2-1);j++)
          {
            printf("\n          -   Point      Celsius ");
            printf("\n        %d              %d ",temp1,data3[j]*60/256);
            temp1++;
          }
       }
       else if(data3[t]==0x03||data3[t]==0x05||data3[t]==0x09)
       {
          for(j=(t+19+2*temp1-2);j<=(t+19+2*temp2-1);j++)
          {
            printf("\n          -   Point      Celsius ");
            printf("\n        %d              %d ",temp1,(data3[j]*60)/256);
            temp1++;
            j=j+1;
          }
       }
       else if(data3[t]==0x07||data3[t]==0x0B||data3[t]==0x0D)
       {
          for(j=(t+19+3*temp1-3);j<=(t+19+3*temp2-1);j++)
          {
            printf("\n          -   Point      Celsius ");
            printf("\n        %d              %d ",temp1,(data3[j]*60)/256);
            temp1++;
            j=j+2;
          }
       }
       else if(data3[t]==0x0F)
       {
          for(j=(t+19+(4*temp1)-4);j<=(t+19+(4*temp2)-4);j++)
          {
            printf("\n          -   Point      Celsius ");
            printf("\n        %d              %d ",temp1,(data3[j]*60)/256);
            temp1++;
            j=j+3;
          }
       }
       else
       {
       }
       setcolor(15);
     }
     else if(channel==2)
     {
       if(data3[t]==0x02)
       {
          for(j=(t+19+temp1-1);j<=(t+19+temp2-1);j++)
          {
            printf("\n          -   Point      Level ");
            printf("\n        %2d               %3d ",temp1,data3[j]);
            temp1++;
          }
       }
       else if(data3[t]==0x03)
```

```
    {
      for(j=(t+19+2*temp1-1);j<=(t+19+2*temp2);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d               %2d ",temp1,data3[j]);          temp1+
+;
        j=j+1;
      }
    }
    else if(data3[t]==0x06||data3[t]==0x0A)
    {
      for(j=(t+19+2*temp1-2);j<=(t+19+2*temp2-1);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d               %2d ",temp1,data3[j]);
        temp1++;
        j=j+1;
      }
    }
    else if(data3[t]==0x07||data3[t]==0x0B)
    {
      for(j=(t+19+3*temp1-2);j<=(t+19+3*temp2);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d             %2d ",temp1,data3[j]);
        temp1++;
        j=j+2;
      }
    }
    else if(data3[t]==0x0E)
    {
      for(j=(t+19+3*temp1-3);j<=(t+19+3*temp2-1);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d             %2d ",temp1,data3[j]);
        temp1++;
        j=j+2;
      }
    }
    else if(data3[t]==0x0F)
    {
      for(j=(t+19+(4*temp1)-3);j<=(t+19+(4*temp2)-3);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d             %2d ",temp1,data3[j]);
        temp1++;
        j=j+3;
      }
    }
    else
    {
    }
    setcolor(15);
  }
  if(channel==3)
  {
    if(data3[t]==0x04)
    {
      for(j=(t+19+temp1-1);j<=(t+19+temp2-1);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d             %2d ",temp1,data3[j]);
        temp1++;
      }
    }
    else if(data3[t]==0x0C)
    {
      for(j=(t+19+2*temp1-2);j<=(t+19+2*temp2-1);j++)
      {
        printf("\n          -   Point      Level ");
        printf("\n        %2d             %2d ",temp1,data3[j]);
        temp1++;
        j=j+1;
      }
    }
    else if(data3[t]==0x05||data3[t]==0x06)
    {
      for(j=(t+19+2*temp1-1);j<=(t+19+2*temp2);j++)
      {
        printf("\n          -   Point      Level ");
```

```
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+3;
    }
  }
  else if(data3[t]==0x0D||data3[t]==0x0E)
  {
    for(j=(t+19+3*temp1-2);j<=(t+19+3*temp2);j++)
    {
      printf("\n            -   Point      Level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+2;
    }
  }
  else if(data3[t]==0x07)
  {
    for(j=(t+19+3*temp1-1);j<=(t+19+3*temp2+1);j++)
    {
      printf("\n            -   Point      Level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+2;
    }
  }
  else if(data3[t]==0x0F)
  {
    for(j=(t+19+(4*temp1)-2);j<=(t+19+(4*temp2)-2);j++)
    {
      printf("\n            -   Point      Level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+3;
    }
  }
  else
  {
  }
  setcolor(15);
}
if(channel==4)
{
  if(data3[t]==0x08)
  {
    for(j=(t+19+temp1-1);j<=(t+19+temp2-1);j++)
    {
      printf("\n            -   Point      level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
    }
  }
  else if(data3[t]==0x09||data3[t]==0x0A||data3[t]==0x0C)
  {
    for(j=(t+19+2*temp1-1);j<=(t+19+2*temp2);j++)
    {
      printf("\n            -   Point      level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+1;
    }
  }
  else if(data3[t]==0x0B||data3[t]==0x0D||data3[t]==0x0E)
  {
    for(j=(t+19+3*temp1-1);j<=(t+19+3*temp2+1);j++)
    {
      printf("\n            -   Point      level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+2;
    }
  }
  else if(data3[t]==0x0F)
  {
    for(j=(t+19+(4*temp1)-1);j<=(t+19+(4*temp2)-1);j++)
    {
      printf("\n            -   Point      level ");
      printf("\n          %2d              %2d ",temp1,data3[j]);
      temp1++;
      j=j+3;
    }
```

```
        }
        else
        {
        }
        setcolor(15);
      }
      else
      {
      }
      printf("\n\n          * MENU :  1. Back          ===> Press 1 "
        "\n        2. Main window ===> Press 2 "
        "\n                          3. Exit        ===> Press 3 ");
      printf("\n         * Press : ");
      scanf("%d",&back);

      if(back==1)
      {
        if(channel==1)
        {
          closegraph();
          goto temp_graph;
        }
        else
        {
          closegraph();
          goto graph;
        }
      }
      else if(back==2)
      {
        closegraph();
        goto win;
      }
      else if(back==3)
      {
        closegraph();
        goto graph_3;
      }
      else
      {
        closegraph();
        goto graph_1;
      }

graph_3:        closegraph();

}

//=============================================================================
// New File
//=============================================================================
unsigned new_file()
{
   int  x_axis=25,y_axis=19;

      initgraph(&gdriver,&gmode,"c:\\tc");
      bioscom( 0 , SETTINGS , 0 );
      setcolor(13);
      settextstyle(BOLD_FONT,HORIZ_DIR,1);
      outtextxy(230,100," New File ");
      gotoxy(25,11);
      printf(" Save File Name : ");
      gets(file_name);
      printf("\n\n");
      if((buffer=fopen(file_name,"w+"))==NULL)
      {
        gotoxy(20,13);
        fprintf(stderr,"Opening file error \n");
        exit(1);
      }
      gotoxy(25,15);
      printf("   Waiting for receive data " );
      setcolor(14);
      rectangle(110,200,510,370);
      setcolor(13);
      rectangle(190,280,435,310);

      for(i=0;i<T;i++)
      {
        data3[i]=readcom();
```

```
        if(i==0)
        {
        gotoxy(25,15);
        printf("       Receiving data      " );
        }
        if(i==input)
        {
        count=count+3;
        gotoxy(38,17);
        printf("%d %",(count*100)/90);
        gotoxy(x_axis,y_axis);
        printf("\xB0");
        x_axis=x_axis+1;
        input=input+910;
        }
      }
      sound(1000);
      gotoxy(25,15);
      printf("                          " );
      gotoxy(28,17);
      printf(" Up Load Data Complete     ");
      delay(1000);
      delay(1000);
      nosound();

      fwrite(data3,sizeof(data3),1,buffer);
      if(ferror(buffer))
      {
         gotoxy(20,14);
         printf("Can not writing file \n");
         exit(1);
      }
      fclose(buffer);
      closegraph();
      return(data3[i]);
}

//=============================================================================
// Open File
//=============================================================================
unsigned open_file()
{
   struct  ftime filet;
   struct  ffblk block;
   int  ch,flag,y_plot;
   char drive_path_name[70];

open:     initgraph(&gdriver,&gmode,"c:\\tc");
      setcolor(13);
      y_plot=14;
      settextstyle(BOLD_FONT,HORIZ_DIR,1);
      outtextxy(230,100," Open File ");
      setcolor(15);
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
      gotoxy(20,11);
      printf("      Find file name : ");
      gets(drive_path_name);
      gotoxy(20,13);
      printf("        File name          Size(byte)   ");
      //gotoxy(20,28);
      flag = findfirst(drive_path_name,&block,32);
      while(!flag)
      {
         gotoxy(28,y_plot);
         printf("%s",block.ff_name);
         gotoxy(50,y_plot);
         printf("%7ld",block.ff_fsize);
         flag = findnext(&block);
         y_plot++;
         if(y_plot==24)
         {
            outtextxy(200,400,"  Press any key to continue ");
            getch();
            y_plot=14;
         }
         //gotoxy(20,29);
      }
      setcolor(0);
      outtextxy(200,400,"  Press any key to continue ");
```

```
    setcolor(15);
    outtextxy(200,400,"  Press <ESC> = Find again");
    outtextxy(200,420,"       <ENTER> = Open file");
    ch=getch();
    setcolor(0);
    outtextxy(200,400,"  Press <ESC> = Find again");
    outtextxy(200,420,"       <ENTER> = Open file");

    if(ch==0x1b)
    {
       closegraph();
       goto open;
    }

    gotoxy(20,25);
    printf("      File Name : ");
    gets(file_name);

    if((buffer=fopen(file_name,"r+"))==NULL)
    {
       gotoxy(20,27);
       fprintf(stderr,"Opening file %s error \n",file_name);
       exit(1);
    }
    fread(data3,sizeof(data3),1,buffer);
    if(ferror(buffer))
    {
       gotoxy(20,27);
       printf("Can not reading file \n");
       exit(1);
    }
    fclose(buffer);
    if(ferror(buffer) != NULL)
    {
       gotoxy(20,27);
       fprintf(stderr,"File %s can not close \n " , file_name);
       exit(1);
    }
    closegraph();
    return(data3[i]);
}

//=============================================================
// Window 4 channel
//=============================================================
void window_channel()
{
   int  x1=20,x2=270,x3=320,x4=570,x5=90,x7=390;
   int  y1=15,y2=165,y3=185,y4=335,y5=5,y6=175;

      outtextxy(x5+30,y5," CHANNEL 1 ");
      outtextxy(x7+30,y5," CHANNEL 2 ");
      outtextxy(x5+30,y6," CHANNEL 3 ");
      outtextxy(x7+30,y6," CHANNEL 4 ");
      outtextxy(x1+55,y1+20,"(Temp.)");
      outtextxy(x2-17,y2-10,"(Time) ");
      outtextxy(x3+55,y1+20,"(Level)");
      outtextxy(x4-17,y2-10,"(Time) ");
      outtextxy(x1+55,y3+20,"(Level)");
      outtextxy(x2-17,y4-10,"(Time) ");
      outtextxy(x3+55,y3+20,"(Level)");
      outtextxy(x4-17,y4-10,"(Time) ");

      setcolor(14);                                  //CHANNEL 1
      rectangle(x1+30,y1,x2+30,y2);
      line(x1+50,y2-20,x2+10,y2-20);          // set row
      moveto(x2+5,y2-25);
      lineto(x2+10,y2-20);
      lineto(x2+5,y2-15);
      line(x1+50,y2-20,x1+50,y2-130);         // set collum
      moveto(x1+45,y1+25);
      lineto(x1+50,y1+20);
      lineto(x1+55,y1+25);

      rectangle(x3+30,y1,x4+30,y2);                  //CHANNEL 2
      line(x3+50,y2-20,x4+10,y2-20);          // set row
      moveto(x4+5,y2-25);
      lineto(x4+10,y2-20);
      lineto(x4+5,y2-15);
      line(x3+50,y2-20,x3+50,y2-130);         // set collum
```

```
    moveto(x3+45,y1+25);
    lineto(x3+50,y1+20);
    lineto(x3+55,y1+25);

    rectangle(x1+30,y3,x2+30,y4);                    //CHANNEL 3
    line(x1+50,y4-20,x2+10,y4-20);                   // set row
    moveto(x2+5,y4-25);
    lineto(x2+10,y4-20);
    lineto(x2+5,y4-15);
    line(x1+50,y4-20,x1+50,y4-130);                  // set collum
    moveto(x1+45,y3+25);
    lineto(x1+50,y3+20);
    lineto(x1+55,y3+25);

    rectangle(x3+30,y3,x4+30,y4);                    //CHANNEL 4
    line(x3+50,y4-20,x4+10,y4-20);                   // set row
    moveto(x4+5,y4-25);
    lineto(x4+10,y4-20);
    lineto(x4+5,y4-15);
    line(x3+50,y4-20,x3+50,y4-130);                  // set collum
    moveto(x3+45,y3+25);
    lineto(x3+50,y3+20);
    lineto(x3+55,y3+25);
    setcolor(15);
}

//=============================================================================
// Plot graph in window channel
//=============================================================================
void plot_4channel()
{
   int  x1=20,x2=270,x3=320,x4=570;
   int  y2=165,y4=335,t=0;

check_code:  if(data3[t]==0x01||data3[t]==0x02||data3[t]==0x03||
                data3[t]==0x04||data3[t]==0x05||data3[t]==0x06||
                data3[t]==0x07||data3[t]==0x08||data3[t]==0x09||
                data3[t]==0x0A||data3[t]==0x0B||data3[t]==0x0C||
                data3[t]==0x0D||data3[t]==0x0E||data3[t]==0x0F)
        {
        t=t++;
        goto ting;
        }
       else
        {
        t=t++;
        goto check_code;
        }

 ting:     setcolor(12);
      gotoxy(8,23);
      printf("\n");
      outtextxy(100,370,"  This is the first time for start get data ");
      printf("\n      Date/Munth/Year      Hours/Minutes/Seconds ");
      printf("\n       %c%c    %c%c    %c%c        %c%c      %c%c    "
             ,data3[t+1],data3[t+2],data3[t+3],data3[t+4],data3[t+5],data3[t+6]
             ,data3[t+7],data3[t+8],data3[t+9],data3[t+10],data3[t+11],data3[t+12]);
      printf("\n          * Sampling time : Hours/Minutes/Seconds ");
      printf("\n            %c%c      %c%c      %c%c   "
             ,data3[t+13],data3[t+14],data3[t+15],data3[t+16],data3[t+17],data3[t+18])
 ;

      if(data3[t]==0x01)          //Channel 1
      {
         setcolor(9);
         moveto(x1+50,y2-20);          // plot graph CHANNEL 1
         data1=0;
         for(i=t+19;i<T;i++)
         {
           if((x1+50+data1/3)<(x2+25))
           {
              lineto(x1+50+data1/3,y2-20-data3[i]/3);
              data1++;
           }
         }
          setcolor(15);
         }
         else if(data3[t]==0x02)                    //Channel 2
         {
            setcolor(10);
```

```
   moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
   data1=0;
   for(i=t+19;i<T;i++)
   {
     if((x3+50+data1/3)<(x4-5))
     {
     lineto(x3+50+data1/3,y2-20-data3[i]/3);
     data1++;
     }
   }
   setcolor(15);
}
else if(data3[t]==0x04)     //Channel 3
{
   setcolor(11);
   moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
   data1=0;
   for(i=t+19;i<T;i++)
   {
     if((x1+50+data1/3)<(x2-5))
     {
     lineto(x1+50+data1/3,y4-20-data3[i]/3);
     data1++;
     }
   }
   setcolor(15);
}
else if(data3[t]==0x08)          //Channel 4
{
   setcolor(12);
   moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
   data1=0;
   for(i=t+19;i<T;i++)
   {
     if((x3+50+data1/3)<(x4-5))
     {
     lineto(x3+50+data1/3,y4-20-data3[i]/3);
     data1++;
     }
   }
   setcolor(15);
}
else if(data3[t]==0x03)  //Channel 1
{
   setcolor(9);
   moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
   data1=0;
   for(i=t+19;i<T;i++)
   {
     if((x1+50+data1/3)<(x2-5))
     {
     lineto(x1+50+data1/3,y2-20-data3[i]/3);
     data1++;
     i=i+1;
     }
   }
   setcolor(15);

   setcolor(10);
   moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
   data1=0;
   for(i=t+20;i<T;i++)
   {
     if((x3+50+data1/3)<(x4-5))
     {
     lineto(x3+50+data1/3,y2-20-data3[i]/3);
     data1++;
     i=i+1;
     }
   }
   setcolor(15);
}
else if(data3[t]==0x05)
{
   setcolor(9);
   moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
   data1=0;
   for(i=t+19;i<T;i++)
   {
     if((x1+50+data1/3)<(x2-5))
```

```
      {
      lineto(x1+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);

    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x06)
{
    setcolor(10);
    moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);

    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x09)
{
    setcolor(9);
    moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
```

```
    setcolor(15);
  }
  else if(data3[t]==0x0A)
  {
    setcolor(10);
    moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);
  }
  else if(data3[t]==0x0C)
  {
    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+1;
      }
    }
    setcolor(15);
  }
  else if(data3[t]==0x07)
  {
    setcolor(9);
    moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(10);
    moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
```

```
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+21;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x0B)
{
    setcolor(9);
    moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(10);
    moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+21;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x0D)
{
    setcolor(9);
    moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y2-20-data3[i]/3);
```

```
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+21;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x0E)
{
    setcolor(10);
    moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
    data1=0;
    for(i=t+19;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y2-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(11);
    moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
    data1=0;
    for(i=t+20;i<T;i++)
    {
      if((x1+50+data1/3)<(x2-5))
      {
      lineto(x1+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);

    setcolor(12);
    moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
    data1=0;
    for(i=t+21;i<T;i++)
    {
      if((x3+50+data1/3)<(x4-5))
      {
      lineto(x3+50+data1/3,y4-20-data3[i]/3);
      data1++;
      i=i+2;
      }
    }
    setcolor(15);
}
else if(data3[t]==0x0F)
{
```

```
            setcolor(9);
            moveto(x1+50,y2-20);        // Plot graph CHANNEL 1
            data1=0;
            for(i=t+19;i<T;i++)
            {
              if((x1+50+data1/3)<(x2-5))
              {
              lineto(x1+50+data1/3,y2-20-data3[i]/3);
              data1++;
              i=i+3;
              }
            }
            setcolor(15);

            setcolor(10);
            moveto(x3+50,y2-20);        // Plot graph CHANNEL 2
            data1=0;
            for(i=t+20;i<T;i++)
            {
              if((x3+50+data1/3)<(x4-5))
              {
              lineto(x3+50+data1/3,y2-20-data3[i]/3);
              data1++;
              i=i+3;
              }
            }
            setcolor(15);

            setcolor(11);
            moveto(x1+50,y4-20);        // Plot graph CHANNEL 3
            data1=0;
            for(i=t+21;i<T;i++)
            {
              if((x1+50+data1/3)<(x2-5))
              {
              lineto(x1+50+data1/3,y4-20-data3[i]/3);
              data1++;
              i=i+3;
              }
            }
            setcolor(15);

            setcolor(12);
            moveto(x3+50,y4-20);        // Plot graph CHANNEL 4
            data1=0;
            for(i=t+22;i<T;i++)
            {
              if((x3+50+data1/3)<(x4-5))
              {
              lineto(x3+50+data1/3,y4-20-data3[i]/3);
              data1++;
              i=i+3;
              }
              else
              {
              }
            }
            setcolor(15);
          }
          else
          {
          }
}
//=============================================================================
// Window Temperature channel
//=============================================================================
void window_temp_channel()
{
   int  x4=570,x8=70,x9;
   int  y11=355,y12=55;

      setcolor(14);
      line(x8,y11,x4+10,y11);             //row
      moveto(x4+5,y11-5);
      lineto(x4+10,y11);
      lineto(x4+5,y11+5);

      moveto(x4,y11);
      lineto(x4,y11+5);
      for(k=1;k<10;k++)
```

```
    {
    moveto(x8+((x4-x8)/10)*k,y11);
    lineto(x8+((x4-x8)/10)*k,y11+5);
    }

    line(x8,y11,x8,y12);              //colmn
    moveto(x8-5,y12+5);               // set symbol ( > )
    lineto(x8,y12);
    lineto(x8+5,y12+5);

    moveto(x8,y12+46);               //set range of colmn
    lineto(x8-5,y12+46);
    for(l=1;l<12;l++)
    {
    moveto(x8,y11-((y11-y12-44)/12)*l);
    lineto(x8-5,y11-((y11-y12-44)/12)*l);
    }
    setcolor(14);

    gotoxy(6,3);         // set range
    printf("Temp.(celsius) ");
    gotoxy(71,22);
    printf("Time(beat) " );
    //gotoxy(4,5);
    //printf(" 55 ");
    gotoxy(4,7);
    printf(" 60");
    gotoxy(4,15);
    printf(" 30");
    gotoxy(8,24);
    printf(" 0 ");
    gotoxy(70,24);
    printf(" %d ",data4);//x8+data4);
    gotoxy(39,24);
    printf(" %d ",data4/2);//(x8+data4)/2);

}

//============================================================================
// Window a channel
//============================================================================
void window_a_channel()
{
   int  x4=570,x8=70,x9;
   int  y11=355,y12=55;

     setcolor(14);
     line(x8,y11,x4+10,y11);              //row
     moveto(x4+5,y11-5);
     lineto(x4+10,y11);
     lineto(x4+5,y11+5);

     moveto(x4,y11);
     lineto(x4,y11+5);
     for(k=1;k<10;k++)
     {
     moveto(x8+((x4-x8)/10)*k,y11);
     lineto(x8+((x4-x8)/10)*k,y11+5);
     }

     line(x8,y11,x8,y12);              //colmn
     moveto(x8-5,y12+5);               // set symbol ( > )
     lineto(x8,y12);
     lineto(x8+5,y12+5);

     moveto(x8,y12+46);               //set range of colmn
     lineto(x8-5,y12+46);
     for(l=1;l<10;l++)
     {
     moveto(x8,y11-((y11-y12-44)/10)*l);
     lineto(x8-5,y11-((y11-y12-44)/10)*l);
     }
     setcolor(14);

     gotoxy(6,3);         // set range
     printf(" Level ");
     gotoxy(71,22);
     printf("Time(beat)" );
     gotoxy(4,6);
     printf(" 255 ");
```

```
     gotoxy(4,14);
     printf(" 12% ");
     gotoxy(8,24);
     printf(" 0 ");
     gotoxy(70,24);
     printf(" %d ",data4);
     gotoxy(39,24);
     printf(" %d ",data4/2);

}

//=============================================================================
// Plot a channel
//=============================================================================
void plot_a_channel()
{
   int  x4=570,x8=70,x9,y11=355,t=0;

check_code_1:  if(data3[t]==0x01||data3[t]==0x02||data3[t]==0x03||
                  data3[t]==0x04||data3[t]==0x05||data3[t]==0x06||
                  data3[t]==0x07||data3[t]==0x08||data3[t]==0x09||
                  data3[t]==0x0A||data3[t]==0x0B||data3[t]==0x0C||
                  data3[t]==0x0D||data3[t]==0x0E||data3[t]==0x0F)
        {
        t=t++;
        goto ting_1;
        }
       else
        {
        t=t++;
        goto check_code_1;
        }

ting_1:   gotoxy(25,2);
     printf(" Date/Month/Year      Hours/Minutes/Seconds ");
     gotoxy(25,3);
     printf("  %c%c    %c%c    %c%c       %c%c      %c%c      %c%c   "
            ,data3[t+1],data3[t+2],data3[t+3],data3[t+4],data3[t+5],data3[t+6]
            ,data3[t+7],data3[t+8],data3[t+9],data3[t+10],data3[t+11],data3[t+12]);
     gotoxy(25,4);
     printf(" Sampling time : Hours/Minutes/Seconds ");
     gotoxy(25,5);
     printf("             %c%c     %c%c      %c%c   ",data3[t+13],data3[t+14],data3[t+1
5]
            ,data3[t+16],data3[t+17],data3[t+18]);

     if(channel==1)
     {
        setcolor(9);
        if(data3[t]==0x01)
        {
           x9=x8;
           moveto(x9,y11-data3[t+19]);
           for(j=t+19;j<T;j++)
           {
             if(x9<=x4)
             {
             lineto(x9,y11-data3[j]);
             x9=x9++;
             }
           }
        }
        else if(data3[t]==0x03||data3[t]==0x05||data3[t]==0x09)
        {
           x9=x8;
           moveto(x9,y11-data3[t+19]);
           for(j=t+19;j<T;j++)
           {
             if(x9<=x4)
             {
             lineto(x9,y11-data3[j]);
             x9=x9++;
             j=j+1;
             }
           }
        }
        else if(data3[t]==0x07||data3[t]==0x0B||data3[t]==0x0D)
        {
           x9=x8;
           moveto(x9,y11-data3[t+19]);
```

```
      for(j=t+19;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8,
      moveto(x9,y11-data3[t+19]);
      for(j=t+19;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
  }
  else if(channel==2)
  {
    setcolor(10);
    if(data3[t]==0x02)
    {
      x9=x8;
      moveto(x9,y11-data3[t+19]);
      for(j=t+19;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        }
      }
    }
    else if(data3[t]==0x03)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x06||data3[t]==0x0A)
    {
      x9=x8;
      moveto(x9,y11-data3[t+19]);
      for(j=t+19;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x07||data3[t]==0x0B)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
```

```
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x05)
    {
      x9=x8;
      moveto(x9,y11-data3[t+10]);
      for(j=t+10;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
}
if(channel==3)
{
    setcolor(11);
    if(data3[t]==0x04)
    {
      x9=x8;
      moveto(x9,y11-data3[t+10]);
      for(j=t+10;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        }
      }
    }
    else if(data3[t]==0x06)
    {
      x9=x8;
      moveto(x9,y11-data3[t+10]);
      for(j=t+10;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x05||data3[t]==0x06)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
```

```
        j=j+1;
        }
      }
    }
    else if(data3[t]==0x0D||data3[t]==0x0B)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x07)
    {
      x9=x8;
      moveto(x9,y11-data3[t+21]);
      for(j=t+21;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+2;
        }
      }
    }
    else if(data3[t]==0x0F)
    {
      x9=x8;
      moveto(x9,y11-data3[t+21]);
      for(j=t+21;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+3;
        }
      }
    }
    else
    {
    }
    setcolor(15);
  }
  if(channel==4)
  {
    setcolor(12);
    if(data3[t]==0x08)
    {
      x9=x8;
      moveto(x9,y11-data3[t+19]);
      for(j=t+19;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        }
      }
    }
    else if(data3[t]==0x03)
    {
      x9=x8;
      moveto(x9,y11-data3[t+20]);
      for(j=t+20;j<T;j++)
      {
        if(x9<=x4)
        {
        lineto(x9,y11-data3[j]);
        x9=x9++;
        j=j+1;
        }
      }
```

```
        }
        else if(data3[t]==0x09||data3[t]==0x0A||data3[t]==0x0C)
        {
           x9=x8;
           moveto(x9,y11-data3[t+20]);
           for(j=t+20;j<T;j++)
           {
             if(x9<=x4)
             {
             lineto(x9,y11-data3[j]);
             x9=x9++;
             j=j+1;
             }
           }
        }
        else if(data3[t]==0x0B||data3[t]==0x0D||data3[t]==0x0E)
        {
           x9=x8;
           moveto(x9,y11-data3[t+21]);
           for(j=t+21;j<T;j++)
           {
             if(x9<=x4)
             {
             lineto(x9,y11-data3[j]);
             x9=x9++;
             j=j+2;
             }
           }
        }
        else if(data3[t]==0x0F)
        {
           x9=x8;
           moveto(x9,y11-data3[t+22]);
           for(j=t+22;j<T;j++)
           {
             if(x9<=x4)
             {
             lineto(x9,y11-data3[j]);
             x9=x9++;
             j=j+3;
             }
           }
        }
        else
        {
        }
        setcolor(15);
     }
     else
     {
     }
}
```

ภาคผนวก จ.Data Sheet

HM62256B Series

256k SRAM (32-kword × 8-bit)

HITACHI

ADE-203-135F (Z)
Rev. 6.0
Nov. 13, 1997

Description

The Hitachi HM62256B Series is a CMOS static RAM organized 32,768-word × 8-bit. It realizes higher performance and low power consumption by employing 0.8 µm Hi-CMOS process technology. The device, packaged in 8 × 14 mm TSOP, 8 × 13.4 mm TSOP with thickness of 1.2 mm, 450 mil SOP (foot print pitch width), 600 mil plastic DIP, or 300 mil plastic DIP, is available for high density mounting. It offers low power standby power dissipation; therefore, it is suitable for battery backup systems.

Features

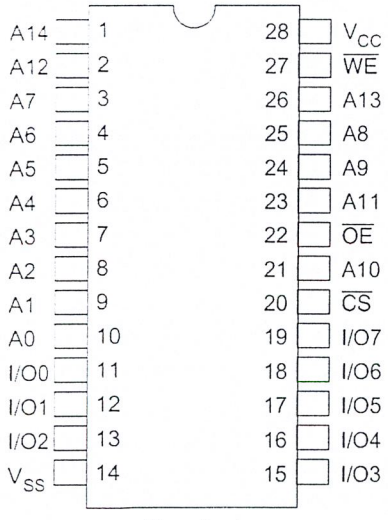
- Single 5.0 V supply: 5.0 V ± 10%
- Access time: 55 ns/70 ns/85 ns (max)
- Power dissipation:
  - — Active: 25 mW (typ) (f = 1 MHz)
  - — Standby: 1.0 µW (typ)
- Completely static memory
  - — No clock or timing strobe required
- Equal access and cycle times
- Common data input and output
  - — Three state output
- Directly TTL compatible all inputs and outputs
- Battery backup operation

Ordering Information

Type No.	Access time	Package
HM62256BLP-7	70 ns	600-mil 28-pin plastic DIP (DP-28)
HM62256BLP-7SL	70 ns	
HM62256BLSP-7	70 ns	300-mil 28-pin plastic DIP (DP-28NA)
HM62256BLSP-7SL	70 ns	
HM62256BLFP-7T	70 ns	450-mil 28-pin plastic SOP (FP-28DA)
HM62256BLFP-5SLT	55 ns	
HM62256BLFP-7SLT	70 ns	
HM62256BLFP-7ULT	70 ns	
HM62256BLT-8	85 ns	8 mm × 14 mm 32-pin TSOP (TFP-32DA)
HM62256BLT-7SL	70 ns	
HM62256BLTM-8	85 ns	8 mm × 13.4 mm 28-pin TSOP (TFP-28DA)
HM62256BLTM-5SL	55 ns	
HM62256BLTM-7SL	70 ns	
HM62256BLTM-7UL	70 ns	

Pin Arrangement

HM62256BLP/BLFP/BLSP Series



(Top view)

Operation Table

$\overline{WE}$	$\overline{CS}$	$\overline{OE}$	Mode	V_{CC} current	I/O pin	Ref. cycle
×	H	×	Standby	I_{SB}, I_{SB1}	High-Z	—
H	L	H	Output disable	I_{CC}	High-Z	—
H	L	L	Read	I_{CC}	Dout	Read cycle (1)to (3)
L	L	H	Write	I_{CC}	Din	Write cycle (1)
L	L	L	Write	I_{CC}	Din	Write cycle (2)

Note: ×: H or L

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Power supply voltage relative to V_{SS}	V_{CC}	–0.5 to +7.0	V
Terminal voltage on any pin relative to V_{SS}	V_T	–0.5*1 to V_{CC}+0.3*2	V
Power dissipation	P_T	1.0	W
Operating temperature range	Topr	0 to +70	°C
Storage temperature range	Tstg	–55 to +125	°C
Storage temperature range under bias	Tbias	–10 to +85	°C

Notes: 1. V- min: –3.0 V for pulse half-width ≤ 50 ns
2. Maximum voltage is 7.0 V

DC Operating Conditions (Ta = 0 to +70°C)

Parameter	Symbol	Min	Typ	Max	Unit	Notes
Supply voltage	V_{CC}	4.5	5.0	5.5	V	
	V_{SS}	0	0	0	V	
Input high voltage	V_{IH}	2.2	—	V_{CC} + 0.3	V	
Input low voltage	V_{IL}	–0.5*1	—	0.8	V	

Note: 1. V_{IL} min: –3.0 V for pulse half-width ≤ 50 ns

DC Characteristics (Ta = 0 to +70°C, V_{CC} = 5 V ± 10%, V_{SS} = 0 V)

Parameter		Symbol	Min	Typ*[1]	Max	Unit	Test conditions
Input leakage current		$\|I_{LI}\|$	—	—	1	µA	Vin = V_{SS} to V_{CC}
Output leakage current		$\|I_{LO}\|$	—	—	1	µA	$\overline{CS}$ = V_{IH} or $\overline{OE}$ = V_{IH} or $\overline{WE}$ = V_{IL}, $V_{I/O}$ = V_{SS} to V_{CC}
Operating current		I_{CC}	—	6	15	mA	$\overline{CS}$ = V_{IL}, Others = V_{IH}/V_{IL}, $I_{I/O}$ = 0 mA
Average operating current	HM62256B-5	I_{CC1}	—	—	60	mA	Min cycle, duty = 100%, $I_{I/O}$ = 0 mA, $\overline{CS}$ = V_{IL}, Others = V_{IH}/V_{IL}
	HM62256B-7	I_{CC1}	—	33	60	mA	
	HM62256B-8	I_{CC1}	—	29	50	mA	
		I_{CC2}	—	5	15	mA	Cycle time = 1 µs, $I_{I/O}$ = 0 mA, $\overline{CS}$ = V_{IL}, V_{IH} = V_{CC}, V_{IL} = 0
Standby current		I_{SB}	—	0.3	2	mA	$\overline{CS}$ = V_{IH}
		I_{SB1}	—	0.2	100	µA	Vin ≥ 0 V, $\overline{CS}$ ≥ V_{CC} – 0.2 V
		I_{SB1}	—	0.2*[2]	50*[2]	µA	
		I_{SB1}	—	0.2*[3]	10*[3]	µA	
Output low voltage		V_{OL}	—	—	0.4	V	I_{OL} = 2.1 mA
Output high voltage		V_{OH}	2.4	—	—	V	I_{OH} = –1.0 mA

Notes: 1. Typical values are at V_{CC} = 5.0 V, Ta = +25°C and not guaranteed.
2. This characteristic is guaranteed only for L-SL version.
3. This characteristic is guaranteed only for L-UL version.

Capacitance (Ta = 25°C, f = 1.0 MHz)

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Input capacitance*[1]	C_{in}	—	—	8	pF	Vin = 0 V
Input/output capacitance*[1]	$C_{I/O}$	—	—	10	pF	$V_{I/O}$ = 0 V

Note: 1. This parameter is sampled and not 100% tested.

AC Characteristics (Ta = 0 to +70°C, V_{CC} = 5.0 V ± 10%)

Test Conditions

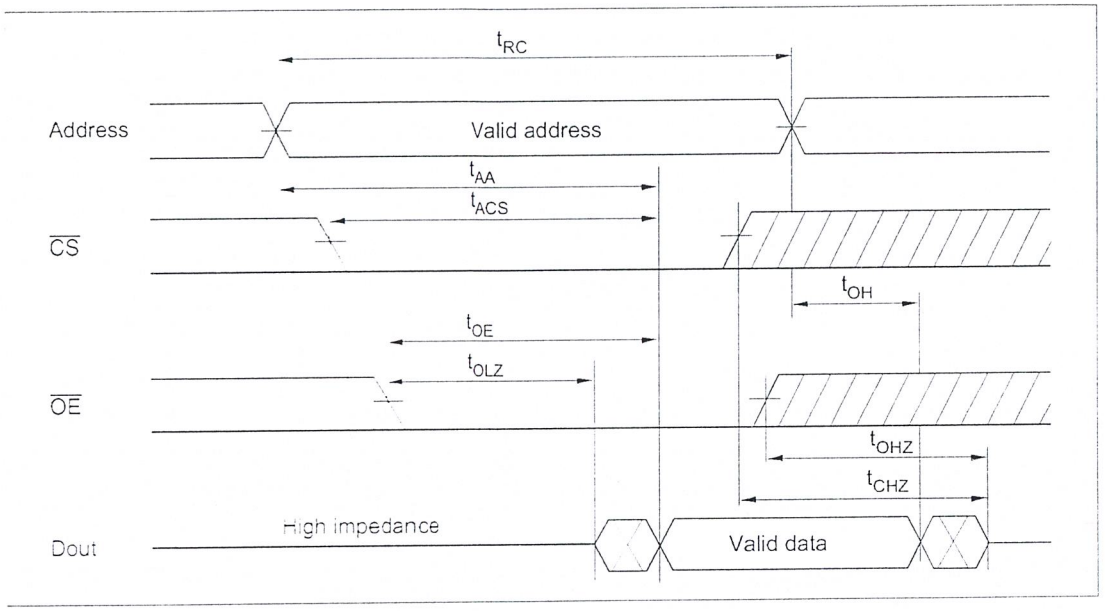
- Input pulse levels: 0.8 V to 2.4 V
- Input rise and fall time: 5 ns
- Input and output timing reference levels: 1.5 V
- Output load: 1 TTL Gate + C_L (50 pF) (HM62256B-5)
  1 TTL Gate + C_L (100 pF) (HM62256B-7/8)
  (Including scope & jig)

Read Cycle

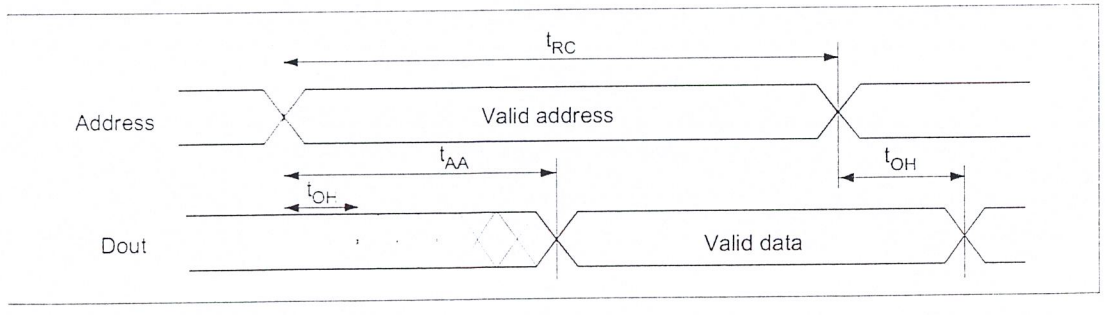
		HM62256B							
		-5		-7		-8			
Parameter	Symbol	Min	Max	Min	Max	Min	Max	Unit	Notes
Read cycle time	t_{RC}	55	—	70	—	85	—	ns	
Address access time	t_{AA}	—	55	—	70	—	85	ns	
Chip select to access time	t_{ACS}	—	55	—	70	—	85	ns	
Output enable to output valid	t_{OE}	—	35	—	40	—	45	ns	
Chip select to output in low-Z	t_{CLZ}	5	—	10	—	10	—	ns	2
Output enable to output in low-Z	t_{OLZ}	5	—	5	—	5	—	ns	2
Chip deselect to output in high-Z	t_{CHZ}	0	20	0	25	0	30	ns	1, 2
Output disable to output in high-Z	t_{OHZ}	0	20	0	25	0	30	ns	1, 2
Output hold from address change	t_{OH}	5	—	5	—	5	—	ns	

Timing Waveform

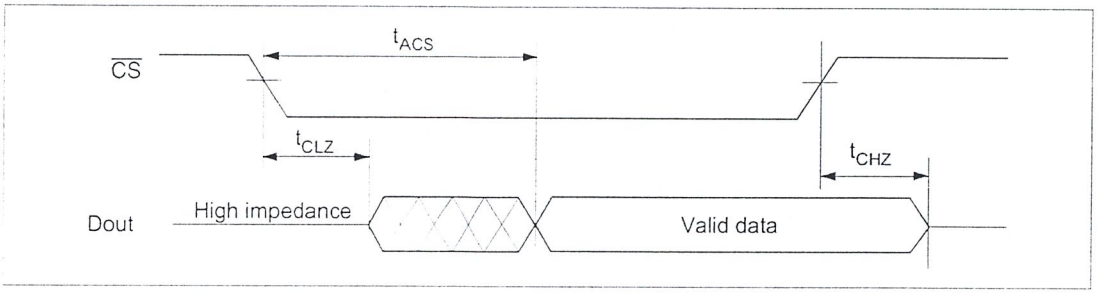
Read Timing Waveform (1) ($\overline{WE}$ = V_{IH})



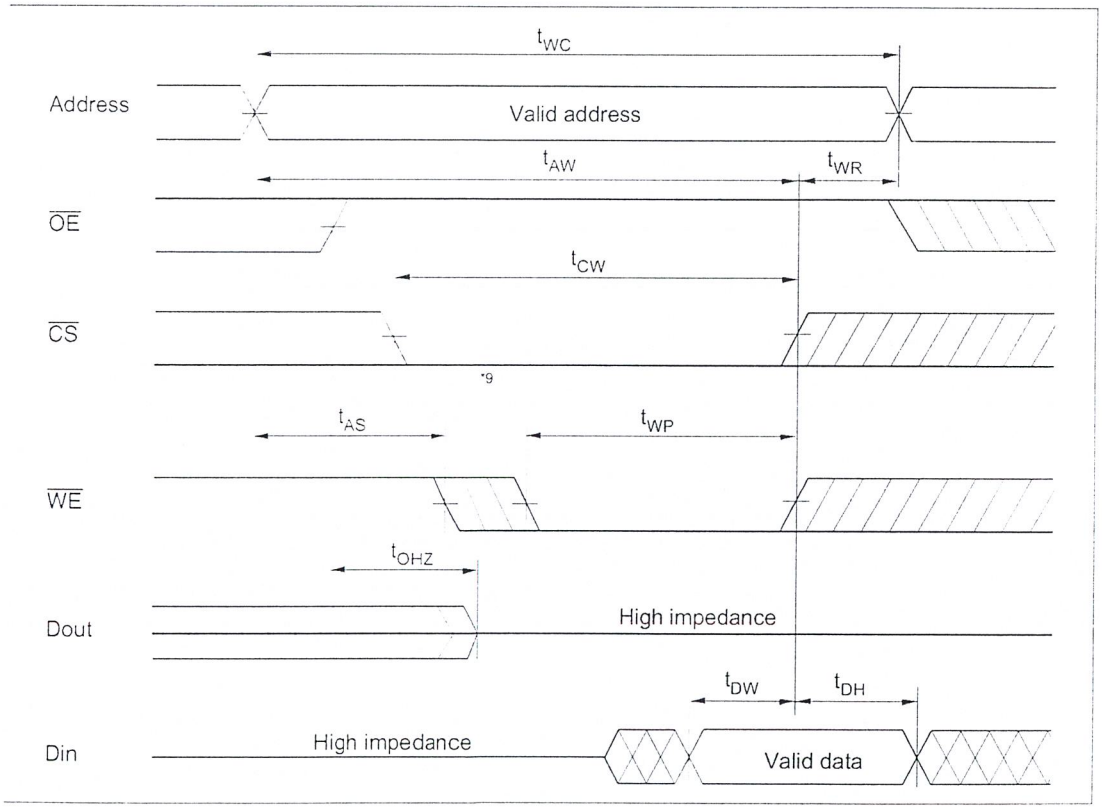
Read Timing Waveform (2) ($\overline{WE}$ = V_{IH}, $\overline{CS}$ = V_{IL}, $\overline{OE}$ = V_{IL})



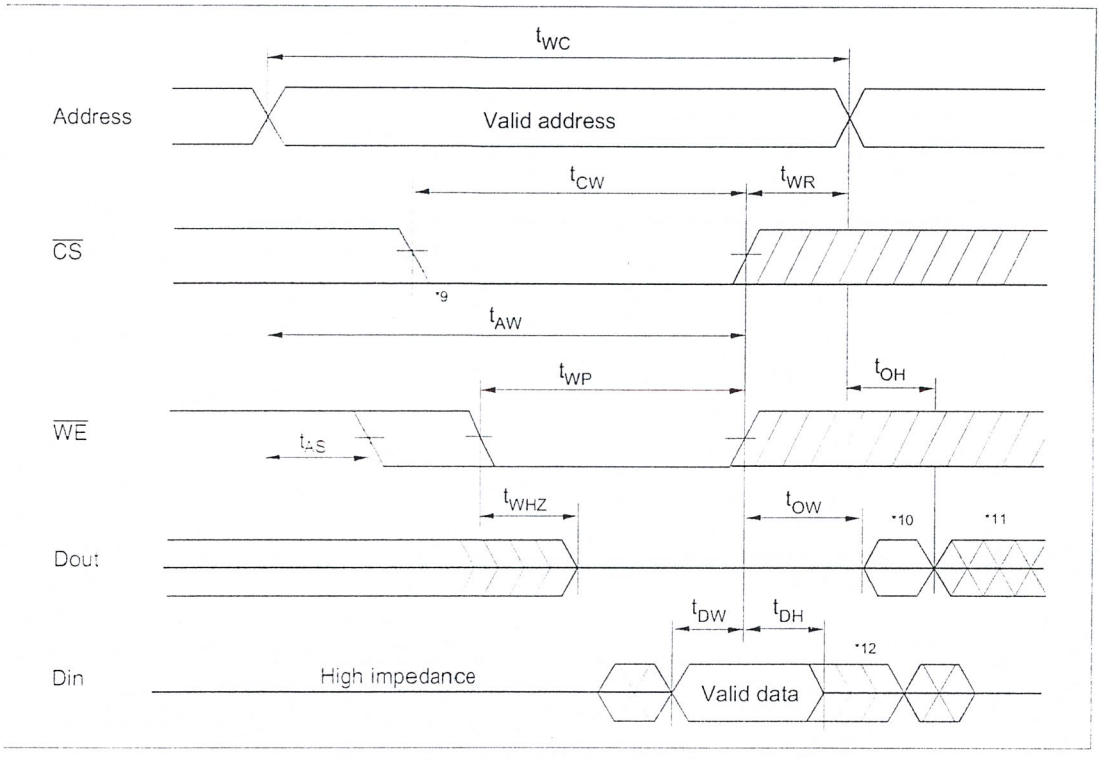
Read Timing Waveform (3) ($\overline{WE}$ = V_{IH}, $\overline{OE}$ = V_{IL})*3



Write Timing Waveform (1) ($\overline{OE}$ Clock)



Write Timing Waveform (2) ($\overline{OE}$ Low Fixed)

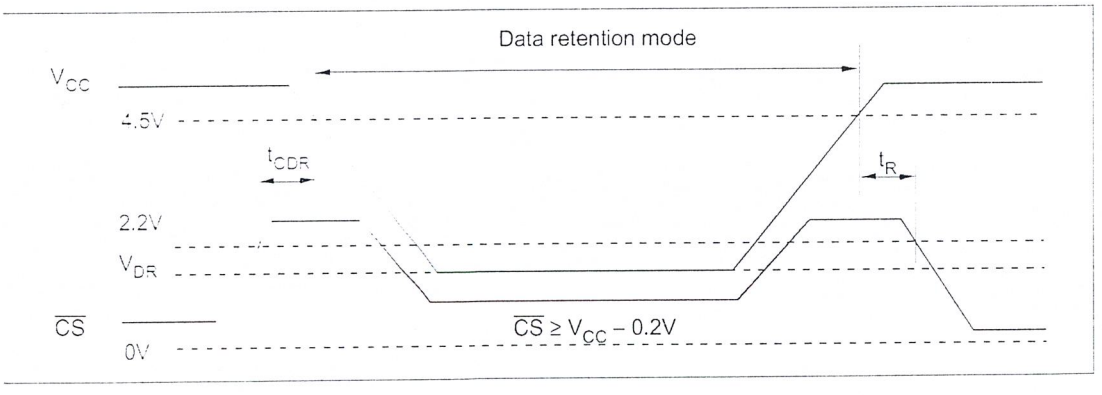


Low V_{CC} Data Retention Characteristics (Ta = 0 to 70°C)

Parameter	Symbol	Min	Typ*[1]	Max	Unit	Test conditions*[6]
V_{CC} for data retention	V_{DR}	2.0	—	5.5	V	$\overline{CS} \geq V_{CC} - 0.2$ V, Vin ≥ 0V
Data retention current	I_{CCDR}	—	0.05	30*[2]	µA	V_{CC} = 3.0 V, Vin ≥ 0V $\overline{CS} \geq V_{CC} - 0.2$ V
	I_{CCDR}	—	0.05	10*[3]	µA	
	I_{CCDR}	—	0.05	3*[4]	µA	
Chip deselect to data retention time	t_{CDR}	0	—	—	ns	See retention Waveform
Operation recovery time	t_R	t_{RC}*[5]	—	—	ms	

Notes: 1. Typical values are at V_{CC} = 3.0 V, Ta = +25°C and not guaranteed.
2. 10 µA max. at Ta = 0 to +40°C.
3. This characteristic is guaranteed only for L-SL version, 3 µA max. at Ta = 0 to +40°C.
4. This characteristic is guaranteed only for L-UL version, 0.6 µA max. at Ta = 0 to +40°C.
5. t_{RC} = Read cycle time.
6. $\overline{CS}$ controls address buffer, $\overline{WE}$ buffer, $\overline{OE}$ buffer, and Din buffer. If $\overline{CS}$ controls data retention mode, Vin levels (address, $\overline{WE}$, $\overline{OE}$, I/O) can be in the high impedance state.

Low V_{CC} Data Retention Timing Waveform



October 1999

National Semiconductor

ADC0808/ADC0809
8-Bit µP Compatible A/D Converters with 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

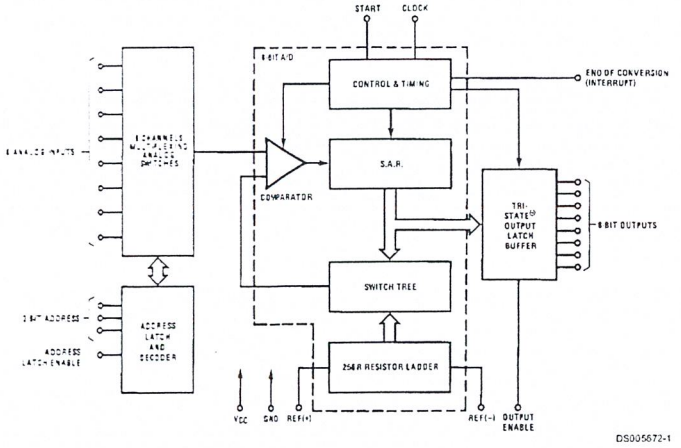
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 µs

Block Diagram

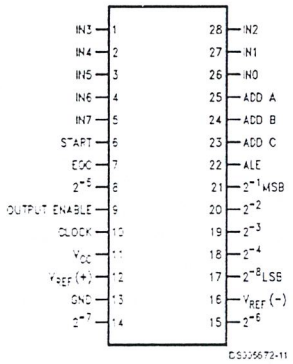


See Ordering Information

TRI-STATE® is a registered trademark of National Semiconductor Corp.

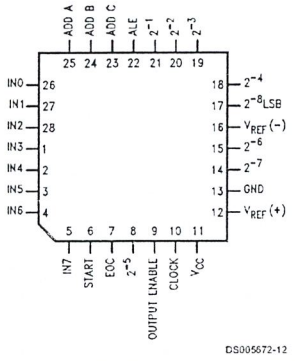
Connection Diagrams

Dual-In-Line Package



Order Number ADC0808CCN or ADC0809CCN
See NS Package J28A or N28A

Molded Chip Carrier Package



Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C			−55°C to +125°C
Error	±½ LSB Unadjusted	ADC0808CCN	ADC0808CCV	ADC0808CCJ	ADC0808CJ
	±1 LSB Unadjusted	ADC0809CCN	ADC0809CCV		
Package Outline		N28A Molded DIP	V28A Molded Chip Carrier	J28A Ceramic DIP	J28A Ceramic DIP

Absolute Maximum Ratings (Notes 2, 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage at Any Pin Except Control Inputs	−0.3V to (V_{CC}+0.3V)
Voltage at Control Inputs (START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)	−0.3V to +15V
Storage Temperature Range	−65˚C to +150˚C
Package Dissipation at T_A=25˚C	875 mW
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260˚C
Dual-In-Line Package (ceramic)	300˚C
Molded Chip Carrier Package	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 8)	400V

Operating Conditions (Notes 1, 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le T_{MAX}$
ADC0808CCN,ADC0809CCN	−40˚C≤T_A≤+85˚C
ADC0808CCV, ADC0809CCV	−40˚C ≤ T_A ≤ +85˚C
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.0 V_{DC}

Electrical Characteristics

Converter Specifications: V_{CC}=5 V_{DC}=$V_{REF(+)}$, $V_{REF(-)}$=GND, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK}=640 kHz unless otherwise stated.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	ADC0808					
	Total Unadjusted Error	25˚C			±½	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±¾	LSB
	ADC0809					
	Total Unadjusted Error	0˚C to 70˚C			±1	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±1¼	LSB
	Input Resistance	From Ref(+) to Ref(−)	1.0	2.5		kΩ
	Analog Input Voltage Range	(Note 4) V(+) or V(−)	GND−0.10		V_{CC}+0.10	V_{DC}
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	V_{CC}+0.1	V
$\frac{V_{REF(+)} - V_{REF(-)}}{2}$	Voltage, Center of Ladder		V_{CC}/2-0.1	V_{CC}/2	V_{CC}/2+0.1	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V
I_{IN}	Comparator Input Current	f_c=640 kHz, (Note 6)	−2	±0.5	2	µA

Electrical Characteristics

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, 4.75≤V_{CC}≤5.25V, −40˚C≤T_A≤+85˚C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
ANALOG MULTIPLEXER						
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=5V,				
		T_A=25˚C		10	200	nA
		T_{MIN} to T_{MAX}			1.0	µA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=0,				
		T_A=25˚C	−200	−10		nA
		T_{MIN} to T_{MAX}	−1.0			µA
CONTROL INPUTS						
$V_{IN(1)}$	Logical "1" Input Voltage		V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage				1.5	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN}=15V			1.0	µA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN}=0	−1.0			µA
I_{CC}	Supply Current	f_{CLK}=640 kHz		0.3	3.0	mA

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40^{\circ}C \le T_A \le +85^{\circ}C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 4.75V I_{OUT} = −360µA I_{OUT} = −10µA		**2.4** **4.5**		V(min) V(min)
$V_{OUT(0)}$	Logical "0" Output Voltage	I_O=1.6 mA			0.45	V
$V_{OUT(0)}$	Logical "0" Output Voltage EOC	I_O=1.2 mA			0.45	V
I_{OUT}	TRI-STATE Output Current	V_O=5V V_O=0	 −3		3	µA µA

Electrical Characteristics

Timing Specifications $V_{CC}=V_{REF(+)}=5V$, $V_{REF(-)}=GND$, $t_r=t_f=20$ ns and $T_A=25^{\circ}C$ unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	(*Figure 5*)		100	200	ns
t_{WALE}	Minimum ALE Pulse Width	(*Figure 5*)		100	200	ns
t_s	Minimum Address Set-Up Time	(*Figure 5*)		25	50	ns
t_H	Minimum Address Hold Time	(*Figure 5*)		25	50	ns
t_D	Analog MUX Delay Time From ALE	R_S=0Ω (*Figure 5*)		1	2.5	µs
t_{1H}, t_{H1}	OE Control to Q Logic State	C_L=50 pF, R_L=10k (*Figure 8*)		125	250	ns
t_{0H}, t_{H0}	OE Control to Hi-Z	C_L=10 pF, R_L=10k (*Figure 8*)		125	250	ns
t_C	Conversion Time	f_c=640 kHz, (*Figure 5*) (Note 7)	90	100	116	µs
f_c	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	(*Figure 5*)	0		8+2 µS	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE Output Capacitance	At TRI-STATE Outputs		10	15	pF

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: A zener diode exists internally from V_{CC} to GND and has a typical breakdown voltage of 7 V_{DC}.

Note 4: Two protection diodes are tied to each analog input which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. The spec allows 100 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 100 mV, the output code will be correct. To achieve an absolute $0V_{DC}$ to $5V_{DC}$ input voltage range will therefore require a minimum supply voltage of 4.900 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors. See *Figure 3*. None of these A/Ds requires a zero or full-scale adjust. However, if an all zero code is desired for an analog input other than 0.0V, or if a narrow full-scale span exists (for example: 0.5V to 4.5V full-scale) the reference voltages can be adjusted to achieve this. See *Figure 13*.

Note 6: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence (*Figure 5*). See paragraph 4.0.

Note 7: The outputs of the data register are updated one clock cycle before the rising edge of EOC.

Note 8: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

National Semiconductor

November 2000

LM135/LM235/LM335, LM135A/LM235A/LM335A Precision Temperature Sensors

General Description

The LM135 series are precision, easily-calibrated, integrated circuit temperature sensors. Operating as a 2-terminal zener, the LM135 has a breakdown voltage directly proportional to absolute temperature at +10 mV/°K. With less than 1Ω dynamic impedance the device operates over a current range of 400 µA to 5 mA with virtually no change in performance. When calibrated at 25°C the LM135 has typically less than 1°C error over a 100°C temperature range. Unlike other sensors the LM135 has a linear output.

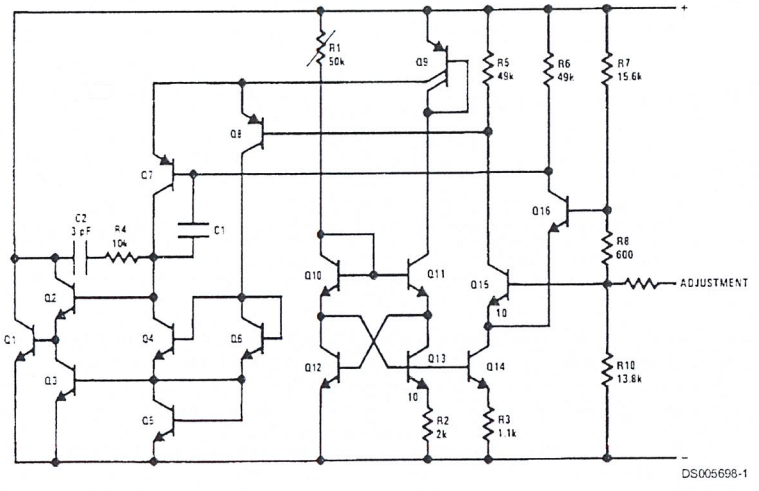
Applications for the LM135 include almost any type of temperature sensing over a −55°C to +150°C temperature range. The low impedance and linear output make interfacing to readout or control circuitry especially easy.

The LM135 operates over a −55°C to +150°C temperature range while the LM235 operates over a −40°C to +125°C temperature range. The LM335 operates from −40°C to +100°C. The LM135/LM235/LM335 are available packaged in hermetic TO-46 transistor packages while the LM335 is also available in plastic TO-92 packages.

Features

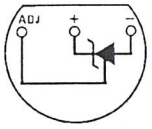
- Directly calibrated in °Kelvin
- 1°C initial accuracy available
- Operates from 400 µA to 5 mA
- Less than 1Ω dynamic impedance
- Easily calibrated
- Wide operating temperature range
- 200°C overrange
- Low cost

Schematic Diagram



Connection Diagrams

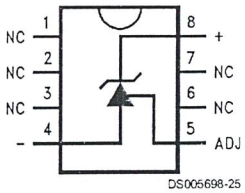
TO-92
Plastic Package



DS005698-8

Bottom View
Order Number LM335Z
or LM335AZ
See NS Package
Number Z03A

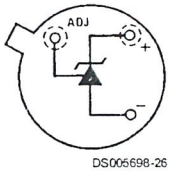
SO-8
Surface Mount Package



DS005698-25

Order Number LM335M
See NS Package
Number M08A

TO-46
Metal Can Package*



DS005698-26

*Case is connected to negative pin

Bottom View
Order Number LM135H,
LM135H-MIL, LM235H,
LM335H, LM135AH,
LM235AH or LM335AH
See NS Package
Number H03H

Absolute Maximum Ratings (Note 4)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Reverse Current	15 mA
Forward Current	10 mA
Storage Temperature	
TO-46 Package	−60°C to +180°C
TO-92 Package	−60°C to +150°C
SO-8 Package	−65°C to +150°C

Specified Operating Temp. Range

	Continuous	Intermittent (Note 2)
LM135, LM135A	−55°C to +150°C	150°C to 200°C
LM235, LM235A	−40°C to +125°C	125°C to 150°C
LM335, LM335A	−40°C to +100°C	100°C to 125°C

Lead Temp. (Soldering, 10 seconds)	
TO-92 Package:	260°C
TO-46 Package:	300°C
SO-8 Package:	300°C
Vapor Phase (60 seconds):	215°C
Infrared (15 seconds):	220°C

Temperature Accuracy (Note 1)

LM135/LM235, LM135A/LM235A

Parameter	Conditions	LM135A/LM235A			LM135/LM235			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage	T_C = 25°C, I_R = 1 mA	2.97	2.98	2.99	2.95	2.98	3.01	V
Uncalibrated Temperature Error	T_C = 25°C, I_R = 1 mA		0.5	1		1	3	°C
Uncalibrated Temperature Error	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		1.3	2.7		2	5	°C
Temperature Error with 25°C Calibration	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		0.3	1		0.5	1.5	°C
Calibrated Error at Extended Temperatures	$T_C = T_{MAX}$ (Intermittent)		2			2		°C
Non-Linearity	I_R = 1 mA		0.3	0.5		0.3	1	°C

Temperature Accuracy (Note 1)

LM335, LM335A

Parameter	Conditions	LM335A			LM335			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage	T_C = 25°C, I_R = 1 mA	2.95	2.98	3.01	2.92	2.98	3.04	V
Uncalibrated Temperature Error	T_C = 25°C, I_R = 1 mA		1	3		2	6	°C
Uncalibrated Temperature Error	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		2	5		4	9	°C
Temperature Error with 25°C Calibration	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		0.5	1		1	2	°C
Calibrated Error at Extended Temperatures	$T_C = T_{MAX}$ (Intermittent)		2			2		°C
Non-Linearity	I_R = 1 mA		0.3	1.5		0.3	1.5	°C

Electrical Characteristics (Note 1)

Parameter	Conditions	LM135/LM235 LM135A/LM235A			LM335 LM335A			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage Change with Current	400 μA≤I_R≤5 mA At Constant Temperature		2.5	10		3	14	mV
Dynamic Impedance	I_R=1 mA		0.5			0.6		Ω
Output Voltage Temperature Coefficient			+10			+10		mV/°C
Time Constant	Still Air		80			80		sec
	100 ft/Min Air		10			10		sec
	Stirred Oil		1			1		sec
Time Stability	T_C=125°C		0.2			0.2		°C/khr

Electrical Characteristics (Note 1) (Continued)

Note 1: Accuracy measurements are made in a well-stirred oil bath. For other conditions, self heating must be considered.

Note 2: Continuous operation at these temperatures for 10,000 hours for H package and 5,000 hours for Z package may decrease life expectancy of the device.

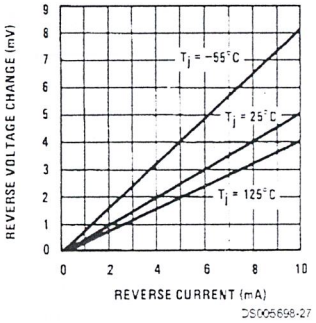
Note 3:

Thermal Resistance	TO-92	TO-46	SO-8
θ_{JA} (junction to ambient)	202°C/W	400°C/W	165°C/W
θ_{JC} (junction to case)	170°C/W	N/A	N/A

Note 4: Refer to RETS135H for military specifications.

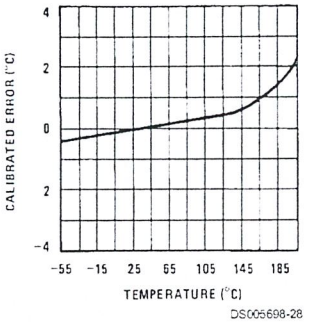
Typical Performance Characteristics

Reverse Voltage Change



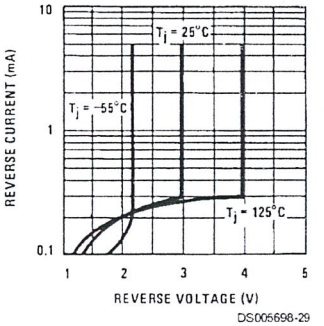
DS005698-27

Calibrated Error



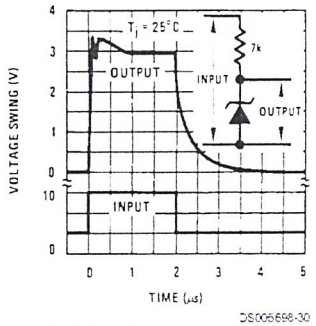
DS005698-28

Reverse Characteristics



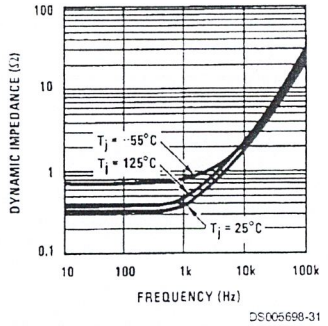
DS005698-29

Response Time



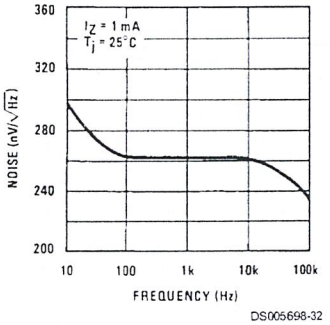
DS005698-30

Dynamic Impedance



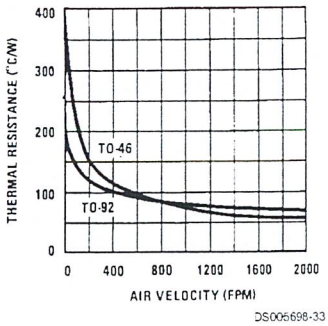
DS005698-31

Noise Voltage



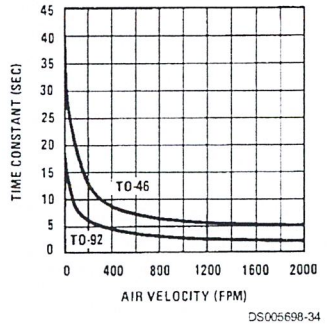
DS005698-32

Thermal Resistance Junction to Air



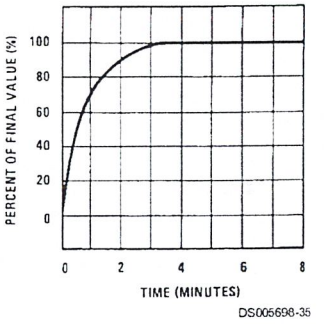
DS005698-33

Thermal Time Constant



DS005698-34

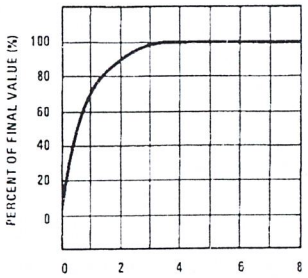
Thermal Response in Still Air



DS005698-35

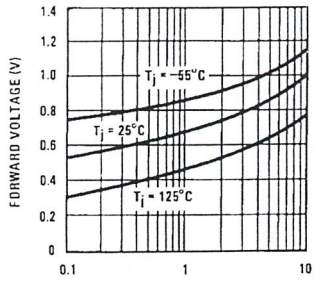
Typical Performance Characteristics (Continued)

Thermal Response in Stirred Oil Bath



DS005698-36

Forward Characteristics



DS005698-37

Application Hints

CALIBRATING THE LM135

Included on the LM135 chip is an easy method of calibrating the device for higher accuracies. A pot connected across the LM135 with the arm tied to the adjustment terminal allows a 1-point calibration of the sensor that corrects for inaccuracy over the full temperature range.

This single point calibration works because the output of the LM135 is proportional to absolute temperature with the extrapolated output of sensor going to 0V output at 0°K (−273.15°C). Errors in output voltage versus temperature are only slope (or scale factor) errors so a slope calibration at one temperature corrects at all temperatures.

The output of the device (calibrated or uncalibrated) can be expressed as:

$$V_{OUT_T} = V_{OUT_{T_o}} \times \frac{T}{T_o}$$

where T is the unknown temperature and T_o is a reference temperature, both expressed in degrees Kelvin. By calibrating the output to read correctly at one temperature the output at all temperatures is correct. Nominally the output is calibrated at 10 mV/°K.

To insure good sensing accuracy several precautions must be taken. Like any temperature sensing device, self heating can reduce accuracy. The LM135 should be operated at the lowest current suitable for the application. Sufficient current, of course, must be available to drive both the sensor and the calibration pot at the maximum operating temperature as well as any external loads.

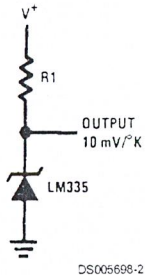
If the sensor is used in an ambient where the thermal resistance is constant, self heating errors can be calibrated out. This is possible if the device is run with a temperature stable current. Heating will then be proportional to zener voltage and therefore temperature. This makes the self heating error proportional to absolute temperature the same as scale factor errors.

WATERPROOFING SENSORS

Meltable inner core heat shrinkable tubing such as manufactured by Raychem can be used to make low-cost waterproof sensors. The LM335 is inserted into the tubing about ½" from the end and the tubing heated above the melting point of the core. The unfilled ½" end melts and provides a seal over the device.

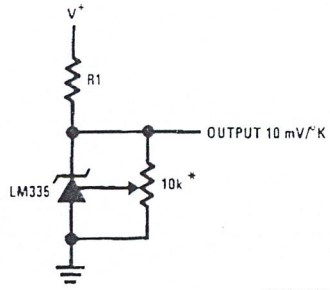
Typical Applications

Basic Temperature Sensor



DS005698-2

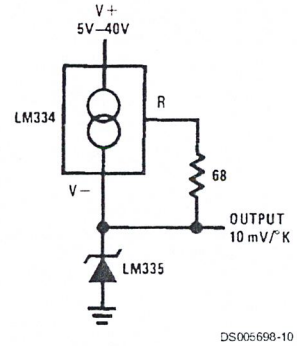
Calibrated Sensor



DS005698-9

*Calibrate for 2.982V at 25°C

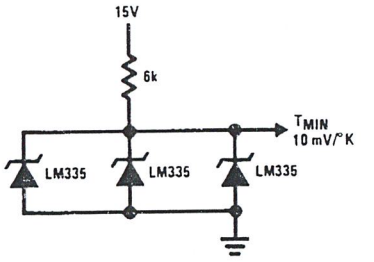
Wide Operating Supply



DS005698-10

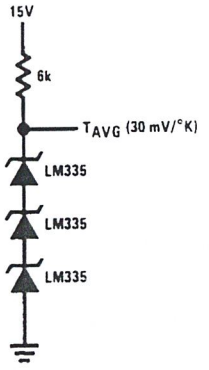
Typical Applications (Continued)

Minimum Temperature Sensing



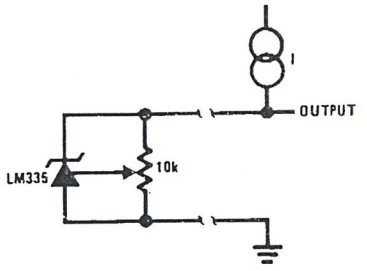
DS005698-4

Average Temperature Sensing



DS005698-18

Remote Temperature Sensing



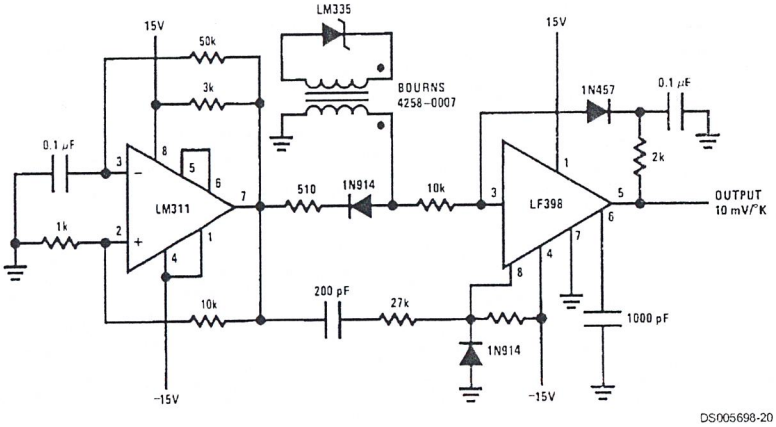
DS005698-19

Wire length for 1°C error due to wire drop

	I_R = 1 mA	I_R = 0.5 mA*
AWG	FEET	FEET
14	4000	8000
16	2500	5000
18	1600	3200
20	1000	2000
22	625	1250
24	400	800

*For I_R = 0.5 mA, the trim pot must be deleted.

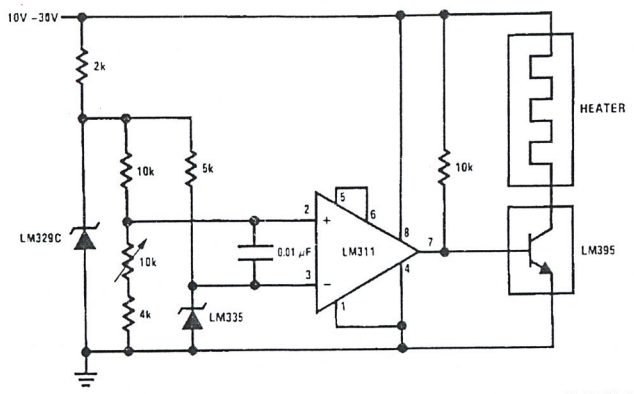
Isolated Temperature Sensor



DS005698-20

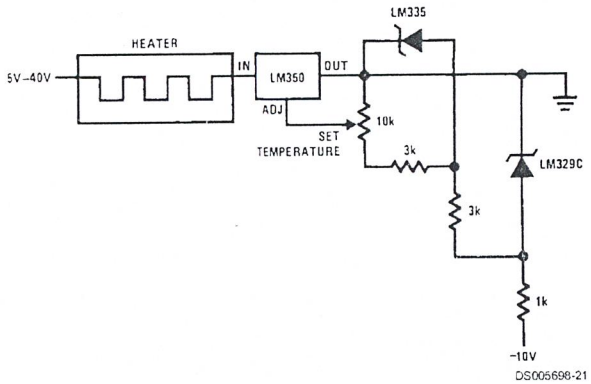
Typical Applications (Continued)

Simple Temperature Controller



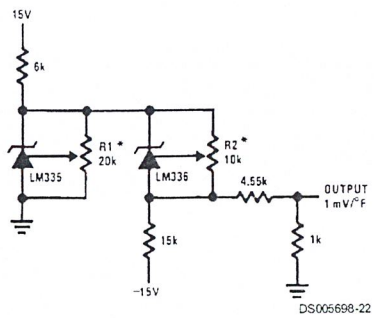
DS005698-5

Simple Temperature Control



DS005698-21

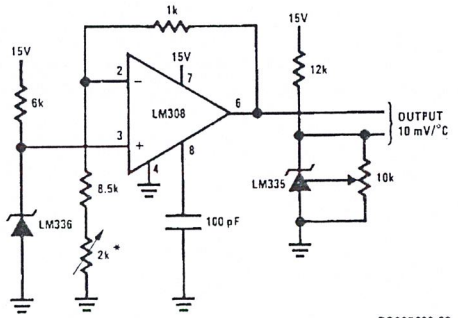
Ground Referred Fahrenheit Thermometer



DS005698-22

*Adjust R2 for 2.554V across LM336.
Adjust R1 for correct output.

Centigrade Thermometer



DS005698-23

*Adjust for 2.7315V at output of LM308

เอกสารอ้างอิง

1. ชัยวัฒน์ ลิ้มพรจิตรวิไล และ วรพจน์ กรแก้ววัฒนกุล " เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51"
2. พ.อ. เจนวิทย์ เหลืองอร่าม และ ปิยวิทย์ เหลืองอร่าม " การเขียนโปรแกรมด้วย C / C++ "
3. LAB EL327, "Data Acquisittion and Conversion" Electronic laboratory, Dept. of Electronic
4. "ET – AFP V 1.0 Flash Programmer " User's Manual , ETT Co.,Ltd.
5. "MCS – 51 Microcontroller Training Board" User's Manual. Innovative Experiment Co.,Ltd.
6. " LM335 " National Semiconductor Operation, 2000, Page 1 – 7 .
7. " ADC 0808 " National Semiconductor Application Note 247, Page 626 – 630 .
8. " DS1307 " Dallas Semiconductor, Page 1 – 6.